File Number S360-36
Form Y24-5017-2

**Program Logic**

# IBM System/360
# Disk Operating System
# System Control
# Program Logic Manual

**Program Number 360N-CL-453, Version 2**

This publication describes the internal logic of the IBM System/360 Disk Operating System, System Control Program. It is intended for use by persons involved in program maintenance and by system programmers who are altering the program design. Program logic information is not necessary for the operation of the System Control Program; therefore, distribution of this publication is limited to those with maintenance and alteration requirements. It is designed to be used as a supplement to the program listing.

Effective use of this manual requires an understanding of IBM System/360 operation and of IBM System/360 Disk Operating System control and service programs, macro instructions, and operating procedures. Reference Publications for this information are listed in the Preface of this manual.

**Restricted Distribution**

Major Revision, July 1967

This edition, Y24-5017-2, is a major revision of, and obsoletes Form Y24-5017-1 and its Technical Newsletters Y24-5058, Y24-5053 and Y24-5066. Changes are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (•) next to a figure title or page number indicates that the entire figure or page should be reviewed.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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

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

This Program Logic Manual (PLM) is a guide to the IBM System/360 Disk Operating System, System Control Programs, Linkage Editor, and Librarian; it supplements the program listings by providing descriptive text and flowcharts.

## PREREQUISITE AND RELATED LITERATURE

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

- IBM System/360 Principles of Operation, Form A22-6821
- IBM System/360 Disk Operating System: System Control and System Service Programs, Form C24-5036
- IBM System/360 Disk Operating System: Supervisor and Input/Output Macros, Form C24-5037
- IBM System/360 Disk Operating System: System Generation and Maintenance, Form C24-5033
- IBM System/360 Disk Operating System: Operating Guide, Form C24-5022

Closely related publications are:

- IBM System/360 Disk Operating System: Data Management Concepts, Form C24-3427
- IBM System/360 Disk and Tape Operating Systems: Assembler Specifications, Form C24-3414.

Titles and Abstracts of other related publications are listed in the IBM System/360 Bibliography, Form A22-6822.

## ORGANIZATION AND USE OF THIS PUBLICATION

This manual presents the components of the DOS System Control Program in a logical manner that emphasizes:

- Interrelationship of the components in an operating system environment.
- Organization, function, and format of system residence.
- Generation and function of the supervisor control program, including physical IOCS.
- Function of the system control programs, IPL and Job Control.
- Function and interrelationship of the Linkage Editor program and the Librarian programs.

The first three sections provide background material essential for an understanding of the individual components of the DOS System Control Program.

This manual is organized to provide quick access to the detailed information on the internal logic of all components of the DOS System Control Program. Cross referencing is provided as follows:

1. The label list, Appendix A, provides a cross reference between the listing and the detail (routine) level flowcharts.
2. Error messages, Appendix F, are cross referenced to the program phase and the detail (routine) level flowchart.
3. Program level flowcharts refer to the detail (routine) level flowcharts.
4. Detail (routine) level flowcharts, where applicable, refer to the program level flowcharts.

The organization of this manual is adaptable to the various ways in which it will be used:

1. Sections 1 through 3 may be read as an introduction to the DOS System Control Program.
2. Sections 4 through 8 may be read, either selectively or completely, for program level concepts.
3. The reader may choose his own point of entry into the manual based on his individual qualifications. Figure 1 is an example of how the various parts of the manual may be used in satisfying a particular situation.

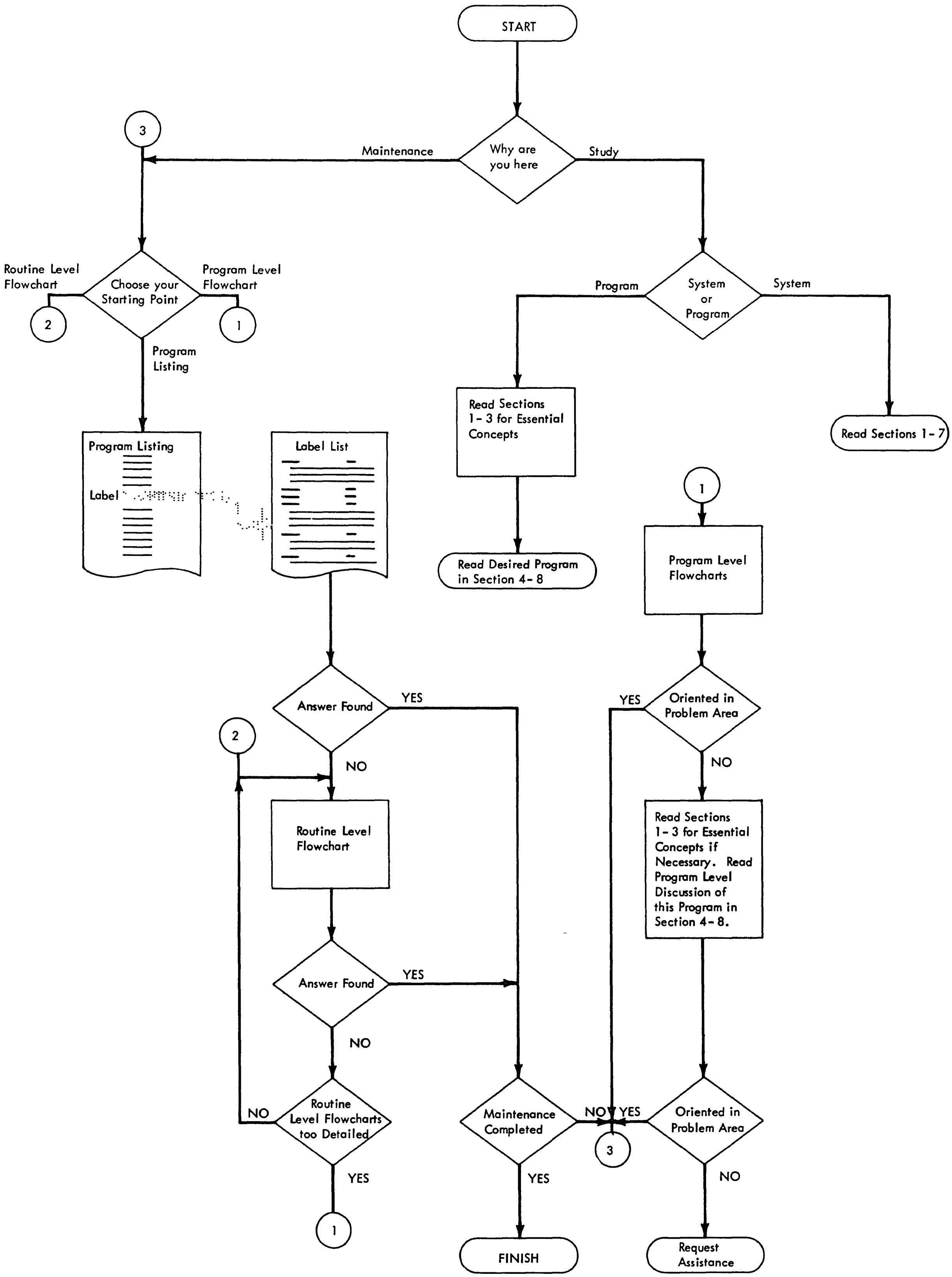



Figure 1. Example of PLM Usage

STRUCTURE

This manual contains eight sections and seven appendixes. The function of each section and appendix is presented below.

## General Information Sections

Section 1: Provides an introduction to the IBM System/360 Disk Operating System, System Control Programs.

Section 2: Provides information about the organization of system residence (SYSRES).

Section 3: Provides information about supervisor generation. This section includes a discussion of:

1. Supervisor generation macros.
2. Common information that is referenced from other sections, such as:
   a. Supervisor storage organization (MAP)
   b. Communications region
   c. Device dependent codes
   d. I/O Tables
      - LUB Table
      - PUB Table
      - TEB Table
      - JIB Table
      - Number in class list (NICL)
      - First in class list (FICL)
      - First on channel list (FOCL)
      - First available pointer (FAVP)
   e. Program Information Block (PIB)
   f. Disk Information Block (DIB)

Note: The background information contained in Sections 1 through 3 is essential for an understanding of the individual components presented in subsequent sections.

## Program Information Sections

Sections 4 through 8 contain program level discussions of the system control programs. These discussions contain the following information when applicable:

1. Program introduction
2. Interface with other programs
3. Program flow (phase to phase)
4. I/O flow
5. Storage maps
6. Key concepts

Note: The program level flowchart for a specific program is located immediately following the program level information for that program. In some cases, it was necessary to group the program level flowcharts immediately following a group of programs. This is particularly true in the case of the B-transients and the A-transients in Section 4.

Section 4: Provides information about the following programs:

1. Initial Program Load (IPL)
2. Job Control ($JOBCTLA)
3. Supervisor Control ($$A$SUP1)
4. A-transients
5. B-transients
   a. Foreground Initiator
   b. Nonresident Attention Routine
   c. Program Terminator

Section 5: Provides information about the Linkage Editor program ($LNKEDT).

Section 6: Provides information about the following Librarian Maintenance programs:

1. Common Library Maintenance program (MAINT)
2. Automatic Condense Limits program (MAINTCL)
3. Core Image Library Maintenance program (MAINTC2)
4. Relocatable Library Maintenance program (MAINTR2)
5. Source Statement Library Maintenance program (MAINTS2)
6. Update Transient, Library-Routine, and Foreground Directories program ($MAINEOJ)
7. Library Condense program (MAINTCN)
8. System Reallocation program (MAINTA)

Section 7: Provides information about the Librarian Organization program CORGZ (Copy System program).

Section 8: Provides information about the following Librarian Service programs:

1. Directory Service program (DSERV).

2. Relocatable Library Service program (RSERV).

3. Source Statement Library Service program (SSERV).

## Appendixes

Appendix A: Contains the label list for all programs in this manual. The structure of this appendix is as follows:

1. Labels are sequenced alphamerically within a phase.

2. Phases are sequenced alphamerically within a program.

3. Programs are ordered to reflect the structure of this manual (Sections 4 through 8, IPL, Job Control,...,DSERV, RSERV, SSERV).

A label may be cross referenced to the detail (routine) level flowchart that contains the label or it may contain the notation "Listing Only." The latter notation designates that this label does not appear in any flowchart. However, the comment following this label presents some information that is not readily clear in the listing.

Appendix B: Contains a list of flowchart abbreviations that have been established as standard within this manual.

Appendix C: Contains an explanation of the flowchart symbols used in this manual.

Appendix D: Contains sample LISTIO printouts.

Appendix E: Contains a detailed description of ESD processing in the Linkage Editor program. It is to be used as a supplement for the Linkage editor charts RA-RJ in Appendix H.

Appendix F: Contains an error message cross reference that identifies the program phase(s) and the detail (routine) level chart(s) associated with a specific error message.

Appendix G: Definition of PIK (Program Interrupt Key), LTK (Logical Transient Key), RIK (Requestor I/O Key), and FIK (Fetch I/O Key).

Appendix H: Contains the detail (routine) level flowcharts for all programs in this manual. Flowchart titles, where applicable, refer to the program level chart associated with the detail chart. An example of this upward cross referencing follows:

Chart SH. Map Processor (Refer to Linkage Editor - Chart 36)

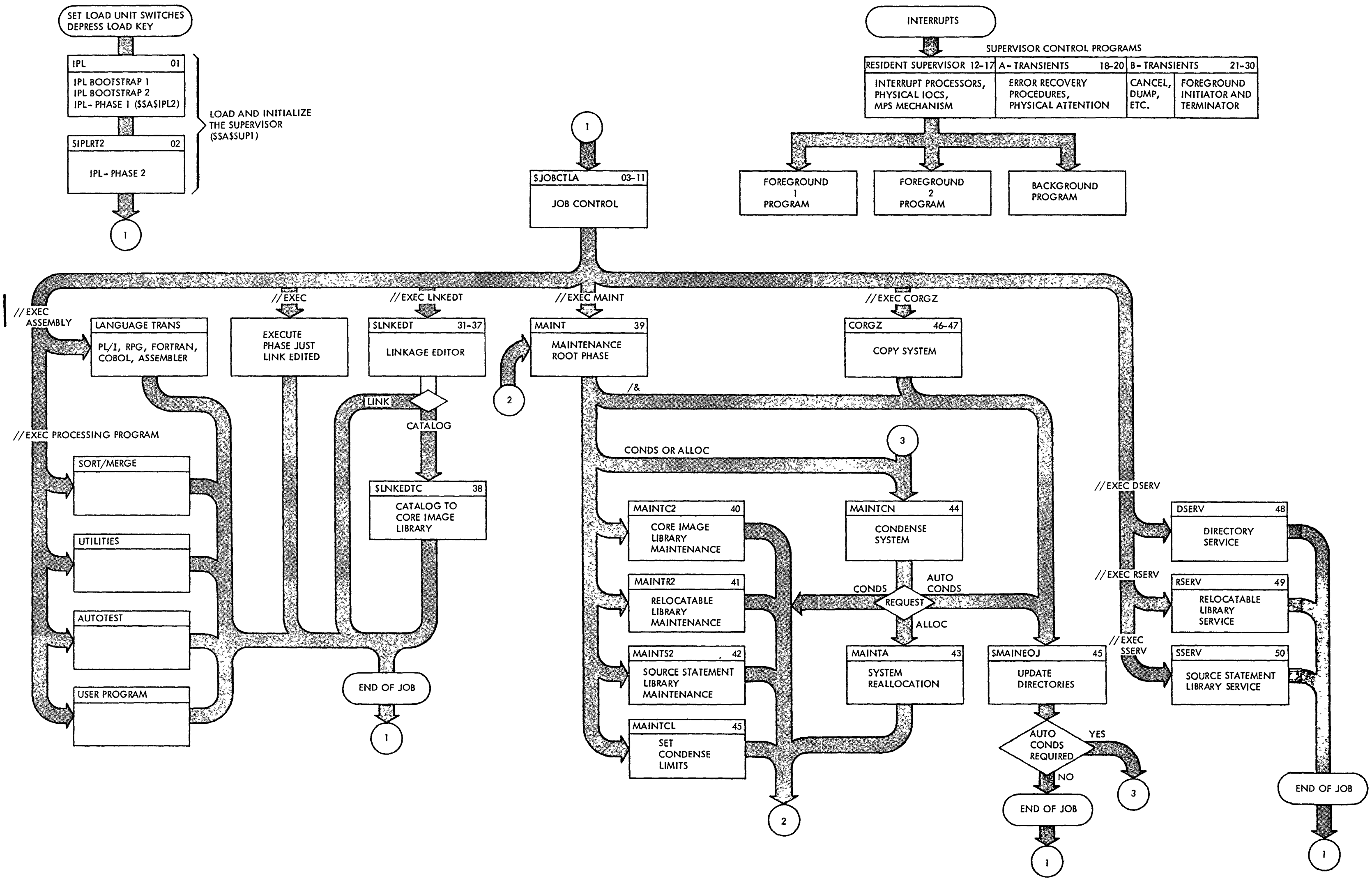
Cross reference from the program level chart to a detail level chart is provided in the program level chart. Each block in a program level chart contains a detail chart designation in the block title line. An example of this downward cross referencing follows:

```
  STMTIN
┌──────────────────────────────────────────┐
│ CONTROL STATEMENT READ      Chart BB     │
├──────────────────────────────────────────┤
│                                          │
│ Where:  BB represents the detail level   │
│ flowchart of this routine.               │
└──────────────────────────────────────────┘
```

It is recommended that all the flowcharts in Appendix H be removed and placed in a separate binder. This procedure, if followed, provides the reader with access to the flowcharts and the rest of the manual with a minimum of page turning. It also divides the manual into two, easier to handle, parts.

CHARTS

Chart 00. System Program Flow



The resident version of the IBM System/360 Disk Operating System (DOS), System Control, Version 2, provides disk operating system capabilities for 16K and larger System/360 configurations. At least one IBM 2311 Disk Storage Drive is required.

Systems larger than 16K can benefit from this 16K package if they do not require the expanded functions of the larger disk operating system packages offered by IBM. The system is disk resident, using the IBM 2311 Disk Storage Drive for on-line storage of all programs. Depending on the requirements of the particular application, the system can be expanded to include all processing programs used to perform the various jobs of a particular installation, or it can be tailored to a minimum system to control a single program.

The operating system is composed of many components, which include: CPU, input/output channels, input/output control units, input/output devices, microprogramming, system control programs, support programs, user programs, user data files, Tele-processing capability, and multiple programming capability. Only the system control programs are within the scope of this publication. Of the system control programs, the supervisor and physical IOCS are specifically designed for a user's configuration by means of a one-time assembly (generation time). They require re-assembly only if the user's configuration changes.

The supervisor and physical IOCS provide the required interface between the program being executed and the other components of the operating system. The program currently being executed is identified to the operating system as the current program (definition used with this manual). The last program interrupted is identified as the problem program. The problem program or the current program can be, at any given time, either a system control program, a support program, or a user program.

## MULTIPROGRAMMING

For those systems with main storage equal to or in excess of 24K, disk operating system offers multiprogramming support. This support is referred to as fixed partitioned multiprogramming, because programs are assigned to fixed locations when they are cataloged to the system. A program occupies a contiguous area of storage. The amount of main storage allotted to programs to be executed may be determined when the system is generated, or the amount may be determined by the operator when the program is loaded into main storage for execution.

### Background vs Foreground Programs

There are two types of problem programs in multiprogramming: background and foreground. Background programs are initiated by job control from the batched-job input stream. Foreground programs are initiated by the operator from the printer-keyboard. Foreground programs do not execute from a stack (batch). When one program is completed, the operator must explicitly initiate the next program.

Background and foregound programs initiate and terminate completely independent of each other.

The system is capable of concurrently operating one background program and one or two foreground programs. Priority for CPU processing is controlled by the supervisor, with foreground programs having priority over background programs. All programs operate with interrupts enabled. When an interrupt occurs, the Supervisor gains control, processes the interrupt, and gives control to the highest priority program which is in a ready state.

Control is taken away from a high priority program when that program encounters a condition that prevents continuation of processing until a specified event has occurred. Control is taken away from a lower priority program at the completion of an event for which a higher priority program was waiting. When all programs in the system are simultaneously waiting (i.e., no program can process), the system is placed in the wait state enabled for interruptions.

Interruptions are received and processed by the Supervisor. When an interruption satisfies a program's wait condition, that program becomes active and competes with other programs for CPU processing time.

In addition to at least 24K positions of main storage, multiprogramming support requires the storage protection feature.

Note that programs produced by the FORTRAN and PL/I compilers may not be run as foreground programs, because object programs produced by these compilers use communication region data or system logical units pertinent only to background programs.

## TELECOMMUNICATIONS

Disk Operating System includes telecommunication capability that is defined as Basic Telecommunications Access Method (BTAM). A BTAM program may be run as either a foreground program or a background program. Normally it is run as a foreground one program so that it has the highest priority of any program being executed at a particular time. As with multiprogramming, BTAM requires a minimum of 24K positions of main storage.

## PURPOSE OF AN OPERATING SYSTEM

All System/360 programs have certain common required functions such as input/output operations, error detection and correction, operator communications, program loading, and five types of interrupt-handling capability. The Supervisor and physical IOCS programs relieve the user of performing these repetitious functions. His attention can be devoted solely to solving his problems.

The operating system provides maximum utilization of System/360 resources, that is, main storage, CPU time, channel time, input/output devices, program libraries, control files, and data files. It also provides maximum throughput (minimum lost time between jobs and minimum set-up time).

## CONFIGURATION

This section presents the minimum configuration requirements as well as the additional features and devices supported by the DOS System Control. Presentation is in the following order:

1. Minimum requirements
2. Additional features
3. I/O devices
4. System I/O devices
5. System I/O flow

## MINIMUM REQUIREMENTS

The minimum configuration required by the DOS System Control is:

1. 16K bytes of main storage (24K bytes are required for multiprogramming and BTAM).
2. Standard instruction set (language translators can require extended instruction sets).
3. One I/O channel, either multiplexor or selector. (Tele-processing requires a multiplexor channel and at least one selector channel.)
4. One card reader (IBM 1442, 2501, 2520, or 2540). See Note 1.
5. One card punch (IBM 1442, 2520, or 2540). See Note 1.
6. One printer (IBM 1403, 1404, or 1443). See Note 1.
7. One IBM 1052 Printer-Keyboard.
8. One IBM 2311 Disk Storage Drive.

Note 1: One 2400-series magnetic tape unit (7- or 9-track) can be substituted for this device. The data-convert feature is required if a 7-track tape unit is substituted for a card reader or a card punch. The data-convert feature is not required if a 7-track tape unit is substituted for a printer. MPS must have a reader or all foreground initiation commands must be entered via a 1052 device.

## ADDITIONAL FEATURES

Additional features supported by the DOS System Control are:

1. Timer feature.
2. Simultaneous read-while-write tape control (2404 or 2804).
3. Any channel configuration up to one multiplexor channel and six selector channels.
4. Tape switching unit (2816).

5. Storage protection feature (required for multiprogramming).

6. Additional main storage up to 16,777,216 bytes.

7. Universal character set.

## I/O DEVICES

I/O devices supported by the DOS System Control are:

1. 1442 Card Read Punch.
2. 2501 Card Reader.
3. 2520 Card Read Punch.
4. 2540 Card Read Punch.
5. 1403 Printer.
6. 1404 Printer (for continuous forms only).
7. 1443 Printer.
8. 1445 Printer.
9. 1052 Printer-Keyboard (Used for operator communications).
10. 2671 Paper Tape Reader.
11. 2311 Disk Storage Drive.
12. 2321 Data Cell Drive.
13. 2401, 2402, 2403, 2404, and 2415 Magnetic Tape Units.
14. 1285 Optical Reader
15. 2260 Display Station.
16. 1030 Data Collection System.
17. 1050 Data Communication System.
18. 1060 Data Communication System.
19. AT&T 83B3 Selective Calling Stations.
20. AT&T Teletypewriter Terminal, Models 33 and 35.
21. Western Union Plan 115A Outstations.

Item 15, the 2260 Display Station, requires an IBM 2848 Display Control Unit and may be attached directly to a system channel or to a designated communication data set.

Items 16 through 21 are attached by means of a private, leased, or common-carrier network to the multiplexor channel through a 2701 Data Adapter Unit, 2702 or 2703 Transmissions Control Unit. With the 2701, 2702, or 2703 attached to the multiplexor channel, burst-mode devices (magnetic tape and DASD) must be attached to a selector channel.

## SYSTEM I/O DEVICES

The I/O devices used to perform system input and output are called system units. The symbolic designations for the system units are:

- SYSRES (system residence) - a 2311 Disk Storage Drive selected for system residence.

- SYSLOG (system log) - a 1052 Printer Keyboard or a printer selected for operator/system communication.

- SYSRDR (system reader) - a card reader or magnetic tape unit, or optionally a 2311, selected as the control-statement input unit. See Note 1.

- SYSIPT (system input) - a card reader or magnetic tape unit, or optionally a 2311, selected as the primary system input unit. See Note 1.

  Note 1: Optionally, SYSRDR and SYSIPT may both be assigned to the same DASD file. SYSIN is a name used when SYSRDR and SYSIPT are assigned to the same card reader or magnetic tape unit. This name must be used when SYSRDR and SYSIPT are assigned to the same disk extent.

- SYSLST (system list) - a printer, or magnetic tape unit, or, optionally, a DASD selected as the primary printed output unit of the system.

- SYSPCH (system punch) - a card punch, or magnetic tape unit, or optionally, a DASD selected as the primary punched output unit of the system. See Note 2.

  Note 2: SYSOPT, of Basic Programming Support (BPS) and Basic Operating System (BOS), is equated to SYSPCH by macro generation in the DOS. SYSOUT is a name that must be used when SYSPCH and SYSLST are assigned to the same magnetic tape unit.

| | SYSRDR | SYSIPT | SYSRES | SYSLOG | SYSLST | SYSPCH | SYS002 | SYSLNK | SYS001 |
|---|---|---|---|---|---|---|---|---|---|
| MAINT | IN | | IN | OUT | OUT | | | | |
| MAINTA | | | I/O | | | | | | |
| MAINTC2 | | | I/O | | | | | | |
| MAINTCN | | | I/O | | | | | | |
| MAINTR2 | | IN | I/O | | | | | | |
| MAINTS2 | | IN | I/O | | | | | | |
| $LNKEDTC | | | I/O | OUT | OUT | | | | |
| $MAINEOJ | | | I/O | | OUT | | OUT* | | |
| CORGZ | IN | | IN | OUT | OUT | | OUT | | |
| DSERV | IN | | IN | OUT | OUT | | | | |
| RSERV | IN | | IN | OUT | OUT | OUT | | | |
| SSERV | IN | | IN | OUT | OUT | OUT | | | |
| LINKAGE EDITOR | | | I/O | OUT | OUT | | | IN | I/O |
| JOB CONTROL | IN | IN | | I/O | OUT | | | OUT | |

I/O = Input and Output
* If called by CORGZ

Figure 2. System I/O Flow

- SYSUSE - Logical unit block (LUB) used exclusively by System Control to schedule all operator-initiated I/O unit manipulation.

- SYSLNK - a magnetic tape, or DASD device used primarily for I/O by the linkage editor program.

- SYSFGI - a logical unit used in foreground initiation.

Note 3: With the exception of SYSRES and SYSLOG, system units are used only with programs running in a batched-job environment (referred to as background programs).

System I/O flow is shown in Figure 2.

COMPONENTS

Functionally, the DOS, Version 2, is subdivided into the following components:

- System residence
- System control programs
- Linkage editor program
- Librarian
- Processing programs

Each component has unique characteristics, which are given a general presentation in this section.

## SYSTEM RESIDENCE

System residence (SYSRES) is the IBM 2311 Disk Storage Drive on which the system residence 2311 disk pack has been mounted.

System residence consists of the elements of the DOS System Control. These elements are:

| | | Cyl. | Trk. |
|---|---|---|---|
| 1. | IPL retrieval program | 0 | 0 |
| 2. | System directory | 0 | 1 |
| 3. | System work area (librarian area) | 0 | 2-4 |
| 4. | Transient directory | 0 | 5 |
| 5. | Open directory | 0 | 6 |
| 6. | Library routine directory | 0 | 7 |
| 7. | Foreground program directory | 0 | 8 |
| 8. | Problem program phase directory | 0 | 9 |
| 9. | Core image master directory | 1 | 0 |
| 10. | Core image library | - | - |
| 11. | Relocatable library directory | - | - |
| 12. | Relocatable library | - | - |
| 13. | Source statement library directory | - | - |
| 14. | Source statement library | - | - |
| 15. | Label storage area (volume area) | - | - |

Elements 1 through 9 have fixed locations in SYSRES. Elements 10 through 14 do not have fixed locations. The starting address of each element is determined by the size (allocation) and the starting address of the preceding element. However, they must appear in the sequence shown.

Elements 1 through 10 and 15 are required for a minimum SYSRES. Elements 11 through 14 are optional.

For additional information on SYSRES refer to Section 2: System Residence Organization.

## SYSTEM CONTROL PROGRAMS (CHART 00)

The DOS, Version 2, is controlled by three major programs:

1. IPL (initial program load) program.
2. Job control program ($JOBCTLA).
3. Supervisor control program ($$A$SUP1).

These programs allow operating system capability by providing the necessary interface between the IBM System/360, its supporting I/O devices, the operator, system residence, and the program being executed.

### IPL

The IPL program must be executed each time it is necessary to load a new supervisor control program or to change the channel and unit assignment for SYSRES.

The IPL program:

1. Operates in the supervisor mode.
2. Loads the supervisor from SYSRES.
3. Initializes the supervisor for system operation.
4. Places the system in the problem mode.
5. Exits to EOJ when it is finished.

For additional information refer to Section 4: System Control Programs, IPL Program.

### Job Control Program ($JOBCTLA-$JOBCTLJ)

The job control program provides job-to-job transition for background programs. It is also used to prepare each background job step for execution. (One or more programs can be executed within a single job. Each such execution is called a job step.)

Job control performs various functions on the basis of information provided in job control statements. These functions are:

- Preparing the system for execution of programs in a batched-job environment.
- Assigning device addresses to symbolic units.
- Setting up fields in the supervisor communication region.

- Editing and storing volume and file label information.
- Preparing for restarting checkpointed programs.
- Clearing the background problem program area to binary zeros between job steps.

Job control is executed in the background program area and is overlaid by the job step it is preparing for execution.

For additional information refer to Section 4: System Control Programs, Job Control Program.

Supervisor Control Program ($$A$SUP1)

The supervisor program operates with problem programs when job processing (problem program execution) occurs. The supervisor program is divided into two parts:

1. the resident part called the supervisor nucleus
2. the nonresident part called a supervisor transient.

The nucleus is loaded into main storage at IPL time and remains there throughout job processing. A transient (one of many) is loaded from the core image library of SYSRES on an as-needed basis. When a transient has finished performing its service, it can be overlaid by some other transient when some other type of service is required. This technique maximizes the use of main storage allotted to the supervisor. The basic functions performed by the supervisor are:

- Storage protection (required for multiprogramming)
- Interrupt handling
- Channel scheduling
- Device error recovery
- Operator communications
- Program retrieval (fetch or load)
- End-of-job processing
- Timer services (optional)

Each installation must generate its own tailor made supervisor by means of a one time assembly. Supervisor generation macros are used to control the generation of the supervisor control program. Reassembly is required whenever the user wants to change the capability of the supervisor. An example of this is when the installation configuration changes.

For additional information refer to Section 4: System Control Programs, Supervisor Program, Supervisor Transient Programs, Physical IOCS Transients, and Section 3: Supervisor Generation and Organization.

LINKAGE EDITOR PROGRAM ($LNKEDT), CHART 00

All programs to be executed in the DOS environment must be link-edited and stored in the core image library before they can be executed. The link-edit function is accomplished by the linkage editor program operating in one of three modes:

1. Catalog mode. An object module is link-edited and permanently stored in the core image library. The core image and system directories are updated in this mode of operation.
2. Load and execute mode. An object module is link-edited for temporary storage in the core image library and is immediately executed.
3. Assemble and execute mode. A source module is assembled or compiled. The object module (output) is link-edited for temporary storage in the core image library and is immediately executed.

Note: When operating in modes 2 or 3, the core image and system directories are not updated.

The linkage editor program is called by job control when a // EXEC LNKEDT control statement is read. Control is always returned to job control when the link-edit function is completed.

For additional information refer to Section 5: Linkage Editor Program.

LIBRARIAN PROGRAMS

This section presents a group of programs that maintain, service, and organize the libraries and directories of a DOS resident system. These programs are collectively referred to as the Librarian. Functionally, they are divided into three groups:

1. Maintenance programs

2. Organization programs

3. Service programs

Maintenance Programs (Chart 00)

These programs perform the functions that catalog, delete, rename, reallocate, and condense the libraries of SYSRES. The following is a list of the maintenance programs:

1. Common library maintenance program (MAINT)

2. Core image library maintenance program (MAINTC2)

3. Relocatable library maintenance program (MAINTR2)

4. Source statement library maintenance program (MAINTS2)

5. Transient and library-routine directory update program ($MAINEOJ)

6. Library condense program (MAINTCN)

7. System reallocation program (MAINTA)

8. Store condense limits program (MAINTCL)

Common Library Maintenance Program (MAINT): This program is in storage during the execution of all system maintenance functions. It is called by job control when a // EXEC MAINT control statement is read or by a $MAINEOJ if an automatic condense is required.

The prime function of MAINT is to fetch the correct maintenance program to perform a specific maintenance function. This is accomplished by reading and analyzing control statements from SYSRDR or SYSIPT. The following is a list of control statements acceptable to MAINT:

| | |
|---|---|
| • RENAMC<br>• DELETC | Fetch MAINTC2 |
| • CATALR<br>• RENAMR<br>• DELETR | Fetch MAINTR2 |
| • CATALS<br>• RENAMS<br>• DELETS | Fetch MAINTS2 |
| • CONDS | Fetch MAINTCN |
| • CONDL | Fetch MAINTCL |
| • ALLOC | Fetch MAINTCN (Note 1) |
| • /* | Fetch job control |
| • IPTCTRL | Read librarian statement on SYSIPT |
| • RDRCTRL | Read librarian statement on SYSRDR |

Note 1: MAINT always fetches MAINTCN when an ALLOC control statement is read. MAINTCN performs the library condense function before fetching MAINTA to perform the library reallocation function specified by the ALLOC control statement.

For additional information, refer to Section 6: Librarian Maintenance Programs, Common Library Maintenance Program.

Core Image Maintenance Program (MAINTC2): This program is fetched by MAINT to perform the rename or delete functions for the core image library. When fetched, MAINTC2 shares the problem program area with MAINT. Control is returned to MAINT when the desired function is completed.

The RENAMC control statement specifies that a phase of the core image library is to be renamed. The DELETC control statement specifies that a phase of the core image library is to be deleted. The catalog function for the core image library is always performed by the linkage editor program (Phase 8, $LNKEDTC).

For additional information, refer to Section 6: Librarian Maintenance Programs, Core Image Library Maintenance Program.

Relocatable Library Maintenance Program (MAINTR2): This program is fetched by MAINT to perform the catalog, rename, or delete functions for the relocatable library. When fetched, MAINTR2 shares the problem program area with MAINT. Control is returned to MAINT when the desired function is completed.

The CATALR control statement specifies that a module is to be cataloged to the relocatable library. The RENAMR control statement specifies that a module of the relocatable library is to be renamed. The DELETR control statement specifies that a module of the relocatable library is to be deleted.

For additional information, refer to Section 6: Librarian Maintenance Programs, Relocatable Library Maintenance Program.

Source Statement Library Maintenance Program (MAINTS2): This program is fetched by MAINT to perform the catalog, rename, or delete functions for the source statement library. When fetched, MAINTS2 shares the problem program area with MAINT. Control is returned to MAINT when the desired function is completed.

The CATALS control statement specifies that a book is to be cataloged to the source statement library. The RENAMS control statement specifies that a book of the source statement library is to be renamed. The DELETS control statement specifies that a book of the source statement library is to be deleted.

For additional information, refer to Section 6: Librarian Maintenance Programs, Source Statement Library Maintenance Program.

Update Transient, Foreground Program, Open and Library-Routine Directories Program ($MAINEOJ): This program may be fetched by MAINTCN (in the case of an automatic condense), MAINT or CORGZ. It updates the transient, foreground program, open and library-routine directories, and to print the system status report on SYSLST after the execution of any of the following:

- A linkage editor catalog function.
- A core image library rename or delete function.
- A library condense function.
- A library reallocation function.
- A copy system function (CORGZ).

For additional information, refer to Section 6: Librarian Maintenance Programs, Update Transient and Library-Routine Directories Program.

Library Condense Program (MAINTCN): This program is fetched by MAINT to perform the condense function for the system libraries. When fetched, MAINTCN shares the problem program area with MAINT.

The CONDS control statement specifies that one of the following condense functions must be performed and that control is to be returned to MAINT:

- Condense all libraries.
- Condense selected libraries.
- Condense an individual library.

MAINTCN is also fetched by $MAINEOJ for automatic condensing and by MAINT when the ALLOC control statement is read. When fetched under this circumstance:

- MAINTCN still shares the problem program area with MAINT.
- All libraries are condensed.
- MAINTA is fetched when the condense function is completed.

For additional information, refer to Section 6: Librarian Maintenance Programs, Library Condense Program.

Store Condense Limits Program (MAINTCL): This program is fetched by MAINT, and, when executed, stores library condense information in the system directory. The information stored by $MAINTCL is used by $MAINEOJ to determine if an automatic condense is required (when a nonzero parameter is specified by the control statement). If an automatic condense is to be done, the condense limit has been posted by MAINTCL.

System Reallocation Program (MAINTA): This program is fetched by MAINTCN when the reallocation function (ALLOC control statement) has been detected by MAINT. The reallocation function is used to redefine the sizes of the libraries and directories of SYSRES.

MAINT detects the ALLOC control statement and fetches MAINTCN to condense all libraries before the reallocation function is performed. When fetched, MAINTA overlays MAINTCN and shares the problem program area with MAINT. $MAINEOJ is fetched when the reallocation function is completed.

For additional information, refer to Section 6: Librarian Maintenance Programs, System Reallocation Program.

Organization Programs (Chart 00)

The Copy System Program (CORGZ) is the only program in this category. It is fetched by job control when the // EXEC CORGZ control statement is read. Its function is to copy

SYSRES, either selectively or completely. A complete copy generates backup; a selective copy generates a reduced system that is to be used for a specific purpose.

The CORGZ program has the additional capability of performing the reallocation function.

Upon completion, the CORGZ program fetches $MAINTEOJ to update the transient and library-routine directories, and print the system status report, of the new SYSRES.

For additional information, refer to Section 7: Librarian Organization Programs, Copy System Program.

### Service Programs (Chart 00)

These programs perform the functions that:

1. Display and/or punch books from the source statement library, and modules from the relocatable library.
2. Display the contents of the directories in SYSRES.

The service programs are briefly described as follows. For additional information, refer to Section 8: Librarian Service Programs, RSERV, SSERV, DSERV.

Relocatable Library Service Program (RSERV): This program displays and/or punches modules from the relocatable library.

Source Statement Library Service Program (SSERV): This program displays and/or punches books from the source statement library.

Directory Service Program (DSERV): This program displays the contents of the directories in SYSRES. All directories can be displayed in a single run or they may be displayed selectively.

## PROCESSING PROGRAMS (CHART 00)

All programs executed in the DOS environment use the functions of the system control programs. A minimum system residence may consist of only the system control programs and one or more user programs. A full system residence may consist of the following components:

1. System control programs
2. Linkage editor program
3. Librarian maintenance programs
4. Librarian organization programs
5. Librarian service programs
6. Processing programs
   a. Language Translators
      (1). Assembler
      (2). COBOL
      (3). FORTRAN
      (4). RPG
      (5). PL/I
   b. Sort/Merge
   c. Utilities
   d. Autotest
   e. User programs

This section presents the organization of a disk resident system as received from the Program Information Distribution Center (PID) and after system generation.

The user receives the disk resident system on a 2311 disk pack. Certain areas of the disk pack are predefined. These areas and their content are as follows:

1. IPL. This area contains the IPL bootstrap program and the volume label.

2. System directory. This area contains the system master directory. It consists of records that show the status, location, description, and allocation of each library and directory in the system. This area also contains the IPL retrieval program ($$A$IPL2).

3. System work area (Librarian area). This area is reserved for use as a system work area by the linkage editor, job control, and the librarian programs.

4. Transient directory. This area' contains the directory of the transient routines located in the core image library.

5. Open (LIOCS) directory. This area contains a directory of the phases of the logical input/output control section (LIOCS) OPEN function.

6. Library-routine directory. This area contains a directory of the system programs located in the core image library.

7. Foreground Program directory. This area contains a directory of the foreground program phases.

8. Phase directory. This area is reserved for the directory of phases of a problem program.

9. Core image directory. This area contains the directory of all the phases in the core image library.

10. Core image library. This area contains the following programs, in core image format:

    a. System control programs

    - IPL program ($$IPLRT2)

    - Supervisor control program ($$A$SUP1), includes PIOCS.

    - Job control program ($JOBCTLA)

    b. Linkage Editor Program ($LNKEDT)

    c. Librarian programs

    - Common library maintenance (MAINT)

    - Core image library maintenance (MAINTC2)

    - Relocatable library maintenance (MAINTR2)

    - Source statement library maintenance (MAINTS2)

    - Transient and library-routine directory update program ($MAINEOJ)

    - Library condense (MAINTCN)

    - Store condense limits (MAINCL)

    - Library reallocation (MAINTA)

    - Copy system (CORGZ)

    - Directory service (DSERV)

    - Relocatable library service (RSERV)

    - Source statement library service (SSERV)

    d. Processing programs

    - Assembler

11. Relocatable library directory. This area contains the directory of all the modules in the relocatable library.

12. Relocatable library. This area contains programs in relocatable format (language translator output). All programs that are in the core image library are contained in this area. In addition this area can contain the following programs:

    - COBOL

    - FORTRAN

    - RPG

- PL/I
- Sort/Merge
- Utilities
- Autotest

13. Source statement directory. This area contains the directory of all the books in the source statement library.

14. Source statement library. This area contains books in source-language format. The books supplied are macro definitions in the assembler sub-library. Included are the supervisor macros and the logical IOCS macros.

## SYSTEM RESIDENCE ORGANIZATION AFTER GENERATION

Once system generation is completed, the user has a system residence that is specifically designed for his configuration and special features.

Certain areas of any system residence are fixed and do not change. Figure 3 shows the organization of a full system residence.

- Items 1 through 10, and 15 are required in any system residence.
- Items 1 through 9 have fixed locations.
- Items 10 through 16 have variable locations that are dependent on the existence and allocation (size) of preceding items.
- Items 11-14 and 16 are optional. If one or both of the optional libraries (items 12 and 14) are not allocated, the associated directory is not allocated.

  - The directory of each library-directory pair (items 9, 11, and 13) starts on a new cylinder (CC) at track (HH) 00.

  - The library of each library-directory pair starts on a new track (HH) and utilizes all of the last allocated cylinder (HH = 9).
- The volume area (item 15) requires a full cylinder.
- System residence is contained in a contiguous area of the disk pack. The starting address is (BB = 00, CC = 00, HH = 00, and R = 1). The ending address is (BB = 00, CC = nn, HH = 09, and R = n).

  nn = the cylinder assigned to the volume area. It is dependent on the allocation specified by the user for the core image, relocatable, and source statement library and directory pairs.

  n = the last record of the last track of the volume area.

<table>
<tr><th rowspan="2">NO.</th><th rowspan="2" colspan="2">COMPONENT</th><th colspan="4">STARTING DISK ADDRESS</th><th rowspan="2">NUMBER OF TRACKS (Allocation)</th><th rowspan="2">R = REQUIRED<br>O = OPTIONAL</th></tr>
<tr><th>BB</th><th>CC</th><th>HH</th><th>R</th></tr>
<tr><td rowspan="4">1</td><td colspan="2">IPL Bootstrap Record 1 ($A$IPL1)</td><td>00</td><td>00</td><td>00</td><td>1</td><td rowspan="4">1</td><td>R</td></tr>
<tr><td colspan="2">IPL Bootstrap Record 2 ($A$IPLA)</td><td>00</td><td>00</td><td>00</td><td>2</td><td>R</td></tr>
<tr><td colspan="2">SYSRES Vol Label (Z)</td><td>00</td><td>00</td><td>00</td><td>3</td><td>R</td></tr>
<tr><td colspan="2">User Vol Label</td><td>00</td><td>00</td><td>00</td><td>4</td><td>O</td></tr>
<tr><td rowspan="5">2</td><td rowspan="4">System Directory</td><td>Record 1</td><td>00</td><td>00</td><td>01</td><td>1</td><td rowspan="5">1</td><td>R</td></tr>
<tr><td>Record 2</td><td>00</td><td>00</td><td>01</td><td>2</td><td>R</td></tr>
<tr><td>Record 3</td><td>00</td><td>00</td><td>01</td><td>3</td><td>R</td></tr>
<tr><td>Record 4</td><td>00</td><td>00</td><td>01</td><td>4</td><td>R</td></tr>
<tr><td colspan="2">IPL Retrieval Program ($$A$IPL2)</td><td>00</td><td>00</td><td>01</td><td>5</td><td>R</td></tr>
<tr><td>3</td><td colspan="2">System Work Area (Librarian Area)</td><td>00</td><td>00</td><td>02</td><td>1</td><td>3</td><td>R</td></tr>
<tr><td>4</td><td colspan="2">Transient Directory ($$A and $$B Transients)</td><td>00</td><td>00</td><td>05</td><td>1</td><td>1</td><td>R</td></tr>
<tr><td>5</td><td colspan="2">Open Directory ($$B0)</td><td>00</td><td>00</td><td>06</td><td>1</td><td>1</td><td>R</td></tr>
<tr><td>6</td><td colspan="2">Library Routine Directory ($ Phasenames)</td><td>00</td><td>00</td><td>07</td><td>1</td><td>1</td><td>R</td></tr>
<tr><td>7</td><td colspan="2">Foreground Program Directory (FGP)</td><td>00</td><td>00</td><td>08</td><td>1</td><td>1</td><td>R</td></tr>
<tr><td>8</td><td colspan="2">Phase Directory (For Problem Program Phases)</td><td>00</td><td>00</td><td>09</td><td>1</td><td>1</td><td>R</td></tr>
<tr><td>9</td><td colspan="2">Core Image Library Directory</td><td>00</td><td>01</td><td>00</td><td>1</td><td>*</td><td>R</td></tr>
<tr><td rowspan="2">10</td><td rowspan="2" colspan="2">Core Image Library</td><td rowspan="2">00</td><td colspan="2">End of CI Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">R</td></tr>
<tr><td>X</td><td>Y+1</td></tr>
<tr><td rowspan="2">11</td><td rowspan="2" colspan="2">Relocatable Library Directory</td><td rowspan="2">00</td><td colspan="2">End of CI Library</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>A+1</td><td>00</td></tr>
<tr><td rowspan="2">12</td><td rowspan="2" colspan="2">Relocatable Library</td><td rowspan="2">00</td><td colspan="2">End of RL Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y+1</td></tr>
<tr><td rowspan="2">13</td><td rowspan="2" colspan="2">Source Statement Library Directory</td><td rowspan="2">00</td><td colspan="2">End of RL Library</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>B+1</td><td>00</td></tr>
<tr><td rowspan="2">14</td><td rowspan="2" colspan="2">Source Statement Library</td><td rowspan="2">00</td><td colspan="2">End of SS Directory</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>X</td><td>Y+1</td></tr>
<tr><td rowspan="2">15</td><td rowspan="2" colspan="2">Volume Area (Label Storage Area)</td><td rowspan="2"></td><td colspan="2">End of SS Library</td><td rowspan="2">1</td><td rowspan="2">10</td><td rowspan="2">R</td></tr>
<tr><td>C+1</td><td>00</td></tr>
<tr><td rowspan="2">16</td><td rowspan="2" colspan="2">User Area</td><td rowspan="2"></td><td colspan="2">End of Volume Area</td><td rowspan="2">1</td><td rowspan="2">*</td><td rowspan="2">O</td></tr>
<tr><td>C+2</td><td>00</td></tr>
</table>

X = Ending CC of the Preceding Directory
* = Allocation Dependent on User Requirements
A = Ending CC of Core Image Library
B = Ending CC of Relocatable Library
C = Ending CC of Source Statement Library
Y = Ending HH of the Previous Directory
Z = This Volume Label Contains the Address of the VTOC Established when the Pack was Initialized.

Figure 3. System Residence Organization

## IPL

Refer to Section 4: System Control Programs for information on IPL record formats.

## SYSTEM DIRECTORY

This directory consists of five records that make up the system master directory. Records 1 through 4 are 80 bytes in length.

Records 1 through 3 contain information describing the core image library and directory, the relocatable library and directory, and the source statement library and directory, respectively.

Record 4 contains information describing the allocation of the library and directory pairs, and the beginning cylinder number of the Label (Volume) area.

Record 5 is the IPL retrieval program ($$A$IPL2).

Figure 4 shows the record formats of the system directory records.

## TRANSIENT DIRECTORY

This single track directory contains entries for the A and B transient routines, which are located in the core image library. The entries in this directory are taken from the core image library directory. A separate directory permits faster retrieval of the A and B transients. The core image library phases that are referenced in this directory have a phase name prefixed by $$A (type A transients) or $$B (type B transients). This directory has a maximum capacity of 144 entries. Track format is identical to the core image library directory (see Figure 6).

## OPEN DIRECTORY

This single track directory contains entries for the LIOCS open phases located in the core image library. The entries in this directory are taken from the core image library directory. A separate directory permits faster retrieval of LIOCS open phases. The core image library phases referenced in this directory have phase names prefixed by the characters $$BO. This directory has a maximum capacity of 144 entries. Track format is identical to the core image library directory (see Figure 6).

## LIBRARY ROUTINE DIRECTORY

This single-track directory contains entries for frequently used core image library phases such as job control, linkage editor, etc. The entries in this directory are taken from the core image library directory. A separate directory permits faster retrieval of these phases. The core image library phases that are placed in this directory have a phase name prefixed by a $, (example $LNKEDT). This entry has a maximum capacity of 144 entries. Track format is identical with the core image library directory (see Figure 6).

Format of records 1 - 3 (Record 1, contains core image directory and library information; record 2, relocatable directory and library information; and record 3, source statement library and directory information.)

| 0 | 7 | 15 | 23 | 30 | 37 | 44 | 46 | 48 | 50 | 52 | 54 | 56 – 79 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BBCCHHR | BBCCHHRE | BBCCHHRE | BBCCHHR | BBCCHHR | BBCCHHR | XX | XX | XX | XX | XX | XX | RESERVED |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Fields (The 1st byte contains blank and the remainder of the record contains zeros if no library allocated).

1. Starting address of the directory.
2. Address of the next available entry in the directory.
3. Ending address of the directory including last entry.
4. Starting address of the library.
5. Address of next available entry in the library.
6. Ending address of the library.
7. Number of active entries in the directory.
8. Number of blocks allocated for the library.
9. Number of active blocks in the library.
10. Number of deleted blocks in the library.
11. Number of blocks available for additions.
12. Automatic condence limit
13. Reserved

Format of record 4 (Contains allocation information for each system library and directory.)

| 0 | | 2 | | 4 | | 6 | | 8 | | 10 | | 12 | 13 – 79 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C | C | T | T | C | C | T | T | C | C | T | T | C | RESERVED |
| 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | 8 |

Fields (The fields contain zeros if not allocated.)

1. Total number of cylinders (library + directory) } core image
2. Number of directory tracks } core image
3. Total number of cylinders (library + directory) } relocatable
4. Number of directory tracks } relocatable
5. Total number of cylinders (library + directory) } source statement
6. Number of directory tracks } source statement
7. Address of label storage cylinder, C, (volume area).
8. Reserved

Record 5 (IPL retrieval program - $$A$IPL2)

Figure 4. System Directory Record Formats

FOREGROUND PROGRAM DIRECTORY

This single track directory contains entries for the foreground program phases located in the core image library. The entries in this directory are taken from the core image library directory. A separate directory permits faster retrieval of foreground program phases. The core image library phases referenced in this directory have phase names prefixed by the characters FGP. This directory has a maximum capacity of 144 entries. Track format is identical to the core image library directory (see Figure 6).

## SYSTEM WORK AREA (LIBRARIAN AREA)

This 3-track area is reserved as a work area for the librarian programs and job control. The format of the records in the librarian area is dependent upon the program using the area at a specific time. Figure 5 shows the record formats that may be found in the librarian area.

## PHASE DIRECTORY

This single track directory contains entries for the phases of the current problem program. The entries in this directory are constructed by job control before each job step is executed. They are taken from the core image library directory. A separate directory permits faster retrieval of the phases of a program.

The phase naming conventions used to permit the use of the phase directory are:

1. All program names must be unique in the first four characters.
2. The first four characters of the name of each phase of a program must be identical to the first four characters of the program name. All 8 characters of the first phase name must be identical to the program name.

   Example: WXYZPROG

   WXYZPROG - phase 1
   WXYZPH1 - phase 2
   WXYZPH2 - phase 3

The maximum capacity of this directory is 144 entries. Track format is identical to the core image library directory (see Figure 6).

## CORE IMAGE LIBRARY DIRECTORY

This directory consists of one or more tracks, dependent on the allocation specified by the user. It contains one entry for each of the phases in the core image library.

Note: A phase is an overlay of a multiphase program or a complete program if not multiphase.

Each directory entry describes one phase in the core image library and contains:

- Phase name
- Loading address
- Entry point
- Starting disk address in the core image library
- Number of blocks
- Length of last block

Figure 6 shows the track, block, and entry format of the core image library directory.

As used by Job Control

| | | | | |
|---|---|---|---|---|
| Each Track | Block 1 | Block 2 | Block 3 | Block 41 (maximum) |

| Block 2 | 1 Entry (20 Bytes) | | | | | |
|---|---|---|---|---|---|---|
| | Phase Name | Load Address | No. of Blocks | Entry Address | CI Library Address (CHR) | Length of Last Block |
| Number of Bytes | 8 | 3 | 1 | 3 | 3 | 2 |

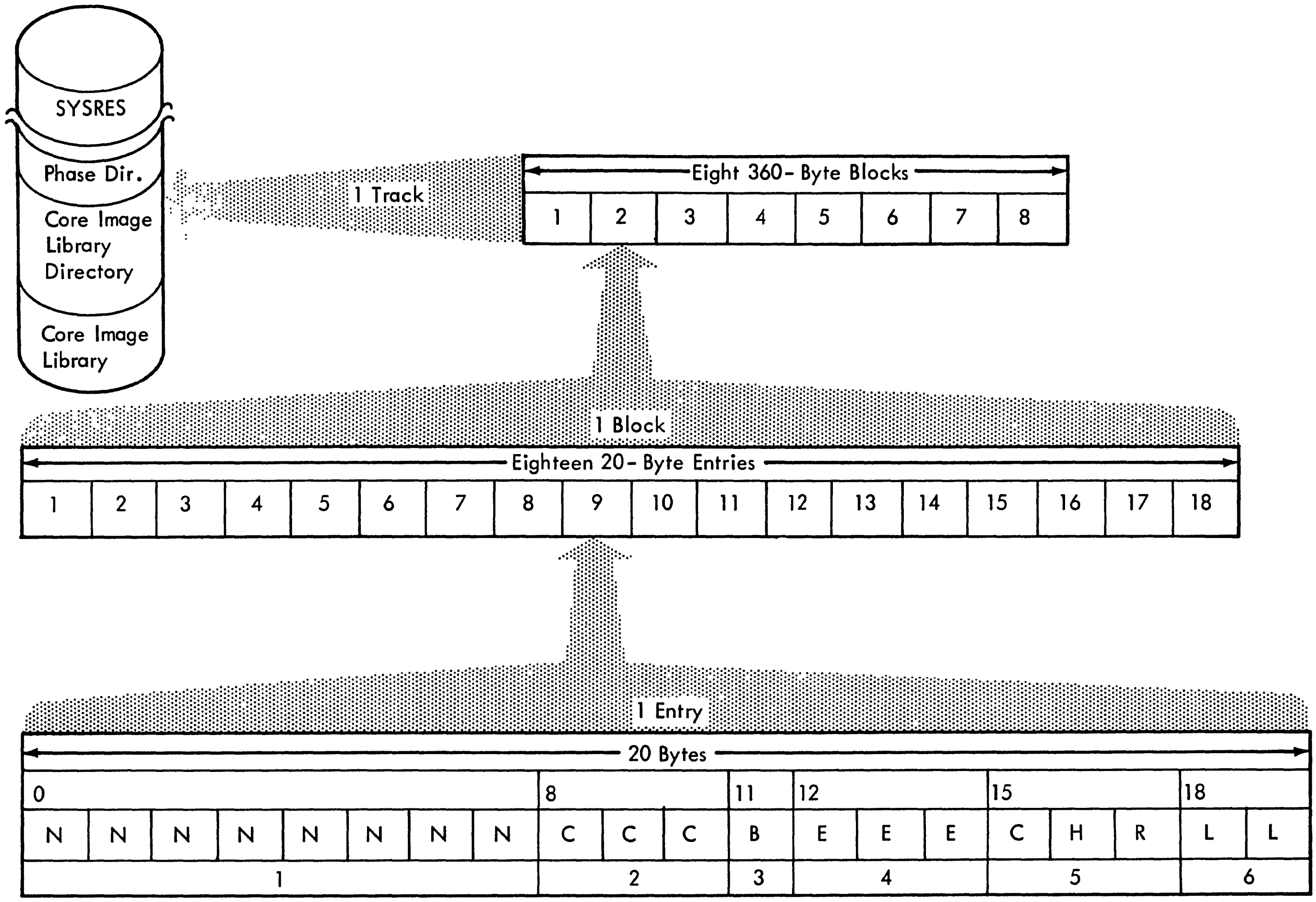
As used by Linkage Editor ($LNKEDTA phase)

| | | | | | | |
|---|---|---|---|---|---|---|
| Each Track | Block 1 | Block 2 | Block 3 | Block 4 | Block 5 | Block X |

| | Unblocked Phase Entries (24 Bytes Each) | | | | | |
|---|---|---|---|---|---|---|
| | Phase Name | Entry Address | No. of Blocks | Length of Last Block | CI Library Address | Load Address |
| | NNNNNNNN | EEEE | BB | LL | CHHR | LLLL |
| Number of Bytes | 8 | 4 | 2 | 2 | 4 | 4 |

Note: This area is also used by the Reallocation Program (MAINTA) for temporary storage of the Volume Label.

Figure 5. System Work Area Record Formats




Field

1. Phase Name — 8 characters from the PHASE card. (an * in the first byte indicates the logical end of the directory).

2. Loading Address — 3- byte hexadecimal storage address provided at linkage edit time. (Can be overridden at object time).

3. Number of Blocks — the number of physical records (blocks) required to contain the phase in the core image library.

4. Entry Point — 3- byte hexadecimal storage address of the first instruction to be executed. Provided at linkage edit time (can be overridden at object time).

5. Phase Address — starting disk address of the first block of the phase in the core image library. The 3- byte address (CHR) is expanded to 5 bytes (0C0HR) by the using routine.

6. Number of Bytes in the Last Block — each block of a phase contains 1728 bytes of data, except the last block, which can contain fewer data bytes. The data in the last block is padded into a 1728- byte record.

Figure 6. Core Image Directory Format

CORE IMAGE LIBRARY

The core image library consists of one or more tracks, dependent on the allocation specified by the user. Each allocated track contains two blocks with a maximum capacity of 1728 bytes each. The number of programs (phases) and the size of each program to be contained in the core image library dictates the number of tracks that must be allocated. Each program starts with a new block and only the last block of a program can contain less than 1728 bytes of data. The last block, if less than 1728 bytes of data, is padded with zeros.

Note: A phase is an overlay of a multiphase program or a complete program if not multiphase.

Figure 7 shows the organization of the core image library.

RELOCATABLE LIBRARY DIRECTORY

This directory consists of one or more tracks, dependent on the allocation specified by the user. It contains one entry for each module in the relocatable library.

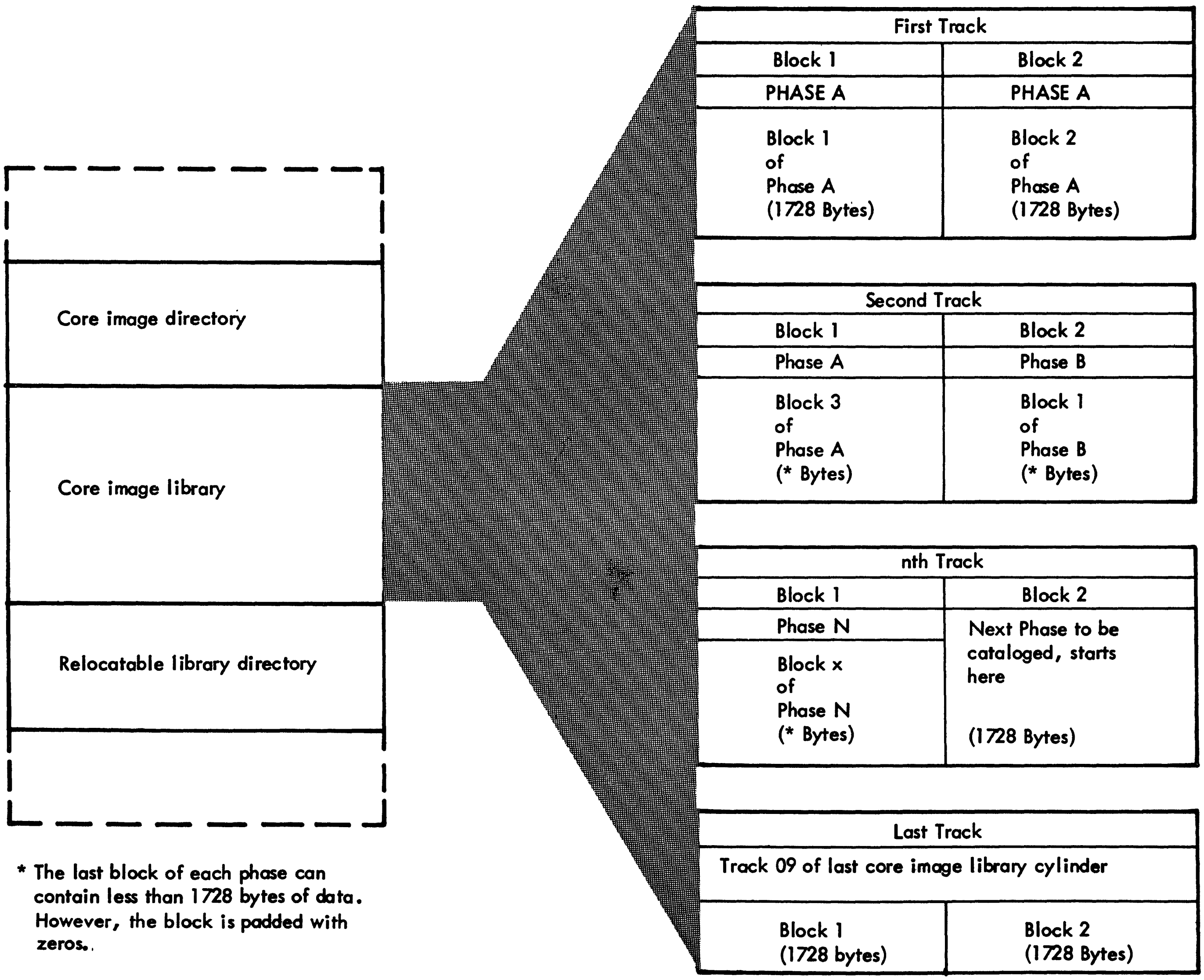


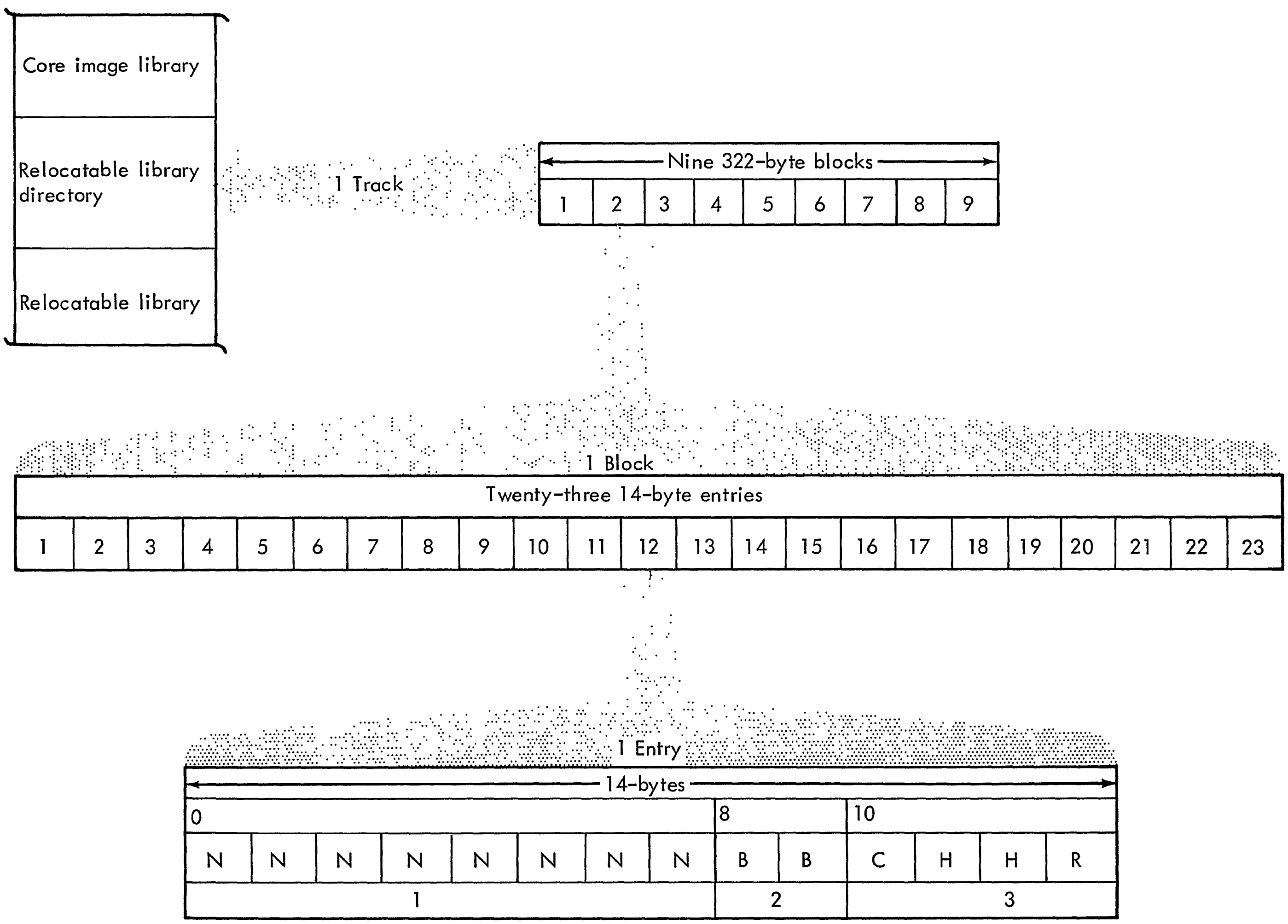

Figure 7. Core Image Library Format

Note: A module is the term applied to the output of a complete language translator run.

Each directory entry describes one module in the relocatable library and contains:

- Module name
- Total number of text-records (blocks) required to contain this module
- Starting disk address of the first text-record of this module.

Figure 8 shows the track, block, and entry format of a single track in the relocatable library directory.




FIELD

1. Module Name - 8 characters from the "CATALR" control statement. An * in the first character indicates the logical end of the directory.

2. Number of Blocks - Total number of text records (BLOCKS) required to contain this module.

3. Disk Address - Starting disk address of the first text record (BLOCK) of this module in the relocatable library.

Figure 8. Relocatable Library Directory Format

## RELOCATABLE LIBRARY

The relocatable library consists of one or more tracks, dependent on the allocation specified by the user. The number of modules and the size of each module to be contained in this library dictates the number of tracks which must be allocated. Each allocated track contains nine blocks with a fixed length of 322 bytes each. Each module starts with a new block but not necessarily a new track.

Figure 9 shows the organization of the relocatable library.

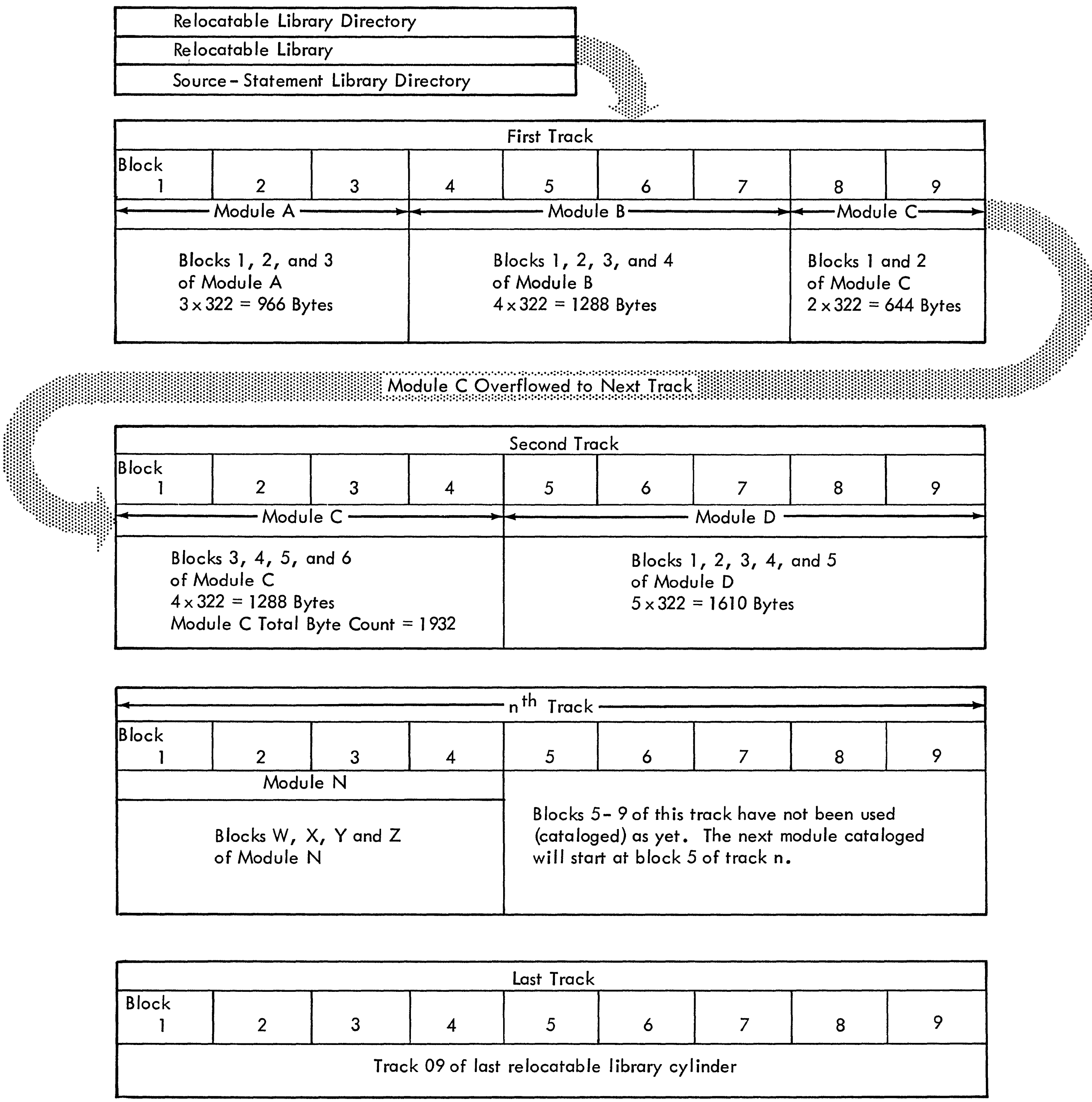



Figure 9. Relocatable Library Format

## SOURCE STATEMENT LIBRARY DIRECTORY

This directory consists of one or more tracks, dependent on the allocation specified by the user. It contains one entry for each book in the source statement library.

Note: A book is the name given to a sequence of source language statements, in compressed card image format, that are accessed by a single name.

Each directory entry describes one book in the source statement library and contains:

- A sublibrary prefix (A = Assembler, C = COBOL)
- Book name
- Starting disk address of the first block of this book
- Total number of blocks required to contain this book in the source statement library

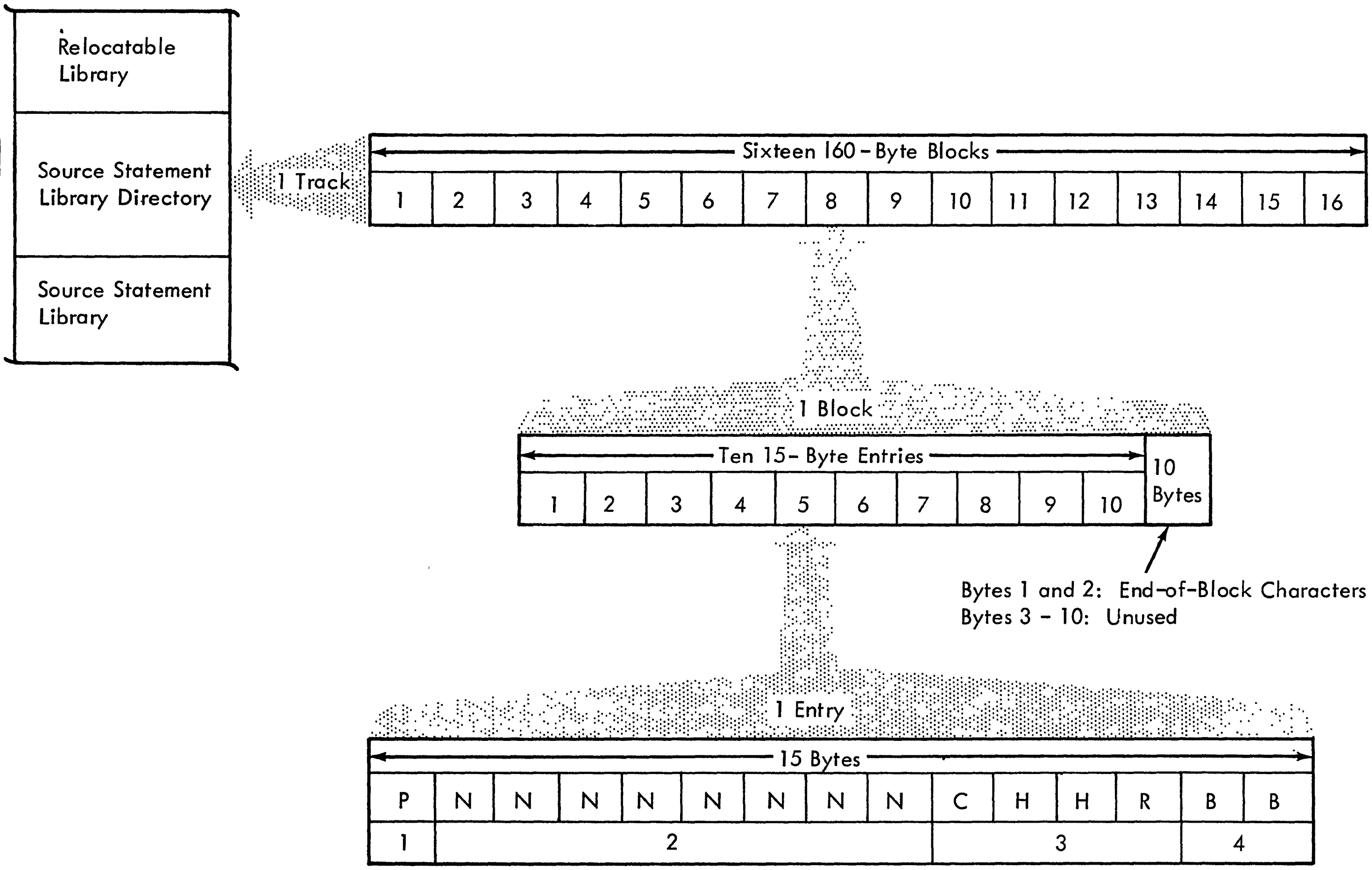
Figure 10 shows the track, block, and entry format of the source statement library directory.

## SOURCE STATEMENT LIBRARY

The source statement library consists of one or more tracks, dependent on the allocation specified by the user. The number of books and the size of each book to be contained in this library dictates the number of tracks which must be allocated. Each track contains 16 blocks with a fixed length of 160 bytes per block. Each book starts with a new block but not necessarily a new track. Each book in the source statement library contains compressed card-images of the source language input to the assembler or COBOL language translators. A compressed card image can overflow from one block to another.

Refer to Section 6: Source Statement Library Maintenance Program for individual record formats.

Figure 11 shows the organization of the source statement library.




Field

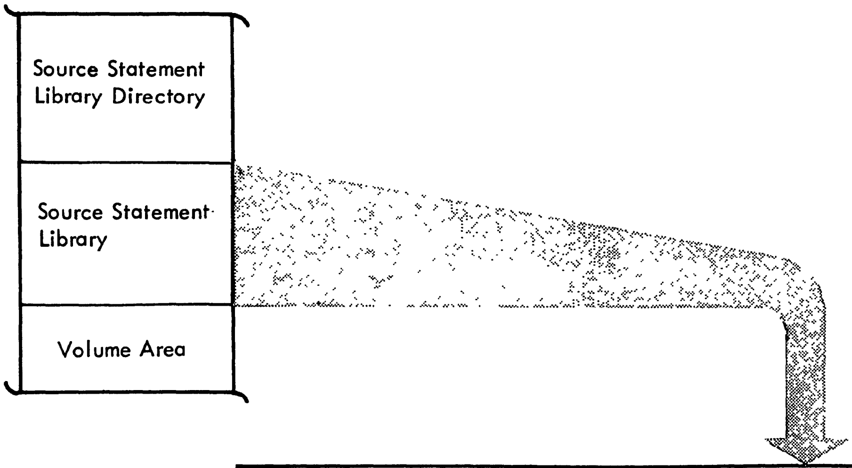
1. Sublibrary prefix - an "A" for the assembler, a "C" for COBOL. An * in this field indicates the logical end of the directory.

2. Book name - 8 characters from the "CATALS" control statement.

3. Disk address - starting disk address of the first block of this book.

4. Number of blocks - the total number of blocks required to contain this book in the source statement library.

Figure 10. Source Statement Library Directory Format




| First Track | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Book A | | | | | Book B | | | | | | | Book C | | | |
| Blocks 1 - 5 of Book A<br>5 x 160 = 800 Bytes | | | | | Blocks 1 - 7 of Book B<br>7 x 160 = 1120 Bytes | | | | | | | Blocks 1 - 4 of Book C<br>4 x 160 = 640 Bytes | | | |

| $n^{th}$ Track | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Book N | | | | | | Blocks 7- 16 of this track have not been used (cataloged) as yet.<br>* The next book cataloged will start at block 7 of track n. | | | | | | | | | |
| Blocks 1 - 6 of Book N<br>6 x 160 = 960 Bytes | | | | | | | | | | | | | | | |

| Last Track | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Track 09 of last source statement library cylinder | | | | | | | | | | | | | | | |

Figure 11. Source Statement Library Format

This section of the manual discusses supervisor generation in the first three major subsections. The first subsection describes general supervisor generation techniques. The second subsection describes the supervisor generation macros and their optional operands. The third subsection describes the relationship between the outer supervisor generation macros and the inner macros that generate the bulk of the supervisor code.

Organization is discussed in the last major subsection.

To understand the third subsection, refer to the IBM publication that explains macro definition language structure and usage. With this information, an SSERV listing of the supervisor generation macros, and the PLM material, the reader can identify those sections of code that are generated for his own supervisor program. The basic instruction used in macro definition language is the AIF (ask if) statement. The following examples show how it is used:

1. AIF (&BG20).MP1
   This instruction asks if multiprogramming support is required (refer to Figure 13 for interpretation of BG20). If BG20 is on, the next line in the SSERV listing that is significant is found at the label, .MP1, and any intervening code is rejected by the language translator. If BG20 is not on, the next sequential line on the SSERV listing is significant.

2. AIF (NOT &BG20).NO23
   This instruction tests the opposite status of the BG20 switch. In this case, the line at location .NO23 is the next significant line in the SSERV listing only when BG20 is not on, that is, only when multiprogramming support is not required.

A detailed description of the AIF instruction and the other instructions used in the SSERV listing is given in the Systems Reference Library (SRL) publication; IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, Form C24-3414.

Basic supervisor organization is described by a core map and descriptions of the various I/O tables and information blocks in the fourth subsection.

## SUPERVISOR GENERATION

The supervisor is assembled with a series of macros that describe the installation's functional requirements, and its configuration. At system generation time, a source deck containing the supervisor generation macros is assembled into an object deck. The job control program places the results of the assembly on SYSLNK (I/O device for the linkage editor program) and calls the linkage editor program. The deck is link-edited and cataloged to the core image library on SYSRES. A corresponding core image library directory entry is posted for the new supervisor and the Program Information Distribution (PID) supervisor directory entry is deleted.

Normally, a condense maintenance program would then be executed to remove the PID supervisor from the core image library. The procedures and sequence of events used in system generation are described in the IBM publication on system generation and maintenance (IBM System/360 DOS System Generation and Maintenance, C24-5033).

Whenever a new supervisor generation is required by the user the same general steps are taken:

1. Punch the macro instructions, together with the selected optional operands, into a card deck.

2. Execute an assembly and put the object modules on SYSLNK (using an include control statement with no operand) via the job control program.

3. Link-edit the new supervisor cataloging it to the core image library, deleting the old supervisor directory entry, and posting the new supervisor directory entry.

4. Execute a maintenance program to condense the core image library, deleting the old supervisor program.

5. Re-IPL with the new supervisor.

## Supervisor Generation Macros

The following list of supervisor macros and optional operands is presented to give the reader:

- Supervisor generation macro names.
- Required macro sequence (as listed).
- Macro parameters (where there is an assumed value, that value is underlined).
- A brief description of what the generated macro does.
- A brief description of what the individual parameter options do.

| Name | Macro Description | Parameter = Option | Option Description |
|---|---|---|---|
| SUPVR | Describes system environment | SYSTEM = $\left\{\begin{matrix}\underline{\text{TAPE}}\\ \text{DISK}\end{matrix}\right\}$ | Indicates the system residence, SYSRES, device type |
| | | MPS = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Indicates multiprogramming support. If YES is specified, the system generated is capable of supporting 2 foreground programs. |
| | | TP = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{BTAM}\end{matrix}\right\}$ | Indicates Teleprocessing support is desired. BTAM is valid only if SYSTEM=DISK. |
| CONFG | Describes hardware features | MODEL = $\left\{\begin{matrix}\underline{30}\\ \text{nn}\end{matrix}\right\}$ | nn defines the System/360 model number (30, 40, etc.). |
| | | SP = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Indicates the storage protection feature is desired. YES is assumed if MPS=YES in the SUPVR macro. |
| | | DEC = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Decimal feature. |
| | | FP = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Floating-point feature. |
| | | TIMER = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Timer feature. TIMER=YES the supervisor macro GETIME is supported. |
| STDJC | Sets standard values for job control variables. | DECK = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Output modules on SYSPCH. |
| | | LIST = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Source modules listings from language translators on SYSLST. |
| | | XREF = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Language translators output symbolic cross-reference lists on SYSLST. |
| | | ERRS = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Compilers summarize all errors in source programs on SYSLST. |
| | | LOG = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Listing of all control statements on SYSLST. |
| | | DUMP = $\left\{\begin{matrix}\underline{\text{YES}}\\ \text{NO}\end{matrix}\right\}$ | Dump of registers and main storage on SYSLST. |
| | | LINES = $\left\{\begin{matrix}\underline{56}\\ \text{nn}\end{matrix}\right\}$ | Number of lines per page on SYSLST. |
| | | DATE = $\left\{\begin{matrix}\underline{\text{MDY}}\\ \text{DMY}\end{matrix}\right\}$ | Format of date. |
| | | CHARSET = $\left\{\begin{matrix}\underline{\text{48C}}\\ \text{60C}\end{matrix}\right\}$ | Specifies the 48 or 60 character set for language translator input on SYSIPT. |
| | | LISTX = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Hexadecimal object module listings from compilers on SYSLST. |
| | | SYM = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Assembler output symbol tables on SYSPCH. |

| Name | Macro Description | Parameter = Option | Option Description |
|---|---|---|---|
| FOPT | Describes functional supervisory options | OC = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Operator initiated communications to problem programs. If OC=YES the facility is available to all programs in MPS. |
| | | PC = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Problem program routine for program check. If PC=YES, the facility is available to all programs in MPS. |
| | | IT = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{BG}\\ \text{F1}\\ \text{F2}\end{matrix}\right\}$ | Problem program ability to set timer intervals and specify a timer interrupt routine. BG, F1, or F2 indicates which program has the facility. F1 or F2 is valid only in MPS. |
| | | TEB = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{n}\end{matrix}\right\}$ | Tape error statistics are to be accumulated and logged where n is the number of tape drives attached to the system. |
| | | CCHAIN = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}$ | Command chaining support for retry on I/O operations. |
| | | DASDFP = $\left\{\begin{matrix}\underline{\text{NO}}\\ \left(\text{n,n}\left[\begin{matrix}\underline{\text{,2311}}\\ \text{2321}\end{matrix}\right]\right)\end{matrix}\right\}$ | Supervisory DASD file protection, where (n,n) indicates the range of channels to which DASD's may be attached. 2321 option indicates the presence of a 2321 device. |
| | | SYSFIL = $\left\{\begin{matrix}\underline{\text{NO}}\\ \text{(2311[,n1,n2])}\end{matrix}\right\}$ | System input and system output (SYSRDR, SYSIPT, SYSLST, SYSPCH) files may be assigned to a 2311,<br><br>n1 = residual capacity (in records) for beginning of operator notification when SYSLST is assigned to a 2311.<br><br>100 ≤ n1 ≤ 65536<br>If n1 is omitted, 1000 is assumed.<br><br>n2 = residual capacity (in records) for beginning of operator notification when SYSPCH is assigned to a 2311.<br><br>100 ≤ n2 ≤ 65536<br>If n2 is omitted, 1000 is assumed. The SYSFIL parameter and its options is valid only when SYSTEM=DISK in the SUPVR macro. |

| Name | Macro Description | Parameter=Option | Option Description |
|---|---|---|---|
| PIOCS | Describes the system I/O configuration | SELCH = { YES<br>NO } | Selector channels attached to the system. |
| | | BMPX = { NO<br>YES } | Burst mode devices will be supported on multiplexor channel. |
| | | CHANSW = { NO<br>RWTAU<br>TSWTCH } | Channel switching tape control unit. RWTAU = 2404 or 2804, TSWTCH = 2816. If either 2404 or 2804 and 2816, RWTAU must be specified. |
| | | TAPE = { 9<br>7<br>NO } | Indicates required tape PIOCS support. 9 = nine track only. This is the assumed value when SYSTEM=TAPE.<br>7 = seven or nine track.<br>NO = No tape drives attached. Invalid when SYSTEM=TAPE. This is the assumed value when SYSTEM=DISK. |
| ALLOC | Partitions storage for MPS (Optional macro). | {F1=nK, F2=nK} | Specifies storage partitioning MPS, where n must be a multiple of 2. |
| IOTAB | Describes installation requirements for I/O tables. | IODEV = { 10<br>n } | Number of I/O devices attached to the system. |
| | | BGPGR = { 10<br>n } | Number of symbolic units of the class SYSnnn for the background program. |
| | | F1PGR = { 5<br>n } | Number of symbolic units for F1. Valid only in MPS. Otherwise zero is assumed. |
| | | F2PGR = { 5<br>n } | Number of symbolic units for F2. Valid only in MPS. Otherwise zero is assumed. |
| | | CHANQ = { 6-8<br>n } | Number of I/O requests in the channel queue. 6 assumed if SYSTEM=DISK, 8 assumed if SYSTEM=TAPE. |
| | | JIB = { 5<br>n } | Number of Job Information Blocks (JIBs) for the system. Requirements are:<br>1. One JIB for each temporary logical unit assignment.<br>2. One JIB for each alternate logical unit assignment.<br>3. One JIB for each open 2311 extent with the DASD file protect feature.<br>4. Two JIBs for each open 2321 extent with the DASD file protect feature. |

| Name | Macro Description | Parameter=Option | Option Description |
|---|---|---|---|
| DVCGEN | Specifies I/O devices. Each device type requires a separate DVCGEN macro. (See note 1 for DVCGEN rules. This is an optional macro.) | CHUN = {X'cuu'} | Specify the hexadecimal number of the channel and unit for the device. |
| | | DVCTYP = {xxxxxx} | Specify the device type. See Figure 26. |
| | | CHANSW = { NO<br>YES } | Specify if the device is attached to more than one selector channel. If it is, the device can be switched. |
| | | MODE = {X'ss'} | Indicate the mode setting. (For seven track tapes only.) The assumed value for ss is 90. See Figure 27 for other values. |
| ASSGN | Sets standard background I/O assignments. A separate macro is required for each standard assignment desired. (Optional macro) CAUTION: The ASSGN macro allows SYSRDR, SYSLST, SYSPCH, and SYSIPT to be assigned to a tape or DASD. However, IPL unassigns any such assignments. | {SYSxxx, X'cuu'} | SYSxxx is any background symbolic logical unit (SYSIPT, SYSLOG, etc.) or programmer logical unit (SYS000, SYS001, etc.). X'cuu' is the hexadecimal number of the channel and unit to which the symbolic device is attached. |
| SEND | Indicates end of supervisor generation. | {n} | Specifies the beginning address of the problem program area. An area should be reserved for supervisor expansion and maintenance. The parameter is optional. If not specified, no area is reserved beyond the assembled last address of the supervisor. |

**Rules for Using DVCGEN**

1. A separate DVCGEN macro instruction is required for each device.
2. The total number of DVCGEN macros must not exceed the total number of devices specified in the IODEV parameter of the IOTAB macro.
3. DVCGEN macros must be specified in ascending channel address sequence.
4. Switchable units (attached to more than one selector channel) must be defined once. They are defined on the lowest channel on which they are addressable.
5. The sequence of the DVCGEN cards determines the priority of the devices on their channel. Switchable units must be the last devices for the channel.
6. The specifications of these macros may be altered by IPL ADD and DEL statements. See Section 4: Initial Program Load.

## Macro Relationships

The code generated by the assembler for any selected supervisor generation is a function of certain macros named in the preceding subsection (generation macros) and a group of inner macros called by the generation macros. The primary purpose of the generation macros is to set global values, based on parameter options, that can be tested by the inner macros. These macros then generate the bulk of the supervisor code. The specific instructions assembled depend on the global settings. Some of the generation macros also generate code; however, these can be treated as exceptions and are identified in this subsection.

The most important global values used in supervisor generation are the B-globals. Therefore, this subsection emphasizes the generation macros that establish B-global values. However, some A-globals are tested in the same manner as B-globals. These are also described in this subsection. A-globals that provide arithmetic values and all C-globals are not described. Two figures in this subsection show macro relationships. Figure 12 shows the code generated, if any, and the globals set, if any. Figure 13 indicates the on-off conditions of the globals.

| Macro | Type | Code Generated | Critical Globals Set |
|---|---|---|---|
| SUPVR | generation | Defines low main storage | BG0<br>BG20<br>BG21 |
| CONFG | generation | None | BG1<br>BG2<br>BG22<br>BG23 |
| STDJC | generation | None | BG34 |
| FOPT | generation | General cancel<br>General exit<br>General entry<br>Communications region | BG6<br>BG7<br>BG8<br>BG30<br>BG32<br>BG33<br>AG21<br>AG22<br>AG23 |
| PIOCS | generation | None directly calls inner macros | BG3<br>BG4<br>BG9<br>BG10<br>BG11<br>BG12<br>BG31 |
| SGTCH | inner | Channel scheduler<br>Start I/O<br>I/O interrupt | none |
| SGUNCK | inner | Unit check<br>Error recovery exits | none |
| SGDFCH | inner | Fetch subroutine | none |
| SGSVC | inner | Supervisor interrupts<br>Program check interrupts<br>External interrupts | none |
| SGDSK | inner | Disk error recovery | none |
| SGTCON | inner | SVEREG subroutine, VLDADR1 subroutine, ATNRTN routine, CCW chain, SVC interrupt table, logical transient save area, disk information blocks, error recovery block, PC option table, IT option table, and OC option table. | none |
| ALLOC | generation | None | none |
| IOTAB | generation | Supervisor table expansions - PIBs, channel queue table (CHANQ), LUB table, PUB table, TEBs, and JIBs | none |
| DVCGEN | generation | Overlays for PUB table entries | none |
| ASSGN | generation | Overlays for LUB table entries | none |
| SEND | generation | Defines end of supervisor nucleus, beginning of A and B transient areas, start of problem program area, BG save area. | none |

Figure 12. Macro Functions

| Global | Purpose | On Setting |
|---|---|---|
| BG0 | Determines whether the system is tape or disk. | SYSTEM = DISK |
| BG1 | Determines whether the storage protect feature is used. | SP = YES |
| BG2 | Determines whether the timer feature is used. | TIMER = YES |
| BG3 | Determines whether channel switching is supported (2816). | CHANSW = TSWTCH |
| BG4 | Determines if tape error statistics are to be accumulated and logged. | TEB = n |
| BG5 | Reserved | |
| BG6 | Determines if the asynchronous user interrupt key routine is supported. | OC = YES |
| BG7 | Determines whether the internal timer option is supported. | IT = F1, or F2, or BG |
| BG8 | Determines if the user program check routine is supported. | PC = YES |
| BG9 | Determines whether channel switching is supported (2404,2804). | CHANSW = RWTAU |
| BG10 | Indicate whether selector channels are supported. | SELCH = YES |
| BG11 | Indicates whether burst mode devices will be supported on the multiplexor channel. | BMPX = YES |
| BG12 | Determines the type of tape support required. | TAPE = 7 or 9 |
| BG20 | Determines whether multiprogramming support is required. | MPS = YES |
| BG21 | Determines whether Tele-processing support is required. | TP = BTAM |
| BG22 | Determines if the decimal feature is used. | DEC = YES |
| BG23 | Determines if the floating point feature is used. | FP = YES |
| BG30 | Determines if command chaining support for retry on I/O operations is used. | CCHAIN = YES |
| BG31 | Determines if 9 track tape support is required. | TAPE = 9 |
| BG32 | Determines whether the DASD file protect feature is supported. | DASDFP = n, n |
| BG33 | Determines if logical system I/O units are a 2311 disk pack. | SYSFIL = 2311 |
| BG34 | Determines the type of date configuration to be supported. | DATE = MDY |
| AG21 | Determines if a timer interrupt routine is for a BG program. | IT = BG |
| AG22 | Determines if a timer interrupt routine is for a F2 program. | IT = F2 |
| AG23 | Determines if a timer interrupt routine is for a F1 program. | IT = F1 |

Figure 13. Global Settings

## ORGANIZATION

The physical organization of the supervisor depends on the sequence of the supervisor generation macros. The sequence is predetermined and can not be changed by the user. The logical organization depends on the parameter options selected at generation time. Figure 14 describes the assembled supervisor by a main storage map. The map illustrates the supervisor physical organization in four major areas:

- Low Core
- Nucleus Code
- I/O Tables and Information Blocks
- Logical and Physical Transient Area

The logical organization is not described by this section of the manual. Because of the variety of options available, the reader must determine the logical organization for each individual supervisor generation. By using the program level flowcharts in Section 4 to point to the detailed flowcharts in Appendix H, the reader selects the correct group of flowcharts for the desired generation.

### Low Core

The main storage locations that make up low core can be classified as PSWs, CSWs, CAWs, and main storage areas. PSWs, CAWs, and CSWs are described in Figures 38, 39 and 40, respectively. The main storage areas include:

| Byte (hex) | Function |
|---|---|
| 0-4 | Message area when SYSLOG is disabled. |
| 0-1 | Contains the error code for the SEREP diagnostic program. |
| 16 | Contains the address of the communications region physically located within the nucleus code. |
| 50 | Contains the system timer used with microprogramming. |
| 54 | Contains system time of day set by job control and IPL, updated by the supervisor timer routine (optional). |
| 80 | Beginning of the diagnostic scan-out area. |

## Nucleus Code

The main storage map (Figure 14) illustrates the major routine and subroutine organization of the supervisor. Specific instructions are included or omitted depending on generation options. This manual describes the disk error recovery as the resident error recovery routine. The communications region is part of the nucleus coding that does not change from generation to generation. Figure 15 illustrates its structure. The starting address of the communications region is made available to a user in general register 1 through the COMRG macro.

## I/O Tables

The I/O tables that comprise this section of the supervisor are designed to establish the interface between a user's program and the hardware channels. Collectively, these tables are called the system control center (Figure 16). For every device used on the system there must be a PUB (Physical Unit Block). For every logical unit name (SYSXXX) used, there must be a LUB (Logical Unit Block). When an I/O request is made, and entry is made in CHANQ (the channel queue).

The entry contains a CCB (Channel Command Block) address which, in turn, points to a CCB that contains a code (LUB table index) for the logical unit name.

The supervisor processes the request when possible on the device assigned to the logical unit. If the TEB=YES option was selected at supervisor generation time, counts of tape errors are kept in TEBs (Tape Error Blocks).

To understand the interaction between the various I/O tables the reader should know the classification and sequence of the symbolic unit references (SYSXXX). The systems class (those symbolic unit names reserved for system use) is made up of:

1. SYSRDR
2. SYSIPT
3. SYSPCH
4. SYSLST
5. SYSLOG
6. SYSLNK
7. SYSRES
8. SYSSLB
9. SYSRLB
10. SYSUSE
11. SYSFGI

Foreground programs can use only the system unit names SYSLOG and SYSRES. The programmer class (those symbolic unit names reserved for programmer use) is made up of SYS000 to SYS243. This class is subdivided into these classifications:

1. Background logical unit class (minimum of 10).
2. Foreground two logical unit class (minimum of 5).
3. Foreground one logical unit class (minimum of 5).

PUBs are built at system generation or IPL time. LUBs are built at system generation time. PUBs are assigned to LUBs at system generation by the job control program, or by the foreground initiator. CHANQ and TEB entries are built and processed by the supervisor program. Figure 17 illustrates the I/O table interrelationships. Figures 18 through 21 are illustrations of individual I/O tables.

## Information Blocks and Other Tables

To accomplish functions such as exit selection, DASD file protection, and record identification, the supervisor program requires pertinent information. At supervisor generation time certain main storage locations are set aside and in some cases initialized to supply the required information. The basic information blocks and their respective functions are:

PIB (Program Information Block): Retains program status information for user and supervisor programs. Supplies routing information when in a multiprogramming environment to allow selective program return. Contains pointers and switches used by the supervisor program (Figure 22).

DIB (Disk Information Block): Built at generation time if the SYSFIL option was selected. Performs a recordkeeping function on system class units assigned to a DASD. The DIB contains the correct seek address when the system is operating in a batched job environment. The block is initialized by job control with extent information and updated by physical IOCS (Figure 23).

JIB (Job Information Block): Contains temporary and alternate LUB assignments. (These blocks are referenced by the LUBs.) The JIB serves another purpose when DASD file protection is selected as a supervisor generation option. Extent information is supplied by the program initiator and logical IOCS open transient routines. The supervisor can then perform the file protect function given the protected file limits. File protection does not include supervisor and transient originated I/O. (Figure 24).

OTHER TABLES: Three optional tables (program check, interval timer, and operator communications) containing user supplied address information are also found within the supervisor (Figure 25). Save areas for background programs and the logical transient programs are located within the supervisor. The background program save area is located just before the problem program area of main storage. The BG save area contains six subfields: program name, PSW, general registers (9 through 8), label length, 6 reserved bytes, and optionally floating point registers.

TRANSIENT AREAS: Main-storage locations are reserved for both the logical (B) and physical (A) transients. Approximately 1200 bytes are set aside as the logical transient area. Approximately 500 bytes are set aside as the physical transient area (Figure 14).

Figures 26 and 27 illustrate device type codes and tape density data (SS).

Note:
For PSW format see Figure 38.
For CSW format see Figure 40.
For CAW format see Figure 39.

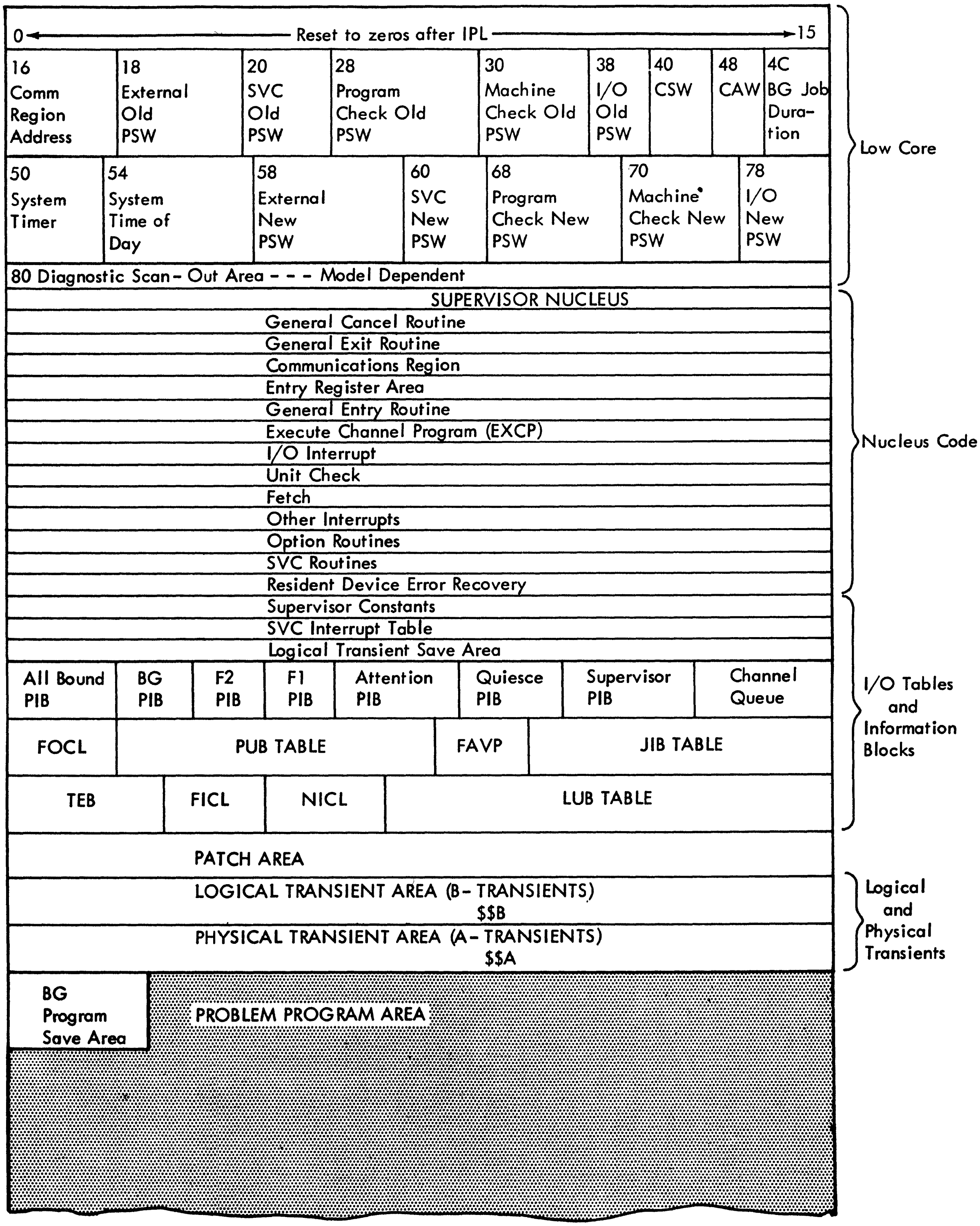



Figure 14. Supervisor Storage Allocation

COMREG

Note: Displacement values illustrated can be used to access the listing and/or the key that follows the figure. The key offers more detailed information about each area when necessary.

| Displacement hexadecimal | 0 | 8 | 0A | 0C | 17 | 18 | 20 | 24 |
|---|---|---|---|---|---|---|---|---|
| Displacement decimal | 0 | 8 | 10 | 12 | 23 | 24 | 32 | 36 |
| | Date | Addr of PPBEG | Addr of EOSSP | Problem Program Use | UPSI Byte | Job Name | Highest Storage Address | End Address of Last Phase Fetched or Loaded |
| | XXXXXXXX | XX | XX | XXXXXXXXXXX | X | XXXXXXXX | XXXX | XXXX |

| Displacement hexadecimal | 28 | 2C | 2E | 30 | 34 | 35 | 36 | 37 | 38 | 39 | 3A | 3B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Displacement decimal | 40 | 44 | 46 | 48 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| | End Address of Longest Phase Fetched or Loaded | Lbl Area Lng | PIK | End of Storage Address | Confg Byte | Date Conv | Standard Options | | Job Control Byte | Linkage Control Byte | Language Translator Control Byte | Job Duration Indicator Byte |
| | XXXX | XX | XX | XXXX | X | X | XX | | X | X | X | X |
| | | | | | | | Job Control Switches | | | | | |

| Displacement hexadecimal | 3C | 3E | 40 | 42 | 44 | 46 | 48 | 4A | 4C | 4E | 4F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Displacement decimal | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 79 |
| | Disk Address of Labels | Addr of FOCL | Addr of PUB | Addr of FAVP | Addr of JIB | Addr of TEB | Addr of FICL | Addr of NICL | Addr of LUB | Line Count for SYSLST | System Date |
| | XX | XX | XX | XX | XX | XX | XX | XX | XX | X | XXXXXXXXX |

| Displacement hexadecimal | 58 | 59 | 5A | 5C | 5E | 60 | 62 | 64 |
|---|---|---|---|---|---|---|---|---|
| Displacement decimal | 88 | 89 | 90 | 92 | 94 | 96 | 98 | 100 |
| | Reserved | LIOCS Byte | PIB Table Addr | ID Number of Last Checkpoint | Lng. of LUB ID Queue = No. of Channel Queue Entries | Address of Disk I/O Position Data | Address of Channel Scheduler Error Block | Address of PC Option Table - 8 |
| | X | X | XX | XX | XX | XX | XX | XX |

| Displacement hexadecimal | 66 | 68 | 6A | 6C | 6E |
|---|---|---|---|---|---|
| Displacement decimal | 102 | 104 | 106 | 108 | 110 |
| | Address of IT Option Table - 8 * IT Support Key | Address of OC Option Table - 8 | Key to Program with Timer Support | Address of the LUBID Queue | Logical Transient Key |
| | XX | XX | XX | XX | XX |

Figure 15. Supervisor Communications Region (Part 1 of 5)

Key to Communications Region Displacements:

| | |
|---|---|
| 0 | MM/DD/YY or DD/MM/YY obtained from the job control date statement. Format controlled by COMREG +53 (date convention byte) bit 0. |
| 8 | Address of the problem program label area. (End of transient area + 1). |
| 10 | Address of the beginning of the problem program area. Y (EOSSP) = Y (PPBEG) if the storage protection option has not been selected. Y (EOSSP) equals the first main storage location with a storage protection key of 1, if storage protection is supported. |
| 12 | User area. |
| 23 | User program switch indicator. |
| 24 | Job name set by the job control program from information found in the job statement. |
| 32 | Address of the uppermost byte of the problem program area as determined by the IPL program. (Clear storage routine determines the address, ENDRD routine of $$A$IPL2 stores it.) |
| 36 | Address of the uppermost byte of the last phase of the problem program fetched or loaded. The initial value (as shown) is overlaid by the first fetch or load to the problem program area. |
| 40 | Address of the uppermost byte of the longest phase of the problem program fetched or loaded. The initial value is overlaid by the first fetch or load to the problem program area. |
| 44 | Length of the problem program label area. |

46 Program Interrupt Key: Value is equal to the displacement from the start of the PIB table to the PIB for the task.

First Byte - always zero.

Second Byte - Contains the key of the program that was last enabled for interrupts. (When an interrupt occurs, the PIK indicates to the supervisor which program was interrupted.)

| Task | PIK Value |
|---|---|
| *All Bound | X'00' |
| BG | X'10' |
| *F2 | X'20' |
| *F1 | X'30' |
| Attn Rtn | X'40' |
| Quiesce I/O | X'50' |
| Supervisor | X'60' |

* These tasks do not exist in a batch-job-only system. (See Appendix H.)

48 Logical end of main storage address.

52 Configuration Byte (Values set at supervisor generation time.)

Bit 0: 1 = Storage protect
0 = No storage protect
1: 1 = Decimal feature
0 = No decimal feature
2: 1 = Floating-point feature
0 = No floating-point feature
3: Reserved
4: 1 = Timer feature
0 = No timer feature
5: 1 = Channel switching device
0 = No channel switching device
6: 1 = Burst mode on multiplex channel support
0 = No burst mode on multiplex support
7: 1 = 7-track SYSRES
0 = No 7-track SYSRES

Figure 15. Supervisor Communications Region (Part 2 of 5)

Key to Communications Region Displacements:

**53** Date Convention Byte

Bit 0: 1 = DDMMYYJJ (Set at generation time by STDJC)
0 = MMDDYYJJ
1: 1 = Multiprogramming environment
0 = Batch job environment
2: 1 = DASD file-protect supported
0 = No file-protect support for DASD
3: 1 = DASD SYSIN- SYSOUT
0 = No DASD SYSIN- SYSOUT
4: 1 = BTAM = YES
0 = BTAM = NO
5- 7: Reserved

**54** This byte contains the standard language translator I/O options (set by the STDJC macro).

| | | |
|---|---|---|
| Bit 0: | DECK option | 1 = yes, output object modules on SYSPCH |
| 1: | LIST option | 1 = yes, output source module listings and diagnostics on SYSLST |
| 2: | LISTX option | 1 = yes, output hexadecimal object module listings on SYSLST (compilers only) |
| 3: | SYM option | 1 = yes, output symbol tables on SYSLST/SYSPCH |
| 4: | XREF option | 1 = yes, output symbolic cross reference list on SYSLST |
| 5: | ERRS option | 1 = yes, output diagnostics on SYSLST (compilers only) |
| 6: | CHARSET option | 1 = 48, input on SYSIPT is 48 or 60 character set |
| 7: | Reserved | |

**55** This byte contains the standard supervisor options for abnormal EOJ and control statement display.

| | | |
|---|---|---|
| Bit 0: | Not used | |
| 1: | DUMP option | 1 = yes, dump registers and storage on SYSLST |
| 2: | Not used | |
| 3: | LOG option | 1 = yes, list all control statements on SYSLST |
| 4- 6: | Not used | |
| 7: | Reserved | |

**56** Job control byte (JBCSW0)

Bit 0: Reserved
1: 1 = Return to caller on LIOCS disk open failure
0 = Do not return to caller on LIOCS disk open failure
2: 1 = Job control input from SYSRDR
0 = Job control input from SYSLOG
3: 1 = Job control output on SYSLOG
0 = Job control output not on SYSLOG
4: 1 = Cancel job
0 = Do not cancel job
5: 1 = Pause at end-of-job step
0 = No pause at end-of-job step
6: 1 = SYSLOG is not a 1052
0 = SYSLOG is a 1052
7: 1 = SYSLOG is assigned to the same device as SYSLST
0 = SYSLOG is not assigned to the same device as SYSLST

Figure 15. Supervisor Communications Region (Part 3 of 5)

57 Linkage control byte

Bit 0: 1 = SYSLNK open for output
0 = SYSLNK not open for output
1: Reserved
2: 1 = Allow EXEC
0 = Suppress EXEC
3: 1 = Catalog linkage editor output
0 = Do not catalog linkage editor output
4: 1 = Supervisor has been updated
0 = Supervisor has not been updated
5: 1 = Executing in AUTOTEST mode
0 = Not executing in AUTOTEST mode
6: 1 = Ignore attention interrupt on 1052
0 = Do not ignore 1052 attention interrupt
7: 1 = Fetch MAINEOJ at end of job to update system directory
0 = Do not fetch MAINEOJ at end of job for update

58 Language processor control byte. This is a set of switches used to specify nonstandard language translator options. The switches within the byte are controlled by job control OPTION statements and when set to 1, override standard options. The format of this byte is identical to the standard option byte (displacement 54).

59 Job duration indicator byte

Bit 0: 1 = Within a job condition
0 = Outside a job condition
1: 1 = Dump on an abnormal end-of-job condition
0 = No dump on abnormal EOJ
2: Reserved
3: 1 = Job control output on SYSLST
0 = Output not on SYSLST
4: 1 = Job is being run out of sequence with a temporary assignment for SYSRDR
0 = Conditions for 1 setting not met
5: 1 = No OPEN
0 = Initial entry to OPEN
6: 1 = OPEN monitor entry is from the DTFCP OPEN phase
0 = Conditions for a 1 setting not met
7: Reserved

60 Binary disk address of the volume label area.

62 – 76 As illustrated (Figures for information blocks, I/O tables, and pointers are found in this Section of this manual, beginning at Figure 17, which refers to more detailed figures).

78 Set to the value nn specified in the LINES = nn parameter of the STDJC macro.

79 The format of the system date contained within this field is determined by the IPL program from information supplied in the date convention byte (displacement 53). Bytes 85 - 87 contain the day count.

88 Reserved.

89 Byte reserved for use by LIOCS. Transient dump programs insert a key to indicate to the LIOCS end-of-volume routine, $$BCMT07, that it was called by a B-transient.

90 Address of the program information block (PIB) table. (See Figure 22.)

92 ID number of the last checkpoint.

94 Length of the LUBID queue (in bytes). This equals the number of channel queue entries. It can also be used to access the REQID queue. (See Figure 21.)

Figure 15. Supervisor Communications Region (Part 4 of 5)

Key to Communications Region Displacements:

| | |
|---|---|
| 96 | Address of disk I/O position data. This is the starting address of the disk information block (DIB) table. |
| 98 | Address of the beginning of the error recovery block. The error recovery block contains addresses of error recovery exits, error recovery queue information that can be used by physical transient routines, and defines storage for the error queue entries. |
| 100 → 104 | As illustrated (See Figure 25). |
| 106 | Key of the program (BG, F2, or F1) that has timer support. |
| 108 | As illustrated (See Figure 21). |
| 110 | Logical Transient Key (LTK) contains the same value as the PIK (displacement 46) when the logical transient is requested. When the transient area is not in use, LTK is equal to zero. The SVC 2 routine sets the LTK. The SVC 11 routine resets the LTK (See Appendix H). |

Figure 15. Supervisor Communications Region (Part 5 of 5)

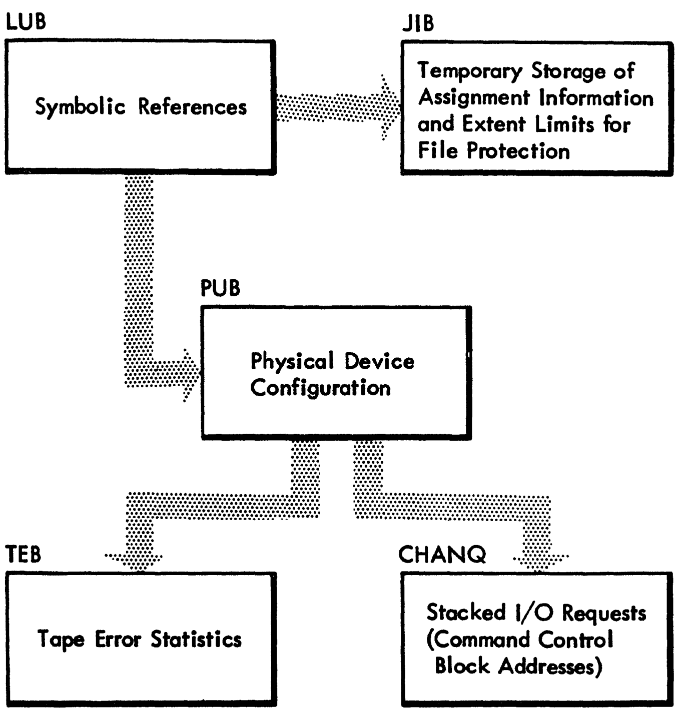
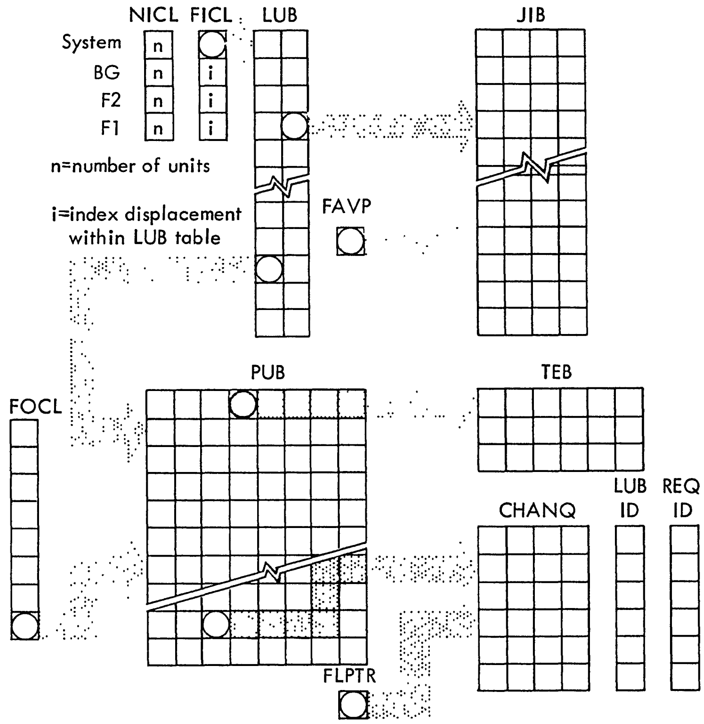



Figure 16. System Control Center

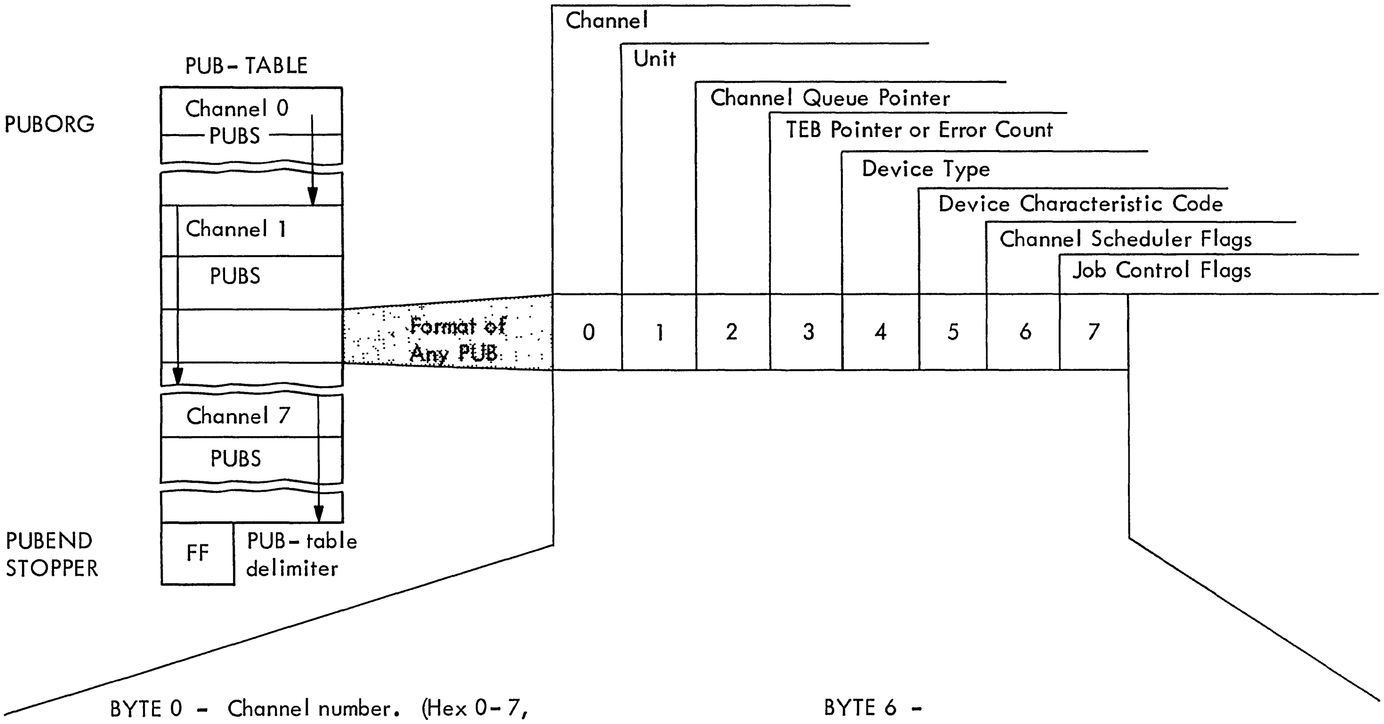



CAUTION: The term pointer is frequently used in this section. It is defined as a one-byte displacement index value. Base addresses of all tables are maintained in the communications region except for the channel queue. For other sections of this manual, a pointer is defined as an address.

KEY:

| | |
|---|---|
| NICL (Number in Class) | : The first byte contains the number of system class units. The second, third, and fourth bytes contain the number of programmer class units. (BG,F2,F1) |
| FICL (First in Class) | : The first byte points to the first system class unit in the LUB table. (Always the first LUB table entry.) The second byte points to the first programmer class unit in the LUB table BG area. The third points to the first programmer class unit in the LUB table F2 area. The fourth points to the first programmer class unit in the LUB table F1 area. |
| LUB (Logical Unit Block) Table | : The first byte points to a PUB table entry. The second byte points to a JIB table entry (Figure 19). |
| PUB (Physical Unit Block) Table | : The first two bytes contain the channel and unit address of the physical device. The third byte contains a CHANQ pointer. The fourth byte contains a TEB pointer. The fifth byte contains device type code information. The sixth byte is reserved for device options. The seventh and eighth bytes contain channel scheduler and job control flags respectively (Figure 18). |
| FOCL (First on Channel List) | : The first byte points to the first PUB (highest priority) on channel zero. The next byte points to the first PUB (highest priority) on channel one, etc. A hexadecimal FF indicates the associated channel is not supported. |
| TEB (Tape Error Block) | : One TEB is built for each tape unit at supervisor generation time if tape error statistics are required (Figure 20). |
| FAVP (First Available Pointer) | : A one-byte pointer to the next available JIB entry. |
| JIB (Job Information Block) | : The first two bytes contain extent or LUB information. The third contains ownership and JIB flags. The fourth contains JIB chaining information (Figure 24). |
| CHANQ (Channel Queue) Table | : The first byte contains the chain field (a pointer to the next in queue). The last three bytes contain the CCB address (Figure 21). |
| LUBID | : A one-byte pointer to the LUB making the I/O request. |
| REQID | : A one-byte pointer to the program containing the CCB (Figure 21). |
| FLPTR | : A one-byte pointer to the next free entry in the channel queue (Figure 21). |

Figure 17. I/O Table Interrelationship




BYTE 0 - Channel number. (Hex 0-7, FF = NULL)

BYTE 1 - I/O device unit number. (HEX 1F = 1052, HEX 80 = magnetic tape unit 0...).

BYTE 2 - HEX 0, 1, 2, ...points to the first channel queue entry for this device.

BYTE 3 - If device is a magnetic tape unit and TEBS are specified, this byte is a TEB pointer (HEX 1, 2, 3...).
If device is a magnetic tape unit but TEBS are not specified, this byte is an error counter.
If device is not a magnetic tape unit, this byte is an error counter.

BYTE 4 - See Figure 26 for device type codes.

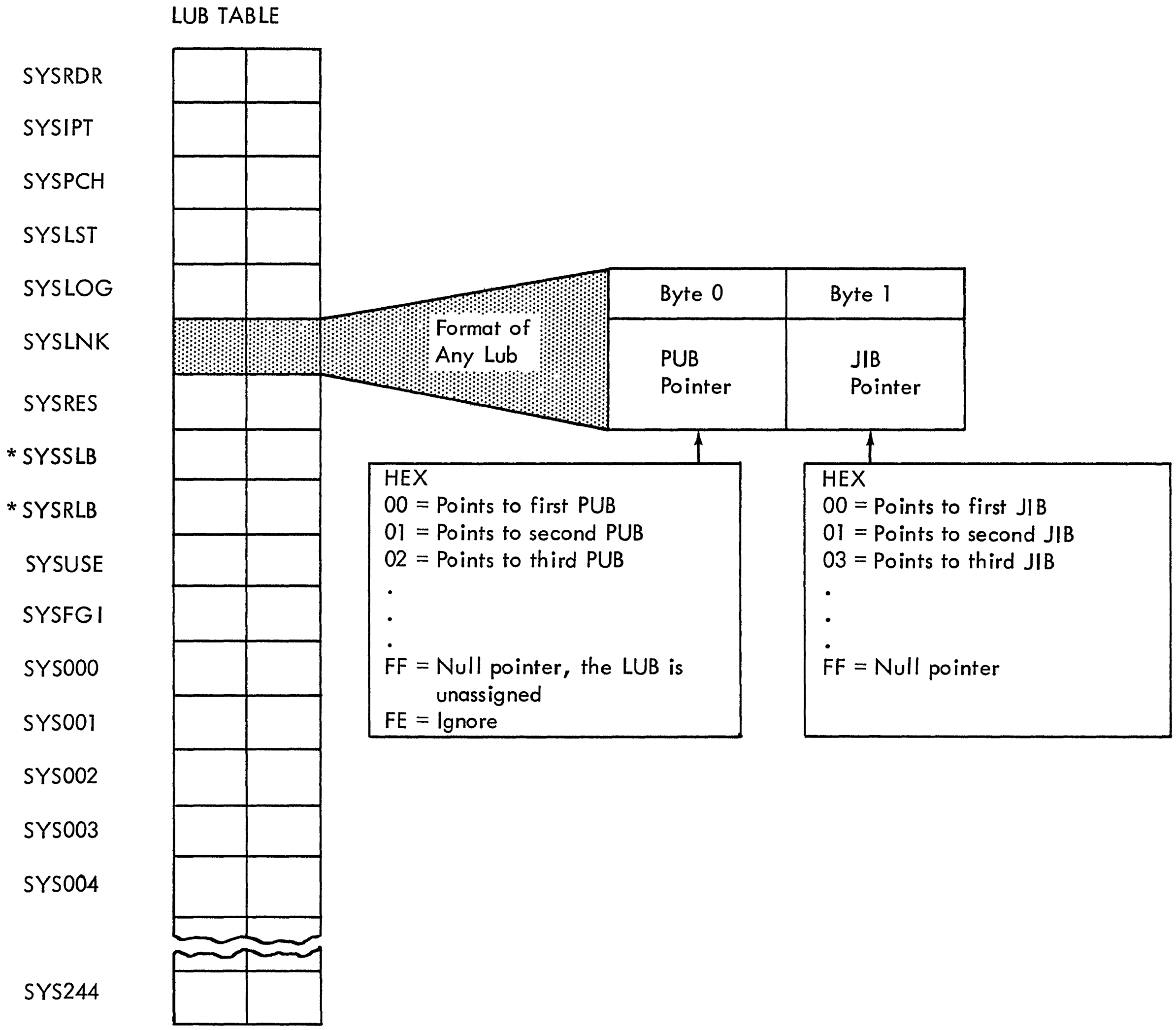
BYTE 5 - SS of the MODE = parameter in the DVCGEN macro for a tape unit. (See Figure 27.)

BYTE 6 -
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 = Device end received
6: 1 = Burst device on MPX
7: 1 = 7-track tape unit

BYTE 7 -
Bit
0-4: standard MODE assignment for 7-track tape (all ones if not tape, all zeros if device is down).
5: device is assigned to a background job
6: device is assigned to a foreground 1 job
7: device is assigned to a foreground 2 job

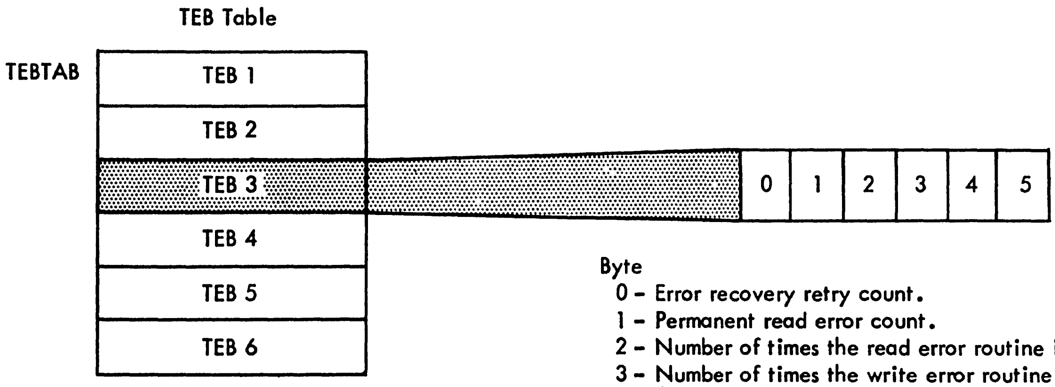
NOTE:
A null PUB is generated for each device to be supported by the supervisor. (See IOTAB macro in this section.)
Standard physical unit assignments are made to the PUB table at supervisor generation time. PUBS are ordered by channel and priority within a channel. (See DVCGEN macro in this section.)

Figure 18. PUB Table




* These LUBs are not used or supported in DOS, Version 2. They may be used in Tape Systems (TOS) to refer to magnetic tape devices on which private source statement and relocatable library tapes are mounted. In DOS, these libraries are on SYSRES.

Figure 19. LUB Table



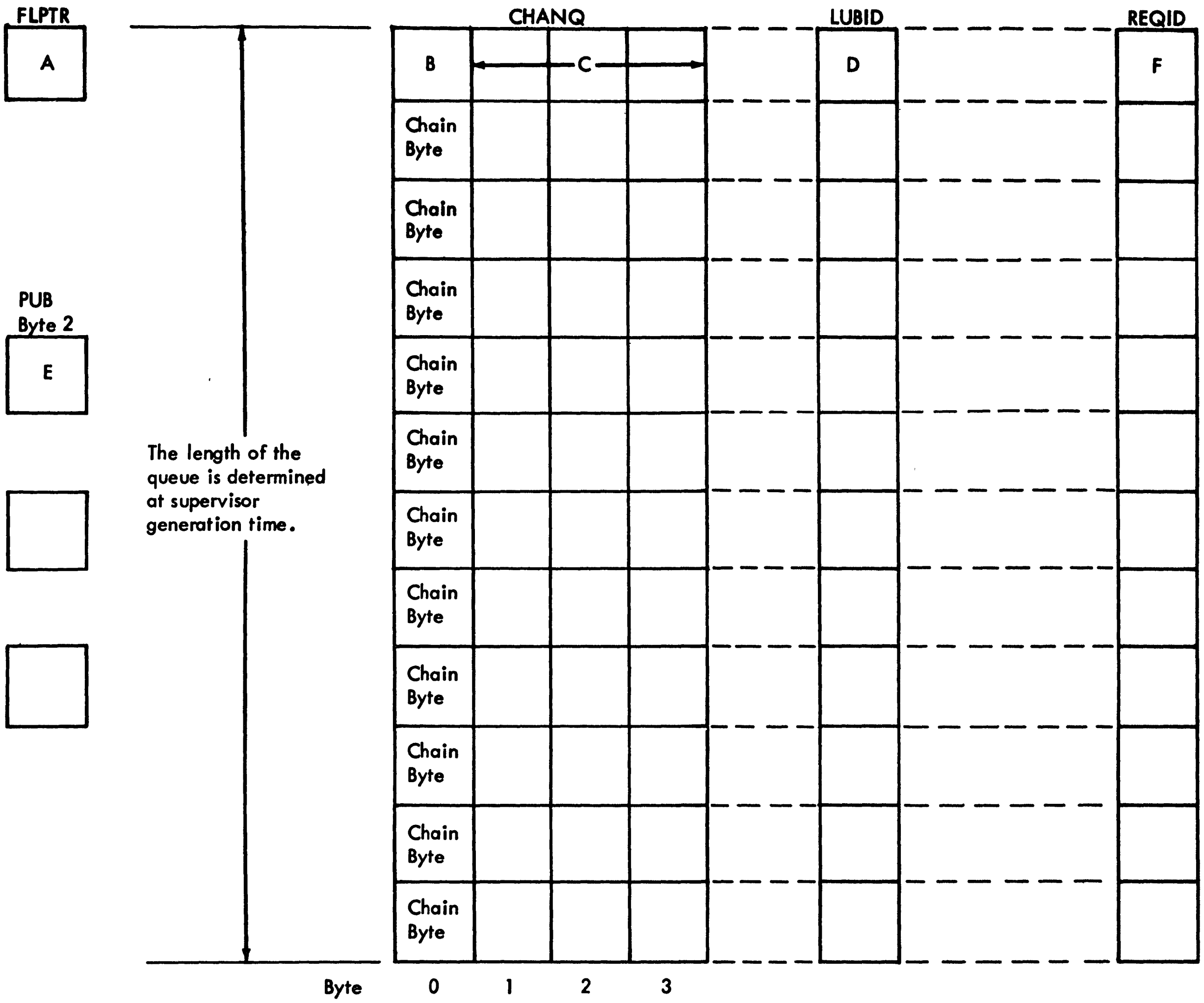

Byte
0 - Error recovery retry count.
1 - Permanent read error count.
2 - Number of times the read error routine is entered.
3 - Number of times the write error routine is entered.
4 - Write skip (erase gap) count.
5 - Noise record count.

One TEB is generated for each magnetic tape unit if the CONFG macro contains the TEB = yes parameter.

Job Control resets each TEB at normal or abnormal End-of-Job. An unused TEB contains HEX 'FF0000000000'.

A TEB is referenced from byte 3 of a magnetic tape unit PUB.

Figure 20. Tape Error Block (TEB)




KEY:

A. The free list pointer contains a displacement index to a free list entry within the channel queue. The free list is a group of entries that function in essentially the same manner as a device queue. When the free list pointer contains a hexadecimal FF, it indicates that no more free list entries are available.

B. The first byte of the channel queue entry (chain byte) contains a pointer (displacement index) to the next channel queue entry for that device. A hexadecimal FF indicates the last channel queue entry for that device. New requests on a given device are queued at the end of a given device queue.

C. CCB address for the specified device.

D. A pointer (displacement index) to the LUB table identifying the logical unit making the I/O request.

E. Contains a pointer (displacement index) to the first channel queue entry for a specific device (Figure 18).

F. Contains a code identifying the program making the I/O request. The one-byte entry is called a RID (Requestor Identification). The RID indicates what program the CCB belongs to. The RID is in the form X'nk'.

n = user-storage protection key.
k = 0 for all user requests and all supervisor CCBs, where n = 0.
k = 1 for supervisor CCBs to SYSLOG that bypass ID prefix.
k = 2 for a fetch CCB.
nk = FF for any unused channel queue entries.

Figure 21. CHANQ, LUBID, REQID Tables

PIB TABLE

| Byte Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All Bound PIB | Flag Byte See A | Reserved | SP Prefix | | Branch Instruction to the All Bound Routine | | | | Reserved | | | | | | | |
| Problem Program PIB * Caution | Flag Byte See B | Cancel Code | SYSLOG ID | | NOP Instruction (CR) | Address of the Save Area | | | Number of Core Blocks | Address of the Origin Area | | | PIB Assign Flag See D | User LUB Index | Number of Program LUBs | Reserved |
| Attention PIB | Flag Byte See E | Cancel Code | SYSLOG ID | | Branch Code (BC) | Active = Address of Save Area Inactive = Remainder of BC Instruction | | | Switch Byte See C | Logical Transient Bucket (contains save area address) | | | X'07' See D | Reserved | Address of the Logical Transient | |
| Quiesce PIB | Flag Byte See A | Cancel Code | Reserved | | Branch Instruction | | | | Reserved | | | | | | | |
| Supervisor PIB | Flag Byte See A | Cancel Code | SP Prefix | | Branch Instruction | | | | Address of Resident PUB | | Reserved | | X'07' See D | Reserved | | |

=16 Byte Length

*CAUTION: The PIB table is built in this sequence when the MPS feature is selected as a generation option:

All Bound PIB
Background PIB
Foreground 2 PIB
Foreground 1 PIB
Attention PIB
Quiesce PIB
Supervisor PIB

When a batch-only environment is established at generation time, the All Bound and Foreground PIBs are excluded from the table.

Figure 22. PIB Table (Part 1 of 2)

A Supervisor, Quiesce, PIB Flag Expansion:

Bit 0: 1 = Registers stored
0 = Registers not stored
1-4: 0 = Always zero
5: 1 = Always one
6: 0 = Always zero
7: 1 = Active
0 = Inactive

B Problem Program PIB Flag Expansion (BG, F2, F1):

Bit 0: 1 = Registers stored
0 = Registers not stored
1-4: 0 = Always zero
5: 0 = Normal execution
1 = Program has seized the system
6: 1 = Unbound
0 = SVC 2-bound (B-transient in progress)
7: 1 = Unbound
0 = SVC 7-bound (waiting for an I/O interrupt)
X'80' indicates the program is not present in the system.

C Attention PIB Switch Byte Expansion:

Bit 0-4: Reserved
5: 1 = Physical Attention Recall Switch ON
0 = Physical Attention Recall Switch OFF
6: 1 = Attention Request Switch ON
0 = Attention Request Switch OFF
7: 1 = External Interrupt Request Switch ON
0 = External Interrupt Request Switch OFF

D PIB Assign Flag Expansion:

X'80' = SYSRES DASD file protect inhibited (allow write operation on SYSRES)
X'40' = Channel appendage exit allowed (BTAM)
X'20' = Cancel in progress (used in terminator function)
X'10' = Cancel control (set on a foreground cancel)
X'08' = Hold-Release flag for foreground assignments
X'07' = Supervisor or Attention routine PIB assign flag setting
X'04' = Background program PIB assign flag setting
X'02' = Foreground 1 program PIB assign flag setting
X'01' = Foreground 2 program PIB assign flag setting

E Attention PIB Flag Expansion:

Bit 0: 1 = Registers stored
0 = Registers not stored
1-5: 0 = Always zero
6: 1 = Attention routine active
0 = Attention routine SVC 2-bound
7: 1 = Active
0 = SVC 7-bound

X'80' indicates the attention routine is not present in the system.

Figure 22. PIB Table (Part 2 of 2)

**DIB TABLE**

| | Current Address | | | | | | | | | | End Address | | | | | | | U.L. | L.L. | R.C. | | Reserved | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYSLNK | C | C | H | H | R | ← Packed Data → | | | | | | | | | | | | | | | | | | |
| SYSIN | B | B | C | C | H | H | R | K | D | D | B | B | C | C | H | H | R | H | H | XX | XX | | | |
| SYSPCH | | | | | | | | | | | | | | | | | | | | | | | | |
| SYSLST | | | | | | | | | | | | | | | | | | | | | | | | |
| Number of Bytes | ← 7 → | | | | | | | ← 3 → | | | ← 6 → | | | | | | 1 | 1 | 1 | ← 2 → | | ← 3 → | | |

KEY: Current Address: The next address to be used (for both input and output).
End Address : The last address within the limits of the extent.
U.L. : Upper head limit
L.L. : Lower head limit
R.C. : Trigger value in number of records. When the end address minus the current address is less than, or equal to, the value in R.C., a warning message is issued by job control.

KDD for SYSIN = X'000050'
KDD for SYSPCH = X'000051'
KDD for SYSLST = X'000079'

NOTE: The address of the SYSLNK entry is kept in the communications region.

Figure 23. DIB Table

JIB Table

| JIB Table |
|---|
| JIB 1 |
| JIB 2 |
| JIB 3 |
| JIB 4 |
| JIB 5 |
| JIB 6 |

Number (length of JIB table) determined at supervisor generation

Caution: Two JIBs are required for a 2321 extent; one for lower limit and one for upper limit. The lower limit defining JIB must be chained to the upper limit defining JIB. Byte 1 of this type JIB contains the sub-cell number times 10 plus the strip number in binary.

| 0 | 1 | 2 | 3 |
|---|---|---|---|

| Type of Entry | Contents |
|---|---|
| Stored standard assignment | LUB entry of standard assignment |
| Alternate assignment | PUB pointer for alternate assignment |
| 2311 Extent (1) | $C_L C_L C_H C_H$ (2) |
| 2321 Extent (1) | $B_L B_L C_L C_L$ or $B_H B_H C_H C_H$ (3) |

(1) Only when file-protect on DASD

(2) Lower Cylinder<br>Upper Cylinder

(3) Cell or combined sub-cell and strip

| Flag Type | Bit | Meaning if Bit=1 |
|---|---|---|
| Contents | 0 | Stored standard assignment |
| | 1 | Alternate assignment |
| | 2 | 2311 Extent |
| | 3 | 2321 Extent |
| Ownership | 4 | Standard assignment |
| | 5 | Background |
| | 6 | Foreground 1 |
| | 7 | Foreground 2 |

Chain Byte.
Contains the displacement index of the next JIB. A hexadecimal 'FF' defines the end of the chain.

● Figure 24. JIB Table

PC Option Table and OC Option Table:

| | A | B |
|---|---|---|
| BG | a | b |
| F2 | a | b |
| F1 | a | b |
| | word | word |

**A**

No STXIT given: a = 0
STXIT issued: a = address of the user program check (operator communications) routine
STXIT issued when the user routine is already in use: a = complement of user program check (operator communications) routine address

**B**

No STXIT given: b = 0
STXIT issued: address of the user save area

IT Option Table:

| C | D |
|---|---|
| c | d |
| word | word |

**C**

No TECB or STXIT issued: c = 0
TECB issued: c = address of the timer event control block
STXIT issued: c = address of the user interval timer routine
STXIT issued when user routine is already in use: c = complement of the user interval time routine address

**D**

No TECB or STXIT issued: d = 0
TECB issued: d = complement of the TECB address
STXIT issued: d = address of the user save area

Figure 25. Option Tables

| Card Code | Actual Device | Dev. Type X'nn' | Device Type |
|---|---|---|---|
| 2400T9 | 9-track Magnetic Tapes | 50 | Tapes |
| 2400T7 | 7-track Magnetic Tapes | 50 | |
| 1442N1 | 1442N1 Card Reader Punch | 30 | Card readers-punches |
| 2520B1 | 2520B1 Card Reader Punch | 31 | |
| 2501 | 2501 Card Reader | 10 | Card readers |
| 2540R | 2540 Card Reader | 11 | |
| 2540P | 2540 Card Punch | 21 | Card punches |
| 2520B2 | 2520B2 Card Punch | 20 | |
| 1442N2 | 1442N2 Card Punch | 22 | |
| 2520B3 | 2520B3 Card Punch | 20 | |
| 1403 | 1403 Printer | 40 | Printers |
| 1403U | 1403 Printer with UCS feature | 42 | |
| 1404 | 1404 Printer | 40 | |
| 1443 | 1443 Printer | 41 | |
| 1445 | 1445 Printer | 41 | |
| 1050A | 1052 Printer-Keyboard | 00 | |
| UNSP | Unsupported device | FF | Unsupported. No burst mode on multiplexor channel |
| UNSPB | Unsupported device | FF | Unsupported with burst mode on multiplexor channel |
| 2311 | 2311 Disk Drive | 60 | DASD |
| 2321 | 2321 Data Cell Drive | 61 | DASD |
| 2701 | 2701 Line Adaptor Unit | D0 | Tele-processing lines |
| 2702 {A, B, C, D} | 2702 Transmission Control Unit | D1 | A = SAD 0 command when enabling the line<br>B = SAD 1 command when enabling the line<br>C = SAD 2 command when enabling the line<br>D = SAD 3 command when enabling the line |
| 2703 | 2703 Transmission Control Unit | D2 | |
| 2671 | 2671 Paper Tape Reader | 70 | |
| 1285 | 1285 Optical Reader | 76 | |

Note: The codes used in the DVCGEN macros are the same codes used in IPL statements.

• Figure 26. Device Type Codes

| Density (Bytes per Inch) | Parity | Convert Feature | Translate | ss |
|---|---|---|---|---|
| 200 | odd | on | off | 10 |
| 200 | odd | off | off | 30 |
| 200 | odd | off | on | 38 |
| 200 | even | off | off | 20 |
| 200 | even | off | on | 28 |
| 556 | odd | on | off | 50 |
| 556 | odd | off | off | 70 |
| 556 | odd | off | on | 78 |
| 556 | even | off | off | 60 |
| 556 | even | off | on | 68 |
| 800 | odd | on | off | 90 |
| 800 | odd | off | off | B0 |
| 800 | odd | off | on | B8 |
| 800 | even | off | off | A0 |
| 800 | even | off | on | A8 |
| 800 | dual density nine-track | | | C8 |
| 1600 | dual density nine-track | | | C0 |

Figure 27. Density Data

This section presents the three major control programs that control the disk operating system (DOS) Version 2:

1. Initial program load program (IPL).

2. Job control program ($JOBCTLA-$JOBCTLJ).

3. Supervisor control program.

These programs allow operating system capability by providing the necessary interface between the IBM System/360, its supporting I/O devices, the operator, system residence, and the problem program.

## INITIAL PROGRAM LOAD PROGRAM (IPL), CHART 01

IPL is a 3-phase program consisting of:

- $$A$IPL1 (24-byte bootstrap routine).
- $$A$IPLA (32-byte bootstrap routine).
- $$A$IPL2 (less than 4096-byte IPL routine).

$$A$IPL1 is a 24-byte bootstrap program located on SYSRES at 00 00 1 (CC HH R). The operator sets the channel and unit of SYSRES in the load unit switches and presses the load key. Microprogramming reads the first record (24 bytes) from SYSRES into main storage starting at location 00. This 24-byte record consists of a PSW starting at location 0 and two CCWs starting at location 8. Microprogramming executes the first CCW at location 8, which reads in the $$A$IPLA program from SYSRES (cylinder 0, track 0, record 2). The first CCW is chained to the second CCW, which is a seek for the $$A$IPL2 program on SYSRES (cylinder 0, track 01, record 5).

$$A$IPLA is a 32-byte program made up of three CCWs. The first two CCWs of the $$A$IPLA program and the two CCWs from $$A$IPL1 are chained together, so that all the CCWs are executed. The CCWs of the $$A$IPLA program are a search, transfer in channel, and read for cylinder 0, track 01, record 5 to load the $$A$IPL2 program. Control is transferred to the $$A$IPL2 program by loading the PSW at location 0. This PSW was loaded as part of $$A$IPL1.

$$A$IPL2 clears storage from the end of itself to the end of main storage. A program check is forced and the program check new PSW returns control to the $$A$IPL2 program. The address at which the program check occurred is saved as the end of storage address. There is no provision in the $$A$IPL2 program to clear main storage below location 12,228.

The transient directory is searched for the core image library disk address of the supervisor. The supervisor is read into main storage starting at location 00. (See Figure 30 for a map of main storage.) The I/O tables that are located within the supervisor are moved to the end of supervisor so that a 2-device system can be built in low storage for the IPL operation.

See Figures 28 and 29 for examples of I/O tables built by $$A$IPL2. Figure 28 shows the I/O tables for a 1-device system, and Figure 29 shows the I/O tables for a 2-device system.

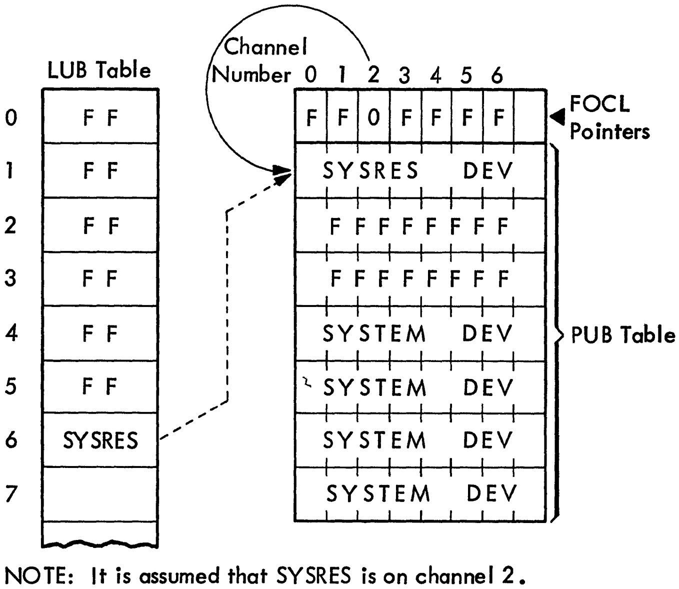



Figure 28. I/O Table for One-Device System

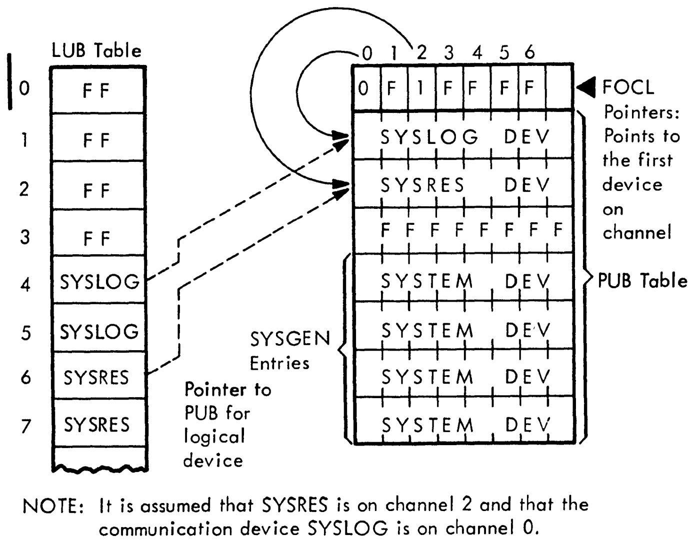
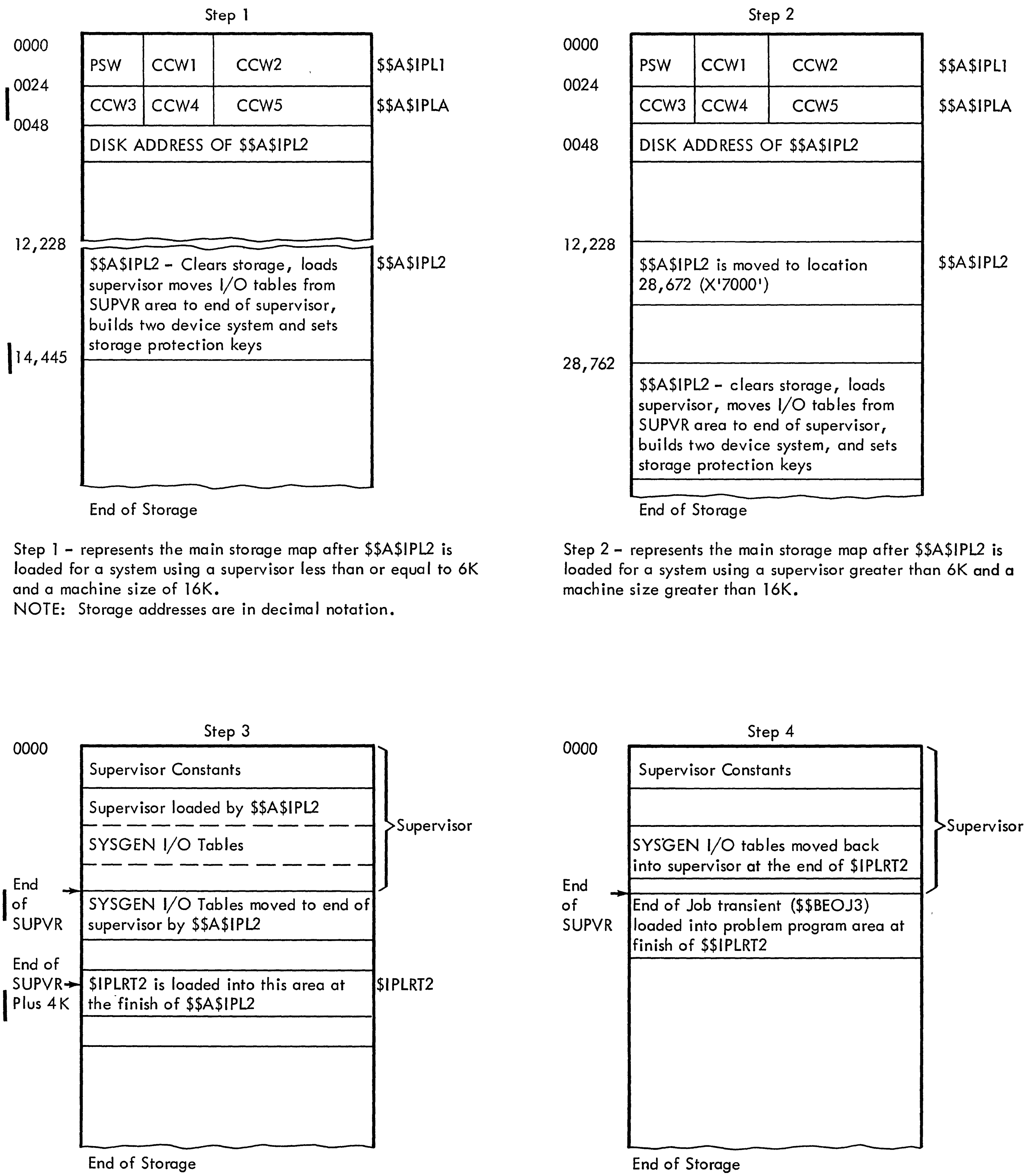



Figure 29. I/O Table for Two-Device System




Step 1 - represents the main storage map after $$A$IPL2 is loaded for a system using a supervisor less than or equal to 6K and a machine size of 16K.
NOTE: Storage addresses are in decimal notation.

Step 2 - represents the main storage map after $$A$IPL2 is loaded for a system using a supervisor greater than 6K and a machine size greater than 16K.

Step 3 - represents the main storage map after $$A$IPL2 loads the supervisor.

Step 4 - represents the main storage map after $$IPLRT2 is executed.

Figure 30. IPL Main Storage Map

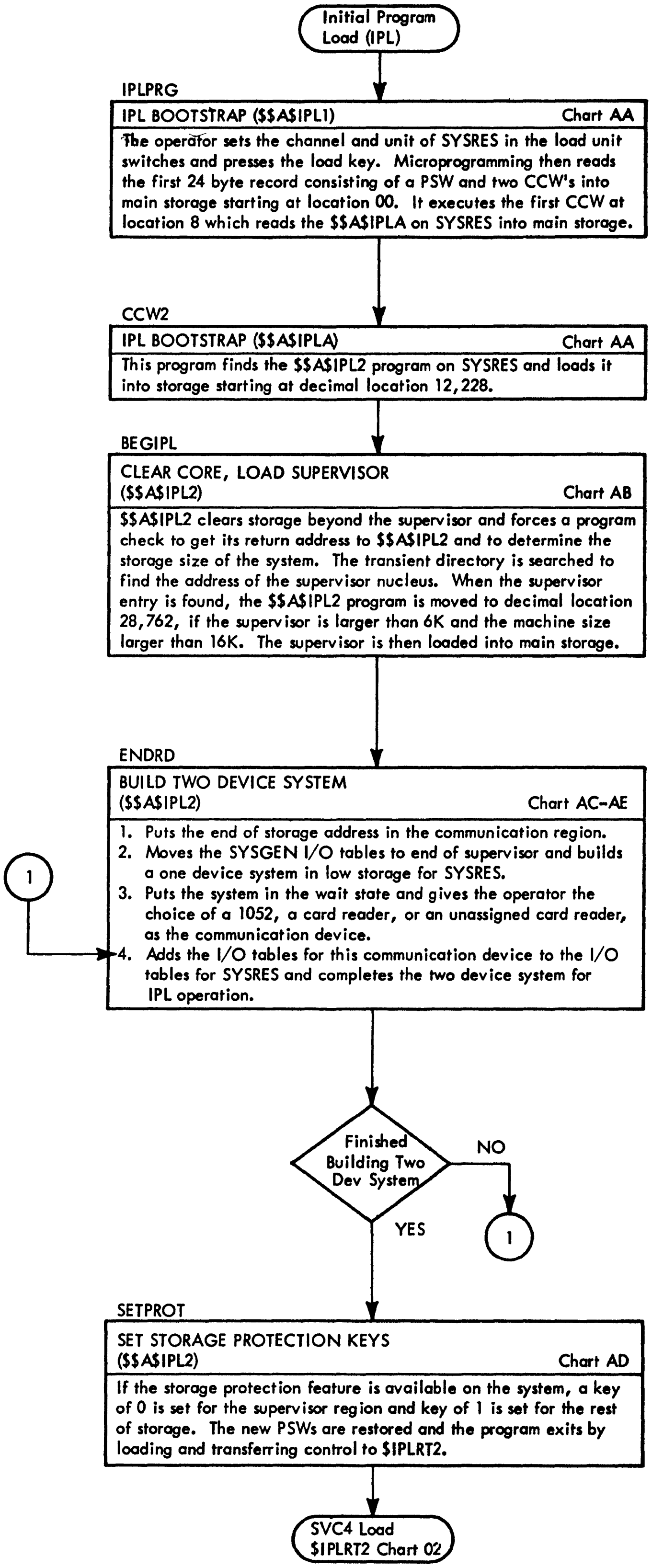



Chart 01. Initial Program Load ($$A$IPL1)

After the system I/O tables have been moved, a PUB is built in low storage for SYSRES. A LUB is assigned for this PUB and the FOCL is set to point to the PUB for the SYSRES device. The system is put into the wait state and the operator has the option of selecting the communication device desired for IPL. If the desired communication device is:

1. A card reader, and it is already assigned as SYSRDR, the operator presses the EXTERNAL INTERRUPT key causing an external interrupt.

2. A card reader, and it is not assigned as SYSRDR, the operator presses the START key on the reader causing an I/O (device end) interrupt.

3. A 1052, the operator presses the REQUEST key causing an I/O attention interrupt.

After the operator has taken the appropriate action for choosing a communication device, a PUB and LUB are added and the FOCL is updated to show the new device. This completes building of the 2-device system for IPL.

A check is made to see if the storage protection feature is supported. If so, the storage protection keys are set. The supervisor area, in blocks of 2K, receives a storage protection key of 0. The upper part of the supervisor that is not an even multiple of 2K and the remainder of main storage are not protected. They receive a storage protection key of 1. The $$A$IPL2 issues a SVC of 4 to load the $IPLRT2 program overlaying the $$A$IPL2 program.

## $IPLRT2, CHART 02

The $IPLRT2 program is loaded and executed every time the operator chooses to IPL the system. It is loaded (by the $$A$IPL2 program) starting at 4K bytes beyond the end of supervisor. See Figure 30 for a map of main storage. Before loading $IPLRT2, the $$A$IPL2 program has moved the system I/O tables to the end of supervisor. A 2-device system, SYSRES and SYSRDR or SYSLOG, has been built by the $$A$IPL2 program for IPL operations.

The $IPLRT2 program performs the following functions:

- Adds a device to the system.
- Deletes a device from the system.
- Sets the system date.
- Sets the system time of day, if the timer feature is supported.

The ADD, DEL, and SET statements are entered from the IPL communication device (SYSRDR or SYSLOG). The formats for these statements follow.

ADD -- Add a Device to the PUB Table

| Operation | Operand |
|---|---|
| ADD | X'cuu'[(k)],devicetype[,X'ss'] |

X'cuu' = Channel and unit numbers in hexadecimal.

k = S, if the device is switchable (is physically attached to two adjacent channels). The designated channel is the lower of the two channels.

k = 0-255 indicates the priority of the device, if the device cannot be switched. The highest priority is 0. If k is not given, a priority of 255 is assumed.

devicetype = 2400T7 for 7-track, 2400-series magnetic tape units.
2400T9 for 9-track, 2400-series magnetic tape units.
1442N1 for 1442N1 card read punch.
See Figure 26 for additional device type codes.

X'ss' = Device specifications used for tape mode. If device specifications are not specified, X'ss' has the following set values:

X'C0' for 9-track tape
X'90' for 7-track tape
X'00' for non-tapes
X'00', X'01', X'02', and X'03' are invalid as X'ss'

* See Figure 27 for a complete list of density settings.

The end-of-block character Ⓑ (alter code 5) must be given after each ADD statement if the communication device is a printer-keyboard.

DEL -- Delete a Device from PUB Table

| Operation | Operand |
|---|---|
| DEL | X'cuu' |

Where cuu is the channel and unit numbers, in hex, of the device to be deleted.

The end-of-block character Ⓑ (alter code 5) must be given after each DEL statement if the communication device is a printer-keyboard.

SET -- Set Date and Time of Day

| Operation | Operand |
|---|---|
| SET | [DATE=n1][,CLOCK=n2] |

The entries in the operand represent the following:

DATE=n1 Sets the system date to the specified value. n1 has one of the following formats:

mm/dd/yy
dd/mm/yy

Where mm specifies the month, dd specifies the day, and yy specifies the year. The format to be used is that selected when the system was generated.

CLOCK=n2 Must be given at IPL time if the timer feature is present. Sets the system clock to the specified value. n2 has the following format:

hh/mm/ss

Where hh specifies hours (00-23), mm specifies minutes (00-50), and ss specifies seconds (00-59).

After a card is read, the operation code is evaluated by a translate and test instruction to determine the type of statement. This instruction then determines the address of the routine for processing the statement.

Add routine: The add routine checks to ensure the device is not already assigned. 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 new entry 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 actual location of the PUB to be deleted in the PUB table. All PUB's beyond this point are moved up one PUB length overlaying the PUB to be deleted. The LUB table and FOCL pointers are updated so they no longer point to a nonexisting PUB entry. The routine returns to read another control statement.

Set Routine: The set time of day routine determines the operand format of the set statement.

- The DATERT subroutine converts the month, day, and year to decimal. This information is then stored in the system date field of the communication region (displacement 79).
- The TIMERT subroutine is used, if the timer feature is supported, to put the time of day (in seconds) into hexadecimal location 54.

The SET card signals the end of the control statements. 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 at the end of supervisor to their permanent location in the supervisor area. This move overlays the two device IPL I/O tables that were built by $$A$IPL2 and finishes the IPL operation.

The End-of-Job transient ($$BEOJ3) is loaded to initiate normal job processing.

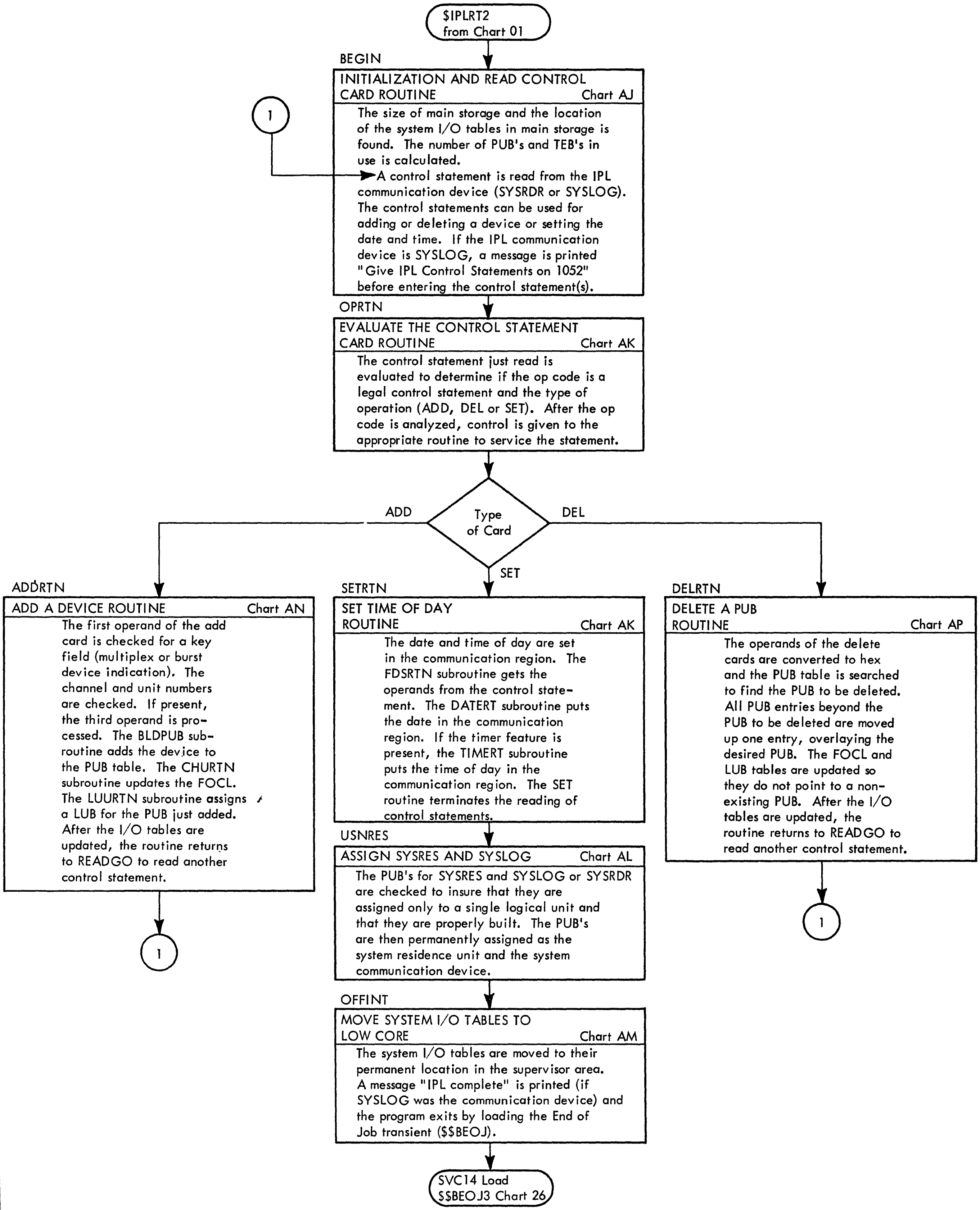



Chart 02. Initial Program Load ($IPLRT2)

## JOB CONTROL PROGRAM

The job control program provides job-to-job transition for background programs. It also prepares background program job steps for execution. (One or more background programs can be executed within a single job. Each such execution is called a job step.) Job control does not prepare foreground programs for execution. They are prepared by the foreground program initiator B-transients.

Job control performs various functions on the basis of information provided in job control statements:

- Prepares programs for execution.

- 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 job control 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 not copied to SYSLNK.

- Assigns device addresses to symbolic units.

- Sets up fields in the supervisor communication region.

- Edits and stores volume and file label information.

- Prepares for restarting checkpointed programs.

- Clears the background program area to binary zeros between job steps.

The job control program is executed in the background program area and is overlaid by the job step it is preparing for execution. 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. A job step is normally ended with the EOJ macro.

## PROGRAM FLOW

Functionally job control consists of four phases and one B-transient, which are identified as $JOBCTLA, $JOBCTLD, $JOBCTLG, $JOBCTLJ, and $$BLSTIO.

$JOBCTLA (Chart 03): This phase is the initial entry into job control. It is loaded every time job control is fetched and is considered the root phase. (It is resident in main storage at all times during job control execution and contains routines that are used by the other phases of job control.)

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 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 main storage as a result of the previous control statement.

See Figure 2 for I/O flow. Figure 31 represents the storage allocation for job control.

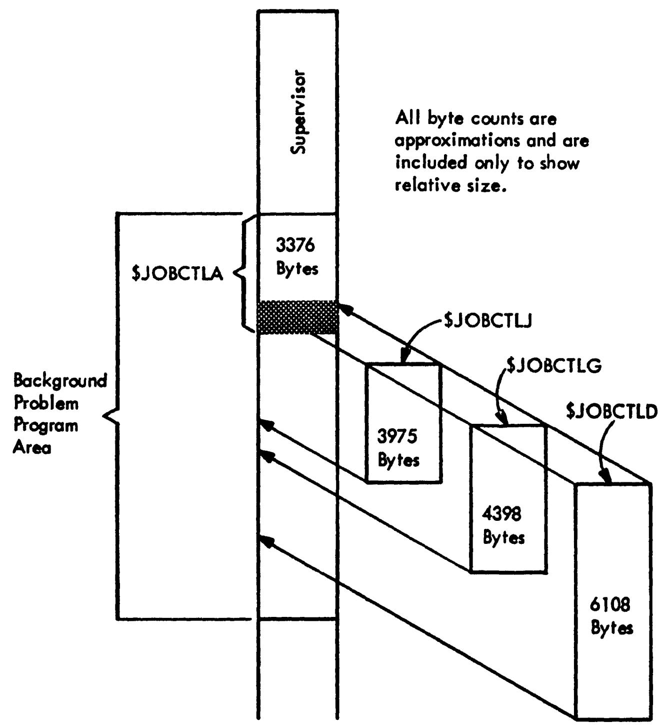



Figure 31. Job Control Storage Allocation

$JOBCTLD (Charts 4 and 5): Contains the processing routines for the following control statements:

1. ASSGN
2. CLOSE
3. DVCDN
4. DVCUP
5. LISTIO

6. RESET

7. UNA

$JOBCTLG (Charts 6, 7, and 8): Contains the processing routines for the following control statements:

1. CANCEL
2. /& (for EOJ)
3. EXEC
4. JOB
5. LOG
6. NOLOG
7. OPTION
8. PAUSE
9. ALLOC
10. MAP
11. STOP

$JOBCTLJ (Charts 9, 10, and 11): Contains the processing routines for the following control statements:

1. ACTION
2. INCLUDE
3. DATE
4. SET
5. UPSI
6. RSTRT
7. MTC
8. LBLTYP
9. VOL
10. TPLAB
11. DLAB
12. XTENT
13. HOLD
14. RELSE
15. UCS

$$BLSTIO: This B-transient contains subroutines used by the DVCDN and LISTIO control statement processors of $JOBCTLD. When required by these processors, it is fetched (SVC 2) into the supervisor B-transient area.

$JOBCTLA

Note:
Job Control is entered from the supervisor fetch routine. It can be entered normally by means of the EOJ macro or abnormally from the B-transients $$BILSVC, $$BPCHK, $$BTERM, $$BEOJ, $$BEOJI. In all cases the B-transient $$BEOJ is used to issue the actual fetch for Job Control ($$JOBCTLA). This phase includes:
1. The initialization routine (JOBCTL).
2. The control statement read routine (CONTROL).
3. The phase vector table (see Figure 79).
4. The root phase subroutines (Charts BD-BK).
5. The root phase error message routines (Chart BL).
   Items 4 and 5 are used by other job control phases.

JOBCTL

**INITIALIZATION** — Chart BA

Initializes the job control program as follows:
1. Loads base register.
2. Seizes control of the system.
3. Sets up the communication region.
4. Checks available record counts if system units are supported on a DASD device. (SYSRES).
5. Saves the LUBs to unassign extent JIBs of the previous job-step.

The main storage area that contains this coding is subsequently used as a 120 byte I/O area. Control statements are read into this area and messages are displayed from it.

* 04-A
05-A
06-A
07-A
08-A
09-A
10-A
11-A

* A

1

CONTROL

**CONTROL STATEMENT READ** — Chart BB

1. Reads control statement.
2. Ignores blank statements.
3. Checks column 1 to determine if the statement is from the operator or the programmer. An indicator is set to control error processing in the phase vector table lookup routine. A blank in column 1 indicates an operator statement. // in columns 1 and 2 indicates a programmer statement.
4. Scans the control statement for the first operand (may be //,/*,*, or the operation field).
5. Performs initialization for the phase vector table lookup.

2

BTLOOP

**PHASE VECTOR TABLE LOOKUP** — Chart BC

1. The phase vector table is searched for an argument that is equal to the current operand (register POINT1 is pointing to column 1 of the statement or to the first character of the operation field). See Figure 79 for the format of an entry in the phase vector table.
2. The operand is checked for validity.
3. The statement is logged as specified in the condition switches of the found entry.
4. If the correct processing phase is not in main storage it is loaded (SVC 4).
5. Processing for the operands //,/*,*, or IGNORE is in the root phase and they are processed as follows:
   // - The statement is scanned to make the next operand (operation field) available for processing. Exit to 2
   /* - Ignored. Exit to 1
   * - Ignored. Exit to 1
   IGNORE - The Job Control input switch is set to indicate SYSRDR. Exit to 1
6. Control is transferred to the correct processing routine as specified by table A, B, or C for all other operands.

TABLE A

| PHASE | $JOBCTLD |
|---|---|
| Operand | Chart |
| ASSGN | 04 |
| CLOSE | 05 |
| DVCDN | 05 |
| DVCUP | 05 |
| LISTIO | 05 |
| RESET | 05 |
| UNA | 05 |

TABLE B

| PHASE | $JOBCTLG |
|---|---|
| Operand | Chart |
| CANCEL | 07 |
| /& (for EOJ) | 07 |
| EXEC | 08 |
| JOB | 07 |
| LOG | 08 |
| NOLOG | 08 |
| OPTION | 06 |
| PAUSE | 08 |
| ALLOC | 08 |
| MAP | 08 |
| STOP | 08 |

TABLE C

| PHASE | $JOBCTLJ |
|---|---|
| Operand | Chart |
| ACTION | 09 |
| INCLUDE | 09 |
| DATE | 09 |
| SET | 09 |
| UPSI | 09 |
| RSTRT | 09 |
| MTC | 09 |
| LBLTYP | 10 |
| VOL | 10 |
| TPLAB | 10 |
| DLAB | 10 |
| XTENT | 10 |
| HOLD | 11 |
| RELSE | 11 |
| UCS | 11 |

Chart 03. Job Control ($JOBCTLA) Root Phase

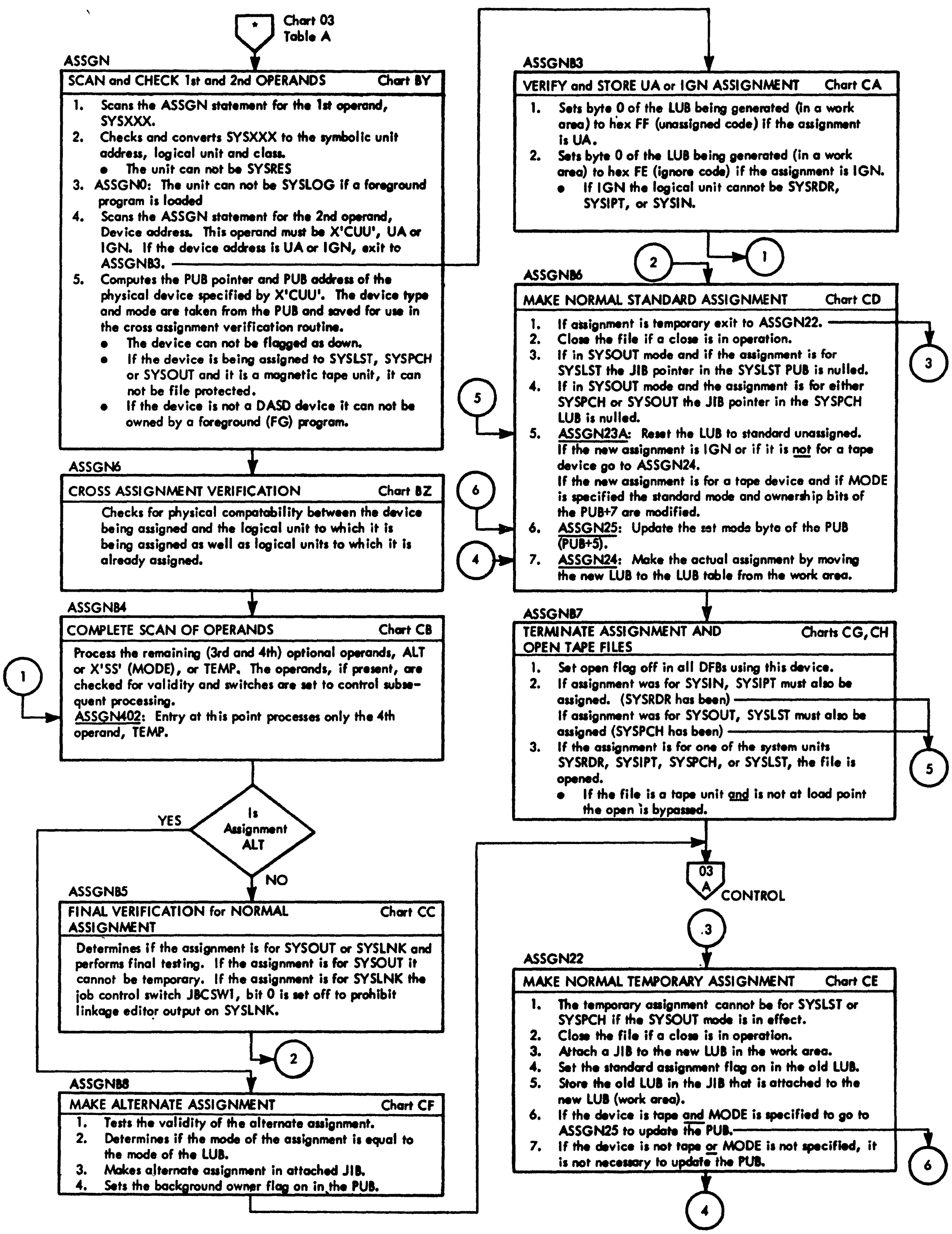



Chart 04. Job Control ($JOBCTLD) Statement Processor (Part 1 of 2)

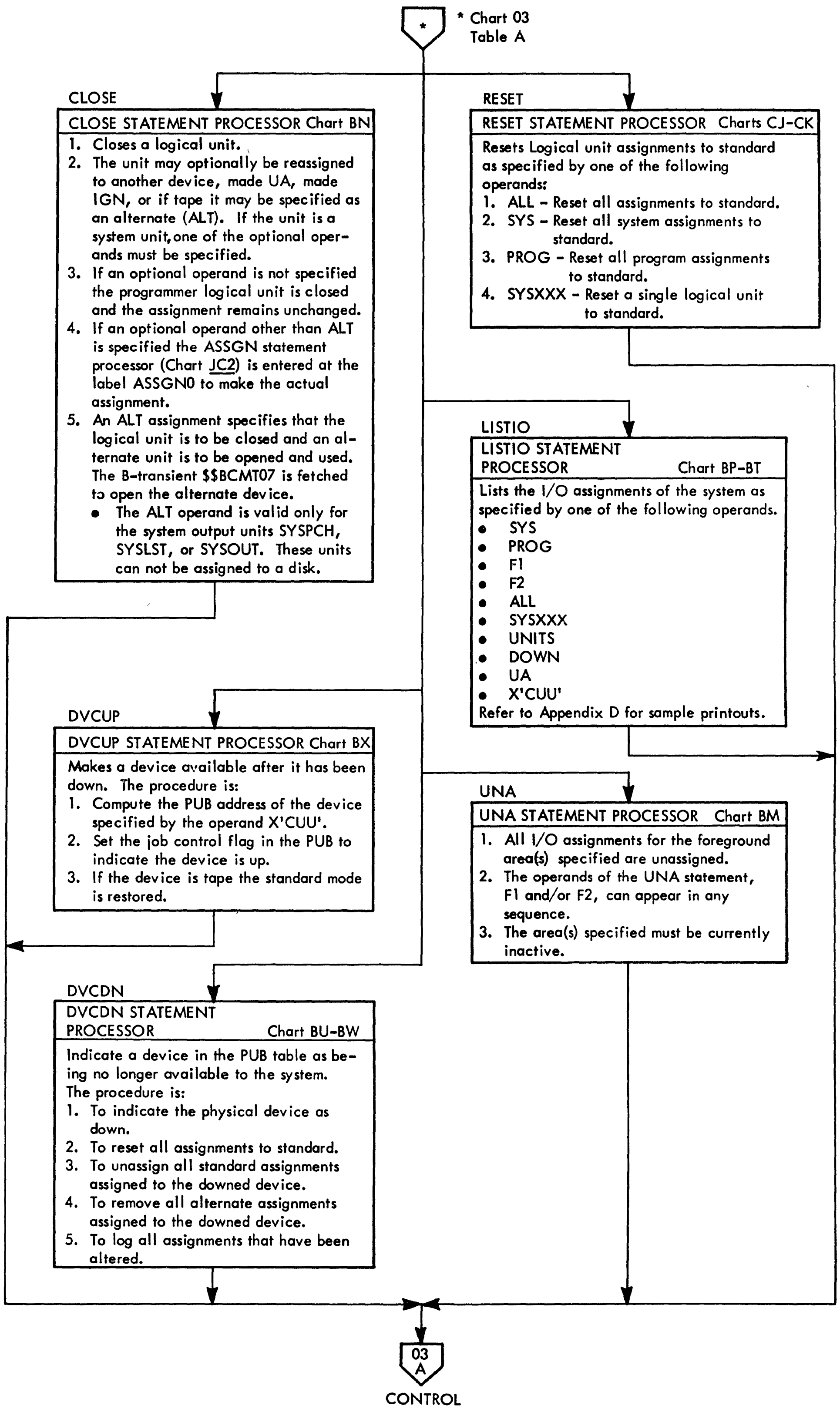



Chart 05. Job Control ($JOBCTLD) Statement Processor (Part 2 of 2)

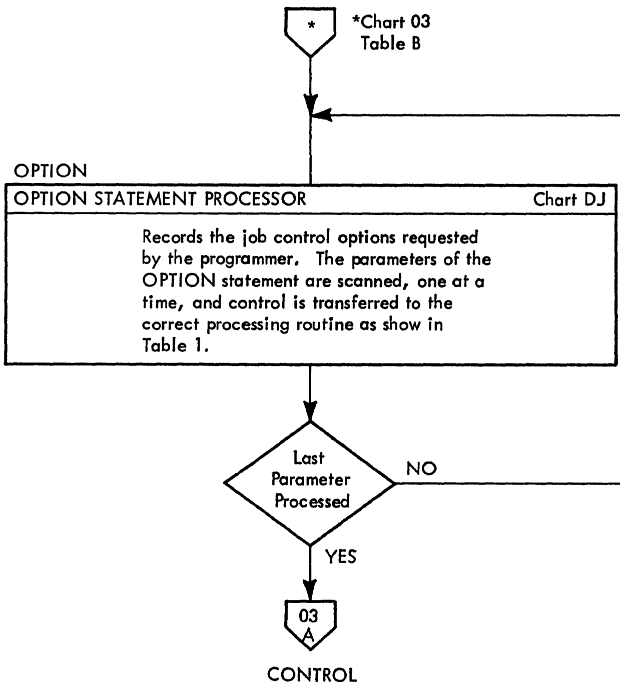



Table #1

| PARAM | RTN LABEL | CHART |
|---|---|---|
| DECK | DECK<br>Set DECK option ON, COMREG+58, bit 0 | DK |
| NODECK | NODECK<br>Set DECK option OFF, COMREG+58, bit 0 | DK |
| LIST | LIST<br>Set LIST option ON, COMREG+58, bit 1 | DK |
| NOLIST | NOLIST<br>Set LIST option OFF, COMREG+58, bit 1 | DK |
| LISTX | LISTX<br>Set LISTX option ON, COMREG+58, bit 2 | DK |
| NOLISTX | NLISTX<br>Set LISTX option OFF, COMREG+58, bit 2 | DK |
| SYM | SYM<br>Set SYM option ON, COMREG+58, bit 3 | DK |
| NOSYM | NOSYM<br>Set SYM option OFF, COMREG+58, bit 3 | DK |
| XREF | XREF<br>Set XREF option ON, COMREG+58, bit 4 | DK |
| NOXREF | NOXREF<br>Set XREF option OFF, COMREG+58, bit 4 | DK |
| ERRS | ERRS<br>Set ERRS option ON, COMREG+58, bit 5 | DK |
| NOERRS | NOERRS<br>Set ERRS option OFF, COMREG+58, bit 5 | DK |
| 48C | C48<br>Set 48 character set<br>option ON, COMREG+58, bit 6 | DK |
| 60C | C60<br>Set 60 character set<br>option ON, COMREG+58, bit 6 | DK |
| NOLOG | OPTNLG<br>Set the SYSLST-LOG option<br>OFF, COMREG+59, bit 3 | DK |
| USRLABEL | USRLBL<br>Initialize Job Control<br>Label Processing for USER LABEL Processing. Set label area disk addr equal track 1, record 0. | DL |
| STDLABEL | STDLBL<br>Initialize Job Control Label Processing for STANDARD LABEL Processing.<br>Set label area disk addr<br>equal to track 0, record 0. | DL |
| DUMP | DUMP<br>Set Job control dump option ON, COMREG+59, bit 1 | DL |
| NODUMP | NODUMP<br>Set job control dump option OFF, COMREG+59, bit 1 | DL |
| LINK | LINK<br>Set LINKEDT control bits, COMREG+57, bits 0, 2 | DL |
| NOLINK | NOLINK<br>Set LINK option OFF, COMREG+57, bit 0 | DL |
| LOG | OPTLOG<br>Set the SYSLST-LOG option ON, COMREG+59, bit 3 | DL |
| CATAL | CATAL<br>Set LINKEDT control bits, COMREG+57, bits 0,2,3 | DL |

Chart 06. Job Control ($JOBCTLG) Statement Processor (Part 1 of 3)

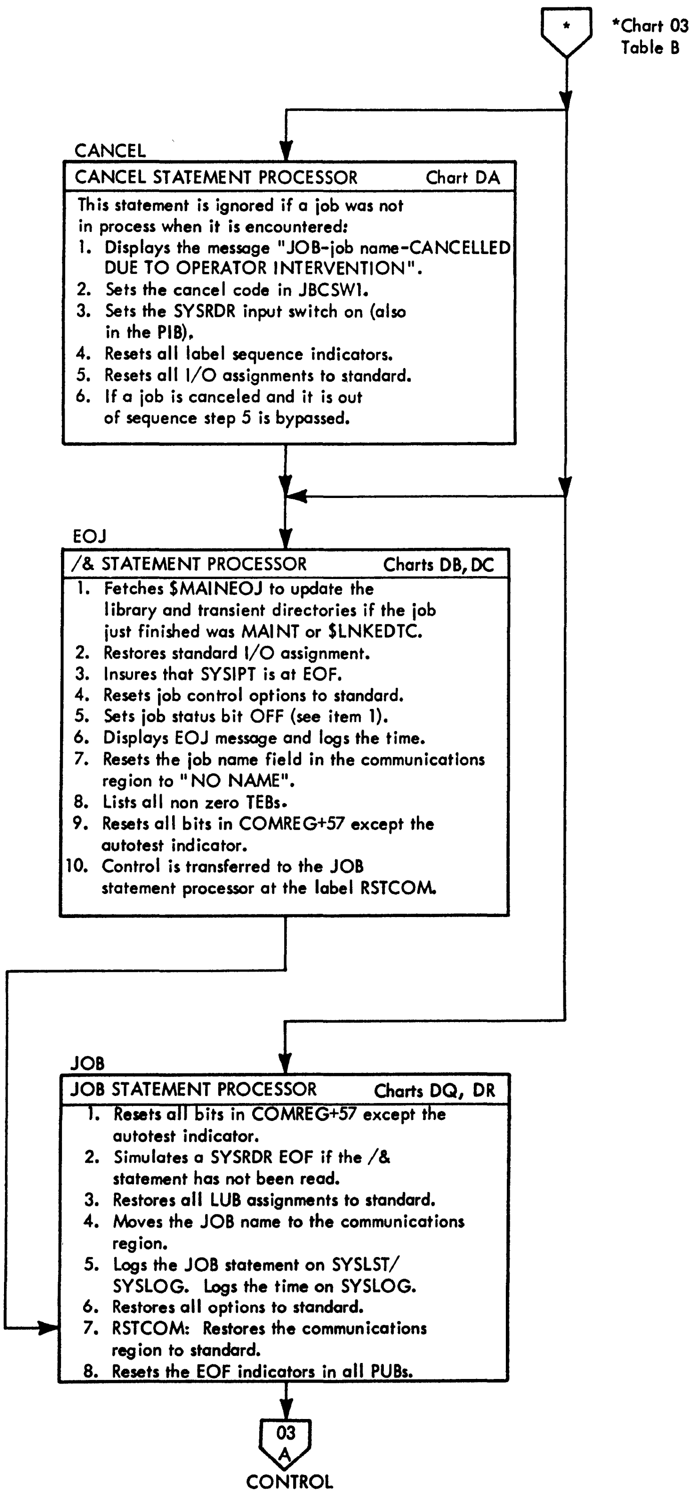



**Chart 07. Job Control ($JOBCTLG) Statement Processor (Part 2 of 3)**

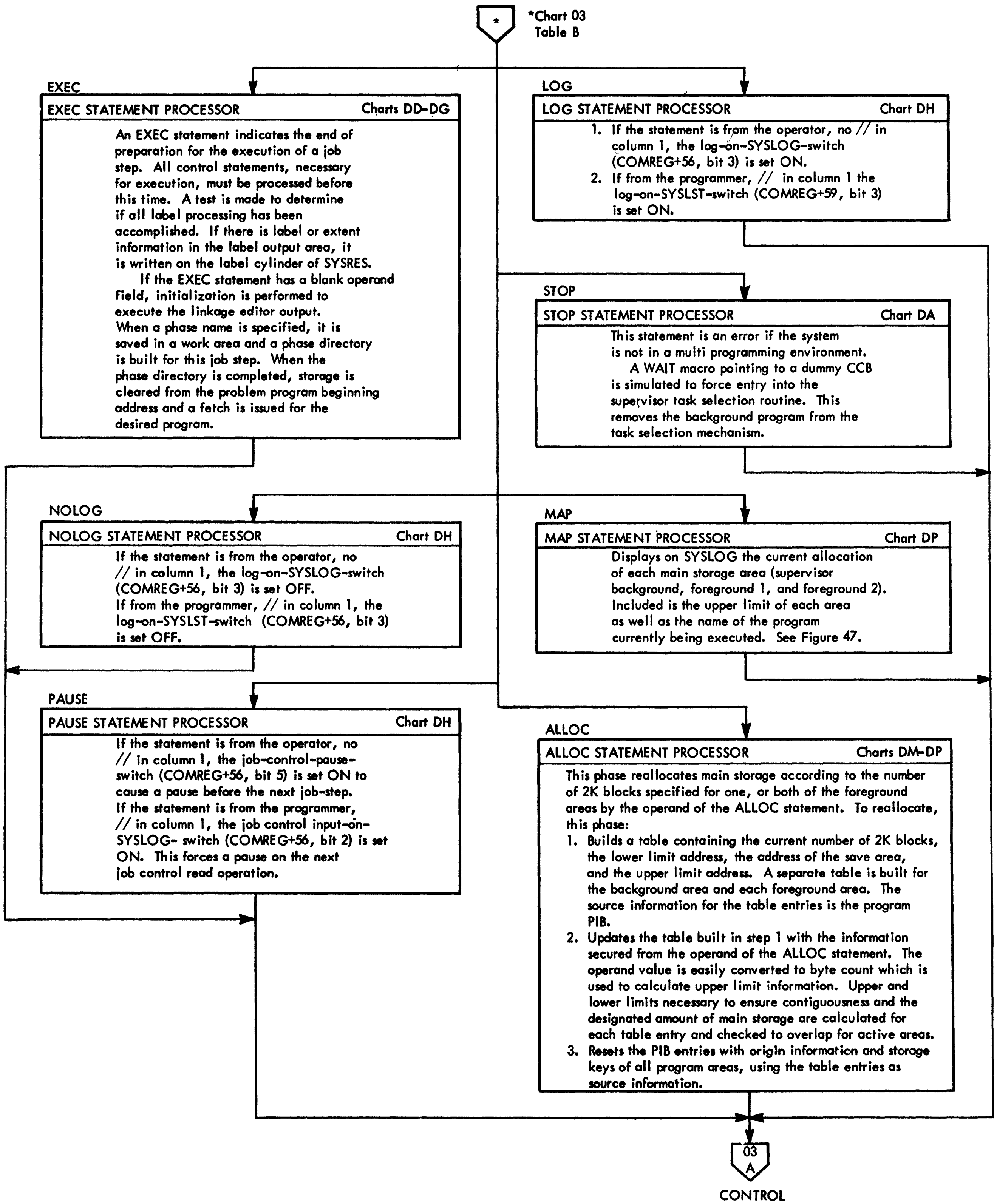



Chart 08. Job Control ($JOBCTLG) Statement Processor (Part 3 of 3)

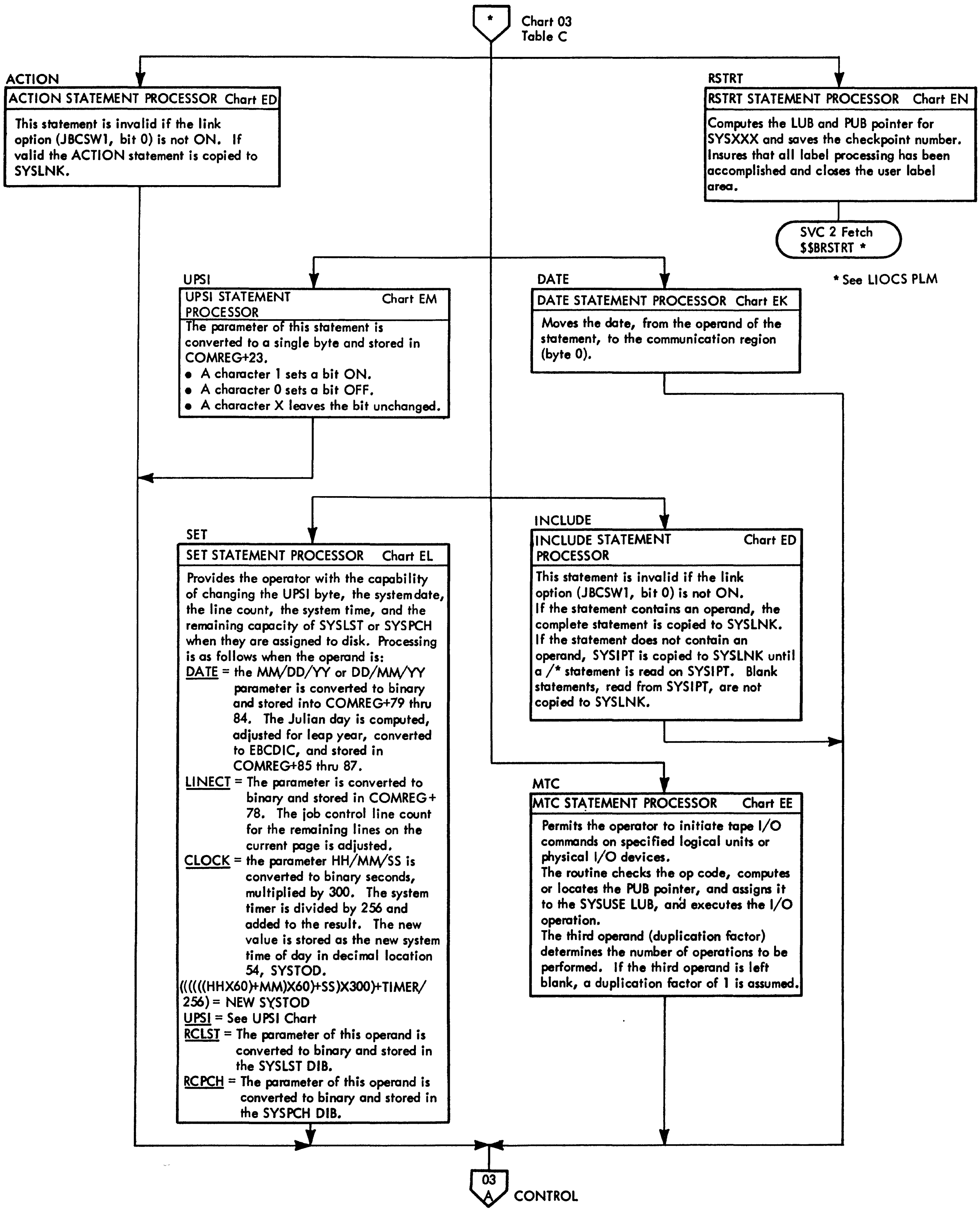



Chart 09. Job Control ($JOBCTLJ) Statement Processor (Part 1 of 3)

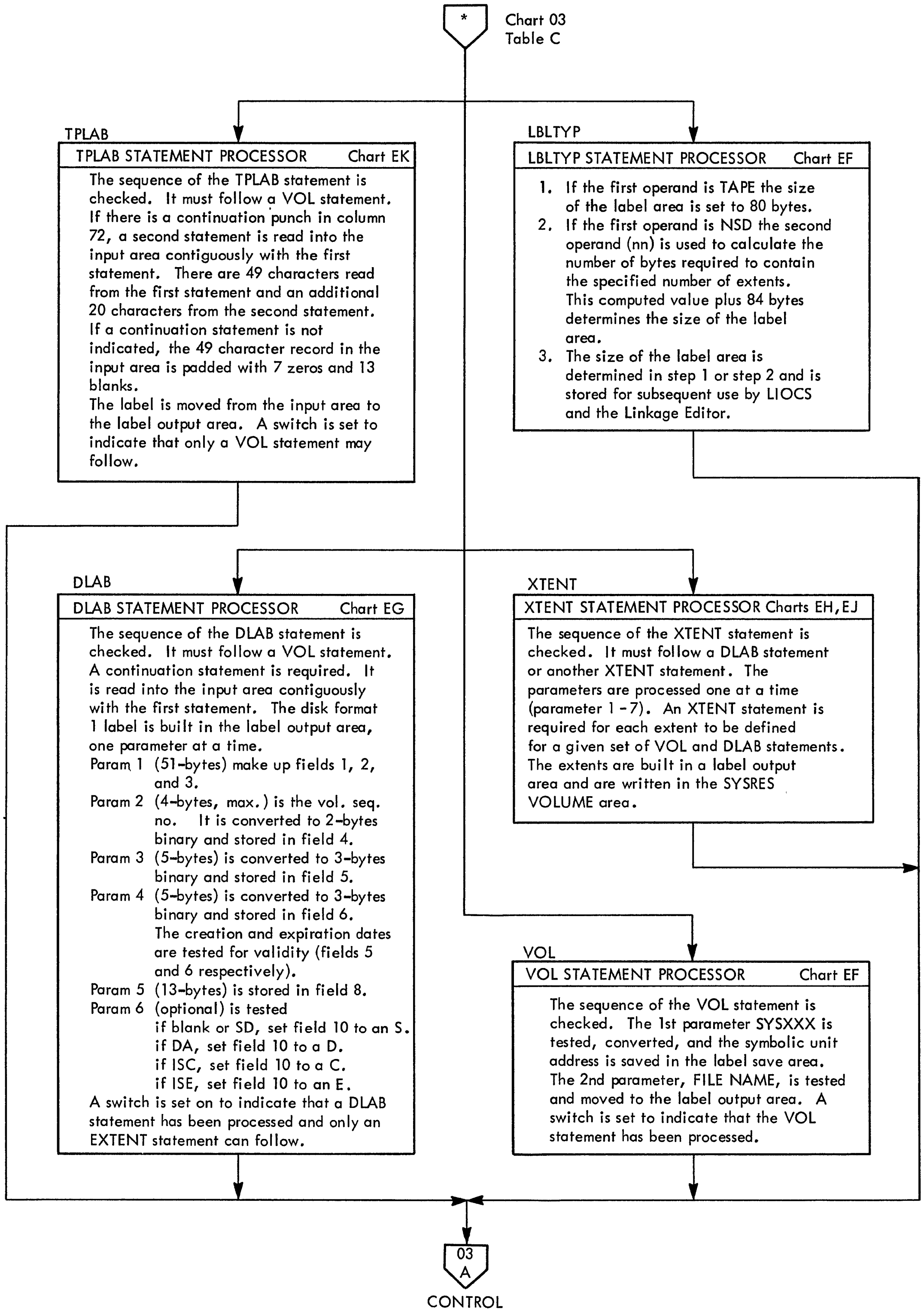



Chart 10. Job Control ($JOBCTLJ) Statement Processor (Part 2 of 3)

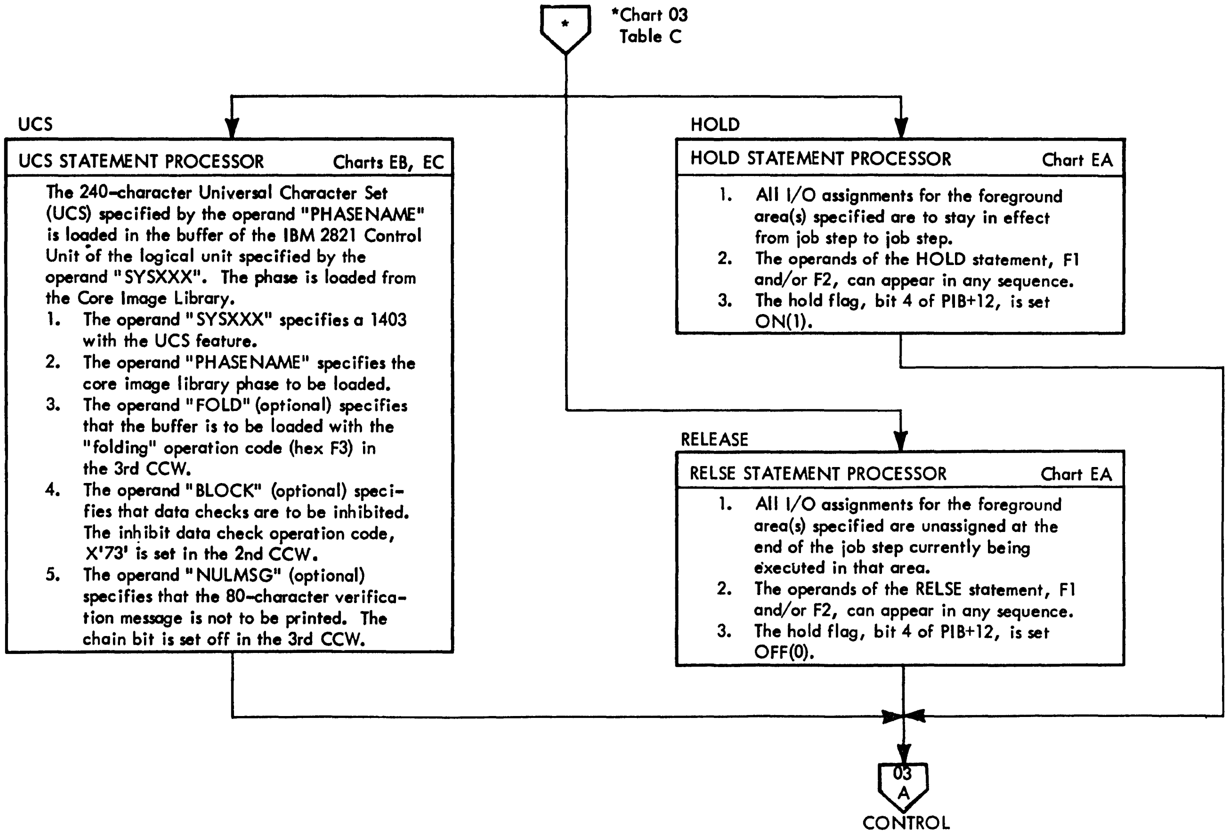



Chart 11. Job Control ($JOBCTLJ) Statement Processor (Part 3 of 3)

## SUPERVISOR CONTROL PROGRAMS

Three divisions of Supervisor Control Programs are presented in the following sequence in this manual:

1. Resident Supervisor ($$A$SUP1)

    a. Supervisor Interrupt Processors

    b. Physical IOCS

2. A-Transient Programs ($$ANERRx)

3. B-Transient Programs ($$Bxxxxx)

RESIDENT SUPERVISOR CHARTS 12 THROUGH 17

Supervisor is the storage resident portion of the Disk Operating System. It is loaded into storage at IPL time and remains there throughout system operations. Refer to Section 3 of this manual for information about generation of the resident supervisor. Refer to Figure 14 in Section 3 for information about the storage organization of the resident supervisor.

Infrequently used supervisory functions are not included in the resident supervisor. They are in the form of transient programs (A and B) and are fetched or loaded from the core image library when needed.

### Supervisor Interrupt Processors

This portion of the resident supervisor processes the following system interrupts:

- Supervisor call interrupt
- I/O Interrupt
- Program check interrupt
- External interrupt
- Machine check interrupt

### Multiprogramming Support (MPS)

General Entry and General Exit routines provide the mechanism for multiprogramming support. Refer to these areas on Chart 12 of additional descriptions for multiprogramming concepts. Figure 33 illustrates the task selection procedure associated with multiprogramming.

### Batch Job Support (BJS)

BJS is an inclusive part of MPS support.

### Supervisor Call Interrupt SVC

SVC is detected by microprogramming, which loads the SVC new PSW. The SVC interrupt processor (Chart 14) analyzes the SVC code placed in the SVC old PSW by microprogramming. Control is transferred to the appropriate processing routine. SVC codes greater than 27 cause a cancel. Some SVCs are optional and cause a cancel if supervisor was generated without the option. (See Figure 32 for a list of supervisor calls.)

SVC 0: Execute the user's channel program (EXCP). The address of the user's command control block (CCB) must be supplied in general register 1 before issuing this SVC. Return may be either to the interrupted program or to the highest priority program ready to run.

Note: When an SVC 0 is issued by supervisor or A-Transient programs, the address of the CCB must be supplied in general register 15 before issuing the SVC.

SVC 1: Fetches a phase. A fetch loads a phase from the core image library and branches to the entry address in that phase. The load and entry addresses are obtained from the core image directory entry for the phase being fetched. The storage address of the phase name must be supplied in general register 1 before issuing this SVC. The user may override the linkage editor entry address by supplying an entry address in general register 0. Return may be either to the interrupted program or to the highest priority program ready to run.

SVC 2: Fetches a B-transient. Loads a B-transient program (phase name prefix equals $$B) from the core image library to the B-transient area (Refer to Figure 14) and enters the B-transient at its load address plus 8 bytes. The storage address of the B-transient phase name must be supplied in general register 1.

An address in general register 0 is ignored. The B-transient is loaded at the beginning address of the B-transient area. General register 15 is loaded with this address and may be used by B-transients as a base register. Return may be either to the interrupted program or to the highest priority program ready to run.

Only one program can use the B-transient area at a time. If the B-transient program is SVC 7 bound, another program is selected. This program becomes SVC 2 bound (waiting for the B-transient area) if it issues an SVC 2. Another program is then selected.

Note: Supervisor may branch directly to the SVC 2 routine when fetching a B-transient. If the transient is not in the library when referenced by the supervisor, the system will enter the wait state.

SVC 3: Fetches or returns from an A-transient. Load an A-transient program (phase name prefix equals $$A) from the core image library to the A-transient area (Refer to Figure 14) and enters the A-transient at its load address plus 8 bytes. The storage address of the A-transient phase name must be supplied in general register 1.

An address in general register 0 is ignored. The A-transient is loaded at the beginning address of the A-transient area. General register 11 is loaded with this address and is used by A-transients as a base register. Return will be to the interrupted program.

Note: Supervisor may branch directly to the SVC 3 routine when fetching an A-transient. Only programs operating in the supervisor mode can issue an SVC 3. If the transient is not in the library, the system will enter the wait state.

CAUTION: SVC 3 is also used as a return from an A-transient program. The last byte of the A-transient name field determines the usage.

- X'00' Returning from error recovery A-transients.

- X'01' Returning from physical attention transients ($$ANERRZ, Y, 0) or post cancel by any A-transient.

- Last byte is alpha - fetch A-transient.

When returning from an A-transient, the branch address is in general register 15. The A-transient must load one of the exit addresses from the error recovery block (ERBLOC). Refer to Figure 42.

SVC 4: Loads a phase from the core image library and returns to the user. See the following Note. The storage address of the phase name must be supplied in general register 1 before issuing this SVC. The user may override the link-edited load address by supplying a load address in general register 0. Upon return to the user, general register 1 contains the phase entry address adjusted for any changes in the phase's load address.

Note: Return may be either to the interrupted program or to the highest priority program ready to run.

SVC 5: Modifies the supervisor communications region. Supplies the supervisory support for the MVCOM macro. The sequence of events is:

1. MVCOM macro issues an SVC 5.

2. SVC 5 fetches $$ANERR1 by branching to the SVC 3 routine.

3. $$ANERR1 alters the supervisor communications region as specified by the MVCOM macro.

Return may be either to the interrupted program or to the highest priority program ready to run.

SVC 6: Cancels a background or foreground program. Cancel code X'23' is posted to the PIB for the program issuing the SVC 6. Refer to Figure 22 for the format of the PIB tables, to Chart 14 for General Cancel Routine, and Figure 34 for cancel codes. The next time the canceled program is selected on general exit, a branch is made to the SVC 2 routine to fetch the cancel B-transient program, $$BEOJ3.

SVC 7: Waits for I/O to complete or a timer interrupt to occur. SVC 7 supplies the supervisory support for the WAIT macro.

With MPS option: Returns directly to the interrupted program if the traffic bit has been posted in the CCB or TECB. See SVC 24 in this list for an explanation of the TECB. If traffic bit is not posted:

1. Change the status of the interrupted program PIB to SVC 7 bound (not ready to run).

2. Select the highest priority program that is ready to run.

When I/O is completed or a timer interrupt occurs,

1. The traffic bit is posted in the CCB or TECB.

2. The PIB is restored to the ready-to-run status.

3. When this program is again selected at general exit, the old PSW will be loaded with the address of the second instruction of the WAIT macro expansion.

Without MPS option: Returns directly to the interrupted program if the traffic bit has been posted in the CCB or TECB. (See SVC 24 in this list for an explanation of the TECB.)

If the traffic bit is not posted, the system enters the wait state with interrupts enabled.

SVC 8: Supplies the supervisory support to temporarily return from a B-transient program to the problem program. The B-transient area is not released. The task selection exit loads the problem program registers. Return to the B-transient program is accomplished by issuing an SVC 9.

SVC 9: Supplies the supervisory support for returning to the B-transient after an SVC 8 is issued. The task selection exit loads the B-transient registers.

SVC 10: Sets a timer interval. This SVC is optional and the issuing program will be canceled if supervisor is generated without IT option. Only the timer supported program can issue an SVC 10. Others will be canceled.

The time interval is specified in general register 1 by the user (SETIME macro). The system time of day (SYSTOD, X'54') is updated to the time that the next interrupt should occur (may change if another SVC 10 is issued). The system timer (SYSTIMER, X'50') is set to the specified time interval. The time interval in SYSTIMER immediately begins to lapse. Refer to IBM System/360 Principles of Operation, Form A22-6821, for information concerning the operation of SYSTIMER.

Note: Current system time of day can be obtained by shifting out the low order byte from the remaining time interval (SYSTIMER) and subtracting it from system time of day (SYSTOD). Time in SYSTOD is represented in the form, seconds x 300. Time in SYSTIMER is in the form, seconds x 300 x 256.

An SVC 10 returns directly to the timer supported program. No task selection is performed.

SVC 11: Returns from a B-transient releasing the B-transient area. SVC 11 is invalid if issued by other than a B-transient. The logical transient area is released for use by other programs or tasks. Return will be to the highest priority program ready to run.

SVC 12: Supplies the supervisory support to reset flags to 0 in the linkage control byte (displacement 57 in the supervisor communications region). The user loads a mask (1 byte, hexadecimal) into general register 1. This mask is ANDed with the linkage control byte. An SVC 12 returns directly to the interrupted program. No task selection is performed.

SVC 13: Supplies the supervisory support to set flags to 1 in the linkage control byte (displacement 57 in the supervisor communications region). The user loads a mask (1 byte, hexadecimal) into general register 1. This mask is ORed with the linkage control byte. An SVC 13 returns directly to the interrupted program. No task selection is performed.

SVC 14: This is the normal end of job (EOJ). Cancel code X'10' is posted to the PIB for the program issuing the SVC 14. Refer to Figure 22 for the format of the PIB tables and to Chart 14 for General Cancel routine. The next time the canceled program is selected on general exit, a branch is made to the SVC 2 routine to fetch the cancel B-transient program $$BEOJ3. Job Control is loaded by $$BEOJ to perform end-of-job-step.

SVC 15: This is the same as SVC 0 (EXCP), with this exception: when the CHANQ table is full, the SVC is ignored. Return is direct to the interrupted program in this case. If the CHANQ table is not full, general register 0 is zeroed and EXCP is issued (see SVC 0 in this list).

Note: The CHANQ table is full when the free list pointer (FLPTR) equals X'FF'. Refer to Figure 21 for the format of the CHANQ table and to Figure 35 for CHANQ operation.

SVC 16 THROUGH 21: These supervisor calls provide supervisory support for the STXIT and EXIT macros. They are optional, and the issuing program will be canceled if supervisor was not generated with the applicable option.

- SVC 16 stores the address of the user's program check (PC) routine and save area address in the PC option table.
- SVC 17 provides a return from the user's PC routine to the program interrupted due to a program check.
- SVC 18 stores the address of the user's interval timer (IT) routine and save area address in the IT option table. SVC 18 can only be issued by the timer supported program.
- SVC 19 provides a return from the user's IT routine to the timer supported program. SVC 19 can only be issued by the timer supported program.

- SVC 20 stores the address of the user's operator communications (OC) routine and save area address in the OC option table.

- SVC 21 provides a return from the user's OC routine to the program interrupted by the external interrupt key.

The address of the user routine is specified in general register 0, and the address of the users save area is specified in general register 1 in all cases. Refer to Figure 25 for the format of the option tables.

SVC 16, 18, and 20 return directly to the interrupted program

SVC 17, 19, and 21 return either to the interrupted program or to the highest priority program ready to run.

SVC 22: Seizes the system and provides a release from such a seizure. The SVC 22 is ignored if supervisor was generated without MPS option. The program issuing an SVC 22 is canceled if the PSW protection key field does not equal 0. (Only Job Control and B-transient programs can issue an SVC 22.)

The first SVC 22 issued seizes the system and the next one issued releases the system. The program can change the system mask by loading the system mask it requires into the last byte of general register 0. If the program masks off all interrupts, the loaded PSW contains its protection key.

The task selection mechanism is altered by the first SVC 22 so that only supervisor or quiesce I/O tasks and the program that issued the SVC 22 can be selected. The next SVC 22 issued restores the task selection mechanism. The contents of the last byte of general register 0 are again used as the system mask.

Return from each SVC 22 is directly to the interrupted program.

CAUTION: There is no way to cancel a program that has seized the system.

- The program must have no pending I/O operations.

- The program cannot issue supervisor calls while the system is seized.

SVC 23: Loads phase header. Retrieves the load address for a specified phase from the core image directory. The program issuing an SVC 23 is canceled if supervisor was generated without MPS option or the PSW protection key does not equal 0. (Only Job Control and B-transient programs can issue an SVC 23.)

The user must specify the address of the core image phase name in general register 1 and the address of where the load address is to be stored in general register 0. The main fetch subroutine scans the core image directory and retrieve the load address. If the phase is found in the directory, the load address (3 bytes) is stored at the user's address specified by general register 0. If the phase is not found, the return is to the interrupted program.

SVC 24: Stores the address of the user's timer event control block (TECB) and sets a timer interval. This SVC is optional, and the issuing program will be canceled if supervisor is generated without IT option. Only the timer supported program can issue an SVC 24. Others will be canceled.

The address of the user's TECB is specified in general register 0, and the time interval is specified in general register 1.

The traffic bit is reset in the user's TECB, and the TECB address is stored in the IT option table. Refer to Figure 25 for the format of the IT option table.

Note: The TECB has the same format as a command control block (CCB), but only the traffic bit is used. The traffic bit is set when a timer interrupt occurs. Refer to Figure 36 for the format of the CCB.

The time interval is set, and the system time of day is updated as for an SVC 10. (See SVC 10.) An SVC 24 returns directly to the timer supported program. No task selection is performed.

The user causes the program to wait for the timer interrupt to occur by issuing an SVC 7. (See SVC 7 in this list.)

SVC 25: Issues halt I/O on a tele-processing device. If supervisor is generated without tele-processing option, a program issuing an SVC 25 will be canceled.

The address of any command control block (CCB) containing the symbolic unit address for this device must be supplied in general register 1 before issuing this SVC.

An HIO instruction is issued to the device if:

1. it is a tele-processing device and

2. there is I/O pending for the device.

In this case, return is to the highest priority program ready to run. The device busy flag is reset at this time. If an SVC 25 is issued for other than a tele-processing device, it is ignored.

SVC 26: Validate address limits. The program issuing an SVC 26 will be canceled if the PSW protection key does not equal 0. (Only Job Control and B-transient programs can issue an SVC 26.)

The upper address must be specified in general register 2, and the lower address must be specified in general register 1. The upper address must be within main storage, and the lower address must be higher than the end of supervisor address, or the program will be canceled (ERR25). Return is to the interrupted program. No task selection is performed.

With MPS option: The PIK of the program issuing the SVC 26 must equal the storage protection key for both addresses or the program is canceled (ERR25).

With BJS option (batch only): SVC 26 is ignored in a BJS system without storage protection.

SVC 27: Same as SVC 25, except the EXCP CCB is not dequeued if the CSW has been stored after a HIO command.

DOS SUPERVISOR CALLS

| Macro Supported | SVC | Function |
|---|---|---|
| EXCP | 0 | Execute channel programs. |
| FETCH | 1<br>2<br>3 | Fetch any phase.<br>Fetch a logical transient (B- transient).<br>Fetch or return from a physical transient (A- transient). |
| LOAD | 4 | Load any phase. |
| MVCOM | 5 | Modify supervisor communications region. |
| CANCEL | 6 | Cancel a problem program. |
| WAIT | 7<br>8 | Wait on a CCB or TECB.<br>Transfer control to the problem program from a logical transient (B- transient). |
| LBRET | 9 | Return to a logical transient (B- transient) from the problem program after a SVC 8. |
| SETIME | *10<br>11<br>12<br>13 | Set timer interval.<br>Return from a logical transient (B- transient).<br>Logical AND (Reset) to second Job Control byte (displacement 57 in communications region).<br>Logical OR (Set) to second Job Control byte (displacement 57 in communications region). |
| EOJ | 14<br>15 | Cancel job and go to Job Control for end of job step.<br>Same as SVC 0 except ignored if CHANQ table is full. (Primarily used by ERP). |
| STXIT (PC) | *16 | Provides supervisor with linkage to user's PC routine for program check interrupts. |
| EXIT (PC) | *17 | Return from user's PC routine. |
| STXIT (IT) | *18 | Provides Supervisor with linkage to user's IT routine for interval timer interrupts. |
| EXIT (IT) | *19 | Return from user's IT routine. |
| STXIT (OC) | *20 | Provides supervisor with linkage to user's OC routine for external or attention interrupts (operator communications). |
| EXIT (OC) | *21<br>*22<br>*23<br>*24<br>*25<br>*26<br>*27 | Return from user's OC routine.<br>The first SVC 22 seizes the system for the issuing program by disabling multiprogram operation. The second SVC 22 releases the system (enables multiprogram operation).<br>Load phase header. Phase load address is stored at user's address.<br>Provide supervisor with linkage to user's TECB and set timer interval.<br>Issues HALT I/O on a Tele- processing device.<br>Validate address limits.<br>Special HIO on teleprocessing devices. |

* = optional

Figure 32. DOS Supervisor Calls

## I/O Interrupt

This is detected by microprogramming, which loads the I/O new PSW. Refer to the I/O Interrupt Processor on Chart 15.

## Program Check Interrupt

This is detected by microprogramming, which loads the program check new PSW. Refer to Program Check Interrupt Processor on Chart 16.

## External Interrupt

This is detected by microprogramming, which loads the external new PSW. External interrupts can be caused by:

- Timer
- External interrupt key
- Signal (not supported)

Refer to External Interrupt Processor on Chart 16.

## Machine Check Interrupt

This is detected by microprogramming, which loads the machine check new PSW. The SEREP action code (S) is stored in storage location 0001, and the system enters the wait state. Refer to Chart 12.

| Priority Table | | |
|---|---|---|
| Sample Status | PIB Tables | MVCFLD |
| X'84' | Supervisor task PIB | X'60' |
| X'84' | Quiesce I/O task PIB | X'50' |
| X'80' | Attention task PIB | X'40' |
| X'83' | † Foreground 1 program PIB | X'30' |
| X'82' | † Foreground 2 program PIB | X'20' |
| X'83' | Background program PIB | X'10' |
| X'85' | † All bound PIB | X'00' |

1. Test status flags in order specified by priority table.
2. Select 1st PIB for which the TRT function is not X'00'.

| PIB Flags During Task Selection | | Table of Selection Criteria | |
|---|---|---|---|
| Meaning of Status | Flag | Label | TRT Function |
| Detached | X'80' | TRTMSK | X'00' |
| Waiting for B-transient area | X'81' | TRTLTK | X'00' or X'03' (Note 1) |
| Waiting for CCB or TECB | X'82' | | X'00' |
| Ready to run | X'83' | TRTRUN | X'03' or X'00' (Note 2) |
| Inactive SUPVR or Quiesce I/O | X'84' | | X'00' |
| Active SUPVR, Quiesce I/O, or All bound | X'85' | | X'05' |

Note 1: X'00' when the B-transient area is in use.

Note 2: X'00' when a task has seized the system. That task's status flag will equal X'84' or X'85'.

† These PIB's are generated for MPS option only.

Figure 33. Task Selection Procedure

| Type | Cancel Code | Condition | Label |
|---|---|---|---|
| Logical Cancels | X'10' | Normal EOJ | ERR10 |
| | X'20' | Program check | ERR20 |
| | X'21' | Illegal SVC | ERR21 |
| | X'22' | Phase not found | ERR22 |
| | X'23' | Program request | ERR23 |
| | X'24' | Operator intervention | ERR24 |
| | X'25' | Invalid address limit | ERR25 |
| | X'26' | Unassigned LUB code | ERR26 |
| | X'27' | Invalid LUB code in CCB | ERR27 |
| Logical I/O Cancels | X'30' | Reading past /& on SYSRDR or SYSIPT. | ERR30 |
| | X'31' | Error queue overflow or no CHANQ entry available for ERP. | ERR31 |
| | X'32' | DASD address not within JIB extents. | ERR32 |
| | X'33' | No long seek in user's channel program. | ERR33 |

Figure 34. Supervisor Cancel Codes

## PHYSICAL INPUT/OUTPUT CONTROL SYSTEM (PIOCS)

Physical IOCS is that portion of the resident Supervisor that:

- Builds a schedule of I/O operations for all devices on the system (CHANQ table). Refer to Channel Scheduler on Chart 15. Also, see Figure 35 for CHANQ operation.
- Starts the actual I/O operations on a device (SIO). Refer to Actual I/O on Chart 15.
- Schedules the starting of all I/O operations and monitors all events associated with I/O. Refer to I/O Interrupt Processor on Chart 15.
- Performs error recovery procedures (ERP). Refer to Unit Check, Quiesce I/O, ERP Exits, and Resident Disk Error Recovery on Chart 17. See Figures 41 and 42 for CSW testing and error recovery block layout, respectively.

Figures 36 through 40 illustrate:

1. Command Control Block (CCB)
2. Channel Command Word (CCW)
3. Program Status Word (PSW)
4. Channel Address Word (CAW)
5. Channel Status Word (CSW)

Because of their usage, these items are included in this section.

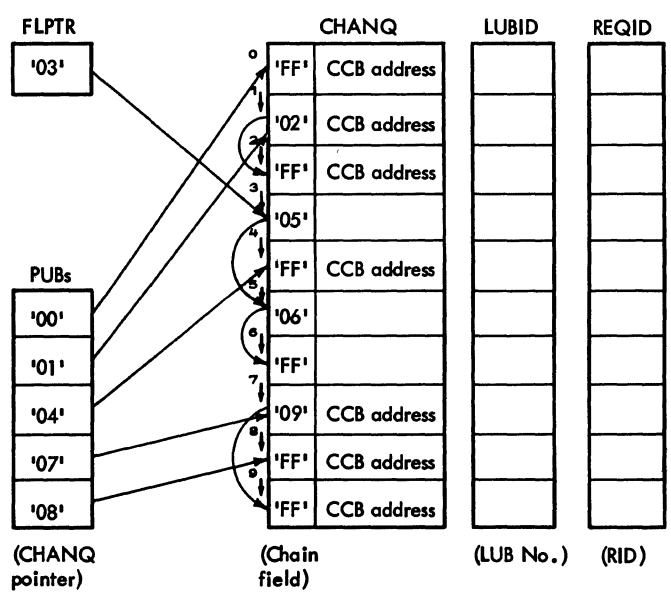



Figure 35. Example of the CHANQ Table Operation

| Bytes | 0–1 | 2–3 | 4–5 | 6–7 | 8 | 9–11 | 12 | 13–15 | 16–23 |
|---|---|---|---|---|---|---|---|---|---|
| Field | Count | Transmission Information | CSW Status Bits | Symbolic Unit Address | Not Used Must Contain X'00' | CCW Address | Reserved for Physical IOCS | CCW Address in CSW | |
| Used For | Residual Count | Transmitting Information Between Physical IOCS and Problem Program | (see Byte 4 / Byte 5) | Hexadecimal Representation of SYSnnn<br>SYSRDR = 0000<br>SYSIPT = 0001<br>SYSPCH = 0002<br>SYSLST = 0003<br>SYSLOG = 0004<br>SYSRES = 0006<br>SYS000 = 0100<br>SYS001 = 0101<br>etc. | Not Used | Address of CCW Associated with This CCB | X'40' - Channel Appendage Routine Present for Teleprocessing Device<br>X'20' - Sense Information Desired<br>X'80' - CCB being used by ERP | Address of CCW in the CSW Stored at Channel End, or Address of the Channel Appendage Routine for Teleprocessing Devices | 8 Bytes Appended to the CCB when Sense Information is Desired |

| Byte 4 BIT | DESIGNATION | Byte 5 BIT | DESIGNATION |
|---|---|---|---|
| 32 | Attention | 40 | Program-controlled interruption |
| 33 | Status modifier | 41 | Incorrect length |
| 34 | Control unit end | 42 | Program check |
| 35 | Busy | 43 | Protection check |
| 36 | Channel end | 44 | Channel data check |
| 37 | Device end | 45 | Channel control check |
| 38 | Unit check | 46 | Interface control check |
| 39 | Unit exception | 47 | Chaining check |

Byte 2

| Bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| | Traffic Bit (Wait) | * End-of-file (/* or /&) | Unrecoverable I/O Error | Acceptable Unrecoverable I/O Error | Return DASD Data Checks, or 2671 errors to the User | Post at Device End | Return Tape Read Data Check, 2540 Punch Equipment Check, or DASD Data Checks on Count, Read Command, or Verify Command | User Error Routine |
| Set On By | PIOCS | PIOCS | PIOCS | Pr. Pr. | Pr. Pr. | Pr. Pr. | PIOCS | Pr. Pr. |

Byte 3

| Bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| | Data Check in Count Area-DASD | Track Overrun-DASD | End of Cylinder-DASD | 2540 Punch Equipment Check or Tape, or DASD Read Error | Questionable Condition | Unused | Carriage Channel 9 or Verify Error for DASD | Command Chaining Retry from the next CCW to be executed |
| Set On By | PIOCS | PIOCS | PIOCS | PIOCS | PIOCS | | PIOCS | PIOCS |

PIOCS = Physical IOCS
Pr. Pr. = Problem Program

Bytes 4 and 5 contain the status bytes of the Channel Status Word (Bits 32-47). If byte 2, bit 5 is on and device end results as a separate interrupt, device end will be ORed into CCB byte 4.

* Indicates /* or /& statement encountered on SYSRDR or SYSIPT. Byte 4, bit 7 (unit exception) is also on.

Figure 36. Command Control Block (CCB)

## Command Control Block

Communication between the problem program and physical IOCS is accomplished by the use of the command control block (CCB). The CCB is two double words in length with eight major fields as shown in Figure 36. All data in the CCB is in the hexadecimal format. The eight fields of the CCB are listed and described as follows:

1. Count Field (bytes 0, 1): Contains the residual count, which is stored in these two bytes by PIOCS when the CCB is removed from the queue.

2. Transmission Information (bytes 2, 3): Used for communication between PIOCS and the problem program.

Note: Bytes 0 through 3 are ANDed off, by PIOCS, when the CCB is placed in the queue. Communication bits set on by the problem program are left on because an AND instruction is used by PIOCS for resetting bytes 0 through 3.

3. CSW Status Bits (bytes 4, 5): Contains the CSW status information, which is stored in these two bytes by PIOCS before control is returned to the problem program.

Note: An information bit, in bytes 2 through 5, indicates the occurrence of the indicated condition when the bit is on.

4. Symbolic Unit Address (bytes 6, 7): Contains the 2-byte hexadecimal representation of SYSnnn. This value represents the location of the logical unit in the LUB table (see Figure 36) and is placed in the CCB by the problem program.

5. Byte 8: Is not used and must contain hexadecimal 0.

6. CCW Address (bytes 9-11): Contains the address of the CCW that is associated with this CCB. This address is placed. in the CCB by the problem program.

7. Byte 12: X'80'-CCB being used by ERP. X'40'-channel appendage routine for a teleprocessing device. X'20'-sense information desired.

8. CCW Address in CSW (bytes 13-15): Contains the CCW address from the CSW. This address is stored by PIOCS before control is returned to the problem program. A CCB that has been queued, by PIOCS, to service a problem program I/O request cannot be used for a second problem program I/O request until the first request has been completed.

Note: Bytes 13-15 contains the address of the channel appendage routine when bit X'40' is set in byte 12.

9. Sense Information (bytes 16-23): Bytes 16-23 are appended to the CCB when X'20' is set in byte 12.

| Byte | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | | 4 | | | | | | | | 5 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| Name | Command Code | | | | | | | | ← Data Address → | | | | | | | | | | | | | | | | | | | | | | | | Flags | | | | | 0 | 0 | 0 | ( Ignored) | | | | | | | |
| Field | A | | | | | | | | B | | | | | | | | | | | | | | | | | | | | | | | | C | | | | | D | | | E | | | | | | | |

| Byte | 6 | | | | | | | | 7 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| Name | ← Count → | | | | | | | | | | | | | | | |
| Field | F | | | | | | | | | | | | | | | |

| FIELD | NAME | DESCRIPTION |
|---|---|---|
| A | Command Code | Bits 0-7:<br>Specify the operation to be performed. (See Note on Part 2 of this Figure) |
| B | Data Address | Bits 8-31:<br>Specify the location of a byte in main storage. It is the first location referred to in the area designated by the CCW. |
| C | Flags | Bits 32-36:<br>Specify the flag bits used in conjunction with the CCW.<br><br>Bit 32-<br>Chain-Data (CD) causes the address portion of the next CCW to be used with the current CCW. †Note<br><br>Bit 33-<br>Chain-Command (CC) causes the command code and data address of the next CCW to be used. The chain data flag (bit 32) takes precedence over this flag.<br><br>Bit 34-<br>Suppress Length Indication (SLI) causes a possible incorrect length indication to be suppressed. The chain data flag (bit 32) takes precedence over this flag.<br><br>Bit 35-<br>Skip (SKIP) suppresses the transfer of information to main storage.<br><br>Bit 36-<br>Program Control Interruption (PCI) causes the channel to generate an interrupt when the CCW is fetched. |
| D | Reserved | Bits 37-39:<br>(Must contain zeros)* |
| E | Ignored | Bits 40-47:<br>Not checked |
| F | Count | Bits 48-63:<br>Specify the number of bytes in the operation |

*The transfer in channel command (TIC) is the one exception to this statement.
† Note: Chain data cannot be done on 360/30 if a high-speed device is being used. Example- 2311, 2400 mod III.

Figure 37. Channel Command Word (CCW), Part 1 of 2

Note,

## CHANNEL COMMAND CODES

**Command Code** assignments are listed in the following table. The symbol X indicates that the bit position is ignored; M identifies a modifier bit.

| CODE | COMMAND |
|---|---|
| MMMM 0 1 0 0 | Sense |
| XXXX 1 0 0 0 | Transfer in channel |
| MMMM 1 1 0 0 | Read backward |
| MMMM MM0 1 | Write |
| MMMM MM1 0 | Read |
| MMMM MM1 1 | Control |

DASD CHANNEL COMMAND CODES (See A26-5988)

| Command for CCW | | Count | Multiple Track (M-T) Off 8-Bit Code 0123 4567 | Hex | Dec | M-T On † Hex | Dec |
|---|---|---|---|---|---|---|---|
| Control | No Op | X | 0000 0011 | 03 | 03 | | |
| | Release* | X | 0001 0111 | 17 | 23 | | |
| | Restore | X | 0001 0011 | 13 | 19 | | |
| | Seek | 6 | 0000 0111 | 07 | 07 | | |
| | Seek Cylinder | 6 | 0000 1011 | 0B | 11 | | |
| | Seek Head | 6 | 0001 1011 | 1B | 27 | | |
| | Sense I/O | 4 | 0000 0100 | 04 | 04 | | |
| | Set File Mask | 1 | 0001 1111 | 1F | 31 | | |
| | Space Record | X | 0000 1111 | 0F | 15 | | |
| | Transfer in Channel | X | XXXX 1000 | X8 | | | |
| Search † | Home Address EQ | 4 (usually) | 0011 1001 | 39 | 57 | B9 | 185 |
| | Identifier EQ | 5 (usually) | 0011 0001 | 31 | 49 | B1 | 177 |
| | Identifier HI | 5 (usually) | 0101 0001 | 51 | 81 | D1 | 209 |
| | Identifier EQ or HI | 5 (usually) | 0111 0001 | 71 | 113 | F1 | 241 |
| | Key EQ | 1 to 255 | 0010 1001 | 29 | 41 | A9 | 169 |
| | Key HI | 1 to 255 | 0100 1001 | 49 | 73 | C9 | 201 |
| | Key EQ or HI | 1 to 255 | 0110 1001 | 69 | 105 | E9 | 233 |
| | Key & Data EQ* | Note 1 | 0010 1101 | 2D | 45 | AD | 173 |
| | Key & Data HI* | | 0100 1101 | 4D | 77 | CD | 205 |
| | Key & Data EQ or HI* | | 0110 1101 | 6D | 109 | ED | 237 |
| Read † | Home Address | 5 | 0001 1010 | 1A | 26 | 9A | 154 |
| | Count | 8 | 0001 0010 | 12 | 18 | 92 | 146 |
| | Record R0 | Number | 0001 0110 | 16 | 22 | 96 | 150 |
| | Data | of bytes | 0000 0110 | 06 | 06 | 86 | 134 |
| | Key & Data | trans- | 0000 1110 | 0E | 14 | 8E | 142 |
| | Count, Key & Data | ferred | 0001 1110 | 1E | 30 | 9E | 158 |
| Write | Home Address | 5 (usually) | 0001 1001 | 19 | 25 | | |
| | Record R0 | 8+KL+DL of R0 | 0001 0101 | 15 | 21 | | |
| | Count, Key & Data | 8+KL+DL | 0001 1101 | 1D | 29 | | |
| | Special Count, Key & Data* | 8+KL+DL | 0000 0001 | 01 | 01 | | |
| | Data | DL | 0000 0101 | 05 | 05 | | |
| | Key & Data | KL & DL | 0000 1101 | 0D | 13 | | |

* Special Feature  Note 1  Includes mask bytes in search argument.
† M-T On = M-T Off except, during Search and Read bit 0 = 1 in M-T On
X = not significant, KL = Key Length DL = Data Length, EQ Equal, HI High

| Device | Command for CCW | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Hex | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1052 | Read Inquiry BCD | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0A | 10 |
| | Read Reader 2 BCD | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 | 02 |
| | Write BCD, Auto Carriage Return | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 09 | 09 |
| | Write BCD, No Auto Carriage Return | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | 01 |
| | No Op | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 03 | 03 |
| | Sense | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 | 04 |
| | Alarm | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0B | 11 |
| 2540 | Read, Feed, Select Stacker SS Type AA | S | S | D | 0 | 0 | 0 | 1 | 0 | | |
| | Read Type AB | 1 | 1 | D | 0 | 0 | 0 | 1 | 0 | | |
| | Read, Feed (1400 compatibility mode only) | 1 | 1 | D | 1 | 0 | 0 | 1 | 0 | | |
| | Feed, Select Stacker SS Type BA | S | S | 1 | 0 | 0 | 0 | 1 | 1 | | |
| | PFR Punch, Feed, Select Stacker SS Type BA | S | S | D | 0 | 1 | 0 | 0 | 1 | | |
| | Punch, Feed, Select Stacker SS Type BB | S | S | D | 0 | 0 | 0 | 0 | 1 | | |
| 1442 N1 | Read | M | M | M | 0 | 0 | 0 | 1 | 0 | | |
| | Write | M | M | M | 0 | 0 | 0 | 0 | 1 | | |
| | Control | M | M | 0 | 0 | 0 | 0 | 1 | 1 | | |
| | No Op | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | |
| | Sense | 0 | 0 | M | M | 0 | 1 | 0 | 0 | | |
| 1403 or 1443 | Write, No Space | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | 01 |
| | Write, Space 1 After Print | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 09 | 09 |
| | Write, Space 2 After Print | 0 | 0 | 0 | 1 | 0 | 0 | Q | 1 | 11 | 17 |
| | Write, Space 3 After Print | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 19 | 25 |
| | Write, Skip To Channel N After Print | 1 | C | H | A | N | 0 | 0 | 1 | | |
| | Diagnostic Read | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 | 02 |
| | Test I/O | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 00 | 00 |
| | Sense | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 | 04 |
| Carriage Control | Space 1 Line Immediately | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0B | 11 |
| | Space 2 Line Immediately | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 13 | 19 |
| | Space 3 Line Immediately | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1B | 27 |
| | Skip To Channel N Immediately | 1 | C | H | A | N | 0 | 1 | 1 | | |
| | No Op | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 03 | 03 |
| 2400 Tape* | Transfer in Channel | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 08 | 08 |
| | Sense | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 04 | 04 |
| | Read Backward** | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0C | 12 |
| | Write | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | 01 |
| | Read | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 02 | 02 |
| | Control | 0 | 0 | C | C | C | 1 | 1 | 1 | | |
| | Mode Set | D | D | M | M | M | 0 | 1 | 1 | | |

2540:

| SS | Stacker |
|---|---|
| 00 | R1 |
| 01 | R2 |
| 10 | RP3 |

| D | Data Mode |
|---|---|
| 0 | EBCDIC |
| 1 | Column Binary |

1442 N1:

| | M | M | M | M | |
|---|---|---|---|---|---|
| Read | 0 | 0 | X | | Eject and SS1 |
| Read | 1 | 0 | X | | Eject and SS1 |
| Read | 0 | 1 | X | | Eject and SS2 |
| Read | 1 | 1 | X | | Eject and SS2 |
| Write | 0 | 0 | X | | SS1 |
| Write | 1 | 0 | X | | Eject and SS1 |
| Write | 0 | 1 | X | | SS2 |
| Write | 1 | 1 | X | | Eject and SS2 |
| Control | 1 | 0 | | | Eject and SS1 |
| Control | 0 | 1 | | | SS2 |
| Control | 1 | 1 | | | Eject and SS2 |
| Sense | | | 1 | 1 | Punch diagnostic |
| Sense | | | 0 | 1 | Read diagnostic |

X = 0 means EBCDIC mode
X = 1 means Column Binary Mode

Carriage Control:

| C | H | A | N | Channel |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 12 |

* 9 track op forces 800 BPI and odd parity, also, it overrides 7 track but does not reset 7 track. Load/Sys Reset forces 7 track to 800 BPI, odd parity, data converter on, translator off.

| C | C | C | Control Codes | Hex | Dec |
|---|---|---|---|---|---|
| 0 | 0 | 0 | REW | 7 | 7 |
| 0 | 0 | 1 | RUN | 0F | 15 |
| 0 | 1 | 0 | ERG | 17 | 23 |
| 0 | 1 | 1 | WTM | 1F | 31 |
| 1 | 0 | 0 | BSR | 27 | 39 |
| 1 | 0 | 1 | BSF | 2F | 47 |
| 1 | 1 | 0 | FSR | 37 | 55 |
| 1 | 1 | 1 | FSF | 3F | 63 |

| D | D | 7 Track Density |
|---|---|---|
| 0 | 0 | 200 |
| 0 | 1 | 556 |
| 1 | 0 | 800 |
| 1 | 1 | 800 |

| M M M (Mode Modifiers) | Set Density | Set Odd Parity | Set Even Parity | Data Converter On | Data Converter Off | Translator On | Translator Off | Request TIE (Track In Error) |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 No Op | | | | | | | | |
| 0 0 1 Not Used | | | | | | | | |
| 0 1 0 Reset Condition | X | X | | X | | | X | |
| 0 1 1 Nine-track only | | | | | | | | X |
| 1 0 0 | | | X | | X | | X | |
| 1 0 1 | X | | X | | X | X | | |
| 1 1 0 Reset Condition | X | X | | | X | | X | |
| 1 1 1 | X | X | | | X | X | | |

**Overrides Data Converter On

Figure 37. Channel Command Word (CCW), Part 2 of 2

Figure 38. Program Status Word (PSW)

<table>
<tr><td>Byte</td><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td><td colspan="8">4</td><td colspan="8">5</td><td colspan="8">6</td><td colspan="8">7</td></tr>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>32</td><td>33</td><td>34</td><td>35</td><td>36</td><td>37</td><td>38</td><td>39</td><td>40</td><td>41</td><td>42</td><td>43</td><td>44</td><td>45</td><td>46</td><td>47</td><td>48</td><td>49</td><td>50</td><td>51</td><td>52</td><td>53</td><td>54</td><td>55</td><td>56</td><td>57</td><td>58</td><td>59</td><td>60</td><td>61</td><td>62</td><td>63</td></tr>
<tr><td>Name</td><td colspan="8">System Mask</td><td colspan="4">Key</td><td colspan="4">CPU Mask</td><td colspan="16">Interruption Code</td><td colspan="2">ILC</td><td colspan="2">CC</td><td colspan="4">Prog. Mask</td><td colspan="24">Instruction Address</td></tr>
<tr><td>Field</td><td colspan="8">A</td><td colspan="4">B</td><td colspan="4">C</td><td colspan="16">D</td><td colspan="2">E</td><td colspan="2">F</td><td colspan="4">G</td><td colspan="24">H</td></tr>
</table>

| FIELD | NAME | DESCRIPTION |
|---|---|---|
| A | System Mask* | Bits 0-7:<br>Are associated with the I/O channels and external signals as follows:<br><br>Bit Interruption source<br>0 Multiplexor channel<br>1 Selector channel 1<br>2 Selector channel 2<br>3 Selector channel 3<br>4 Selector channel 4<br>5 Selector channel 5<br>6 Selector channel 6<br>7 { Timer<br>Interrupt key<br>External signal }<br>*A one-bit equals ON and permits an interrupt. |
| B | Protection Key | Bits 8-11:<br>Form the CPU protection key. The key is matched with a storage key whenever a result is stored. If the protection feature is not implemented, bits 8-11 must be zero when loaded and are zero when stored. |
| C | CPU Mask (AMWP) | Bits 12-15:<br>Form the CPU mask as follows:<br><br>Bit Meaning<br>(A) 12 [If 1 - generate extended ASCII code<br>If 0 - generate EBCDIC<br><br>(M) 13 [If 1 - permits machine check interrupt<br>If 0 - prohibits machine check interrupt<br><br>(W) 14 [If 1 - the CPU is in the wait state<br>If 0 - the CPU is in the running state<br><br>(P) 15 [If 1 - the CPU is in the problem mode<br>If 0 - the CPU is in the supervisor mode |
| D | Interruption Code | Bits 16-31:<br>Identify the cause of the interruption. (See NOTE for specific interruption codes.) |

| FIELD | NAME | DESCRIPTION |
|---|---|---|
| E | Instruction Length Code | Bits 32 and 33:<br>Indicate the length, in halfwords, of the instruction last executed, as follows:<br><br>00 (0) Not available (unpredictable)<br>01 (1) 1 halfword<br>10 (2) 2 halfwords<br>11 (3) 3 halfwords |
| F | Condition Code | Bits 34 and 35:<br>Indicate the last condition code setting. All instructions do not set a condition code.<br><br>00 Condition code 0<br>01 Condition code 1<br>10 Condition code 2<br>11 Condition code 3 |
| G | Program Mask** | Bits 36-39:<br>Form the program mask for the following program exceptions.<br><br>Bit Exception<br><br>36 Fixed-point overflow<br>37 Decimal overflow<br>38 Exponent underflow<br>39 Significance<br><br>**A one-bit equals ON and permits a program check interrupt for a specific exception. |
| H | Instruction Address | Bits 40-63:<br>Indicate the address of the leftmost byte of the next instruction to be executed. |

NOTE

| SOURCE IDENTIFICATION | INTERRUPTION CODE PSW BITS 16-31 | MASK BITS | ILC SET | EXE-CUTION |
|---|---|---|---|---|
| *Input/Output* (old PSW 56, new PSW 120, priority 4) | | | | |
| Multiplexor channel | 00000000 aaaaaaaa | 0 | x | complete |
| Selector channel 1 | 00000001 aaaaaaaa | 1 | x | complete |
| Selector channel 2 | 00000010 aaaaaaaa | 2 | x | complete |
| Selector channel 3 | 00000011 aaaaaaaa | 3 | x | complete |
| Selector channel 4 | 00000100 aaaaaaaa | 4 | x | complete |
| Selector channel 5 | 00000101 aaaaaaaa | 5 | x | complete |
| Selector channel 6 | 00000110 aaaaaaaa | 6 | x | complete |
| *Program* (old PSW 40, new PSW 104, priority 2) | | | | |
| Operation | 00000000 00000001 | | 1,2,3 | suppress |
| Privileged operation | 00000000 00000010 | | 1,2 | suppress |
| Execute | 00000000 00000011 | | 2 | suppress |
| Protection | 00000000 00000100 | | 0,2,3 | suppress/ terminate |
| Addressing | 00000000 00000101 | | 0,1,2,3 | suppress/ terminate |
| Specification | 00000000 00000110 | | 1,2,3 | suppress |
| Data | 00000000 00000111 | | 2,3 | terminate |
| Fixed-point overflow | 00000000 00001000 | 36 | 1,2 | complete |
| Fixed-point divide | 00000000 00001001 | | 1,2 | suppress/ complete |
| Decimal overflow | 00000000 00001010 | 37 | 3 | complete |
| Decimal divide | 00000000 00001011 | | 3 | suppress |
| Exponent overflow | 00000000 00001100 | | 1,2 | terminate |
| Exponent underflow | 00000000 00001101 | 38 | 1,2 | complete |
| Significance | 00000000 00001110 | 39 | 1,2 | complete |
| Floating-point divide | 00000000 00001111 | | 1,2 | suppress |
| *Supervisor Call* (old PSW 32, new PSW 96, priority 2) | | | | |
| Instruction bits | 00000000 rrrrrrrr | | 1 | complete |
| *External* (old PSW 24, new PSW 88, priority 3) | | | | |
| External signal 1 | 00000000 xxxxxxx1 | 7 | x | complete |
| External signal 2 | 00000000 xxxxxx1x | 7 | x | complete |
| External signal 3 | 00000000 xxxxx1xx | 7 | x | complete |
| External signal 4 | 00000000 xxxx1xxx | 7 | x | complete |
| External signal 5 | 00000000 xxx1xxxx | 7 | x | complete |
| External signal 6 | 00000000 xx1xxxxx | 7 | x | complete |
| Interrupt key | 00000000 x1xxxxxx | 7 | x | complete |
| Timer | 00000000 1xxxxxxx | 7 | x | complete |
| *Machine Check* (old PSW 48, new PSW 112, priority 1) | | | | |
| Machine malfunction | 00000000 00000000 | 13 | x | terminate |

a Device address bits
r Bits of $R_1$ and $R_2$ field of SUPERVISOR CALL
x Unpredictable

**Mask bits 0 - 7 refer to the system mask.**
**Mask bits 36 - 39 refer to the program mask.**

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Byte | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | |
| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Name | Key | | | | 0 | 0 | 0 | 0 | ← Command Address → | | | | | | | | | | | | | | | | | | | | | | | |
| Field | A | | | | B | | | | C | | | | | | | | | | | | | | | | | | | | | | | |

| FIELD | NAME | DESCRIPTION |
|---|---|---|
| A | Protection Key | Bits 0 - 3 form the storage protection key for all commands associated with START I/O. This key is matched with a storage key whenever data is placed in storage. (Must contain zeros whenever storage protection is not implemented.) |
| B | Reserved | Bits 4 - 7 (Must contain zeros.) |
| C | Command Address | Bits 8 - 31 Designates the location of the first CCW in main storage associated with the START I/O. (The three low order bits, 29 - 31, must be zeros, specifying a CCW address on integral boundaries of a double word.) |

Figure 39. Channel Address Word (CAW)

Figure 40. Channel Status Word (CSW)

| Byte | 0 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Bit | 0 1 2 3 | 4 5 6 7 | 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 | 24 25 26 27 28 29 30 31 | 32 33 34 35 36 37 38 39 | 40 41 42 43 44 45 46 47 | 48 49 50 51 52 53 54 55 | 56 57 58 59 60 61 62 63 |
| Name | Key | 0 0 0 0 | ← Command Address → | | | ← Status → | | ← Count → | |
| Field | A | B | C | | | D | | E | |

| FIELD | NAME | DESCRIPTION |
|---|---|---|
| A | Protection Key | Bits 0-3 form the storage protection key used in the chain of operations: at the subchannel. |
| B | Reserved | (Must be zeros.) |
| C | Command Address | Bits 8-31 form an address that is eight higher than the address of the last CCW used. * Note |
| D | Status | Bits 32-47 identify the conditions in the device and channel that caused the CSW to be stored. |
| | | Bits 32-39 are obtained over the I/O Interface and indicate conditions detected by the device or the control unit. |
| | | Bits 40-47 are provided by the channel and indicate conditions associated with the subchannel. |
| | | Each status bit represents one type of condition as follows: |
| E | Count | Bits 48-63 form the residual count for the last CCW used. |

| DEVICE OR CONTROL UNIT | |
|---|---|
| **Bit Position** | **Designated Condition** |
| 32 | Attention |
| 33 | Status Modifier |
| 34 | Control Unit End |
| 35 | Busy |
| 36 | Channel End |
| 37 | Device End |
| 38 | Unit Check |
| 39 | Unit Exception |

| CHANNEL/SUBCHANNEL | |
|---|---|
| **Bit Position** | **Designated Condition** |
| 40 | Program-Controlled Interrupt |
| 41 | Incorrect Length |
| 42 | Program Check |
| 43 | Protection Check |
| 44 | Channel Data Check |
| 45 | Channel Control Check |
| 46 | Interface Control Check |
| 47 | Chaining Check |

* Note: This address is not 8 higher on a command reject.

| Status Bit | Status Condition | Action |
|---|---|---|
| 45<br><br>46 | Channel control check<br><br>Interface control check | Enter wait state with all interrupts masked off. |
| 38<br>42<br>43<br>44<br>47 | Unit check<br>Program check<br>Protection check<br>Channel data check<br>Channel chaining check | Exit to unit check on Chart 17 for error recovery. |
| 32 | Attention | For attention from a 1052, include attention task in task selection and take general exit (EXT03). Attention interrupts are ignored if:<br>1. System reallocation or condense is in operation.<br>2. Attention is not from a 1052. |
| 35 | Device busy | Skip channel end test. |
| 36 | Channel end | See Chart FQ for actions taken. Attempts to reschedule the channel (No attempt is made for the multiplex channel unless this is a burst-multiplex device). |
| 37<br><br>34 | Device end<br><br>Control unit end | See Chart FP for actions taken. Attempts to reschedule the channel (If the multiplex channel is being rescheduled, only the device is rescheduled. If the device on the multiplex channel is a burst-multiplex device, both channel and device are rescheduled). |
| 33 and 35 | Control unit busy | Reset device to available. The status is not tested unless neither channel end, device end, nor control unit end has occurred. |

Figure 41. CSW Testing in I/O Interrupt Processor

| Displacement from ERBLOC in Decimal | Length in bytes | Description |
|---|---|---|
| -2 | 2 | Address of first error queue entry in table. |
| 0 | 2 | Address of retry ERP exit (EXRTY). |
| 2 | 2 | Address of ignore ERP exit (EXIGN). |
| 4 | 2 | Address of DISWHY (retry) ERP exit (EXWHY). |
| 6 | 2 | Address of the channel queue table (CHANQ). |
| 8 | 2 | Address of cancel ERP exit (EXCAN). |
| 10 | 2 | Address of last entry in error queue table. |
| 12 | 2 | Address of last entry queued to table (initially ERQUE-22). |
| 14 | 2 | Requestor I/O key (RIK). |
| 16 | 2 | Address of cancel attention exit (ATNCNL). |
| 18 | 2 | Address of attention dequeue exit (PUBDEQ). |
| 20 | 2 | Address of attention exit (EXT02). |
| 22 | 8 | A-transient phase name field ($$ANERRb̸) |
| 30 | 5 x 22 | Five 22 bytes error queue entries. |

Error Queue Entry (22 bytes)

| 0 → 8 | 8 → 10 | 10 → 11 | 11 → 12 | 12 → 18 | 18 → 22 |
|---|---|---|---|---|---|
| CSW (8 bytes) | Displacement Within Pub Table (2 bytes) | Flag Byte | Message Number | Sense Data (6 bytes) | Disk Seek Address (4 bytes) |

| Bit | Designation |
|---|---|
| 0 | Unused |
| 1 | Intervention required |
| 2 | Passback* |
| 3 | Allow ignore |
| 4 | Reserved (DASD) |
| 5 | Allow retry |
| 6 | No CCB available |
| 7 | Unused |

*Put on by device ERP when user wants control returned to him on error.

Figure 42. Error Recovery Block (ERBLOC)

PHYSICAL TRANSIENT PROGRAMS ($$A)--CHARTS 18 THROUGH 20

Physical transient programs are commonly referred to as A-transients. These infrequently used sections of the supervisor reside in the core image library and are fetched by the resident supervisor (SVC 3) only when needed. Each program phase name begins with the prefix characters $$A. These phases are loaded singly into the A-transient area. See Figure 14 for Supervisor storage organization. The A-transients functions within DOS are:

1. Provide device-dependent Error Recovery Procedures (ERP).
2. Issue messages associated with ERP operations, Message Writer.
3. Process 1052 attention requests, Physical Attention Routines.

Figure 43 illustrates each A-transient in terms of phase name, function, and program level chart identification.

ERP: To understand the error recovery procedures detailed in the flowcharts, the reader should be familiar with the sense information that corresponds to the individual I/O devices supported by this system. The latter part of the Physical Transient Programs section lists the devices supported by ERP and also the sense byte data associated with the device. In addition, a brief statement describing the actual ERP is made. Detailed procedures can be found in Appendix H (detailed flowcharts).

Note: Figure 44 is omitted intentionally.

Figure 45 illustrates the unit record equipment supported by ERP and also indicates the sense bits associated with each device.

CAUTION: Although the 2311 disk error recovery procedures are not an A-transient when the SYSTEM=DISK generation option is selected, the sense data and action-taken information is included here. The inclusion of this material consolidates the sense data in this section of the manual. The 2311 disk ERP are part of the supervisor nucleus. See Chart 17.

MESSAGE WRITER: The message writer is a group of seven A-transients that build error messages, issue the message, analyze operator responses, and select the proper exit.

Physical Attention Routines: The physical attention routines are three A-transients fetched by the supervisor when an attention interrupt has been determined. The attention key signals operator communication with the system. If the operator chooses to initiate a foreground program or to use the nonresident attention routine facilities, (other B-transients) the physical attention transients get the $$BATTNA root phase. If the operator is satisfying an operator intervention condition or canceling the job, the physical attention transients process the attention interrupt. When the physical attention routines are processing the interrupt, they perform parameter passing by using a common area called the interphase communications area. Figure 46 illustrates this area and its relationship to the entire A-transient area.

| Phase Name | Function | Program Level Chart ID |
|---|---|---|
| $$ANERRA | Error Recovery Monitor | 18 |
| $$ANERRB | | 18 |
| $$ANERRC | | 18 |
| $$ANERRD | Tape (2400) Error Recovery | 18 |
| $$ANERRE | | 18 |
| $$ANERRF | | 18 |
| $$ANERRL | | 18 |
| $$ANERRG | Data Cell (2321) Error Recovery | 18 |
| $$ANERRH | | 18 |
| $$ANERRI | | 18 |
| $$ANERRJ | | 18 |
| $$ANERRK | | 18 |
| $$ANERRM | Message Writer | 19 |
| $$ANERRN | | 19 |
| $$ANERRO | | 19 |
| $$ANERRP | | 19 |
| $$ANERRQ | | 19 |
| $$ANERRR | | 19 |
| $$ANERRU | Unit Record Error Recovery | 18 |
| $$ANERRV | | 18 |
| $$ANERRX | Paper Tape Error Recovery | 18 |
| $$ANERR9 | Optical Reader (1285) Error Recovery | 18 |
| $$ANERRZ | Physical Attention | 20 |
| $$ANERRY | | 20 |
| $$ANERR0 | | 20 |
| $$ANERR1 | Modify Communications Region | None (See Chart JY.) |

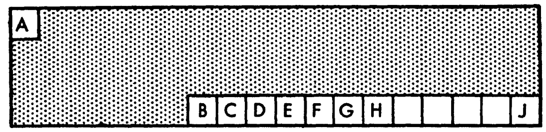
Figure 43. A-Transient Programs

| Device | Sense Bits | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1052 | X | X | X | X | | | | |
| 2501 | X | X | X | X | X | X | | |
| 2540R | X | X | X | X | X | | X | |
| 2520P | X | X | X | X | | | | X |
| 2540P | X | X | X | X | X | | X | |
| 1442P | X | X | X | X | | | | |
| 1442 R/P | X | X | X | X | X | X | | |
| 2520 R/P | X | X | X | X | X | X | | X |
| 1403 | X | X | X | X | X | | | X |
| 1443 | X | X | X | X | | | | X |
| 2671 | X | X | X | X | X | | | |

R = reader
P = punch

Figure 45. Unit Record Devices Supported by Device Error Recovery

504 - byte Physical Transient Area




The labels which are associated with these bytes are as designated below. Byte A is the first byte of the Physical Transient Area, Byte J is the last. Bytes B through H constitute the interphase communications area; when phases Z, Y and 0 are fetched or refetched, these bytes (B through H) are not overlaid and remain with information for the other phases.

| Byte | Label | Phase |
|---|---|---|
| A | IJBPAR1 | Z |
| | IJBPAR2 | Y |
| | IJBPAR3 | 0 |
| B | PARLTK | Z |
| | PARCOMM-1 | |
| C | IJBPAR1+493 | |
| | PARCOMM | Z |
| | IJBPAR2+493 | |
| | PARCOMM2 | Y |
| | PARCOMMC | Z |
| | PARCOMMD | Y |
| F | PARCOMMI | Z |
| | PARCOMMJ | Y |
| | PARCOMMC+3 | |
| | PARCOMMD+3 | |
| D, E, G, H | Addressed by incrementing or decrementing one of these labels. | |

Note: Bytes C, D and E are used to indicate the program(s) F1, F2 or BG, to be canceled.
Bytes F, G and H indicate the programs which use devices which require operator intervention.
Byte B indicates if a canceled program has fetched a logical transient.

Figure 46. Interface Communication Area (For Physical Transient Phases $$ANERRZ, $$ANERRY, and $$ANERR0)

## I/O ERROR RECOVERY PROCEDURES AND SENSE DATA

### 2400 Tape Error Recovery

CSW Bit 44--Channel Data Check
Action: Initial Selection--eight retries without respositioning. Read data transfer--no retries. Write data transfer--eight retries with repositioning. After stated number of retries, take equipment error exit (cancel).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 2--Bus Out Check
Action: If retry count is greater than seven (eight retries), take equipment error exit (cancel). If initial selection, take retry exit. Otherwise, perform repositioning and take retry exit.

Message: OP09 BUSOUT CHK.

Byte 0, Bit 3--Equipment Check
Action: Take equipment error exit (cancel).

Message: OP10 EQUIP CHK.

Byte 0, Bit 1--Intervention Required
Action: Check for Rewind and Unload (intervention required at device end). If yes, take continue exit; otherwise, take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 5 Overrun
Action: Allow eight retries, repositioning the tape. After eight retries, take equipment error exit (cancel).

Message: OP14 OVERRUN.

Byte 0, Bit 4 - Data Check
Action: 1. Read Commands--CCB option. If the record length is less

than twelve and Byte 1, Bit 0 (noise) is off, take retry exit. Otherwise, retry 100 times with repositioning (back space/forward space) performing CRC correction. Perform tape cleaning every eight retries. Tape cleaning consists of five backspaces and four forward spaces. For a read backward, tape cleaning is done by five forward spaces and four backspaces. Detection of load-point causes termination of the backspacing sequence. After 100 retries, take equipment error exit (cancel, ignore).

2. Write and WTM Commands--Backspace erase and retry fifteen times, then take equipment error exit (cancel). For write commands, if unit exception is present in the CSW, post it to the CCB (Byte 4, Bit 7).

3. Erase Gap Commands--After fifteen retries, without repositioning take equipment error exit (cancel).

Message: OP11 DATA CHECK.

Byte 0, Bit 7--Data Converter Check
Action: Take equipment error exit (cancel).

Message: OP30 CONVRT CHK.

Byte 0, Bit 0--Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT

Byte 1, Bit 4--Load Point and Byte 3, Bit 6-Backward status
Action: Take program check exit.

Message: OP29 BK INTO LP (Backward Command into Load Point).

Byte 1, Bit 7--Not Compatible
Action: Issue a rewind and unload command to the unit and then take operator intervention exit.

Message: OP32 NOT COMPAT.

CSW Bit 47--Chaining Check
Action: Allow eight retries, repositioning the tape. After eight retries, take equipment error exit (cancel).

Message: OP14 OVERRUN

Note: If an I/O error occurs during tape repositioning (other than backspace into Load Point on tape cleaning), equipment error exit (cancel) is taken with the message: OP20 ERR ON REC (Error During Recovery).

To achieve data check error recovery on write tape mark and erase gap commands, they must be command-chained to a no-op because the command code is not available for analysis when the error occurs (device end).

1052 Error Recovery
CSW Bit 44--Channel Data Check
Action: One retry, equipment error exit (cancel, retry, ignore).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: One retry, equipment error exit (cancel, retry, ignore).

Message: OP10 EQUIP CHK.

Byte 0, Bit 1--Intervention Required
Action: Execute audible alarm command and take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: One retry, equipment error exit (cancel, retry, ignore).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 0 - Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT.

1403-1443 Error Recovery
CSW Bit 44--Channel Data Check
Action: If initial selection, one retry--take equipment error exit (initial selection: cancel, retry; channel end: cancel, retry, ignore).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Take equipment error exit (cancel, ignore).

Message: OP10 EQUIP CHK.

Byte 0, Bit 5--Code General Storage Parity Error (1403 only)
Action: Take equipment error exit (cancel). UCS buffer must be reloaded.

Message: OP33 UCB PARITY.

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: If initial selection, one retry; otherwise, take equipment error exit. (Initial selection: cancel, retry; channel end: cancel, retry, ignore).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 7--Channel 9
Action: Post CCB, take continue exit.

Note: This test is main storage resident.

Byte 0, Bit 0--Command Reject
Action: If command code is UCS enable or inhibit data check, take continue exit; otherwise, take program check exit. This procedure allows UCS-oriented programs to operate on non-UCS hardware.

Message: OP18 COMM REJCT.

Byte 0, Bit 4--Data Check (1403 Only)
Action: Take equipment error exit (cancel, ignore).

Message: OP11 DATA CHECK.

1442 Error Recovery

CSW Bit 44--Channel Data Check
Action: If initial selection, one retry; then equipment error exit (cancel, retry). If data transfer, take operator intervention exit.

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Take operator intervention exit.

Message: OP10 EQUIP CHK.

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: If initial selection, do one retry; then take equipment error exit (cancel, retry). If data transfer, take operator intervention exit.

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4--Data Check
Action: Take operator intervention exit.

Message: OP11 DATA CHECK.

Byte 0, Bit 5--Overrun
Action: Take operator intervention exit.

Message: OP14 OVERRUN.

Byte 0, Bit 0--Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT.

CSW Bit 47--Chaining Check
Action: Take operator intervention exit.

Message: OP14 OVERRUN.

2501, 2520, 2540 Error Recovery

CSW Bit 44--Channel Data Check
Action: If initial selection, one retry; then equipment error exit (cancel, retry). If read data transfer, take operator intervention exit. If punch data transfer, one retry; then equipment error exit (cancel, retry).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Reader-- Take operator intervention exit. Punch--CCB option. Take equipment error exit (cancel, ignore). For 2520, Byte 0, Bit 7 indicates punch check.

Message: OP10 EQUIP CHK.

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: Do one retry; then take equipment error exit (cancel, retry). If the

device is a 2520, do not retry if this is not initial selection (cancel, retry).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4--Data Check (Can not occur on a 2520 punch)
Action: Take operator intervention exit.

Message: OP11 DATA CHECK.

Byte 0, Bit 5--Overrun (Cannot occur on 2540 or 2520 punch)
Action: Take operator intervention exit.

Message: OP14 OVERRUN.

Byte 0, Bit 0--Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT.

Byte 0, Bit 6--Unusual Command Sequence (2540 read only)
Action: Post CCB--take continue exit.

CSW Bit 47--Chaining Check (2501, 2520 read only)
Action: Take operator intervention exit.

Message: OP14 OVERRUN.

2671 Error Recovery

CSW Bit 44--Channel Data Check
Action: If initial selection, do one retry. Take equipment error exit (cancel).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Test CCB for ignore option (byte 2, bit 4) and if on, turn on byte 3, bit 1 of the CCB and take equipment error exit (cancel, ignore, retry). Otherwise, take operator intervention exit. See Note 2.

Message: OP10 EQUIP CHK.

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2 - Bus Out Check
Action: Do one retry; if error persists, take equipment error exit (cancel, retry).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4--Data Check
Action: Test CCB for ignore option (byte 2, bit 4) and if on, turn on byte 3, bit 3 of the CCB and take equipment error exit (cancel, ignore, retry). Otherwise, take operator intervention exit. See Note 1.

Message: OP11 DATA CHECK.

Byte 0, Bit 0--Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT.

Note: A record may not be partly on one tape and partly on another.

Note 1: When a data check occurs, the user's CCW is modified by the error routine to allow rereading of the last character. The data address will be the last character read (character in error) and the byte count is decreased by the number of valid characters read. If the CCB ignore option is chosen and the operator responds ignore, the I/O operation is dequeued and posted with the disaster-error bit on (CCB byte 2, bit 2) and 2671 data-check bit on (CCB byte 3, bit 3).

To read the rest of the record, the problem program (logical IOCS) should add one to the CCW data address and subtract one from the byte count to adjust for not rereading the bad character and then reissue the EXCP. The operator must backspace the tape two characters for retry (option retry or on the A-type message when ignore is not allowed). If the operator chooses the ignore option (the character in error is not to be reread), he must backspace the tape one character if the load key was pressed to free the tape or if the character preceding the character under the read head is an EOR (End-of-Record). Otherwise, no manual intervention is required for the ignore option. The ignore option is available to the operator whenever the user specifies any of the DTFPT ERROPT entry options.

Note 2: When an equipment check occurs, the operator must reposition the paper tape to the beginning of the record in error to perform the retry operation. The device must not be readied until this repositioning has been performed. If the ignore option is available to the operator, he can exercise this option by repositioning the tape to the beginning of the next record on the tape and then responding ignore on the 1052 keyboard. The ignore option is available to the operator whenever the user specifies any of the DTFPT ERROPT entry options.

2311 DASD Error Recovery

CSW Bit 44--Channel Data Check
Action: One retry; then equipment error exit (cancel, retry).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3 - Equipment Check
Action: Take equipment error exit (cancel, retry).

Message: OP10 EQUIP CHK.

Byte 1, Bit 4 - No Record Found*
Action: Test for Byte 1, Bit 6 (Missing Address Marker). If present, execute restore command and take retry exit. After ten retries, take equipment error exit (cancel, retry). If not present, read Home Address and compare to user's Seek Address. If equal, post No Record Found to the CCB and take continue exit. If not equal, treat as a Seek Check.

Messages: OP21 NRF - MADDMK (No Record Found/Missing Address Marker)
OP15 SEEK CHECK (Home Address unequal to Seek Address)

Byte 0, Bit 7--Seek Check
Action: If Byte 0, Bit 0 (command reject) is on, take program check exit. Otherwise, execute restore command and take retry exit. After ten retries, take equipment error exit (cancel, retry).

Messages: OP26 INVAL SEEK (Seek Check/Command Reject) OP15 SEEK CHECK.

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: If retry count greater than nine, take equipment error exit (cancel, retry); otherwise, take retry exit.

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4 - Data Check*
Action: CCB options (all data checks, data check on read or verify). If retry count is greater than nine, take equipment error exit (cancel, retry); otherwise, take retry exit. After nine retries, post data check on count to CCB, if present; otherwise, post data check. If command code is verify, post verify error to CCB.

Messages: OP12 VERIFY CHK (Data Check on Verify Command).
OP11 DATA CHECK (Data Check/not Data Check on Count or Verify).
OP16 DTA CHK CT (Data Check on Count).

Byte 0, Bit 5--Overrun
Action: If retry count is greater than nine, take equipment error exit (cancel, retry); otherwise, take retry exit.

Message: OP14 OVERRUN.

Byte 1, Bit 6 - Missing Address Markers*
Action: If retry count is greater than nine, take equipment error exit (cancel, retry); otherwise, take retry exit.

Message: OP13 ADDR MRKER.

Byte 0, Bit 0 - Command Reject
Action: Check for Byte 1, Bit 5 (File Protect); in either case, take program check exit.

Messages: OP18 COMM REJCT.
OP17 FILE PROT.

Byte 0, Bit 6--Track Condition Check
Action:
1. Read Home Address and R0 in the error recovery routine and move CCHH from R0 to Seek command executed below.
2. If alternate track: update seek address to the next track address. If the track address equals 10, treat it as End of Cylinder; otherwise, proceed to step 3.
3. Set up the channel program: Seek, Read Home Address (with skip bit on), TIC to CSW address minus eight. Execute this channel program in error recovery. At channel end, exit to channel scheduler CSW processing routine. If DASD file protection is present, set the appropriate file mask following Seek.

Byte 1, Bit 1--Track Overrun
Action: Post track overrun to the CCB and take continue exit.

Byte 1, Bit 2--End of Cylinder
Action: Post End of Cylinder to the CCB and take continue exit.

Byte 1, Bit 5--File Protect
Action: Take program check exit.

Message: OP17 FILE PROT.

CSW Bit 47--Chaining Check
Action: If retry count is greater than nine, take equipment error exit (cancel, retry); otherwise, take retry exit.

Message: OP14 OVERRUN.

*For these errors, Home Address is read and the track address is provided for the error message. For other errors, the track address is obtained from the user seek address if error occurs during channel program execution.

Note: If the 2311 error routine gets an error while trying to execute a Restore command or Read Home Address or R0, equipment error exit is taken with retry and cancel options with the message: OP20 ERR ON REC (Error During Recovery).

2321 DASD Error Recovery

CSW Bit 44--Channel Data Check
Action: One retry; then equipment error exit (cancel, retry).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Take equipment error exit (cancel, retry).

Message: OP10 EQUIP CHK.

Byte 1, Bit 4--No Record Found
Action:
1. If Byte 1, Bit 6 (missing Address Markers) is present, go to step 2. Otherwise, go to step 6.

2. If retry count is less than 3, issue a Restore command and go to step 5.

3. If retry count is equal to 3, issue a Read Home Address to the first and last tracks of the cylinder. If neither is successful (unit checks), take equipment error exit (cancel, retry). Otherwise, go to step 4.

4. If retry count is equal to 15, take equipment error exit (cancel, retry). Otherwise, go to step 5.

5. Increment retry count and take retry exit.

6. Issue a Read R0 and compare CCH to user's Seek Address. If equal, post No Record Found to the CCB and take continue exit. Otherwise, go to routine for Seek Check (alone).

Messages: OP15 SEEK CHECK (No Record Found/R0 unequal to Seek Address).
OP23, BLNK STRIP (Step 3, cannot read Home Address).

OP21 NRF - MADDMK (Step 4, 15 retries).

Byte 0, Bit 7--Seek Check
Action: If Byte 0, Bit 0 (command reject) is present, take program check exit. If Byte 1, Bit 6 (missing Address Markers) is present, take operator intervention exit. Otherwise, issue a Seek to BB1111, a Seek to BB2222, and take retry exit. After ten retries, take equipment error exit (cancel, retry).

Messages: OP26 INVAL SEEK (Seek Check/Command Reject).

OP22 BALST CELL (Seek Check/Missing Address Markers).

OP15 SEEK CHECK (Seek Check alone).

Byte 0, Bit 1--Intervention Required
Action: Take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 2--Bus Out Check
Action: Take retry exit. After 15 retries, take equipment error exit (cancel, retry).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4--Data Check**
Action:
1. If retry count is less than eight, go to step 5.

2. If retry count is equal to

226, take equipment error exit (cancel, retry).

3. If retry count is an even number, issue a Seek to X-X-X-4-19 (last track of strip) and a Seek to X-X-X-0-0 (first track of strip). Perform this operation eight times. Then proceed to step 4.

4. If retry count is any multiple of 32 (32, 64, 96, . . . ), issue a Seek to next lower strip. (If this is the lowest strip - 00000 - seek the next higher strip.) Proceed to step 5.

5. Increment retry count and take retry exit.

Messages: OP11 DATA CHECK (Data Check/not Data Check on Count or Verify).

OP12 VERIFY CHK (Data Check on Verify Command).
OP16 DTA CHK CT (Data Check on Count).

Byte 0, Bit 5--Overrun
Action: Take retry exit. After 15 retries, take equipment error exit (cancel, retry).

Message: OP14 OVERRUN.

Byte 1, Bit 6--Missing Address Markers**
Action: Perform action indicated under Data Check just described.

Message: OP13 ADDR MRKER.

Byte 0, Bit 0--Command Reject
Action: Check for Byte 1, Bit 5 (file protect); in either case, take program check exit.

Messages: OP17 FILE PROT (Command Reject/File Protect).
OP18 COMM REJCT (Command Reject alone).

Byte 0, Bit 6--Track Condition Check
Action: 1. Read Home Address and R0 and move CCHH from R0 to Seek command executed CCHH from R0 to Seek command executed below.

2. If alternate track: Update Seek Address to the next track address. If track address equals 20, treat it as End of Cylinder; otherwise, proceed to step 3.

3. Set up the channel program: Seek, Read Home Address (with skip bit on), TIC to CSW command address minus eight (last CCW executed). Execute this channel program in error recovery. At channel end, exit to channel scheduler CSW processing routine. If DASD file protection is present, set file mask (inhibit long Seeks) following the seek.

Byte 1, Bit 1--Track Overrun
Action: Post track overrun to the CCB and take continue exit.

Byte 1, Bit 2--End of Cylinder
Action: Post End of Cylinder to the CCB and take continue exit.

Byte 1, Bit 5--File Protect
Action: Take program check exit.

Message: OP17 FILE PROT.

CSW Bit 47--Chaining Check
Action: Take retry exit. After 15 retries, take equipment error exit (cancel, retry).

Message: OP14 OVERRUN.

Note: If the 2321 Error Routine gets an error while trying to execute a Restore command, a Seek command (data-check procedure), or a Read Home Address or a Read R0, equipment error exit is taken with retry and cancel options with the message: OP20 ERR ON REC (Error During Recovery).

**For these errors, Home Address is read and the track address is provided for the error message. For other conditions, the track address is obtained from the user's initial Seek address if the error occurs during channel program execution.

1285 Optical Reader

CSW Bit 44--Channel Data Check
Action: One retry; then take equipment error exit (retry, cancel).

Message: OP28 CHAN DTCHK.

Byte 0, Bit 3--Equipment Check
Action: Post Byte 3, of CCB and then continue exit. (See Note.)

Byte 0, Bit 1--Intervention Required
Action: Test for Byte 1, Bit 6 (Non-recovery)--if present, post Byte 3, Bit 4 of the CCB. This indicates that the error is passed back to the problem program. Exit via equipment error.

Message: OP35 NON RECOV. If Byte 0, Bit 6 is not present, take operator intervention exit.

Message: OP08 INTERV REQ.

Byte 0, Bit 6--Nonrecovery
Action: Post Byte 3, Bit 4, of CCB and take continue exit.

Byte 0, Bit 2--Busout Check
Action: One retry; then equipment error exit (retry, cancel).

Message: OP09 BUSOUT CHK.

Byte 0, Bit 4--Data Check
Action: Post Byte 3, Bit 0, of CCB and take continue exit. (See Note.)

Byte 0, Bit 5--Overrun
Action: Four retries; then equipment error exit (retry, cancel).

Message: OP14 OVERRUN.

Byte 0, Bit 0--Command Reject
Action: Take program check exit.

Message: OP18 COMM REJCT.

CSW Bit 47--Chaining Check
Action: Four retries; then equipment error exit (retry, cancel).

Message: OP14 OVERRUN.

Byte 0, Bit 7--Keyboard Correction
Action: Post Byte 3, Bit 1, of CCB and take continue exit.

Note: Data Check and Equipment Check, which indicate unreadable character and unreadable line, respectively, are retried by Logical IOCS in an attempt to correct the error.

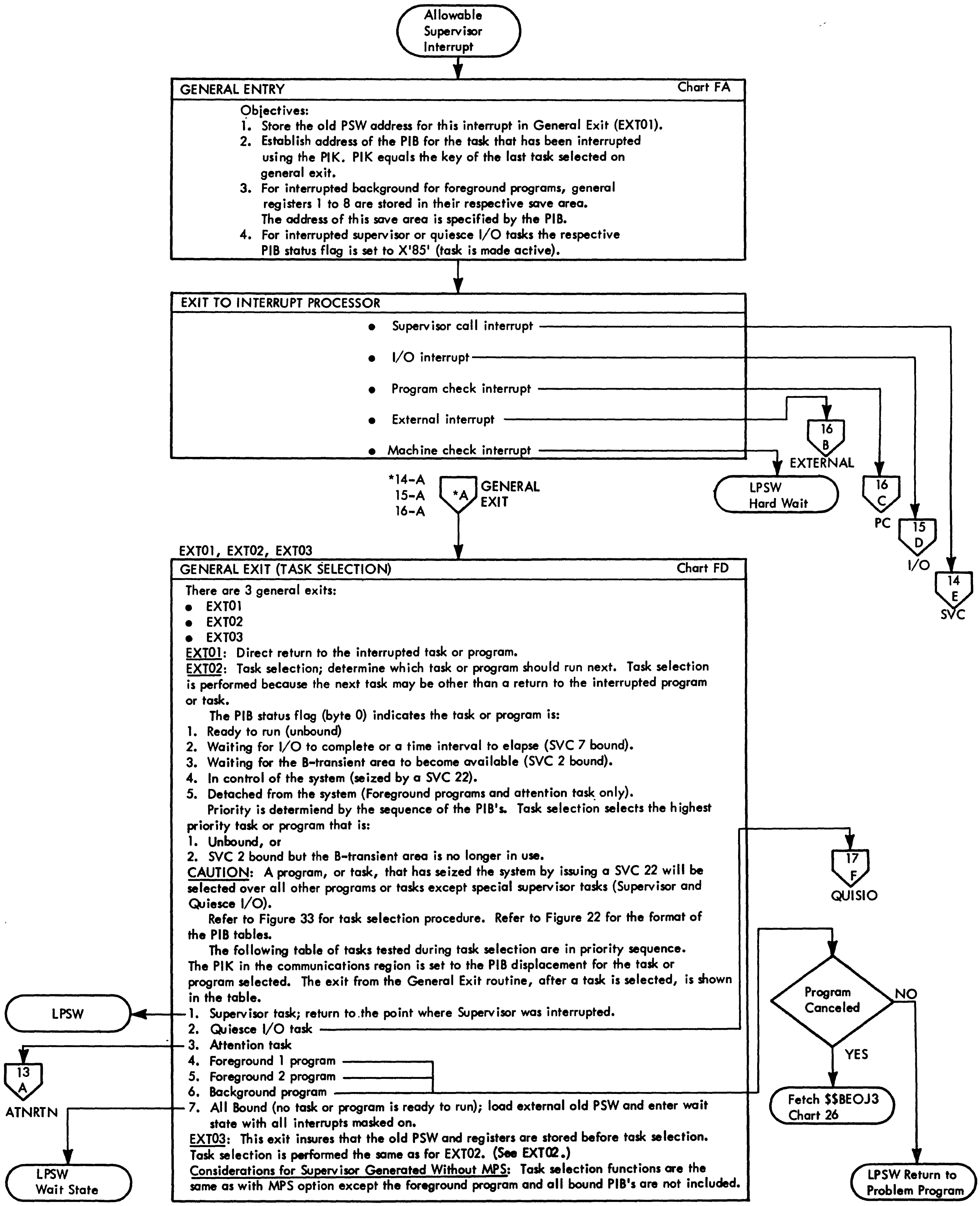



Chart 12. Supervisor General Entry, General Exit, and Processor Exit

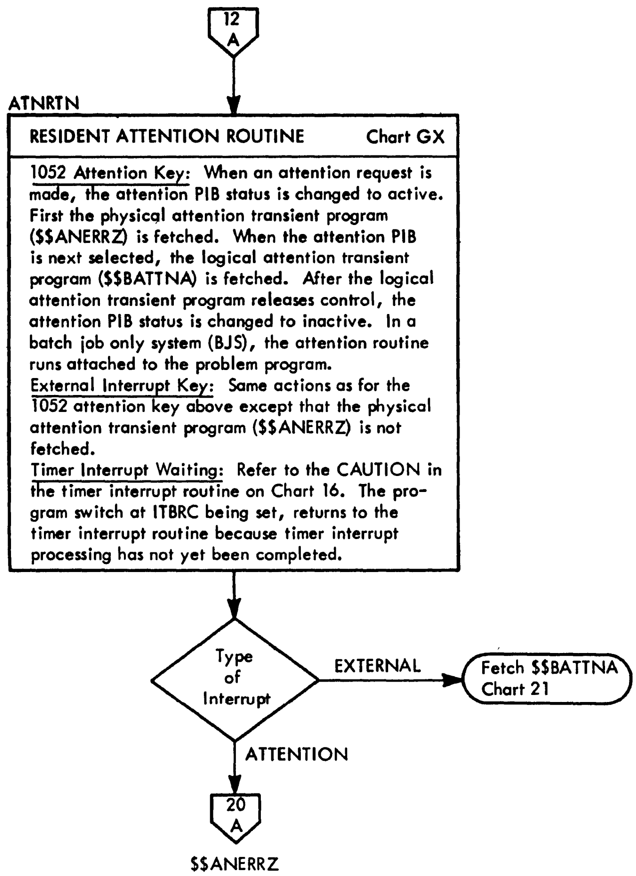



Chart 13. Resident Attention Routine

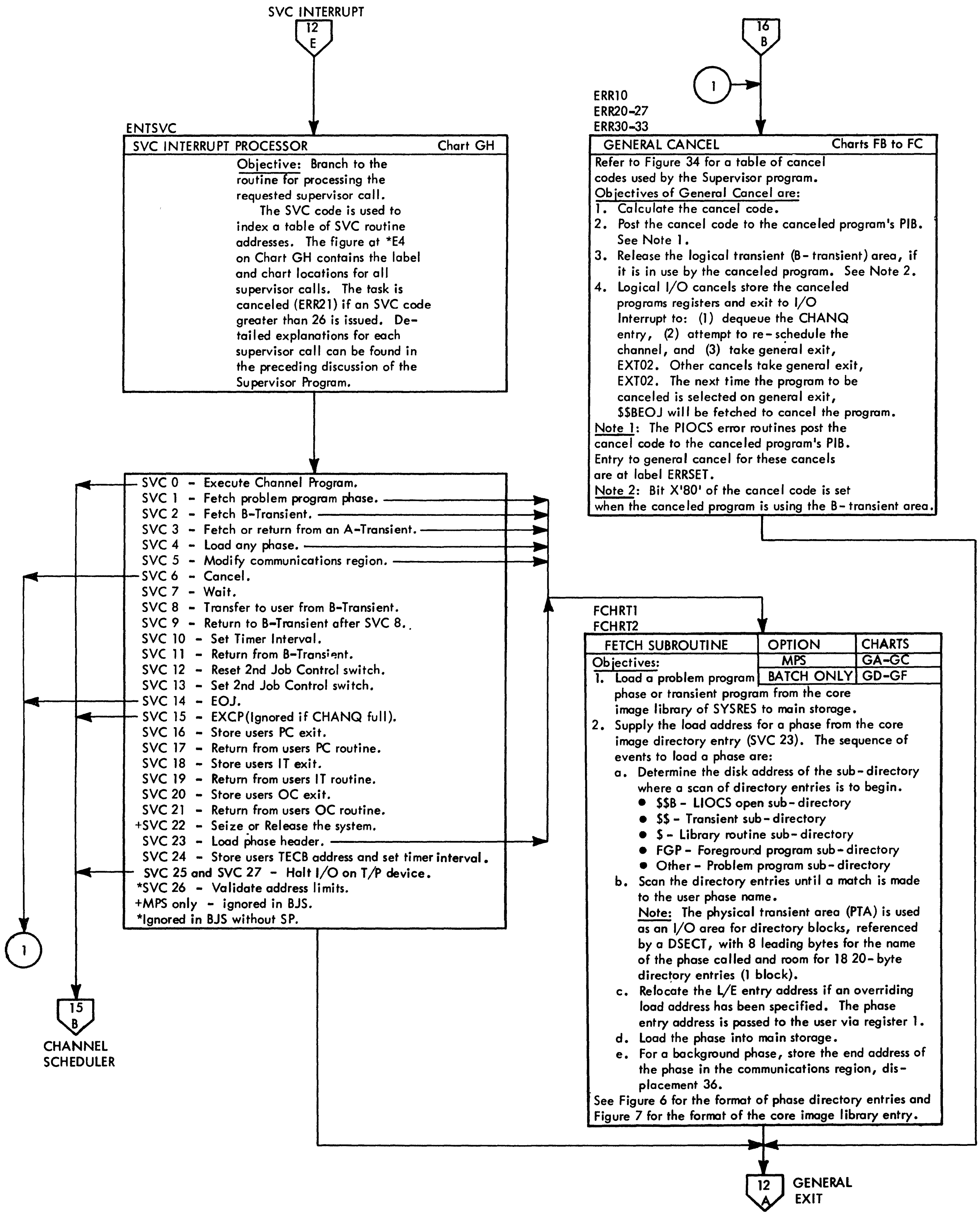



Chart 14. SVC Interrupt Processor, General Cancel, and Fetch

I/O INTERRUPT

*12-D
17-D

*D

INTRTN

**I/O INTERRUPT PROCESSOR** — Charts FN to FS

I/O interrupts are caused by termination of an I/O operation or operator intervention on the I/O device. Each I/O interrupt is processed in this manner:

1. Find PUB for the device that causes the I/O interrupt. The device last started is tested first. If the device that caused the I/O interrupt is not found, an attempt is made to re-schedule the channel. See Note 1.
2. Exit to the channel appendage routine if all of the following conditions exist:
   a. Supervisor has been generated with tele-processing option (TP=BTAM).
   b. The device causing the I/O interrupt has I/O pending (CHANQ pointer is not X'FF').
   c. The user has specified an appendage routine address in the CCB.
   d. The program has a flag on in its PIB indicating an appendage routine address in the CCB is allowed.
3. Evaluate CSW. Refer to Figure 41 for CSW testing on an I/O interrupt.
4. If status is attention from a 1052 the attention routine PIB flag is posted to include the attention routine in task selection and general exit, EXT03, is taken.
5. If the device status indicates that the channel program has been successfully completed:
   a. Post traffic bit in the CCB.
   b. Dequeue the CHANQ entry if device end posting was not requested. (See Figure 21 for format of the CHANQ table).
   c. Exit to START I/O if there is more I/O pending on this device. If not, attempt to re-schedule the channel. See Note 1.

Note 1: Re-schedule the channel, with MPS: The PUB's in the channel list are scanned, beginning with the PUB following the last one started, and the first one with I/O pending will be the next PUB started. If the end of the channel list is reached, the scan continues from the first PUB in the channel list. If no I/O is pending in the channel go to the general exit routine.

This method is used to insure the sharing of the channel between programs when operating in multi-programming and is referred to as a rotating PUB scan.

Re-schedule the channel, batch only: Same as with MPS except the channel list is always scanned beginning with the first PUB in the channel list (FOCL).

Error — YES: 17-B UNIT CHECK; NO: I/O Pending

I/O Pending — YES: 1; NO: 12-A GENERAL EXIT

14-B

SVC15, SVC00

**CHANNEL SCHEDULER** — Charts FF to FH

SVC 0 and SVC 15: The objective of the channel scheduler is to build an entry for the channel program in the CHANQ, LUBID, and REQID tables. Refer to Figure 35 for a sample of a device queue in the CHANQ table. Refer to Figure 21 for the format of the CHANQ, LUBID, and REQID tables. Refer to Figure 36 for the format of the CCB (channel command block).

Communications and status bytes 2 through 6 of the CCB are reset before making the CHANQ entry.

Exit from channel scheduler is to START I/O if:

1. This I/O request for the device is the first in queue (CHANQ pointer in the PUB contained the null code, X'FF') and
2. This device is not busy (busy flag in the PUB is reset).

If the above conditions are not met, general exit, EXT01, is taken.

Device Busy — YES: 12-A GENERAL EXIT; NO: ACTUAL I/O

1

GIOADR

| ACTUAL I/O | OPTIONS | CHARTS |
|---|---|---|
| System I/O Files | MINIMUM | FJ |
| Assigned to a 2311: | MAXIMUM | FK-FM |

SYSIPT, SYSRDR, SYSPCH, and SYSLST are System I/O files. Before a start I/O is issued to these files, the search address in the user CCW chain is checked against the current address in the DIB. Refer to Figure 23 for the format of the DIB tables.

DASD File Protect: Extent information must be present in the JIB's for all files assigned to a 2311 or 2321 (data cell) except system I/O files assigned to a 2311 and SYSRES. Any access to DASD file must fall within the extents specified in the JIB's. Refer to figure 24 for the format of the JIB's.

Start I/O: If the device is burst-multiplex, but the channel is presently operating in burst-multiplex mode, take general exit, EXT01. An SIO command is issued to start the device, if the channel is available. The alternate channel is tested if the device is switchable. Before using the SIO command, the correct CCW address must be set in the CAW.

1. For 7-track tape, CAW= set mode CCW.
2. For SYSLOG, CAW = Prefix ID CCW.
3. For DASD file protected devices, CAW equals CCW-chain in the supervisor, which TIC's to the user's chain.
4. For all others, CAW= CCW specified by CCB.

If the device receives a normal start (condition code 8 set on SIO) post the device busy flag in the PUB and take general exit, EXT01.

12-A

GENERAL EXIT

Chart 15. I/O Interrupt Processor and Channel Scheduler

12 C PROGRAM CHECK INTERRUPT

ENTPCK

**PROGRAM CHECK INTERRUPT PROCESSOR**

The program is canceled (ERR20) if:

1. Supervisor was generated without user PC routine option.
2. The user has not previously supplied the address of a PC routine by issuing a STXIT PC macro.
3. The user has specified an invalid save area.
4. The program check occurs while a B-Transient is in operation.
5. The program check occurs while the user's PC routine is in operation (routine in use).

| OPTION | CHARTS |
|---|---|
| NO PC RTN | FB |
| PC, NO IT, OC, OR SP | GQ |
| PC WITH IT, OC, OR SP | GR |

To exit to the user's PC routine:

1. Save the PC old PSW and problem program general registers in the user-supplied save area (72 bytes).
2. Store the address of the user PC routine in the PC old PSW.
3. Branch to general exit (EXT03).
   The user's PC routine will be executed when this task is selected in the general exit routine. Return from the user's PC routine must be with an EXIT PC macro.

See Figure 25 for PC option tables.

User Exit Set — NO: 14 B GENERAL CANCEL; YES: 12 A GENERAL EXIT

EXTERNAL INTERRUPT

12 B

Type — SIGNAL: 12 A GENERAL EXIT; KEY: 1; TIMER: 2

1

**EXTERNAL INTERRUPT KEY**

The attention routine PIB is posted to include the attention routine in task selection. The non-resident attention routine ($$BATTNA) processes the request when the attention task is selected in general exit.

| OPTION | CHART |
|---|---|
| NO IT OR OC | FE |
| WITH IT OR OC | GS |

Note: If supervisor was generated without user OC (operator communications) routine option, the interrupt is ignored.

2

**EXTERNAL INTERRUPT, TIMER**

The timer (SYSTIMER) is reset to its maximum value (X'FFFFFFFF') and the system time of day (SYSTOD) is updated to the time that the next timer interrupt will occur (current time + 15.5 hours).

| OPTION | CHART |
|---|---|
| NO IT OR OC | FE |
| WITH IT OR OC | GS |

Note: Current time of day is calculated by subtracting the remaining interval (SYSTIMER) from SYSTOD.

The timer interrupt is ignored if:

1. Supervisor was generated without interval timer feature option.
2. The timer supported program has been canceled.
3. The timer interrupt occurred while the user's IT routine was in operation (routine in use).

With User TECB Address: The traffic bit is posted in the user's TECB (timer event control block) and the timer supported program is included in task selection. The TECB address is cleared in the IT option table.

With User IT Routine: To exit to the user's IT routine:

1. Save the external old PSW and timer-supported-program registers in the user-supplied save area (72 bytes).
2. Store the address of the user IT routine in the external old PSW.
3. Branch to general exit (EXT03).

The user IT routine will be executed when this task is selected in the general exit routine. Return from the user's IT routine must be with an EXIT IT macro.

See Figure 25 for the IT option table. Refer to Figure 36 for the format of the CCB (TECB has the same format as the CCB. Only the traffic bit X'80' in byte 2 is used).

CAUTION: If a B-Transient is operating for the timer supported program when the timer interrupt occurs for a user IT routine, the exit is deferred. The attention routine PIB is posted to include the attention routine in task selection when the B-transient area is released. A switch (ITBRC) is set in the resident attention routine to continue timer interrupt processing when the attention task is selected on general exit.

12 A GENERAL EXIT

Chart 16. Program Check and Internal Interrupt Routines

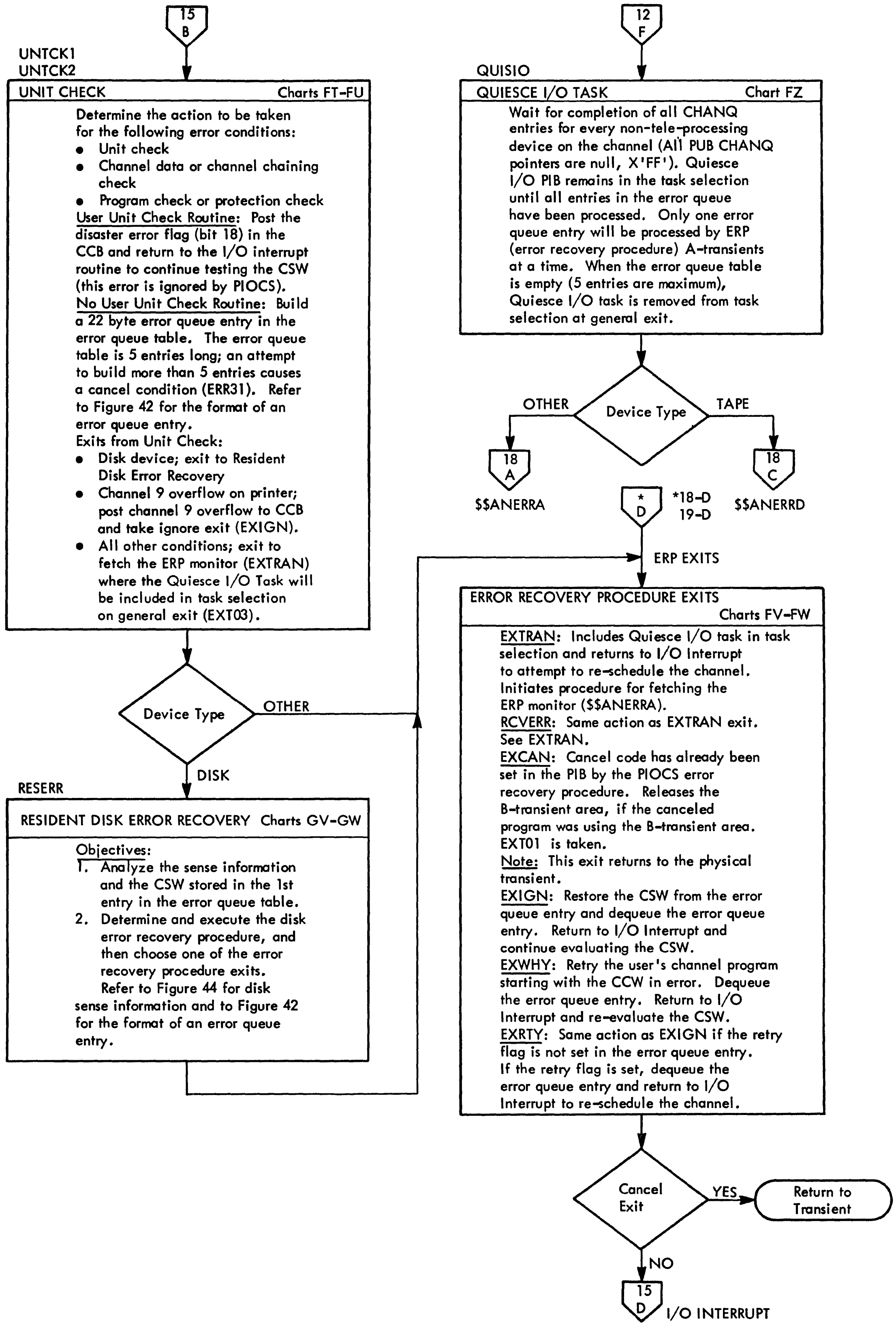



Chart 17. Unit Check, Resident ERP, and Quiesce I/O Routines

17 A

$$ANERRA

**ERP MONITOR** Charts HA-HD

The ERP monitor is composed of two A-transient programs. These are:

1. $$ANERRA - determine device type and identify 2311 unrecoverable errors.
2. $$ANERRB - continue identification of 2311 unrecoverable errors.

Determine device type to be:

- Data cell (2321) - fetch $$ANERRG
- Unit record - fetch $$ANERRU
- Paper tape - fetch $$ANERRX
- Disk (2311) - analyze CSW and sense data to identify the unrecoverable error, post message code to the error queue entry, and fetch the message writer ($$ANERRM).

Refer to I/O Error Recovery Procedures for disk sense data and to Figure 42 for the format of the error recovery block.

- Optical reader - fetch $$ANERR9

17 C

$$ENERRD

**TAPE ERROR RECOVERY** Charts HE-HM

Tape error recovery is accomplished by three A-transient programs. These are:

1. $$ANERRD - error analysis and selected error procedures.
2. $$ANERRE - selected error procedures.
3. $$ANERRF - data check procedure.

Objectives:

1. Analyze the sense information and the CSW stored in the 1st entry in the error queue table.
2. Determine the tape error recovery procedure, and choose one of the error recovery procedure exits.

Refer to I/O Error Recovery Procedures for data sense information and to Figure 42 for the format of the error recovery block.

$$ANERRG

**DATA CELL ERROR RECOVERY** Charts HN-HQ

Data cell error recovery is accomplished by five A-transient programs. These are:

1. $$ANERRG - error analysis.
2. $$ANERRH - track condition check procedure.
3. $$ANERRI - data check and missing address marker procedures.
4. $$ANERRJ - no record found and seek check procedures.
5. $$ANERRK - continuation of number 3 above.

Objectives:

1. Analyze the sense information and CSW stored in the 1st entry in the error queue table.
2. Determine the data cell error recovery procedure, and choose one of the error recovery procedure exits.

Refer to I/O Recovery Procedures for data cell sense information and to Figure 42 for the format of the error recovery block.

$$ANERRU

**UNIT RECORD ERROR RECOVERY** Charts JK-JL

Unit record error recovery is accomplished by two A-transient programs. These are:

1. $$ANERRU - unit record test cannot occur conditions, channel data check, and 1052 errors.
2. $$ANERRV - 1403, 1442, 1443, 2501, 2520 and 2540 error recovery procedures.

Objectives:

1. Analyze the sense information and the CSW stored in the 1st entry in the error queue table.
2. Determine the unit record error recovery procedure, and choose one of the error recovery procedure exits.

Refer to I/O Error Recovery Procedures for unit record sense information and to Figure 45 for unit record devices supported. Refer to Figure 42 for the format of the error recovery block.

$$ANERRX

**PAPER TAPE ERROR RECOVERY** Charts JP-JQ

Paper tape error recovery is accomplished by one A-transient program, $$ANERRX.

Objectives:

1. Analyze the sense information and the CSW stored in the 1st entry in the error queue table.
2. Determine the paper tape error recovery procedure, and choose one of the error recovery procedure exits.

Refer to I/O Error Recovery Procedures for paper tape sense information and to Figure 42 for the format of the error recovery block.

$$ANERR9

**OPTICAL READER ERROR RECOVERY** Charts JR-JRB

Optical Reader error recovery is accomplished by one A-transient program, $$ANERR9.

Objectives:

1. Analyze the sense information and the CSW stored in the error queue table.
2. Determine the optical reader error recovery procedure and choose one of the error recovery procedure exits.

Refer to I/O Error Recovery Procedures for optical reader sense information and to Figure 42 for the format of the error recovery block.

17 D

Chart 18. Physical Transients ERP

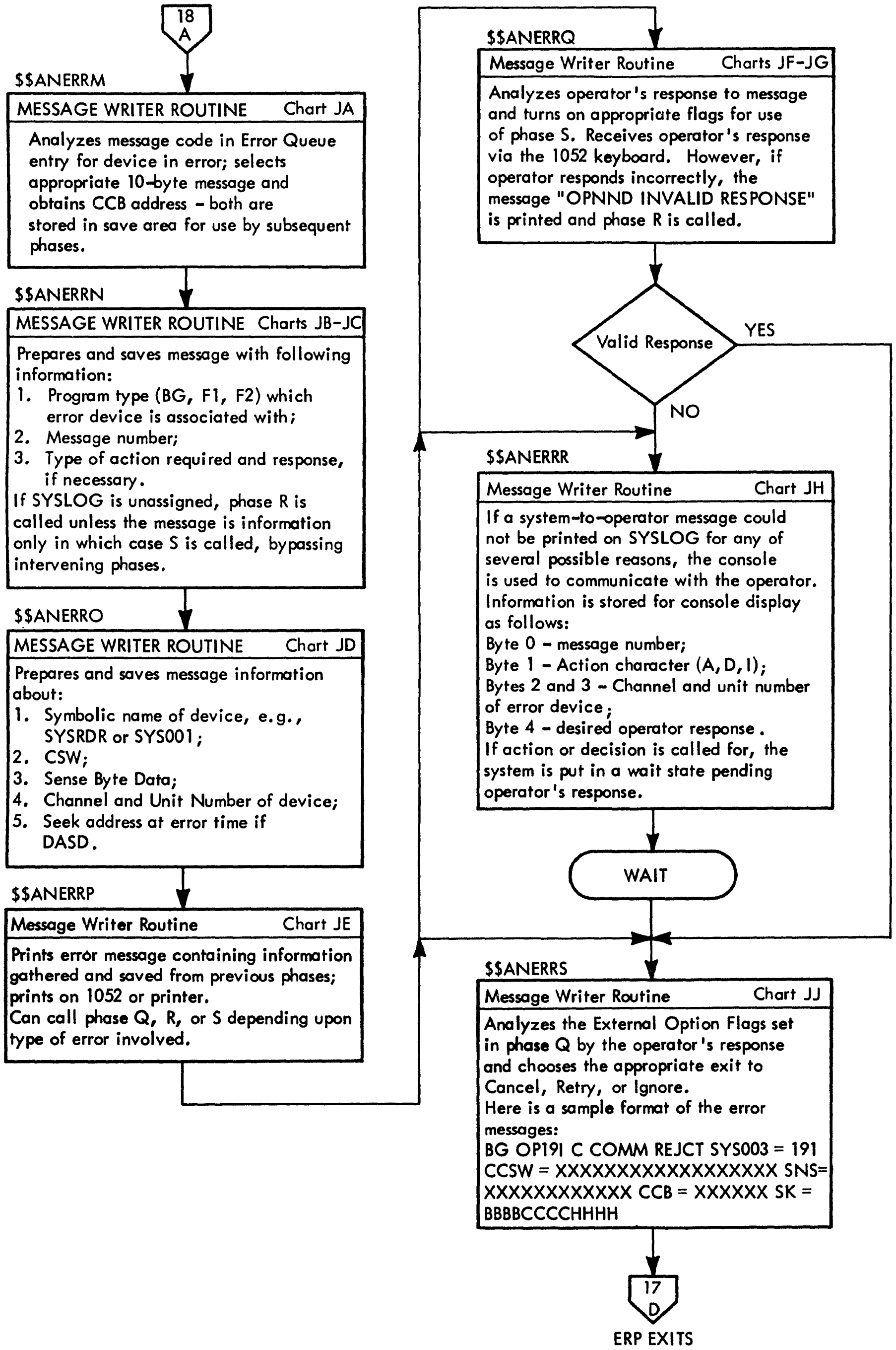



Chart 19. Physical Transients Message Writer

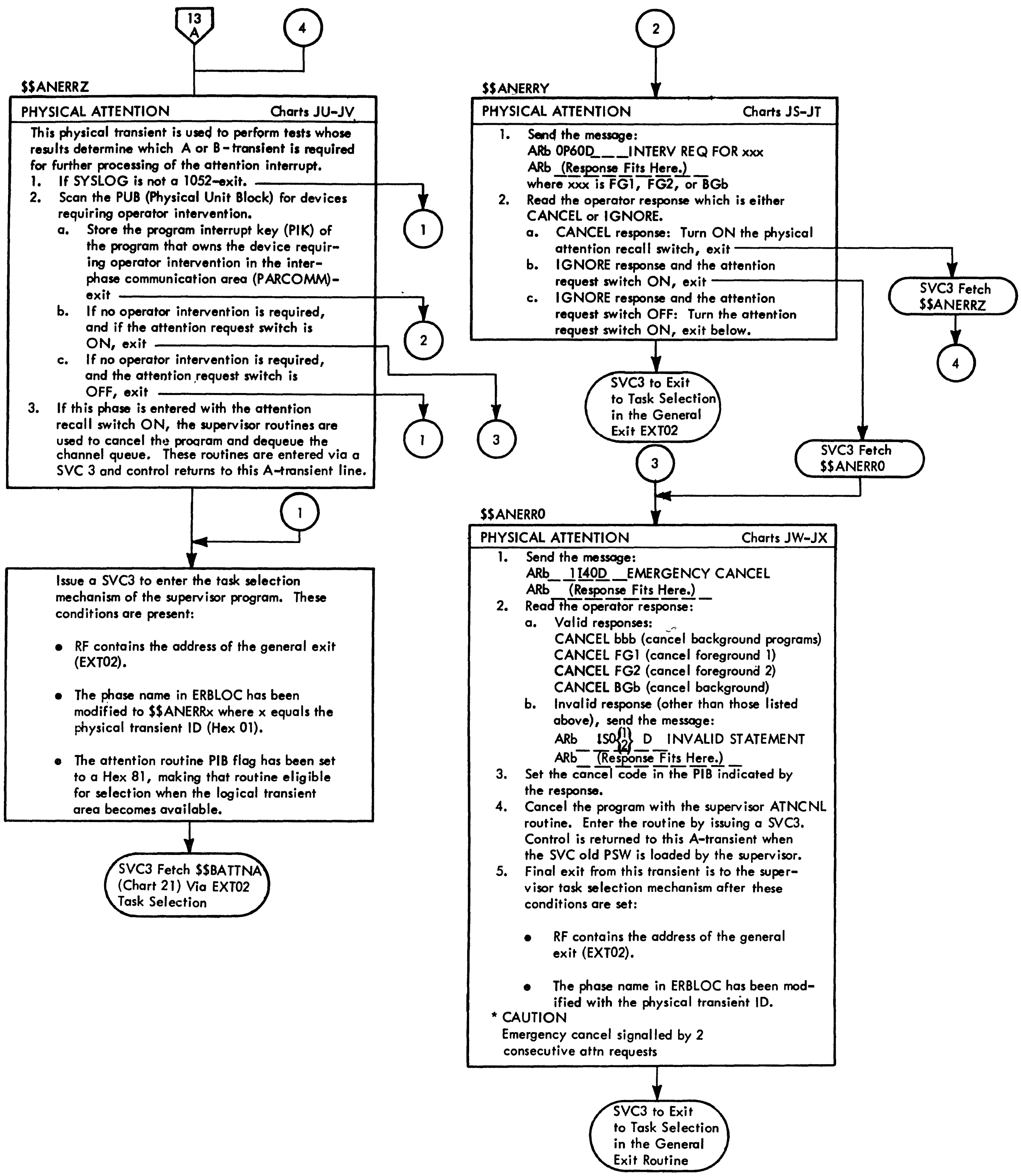



Chart 20. Physical Transients--Physical Attention Routine

SUPERVISOR B-TYPE TRANSIENT PROGRAMS (CHARTS 21 THROUGH 30)

B-transient programs are infrequently-used routines; therefore they are not resident in main storage. They may be fetched or loaded from the core image library when needed. The B-transients occupy an area of 1200 bytes (1000 bytes in TOS) referred to as the Logical Transient Area (LTA).

A SVC 2 instruction loads and executes a B-transient phase. A prefix of $$B to the name of a phase identifies it as a B-transient. The normal return to supervisor nucleus control is an SVC 11, but some of the transient programs exit by fetching another B-transient with an SVC 2. In the latter case, the calling B-transient will be overlaid by the transient being fetched.

Register 1 is loaded with the address of the transient name prior to issuing the SVC 2, so the fetch or load routine has access to the name for purposes of searching the disk directories or tape records for the desired transient.

B-Transient Grouping

The supervisor B-transient programs can be grouped by the various functions performed. These functions are nonresident attention routine, foreground program initiator, and program terminator.

INITIATOR

Foreground Initiator (Charts 21-23) includes these B-transients:

$$BATTNA Chart KA

$$BATTNC Chart KG

$$BATTNH Chart KT

$$BATTNI Chart KV

$$BATTNJ Chart LH (see Figure 48)

$$BATTNK Chart LP

$$BATTNL Chart LX

$$BATTNM Chart MC

Program terminator (Charts 26-30) includes these B-transients:

$$BEOJ Chart NA (see Figure 50)

$$BEOJ3 Chart ND

$$BTERM Chart NE

$$BPCHK Chart NT

$$BILSVC Chart NN

$$BEOJ2 Chart NL

$$BEOJ1 Chart NJ

$$BPSW Chart NR

$$BDUMP Chart NV

$$BDUMPF Chart NX

$$BDUMPB Chart PB

$$BDUMPD Chart PG

$$BPDUMP Chart PL

$$BPDUM1 Chart PM

Nonresident Attention (Charts 21, 24, and 25) includes these B-transients:

$$BATTNA Chart KA

$$BATTNB Chart KE

$$BATTNC Chart KG

$$BATTND Chart KJ (see Figure 47)

$$BATTNE Chart KN

$$BATTNF Chart KQ

$$BATTNG Chart KS

$$BATTNH Chart KT

$$BATTNN Chart MH

$$BSYSWR Chart PS

INITIATION (FIGURE 49)

Foreground programs are initiated by the operator through the 1052 assigned to SYSLOG. The operator may initiate a foreground program whenever an allocated foreground area does not contain a program.

The operator initiates a foreground program by pressing the 1052 request key. The attention interrupt causes control to be given to the system's Attention routine.

Note: If the transient area is in use by a routine other than the Attention routine, the attention interrupt is posted and serviced when the transient area becomes available.

The Attention routine reads a command from the operator. The command START (F1 or F2) indicates a foreground program is to be initiated. The Attention routine determines if the area specified is allocated and does not contain a program; if so, it transfers control to the foreground initiator. Otherwise, the operator is notified that an invalid command has been given.

The foreground initiator reads subsequent commands required to initiate the program. These commands are used primarily to specify I/O assignments and label information. When an I/O assignment is attempted, the following verification is made:

1. The symbolic unit is of the class SYSnnn.

2. The symbolic unit is contained within the number specified for the area at system generation.

3. If the symbolic unit is to be assigned to a non-DASD, the device must not be in use by the other foreground program nor can it be assigned to a background job either as a standard, temporary, or alternate unit.

The label information for each file in the job is written on SYSRES as a label information block for later retrieval and processing by the data management routines. A main storage area for label information is required under the same conditions as for background jobs and is calculated and reserved by the initiator.

When the EXEC statement is encountered, the foreground initiator directs the supervisor to provide loading information for the program to be invoked. If the program has not been cataloged, the operator is notified by the initiator. He may correct the command (for example, if the name was misspelled) or cancel the initiation.

After the loading information is received, the initiator checks to determine if a self-relocating program is to be loaded. This is determined by the load address being zero. The foreground initiator directs this program to be loaded following the label information area. It also calculates the entry point to the program by adding the address at which it will be loaded to the previously-calculated entry point (derived when the program was linkage edited and cataloged onto the system). A non-self-relocating program will be directed to be loaded utilizing the information derived when the program was cataloged.

Diagnostics, such as the program being outside the limits of the foreground area, are not performed by the initiator, but are performed by the Supervisor when the program is loaded and causes the program to be terminated.

When initial control is given to the user's foreground program, register 2 contains the address of the uppermost byte of storage available to this program. This may be used to calculate the total storage available to the program. A foreground program can dynamically determine the storage available to it by storing the contents of this register for later reference.

Note that a program capable of either foreground or background operation (with proper linkage editing) can utilize the same programming to determine its storage allocation independently of its actual area assignment.

TERMINATION (FIGURE 50)

A foreground program is terminated under its own control by issuing an EOJ, DUMP, or CANCEL macro or through operator action or a program error or certain I/O failures. When a foreground program is terminated, the following actions are taken:

1. All I/O operations that the program has requested are allowed to quiesce.

2. Tape error statistics for all tape drives assigned to the program being terminated on which an error has occurred are logged out on SYSLOG and the statistics reset (system generation option).

3. DASD extents in use by this program for DASD file protection are dequeued (system-generation option).

4. All I/O assignments made for the program are canceled so that these devices may be available to subsequent programs.

5. The operator is notified that the program is completed. The storage used by the program remains allocated for the foreground area.

6. The program is detached from the system's task selection mechanism.

After a foreground program is completed, the operator may initiate another program for the area by pressing the SYSLOG request key and continuing with the initiation procedure previously described.

Figure 51 illustrates the relationship among programs in a multiprogramming environment. In addition, this figure shows the format of the save area used with each program.

NONRESIDENT ATTENTION ROUTINES (FIGURE 49)
Attention commands are submitted when the operator presses the request key on the 1052 keyboard. The system's Attention transient routine ($$BATTNA) is loaded and issues the message READY FOR COMMUNICATIONS. It then reads input statement information and selects the appropriate statement processor. Commands accepted by the nonresident attention routines are:

- PAUSE: Indicates job control pauses for operator communication at the end of the current batch job step.
- CANCEL: Indicates one of the programs in the system is to be canceled. See Figure 52 for cancel code information.
- MAP: Provides a map of main-storage utilization. See Figure 47.
- ALLOC: Permits the operator to allocate storage among foreground and background programs.
- MSG: Causes control to be given to a foreground program operator communications routine previously activated by a STXIT command.
- TIMER: Causes interval timer support to be given to the program specified.
- START: Indicates the foreground initiation function has begun.

| | | | |
|---|---|---|---|
| SP | | upper limit | |
| BG | size | upper limit | NAME |
| F2 | size | upper limit | NAME |
| F1 T | size | upper limit | NAME |
| field 1 | field 2 | field 3 | field 4 |

Field 1 - area identification

SP - supervisor

BG - background area

F2 - foreground area 2

F1 - foreground area 1

T - indicates which program has interval timer support.

Field 2 - length of area.

The number of bytes allocated to the corresponding area of storage. Where 2K equals 2048 bytes of storage. For the background area this represents the number of full 2K blocks. For example, if the area were 11.2K, the MAP would indicate 10K.

Field 3 - area upper storage limit.

The highest storage address allocated to the corresponding area in hexadecimal.

Field 4 - user name

BG - background job name

F2 - foreground 2 program name

F1 - foreground 1 program name

Absence of a name indicates there is no active program in the area.

Figure 47. MAP Output

```
      LISTIO ALL
   F1             **** SYSTEM ****
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYSRDR            0     0C
   F1  SYSIPT            0     0C
   F1  SYSPCH            0     0D
   F1  SYSLST            0     0E
   F1  SYSLOG            0     1F
   F1  SYSLNK            * UNA *
   F1  SYSRES            1     90
   F1
   F1             *** PROGRAM ****
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYS000            1     91
   F1  SYS001            1     91
   F1  SYS002            1     91
   F1  SYS003            1     91
   F1  SYS004            1     92
   F1  SYS005            1     92
   F1  SYS006            1     92
   F1  SYS007            1     92
   F1  SYS008            * UNA *
   F1  SYS009            * UNA *
   F1  SYS010            * UNA *
   F1  SYS011            * UNA *
   F1
   F1             * FOREGROUND 1 *
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYS000            * UNA *
   F1  SYS001            * UNA *
   F1  SYS002            * UNA *
   F1  SYS003            * UNA *
   F1  SYS004            * UNA *
   F1
   F1             * FOREGROUND 2 *
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYS000            * UNA *
   F1  SYS001            * UNA *
   F1  SYS002            * UNA *
   F1  SYS003            * UNA *
   F1  SYS004            * UNA *
   F1
   F1 LISTIO BG
   F1             **** SYSTEM ****
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYSRDR            0     0C
   F1  SYSIPT            0     0C
   F1  SYSPCH            0     0D
   F1  SYSLST            0     0E
   F1  SYSLOG            0     1F
   F1  SYSLNK            * UNA *
   F1  SYSRES            1     90
   F1
   F1             *** PROGRAM ****
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYS000            1     91
   F1  SYS001            1     91
   F1  SYS002            1     91
   F1  SYS003            1     91
   F1  SYS004            1     92
   F1  SYS005            1     92
   F1  SYS006            1     92
   F1  SYS007            1     92
   F1  SYS008            * UNA *
   F1  SYS009            * UNA *
   F1  SYS010            * UNA *
   F1  SYS011            * UNA *
   F1
   F1 LISTIO F1
   F1             * FOREGROUND 1 *
   F1
   F1 I/O UNIT   CMNT   CHNL  UNIT   MODE
   F1
   F1  SYS000            * UNA *
   F1  SYS001            * UNA *
   F1  SYS002            * UNA *
   F1  SYS003            * UNA *
   F1  SYS004            * UNA *
   F1
   F1 LISTIO UA
   F1             ** UNASSIGNED **
   F1
   F1                CHNL  UNIT
   F1
   F1                  1    90
   F1
   F1
```

Figure 48. List I/O Examples for Nonresident Attention Request

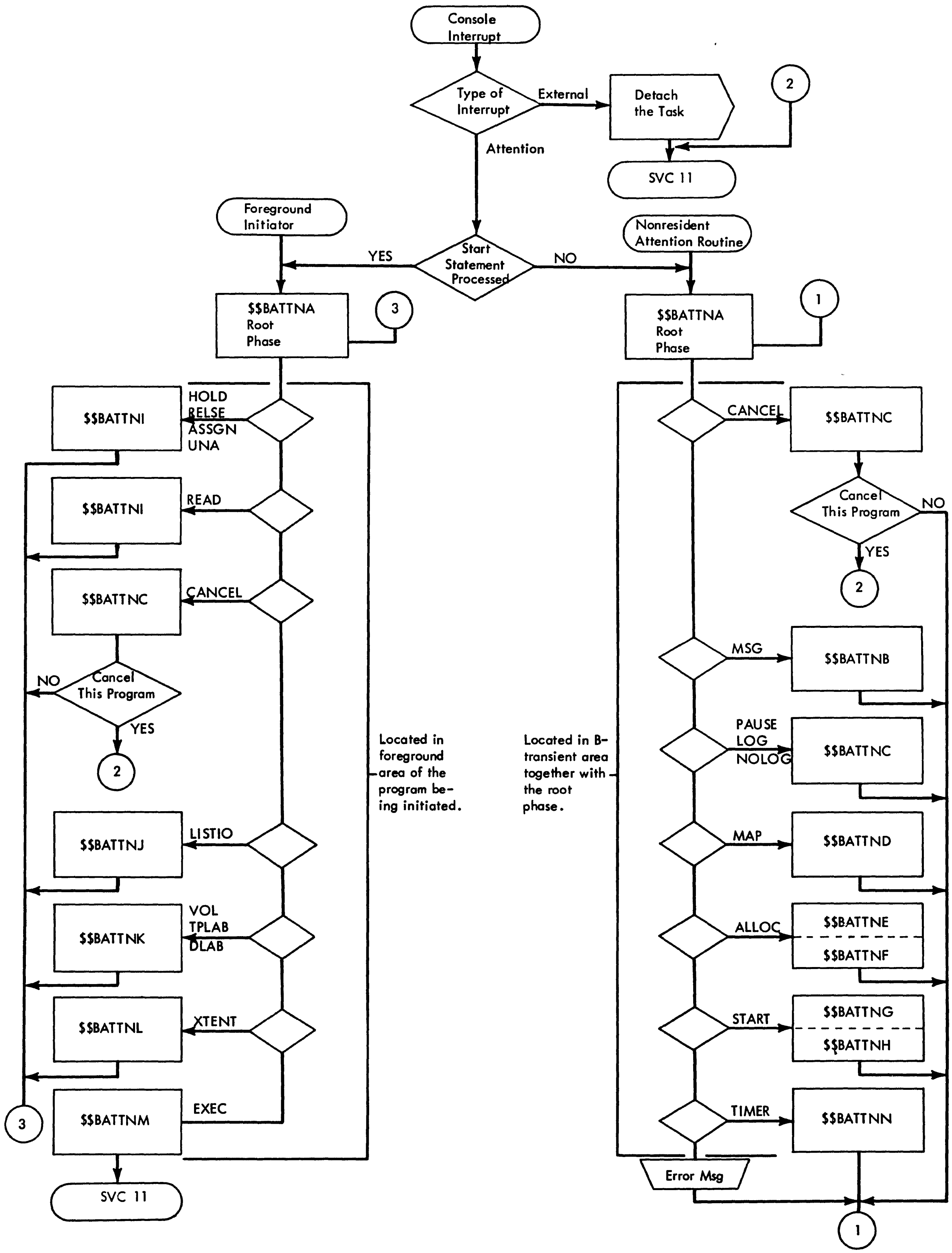



Figure 49. Initiator Phase Map

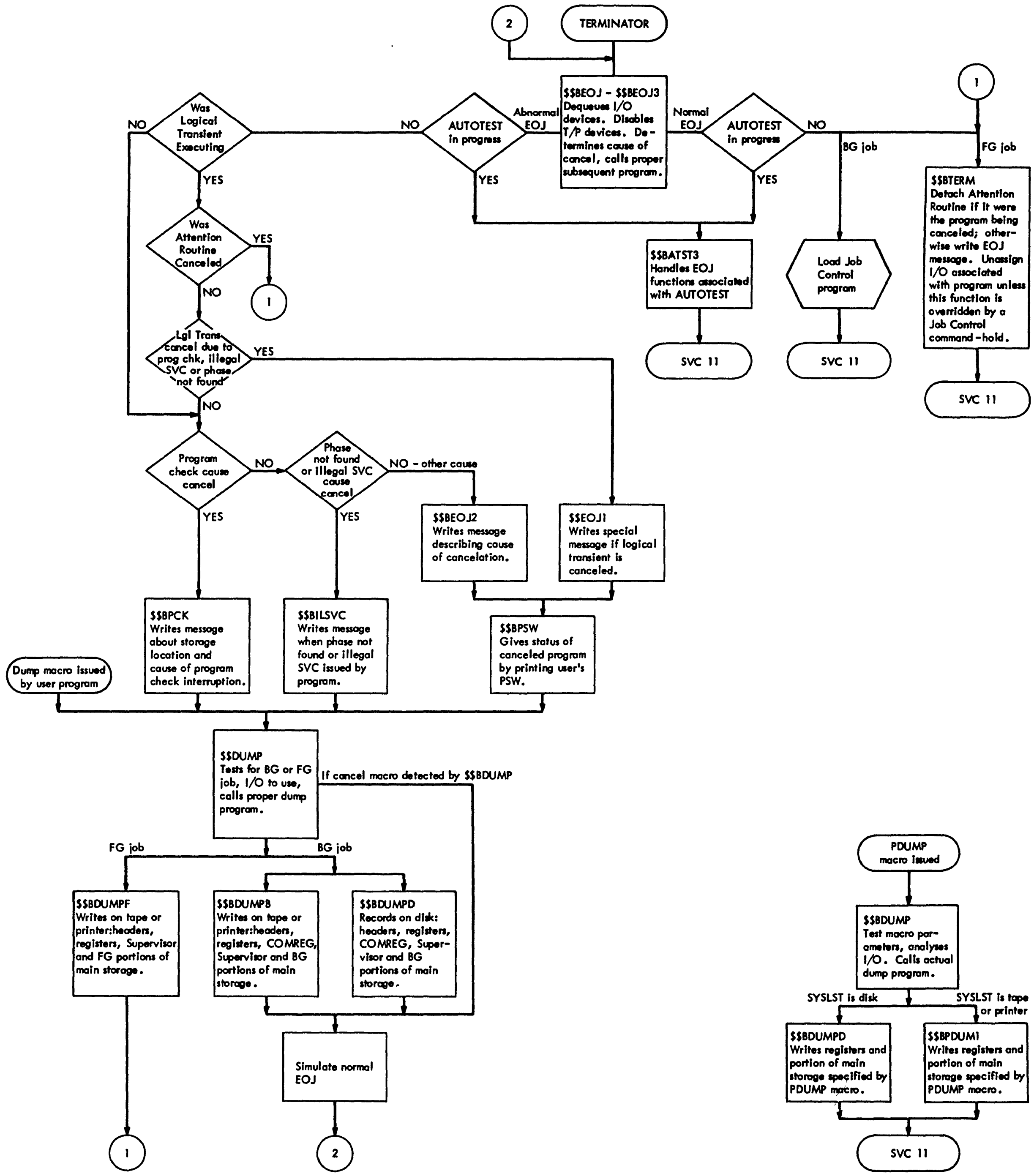



Figure 50. Terminator Phase Map

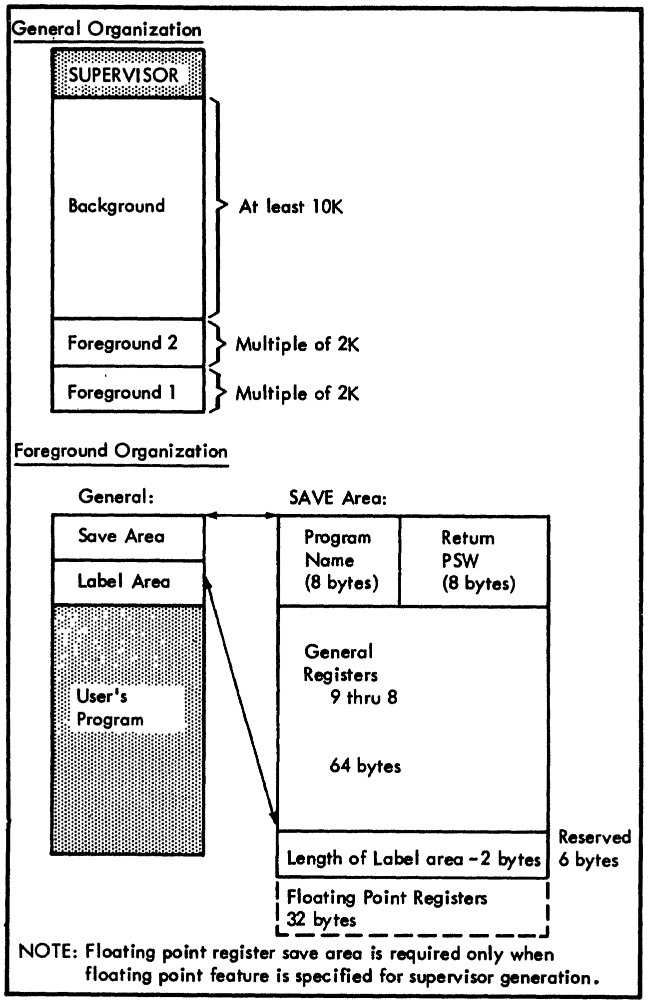


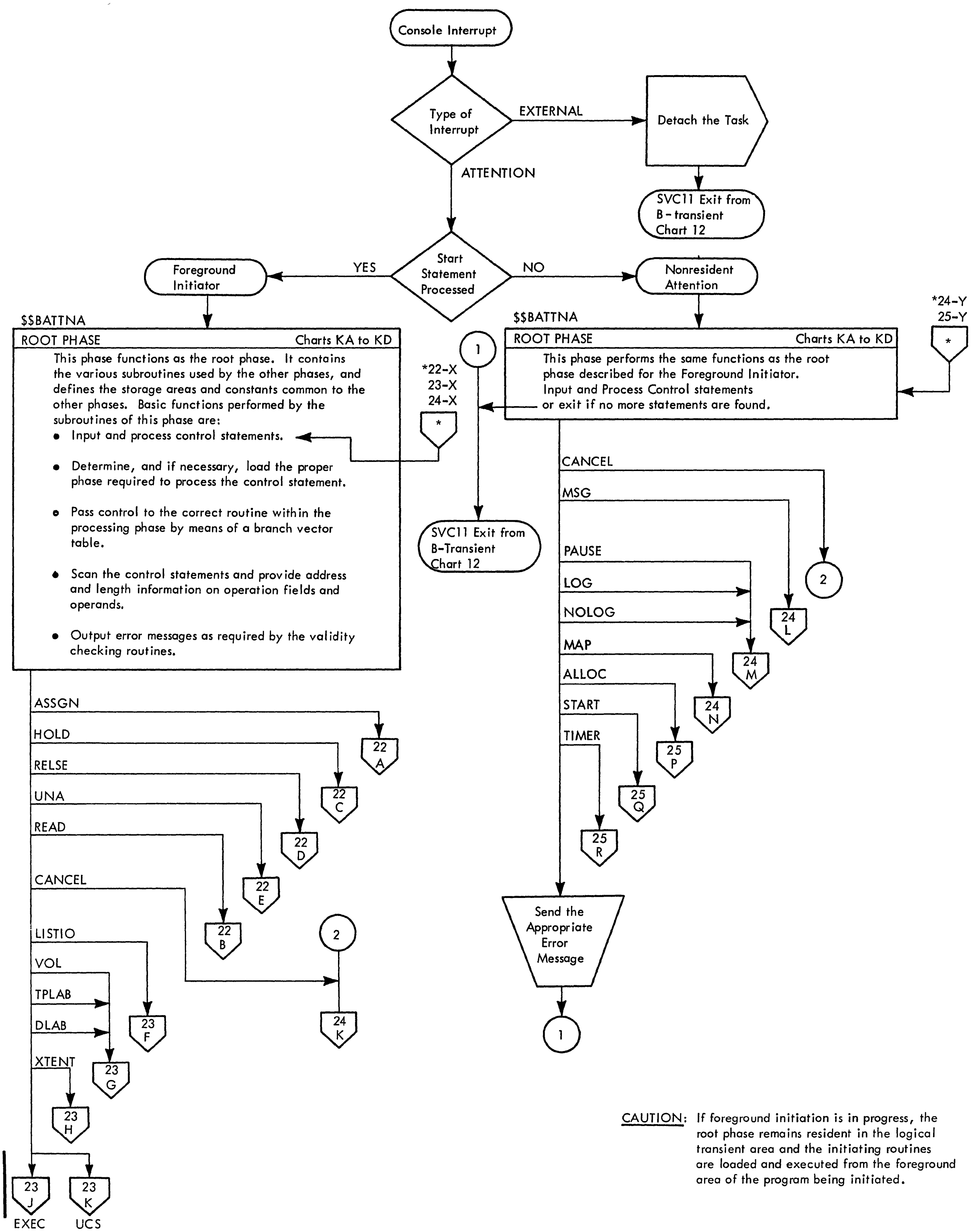

Figure 51. Multiprogram Main-Storage Organization

| Cancel-code in HEX | MSG-Code | Descriptive Part of Message |
|---|---|---|
| 10 | | Normal EOJ |
| 19 | 0P74 | I/O Operator Option |
| 1A | 0P73 | I/O Error |
| 20 | 0S03 or 0S11 | Program Check |
| 21 | 0S04 or 0S09 | Illegal SVC |
| 22 | 0S05 or 0S06 | Phase Not Found |
| 23 | 0S02 | Program Request |
| 24 | 0S01 | Operator Intervention |
| 25 | 0P77 | Invalid CCB-Address |
| 26 | 0P71 | Device Not Assigned |
| 27 | 0P70 | Undefined Logical Unit |
| 30 | 0P72 | Reading Past /& Statement |
| 31 | 0P75 | I/O Error Queue Overflow |
| 32 | 0P76 | Invalid DASD Address (Disk Only) Irrecoverable I/O Error (Tape Only) |
| 33 | 0P79 | No Long Seek (Disk Only) |
| FF | 0P78 | Unrecognized CANCEL Code |

All cancel-codes except in connection with DUMP-macro (code = X'00' - not a true cancel-condition) initially have a value X'40' higher than indicated above, but the X'40' bit is stripped by the SUPVR before fetching the Terminator.

In addition to recognizing the cancel-codes above, the Terminator also recognizes the same codes with the X'80' bit on. The X'80' bit indicates that the cancellation occurred in a Logical Transient routine and it is tested for by $$BEOJ and subsequently reset.

Figure 52. Cancel Code Messages




CAUTION: If foreground initiation is in progress, the root phase remains resident in the logical transient area and the initiating routines are loaded and executed from the foreground area of the program being initiated.

Chart 21. Logical Transient Root Phase

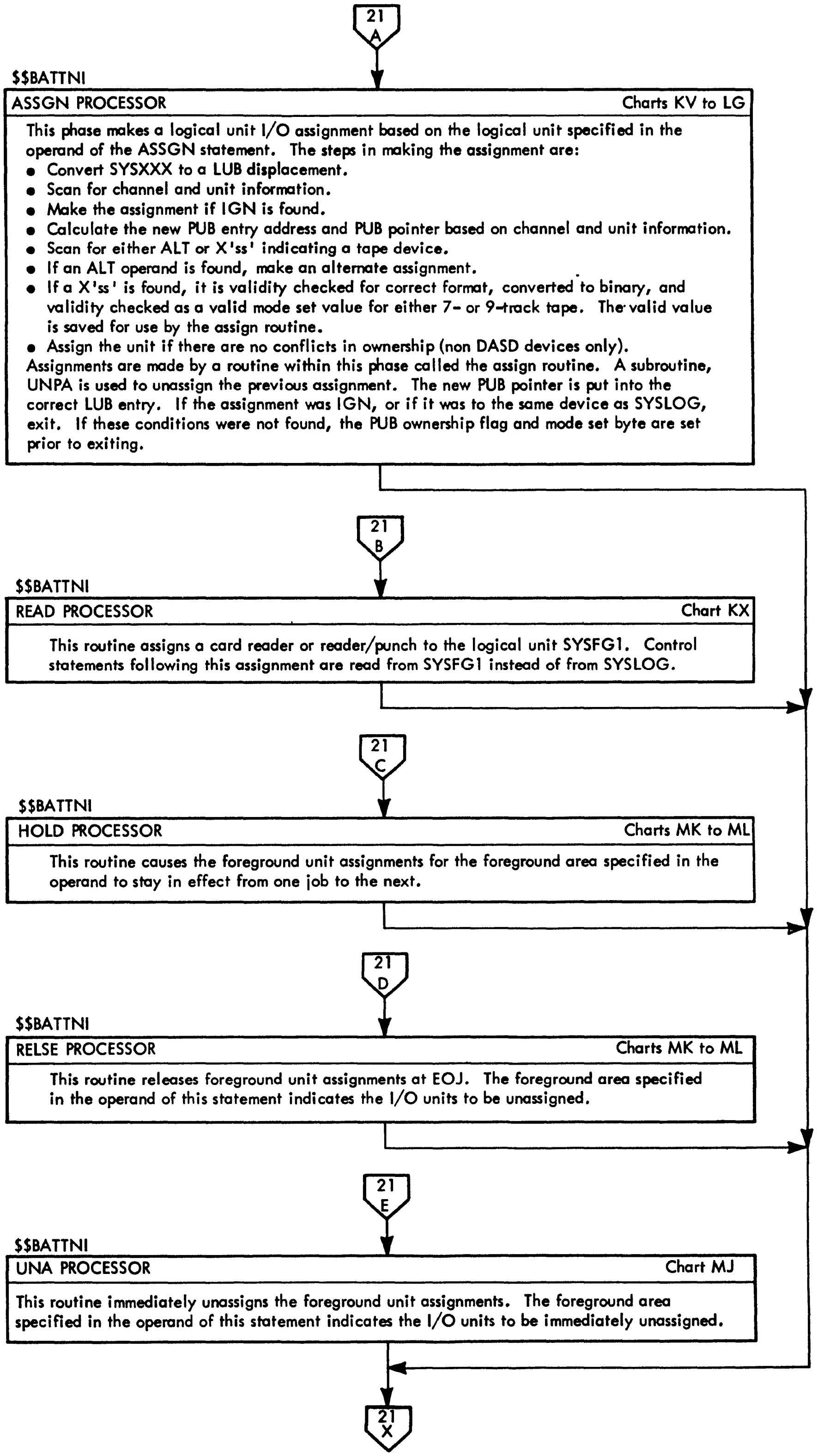



Chart 22. Logical Transient Foreground Initiator (Part 1 of 2)

21 F

$$BATTNJ

**LISTIO PROCESSOR** — Charts LH to LN

This phase provides a listing on SYSLOG of the information pertaining to the correct status of logical unit I/O assignments. The root phase scan routine is used to get the operand (BG, F1, F2, ALL, or UA).

If BG is specified, the system class units, and the BG programmer class units are listed under a BG heading. If F1 or F2 is specified, the appropriate programmmer class units are listed under the desired foreground heading. If the operand ALL is specified all units are listed under their proper heading.

If the operand specified is UA, an UNASSIGNED heading is established and each PUB is checked for the unassigned, not down conditions. If these conditions are found, the unit is listed as unassigned. If no units are found unassigned the heading, NONE, is listed.

21 G

$$BATTNK

**VOL/TPLAB/DLAB PROCESSOR** — Charts LP to LW

This phase processes the VOL, TPLAB, and DLAB statements for standard volume and file labels. The sequence of the statements is critical and therefore, this phase processes in this order: VOL - TPLAB or VOL, DLAB, EXTENT (any number of extents).

When a VOL statement is encountered:

- The temporary label area is output on the SYSRES label tracks.
- The operand is found by the scan routine.
- The operand is validity checked and stored in the temporary label area (Foreground origin + 1728).

When a TPLAB statement is encountered:

- The operand field is tested to determine if concantenation is required.
- A correct sequence check is made and a switch is set to indicate the current statement being processed.
- The operand is validity checked and moved to the temporary label area.

When a DLAB statement is encountered the same functions performed for TPLAB statements are also performed for the DLAB statement.

21 H

$$BATTNL

**XTENT PROCESSOR** — Charts LX to MB

This phase processes XTENT statements in the same manner as it processes file label statements. It also tests for overlapped limits.

21 X

21 J

$$BATTNM

**EXEC PROCESSOR** — Charts MC to MF

This phase determines the type of program to be executed by using the scan routine to get the operand. It tests a program switch to determine if any additional label information is to be output to the SYSRES label track. If necessary, it performs the I/O to write and verify the label data.

This phase relocated parts of itself in the B-transient area of main storage. It clears main storage within the foreground program area except for the foreground save area. The foreground save area is initialized with the name of the program to be executed, the PSW values for that program and storage space for all registers used.

SVC11 End of Initiation Chart 12

21 K

$$BATTNM

**USC STATEMENT PROCESSOR** — Chart MG

The 240-character Universal Character Set (UCS) specified by the operand "PHASENAME" is loaded in the buffer of the IBM 2821 Control Unit of the logical unit specified by the operand "SYSXXX". The phase is loaded from the Core Image Library.

1. The operand "SYSXXX" specifies a 1403 with the UCS feature.
2. The operand "PHASENAME" specifies the core image library phase to be loaded.
3. The operand "FOLD" (optional) specifies that the buffer is to be loaded with the "folding" operation code (hex F3) in the 3rd CCW.
4. The operand "BLOCK" (optional) specifies that data checks are to be inhibited. The inhibit data check operation code, X'73' is set in the 2nd CCW.
5. The operand "NULMSG" (optional) specifies that the 80-character verification message is not to be printed. The chain bit is set off in the 3rd CCW.

21 X

Chart 23. Logical Transient Foreground Initiator (Part 2 of 2)

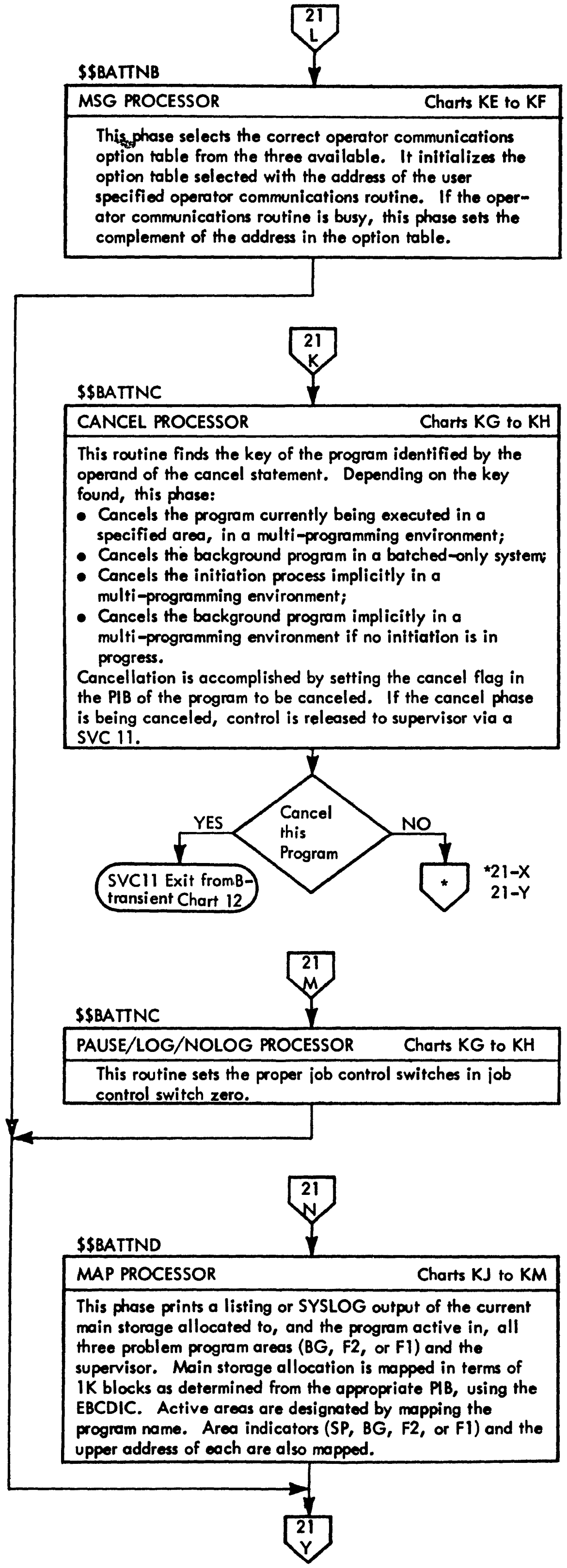



Chart 24. Logical Transient Nonresident Attention Routines (Part 1 of 2)

21
P

$$BATTNE and $$BATTNF

**ALLOC PROCESSOR** — Charts KN to KR

This routine reallocates main storage according to the number of 2K blocks specified for one, or both of the foreground areas by the operand of the ALLOC statement. To reallocate, this routine:

1. Builds a table containing the current number of 2K blocks, the lower limit address, the address of the save area, and the upper limit address. A separate table is built for the background area and each foreground area. The source information for the table entries is the program PIB;
2. Updates the table built in step 1 with the information secured from the operand of the ALLOC statement. The operand value is easily converted to byte count which is used to calculate upper limit information. Upper and lower limits necessary to ensure contiguousness and the designated amount of main storage are calculated for each table entry and checked to overlap for active areas;
3. Resets the PIB entries with storage key and origin information and storage keys of all program areas using the table entries as source information.

21
B

21
Q

$$BATTNG and $$BATTNH

**START PROCESSOR** — Charts KS to KW

This phase initiates a foreground program or the background area previously put in a dormant state by the job control command STOP. The operand of the START statement is obtained by the root phase scan subroutine. If the background (BG) operand is found, the BG PIB is activated X'83'. If a foreground operand is found, the proper key is calculated, the attention PIB is detached from task selection, and the initiator is attached as the foreground program. Subsequent initiator phases are executed from the foreground program area of the program being initiated. However, even though physically located in the foreground area, the subsequent initiator phases are logically considered as part of the transient attention routine.

21
B

21
R

$$BATTNN

**TIMER PROCESSOR** — Chart MH

This phase changes the ownership of the interval timer routine based on the operand specified in the TIMER statement.

21
Y

Chart 25. Logical Transient Nonresident Attention Routines (Part 2 of 2)

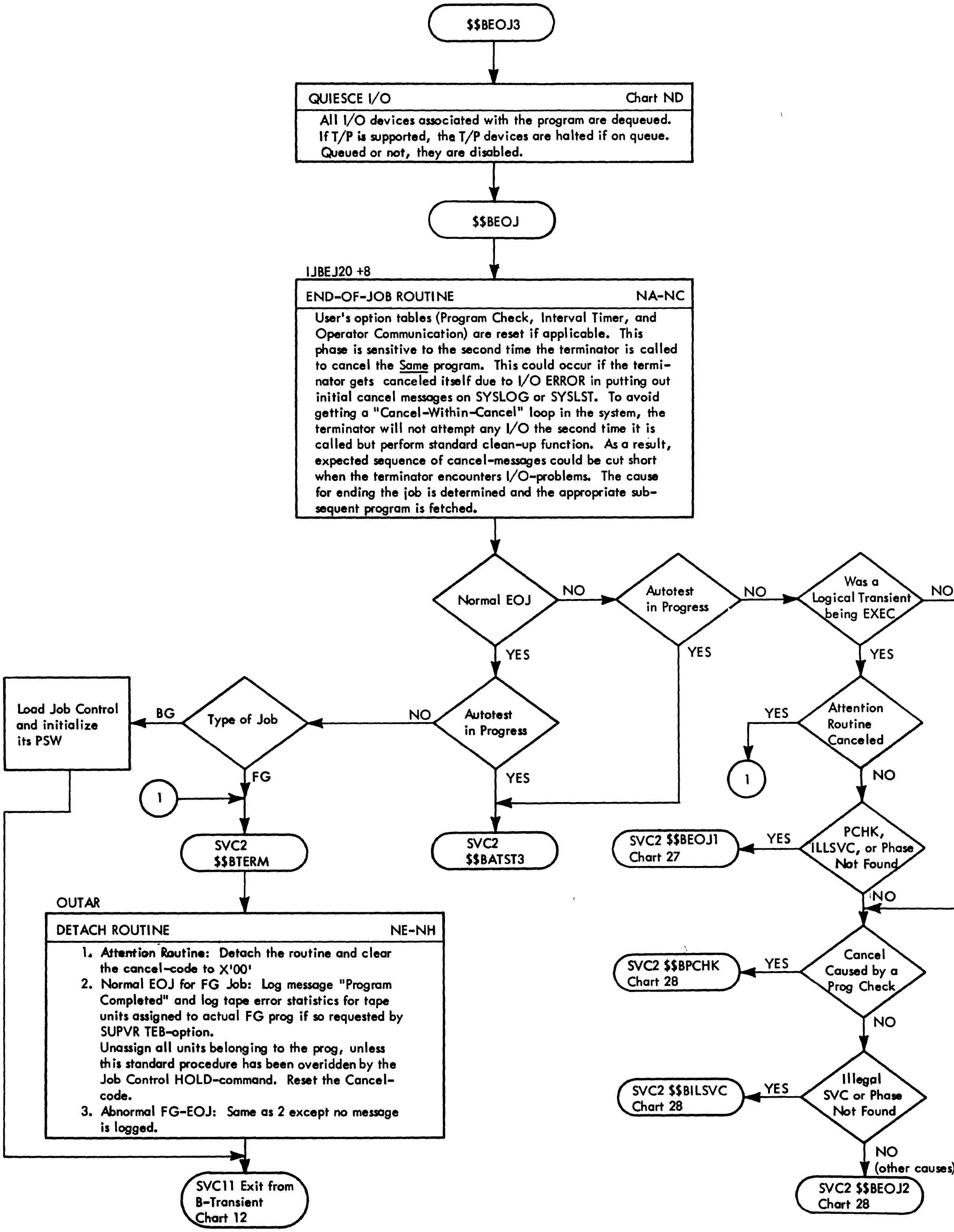



Chart 26. Logical Transient--Terminator (Part 1 of 5)

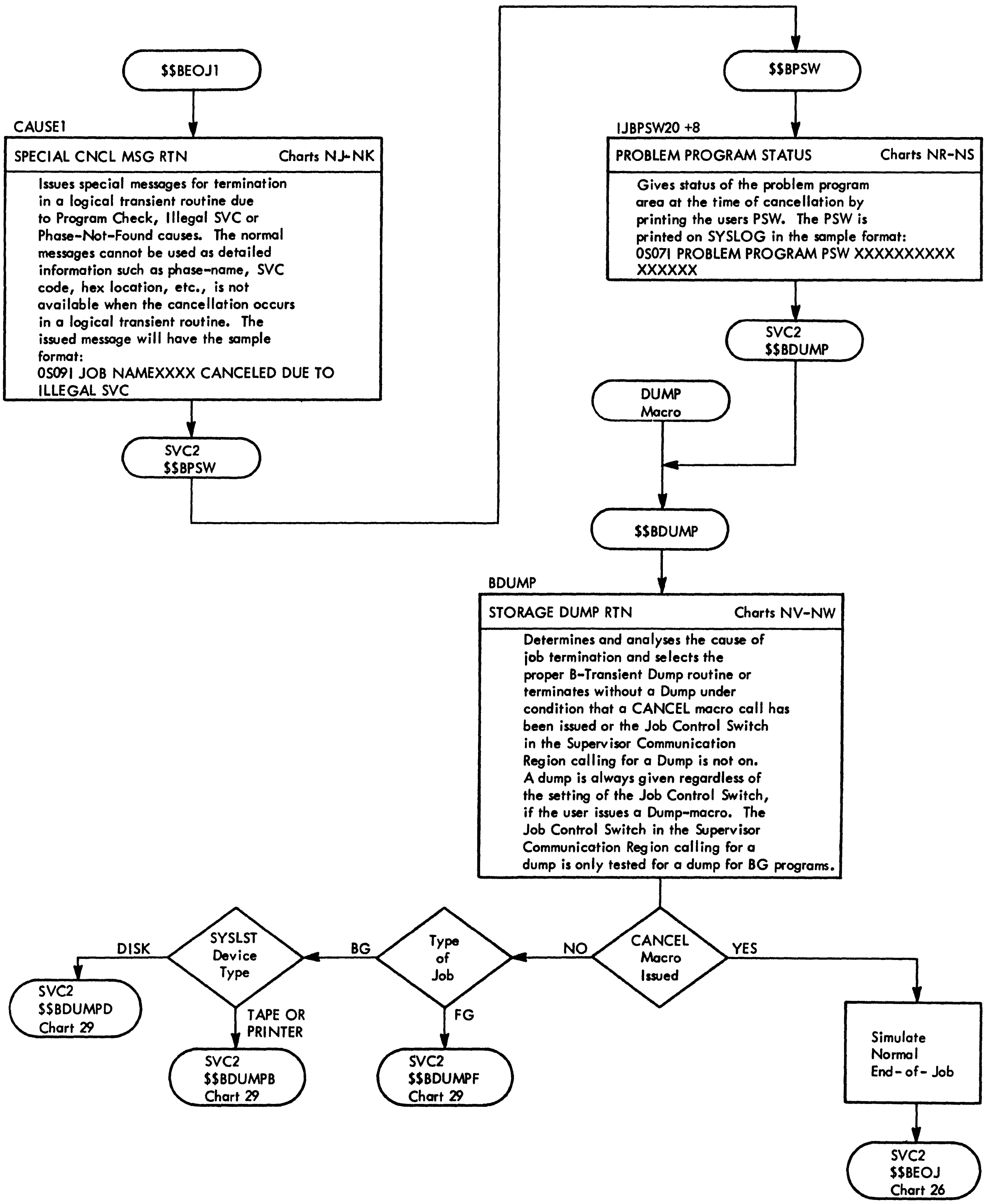



Chart 27. Logical Transient--Terminator (Part 2 of 5)

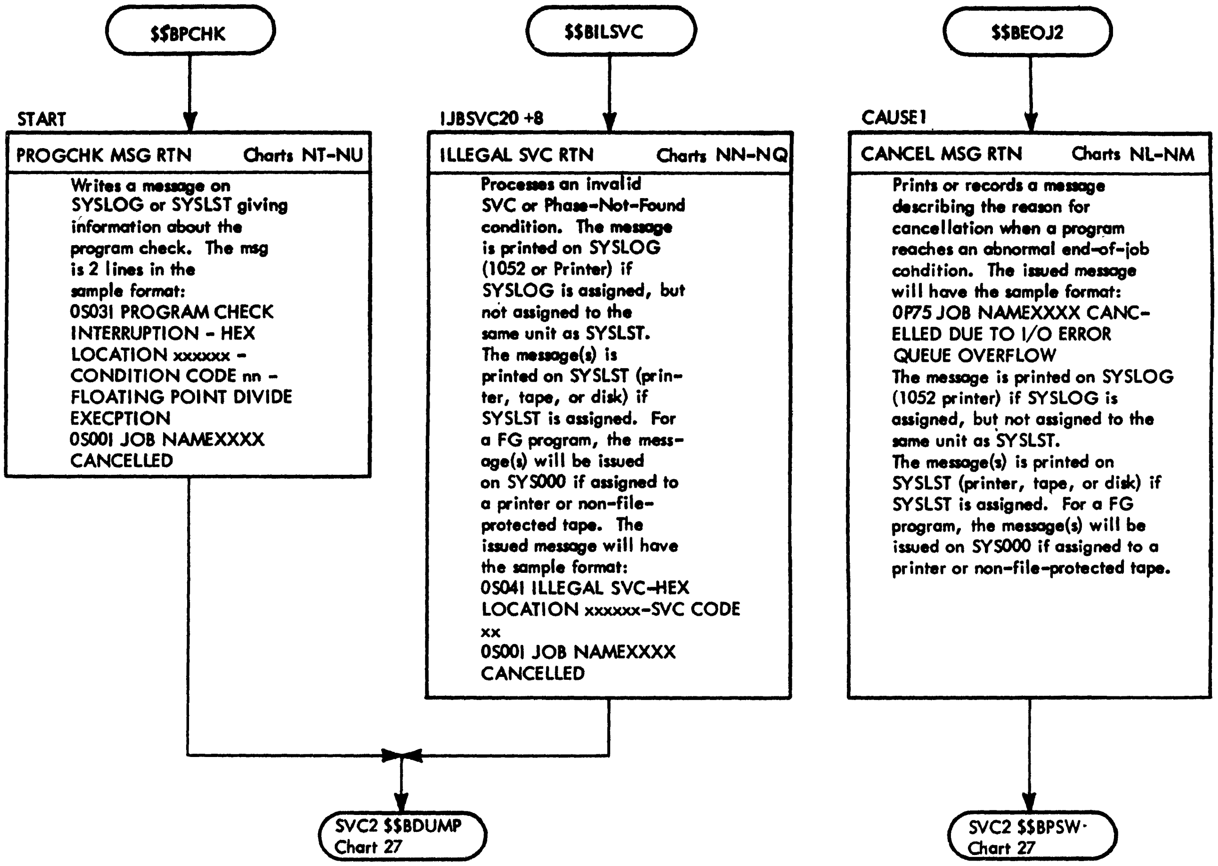



Chart 28. Logical Transient--Terminator (Part 3 of 5)

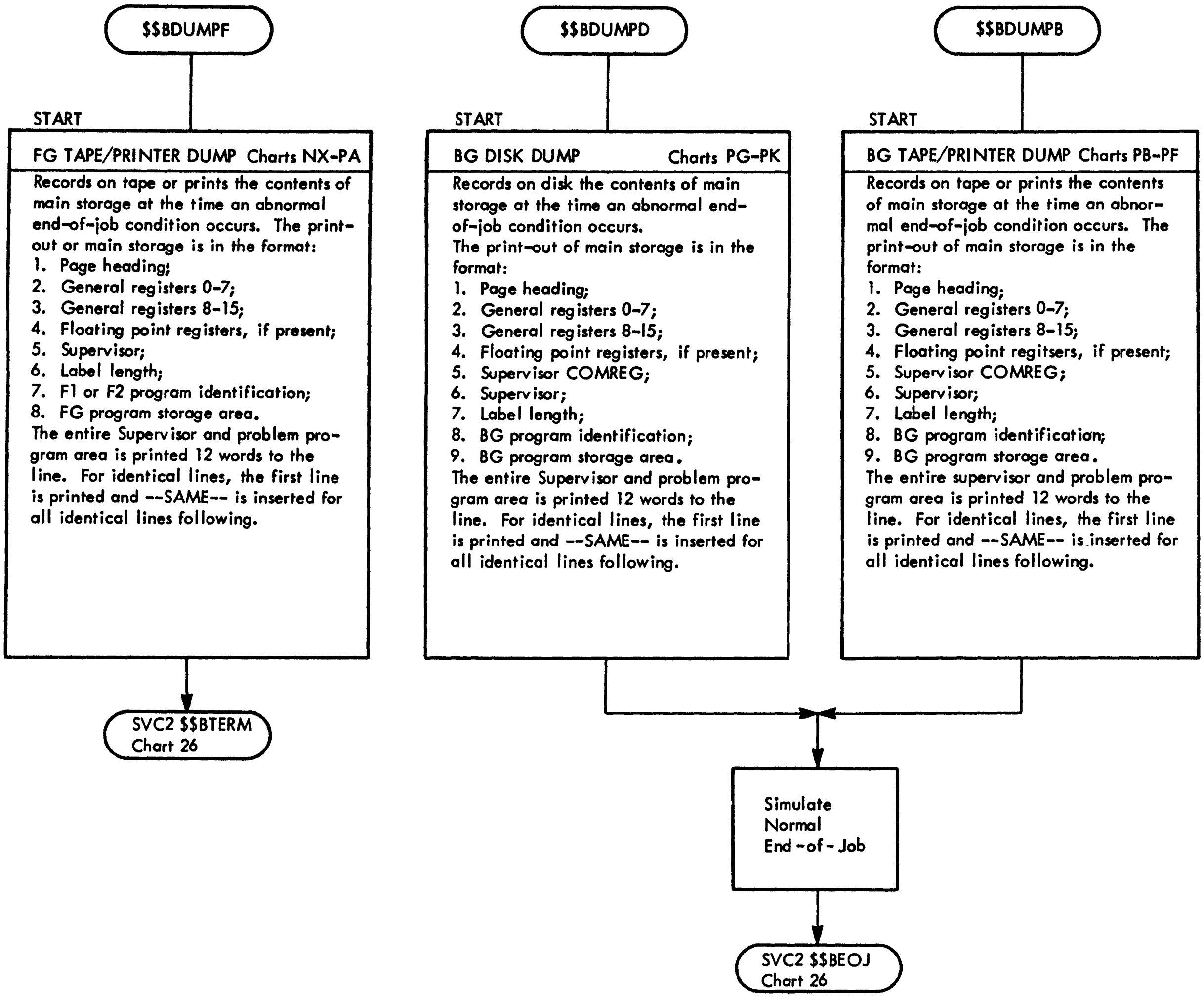



Chart 29. Logical Transient--Terminator (Part 4 of 5)

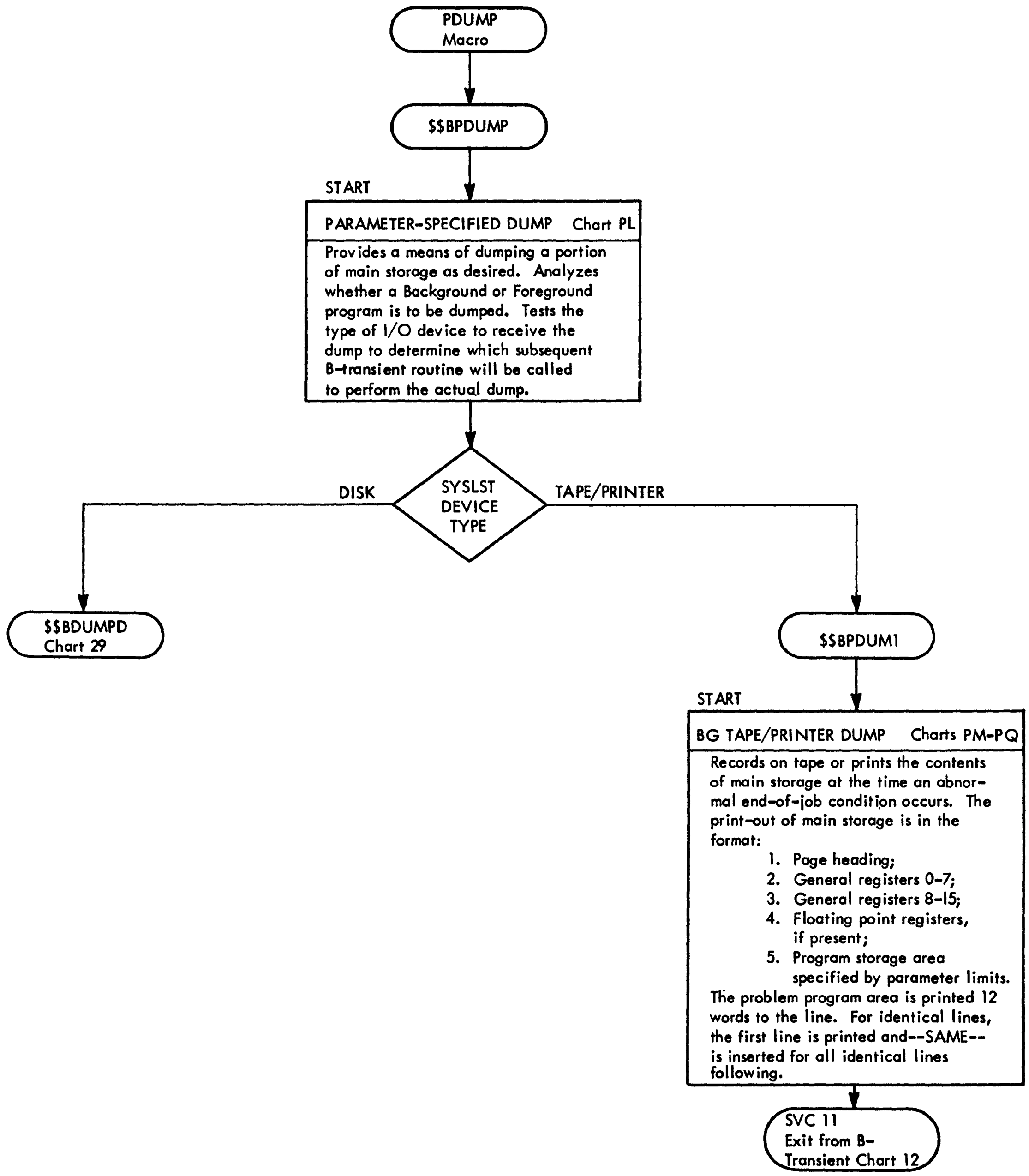



Chart 30. Logical Transient--Terminator (Part 5 of 5)

The linkage editor prepares programs for execution on DOS, and accepts as input the relocatable object modules produced by the language translators. It processes these modules into program phases, which may be immediately executed or cataloged into the core image library.

The linkage editor control cards direct the program to read input module(s) and to form phases from the control sections within the modules. Figure 53 shows how phases can be formed. The linkage editor relocates the origin of each control section in the phase, assigns each phase an area of main storage and a transfer address, and modifies the contents of the address constants in the phase.

| Sample of a 2-module input resulting in a 3-phase output | |
|---|---|
| **Language Translator Output** | **Linkage Editor Output** |
| **Module A**<br>**ESDs**<br>**TXT - CSECTA**<br>**TXT - CSECTB**<br>**TXT - CSECTC**<br>**RLDs** | **Phase 1**<br>**CSECTA**<br>**CSECTB** |
| **Module B**<br>**ESDs**<br>**TXT - CSECTD**<br>**TXT - CSECTE**<br>**TXT - CSECTF**<br>**TXT - CSECTG**<br>**RLDs** | **Phase 2**<br>**CSECTC**<br>**CSECTD**<br>**CSECTE**<br><br>**Phase 3**<br>**CSECTF**<br>**CSECTG** |

Figure 53. Module Phase Relationship

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

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

## LANGUAGE TRANSLATOR MODULES

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

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

Six card types are produced by the language translators or the programmer to form a module. They appear in the following order:

| Card Type | Definition |
|---|---|
| ESD | External symbol dictionary |
| SYM | Ignored by linkage editor |
| TXT | Text |
| REP | Replacement to text made by the programmer |
| RLD | Relocation list dictionary |
| END | End of module |

The external symbol dictionary contains control section definitions and intermodule references. When the linkage editor has the ESDs from all modules, it can relocate the sections and resolve the references. Five types of entries are defined in the control dictionary.

| ESD Type | Definition |
|---|---|
| SD | Section Definition: provides control section name, assembled origin and length. |
| PC | Private Code: provides assembled origin and length for an unnamed control section. |
| LD | Label Definition: specifies the assembled address and the associated SD of a label that may be referred to by another module. |
| ER | External Reference: specifies |

the location of a reference made to another module.

CM — Common: indicates the amount of main storage to be reserved for common use by different phases.

The relocation list dictionary identifies portions of text that must be modified on relocation (address constants).

When the linkage editor reads a module, it stores ESDs in its control dictionary, writes TXT and REP items in core image blocks in the library, and writes RLD items on an RLD file. Each item, identified by the language translators with an ESID number, is identified by the linkage editor with a control dictionary number to avoid duplication of identification between modules.

## LINKAGE EDITOR PROGRAM FLOW

The linkage editor is physically divided into eight phases. The phase names assigned and functions are:

| Phase Name | Function |
|---|---|
| Phase 1 ($LNKEDT) | Initialize/Overhead, Chart 31 |
| Phase 2 ($LNKEDT0) | 12-2-9 Processor (ESD only), Chart 32 |
| Phase 3 ($LNKEDT2) | 12-2-9 Processor (other than ESD), Chart 33 |
| Phase 4 ($LNKEDT4) | Control Card Processor Chart 34 |
| Phase 5 ($LNKEDT6) | Control Card Processor Chart 35 |
| Phase 6 ($LNKEDT8) | MAP Processor, Chart 36 |
| Phase 7 ($LNKEDTA) | Pass 2 Processor, Chart 37 |
| Phase 8 ($LNKEDTC) | Catalog Processor, Chart 38 |

The first phase ($LNKEDT) is fetched by the job control program. This phase finds the existing machine configuration. The determining factor as to how the remaining linkage editor phases are fetched is main storage availability. If less than 14K available main storage for BG programs is found, a 2-part initialization is executed. The second part, executed first, checks I/O unit assignments, opens SYSLNK and SYS001, and saves the DTF table. The first part, executed last, sets up the control dictionary and linkage table. It processes any ACTION cards, and loads main storage with the card processing phase required by the type of card image found in the input stream. The card processor loaded (12-2-9 or control card) overlays the initialization part of the $LNKEDT phase, leaving the overhead part for use by other linkage editor phases. When an ENTRY card is finished processing, indicating the end of a linkage editor run, the MAP and Pass 2 Processors are fetched and executed sequentially.

If more than 14K available main storage is found, $LNKEDT is fetched by job control as in the smaller core allocation. The second part of initialization is executed first to perform the initialization steps described for the smaller core allocation. The first part of initialization is then executed to load all the card processing phases, to initialize the control dictionary and the linkage table, and to process ACTION cards. Individual phase loading is finished at this point, and the program continues executing instructions in the card processing phase as determined by the type of input. When an ENTRY card is finished processing (in $LNKEDT6), the MAP and Pass 2 Processors are fetched and executed sequentially.

The last phase ($LNKEDTC) is fetched by the Pass 2 Processor ($LNKEDTA), if the CATAL option has been selected. If this option is not chosen, the $LNKEDTA phase fetches job control.

### CATALOG CORE IMAGE LIBRARY PHASE ($LNKEDTC), CHART 38

$LNKEDTC is called by $LNKEDTA when the option catalog bit is on in JBCSW1 in displacement 57 of the supervisor communication region. $LNKEDTC catalogs programs to the core image library by adding an entry to the core image directory for each phase of the program. The phase entries are built by a previous phase of the linkage editor in the system work area. (Refer to Figure 3 for the location of the system work area.) If a phase being cataloged has the same name as a phase already in the directory, the phase in the directory is deleted by $LNKEDTC.

Refer to Figure 4 for the format of the system directory and to Figure 6 for the format of the core image directory.

Figures 54 and 55 show Linkage Editor storage maps for core allocation of less than 14K and equal or greater than 14K. Figure 56 shows the I/O flow of the linkage editor program.

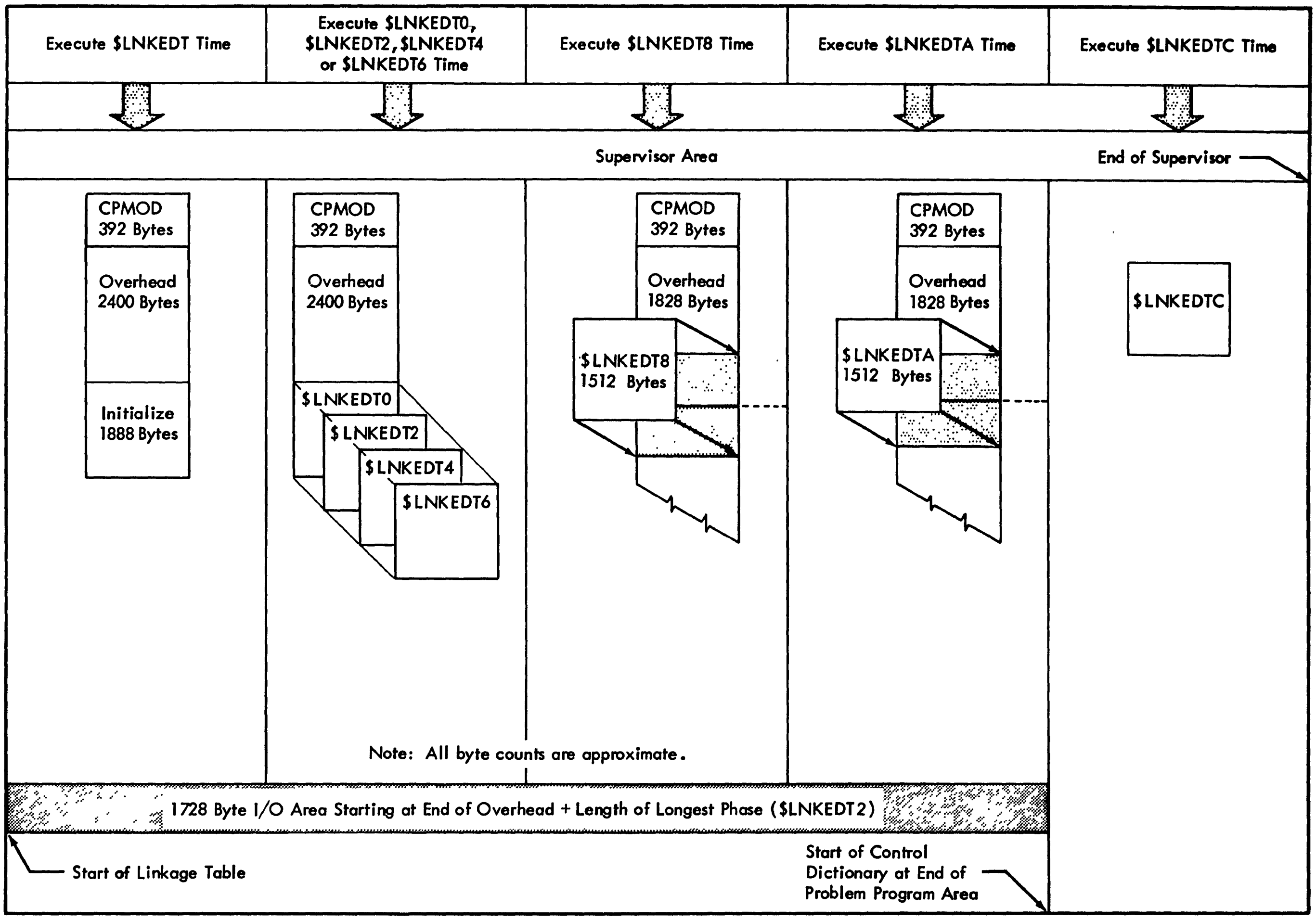



Figure 54. Linkage Editor Storage Map for Less Than 14K Available Main Storage

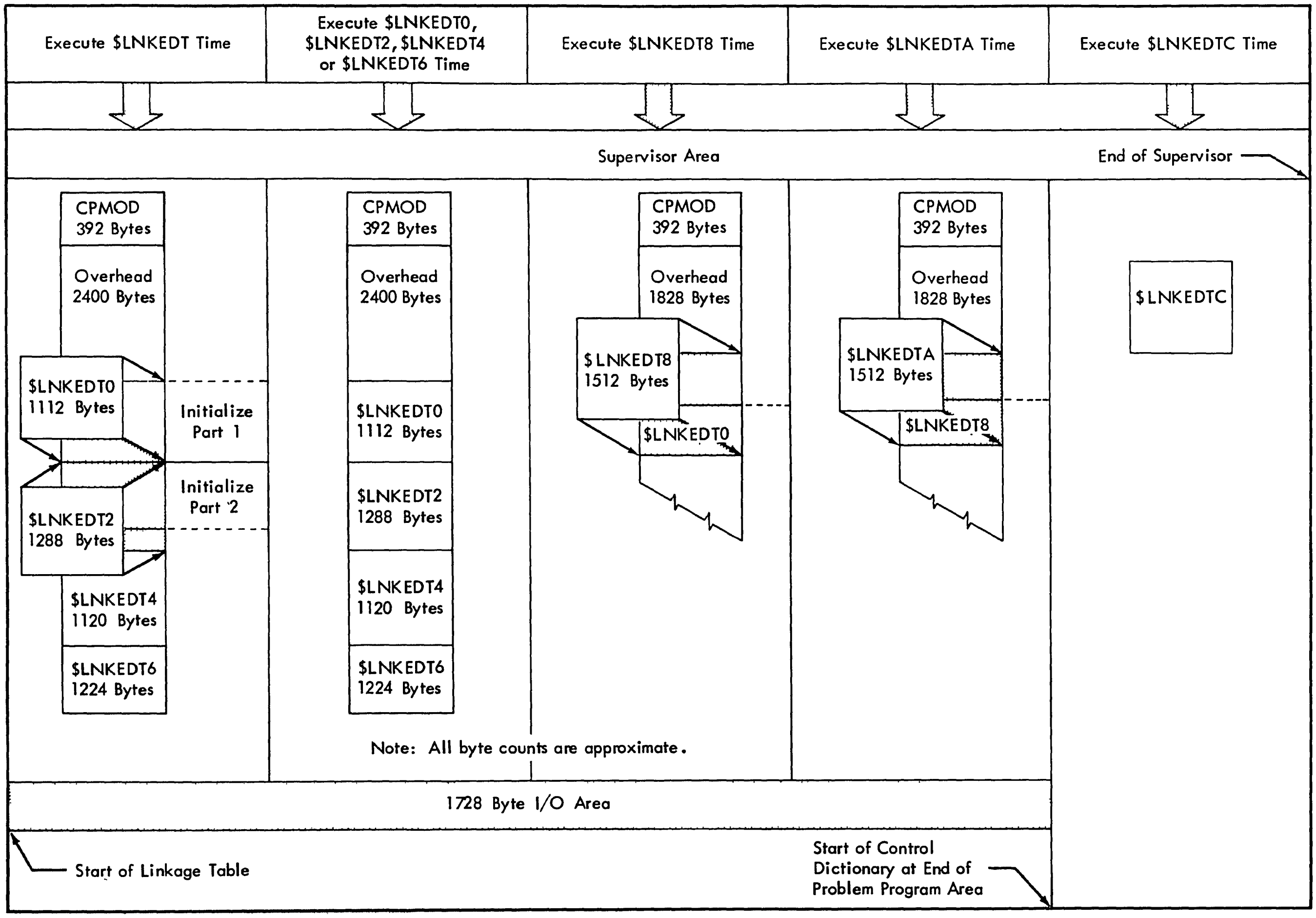



Figure 55. Linkage Editor Storage Map for 14K or More Available Main Storage

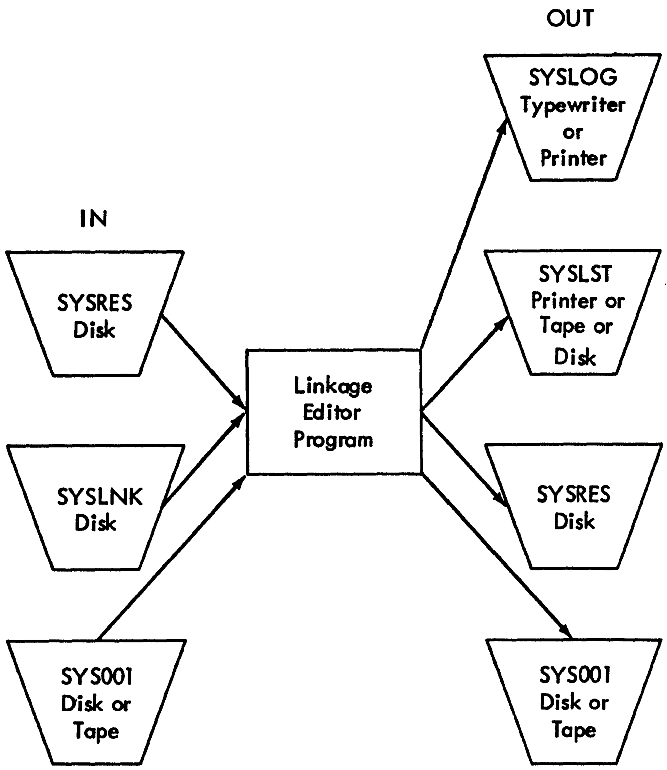



Figure 56. Linkage Editor I/O Flow

KEY CONCEPTS

OVERHEAD PROCESSOR: In addition to the initialization steps, the first phase of the linkage editor contains most of the subroutines used by the various other linkage editor phases. These are often called overhead and therefore, this part of $LNKEDT phase is often called the overhead processor. The subroutines in the first phase are labeled:

| Subroutine | Use |
|---|---|
| RDS000 | Reads blocked input. |
| DERDAD | Sets up core image blocks of text in a work area. |
| LTESID | Finds control dictionary information and the relocation factor using the linkage table. |
| SRCHCD | Searches the control dictionary for a matching label. |
| CHVHEX | Converts hexadecimal input to binary output. |
| PRINT | Performs print and carriage control operations. |
| AD1DSK | Updates disk addresses using predetermined overflow factors to optimize the update operation. |
| READ/WRITE | Reads or writes core image library blocks. |
| XTPHNO | Extracts the phase number from a control dictionary entry. |
| ABTERR | Fetches the $LNKEDTA phase for abnormal termination (abort) error handling. |
| CDSIZE | Checks for control dictionary - linkage table overlap. |
| RDNEXT | Reads the input stream. |
| ALNKPR | Initializes for a scan of the relocatable directory, when the autolink feature is not suppressed. Puts all unresolved ERs, found in the control dictionary, into the correct collating sequence. |
| ERROR | Sets up to print non-abort error messages. |
| OVRLAY | Performs print and carriage control operations when a NOMAP option is found (overlays the first part of the print subroutine). |

CONTROL DICTIONARY: The control dictionary is an internal linkage editor mechanism used to tabulate phase and external symbol dictionary information. It is composed of a variable number of fixed 16-byte entries. Each entry is numbered sequentially, and therefore, the physical structure of the control dictionary roughly outlines the structure of the program. Valid new entries (Phase or ESD) are posted when they are found by the ESD processing routines. Location CDENT1 contains the address of the first entry. Location CTLDAD contains the address of the last entry. The label CDENT1 is located in a high storage location relative to the label CTLDAD, because the control dictionary is built in reverse order. Figure 57 illustrates the control dictionary.

LINKAGE TABLE: The linkage table is an internal linkage editor mechanism used to link the ESID number supplied by the language translator output to the corresponding control dictionary number that belongs to a control dictionary entry.

Control Dictionary ... Pass 1 ... ESD Item

| Label of ESD Item | ESD Type | Assembled Origin | Phase Number and ESD Type | CSECT Relocation Factor --- or ESID of CSECT for Type LD/LR |
|---|---|---|---|---|
| 8 Bytes | 1 Byte | 3 Bytes | 1 Byte | 3 Bytes |

Control Dictionary ... Pass 1 ... Phase Entry

| Phase Name | Type 111 | Phase Origin | Phase Disk Address CHHR |
|---|---|---|---|
| 8 Bytes | 1 Byte | 3 Bytes | 4 Bytes |

Linkage Table

| Control Dictionary Number | ESD Type |
|---|---|
| 2 Bytes | 1 Byte |

ESD Type Field of the Control Dictionary

| ESD Key | Offset High Order 3 Bits of Phase Number | Sign of Relocation Factor | ESD Type |
|---|---|---|---|
| 0 | xxx | 0 = +<br>1 = - | 000 = SD<br>001 = LD<br>010 = ER<br>011 = LR<br>100 = PC<br>101 = CM<br>111 = Phase Entry |

Figure 57. Control Dictionary/Linkage Table

This table is composed of a variable number of fixed 3-byte entries. It is built separately for each object module. When an end card is processed, signaling the end of a module, it is reset to zeros. (Location LTMIN3 contains the address of the first item in the linkage table minus 3 bytes. LNKTAD contains the address of the last item in the linkage table plus 3 bytes. The label LTMIN3 is located in a low storage location relative to the label LNKTAD. Figure 57 illustrates the linkage table.)

USE OF THE LINKAGE TABLE AND CONTROL DICTIONARY: The linkage table is designed to associate text and RLD information with the proper relocation attribute from the control dictionary. The steps taken in processing some text are:

1. Get the ESID number and calculate the linkage table entry.

2. Go to the linkage table.

3. Extract the control dictionary number

field of the linkage table, and calculate the control dictionary entry location.

4. Go to the control dictionary entry.

5. Extract the relocation factor.

6. Add the relocation factor to the assembled origin of the text to be loaded.

7. Substitute the result of the calculation in step 6 (the load origin) for the language translator supplied assembled origin (for the text).

8. Calculate the block of the core image library that this text belongs to (next available block).

9. Get the proper core image block.

10. Put the text into the core image block.

Note: If a TXT card or P pointer points to a negative control dictionary number, that control section is skipped. If the R pointer points to a negative control dictionary number, that control section is needed (CSECT is not in this phase in main storage).

LINKAGE EDITOR FUNDAMENTAL CALCULATIONS: For the examples in this presentation:

- The symbol A/O represents the assembled origin.
- The symbol R/F represents the relocation factor.
- The symbol L/O represents the load origin.
- The symbol P/O represents the phase origin.

Example 1: The language translator provided A/O is added to an R/F that has been determined by the phase origin information. The result, the L/O, is the main storage address that is the physical location of this text, RLD item, or control section.

A/O + R/F = L/O

Example 2: The assembled origin of the CSECT being processed is subtracted from the address that is the next possible phase origin. This results in the relocation factor for that control section.

P/O - A/O = R/F

Example 3: Current control dictionary entry - 16 = next control dictionary entry.

Example 4: Current linkage table entry + 3 = next linkage table entry.

Example 5: Disk address + overflow factors = updated disk address.

The overflow factors are a constant, established by the programmer, that simplify the updating of disk addresses (CHHR). These factors, when added to the disk address, provide the correct cylinder and head after only one calculation.

USE OF THE AUTOLINK FEATURE: This feature tries to locate a module in the relocatable library for any unresolved ERs found in the preceding phase. The signal indicating a phase has finished processing is either a new phase card or an ENTRY card. When the signal is detected, autolink is attempted unless the feature has been suppressed by a NOAUTO phase card or action card option.

EXAMPLE OF AUTOLINK WITH LIOCS: Whenever a DTF macro is expanded during a language translator run, an ER is generated with a label corresponding to a label of a LIOCS module. The label of the ER is used as the search argument in autolink. The autolink processing searches the relocatable directory for the corresponding label. The directory entry contains the disk address of the module in the relocatable library. The module is the macro expansion and is then treated as an include statement.

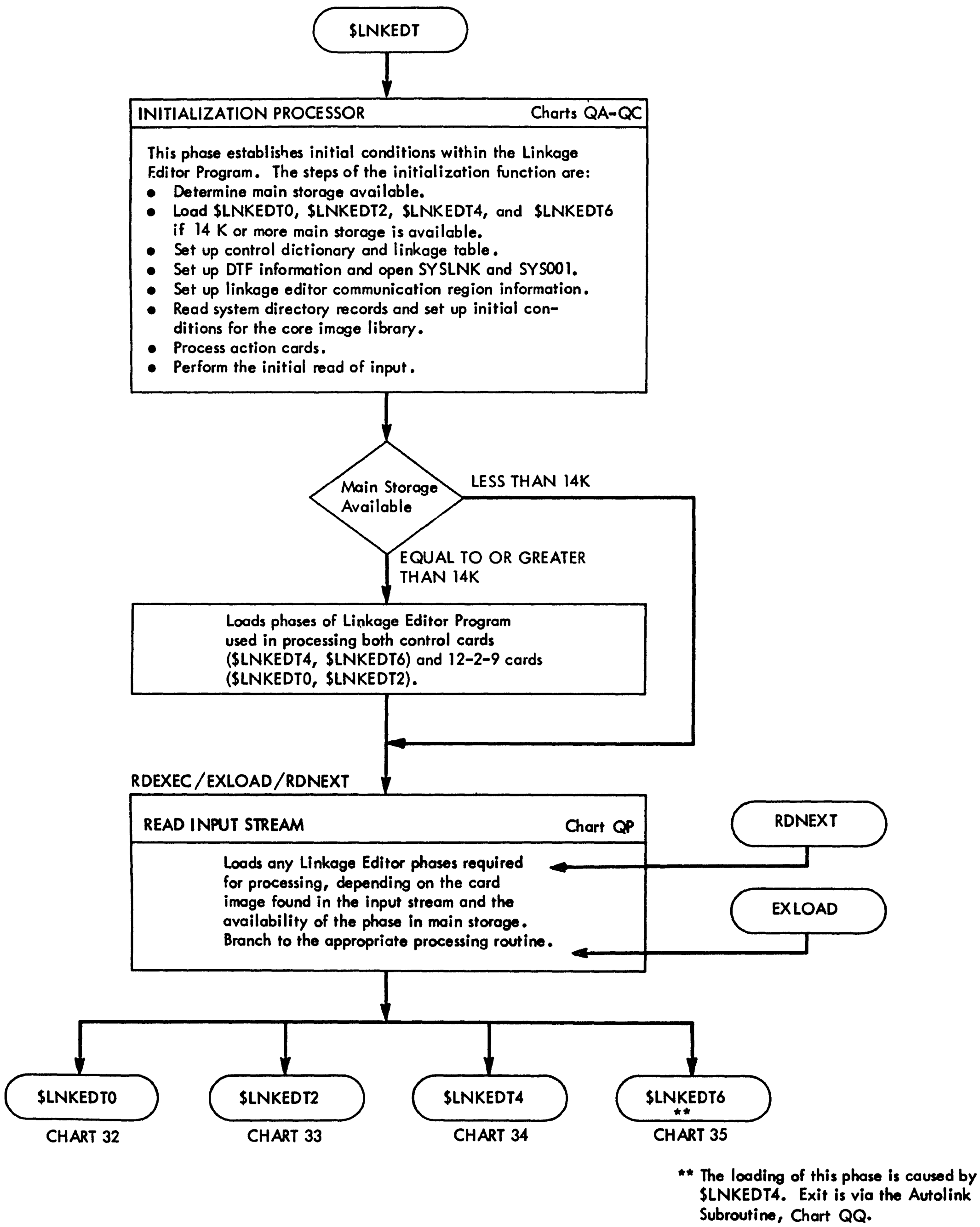



Chart 31. Linkage Editor - Initialization Phase ($LNKEDT)

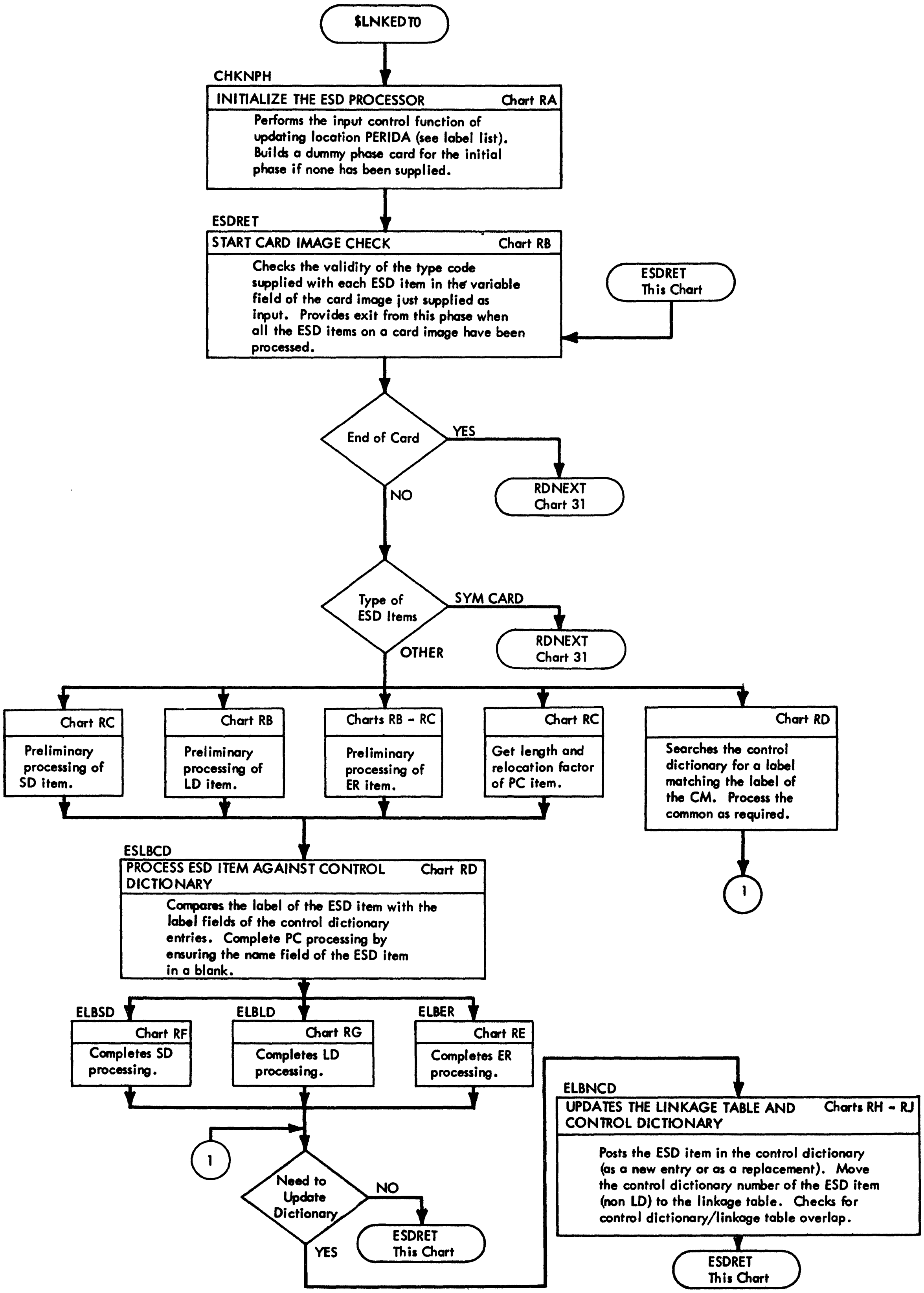



Chart 32. Linkage Editor - ESD Processing Phase ($LNKEDT0)

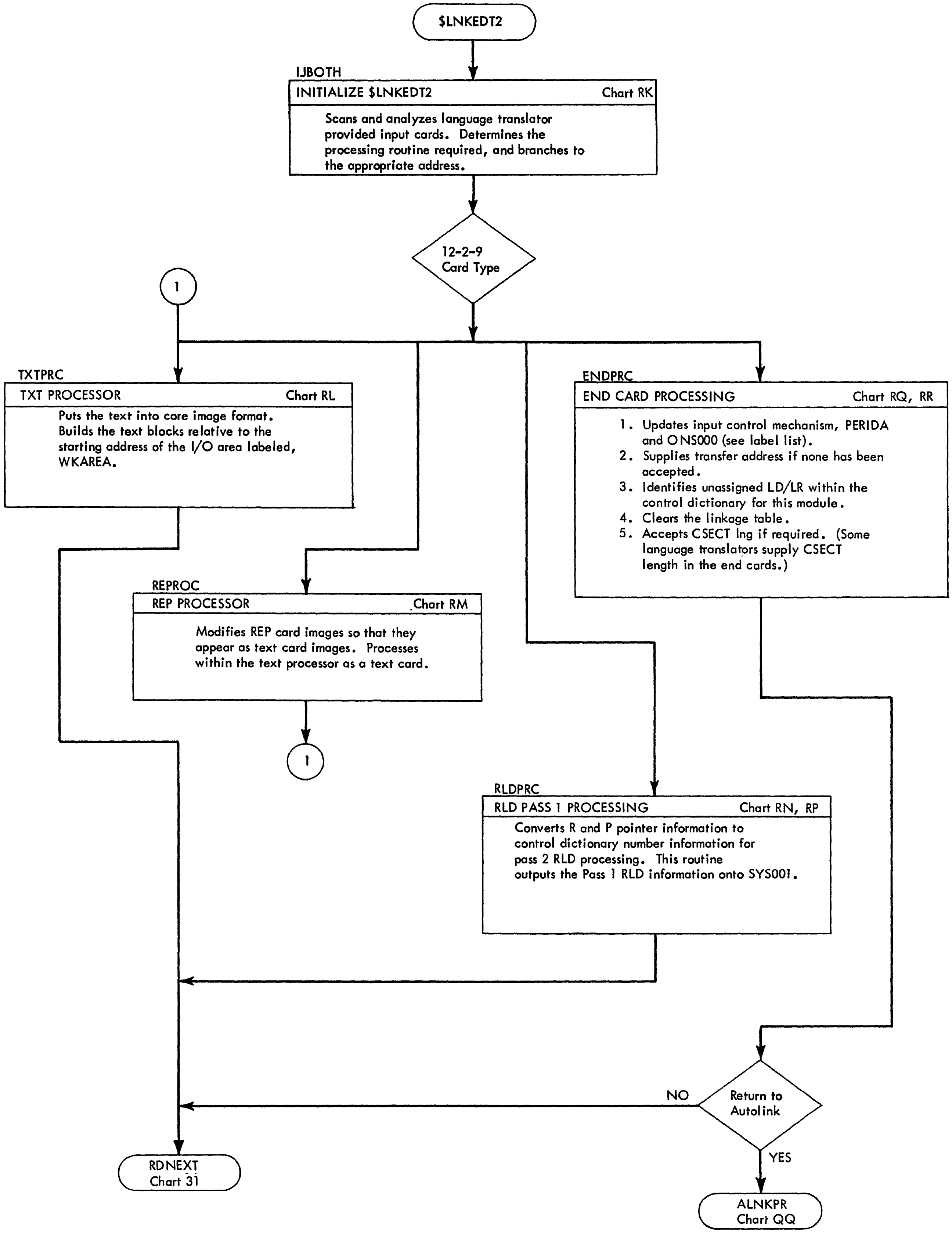



Chart 33. Linkage Editor - TXT, REP, RLD, and END Processing Phase ($LNKEDT2)

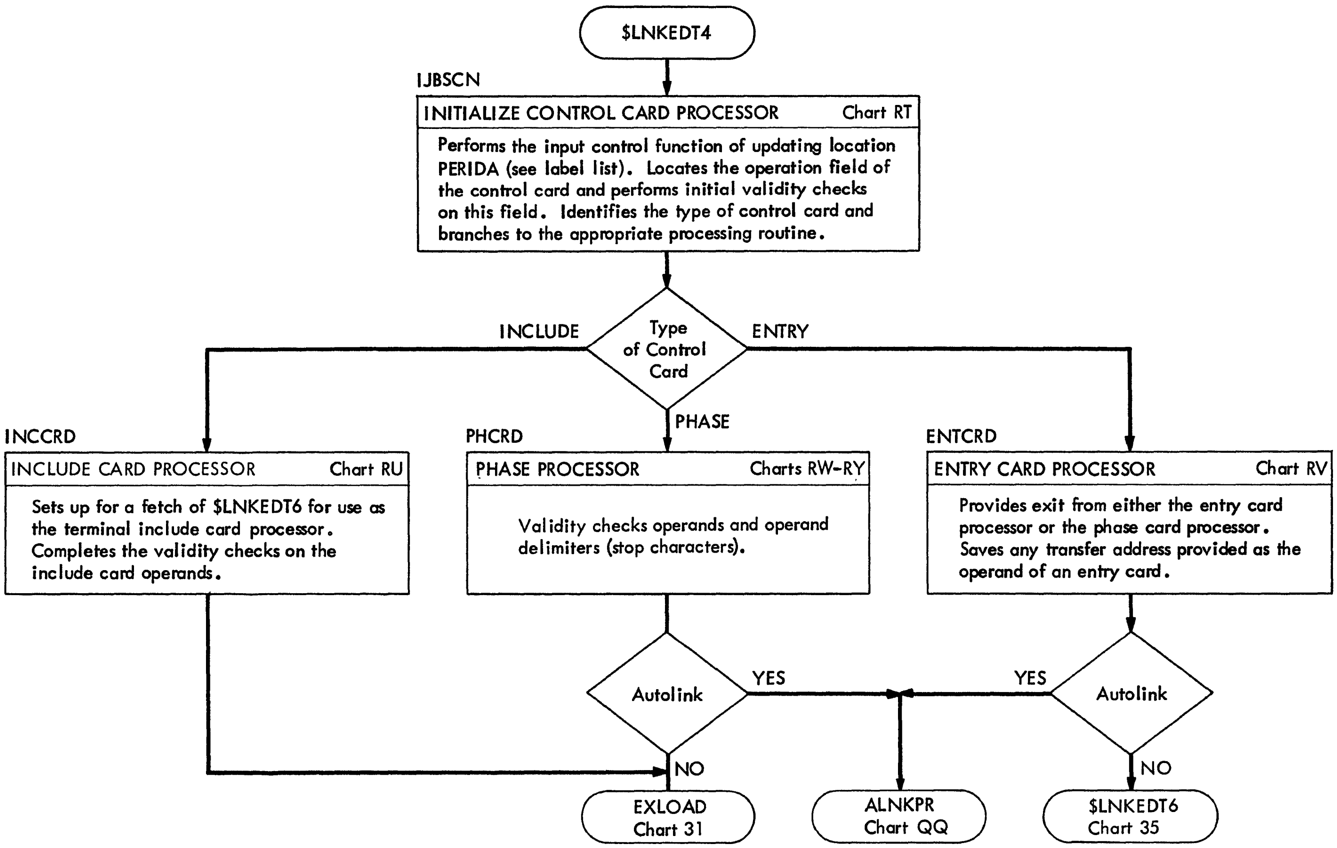



Chart 34. Linkage Editor - Control Statement (INCLUDE, PHASE and ENTRY) Scan and Processing Phase ($LNKEDT4)

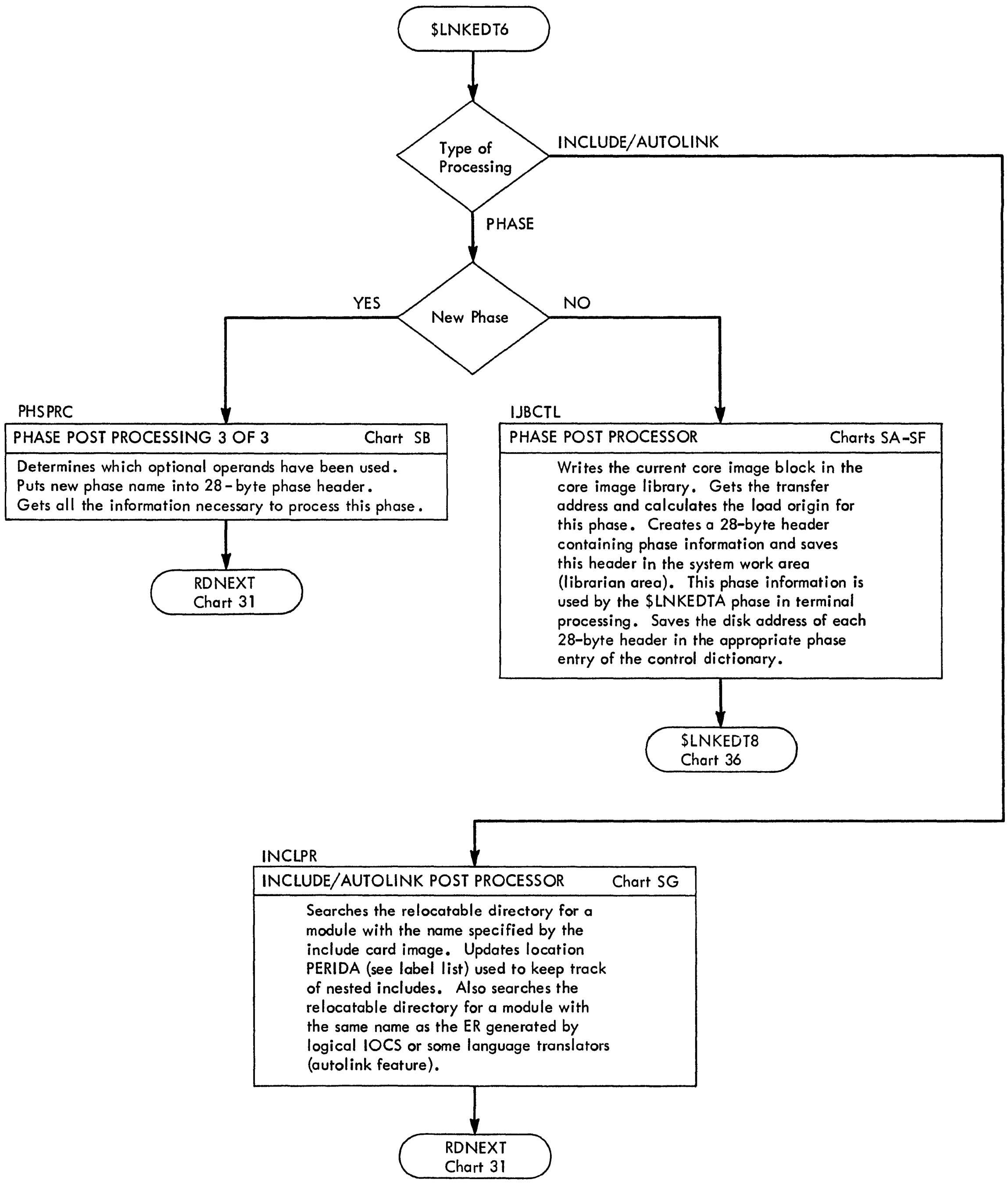



Chart 35. Linkage Editor - End of Control Statement Processing Phase ($LNKEDT6)

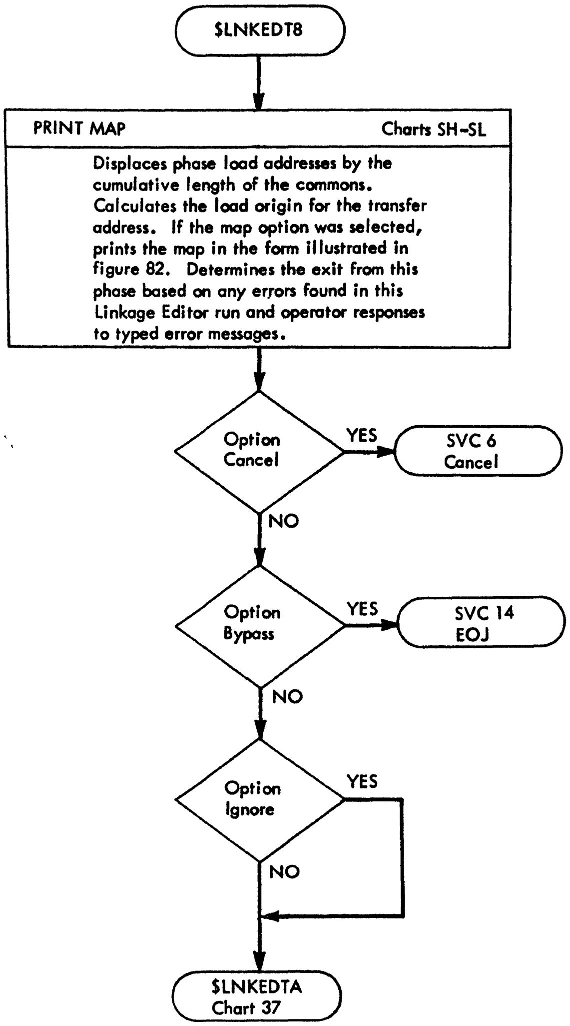



Chart 36. Linkage Editor - Print Map Phase ($LNKEDT8)

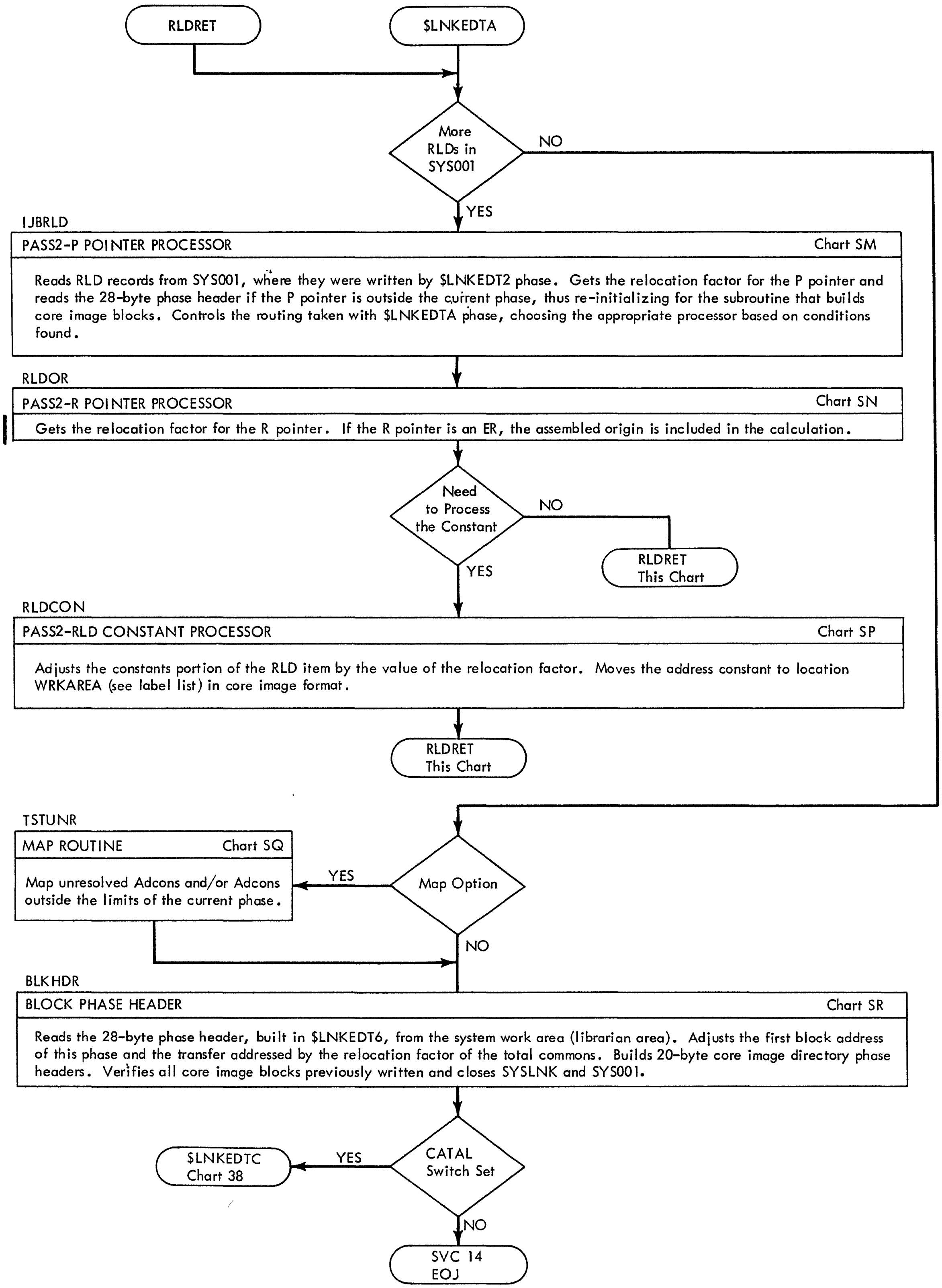



Chart 37. Linkage Editor - Pass 2 RLD and Terminal Processing Phase ($LNKEDTA)

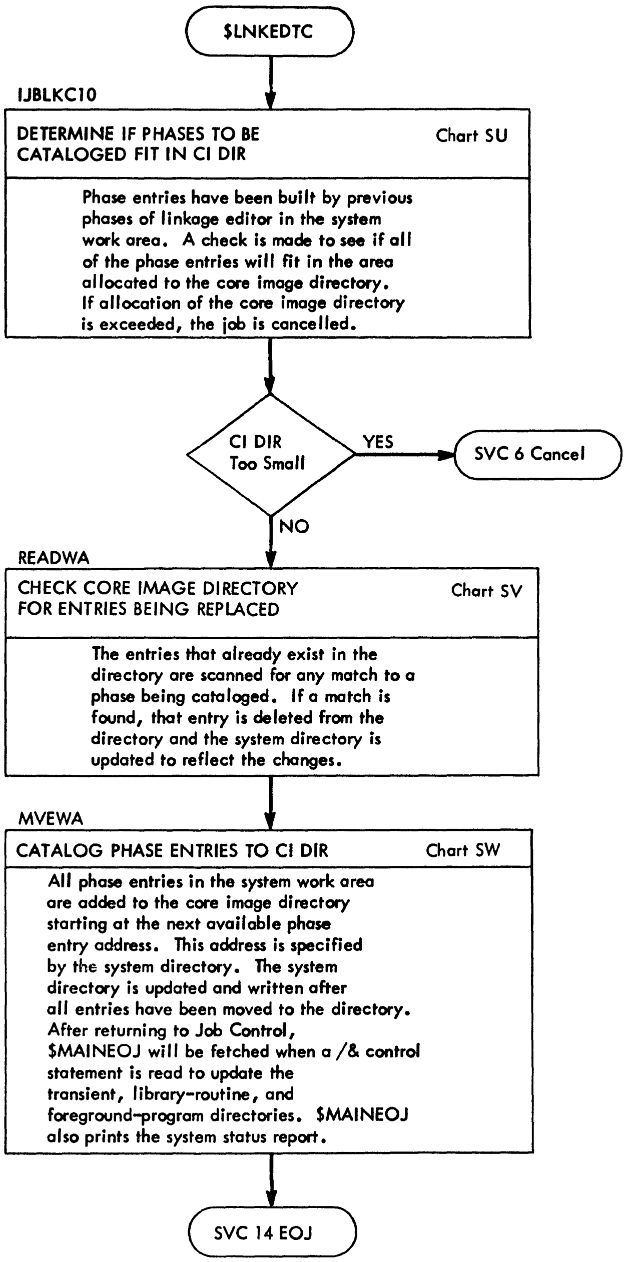



Chart 38. Linkage Editor - Catalog Core Image Directory Phase ($LNKEDTC)

This section presents the programs that perform the functions required for maintaining the libraries and directories of SYSRES. These functions are as follows:

- Catalog function for all libraries except the core image library. The core image library catalog function is performed by the linkage editor program (Phase 8, $LNKEDTC).
- Delete function for all libraries
- Rename function for all libraries
- Reallocate function for all libraries
- Condense function for all libraries

Maintenance of the system, in certain cases, will cause the directories to be incompatible with their corresponding libraries. These cases occur, in particular, when the reallocation program (MAINTA) and the condense program (MAINTCN) are being executed. If the execution of either of these programs is not completed, the status of the system is unpredictable and the system may have to be rebuilt. It is therefore imperative that during the execution of either program, the supervisor be prevented from fetching any transient. To safeguard against these incompatibilities, PIOCS performs the following:

- Masks attention if bit 6 of the linkage control byte (displacement 57 of the communication region) is on. This bit is turned on by both programs and is turned off by $MAINEOJ (the program that updates both the $ and $$ directories).
- Enters the system into a "hard wait" if the linkage control byte has a configuration of X'FF' when an I/O error occurs on SYSRES. This byte is set to X'FF' during reallocation involving the core image directory. It is restored to its original configuration when the reallocation is completed.

The programs included in this section are presented in the following order:

- Common library maintenance program (MAINT)
- Core image library maintenance program (MAINTC2)
- Relocatable library maintenance program (MAINTR2)
- Source statement library maintenance program (MAINTS2)
- System reallocation program (MAINTA)
- Library condense program (MAINTCN)
- Set condense limits program (MAINTCL)
- Update subdirectories program ($MAINEOJ)

## COMMON LIBRARY MAINTENANCE PROGRAM (MAINT), CHART 39

During execution of all maintenance functions (except when $LNKEDTC is used to catalog in the core image library and when $MAINEOJ is resident), MAINT is resident in storage with one of the following maintenance programs:

- MAINTC2
- MAINTR2
- MAINTS2
- MAINTA
- MAINTCN
- MAINTCL

The MAINT root phase is composed of 3 CSECTS:

1. LIOCS logic module (GET or PUT)
2. Error message routine (ERRRTN)
3. Card handling and fetch routine (MAINT)

MAINT is fetched from SYSRES by job control when a //EXEC MAINT control statement is read. The MAINT root phase performs the following functions:

1. Loads base registers of phases.
2. Reads control cards from SYSRDR or SYSIPT.
3. Analyzes the operation field in control statements.

4. Loads or fetches appropriate phases.

5. Branches to appropriate entries in phases.

6. Sets up pointers to operands in control statements.

7. Writes error messages.

8. Performs I/O operations for GET and PUT macros issued in the phases.

Refer to Figure 58 for a storage map showing the relationship of MAINT and its phases. Refer to Chart 00 for interaction between MAINT and other programs in the system.

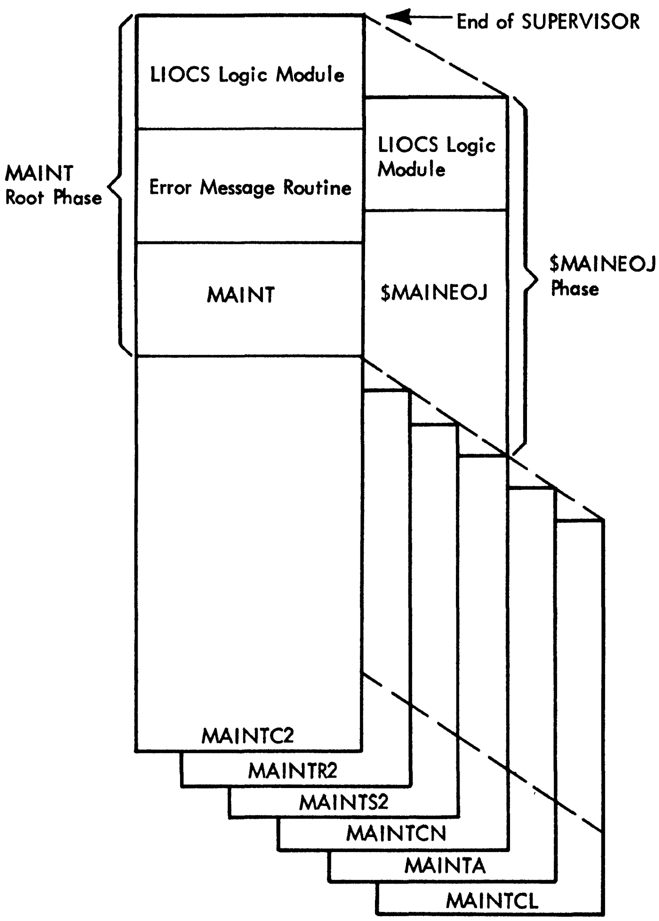



Figure 58. Maintenance Storage Map

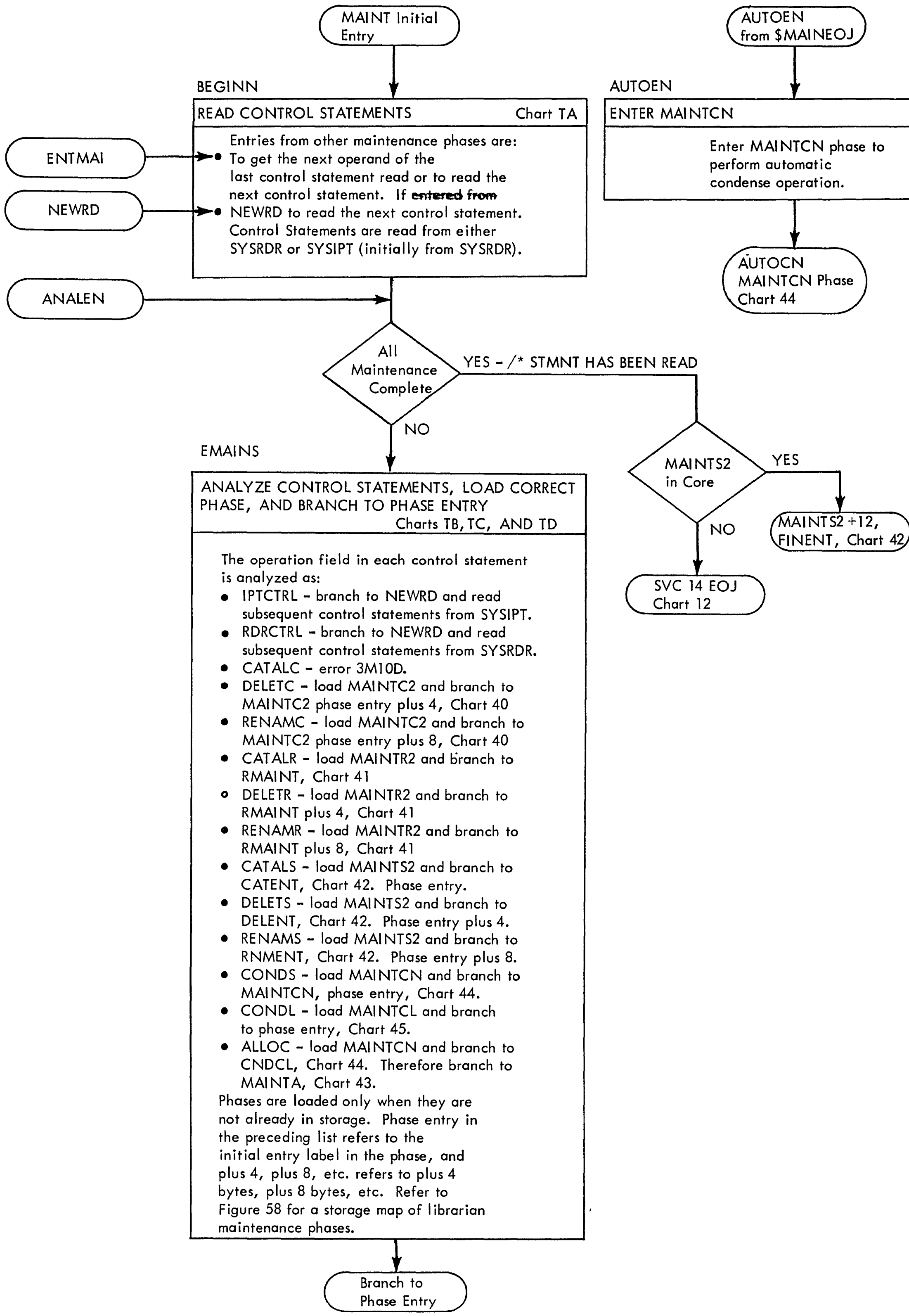



Chart 39. Common Library Maintenance Program (MAINT)

| Function | Element | Control Statements Required |
|---|---|---|
| Delete | Phase | DELETC phase 1 [,phase2,...] |
| | Program | DELETC XXXX.ALL[,YYYY.ALL,...] |
| Rename | Phase | RENAMC old name, new name [,old name2, new name2,...] |

Figure 59. Core Image Library Maintenance Control Statements

CORE IMAGE LIBRARY MAINTENANCE PROGRAM (MAINTC2), CHART 40

MAINTC2 is fetched from SYSRES by the root phase MAINT when a control statement concerning core image library maintenance is read by MAINT. MAINTC2 deletes or renames phases in the core image library. Phases are cataloged in the core image library only by $LNKEDTC. See Figure 59 for control statements acceptable to MAINTC2.

There may be any number of these control statements in any sequence. The operands in DELETC control statements may be in any sequence. Preceding these control statements is the job control statement //EXEC MAINT. A /* (end-of-file) always follows the statements.

Refer to Figure 6 for the format of the core image (CI) directory and to Figure 4 for the format of the system directory.

To delete a phase from the library, MAINTC2 deletes the phase entry in the CI directory. To rename a phase, MAINTC2 changes the phase name in the CI directory.

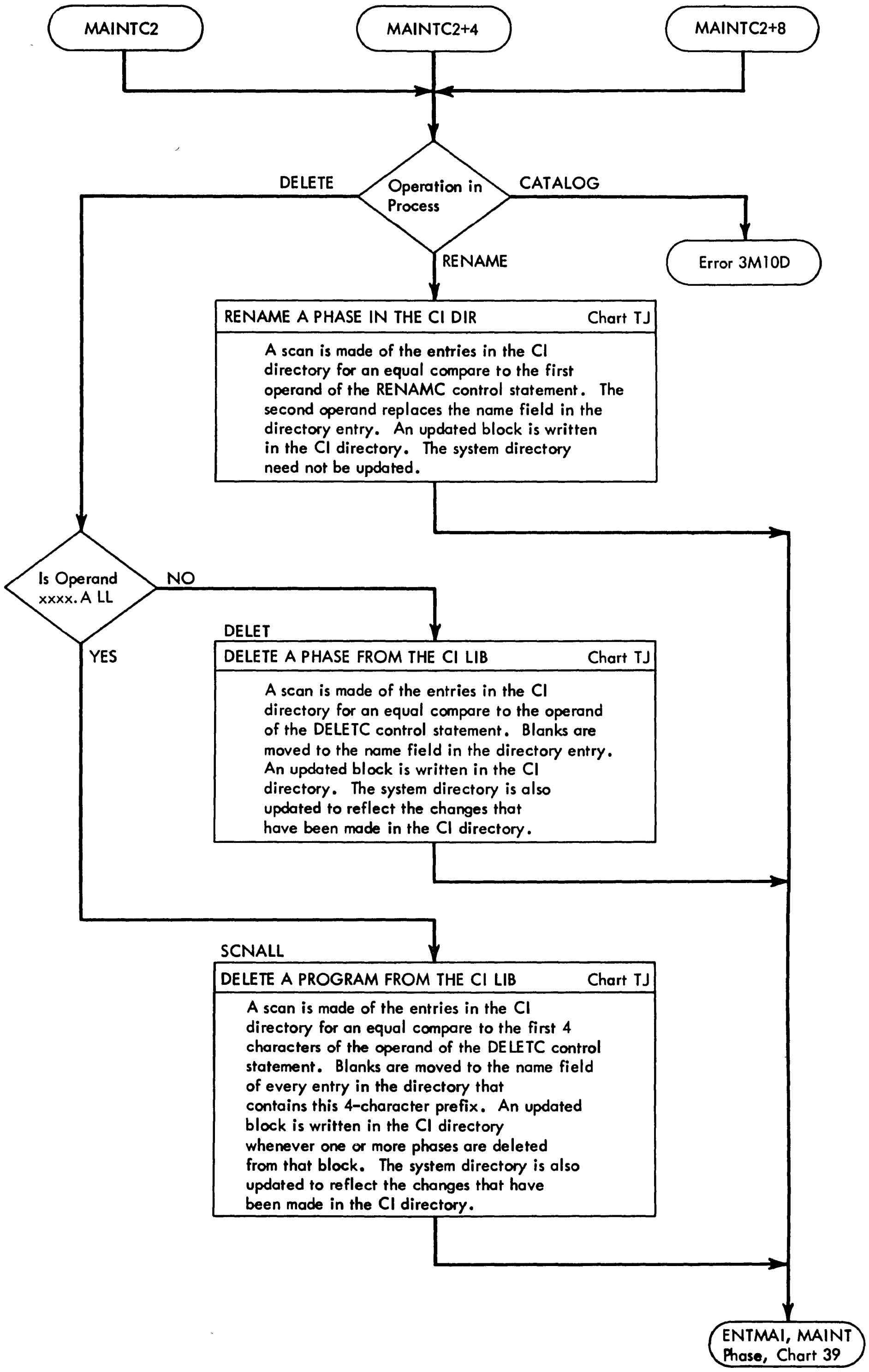



Chart 40. Core Image Library Maintenance Program (MAINTC2)

| Function | Element | Control Statements Required |
|---|---|---|
| Catalog | Module | CATALR module 1 |
| | | CATALR module 2 |
| Delete | Module(s) | DELETR module 1 [, module 2,...] |
| | Program | DELETR XXX.ALL [,YYY.ALL,...] |
| | Library | DELETR ALL |
| Rename | Module | RENAMR old name 1, new name 1 [,old name 2,new name2,...] |

Figure 60. Relocatable Library Maintenance Control Statements

RELOCATABLE LIBRARY MAINTENANCE PROGRAM (MAINTR2), CHART 41

MAINTR2 is fetched from SYSRES by the root phase MAINT when a control statement concerning relocatable library maintenance is read by the root phase MAINT. MAINTR2 catalogs, deletes, or renames modules in the relocatable library. See Figure 60 for control statements for these operations.

There may be any number of these control statements in any sequence. Module names specified in the DELETR control statements may be in any sequence. Preceding these statements is the job control statement //EXEC MAINT. The statements are always followed by a /* (end of file) statement.

Refer to Figure 8 for the format of the relocatable library directory and to Figure 9 for the format of the relocatable library. All records in the relocatable format have the same structure, the only difference being in the length of the variable field. Refer to Figures 62, 63 and 64 for relocatable formats of ESD, TXT, and RLD records. All other records are card images of the input. Figure 61 is an example of a module as it might appear in the relocatable library. Figure 65 is to be used with Chart TS as an aid in determining the new ESID numbers of the ESD records when they are being converted to the library format.

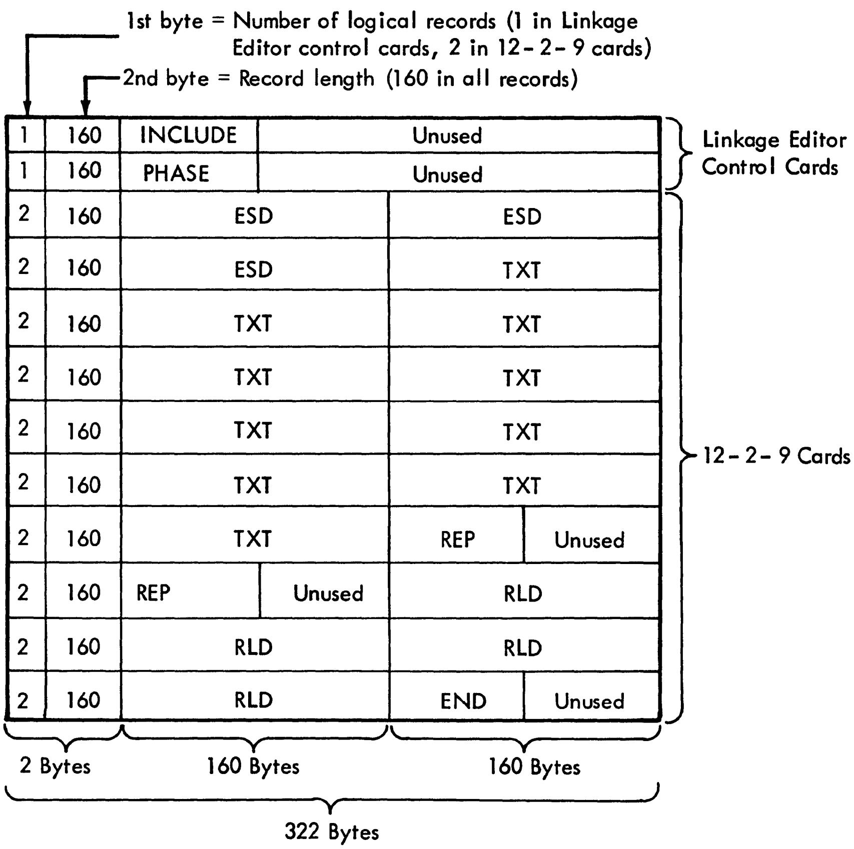



Figure 61. Module in the Relocatable Library

EXTERNAL SYMBOL DICTIONARY

| SYMBOL | TYPE | ID | ADDR | LENGTH | LD ID |
|---|---|---|---|---|---|
| IJBLNK10 | SD | 01 | 001900 | 000928 | |
| IJBLNK | LD | | 001900 | | 01 |
| IJBLOV | LD | | 002008 | | 01 |
| IJBINL10 | SD | 02 | 002228 | 000650 | |
| IJBINL | LD | | 002228 | | 02 |
| IJJCPD3 | ER | 03 | | | |
| IJJCPD1 | ER | 04 | | | |
| IJBESD10 | SD | 05 | 002878 | 000458 | |
| ETC.... | | | | | |

Example of 8 ESD Items from Assembler Output Listing.

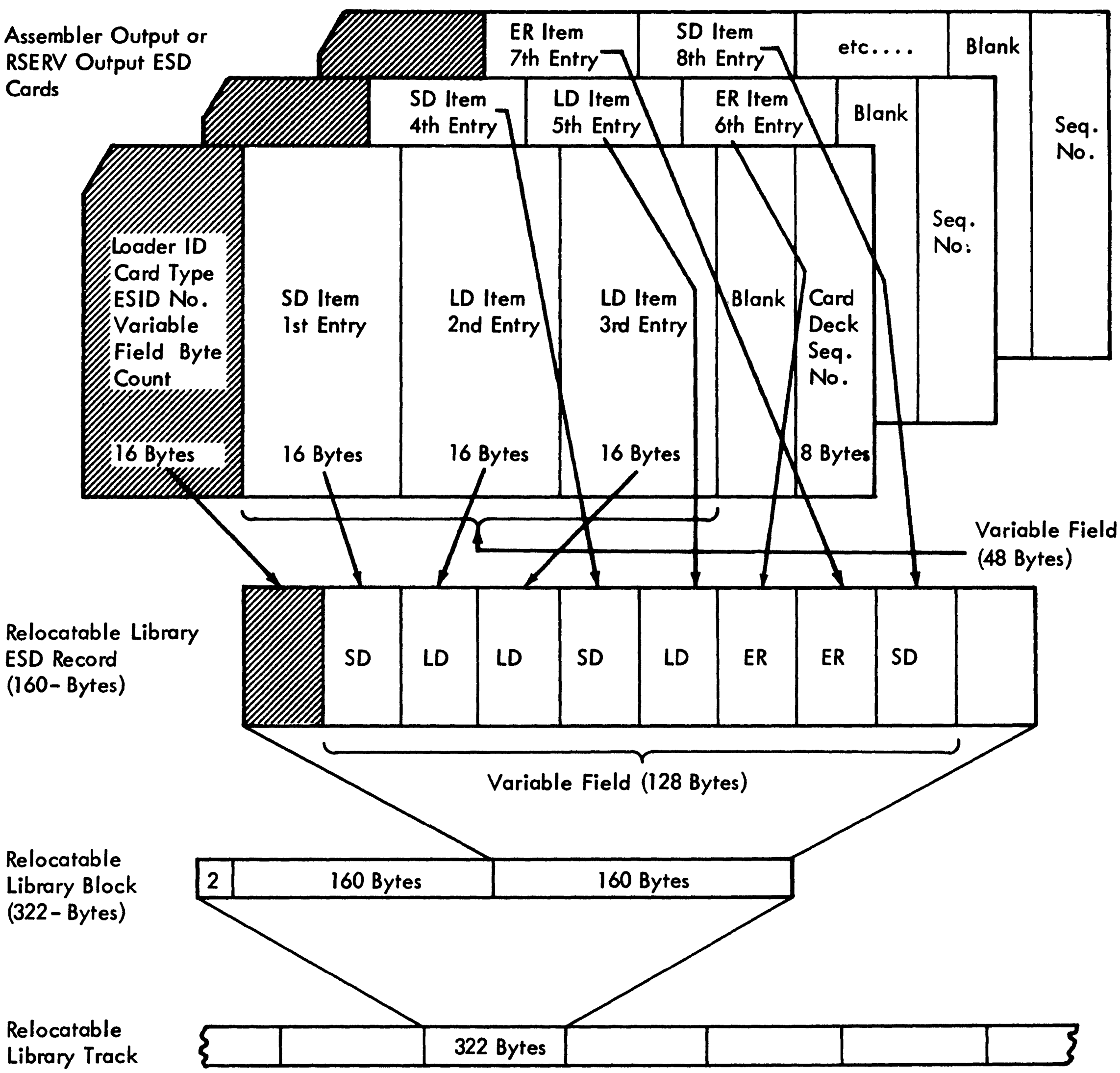



Figure 62. Relocatable Format of ESD Records

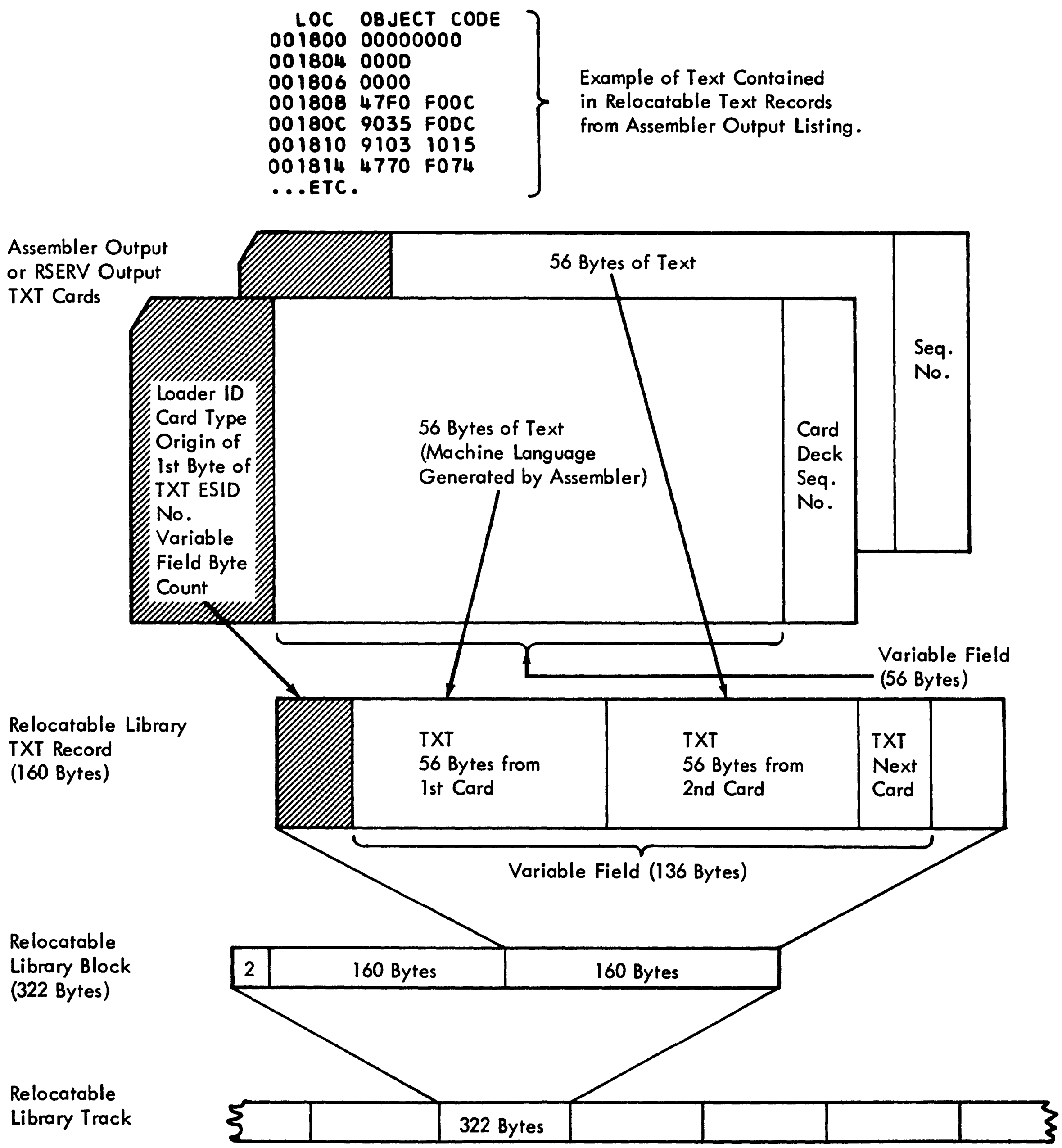



Figure 63. Relocatable Format of TXT Records

RELOCATION DICTIONARY

| POS.ID | REL.ID | FLAGS | ADDRESS |
|---|---|---|---|
| 01 | 01 | 0C | 001928 |
| 01 | 01 | 08 | 001B39 |
| 01 | 02 | 08 | 002168 |
| 02 | 02 | 08 | 0021D5 |
| 02 | 02 | 0C | 0021D8 |
| 02 | 02 | 08 | 002475 |
| 02 | 02 | 0C | 002478 |
| 03 | 03 | 08 | 002899 |
| 03 | 04 | 08 | 0028A0 |

... etc.

Example of RLD items from Assembler output listing

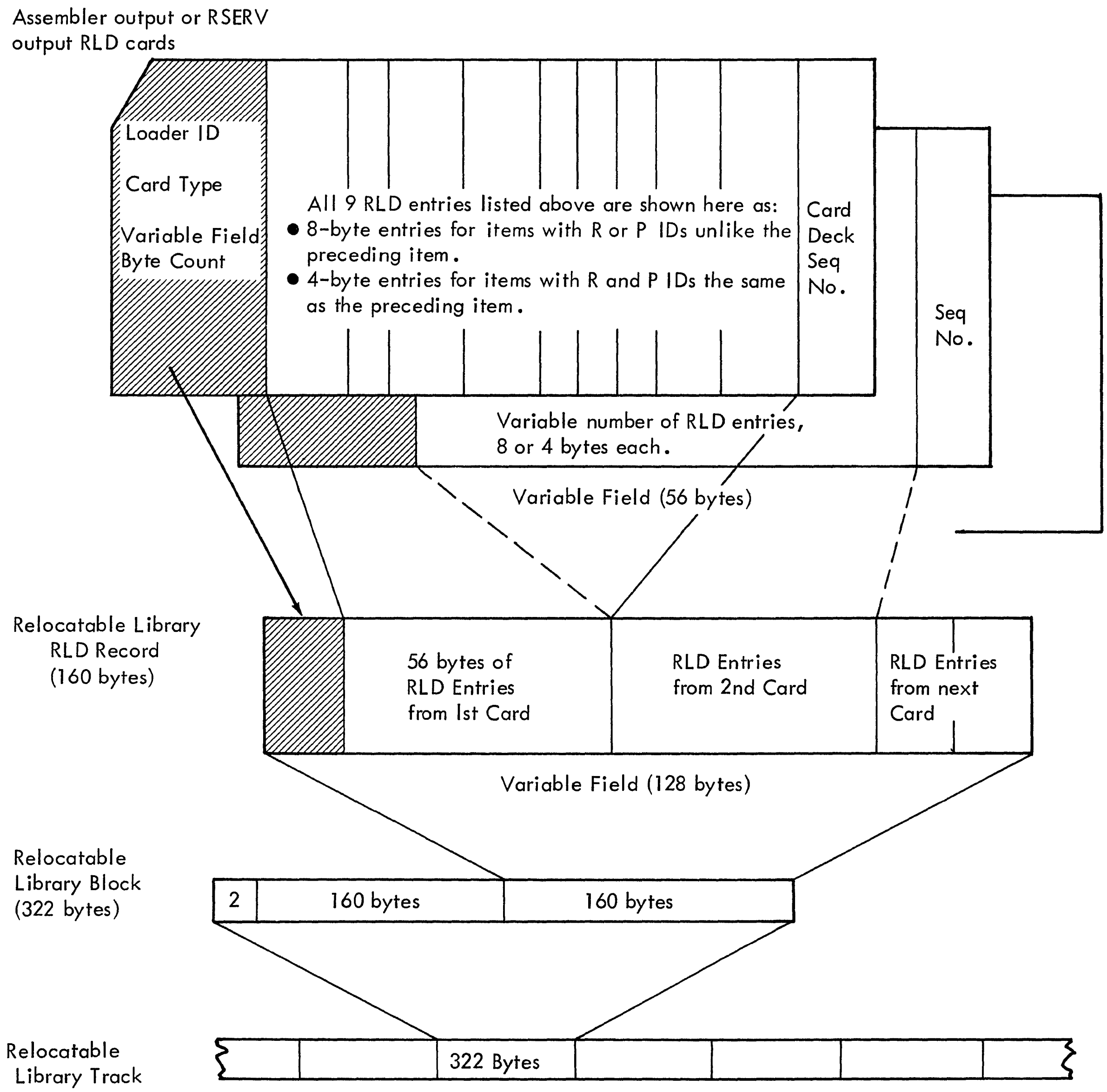



Figure 64. Relocatable Format of RLD Records

The following chart is to be used as an aid in determining the ESID number being calculated on Chart TS and TT in MAINTR2 program. ESID in the chart refers to the ESID number of the input ESD record. The chart is followed by an example where one ESD item from the input record will fill the relocatable library record.

| 1 | 2 | 3 | AREA1 | AREA2 | AREA3 | AREA4 |
|---|---|---|---|---|---|---|
| | | | ESID | ESID | ESID+1 | ESID+1 |
| X | | | blank | ESID | ESID+1 | ESID |
| | X | | ESID | ESID | ESID+1 | ESID+1 |
| | | X | ESID | ESID | blank | ESID+1 |
| X | X | | blank | blank | ESID | ESID |
| X | | X | blank | ESID | blank | ESID |
| | X | X | ESID | ESID | blank | blank |
| X | X | X | blank | blank | blank | blank |

Note: X = LD entries in input record.

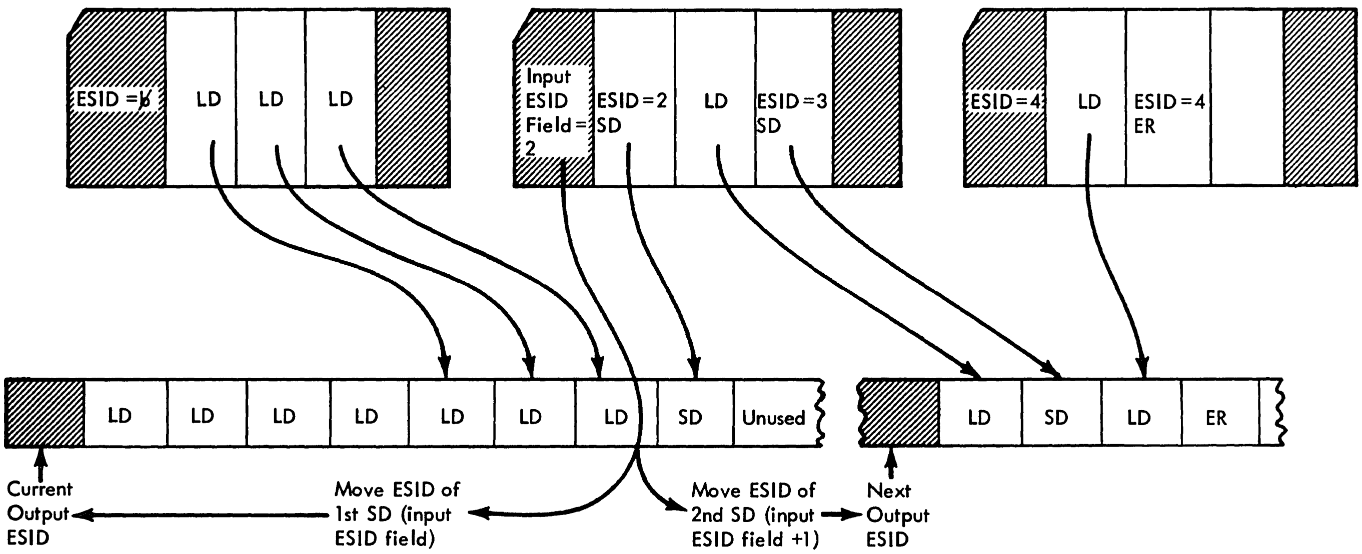



Figure 65. Calculation of ESID Numbers in MAINTR2

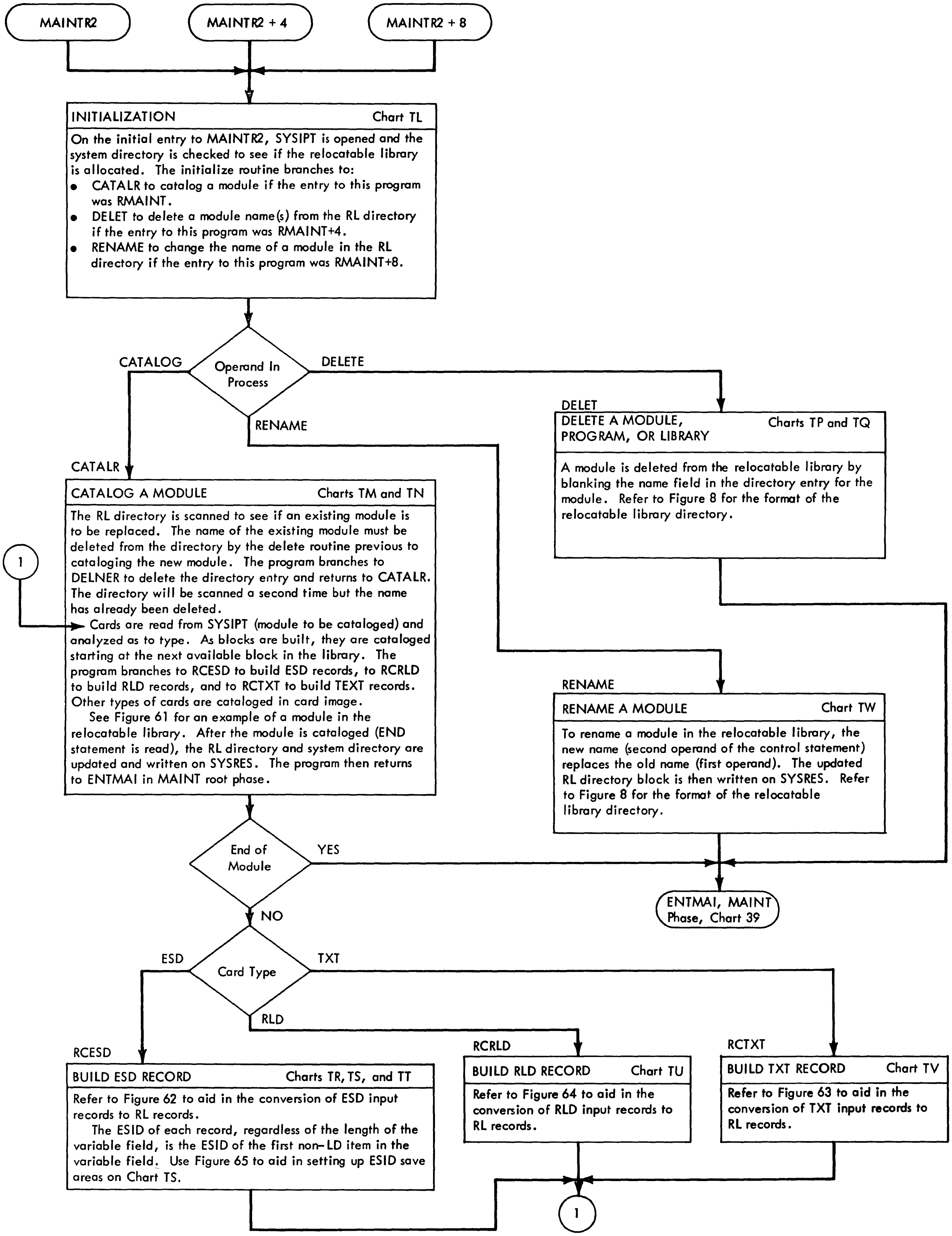



Chart 41. Relocatable Library Maintenance Program (MAINTR2)

SOURCE STATEMENT LIBRARY MAINTENANCE PROGRAM MAINTS2, CHART 42

MAINTS2 is fetched by the root phase MAINT to service the source statement library. It is fetched from SYSRES when a control statement involving the source statement library is read by the MAINT root phase. MAINTS2 catalogs, deletes, or renames books in the source statement library. See Figure 66 for the format of the control statements used in MAINTS2.

| | |
|---|---|
| CATALS sublib. bookname | = Catalog Control Statement |
| DELETS sublib. book1 [,sublib. book2, ...] | Delete Control Statements |
| DELETS sublib. ALL | |
| RENAMS. sublib. oldname, sublib. newname | = Rename Control Statement |

Figure 66. Source Statement Library Maintenance Control Statements

There are two types of information stored in the source statement library: MACRO definition books and source deck books. See Figure 11 for the source statement library format and Figure 10 for the source statement library directory format.

There are two types of source statement library book header cards: BKEND header cards and MACRO header cards. See Figure 67 for header card formats. The BKEND header card provides the user with any or all of the following options:

- Input sequence checking.
- Input card counting.
- Accepting input in compressed format.

The BKEND card or parts of it may be omitted when cataloging, but must be present if the input is in compressed format. A BKEND trailer card must end a book if a BKEND header is used.

A MACRO header card starts a MACRO definition book and the MEND card must be the last card of that book. There may be any number of control statements in any sequence involving the source statement library. However, it is <u>not</u> possible to catalog and to delete the same book from the source statement library in the same job step.

The MAINTS2 phase is completed when a /* or /& card is read. After the end card has been read, the system directory is updated to reflect the changes that were made in the source statement library. Control is returned to the MAINT root phase and MAINTS2 is completed.

| | |
|---|---|
| BKEND [sub.book] [,SEQNCE] [,count [,COMPRSD] | = Book End Header Statement |
| BKEND [sub.book] | = Book End Trailer |
| MACRO | = Macro header statement |
| MEND | = Macro trailer statement |

Figure 67. Book Header Card Formats

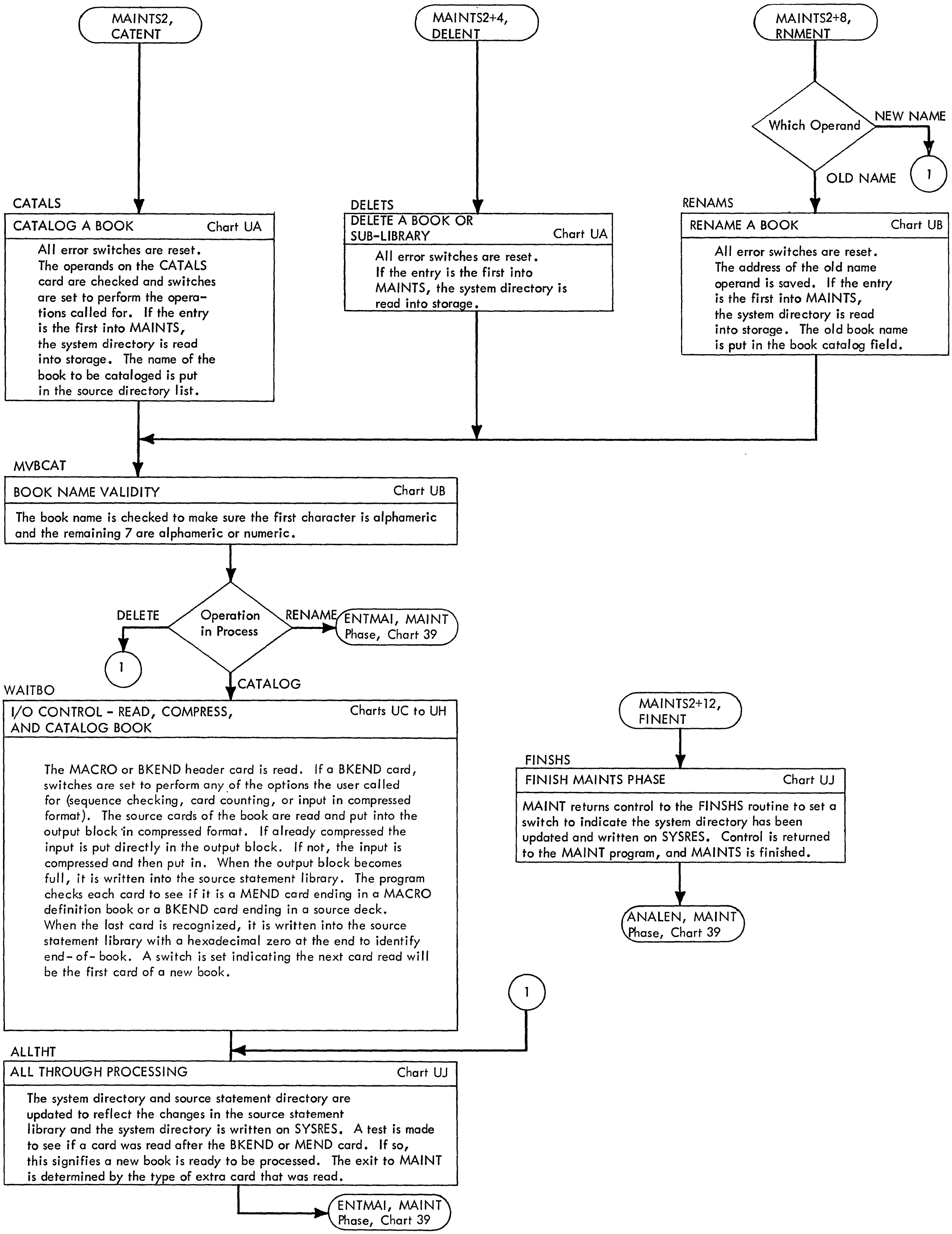



Chart 42. Source Statement Library Maintenance Program (MAINTS2)

```
ALLOC id=cylin(track) [,id=cylin(track),...]

   Where:
   id refers to library identification (either CL, RL, or SL.)

   cylin refers to the total number of cylinders to be allocated to the library,
         including the directory.

   track refers to the number of tracks to be allocated to the directory.

Note: All operands used must be on one control statement.
```

Figure 68. Reallocation Control Statements

SYSTEM REALLOCATION PROGRAM (MAINTA), CHART 43

MAINTA is fetched by the phase MAINTCN. When a control statement requesting system reallocation is read by MAINT, the library condense program (MAINTCN) is fetched to condense all directories and libraries before reallocation. After the condense is complete, MAINTA is fetched from SYSRES by MAINTCN. SYSRES is reallocated by redefining the sizes of the directories and libraries. See Figure 68 for the format of the reallocation control statement.

Control statement input for the reallocation function, read from the device assigned to SYSRDR, is as follows:

```
// JOB jobname

// VOL SYSRES,IJSYSRES

// DLAB 'DOS 16K DISK SYSTEM RESIDENCE
        FILE'

// EXTENT extent information

// EXEC MAINT

   ALLOC  id=cylin (tracks)
          [,id=cylin(tracks)...]

/*

/&
```

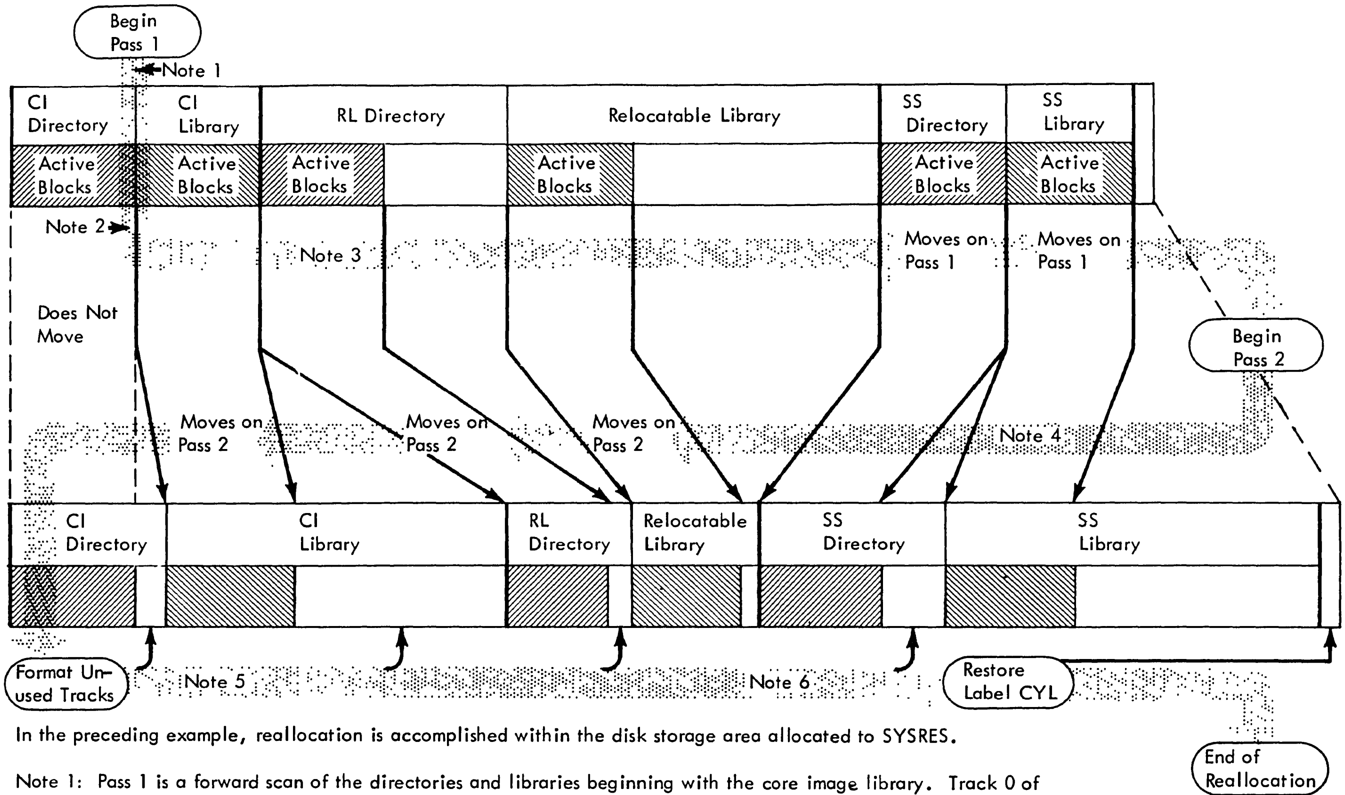
Refer to Figure 69 for the format of the reallocation tables and to Figure 70 for an example of the method used by MAINTA to reallocate SYSRES.

| CDOSA | DISPLACEMENT (DECIMAL) | | DIRECTORY TABLE |
|---|---|---|---|
| RDOSA | | 0 | Old starting address (CCHH) |
| SDOSA | | 4 | New starting address (CCHH) |
| | | 8 | Number of tracks used |
| | | 10 | Number of tracks allocated |
| | | 12 | Number of blocks used |
| | | 14 | Tracks of displacement (Note 1) |
| | | 16 | Block size |
| | | 18 | Update code (Note 2) |
| | | 20 | Number of blocks per track |
| | | 22 | Entry size |
| | | 24 | Number of entries per block |
| | | 26 | Displacement of disk address in entry |
| CLOSA | | | LIBRARY TABLE |
| RLOSA | 0 | 28 | Old starting address (CCHH) |
| SLOSA | 4 | 32 | New starting address (CCHH) |
| | 8 | 36 | Number of tracks used |
| | 10 | 38 | Number of tracks allocated |
| | 12 | 40 | Number of blocks used |
| | 14 | 42 | Tracks of displacement (Note 1) |
| | 16 | 44 | Block size |
| | 18 | 46 | Update code (Note 2) |
| | 20 | 48 | Number of blocks per track |
| | 22 | 50 | Record size |
| | 24 | 52 | Number of records per block |
| | 26 | 54 | Library identification |
| | 28 | 56 | Table for next directory |

Note 1: The tracks of displacement is the number of tracks that must be added to or subtracted from the old disk address (displacement 0) to arrive at the new disk address (displacement 4).

Note 2: The update code is a 0 if the tracks of displacement (displacement 14) is 0, 1 if the tracks of displacement is positive, and 2 if the tracks of displacement is negative.

Figure 69. MAINTA Reallocation Table




In the preceding example, reallocation is accomplished within the disk storage area allocated to SYSRES.

Note 1: Pass 1 is a forward scan of the directories and libraries beginning with the core image library. Track 0 of label cylinder has been stored on cylinder 0.

Note 2: The CI directory will never be moved from its predetermined starting disk address (Cyl 1, track 0.) by MAINTA.

Note 3: On pass 1, all libraries and directories that must be moved to a lower disk address are moved. Only active blocks are moved.

Note 4: On pass 2, all libraries and directories to be moved to a higher disk address are moved. Only active blocks are moved.

Note 5: To format an unused track, the key field and the data field are written in each unused block of the directory or library. The data field is blank except for an asterisk in byte position 1.

Note 6: The relocatable library and the source statement library are not formatted.

Figure 70. Method Used by MAINTA to Reallocate SYSRES

MAINTA

**MAINTA**

**PROCESS ALLOCATE CONTROL STATEMENT** — Charts VA, VB

Refer to Figure 68 for the format of the ALLOC control statement. The operands of the ALLOC statement are analyzed and a table containing the reallocation information is built from the statement.

**GOON1**

**UPDATE RECORD 4 OF SYSTEM DIRECTORY** — Chart VC

All 4 system directory records are read into storage. Using the information stored from the ALLOC statement, record 4 (system allocation record) is updated. Refer to Figure 4 for system directory formats.

**BLDTB**

**BUILD REALLOCATION TABLES** — Charts VD, VE

A reallocation table is built for each of the three directories and each of the three libraries in this sequence:

1. CI directory at label CDOSA
2. CI library at label CLOSA
3. Reloc directory at label RDOSA
4. Reloc library at label RLOSA
5. SS directory at label SDOSA
6. SS library at label SLOSA.

Each of these tables are 28 (decimal) bytes in length. Refer to Figure 69 for the displacement and name of each field in the reallocation tables. Refer to the program listing for initial values in the fields of each table. The tables begin at label CDOSA.

**UPSYSN**

**UPDATE SYSTEM DIRECTORY RECORDS 1, 2, AND 3** — Chart VF

Using the reallocation tables that have been built, system directory records 1, 2, and 3 are updated in storage. Refer to Figure 4 for system directory formats and Figure 69 for the reallocation tables. The first track of the label cylinder is stored on cylinder 0, track 2.

SYSRES is opened to determine if the upper extent of file IJSYSRES is valid. The message 3M66I BEGIN REALLOCATION is printed.

**WRITE**

**WRITE UPDATED SYSTEM DIRECTORY AND BLANK TRANSIENT, LIBRARY ROUTINE, FOREGROUND PROGRAM, AND PROBLEM PROGRAM PHASE DIRECTORIES** — Chart VG

Refer to Figure 3 for allocation of these directories.

1

1

**DIRUP**

**UPDATE LIBRARY DIRECTORIES** — Chart VH

The library directories are read into storage one block at a time. Each entry in the directory block is updated using the reallocation tables. The updated directory blocks are written at the same disk address from where they were read. No relocation of the directories takes place at this time. Refer to Figures 6, 8, and 10 for the formats of the CI library, Reloc library, and SS library directories, respectively. Refer to Figure 69 for the format of the reallocation tables.

**MOVE**

**RELOCATE DIRECTORIES AND LIBRARIES** — Chart VJ

Directories and libraries are moved as specified in the reallocation tables. This operation is accomplished in two passes of the tables. Refer to Figure 70.

If the move is to a lower disk address, the move is made on pass 1. If the move is to a higher disk address, the move is made on pass 2. Only the active blocks are moved. If the disk address is to remain the same, that directory or library is bypassed on pass 1 and pass 2. One block at a time is moved. The update disk address subroutine, TKCOMP, can either increment or decrement the disk address by any predetermined displacement.

**TKFMT**

**FORMAT UNUSED TRACKS AND FETCH $MAINEOJ** — Chart VK

The format for unused tracks is the data field with an asterisk in byte position 1. The unused tracks are formatted in all directories and in the CI library in this sequence:

1. CI directory
2. CI library
3. RL directory
4. SS directory

Restore first track of label cylinder from cylinder 0, track 2 to new cylinder. The message 3M67I END REALLOCATION is printed.

$MAINEOJ is fetched to update the transient library routine, and foreground program directories.

NEWRD, MAINT Phase, Chart 39

Chart 43. System Reallocation Program (MAINTA)

## LIBRARY CONDENSE PROGRAM (MAINTCN), CHART 44

MAINTCN is fetched from SYSRES by the root phase MAINT when a control statement requesting a condense or reallocation function is read. MAINTCN is also loaded, together with MAINT, when $MAINEOJ requests an automatic condense. MAINTCN condenses any or all of the libraries and their respective directories. See Figure 71 for control statements that cause MAINTCN to be fetched.

| Function | Element | Control Statements Required |
|---|---|---|
| Condense | Core Image Library | CONDS CL |
| | Relocatable Library | CONDS RL |
| | Source Statement Library | CONDS SL |
| | All Libraries | CONDS CL, RL, SL |
| Reallocate | All Libraries | Any ALLOC control statement |

Figure 71. Condense Control Statements

Refer to Figure 4 for the format of the system directory. Refer to Figures 6, 8, and 10 for formats of the core image directory, relocatable directory, and source statement directory, respectively. Refer to Figures 7, 9, and 11 for formats of the core image library, relocatable library, and source statement library, respectively.

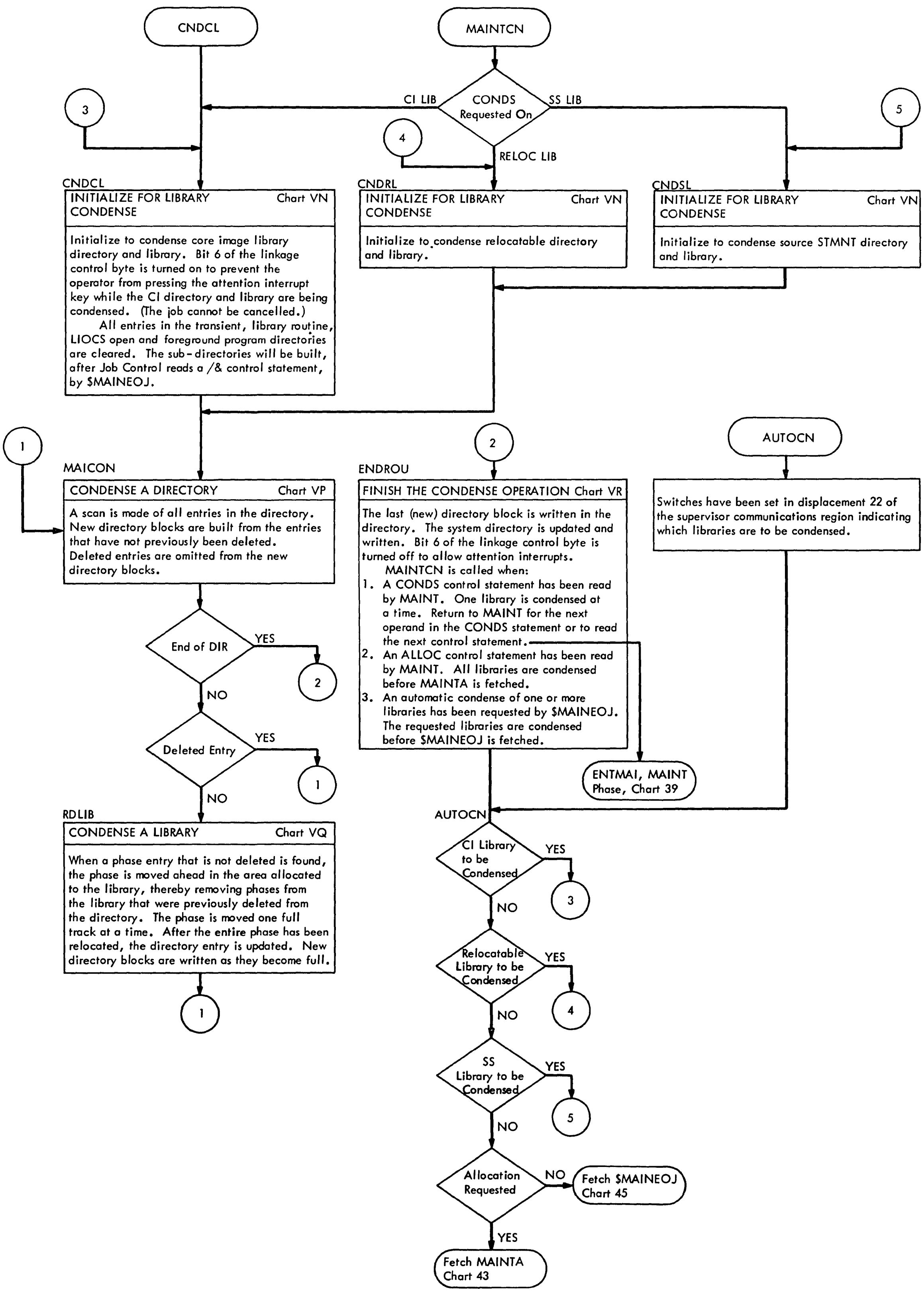



Chart 44. Library Condense Program (MAINTCN)

SET CONDENSE LIMITS PROGRAM (MAINTCL), CHART 45

MAINTCL is fetched from SYSRES by the root phase MAINT when a control statement requesting the setting of the automatic condense limits in the system directory is read. See Figure 4 for the format of the system directory. MAINTCL sets condense limits for all or any of the libraries. The control statement that causes MAINTCL to be called is:

```
CONDL    CL=nnnn,RL=nnnn,SL=nnnn
```

Whenever the number of active entries is equal to or less than the condense limits set by MAINTCL, $MAINEOJ calls MAINTCN to condense.

UPDATE SUB-DIRECTORIES PROGRAM ($MAINEOJ), CHART 45

$MAINEOJ is fetched by:

- Job Control at the completion of MAINT or $LNKEDTC.
- CORGZ after copying the system to SYS002.
- MAINTCN after performing an automatic condense operation requested by $MAINEOJ.

Objectives of $MAINEOJ are:

1. Determine if the condense limits in the system directory have been exceeded and call MAINTCN if an automatic condense is necessary.

2. Build subdirectories by scanning the core image directory and extracting those entries that belong in the subdirectories. Refer to Figure 3 for the organization of the directories on SYSRES. Subdirectories maintained in this system are:

   - Transient subdirectory ($$)
   - LIOCS open subdirectory ($$BO)
   - Library routine subdirectory ($)
   - Foreground program subdirectory (FGP)

   Note: The library routine subdirectory may be referred to as the preferred program directory.

3. Print a system status report on SYSLST using the entries in the system directory. Refer to Figure 76 for a sample printout of the system status report.

4. Issue a SVC 14 (EOJ). This is the completion of the maintenance run.

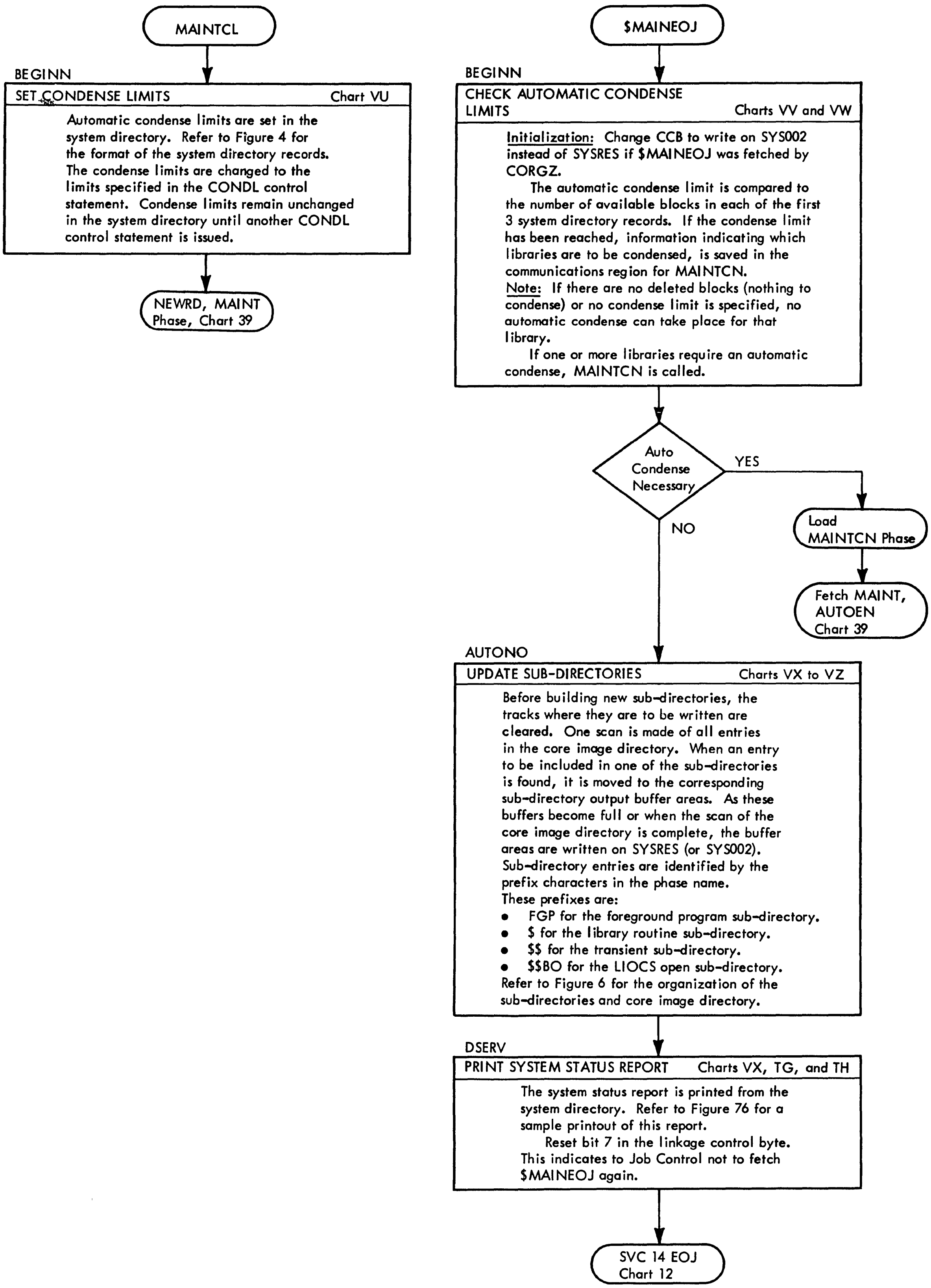



Chart 45. Update Directory and Set Condense Limit Programs ($MAINEOJ and MAINTCL)

The copy program (CORGZ) is the only program that is presented in this section.

## COPY SYSTEM PROGRAM (CORGZ), CHARTS 46 AND 47

The copy system program selectively or completely copies the system residence onto another disk pack. In addition to the copy function, the ability to define the limits for the new disk pack (allocation) is provided. All $ and $$ phases of the core image library and the standard label track, track 1 of the volume cylinder, are copied automatically.

The CORGZ program has two phases: CORGZ and CORGZ2. CORGZ consists of three modules, which are link edited into a single phase. The first module, IJBLBB, consists of a common LIOCS logic module used to output an image of erroneous control statements on SYSLST. The second module, IJBLBC, is a common error routine used to display CORGZ error messages on SYSLST. The third module consists of the CORGZ (phase 1) processing routines.

IJBLBB and IJBLBC are common to all maintenance programs. Flowcharts and descriptions of these two modules are contained in Section 6: Common Library Maintenance Program (MAINT).

Phase 2 (CORGZ2) consists of a single module containing the phase 2 processing routines. Figure 72 shows the main storage allocation for both phases of CORGZ.

### I/O Assignments

CORGZ requires the following I/O assignments:

- SYSRES: Must be an IBM 2311 disk pack.
- SYS002: Must be an IBM 2311 disk pack. Data is copied from SYSRES to SYS002. SYSRES and SYS002 can not be the same physical unit.
- SYSRDR: Must be a card reader, a magnetic tape, or a 2311 disk from which CORGZ control statements are read.

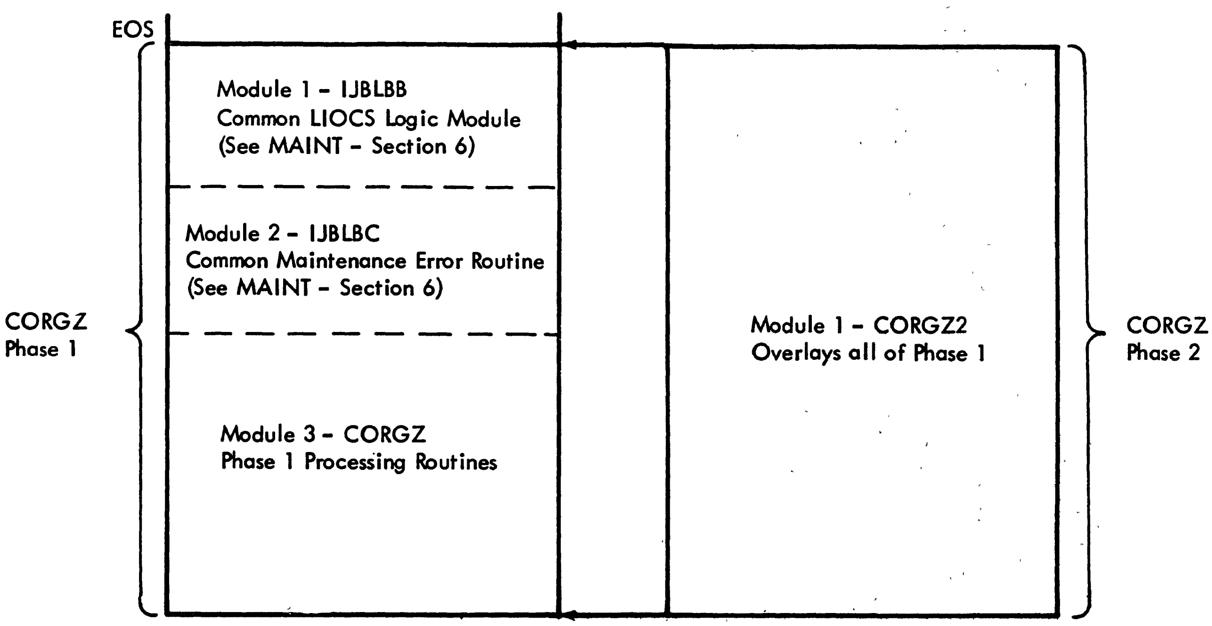



Figure 72. CORGZ Storage Map

* SYSLST: Must be a magnetic tape or a printer. CORGZ diagnostic messages are displayed on this device.

Figure 73 shows the I/O flow for both phases of CORGZ.

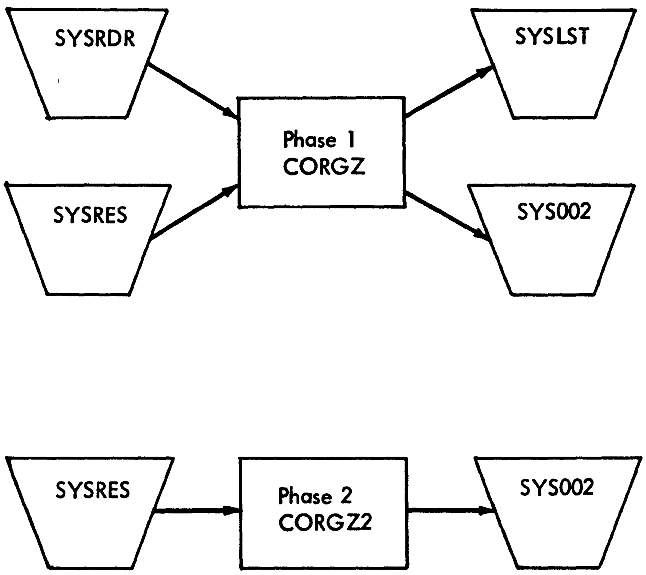



Figure 73. CORGZ I/O Flow

Job Control Statements Required to Request CORGZ

The following job control statements are required to execute CORGZ.

1. //JOB jobname.
2. ASSGN or //ASSGN statements for SYSRDR, SYSLST, and/or SYS002 if the standard assignments are not to be used.
3. VOL, DLAB, and EXTENT statements for the SYS002 disk pack.
4. //EXEC CORGZ.

Control Statements Acceptable to CORGZ

The CORGZ program is designed to accept two types of control statements: ALLOC and COPY-type statements. The ALLOC statement is optional and is required only if the allocation of one or more of the libraries on SYS002 is different from the allocation on SYSRES. When used, the ALLOC statement, or statements, must precede the COPY-type statement or statements.

ALLOC: The format of the ALLOC statement is shown in Figure 68. The value to be substituted for ID is:

1. CL -- Core image library and directory
2. RL -- Relocatable library and directory
3. SL -- Source statement library and directory

The value substituted for cylinder is:

1. A number representing the total number of cylinders to be allocated for the specified library and directory.
2. The value can not exceed four characters.
3. Each character must have a decimal value zero through nine, inclusive.

The value substituted for tracks is:

1. A number representing the total number of tracks to be allocated for the specified directory.
2. The value can not exceed four characters.
3. Each character must have a decimal value zero through nine, inclusive.

COPY: The valid formats of the COPY statement are shown in Figure 74. The operand of the COPY statement must be ALL. This indicates that all libraries of SYSRES are to be copied.

A COPYC statement specifies the core image library. The operand specifies the phases or programs that are to be copied. An operand of ALL specifies that all phases in the core image library are to be copied.

A COPYR statement specifies the relocatable library. The operand specifies the modules or programs that are to be copied. An operand of ALL specifies that all modules in the relocatable library are to be copied.

A COPYS statement specifies the source statement library. The operand specifies the books to be copied from one or more sublibraries. A complete sublibrary can be copied by the operand: sublibrary .ALL. An operand of ALL specifies that all sublibraries are to be copied.

```
COPY   ALL

COPYC  phase1 [,phase2,...]
COPYC  program1.ALL [,program2.ALL,...]
COPYC  ALL

COPYR  Module1 [,Module2,...]
COPYR  program1.ALL [,program2.ALL,...]
COPYR  ALL

COPYS  sublibrary.book1 [,sublibrary.book2,...]
COPYS  sublibrary.ALL
COPYS  ALL
```

Figure 74. Copy Statement Formats

## Phase 1 of CORGZ (CORGZ)

The //EXEC CORGZ job control statement loads and executes CORGZ. The prime functions performed by this phase are:

1. Initializing, copying IPL, and formatting cylinder 0 of SYS002.
2. Reading and analyzing control statements.
3. Processing ALLOC statements, if present.
4. Generating system directory records 1, 2, 3, and 4 on SYS002.
5. Processing each COPY statement completely before reading another.
   a. The operands of the COPY statement are scanned and processed one at a time.
   b. The SYS002 directories are built as each operand is processed.
   c. When the last operand of a copy statement has been processed, the updated system directory record for this library is written on SYS002 and the next statement is read.
6. Steps 5a, 5b, and 5c are repeated until the /* (EOF) statement is read. At this time, the core image directory entries of all $ and $$ programs are copied to SYS002.

## Phase 2 of CORGZ (CORGZ2)

CORGZ2 overlays phase 1. The prime functions performed by this phase are:

1. Copying the desired core image library phases. The SYS002 core image library directory entries determine the phases to be copied.
2. Copying the desired relocatable library modules. The SYS002 relocatable library directory entries determine the modules to be copied.
3. Copying the desired source statement library books. The SYS002 source statement library directory entries determine the books to be copied.
4. Copying the standard label track (first track) of the volume cylinder.
5. Fetching $MAINEOJ to build the transient and library routine directories on SYS002 and to print the system status report.

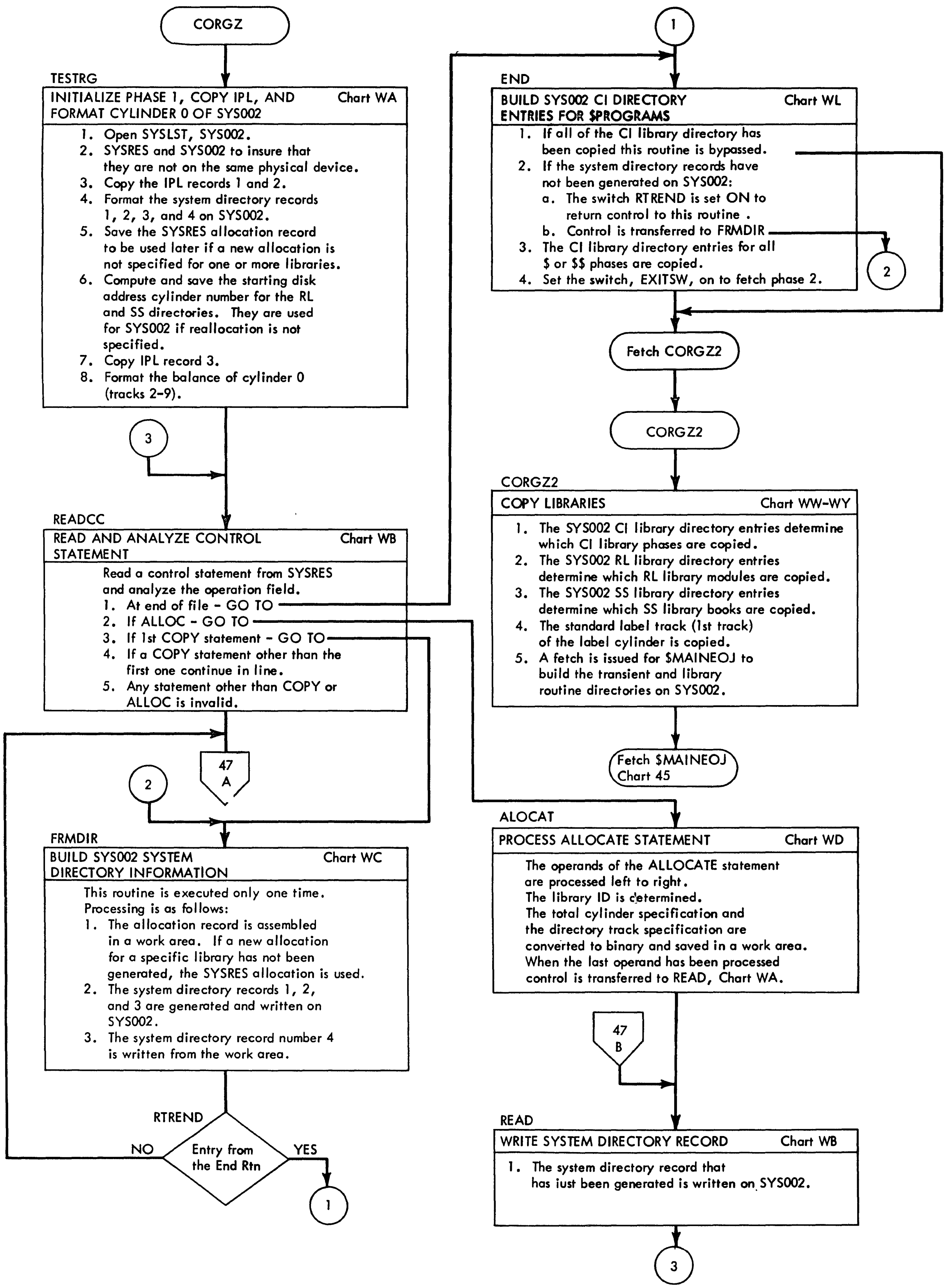
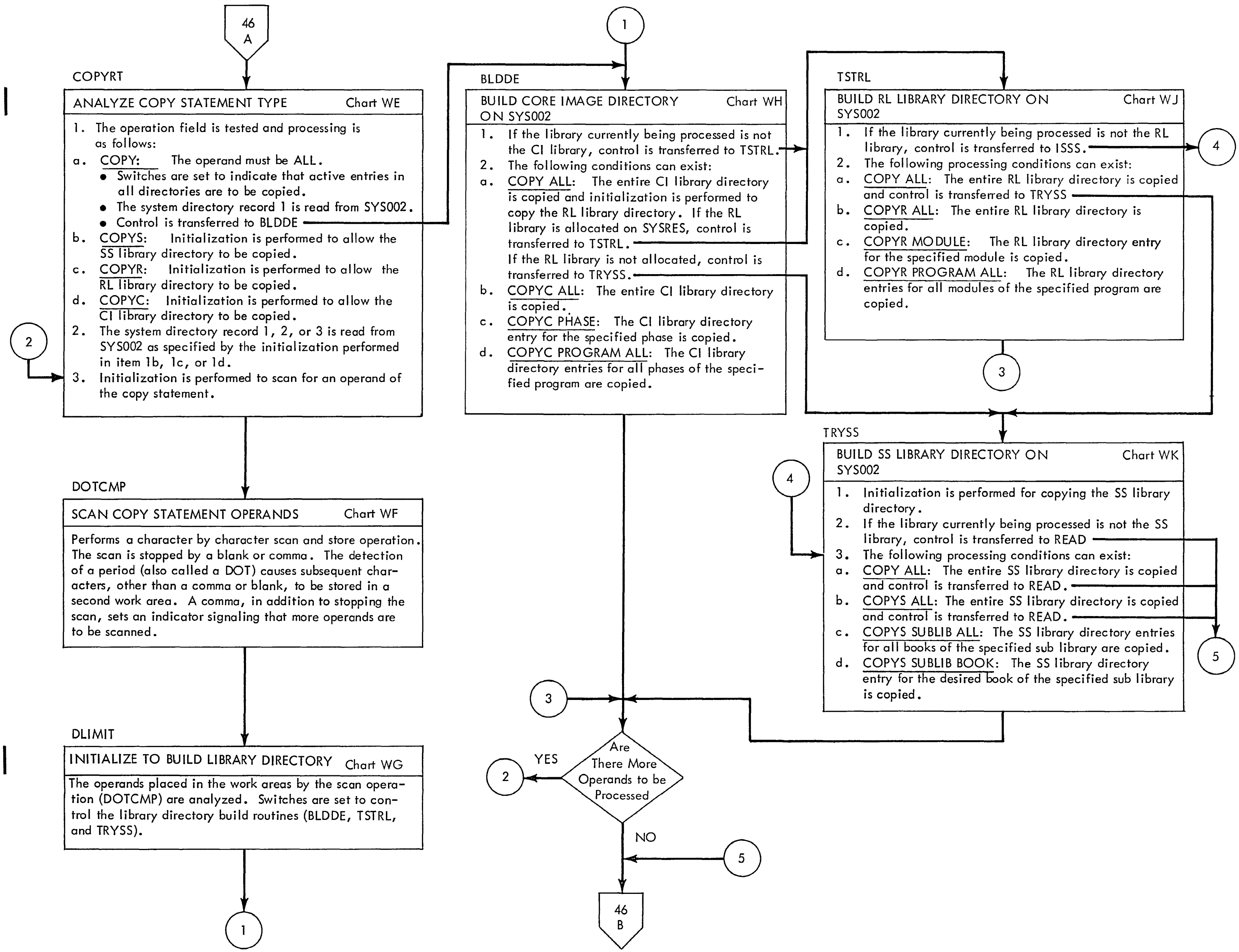



Chart 46. Copy System Program (CORGZ), Part 1 of 2

Chart 47. Copy System Program (CORGZ), Part 2 of 2



This section contains the programs that perform the display and/or punch functions required to maintain SYSRES. Books from the source statement library and modules from the relocatable library can be displayed and/or punched. The contents of any or all directories can be displayed.

The programs that perform these functions are presented in this section in the following order:

- Directory service program (DSERV) - Displays the contents of the directories in SYSRES. All directories can be displayed in a single run or they may be displayed selectively.
- Relocatable library service program (RSERV) - Displays and/or punches modules from the relocatable library.
- Source statement library service program (SSERV) - Displays and/or punches books from the source statement library.

## DIRECTORY SERVICE PROGRAM (DSERV), CHART 48

DSERV is a 1-phase program fetched from SYSRES when a //EXEC DSERV control statement is read by job control. DSERV prints the contents of the following directories from SYSRES:

- System directory
- Transient program directory
- Core image library directory
- Relocatable library directory
- Source statement library directory

The DSERV control statements are shown in Figure 75.

| Function | Element | Control Statements Required |
|---|---|---|
| Print | Transient Directory | DSPLY TD |
| | Core Image Directory | DSPLY CD |
| | Relocatable Directory | DSPLY RD |
| | Source Statement Directory | DSPLY SD |
| Print | ALL Directories | DSPLY ALL |

Figure 75. DSERV Control Statements

Any or all of the valid operands may be in the same control statement and they may be in any sequence. The system directory (system status report) is printed during every DSERV execution. The last DSERV control statement is always a /* (end-of-file). Refer to Figures 4, 6, 8, and 10 for the formats of the system directory, core image library directory, relocatable library directory, and source statement library directory, respectively. The entries in the transient directory have the same format as the core image library directory. Refer to Figure 76 for a sample of the system status report.

| SYSTEM DIRECTORY | CORE-IMAGE | RELOCATABLE | SOURCE-STATEMENT |
|---|---|---|---|
| 11/23/66 | ----------DECIMAL---------- | | |
| | C H R E | C H R E | C H R E |
| DIRECTORY STARTING ADDRESS | 01 00 01 | 42 00 01 | 101 00 01 |
| DIRECTORY NEXT ENTRY | 01 02 04 06 | 42 02 01 01 | 101 00 14 02 |
| DIRECTORY LAST ENTRY | 01 02 08 17 | 42 02 09 22 | 101 00 16 09 |
| LIBRARY STARTING ADDRESS | 01 03 01 | 42 03 01 | 101 01 01 |
| LIBRARY NEXT AVAILABLE ENTRY | 38 09 01 | 93 01 04 | 141 05 11 |
| LIBRARY LAST AVAILABLE ENTRY | 41 09 02 | 100 09 09 | 149 09 16 |
| | ----------STATUS INFORMATION---------- | | |
| DIRECTORY ENTRIES ACTIVE | 345 | 415 | 132 |
| LIBRARY BLOCKS ALLOCATED | 814 | 5283 | 7824 |
| LIBRARY BLOCKS ACTIVE | 716 | 4575 | 6474 |
| LIBRARY BLOCKS DELETED | 36 | 00 | 00 |
| LIBRARY BLOCKS AVAILABLE | 62 | 708 | 1350 |
| AUTOMATIC CONDENSE LIMIT | 00 | 00 | 00 |
| LIBRARY ALLOCATED CYLINDERS | 41 | 59 | 49 |
| DIRECTORY ALLOCATED TRACKS | 03 | 03 | 01 |

Figure 76. System Status Report

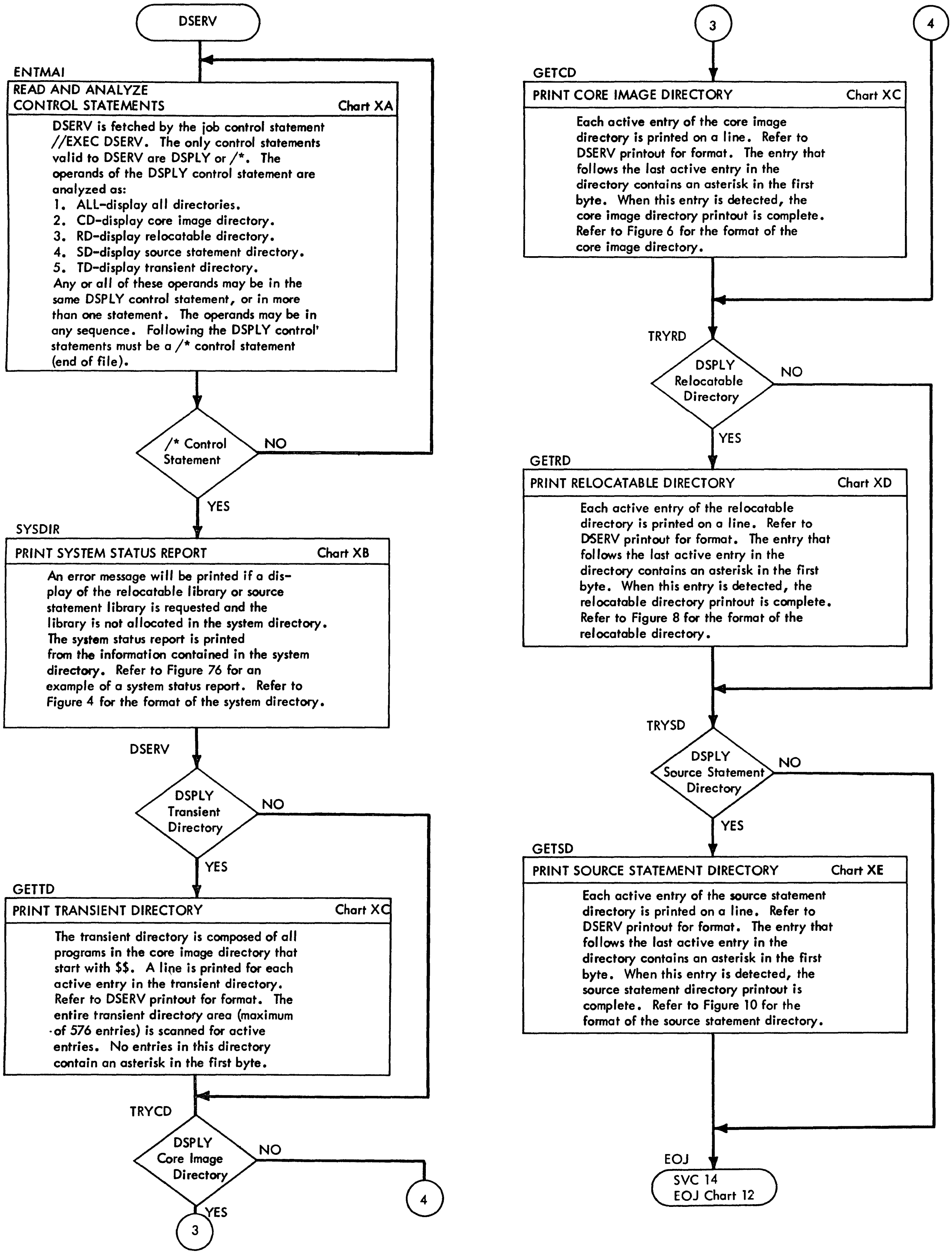



Chart 48. Directory Service Program (DSERV)

## RELOCATABLE LIBRARY SERVICE PROGRAM (RSERV), CHART 49

RSERV is a one phase program fetched from SYSRES when a // EXEC RSERV control statement is read by Job Control. RSERV will print, punch, or print and punch modules from the relocatable library. All punch output will be ejected into stacker two. If SYSRDR and SYSPCH are assigned to the same device, a /* statement will not be punched. The last RSERV control statement, which is a /* statement, will be ejected into stacker two. See Figure 77 for RSERV control statements.

| Function | Element | Control Statements Required |
|---|---|---|
| Print | Module | DSPLY module 1 [,module 2,...] |
| | Program | DSPLY XXX. ALL [,YYY.ALL,...] |
| | Library | DSPLY ALL |
| Punch | Module | PUNCH module 1 [,module 2,...] |
| | Program | PUNCH XXX.ALL[,YYY.ALL,...] |
| | Library | PUNCH ALL |
| Print and punch | Module | DSPCH module 1 [,module 2,...] |
| | Program | DSPCH XXX.ALL[,YYY.ALL,...] |
| | Library | DSPCH ALL |
| Note: XXX, YYY } Represents module prefix | | |

Figure 77. RSERV Control Statements

There may be any number of RSERV control statements in any sequence. Module names specified in the control statements may be in any sequence. If ALL is specified, it must be the first operand. The last RSERV control statement is always a /* (end-of-file).

Refer to Figure 9 for the format of the relocatable library. Each block in the relocatable library contains two logical records that are 160 bytes in length. Refer to Figure 61. All records in the relocatable format have the same structure, the only difference being in the length of the variable field.

RSERV must analyze each record in the module as to type and convert the 160-byte records to smaller records by dividing up the information in the variable field. For example, one ESD record in the relocatable library that contains eight ESD items is punched into three ESD cards. The byte count field in the record must be updated to reflect the change in length of the variable field.

Refer to Figures 62, 63, and 64 for relocatable formats of ESD, TXT, and RLD records. All other records are card images of the input.

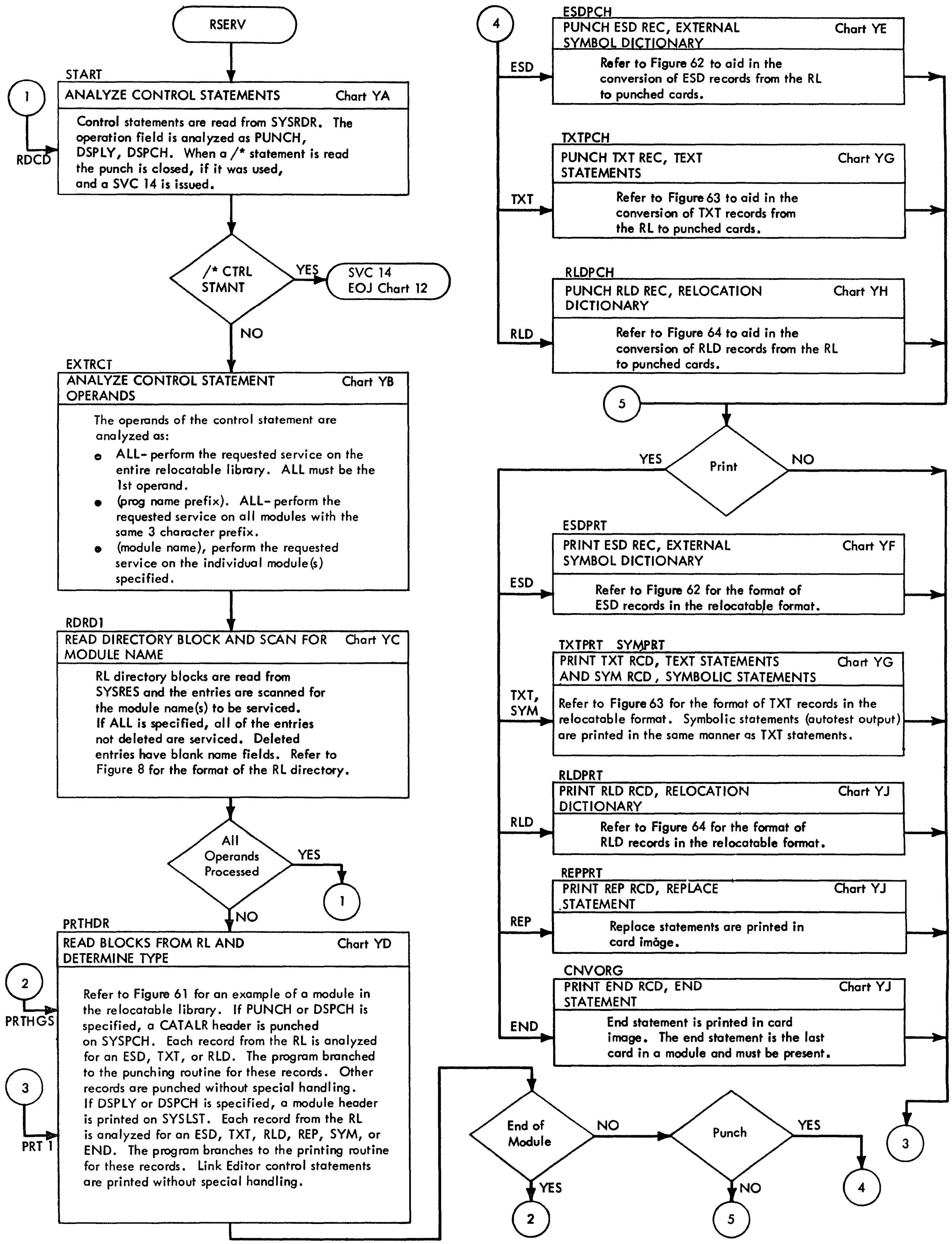



Chart 49. Relocatable Library Service Program (RSERV)

## SOURCE STATEMENT LIBRARY SERVICE PROGRAM (SSERV), CHART 50

SSERV displays books from the source statement library on SYSLST, or SYSPCH, or both. All punch output will be ejected into stacker two. If SYSRDR and SYSPCH are assigned to the same device, a /* statement will not be punched. The last SSERV control statement, which is a /* statement, will be ejected into stacker two. SSERV is a 1-phase program fetched when a //EXEC SSERV card is read by job control. Chart 50 shows the program flow of SSERV.

The control statements requesting services from the source statement library are read from SYSRDR. They may request the displaying of a book or an entire sublibrary. If the operand on the control statement is [,A.ALL] the entire sublibrary is displayed. If the operand is [A.Bookname], the book called for is displayed. The last operand on the control statement is [CMPRSD], if the punched output is to be in compressed format.

The first time a book is to be serviced, a B-transient $$BOPNLB is called to read the system directory and to find the location of the source statement library directory. $$BOPNLB is a B-transient because it is also used by the assembler and the COBOL compiler to locate the source statement library. (For additional information on this B-transient, refer to the DOS LIOCS PLM.)

Because the control statement may have multiple operands, each time a book is serviced, the control statement is checked to see if it contains another operand. If so, the operand is brought in and serviced. When the last operand on a control statement is serviced, the next control statement is read from SYSRDR. When an EOF (/*) condition is encountered on SYSRDR, the SSERV program is terminated and job control is fetched.

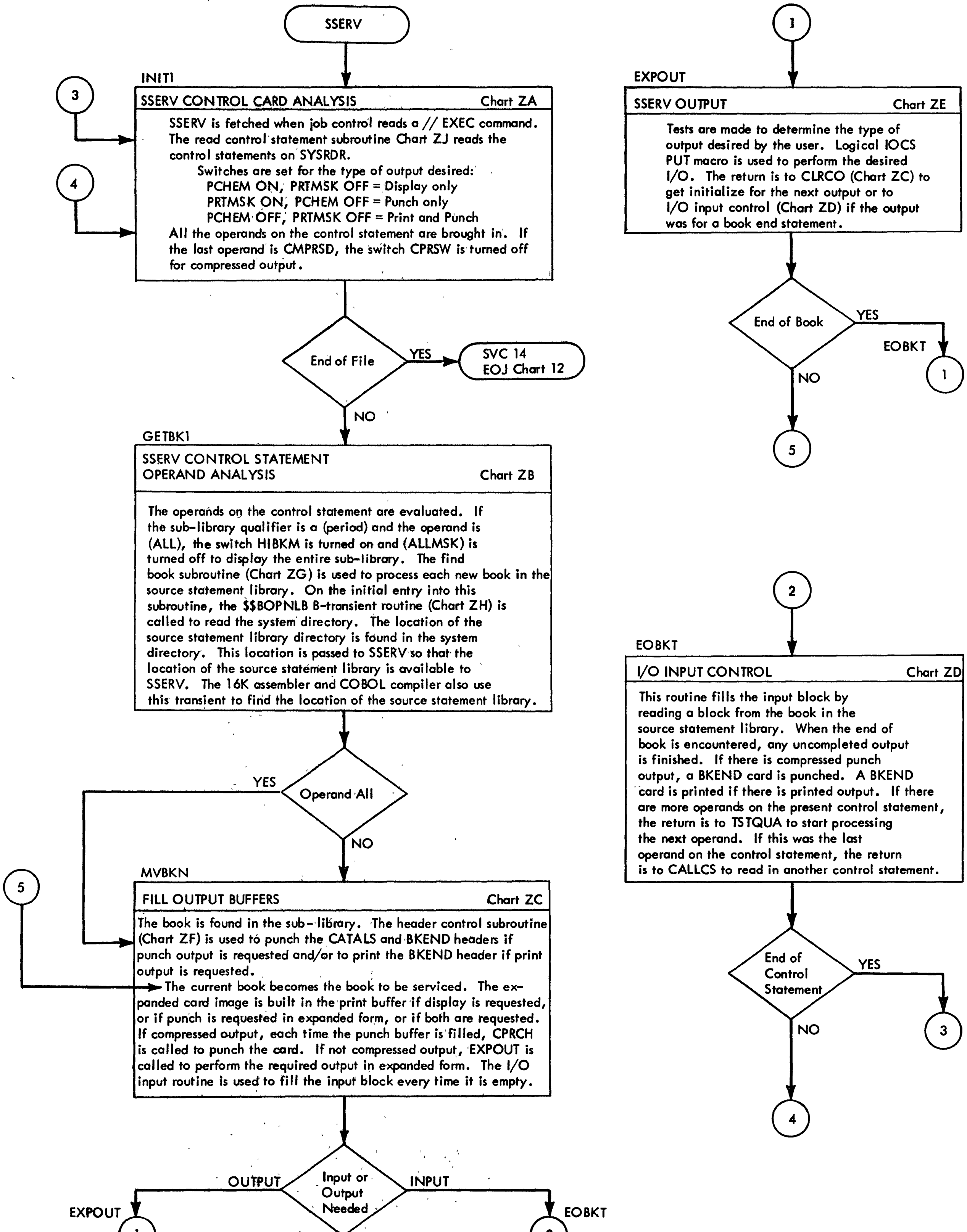



Chart 50. Source Statement Library Service Program (SSERV)

## SYSTEM CONTROL PROGRAMS (SECTION 4)

### Initial Program Load ($$A$IPL1, $$A$IPLA, and $$A$IPL2), Charts AA-AH

Label ........... Chart

ADCLOP ......... AC
BCHEST ........ AF
BEGIPL ........ AB
  Starting label for the $$A$IPL2 program.
BLDPUB ........ AF

CCW1 .......... AA
  Used to read in $$A$IPLA program.
CCW2 .......... AA
  Used to seek the $$A$IPL2 program.
CCW3 .......... AA
  Used to search for the $$A$IPL2 program.
CCW4 .......... AA
  A transfer in channel CCW used to see if the record being searched has been found.
CCW5 .......... AA
  Used to read in the $$A$IPL2 program.
CHCLOP ........ AF
  No devices on the channel yet. The FOCL for this channel must be built.
CHCNT ......... AF
CHFIN ......... AF
CHGADR ........ AB
CHSRT ......... AF
CHURTN ........ AH
CLDLP2 ........ AF
CLDRTN ........ AF
CLEAR ......... AB
  Label of the clear routine used to clear storage above the $$A$IPL2 program.
CONTIN ........ AB
  Return address after program check is forced at end of storage while clearing storage.

DECRL ......... AG
DECRR ......... AG
DSKADR ........ AH
  Subroutine to update disk addresses.
ENFND ........ AF
ENDRD ......... AC
ENTER ......... AD
  Label of the address that the program branches to after an I/O interrupt while trying to assign a communication device. A 1052 or unassigned card reader could cause this interrupt.
ERR ........... AB
  An error halt when IPL cannot continue.
EXIT1 ........ AH
EXTRTN ........ AD
  Label of the address that the program returns to if SYSRDR is assigned and the operator indicates he has add and/or delete cards in SYSRDR. The return is via an external interrupt.

FOUND ......... AB

GO ............ AD

HALT .......... AB
  Masks off interrupts and enters wait state on an error encountered during IPL.
IJBIP210 ...... AB

IPLPRG
  Listing only: Starting label for execution of the $$A$IPL1 program.

LASTRD ........ AC
LOGWRK ........ AE
LOKCID ........ AB
LOOKUP ........ AB
  Searches the directories for the supervisor entry.
LUBLOP ........ AE
LUBMVC ........ AE
LUBRTN ........ AE
LUBSET ........ AD
  Branches to subroutine to assign a device.
LUUEND ........ AH
LUUPLP ........ AH
LUURTN ........ AH

MAXKEY
  Listing only: Contains a value of 255. It is used to compare with the FOCL to see if any devices are on the channel.
MVCEND ........ AC
NOMVP ........ AF
ORGEST ........ AE
PSWGO ........ AD

PUBMKE ........ AE

RDBLOK ........ AC
RDDIR2 ........ AB
RDRIPT ........ AD
READER ........ AB
RECNO .......... AH
RESIDL ........ AG
RESIDR ........ AG
RESWRK ........ AE
RIGHT ........ AG
RTRAK ........ AB

SCHSCH ........ AF
SCHTST ........ AF
SETBLK ........ AC
  Initializes to read the supervisor into storage.
SETKEY ........ AD

Sets protection key for each 2K block of storage.
SETLOG ........ AD
SETPROT ....... AD
Initializes to set protection keys.
SHASTA ........ AF
SKADR1
Listing only: Label of the address in which the disk address for the $$A$IPL2 program is stored.
SPRSW ......... AF
SW ............ AB
This switch is initially off (NOP). It is used after the transient directory has been searched and the supervisor nucleus has been found to branch around the instructions used in searching for the supervisor nucleus.
SW1 ........... AB
NOP/BR switch that is turned on (BR) for reading the last block of the supervisor into main storage.
SW2 ........... AB
NOP/BR switch that is turned on (BR) when first block of the supervisor is read.
SYSMVC ........ AG

UPDAT ......... AB

WAIT .......... AD
The system is put into the wait state and the operator has an option of assigning the communication device.

## Initial Program Load ($IPLRT2), Charts AJ-AY

| Label | Chart |
|---|---|

ABNCHK ........ AP
ADDRTN ........ AN
Starting address of the add a device routine.
ADREST ........ AQ
Subroutine used to find the date fields in the SET control statement.
ALCERR ........ AY
ALCRTN ........ AY
ALCRT1 ........ AY
ALCRT2 ........ AY
ASNEND ........ AX
ASNLP1 ........ AX
ASNLP2 ........ AX
ASNRTN ........ AX
Start of subroutine that resets the job control flags in the PUB with a logical OR to make certain they are set correctly.
ASNSTP ........ AX

BCHEST ........ AU
BEGIN ......... AJ
Starting label for the $IPLRT2 program.
BLDPUB ........ AU
Subroutine that, given the channel and desired position, inserts a PUB entry in the PUB table.
BSTOFF ........ AR
BSTOK ......... AR

CDSCH ......... AK
CHCLOP ........ AU
CHCNT ......... AU
CHEXT ......... AS
CHFIN ......... AU
CHKCOM ........ AL
CHSLOP ........ AS
CHUPD ......... AS
CHURTN ........ AS
Subroutine used to update the FOCL pointers after a PUB has been added or deleted.
COMCHK ........ AT
Subroutine that checks a device against a list of allowable devices. Error return if no entry found.
COMDOK ........ AL
COMMA
Listing only: Label in the TRT delimiter table. It is a valid end-of-field character. When checking for a blank end-of-field character, it is changed to an invalid character.

COMNFD ........ AL
CYLDP ......... AL

DATERT ........ AQ
Starting address of the date subroutine.
DBLADD ........ AN
DBLSCN ........ AN
DDLUB ......... AS
DECLP ......... AT
DECRTN ........ AT
Subroutine that converts a field from hexadecimal to binary.
DELEXT ........ AP
DELIM
Listing only: Table used with a TRT instruction. It contains function values for finding the end of fields (OP code or operand).
DELLOP ........ AP
DELRTN ........ AP
Starting address of the delete a PUB routine.
DSDP1 ......... AV
DSDP2 ......... AV
DSDP3 ......... AV
DSDRTN ........ AV

ENADR ......... AN
ENFND ......... AU

FDSRTN ........ AR
Subroutine used to locate the next control statement operand in the input area.
FNDTYP ........ AR
Analyzes the device type specified in an

ADD statement to obtain table information.

HEXRTN ........ AT
Subroutine that converts a field from hexadecimal to binary.

ILLCD ......... AP
IOHLD ......... AW
Starting address of subroutine to issue SVC for 1052 operations.
IOHLD2 ........ AW
IOSTOP ........ AP
IPLEND ........ AM
KEYCHK ........ AN
LBLPED ........ AS
LOGRED ........ AJ
LOGSTR ........ AJ
LUBHLP ........ AL
LUBLPL ........ AS
LUDRTN ........ AS
LUURTN ........ AS
When given the PUB number of an added or deleted PUB entry, this subroutine modifies PUB pointers in the LUB table to reflect the added or deleted PUB.

MLUUR ......... AS
MPXCHK ........ AW
MPXHLT ........ AW
MPXHL1 ........ AW
MPXHL2 ........ AW
MPXLOP ........ AW
MPXMOV ........ AW
MPXRTN ........ AW
Subroutine used to reorder the LUBs and PUBs for multiplex devices.
MSGRTN ........ AW
Starting address of the print message subroutine.

NLPYR ......... AQ
NOMTEB ........ AM
NONBLK
Listing only: Table used with TRT instruction to find first character of an OP code.
NUMCVT ........ AN

OFFINT ........ AM
Starting label of the move I/O tables to low storage function.
OPNEND ........ AM
OPNLOP ........ AM
OPNRTN ........ AM
Subroutine that unassigns any system I/O units (SYSRDR, SYSIPT, SYSPCH) that have a standard (generated) assignment to a tape, disk, or data cell file.
OPNUSN ........ AM
OPRTN ......... AK
Starting label for the evaluate control statement routine.

PBFEND ........ AV
PBFFIN ........ AL
PBFLOP ........ AV
PBFRTN ........ AV
Subroutine, given an IPL low storage PUB table entry, finds an equal system high storage PUB table entry.
PUBCLC ........ AJ
PUBDEQ ........ AP
PUBEXD ........ AN
PUBMKE ........ AN

RDRTST ........ AL
READGO ........ AJ
Label of the branch and link instruction to the read routine.
READRT ........ AJ
Starting address of the read card routine.
REREAD ........ AJ
RESNFD ........ AL
RSTCHQ ........ AW
RSTLUB ........ AS

SCHLOP ........ AR
SCHSCH ........ AU
SCHSTA ........ AU
Switch used to branch around the SCHTST instruction. Initially set in NOP switch position.
SCHTST ........ AU
Tests for next FOCL entry to see if it is in use.
SCNEND ........ AP
SCNLOP ........ AP
SETFCL ........ AS
SETRTN ........ AK
Starting label of the set routine.
SKPINC ........ AQ
SKPKEY ........ AN
SPRSW ........ AU
SSKLOP ........ AY

TAPE .......... AR
TEBCLC ........ AJ
TEBDEQ ........ AP
TEBEST ........ AR
TEBEXT ........ AR
TEBLOP ........ AP
TIMCHK ........ AK
TIMERT ........ AQ
Starting address of the time subroutine.
TRTBRC ........ AV
Subroutine used to test for the delimiter or end character of a field.

UPLUB ......... AS
UNSRES ........ AL
Starting address for the assign SYSRES and SYSLOG for system operation routine.

YLPYR ......... AQ

Job Control ($JOBCTLA), Charts BA-BL

Label Chart

$JOBCTLA ...... BA
Phase name of the job control root phase. The first executable instruction in this phase is at the label JOBCTL.

ACTRSP ....... BB
ARGUMT
  Listing only: A 7-byte field in the Branch table which contains the current job control statement operation field during the control statement table lookup.
ATNCUU ....... BL
BASRG1
  Listing only: Base register for job control root phase, $JOBCTLA. See BASVCT.
BASRG2
  Listing only: Base register for the $JOBCTLD, $JOBCTLG, $JOBCTLJ phases of job control. See OVRVCT.
BASRG3
  Listing only: Initialized by $JOBCTLA root phase, to equal BASRG2 + 4096. Serves as base register for $JOBCTLD, $JOBCTLG, $JOBCTLJ.
BASRG4
  Listing only: Contains the communication region address.
BASVCT
  Listing only: $JOBCTLA origin. It is the address contained in BASRG1. This address contains a branch to the common control statement read routine (CONTROL). Any phase desiring to return to CONTROL, does so by branching on BASRG1.

BTLOOP ....... BC
BUFFER
  Listing only: 120-byte I/O area that allows control statements to be read into the main storage area previously occupied by the job control initialization routine, JOBCTL.

CHKASG ....... BK
  Root phase subroutine to check the assignment of a logical unit. If the specified unit is unassigned the condition code 1 is set and control is returned to the calling sequence. If the unit is assigned, the PUB pointer is supplied to the calling sequence in WRKRG3.
CHKASG3 ...... BK
  Equated to CHKASG+4. This is the entry into the CHKASG subroutine that is used when the LUB table address of the unit to be checked has already been loaded in WRKRG3.
CHKCNL ....... BL
CHKCNT ....... BE
  Subroutine used to check the area available for output records in the disk area allocated for SYSPCH or SYSLST. This subroutine is a part of $JOBCTLA and is overlaid when any other phase (D, G, or J) is loaded.
CHKJIB ....... BD
CHKLST ....... BD
CHKOVR ....... BJ
COMREG
  Listing only: Abbreviation used for the communication region. Text reference to a field in the communication region is written as COMREG+X. Where X represents the decimal displacement of the field in the communication region.
CONTROL ...... BB
CTRLSW ....... BD
DCUXTN ....... BD
DFB
  Listing only: Data file block (Figure 78).

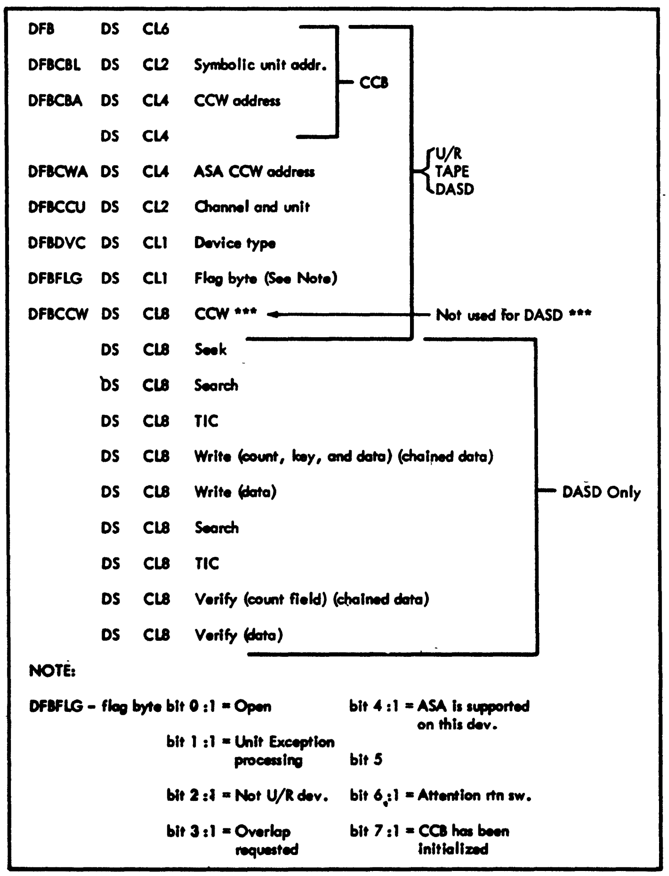


Figure 78. DFB Format

DFBCBA
  Listing only: See DFB.

DFBCBL
   Listing only:  See DFB.
DFBCCW
   Listing only:  See DFB.
DFBCUU
   Listing only:  See DFB.
DFBFLG
   Listing only:  See DFB.
DFBDVC
   Listing only:  See DFB.
DSKIND ......... BH
DSKINT ...... BD
   Subroutine to perform Job Control initialization for DASD. If SYSLST/SYSPCH are assigned to disk the available record count is checked. All extent JIBs currently attached to programmer units are unassigned and placed on the free list. This subroutine is a part of $JOBCTLA and is overlaid when any other phase (D, G, or J) is loaded.

EOFPRC ....... BJ
ERRRTN ....... BL
   Root phase subroutine that displays invalid control statements and associated error messages on SYSLOG and SYSLST. Register POINT1 contains the address of a move instruction that is executed to move the desired error message to the output area, BUFFER.
EXCECP ....... BJ

EXCPRG ....... BH
   Root phase subroutine to perform I/O on a specified logical unit. If the file has not been opened this routine initializes its DFB including the CCB before executing the I/O.
EXPEXT ....... BJ
FINOPN ....... BH
IGNORE ....... BG
IJSYSLN
   Listing only:  File name specified in the SYSLNKDTF.
ISSUIO ....... BJ
JBCSW0
   Listing only:  Displacement 56 in the communication region (COMREG + 56). See Figure 15.
JBCSW1
   Listing only:  Displacement 57 in the communication region (COMREG + 57). See Figure 15.
JBCSW2
   Listing only:  Displacement 58 in the communication region (COMREG + 58). See Figure 15.
JBCSW3
   Listing only:  Displacement 59 in the communication region (COMREG + 59). See Figure 15.
JBCSW4
   Listing only:
   Bit 0:  0 = Job control statement started with a //
           1 = Job control statement started without a //
   Bit 1:  0 = Statement is to be logged on SYSLST in case of error.
           1 = Statement is not to be logged on SYSLST in case of an error.
   Bit 2:  0 = Statement is to be logged on SYSLOG in case of error.
           1 = Statement is not to be logged on SYSLOG in case of error.
   Bit 3:  0 = A label block is in the output area ready to be written on the SYSRES label cylinder.
           1 = No label block present.
   *Bit 4: 1 = Only a VOL statement may follow.
   *Bit 5: 1 = Only an EXTENT or VOL statement may follow.
   Bit 6: 1 = Only an EXTENT statement may follow.
   Bit 7: 1 = Only a DLAB or TPLAB statement may follow.
   *Bits 4, 5 are also used by the RSTRT routine to insure that label statements have been processed.

JOBCTL ....... BA
   Label of first executable instruction of Job Control. This initialization routine is overlaid by control statements read into the I/O area labeled BUFFER.

LINCNT
   Listing only:  Maximum line count for SYSLST. Maintained in COMREG + 78.
LNKINT ....... BH
LOGCHK ....... BG
   Root phase subroutine used to set switches in JBCSW0 (COMREG + 56) bits 6 and 7 to indicate SYSLOG device type and assignment.
   Bit 6: 0 = 1052
          1 = line printer
   Bit 7: 0 = SYSLOG ≠ SYSLST
          1 = SYSLOG = SYSLST
LOGIN ........ BG
LOGOUT ....... BE
   Root phase subroutine used to output a control statement or a message on SYSLOG.
LOGPRT ....... BG
LSTOUT ....... BE
   Root phase subroutine used to output a control statement or a message on SYSLST.

MSGOUT ....... BE
   Root phase entry into the LOGOUT subroutine. Sets switches to allow output on both SYSLST and SYSLOG.
MTNCNT ....... BK
   Root phase subroutine to set the system mask to X'FF' (allow all interrupts) and to seize or release the system (SVC 22).

NDSCAN ....... BF
NODSYS ....... BD
NOEERR ....... BL
NOEERT ....... BL
NOTDKS ....... BJ
NTINJB ....... BL
NVSERR ....... BL
NXTJIB ....... BD
NXTLUB ....... BD

OERRTN ....... BL
Root phase subroutine that displays a specified message on SYSLOG. Register POINT1 contains the address of a move instruction that is executed to move the desired message to the output area, BUFFER.

OPNUMH
<u>Listing only</u>: A 1-byte field which is the data byte in a LOAD instruction. This byte is used to maintain a parameter count by the control statement processing routines. It is reset to zero following each successful control statement read operation.

OVRLP1 ....... BJ

OVRVCT
<u>Listing only</u>: Beginning address of the overlay area where $JOBCTLD, $JOBCTLG, and $JOBCTLJ are loaded. This address is within the $JOBCTLA root phase. $JOBCTLA initializes the base register, BASRG2, with this address each time it is loaded.

RDRIN ........ BG

RDSTMT ....... BG
Root phase subroutine used to read a statement from SYSRDR or SYSLOG. The job control switch, JBCSW0 (COMREG + 56) bit 0, is tested to determine if SYSRDR or SYSLOG is to be used. If SYSLOG is specified but it is not a 1052, the switch is changed to indicate SYSRDR and the subroutine is reentered via the control routine.

RLINDT ....... BG

RLNCNT
<u>Listing only</u>: Save area for SYSLST current line count.

RNAERR ....... BL

SCANR1 ....... BF
Root phase subroutine used to scan a control statement and make a parameter available for processing.

- A comma, blank, or equal sign ends the scan.
- Register POINT1 contains the address of the 1st character of the parameter.
- Register POINT2 contains the number of characters remaining to be scanned.
- Register POINT3 contains a count of the number of characters in the parameter. This character count is 1 less than the actual character count and is used to control the character count in move and compare instructions.
- Between entries into the subroutine, registers POINT1 and POINT2 are saved in an area labeled TMPAR1.
- The entry SCANR1 is used when it is desired to scan for the operation field.
- The entry SCANR2 scans for prime operands.
- The entry SCANR3 is used to scan for the parameter of a prime operand. Example:

```
SET RCLST = 2000, LINCNT = 99,...

 1   2       3      2       3
```

1 indicates the operation field
2 indicates the prime operands
3 indicates the parameters of the prime operands.

SCANR2 ....... BF
Entry into the SCANR1 subroutine that is used to make a prime operand available for processing. See SCANR1.

SCANR3 ....... BF
Entry into the SCANR1 subroutine that is used to make a parameter available for processing. See SCANR1.

SCANRL1 ...... BF
SCANRL2 ...... BF
SETWRT ....... BJ
SPCEXC ....... BJ
SUPLOG ....... BC

TAPINT ....... BH

TBLADR
<u>Listing only</u>: Label of the Phase-Vector Table contained in the root phase ($JOBCTLA). This table 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 one byte condition switch bank used to control processing for format verification, logging conventions, and cancel procedures for the statement. Figure 79 shows the format of an entry in the Phase-Vector Table.

| Byte | 0 6 | 7 | 8 | 9 |
|---|---|---|---|---|
| | Operation Field | Condition Switches | Branch Vector Displacement | Phase Identification Letter |

Figure 79. Phase-Vector Table Entry Format

Operation Field: EBCDIC representation of the operation field.

Condition Switches:

Bit 0 - reserved.
1 - statement is to be processed even though a cancel condition exists.
2, 3 - Both on; suppress logging.
2 off, 3 on; unconditional SYSLOG logging and conditional SYSLST logging.
Both off; conditional logging on SYSLOG and SYSLST.
4 - statement may start with //.
5 - statement may start without //.
6 - statement may start in column 1.
7 - statement may start in other than column 1.

Branch Vector Displacement: Displacement within the phase that is added to the phase origin address to develop the address of a branch instruction which transfers control to the correct processing routine.

Phase Identification Letter: Contains the EBCDIC character A, D, G, or J and identifies the job control phase containing the processing routine.

Example of the JOB control statement entry:

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

1. The JOB statement is to be processed even if a cancel is being executed.
2. Logging on both SYSLOG and SYSLST is suppressed.
3. The statement may not start without // and may not start in other than column one.
4. The branch-vector table entry is located at a displacement of 12 bytes from the beginning of the phase with suffix 'G' ($JOBCTLG).

TMPAR1 ....... BA
Initially a part of the initialization routine. It is overlaid after initialization is complete and becomes a save area in which registers POINT1 and POINT2 are saved between control statement scan operations.

TSTYPE ....... BC

UNCLOG ....... BC

ZRMVDN ....... BE
ZRMVLP ....... BE

Job Control ($JOBCTLD), Charts BM-CY

ALTASW
Listing only:
Bank 1, bit 1: if on, specifies ALT assignment.
- Set ON: ASSGN statement processor (Chart CB)
- Set OFF: INITL subroutine (Chart CR)
- Controls processing in the ASSGN processor (Chart CB)

ASSGN ........ BY
Initial entry into the ASSGN statement processor, scan and check 1st and 2nd operands.

ASSGNB3 ...... CA
Initial entry into the verify and store routine for UA or IGN assignments.

ASSGNB4 ...... CB
Entry into the scan routine, from the cross-assignment verification routine, to process the optional operands X'SS', ALT, or TEMP.

ASSGNB5 ...... CC
Perform verification of a normal assignment (temporary or standard).

ASSGNB6 ...... CD
Make a normal standard assignment.

ASSGNB6A ..... CC
ASSGNB6B ..... CC
ASSGNB6C ..... CC
ASSGNB6D ..... CC
ASSGNB7 ...... CG
Complete assignment and open files if assignment is for SYSRDR, SYSIPT, SYSPCH, or SYSLST.

ASSGNB8 ...... CF
Make alternate assignment.

ASSGNNT ...... BY
ASSGNTS ...... BY
ASSGN0 ....... BY
ASSGN10 ...... BZ
ASSGN101 ..... BZ
ASSGN11 ...... BZ
ASSGN12 ...... BZ
ASSGN13 ...... CA
ASSGN14 ...... CB
ASSGN15 ...... CB
ASSGN16 ...... CB
ASSGN17 ...... CB
ASSGN18 ...... CB
ASSGN19 ...... CB
ASSGN19A ..... CB
ASSGN20 ...... CB
ASSGN21 ...... CC
ASSGN22 ...... CE
Make normal temporary assignment.

ASSGN23 ...... CD

ASSGN23A ..... CD
ASSGN23B ..... CD
ASSGN23C ..... CD
ASSGN23D ..... CD
ASSGN24 ...... CD
ASSGN25 ...... CD
ASSGN26 ...... CE
ASSGN27 ...... CE
ASSGN28 ...... CE
ASSGN29 ...... CG
ASSGN3 ....... CY
ASSGN30 ...... CG
ASSGN31 ...... CG
ASSGN32 ...... CG
ASSGN33 ...... CG
ASSGN34 ...... CF
ASSGN35 ...... CF
ASSGN36 ...... CF
ASSGN37 ...... CF
ASSGN4 ....... BY
ASSGN40 ...... CG
ASSGN402 ..... CB
Entry into the scan routine, from the UA or IGN assignment routine, to process the optional operand, TEMP.
ASSGN403 ..... CB
ASSGN404 ..... CB
ASSGN41 ...... CH
ASSGN42 ...... CH
ASSGN43 ...... CY
ASSGN5 ....... BY
ASSGN6 ....... BZ
Initial entry into the cross-assignment verification routine.
ASSGN7 ....... BZ
ASSGN8 ....... BZ
ASSGN9 ....... BZ
ASSGN901 ..... CF

BANK1
<u>Listing only</u>: 2-byte switch bank.
Reset to zeros in the INITL subroutine.
Byte 0, bit 0 see CLOSESW
1 see ALTASW
2 see STDFDSW
3 see RETSW
4 see EOLSW
5 see PROGSW
6 see LIOSW
7 see RESETSW

Byte 1, bit 0 see DDSW
1 see TRANSW
2 not used
3 not used
4 not used
5 see UNSW
6 see TMPSW
7 see MODSW

CHKDIB ....... CC
CHKOPN ....... CN
Subroutine: Sets the open indicator off in the DFB if the CUU of the DFB does not equal the CUU of the assignment.
CHKOPN1 ...... CN
CHKOPN2 ...... CN
CHKRNG ....... CR
Subroutine: Checks the range of each character in a parameter. Upon entry:

- Register POINT1 contains the address of the first byte of the parameter.

- Register POINT3 contains the number of characters to be checked minus 1.

The compare immediate instruction, RNGTOP, is modified by the calling sequence to compare for the maximum character (numeric = 9, hex = F).
CLOSE ........ BN
Initial entry point into the CLOSE statement processor.
CLOSED ....... BN
CLOSESW
<u>Listing only</u>:
BANK1, bit 0: If on specifies that a close is in process.
- Set ON: CLOSE statement processor (Chart CR)
- Set OFF: INITL subroutine (Chart CR) and CLOSE1 subroutine (Chart CM)
- Controls processing in:
  1. CLOSE1 subroutine (Chart CM)
  2. ASSGN statement processor (Charts CC, CE)

CLOSE1 ....... CM
Entry point to the subroutine that closes tape files.
CLOSE10 ...... CL
CLOSE11 ...... CL
CLOSE12 ...... CL
CLOSE2 ....... BN
CLOSE3 ....... BN
CLOSE7 ....... BN
CLOSE8 ....... CL
Entry to subroutine to close SYSIN/SYSOUT files (SYSRDR, IPT, PCH, or LST on SYSRES). The DIB is updated and a file mark is written if required.
CLOSE9 ....... CL
CNIOAG ....... CY

DDSW
<u>Listing only</u>:
BANK1+1, bit 0: If on, specifies DVCDN in progress.
- Set ON: DVCDW statement processor (Chart BU)
- Set OFF: INITL subroutine (Chart CR)
- Controls exit from the RESET statement processor (Chart CK).

DOWN ......... BT
Routine within the LISTIO statement processor used to process the operand DOWN.
DVCDN ........ BU
Initial entry into the DVCDN statement processor.
DVCDNL ....... BU
DVCDNS ....... BU
DVCDN1 ....... CP
DVCDN10 ...... BW
DVCDN11 ...... BW
DVCDN12 ...... BV

DVCDN13 ...... BV
DVCDN14 ...... BU
DVCDN15 ...... BW
DVCDN2 ....... BU
DVCDN3 ....... CP
Subroutine: Computes the number of classes in the LUB table. Batch only equals 2, MPS equals 4. WRKRG1 contains the value 2 or 4 when control is returned to the calling sequence.
DVCDN4 ....... BU
DVCDN5 ....... BU
DVCDN7 ....... BV
DVCDN8 ....... BW
DVCDN9 ....... BV
DVCUP ........ BX
Initial entry into the DVCUP statement processor.

EOLSW
Listing only:
BANK1, bit 4: If on, specifies that the end of a class in the LUB table has been reached.
- Set ON: SCANLUB subroutine (Chart CN)
- Set OFF: INITL subroutine (Chart CR) and SCANLUB subroutine (Chart CN)
- Controls processing in:
  1. ASSGN statement processor (Chart BZ)
  2. DVCDN statement processor (Chart BU)
  3. LISTIO statement processor (Chart BR, BQ)

ERRRTN ....... CY
Entry into the common error routine. Releases the B-transient area if the switch, TRANSW is on ($$BLISTIO has been loaded). Control is transferred to the common error routine ERRRTN (Chart BL) in the root phase.
ERRRTN1 ...... CY
ERRRTN2 ...... CY
EXCP ......... CT
Subroutine: performs I/O
- CCW address is supplied in register 0.
- Symbolic unit address (class and order) is supplied in register 1.

EXCPROG ...... CT
Subroutine: Used to perform I/O on tape when user tape density is to be used.
- The symbolic unit address (class and order) is supplied in register POINT1.
- The PUB address is supplied in register POINT2.

EXCPROG1 ..... CT
Entry point to the EXCPROG subroutine when IBM standard tape density is to be used.
- The symbolic unit address (class and order) is supplied in register POINT1.
- The PUB address is supplied in register POINT2.

EXCPROG2 ..... CT
EXCPROG3 ..... CT
EXCPROG4 ..... CT

FLOC
Listing only: Label of a 2-byte location used to hold the address of the first LUB of a class. Loaded by the GETLAN subroutine.
FNIOAG ....... CY

GETJIB ....... CU
Subroutine: Attaches a JIB to a LUB.
GETJIB1 ...... CU
GETJIB2 ...... CU
GETLAN ....... CR
Subroutine: Determines the number of logical units in a class and the address of the first LUB of a class. The number of units in a class is returned in WRKRG3 and the locations NOC and SNICL. The address of the first LUB of a class is returned in WRKRG2 and the locations FLOC and SLADD.
GETLAN2 ...... CR

HEXCON ....... CM
Subroutine: Converts EBCDIC 'CUU' to packed binary and makes result available in WRKRG3.

ILUS ......... CY
INDVTP ....... CY
INITL ........ CR
Subroutine: Performs initialization for all statement processors in $JOBCTLD phase.
IVDS ......... CY

JIBCHN
Listing only: Label of a 2-byte location used as a 2-byte work area for LUBs and JIBs.

LIOCUU ....... BS
Routine within the LISTIO statement processor used to process the operand CUU.
LIOEOJ ....... BP
LIOEOJ1 ...... BP
LIOL ......... BQ
Routine within the LISTIO statement processor used to process the operands, SYS, PROG, F1, F2, or ALL.
LIOL2 ........ BQ
LIOLL201 ..... BQ
LIOLL3 ....... BQ
LIOL1 ........ BQ
LIOL202 ...... BQ
LIOL4 ........ BQ
LIOSW
Listing only: BANK1, bit 6: If ON specifies LIOTIO in progress.
- Set ON: LISTIO statement processor (Charts BQ, BS)
- Set OFF: UNPA subroutines (Chart CP)
  SFPPE subroutine (Chart CV)
  INITL subroutine (Chart CR)
- Controls processing in the SFPPE

subroutine (Charts CV, CW)
LIOSYX ....... BS
Routine within the LISTIO statement processor used to process the operand SYSXXX.
LISTIO ....... BP
Initial entry into the LISTIO statement processor.
LUBADD
Listing only: Label of a 2-byte location used to hold the address of the LUB being assigned. Loaded by the subroutine, SYSXXX.

LUBCOM
Listing only: Label of a 2-byte location used for temporary storage of a LUB for comparison. Used by the SCNLUB subroutine.

MODSW
Listing only:
BANK1+1, bit 7: If on, specifies that the mode must be set for this assignment.
- Set ON: ASSGN statement processor (Chart CB)
- Set OFF: INITL subroutine (Chart CR)
- Controls processing in the ASSGN statement processor (Charts CD, CE)

NEWLUB
Listing only: Label of a 2-byte work area in which a LUB is built for a new assignment. The subroutine TXCUU computes a PUB pointer and stores it in byte 0.
NEWPUB
Listing only: Label of a 2-byte location used to hold the address of the PUB to be assigned to the LUB that is being assigned. Loaded by the subroutine TXCUU.
NOC
Listing only: Label of a 2-byte location used to hold the number of logical units of a class. Loaded by the GETLAN subroutine.
NOEXC ........ CL
NOMRJB ....... CY
NUMCON ....... CR
Subroutine: Converts to EBCDIC numbers (0-9) to binary in WRKRG3.

OLDPUB
Listing only: Label of a 2-byte location used to hold the address of the PUB currently assigned to the LUB that is being reassigned. Loaded by the subroutine SYSXXX.
OUTPUT ....... CQ
Subroutine: Displays on SYSLST and/or SYSLOG.
OUTPUTS ...... CQ
Subroutine: Displays a line and skips a line on SYSLST and/or SYSLOG. Clears I/O buffer and work areas to blanks.
OUTPUT1 ...... CQ
Subroutine: Displays a line on SYSLST and/or SYSLOG. Clears I/O buffer and work areas to blanks.
OUTPUT2 ...... CQ

PROGSW
Listing only:
BANK1, bit 5: If ON, specifies that the programmer LUBs are to be scanned.
- Set ON: LISTIO statement processor (Chart BS), ASSGN statement processor (Chart BZ), RESET statement processor (Chart CJ)
- Set OFF: INITL subroutine (Chart CR) ASSGN statement processor (Chart BZ)
- Controls processing in:
1. LISTIO statement processor (Chart BQ)
2. ASSGN statement processor (Chart BZ)

PUBMSK ....... CV

RESET ........ CJ
Initial entry into the RESET statement processor.
RESETSW
Listing only:
BANK1, bit 7: If ON, specifies RESET in progress.
- Set ON: RESET statement processor (Chart CJ)
- Set OFF: INITL subroutine (Chart CR)
- Controls exit from the RSTSTD subroutine (Chart CU)

RESET01 ...... CJ
RESET1 ....... CJ
RESET11 ...... CK
Routine in the RESET statement processor used to reset all LUBs to standard. This routine is also entered as a subroutine from the DVCDN statement processor.
RESET2 ....... CK
RESET3 ....... CK
RESET4 ....... CK
RESET5 ....... CK
RESET8 ....... CK
RETADD ....... BP
RETSW
Listing only:
BANK1, bit 3:
- Set ON: SFPPE subroutine (Chart CV) ASSGN statement processor (Chart CE)
- Set OFF: ASSGN statement processor (Charts CB, CE, CF)

DVCDN statement processor (Chart BU)
INITL subroutine (Chart CR)
- Controls processing in the SFPPE subroutine (Charts CV, CW)
- Controls processing in the GETJIB subroutine (Chart CU)

RSTSTD ....... CU
Subroutine: Restores a LUB to its standard I/O assignment or unassigns a non-standard assignment.
RSTSTD1 ...... CU
RSTSTD3 ...... CU
RSTSTD4 ...... CU

SCNJIB ....... CN
Subroutine: Computes address of a JIB in register POINT3. The stored LUB of the JIB is moved to the location JIBCHN.

SCNLUB ....... CN
Subroutine: Makes all LUBs of a class available one at a time. The current LUB is made available in the location JIBCHN. The address of the next LUB is saved in the location SLADD. The residual number of LUBs (in a class) is saved in the location SNICL.

SFNC ......... CY

SFPPE ........ CV
Subroutine: Scans the LUBs for equal PUB pointers. LINKR4 is used to re-enter the subroutine from the calling sequence.

SFPPE01 ...... CV
SFPPE02 ...... CV
SFPPE03 ...... CP
SFPPE1 ....... CV
SFPPE10 ...... CW
SFPPE12 ...... CW
SFPPE13 ...... CX
SFPPE14 ...... CW
SFPPE15 ...... CX
SFPPE16 ...... CX
SFPPE17 ...... CX
SFPPE18 ...... CX
SFPPE19 ...... CW
SFPPE2 ....... CV
SFPPE201 ..... CW
SFPPE3 ....... CP
SFPPE4 ....... CV
SFPPE5 ....... CV
SFPPE6 ....... CV
SFPPE7 ....... CV
SFPPE8 ....... CW
SFPPE8A ...... CW
SFPPE9 ....... CW

SKIPLN ....... CQ
Subroutine: Prints a blank line on SYSLST and/or SYSLOG to simulate a line skip.

SLADD
Listing only: Label of a 2-byte location used to hold the address of the first LUB of a class. Loaded by the GETLAN subroutine.

SNICL
Listing only: Label of a 2-byte location used to hold the number of logical units of a class. Loaded by the GETLAN subroutine.

STDFDSW
Listing only:
BANK1, bit 2: If ON, specifies that a stored standard assignment has been found.
- Set ON: SFPPE subroutine (Chart CW)
- Set OFF: SFPPE subroutine (Chart CW)
  INITL subroutine (Chart CR)
- Controls processing in the SFPPE subroutine (Chart CW)

SVCBTRNS ..... CT
Subroutine: Fetches the B-transient $$BLSTIO. The register LINKR2 is used to return control to the calling sequence from the B-transient.

SYSXXX ....... CS
Subroutine: Converts a logical unit, designated as SYSXXX, into:
1. Symbolic unit address (class and order) in the location CLOARD.
2. LUB address in the location LUBAD.
3. PUB pointer of the PUB currently assigned to this logical unit in the location OLDPUB.
4. The internal representation of device type in location NEWTYP.

SYSXXX01 ..... CS
SYSXXX1 ...... CS
SYSXXX2 ...... CS
SYSXXX3 ...... CS
SYSXXX4 ...... CS
SYSXXX5 ...... CS
SYSXXX6 ...... CS

TEST1 ........ BT
TEST2 ........ BT
TIAERR ....... CY

TMPSW
Listing only:
BANK1+1, bit 6: If ON, specifies that the assignment is temporary.
- Set ON: ASSGN statement processor (Chart CB)
- Set OFF: INITL subroutine (Chart CR)
- Controls processing in the ASSGN statement processor (Charts CC, CD, CF)

TNVSERR ...... BP

TRANSW
Listing only:
BANK1+1, bit 1: If ON, specifies that the Job Control transient, $$BLISTIO, has been loaded in the B-transient area.
- Set ON: SVCBTRNS subroutine (Chart CT)
- Set OFF: LISTIO statement processor (Chart BP)
  INITL subroutine (Chart CR)
- Controls processing in:
  1. SFPPE subroutine (Chart CV, CW)
  2. ERRRTN common error routine (Chart CY)

TXCUU ........ CM
Subroutine:
1. Converts X'CUU' to binary
2. Searches PUB table for equal CUU
3. Saves device type from PUB in location DEVTYPE
4. Saves PUB address in location NEWPUB
5. Computes PUB pointer and saves it in location NEWLUB (byte 0).

TXCUU2 ....... CM
TXCUU3 ....... CM
Entry point into the TXCUU subroutine that is used to compute PUB pointers only. The PUB pointer is returned in location NEWLUB (byte 0).

UA ........... BT
Routine within the LISTIO statement processor used to process the operand UA.
UADN1 ........ BT
UADN2 ........ BT
UADN3 ........ BT
UADN4 ........ BT
UADN5 ........ BT
UADN6 ........ BT
UNA .......... BM
Initial entry point into the UNA statement processor.
UNAENT ....... CP
Entry into the SFPPE subroutine to unassign Foreground program LUBs.
1. LUB address is contained in LUBAD & WRKRG3
2. The LUB has been saved in LUBCOM
3. Number of LUBs in class is contained in WRKRG1
This entry is used by the UNA statement processor.
UNA1 ......... BM
UNA2 ......... BM
UNAE ......... BM
UNCU ......... BR
UNITS ........ BR
Routine within the LISTIO statement processor used to process the operand UNITS.
UNITS2 ....... BR
UNITS3 ....... BR
UNITS401 ..... BR
UNITS402 ..... BR
UNITS5 ....... BR
UNITS501 ..... BR
UNITS6 ....... BR
UNITS7 ....... BR
UNITS8 ....... BR
UNITS9 ....... BR
UNPA ......... CP
Entry into the SFPPE subroutine to unassign a standard assignment. This entry is used by the ASSGN routine and the RSTSTD subroutine.
UNPA1 ........ CP
Entry into the SFPPE subroutine to unassign a specific LUB. This entry is used by the DVCDN routine.
UNPA3 ........ CP
UNSW
Listing only:
BANK1+1, bit 5: If ON, specifies a single unit is to be listed.
- Set ON: LISTIO statement processor when performing a LIOCUU function (Chart BS)
- Set OFF: INITL subroutine (Chart CR)
- Controls processing in the LISTIO statement processor (Chart BR)

## Job Control ($JOBCTLG), Charts DA-DY

ALLOC ........ DM
Entry point in the ALLOC statement processor.
BTLOOP ....... DJ
Routine within the OPTION statement processor used to transfer control to the correct processor for each operand of the OPTION statement.
BLNKLD ....... DX

CANCEL ....... DA
Entry point in the CANCEL statement processor.
CATAL ........ DL
Processor for the CATAL operand of the OPTION statement.
CHGSTT ....... DX
Subroutine: Changes system status from problem program state to supervisor state and from supervisor state to problem program state. Used by the ALLOC statement processor.
CHKCND ....... DE
CHKLNK ....... DS
Subroutine: Checks SYSLNK assignment and device type.
CHKPRN ....... DM
CHKRNG ....... DN
CIDEND ....... DF
CKNDAR ....... DP
CKSCST ....... DM
CLRDOA ....... DF
Portion of the EXEC statement processor used as a subroutine to clear the output area to blanks. Register POINT1 is reset to the beginning address of the area.
CLRRTN ....... DG
CMNWLM ....... DN
CNVBCD ....... DX
Subroutine: Converts data to EBCDIC for output.
COPYSW ....... DE
CRJBSQ ....... DR
CRTBLD ....... DM
C48 .......... DK
Processor for the 48C operand of the OPTION statement.
C60 .......... DK
Processor for the 60C operand of the OPTION statement.

DECK ......... DK
Processor for the DECK operand of the OPTION statement.
DUMP ......... DL
Processor for the DUMP operand of the OPTION statement.

EDTEST ....... DD
EOJ .......... DB
Entry point in the EOJ statement processor.
EOJOFF ....... DR
ERRS ......... DK
Processor for the ERRS operand of the OPTION statement.
EXCEDT ....... DD
EXCUSR ....... DD
EXEC ......... DD

Entry point in the EXEC statement processor.

FETCHR ....... DG
FINIS2 ....... DG

GETIME ....... DV
Subroutine: Computes elapsed time and converts it to EBCDIC.
GETPUB ....... DT
Subroutine: Computes the address of a given PUB entry and stores it in register POINT3.
GOCAT ........ DL
GTMXHN ....... DS
Subroutine: Computes the upper head limit for the background program area of the VOL cylinder. Value equals 9 if no MPS. Value equals 3 if MPS.
GTNAME ....... DE
GTNXNT ....... DE
GTNXOP ....... DM

INNXEN ....... DM
IOROUT ....... DS
Subroutine: Performs I/O and waits for traffic bit to be posted.

JBINPR ....... DB
JIBPTR ....... DU
JIBSCN ....... DT
Subroutine: Scans the JIB table and makes the next JIB in a chain available for processing.
JOB .......... DQ
Entry point in the JOB statement processor.

LAXERR ....... DY
LBLOUT ....... DW
Subroutine: Writes label information in the VOL area of SYSRES from the job control label area.
LINK ......... DL
Processor for the LINK operand of the OPTION statement.
LIST ......... DK
Processor for the LIST operand of the OPTION statement.
LISTX ........ DK
Processor for the LISTX operand of the OPTION statement.
LNAERR ....... DY
LNKNOP ....... DE
LOG .......... DH
Entry point in the LOG statement processor.
LUBSCN ....... DT
Subroutine: Makes the address of the next LUB entry of a class available in register POINT1.

MAP .......... DP
Entry point in the MAP statement processor.
MOVLOP ....... DG
MVKIND ....... DX
MVMVRT ....... DF

NDTERR ....... DY
NLISTX ....... DK
Processor for the NOLISTX operand of the OPTION statement.
NOCATL ....... DB
NODECK ....... DK
Processor for the NODECK operand of the OPTION statement.
NODUMP ....... DL
Processor for the NODUMP operand of the OPTION statement.
NOERRS ....... DK
Processor for the NOERRS operand of the OPTION statement.
NOIPT ........ DB
NOLINK ....... DL
Processor for the NOLINK operand of the OPTION statement.
NOLIST ....... DK
Processor for the NOLIST operand of the OPTION statement.
NOLOG ........ DH
Entry point in the NOLOG statement processor.
NOMPSS ....... DE
NOPHDB ....... DF
NOSYM ........ DK
Processor for the NOSYM operand of the OPTION statement.
NOTNOS ....... DJ
NOXREF ....... DK
Processor for the NOXREF operand of the OPTION statement.
NVAERR ....... DY
NXPBNT ....... DP

ONOLOG ....... DH
OPAUSE ....... DH
OPLBNF ....... DW
OPLOG ........ DH
OPNLNK ....... DS
Subroutine: Opens SYSLNK DFB.
OPTION ....... DJ
Entry point to the OPTION statement processor.
OPTLOG ....... DL
Processor for the LOG operand of the OPTION statement.
OPTNLG ....... DK
Processor for the NOLOG operand of the OPTION statement.
OTSERR ....... DY

PAUSE ........ DH
Entry point in the PAUSE statement processor.
PNPERR ....... DY

RANXJB ....... DT
RASCAN ....... DT
Subroutine: Scans BG program LUBs and sets BG assignment flags in assigned PUBs.
RASSGN ....... DT
RDCID ........ DE
RESFCH ....... DD
RSPASG ....... DV
Subroutine: Entry into the RSTASG

subroutine used by the JOB statement processor to restore BG programmer class assignments.
RSPPEA ....... DP
RSRMCP ....... DW
RSSASG ....... DR
RSTASG ....... DV
Subroutine: Restores System and BG programmer assignments to standard.
RSTCOM ....... DR
RSTLAD ....... DS
Subroutine: Used by the OPTION statement processor to restore VOL area track capacity to 3625 and disk address record numbers to 0.
RSTSW4 ....... DQ

SCNINT ....... DU
Subroutine: Performs initialization for LUB table scan. Used by the RSTASG and RSPASG subroutines.
SETLOD ....... DD
SIMEND ....... DQ
SIMRET ....... DR
STDLBL ....... DL
Processor for the STDLABEL operand of the OPTION statement.
STLLMT ....... DN
STLNKA ....... DE
STOP ......... DA
Entry point in the STOP statement processor.
STSRWR ....... DF
STUCRL ....... DX
Subroutine: Writes a line from the I/O area labeled BUFFER and initializes for the next line.
STUFIU ....... DX
Subroutine: Converts the upper limit to EBCDIC and stores it in the correct field of a print line for the MAP processor.
STUSPC ....... DX
Subroutine: Builds that portion of a MAP print line indicating program prefix and upper limit.
SYM .......... DK
Processor for the SYM operand of the OPTION statement.

TBNHDR ....... DC
TBNXPB ....... DC
TBPBLP ....... DC
TEBHDR ....... DC
TEBLOP ....... DC
TIMOUT ....... DV
Subroutine: Used by the JOB and EOJ statement processors to log the time of day.

UANXJB ....... DU
UASCAN ....... DU
Subroutine: Restores BG program LUBs to standard assignments. Detaches BG program JIBs and places them in free list. Resets BG program ownership flags in PUBs.
UNASGN ....... DU

USRLBL ....... DL
Processor for the USRLABEL operand of the OPTION statement.

WRTPHD ....... DF
Subroutine: Used by the EXEC statement processor to write the phase directory.

XREF ......... DK
Processor for the XREF operand of the OPTION statement.

Job Control ($JOBCTLJ) Charts EA-ET

ACTION ....... ED
Entry to the ACTION statement processor.

BINCON ....... ES
Subroutine: Converts DATE and CLOCK parameters of the SET statement to binary in registers POINT1, POINT2, and POINT3.

BTOFRT ....... EM
BTONRT ....... EM
CHKNXC ...... EUA
CHKPGU ....... EQ
CHKPUN ....... EJ
CHKRNG ...... EUA
CHKRNG1 ..... EUA
CHKTIM ....... EL
CKSTDM ....... EG
CNUNCO ....... EQ
Subroutine: Converts the operand SYSXXX to system and unit class in the location UNCLOR.
- UNCLOR, byte 0 = System class.
  0 = Syst, 1 = programmer
  byte 1 = Unit class = LUB pointer
CONCAT ....... EP
Subroutine: Combines the operand fields of two control statements, the first control statement contains a continuation punch.
COPYLP ....... ED
Routine in the INCLUDE statement processor used to copy SYSIPT to SYSLNK.

DATE ......... EK
Entry into the DATE statement processor.
DIBRC ........ EL
DLAB ......... EG
Entry into the DLAB statement processor.
DNEERR ....... ET

DOP34 ........ ES
Subroutine: Converts creation date and expiration date to binary and stores in label output area.
DSEQMSK ...... EG
ENDINC ....... ED

FDKTDAT ..... EUB
FDKTDAT1 .... EUB
FDKTDAT2 .... EUB
FDKTID ...... EUB

FDKTV ....... EUA
FDKTVNM ..... EUA
FDKTV2 ...... EUA
FDSYSU ....... EQ

FETINSRT ..... EK
FETINSR0 ..... EK
FETINSR1 ..... EK
FSCAN ....... EUA
FSCAN1 ...... EUA
FTEND ........ EK

GTMXHN ....... ER
Subroutine: Sets upper head limit for BG program portion of the SYSRES volume area.
- HH=3 if MPS
- HH=9 if no MPS

HEXCON ....... EQ
Subroutine: Converts the operand X'CUU' to binary and saves it in the location IORGSA.
HOLD ......... EA
Entry point into the HOLD statement processor.
HOLD1 ........ EA
HOLD2 ........ EA

INAERR ....... ET
INCLUDE ...... ED
Entry to the INCLUDE statement processor.
INDSEQ ....... EH
ISCKSQ ....... EH
ISTYP4 ....... EH

LAXERR ....... ET
LBLOUT ....... EP
Subroutine: Writes label and extent information in the SYSRES volume area.
LBLTYP ....... EF
Entry into the LBLTYP statement processor.
LBTOUT ....... EF
LNKOUT ....... ER
Subroutine: Controls block count and byte count for writing on SYSLNK.
LOADRS ....... EN

MTC .......... EE
Entry point to the MTC statement processor.
MTC1 ......... EE
MTC2 ......... EE
MTC3 ......... EE
MTC4 ......... EE
MTC5 ......... EE

NDSCAN ...... EUB
NDSCAN1 ..... EUB
NDSERR ....... ET
NDTERR ....... ET
NEWXTN ....... EJ
NLSERR ....... ET
NLUERR ....... ET
NODCUX ....... EJ
NOTSEQ ....... EH

NULCHK ....... EB
NUMCON ....... ER
Subroutine: Converts EBCDIC character 0-9 to binary in WRKRG1.
NXTBIT ....... EM

OPLBNF ....... EP
OTSERR ....... ET
OUTLBL ....... EJ

PACKCG ....... EJ
PNPERR ....... ET
PRGUNT ....... EQ

RELEASE ...... EA
Entry point into the RELSE statement processor.
RLSENT ....... EA
Entry point to the coding that is common for both the RELSE and HOLD statements. Register POINT4 contains the address of an instruction to set the PIB flag on or off (Bit 8 of PIB+12).
- If processing the RELSE statement, set the bit off.
- If processing the HOLD statement, set the bit on.
RNGTOP ...... EUA
RSRMCP ....... EP
RSTRT ........ EN
Entry into the RSTRT statement processor.

SCNRL2 ...... EUB
SET .......... EL
Entry into the SET statement processor.
SETEXT ....... EL
SLINCT ....... EL
SXTPOK ....... EH
SYSDATE ...... EL
SYSUPI ....... EL
SYSUPI1 ...... EM

TFILL ........ EF
TLBL ......... EK
TPLAB ........ EF
Entry into the TPLAB statement processor.
TPLEND ....... EF
TXCUU ........ EQ
Subroutine: Converts the operand X'CUU' from hex to binary.
- PUB address is saved in register POINT4.
- PUB pointer is computed in register WRKRG3.
- Device type (from the PUB) is saved in the location DVCTYP.
TXCUU1 ....... EQ
TXCUU2 ....... EQ
UCS .......... EB
Entry point to the UCS statement processor.
UCSDN ........ EC
UCSSCN ....... EC
UCS1 ......... EB
UCS2 ......... EB
UCS3 ......... EC

Job Control ($$BLSTIO), Charts EV-EZ



Supervisor ($$A$SUP1), Charts FA-GY

Phase $$ANERRA, Charts HA-HB

Phase $$ANERRB, Charts HC-HD



Phase $$ANERRD, Charts HE-HF



Phase $$ANERRE, Charts HG-HJ



Phase $$ANERRF, Charts HK-HM

Phase $$ANERRG, Charts HN-HQ



Phase $$ANERRH, Chart HR



Phase $$ANERRI, Chart HS



Phase $$ANERRJ, Charts HT-HU



Phase $$ANERRK, Chart HV

Phase $$ANERRM, Chart JA

| | |
|---|---|
| CCBSTR ................ | JA |
| MSG2 .................. | JA |
| MSG3 .................. | JA |
| MSG4 .................. | JA |
| PHASE1 ................ | JA |

Phase $$ANERRN, Charts JB-JC

| | |
|---|---|
| ATYPE ................. | JC |
| CALPHS3 ............... | JB |
| CCBUNAV ............... | JB |
| EXIT .................. | JC |
| MPTST ................. | JB |
| NOLOG ................. | JC |
| OPFLAG ................ | JC |
| RESET ................. | JC |
| SVCALL ................ | JC |
| TSTCCB ................ | JB |
| TSTRTY ................ | JB |

Phase $$ANERRO, Chart JD

| | |
|---|---|
| CALPH4 ................ | JD |
| CUU ................... | JD |
| LACSW ................. | JD |
| NOCCB ................. | JD |
| PHASE3+8 .............. | JD |
| SWITCH ................ | JD |
| SYCLAS ................ | JD |
| UNPCH ................. | JD |

Phase $$ANERRP, Chart JE

| | |
|---|---|
| AACTION ............... | JE |
| ACTIONA ............... | JE |
| CALPH5 ................ | JE |
| COUNTRG ............... | JE |
| ERRTYP ................ | JE |
| INST .................. | JE |
| IODONE ................ | JE |
| IOERR ................. | JE |
| LOGENT ................ | JE |
| NOLOG ................. | JE |
| PHASES ................ | JE |
| PHASE4+8 .............. | JE |
| RELOC ................. | JE |
| RSTFLG ................ | JE |
| RSTQPT ................ | JE |
| RTYRTN ................ | JE |

Phase $$ANERRQ, Charts JF-JG

| | |
|---|---|
| COUNTRG ............... | JF |
| DECCTR ................ | JG |
| EXCONT ................ | JG |
| IOCOMP ................ | JG |
| KANEXT ................ | JG |
| KRETRY ................ | JG |
| LOGENT ................ | JF |
| MPSTST ................ | JF |
| NOLOG ................. | JG |
| NOPBR ................. | JF |
| PHASES ................ | JG |
| PHASR ................. | JG |
| REPERR ................ | JG |
| RSTFLG ................ | JF |
| RSTQPT ................ | JF |
| SSMASK ................ | JF |

Phase $$ANERRR, Chart JH

| | |
|---|---|
| EXINTR ................ | JH |
| IOINTR ................ | JH |
| KEYINT ................ | JH |
| LOADPSW ............... | JH |
| LDPSW ................. | JH |
| PHASES ................ | JH |
| RESPNS ................ | JH |
| RETRY ................. | JH |
| TSTATD ................ | JH |
| TSTCAN ................ | JH |

Phases $$ANERRS, Chart JJ



Phase $$ANERRU, Charts JK-JL



Phase $$ANERRV, Charts JM-JN



Phase $$ANERRX, Charts JP-JQ



Phase $$ANERR9, Charts JR-JRB

Phase $$ANERRY, Charts JS-JT



Phase $$ANERRZ, Charts JU-JV



Phase $$ANERR0, Charts JW-JX



$$ANERR1, Chart JY



B-Transient Initiator and Nonresident Attention Routines (Section 4)

The first section of this label list contains the labels that are found in the listing only. Although these labels do not appear on any flowchart, an understanding of them is important.

Listing Only Labels

ATABLE
Listing only: Defines a set of internal allocation tables built by the ALLOC processor. Because of this label's physical placement, it also defines the internal allocation table for background programs. The ALLOC processor uses existing limit information, found in the PIB, and allocation information acquired from the ALLOC statement to establish the table. After the allocation data has been validity checked, and posted to the allocation table, it is used to update the appropriate PIB entry. The table expansion is:

1. A halfword of padding for proper alignment. (This field can contain a constant.)

2. A halfword containing the current number of 2K blocks.

3. A fullword containing the save area address.

4. A fullword containing the lower limit address.

5. A fullword containing the upper limit address.

BUFFER
<u>Listing only</u>: Defines a 72-byte I/O area used by either the nonresident attention routine or the foreground initiator.

FLGBYO
<u>Listing only</u>: Defines a byte of program switches:

| | |
|---|---|
| Bit 0 = 1: | Input is from SYSFGI. |
| Bit 1 = 1: | A read has been issued. |
| Bit 2 = | Not used. |
| Bit 3 = 0: | A label block in main storage is to be written on the label cylinder of SYSRES. |
| Bit 4 = 1: | Only a VOL statement can follow. |
| Bit 5 = 1: | Only an EXTENT, or VOL label statement can follow. |
| Bit 6 = 1: | Only an EXTENT label statement can follow. |
| Bit 7 = 1: | Only a DLAB or TPLAB label statement can follow. |

FWRKFL
<u>Listing only</u>: Contains the load address for the foreground program being initiated. Used by the EXEC processor.

F1TBEN
<u>Listing only</u>: An internal allocation table for Foreground 1 programs. (See ATABLE in the listing only section.)

F2TBEN
<u>Listing only</u>: An internal allocation table for Foreground 2 programs. (See ATABLE in the listing only section.)

JBSLUB
<u>Listing only</u>: A halfword work area primarily used with the scan JIB or scan LUB subroutines. When used with the scan JIB subroutine, this area contains the LUB image information (PUB pointer and JIB pointer) found within a JIB entry. When used with the scan LUB subroutine, this area contains a true LUB entry used within the unassign routine for comparisons.

LBLADR
<u>Listing only</u>: Defines an area that contains the address of the temporary label storage area (Foreground origin plus 1728).

LBLSTR
<u>Listing only</u>: Defines an area that contains the address of the label storage area. This address follows the register save area.

LBSLUB
<u>Listing only</u>: Defines an area that contains the LUB entry found by a scan of the LUB table. This label can be described as a parameter passing area.

MTRSVD
<u>Listing only</u>: A one-byte switch used when the file type has been found to be sequential disk (SD).

| | |
|---|---|
| Bit 0 = 1: | The logical units, or volume serial numbers, or the bin numbers are different between this extent and the previous extent. |
| Bit 1 = | Not used. |
| Bit 2 = 1: | This is the last extent. |
| Bit 3 = 1: | The bin numbers are different but the unit numbers are the same between this extent and the previous extent. |
| Bits 4-7 = | Not used. |

PHSNAM
<u>Listing only</u>: Defines a seven-byte area containing the first seven characters of the phase name of the B-transient to be loaded.

PHSNUM
<u>Listing only</u>: Defines a one-byte field containing the last character of the phase name.

RGSVAR
<u>Listing only</u>: Defines the register save area.

RG2SVA
<u>Listing only</u>: Defines an area within the register save area that contains the contents of register 2.

SCNSTP
<u>Listing only</u>: Defines a one-byte area that contains the character that caused the general scan routine to stop scanning. (Contains the scan delimiter)

SYSTBL
<u>Listing only</u>: Contains the table arguments for a translate operation searching for a system class unit.

TABLE
<u>Listing only</u>: Defines the branch vector table used to load and execute B-transients required by either the nonresident attention routine or the foreground initiator. If a START statement has been processed indicating the B-transients are functioning as a foreground initiator, the branch vector table is expanded to handle this function by adding the table entries starting at location INTABL. Each table entry consists of:

- Operation field of the control statement.

- Phase identifier (an alphabetic character).

- Branch vector index factor used to get the first executable instruction of the processing phase.

Phase $$BATTNA Root, Charts KA-KD

BTLOOP .............. KB
Beginning of a table lookup in the branch vector table to find the appropriate B-transient required for further processing.

CHKSTT .............. KB
CONTROL .............. KB
Entry point for the routine to obtain the input statement.

DTCHAT .............. KA
Entry point to the coding that detaches the attention routine from the task selection operation within the supervisor program.
DTCHSZ .............. KA
Test to determine if the task is to be detached.
DTINUN .............. KC
Entry point to the read subroutine (RDSTMT) when continuation information is expected.

ERRRTN .............. KC
Entry point for error processing.
EXCPRG .............. KC
Entry point to the subroutine used to issue a SVC 0.

NDSCAN .............. KD
NVSERR .............. KC
Entry point to the subroutine used to send appropriate error messages.

RDSTMT .............. KC
Entry point to the subroutine used to read input from SYSLOG or SYSFGI.

SCANR1 .............. KD
Entry point to the general scan routine when the operation code field of a control statement is required by the calling routine.
SCANR2 .............. KD
Entry point to the general scan routine when the first operand field of a control statement is required by the calling routine.
SCANR3 .............. KD
Entry point to the general scan routine when other operands are required by the calling routine.

SCNRL1 .............. KD
SCNRL2 .............. KD

TMPAR1 .............. KA
Defines a doubleword save area used in conjunction with the general scan routine (Chart KD). The first word is loaded from register POINT1 with the address of a statement field (operation code or operand). The second word is loaded from register POINT2 with the remaining I/O area (see label BUFFER). Also used as an entry point during B-transient initialization.

Phase $$BATTNB Message Processor, Charts KE-KF

CKEF1F2 .............. KE
Entry point to a subroutine used to return the key of the area referenced as either F1 or F2 in the operand of a MSG statement.

EXTINT .............. KE

MSG ................. KE
Entry point to the MSG statement processor.

SETEXT .............. KF
STEXCD .............. KF
Entry point to a routine used to check and set the external interrupt exit table for linkage to a program that is identified by the key specified.

Phase $$BATTNC CANCEL, LOG, NOLOG, and PAUSE Processor, Charts KG-KH

ANAERR .............. KG
Entry point for error processing.

CANCEL .............. KG
Entry point to the CANCEL statement processor.
CANCLB .............. KG
Start of CANCEL statement processing for a batch only system.
CKBF12 .............. KG
Entry point to a subroutine used to:

1. Identify the operand.

2. Return the corresponding key to the calling routine.

CNCLIN .............. KG
CNCLME .............. KG
Cancel processing steps when it has been determined that the B-transient now being executed is now being canceled.
CNLRTN .............. KG

Entry point to the subroutine used to set the cancel code in the PIB.

LOG ................. KH
Entry point to the LOG statement processor.

NOLOG ............... KH
Entry point to the NOLOG statement processor.

OPRSNT .............. KG

PAUSE ............... KH
Entry point to the PAUSE statement processor.

Phase $$BATTND MAP Statement Processor, Charts KJ-KM

BLNKLD .............. KN

CNVBCD .............. KM
Entry point to a subroutine used to convert a specified binary number to EBCDIC and remove leading zeros.

MAP ................. KJ
Entry point to the MAP statement processor.

OUTPUT .............. KM
Entry point to the Output subroutine.

RVRSCN .............. KM
Start of a reverse scan of the I/O area, BUFFER.

SKPLIN .............. KM
Entry point to the Output subroutine when a line is to be skipped.

STUBGL .............. KL
Entry point to the Output subroutine used to establish a background line for the MAP processor.

STUCRL .............. KL
Entry point to the Output subroutine used when a line of data in the I/O area, BUFFER, is written on SYSLOG and the next line is initialized.

STUF1U .............. KL
Entry point to the Output subroutine used to establish a foreground line for the MAP processor.

STUSPC .............. KL
Entry point to a routine to put the upper limit address in a line for output.

Phase $$BATTNE ALLOC Statement Processor, Charts KN-KP

ALLOC ............... KN
Entry point to the ALLOC statement processor. (See Phase $$BATTNF, label STARTF+2.)

CHKPRN .............. KN
Test for a duplicate operand in an ALLOC statement.

CHKRNG .............. KP
Validity check of the main storage allocation value specified by the ALLOC statement.

CKSCST ................ KN
Valid delimiter check.

CRTBLD .............. KN

GTNXOP .............. KN
Start of a repetitive sequence of code used to get the operands of an ALLOC statement.

INNXEN .............. KN
Start of a repetitive sequence of code to build internal allocation tables. (See Listing-only Section, labels: ATABLE, F1TBEN, or F2TBEN.)

Phase $$BATTNF ALLOC Processor, Charts KQ-KR

CHGSTT .............. KR
Entry point to the subroutine used to enter and exit from the supervisory state. The supervisory state is entered so that the B-transient program can issue a privileged instruction.

CKNDAR .............. KR
Start of a repetitive sequence of code used to set storage protection keys.

CMNWLM .............. KQ
Start of a repetitive sequence of code used to update the internal allocation table (see Listing-only Section, labels: ATABLE, F1TBEN, or F2TBEN) with new limit information.

NVAERR .............. KQ
Error exit.

NXPBNT .............. KR
Start of a repetitive sequence of code to update the PIB table with values from the internal allocation table.

RSPPEA .............. KR

STARTF+2 ............ KQ
Start of the second phase of ALLOC processing. Allocation is performed using three basic steps:

1. Get limit information from the PIB.

2. Get allocation information from the ALLOC statement.

3. Update the PIB information with the ALLOC statement information.

STLLMT .............. KQ

Phase $$BATTNG START Statement Processor, Chart KS

OPLGT ............... KS

START+2 ............. KS
Entry point to the first phase of the START processor.

STARTBG ............. KS

TERM71 .............. KS

Phase $$BATTNH START Statement Processor, Charts KT-KU

ADDRLP .............. KT
Start of a repetitive sequence of code used to relocate the CCW string into the root phase, $$BATTNA.

CHKFGA .............. KU

LOGGER .............. KU
Entry point to the subroutine used to print the exact number of significant characters (nonblank) found in the I/O area, BUFFER, on the logical unit SYSLOG.

MOVTBL .............. KT
Entry point to the second phase of the START processor.

REVSCN .............. KU
Start of a repetitive sequence of code used to perform a reverse scan of the I/O area called BUFFER. The scan searches for a nonblank character to signify an output.

WFMRES .............. KU
Entry point to a subroutine used to set up the label area. The subroutine writes a filemark on the label cylinder.

Phase $$BATTNI, Charts KV-KZ; LA-LG; MJ-ML

ALT ................. KW
Start of a routine to make an alternate assignment.

ASGCHG .............. MK

ASGPUB .............. KW

ASSGN ............... KV
Entry point to the ASSGN statement processor.

CASERR .............. LG
Error exit.

CHKFUA .............. ML

CHKJIB .............. LD
JBSLUB is replaced with LBSLUB. (See Listing-only Section of this label list.) This routine finds any JIBs assigned to a LUB that has been unassigned prior to making an assignment.

CHKMOD .............. KV
Start of a repetitive sequence of code used to check the mode value in the ASSGN statement against the valid mode values in a table.

CHKNXC .............. LF

CHKOWN .............. KX

CHKPUB .............. KZ
Start of a table look-up to find the channel and unit, specified in the ASSGN statement, in the PUB table.

CHKRNG .............. LF
Entry point to a subroutine used to check a field of characters, one by one, for valid limits.

CKNDCH .............. KW
Begins a search for an unchained JIB entry.

CKNXJB .............. LD
Exit point to the scan JIB subroutine, SCNJIB. (See label list for this phase.) The subroutine is entered to reset JBSLUB (see Listing-only Section of the label list) according to any JIB chained to the logical unit.

CMPBPT .............. LA
Test for identical PUB pointers. When the PUB pointers are equal, it means that another LUB is assigned to the physical unit pointed to by the LUB just unassigned. (See UNPA this label list.) If no other LUB with a matching PUB pointer is found, the ownership flag of the PUB pointed to by the LUB in LBSLUB is reset to indicate that PUB is not assigned to any LUB.

DNEERR .............. LG
Error exit.

GETKEY .............. ML

GTNXJB .............. LA
Continues search for LUBs with a PUB pointer that matches the pointer in LBSLUB. However, the search is within the JIB table.

GTNXLB .............. LA
Start of a repetitive sequence of code to get each LUB of a given class and compare its PUB pointer with the PUB pointer of the LUB in LBSLUB.

HEXCON .............. LE
Entry point to a subroutine used to

convert a variable length field of a hexadecimal number in the form X'nn.....' to binary.

HOLD ................ MK
Entry point to the HOLD processor. This routine is used to set a switch in the appropriate PIB assign flag. This switch can be interrogated later by the Job Control program.

IDSERR .............. LG
Error exit.

MKASGN .............. KW
Entry point to a routine used to make the actual assignment during ASSGN processing. The assignment is made by:
1. Establishing the PUB pointer in the LUB.
2. Setting the ownership byte in the PUB.
3. Setting the mode byte in the PUB. (For tape devices only.)

MODRST .............. KV
Test of the mode value. The instruction at this label is modified so that the immediate field contains the set mode command.

NASERR .............. LG
Error exit.

NDCHFD .............. KW
Entry point to a routine used to chain a JIB entry to the JIB table.

NDTERR .............. LG
Error exit.

NJPERR .............. LG
Error exit.

NLUERR .............. LG
Error exit.

NUMCON .............. LE
Entry point to a subroutine used to convert a variable length field of the form nnn....n to binary.

NWPBPT .............. KW

OWNRSH .............. KV
Test to determine if a device has already been assigned to a different area.

READ ................ KX
Entry point to the read processor.

RELSE ............... MK
Entry point to the RELSE processor. This routine turns off a switch in the appropriate PIB assign flag. This switch can be interrogated later by the Job Control program.

RETURN .............. KW
Exit to the root phase, $$BATTNA.

RNGTOP .............. LF

RSTOWN .............. LA

SCNJIB .............. LC
Entry point to a subroutine used to:
- Initialize JBSLUB (see Listing-only Section) with the first and last bytes of the JIB chained to the current pseudo-LUB entry of JBSLUB.
- Return immediately to the calling sequence when an end-of-JIB-chain condition is found.

SCNLBS .............. LA
Entry point to a subroutine used to:
- Return, sequentially, each LUB entry in a given class to the calling routine.
- Return immediately to the calling routine when there are no more entries in a given class.

SCNLUB .............. LB

SNGCHG .............. MK

SNGUNA .............. MJ

SYSXN2 .............. KY
Entry point to the subroutine described below (SYSXXX) when it is entered as a result of UNA statement processing.

SYSXXX. ............. KY
Entry point to a subroutine used to check and convert a field in the form SYSnnn to an address pointer to a LUB entry and PUB entry associated with the logical unit.

TMFAVP .............. LD
Save area for the old JIB pointer. This pointer is stored by the scan JIB subroutine, SCNJIB.

TXCUU ............... KZ
Entry point to a subroutine used to check and convert a field in the form X'cuu' to a PUB entry address, a PUB pointer, device type, and mode reset byte.

UCUERR .............. LG
Error exit.

UNA ................. MJ
Entry point to the UNA (unassign) processor. This routine is used to selectively unassign all the programmer class units of the area specified by the operand of the UNA statement.

UNALOP .............. ML

UNARTN .............. ML

UNPA ................ LA
Entry point to a routine to unassign currently assigned logical units. The subroutine saves the LUB entry of the LUB to be unassigned in location LBSLUB. It then unassigns the LUB in the LUB table. It checks the LUB table and JIB table for other LUBs that point to the physical unit pointed to by the LUB just unassigned. It resets the ownership flag in the PUB if no other LUBs point to that physical unit. Any stored alternate assignments found in the JIB table are treated as LUBs (unassigned followed by a search for matching PUB pointers).

UNPAN2 .............. LA
Entry point to the unassign subroutine for the UNA statement processor.

Phase $$BATTNJ LISTIO Statement Processor, Charts LH-LM

ASGLST ............... LK
Entry point to the subroutine for listing I/O assignments when all preliminary setup steps have been completed.

CHKF1 ............... LH
Test for an F1 operand.
CHKF2 ............... LH
Test for an F2 operand.
CHKUA ............... LH
Test for a UA operand. If UA is found, the header 'UNASSIGNED' is put into the I/O area, BUFFER.
CKPBUA .............. LH
Start of a repetitive sequence of code to test the status of the devices in the PUB table, searching for an unassigned device that is not down.
CUAPNX .............. LH
Check the next PUB entry.

LANXJB .............. LK
Entry point to the routine used to get a chained JIB entry and inspect its status.
LISTIO .............. LH
Entry point to the LISTIO processor.
LSTASG .............. LM
Entry point to the subroutine used to calculate address of the PUB, convert the channel and unit information to hexadecimal EBCDIC, set up IGN and UNA headers as required, and call the subroutine to output the I/O area.
LSTAUN .............. LK
Entry point to the subroutine used to list the assignments for either F1 or F2 programmer class units. The subroutine sets up primary and secondary headers, calls the LUB scanning subroutine and the JIB scanning subroutine, and calls the final output subroutine.
LSTBG ............... LJ
Entry point to the subroutine used to list BG units. When this routine is used the System Class units are also listed preceding the BG units.
LSTBUN .............. LK
Entry point to the subroutine used to list assignments for BG programmer class units or System Class units.
LSTF1 ............... LJ
Entry point to the subroutine used to list F1 units.
LSTF2 ............... LJ
Entry point to the routine used to list F2 units.
LSTPRG .............. LL
Spaces one line and resets program switch SYSSWH causing the listing of System Class units to stop (Chart LK).
LSTSTD ................ LL
Reset switch STDSWH causing the listing of standard assignments to stop (Chart LK).
LSTUA ............... LN
Program switch set to NOP (Chart LH) when a UA operand is found. The switch is reset to branch when the header, 'UNASSIGNED' has been printed.

NDSCAN .............. LL
NOTASG .............. LM
Sets IGN in the I/O area, BUFFER.
NOUNIT .............. LL
Entry point to output a 'NONE' header if no units were listed as unassigned.
NXTLUB .............. LK
Start of a repetitive sequence of code used to get each LUB entry in a given class.

OUTPUT .............. LM
Entry point to the subroutine used to output and clear the I/O area.

SAVLUS .............. LK
Save the logical unit specification.
SPACE ............... LL
Entry point to a subroutine used to clear the I/O area, BUFFER, so that a blank line will be logged.
STDSWH .............. LK
Program switch set to branch when stored standard assignments are to be logged. The branch is taken at the end of the JIB table scan. The scan finds any stored standard assignments. The switch is reset at location LSTSTD, Chart LL.
SYSSWH .............. LK
Program switch set to branch when system units are to be logged. The switch is set to branch by the list BG routine, Chart LJ. The switch is reset to NOP after the System Class units have all been logged (Chart LL).
SYSUNT .............. LK

UALNOT .............. LN
UAPSWH .............. LH
Program switch set to branch after unit assignments have been listed. This switch is initialized in the NOP state. It is set to branch just before the 'UNASSIGNED' header is logged.

Phase $$BATTNK VOL Statement Processor, Charts LP-LW

CNUNCO .............. LS
Entry point to a subroutine used to check and convert a field of the form SYSnnn, when n equals any number in the range 0-9, to a logical unit class.
CONCAT .............. LR
Entry point to a subroutine used to:

1. Read the second half of a statement.

2. Join the first and second parts of a statement forming a single statement. (This operation is called concatenation.)

3. Reset the address of the operand in the I/O area named BUFFER.

4. Reset the length of the operand.

DLAB ................ LT
Entry point to the disk label (DLAB) processor. A DLAB statement must be preceded by a VOL statement and followed by one or more EXTENT statements. Label information is written on the SYSRES label cylinder for use by the LIOCS open routines.

DLBOUT .............. LT

DOP34 ............... LU
Entry point to the subroutine used to get the third and fourth operands of a DLAB statement. After obtaining the operand, the subroutine checks its validity and converts EBCDIC information to binary.

LAXERR .............. LV
Error exit.

LBLOUT .............. LW
Entry point to the subroutine used to output the label information that has been accumulated in the I/O area, BUFFER. The subroutine:

1. Sets length information in the write and verify CCWs.

2. Determines if space is available on the label track within SYSRES.

3. Updates the disk address if necessary.

4. Checks to ensure label area extents on SYSRES are not exceeded.

5. Sets up the seek address and CCB.

6. Branches to the I/O subroutine (EXCPRG) to write and verify the label information on SYSRES.

NLSERR .............. LV
Error exit.

NLUERR .............. LV
Error exit.

OTSERR .............. LV
Error exit.

RSRMCP .............. LW

TPLAB ............... LQ
Entry point to the tape label (TPLAB) processor. This statement must be preceded by a VOL statement. Label information is written on the SYSRES label cylinder for use by the LIOCS open routines.

TPLEND .............. LQ

TSHORT .............. LQ

VOL ................. LP
Entry point to the volume label (VOL) processor. This statement must precede DLAB or TPLAB statements. The volume label processor:
1. Tests for proper statement sequence.
2. Outputs any label information previously accumulated in the I/O area, BUFFER.
3. Checks the volume information and stores it in the I/O area.

## Phase $$BATTNL XTENT Statement Processor, Charts LX-LZ; MA-MB

INDSEQ .............. LY
Start of extent type, and sequence number checking for an indexed sequential file type.

ISCKSQ .............. LY

ISTYP4 .............. LY
Start of sequence number checking for a type 4 extent.

NEWXTN .............. MB
Exit from XTENT processor. This sequence of code re-initializes the extent processor for subsequent XTENT statements.

NODCUX .............. MB
Sets program switch, MTRSVD, for LIOCS. (See Listing-only Label List.)

NOTSEQ .............. LX
Test for direct access file type.

OUTLBL .............. MB
Calling sequence for the subroutine used to output label information from the I/O area into the label cylinder of SYSRES.

PACKCG .............. MB

SXTPOK .............. LX
Calling sequence for the subroutine used to get the extent sequence number (XTOP12). The sequence number is validity checked to determine if it is in ascending sequence.

XTENT ............... LX
Entry point to the extent (XTENT) processor. The XTENT statement must follow either a DLAB statement or another XTENT statement to be valid.

XTOP3 ............... LZ
Entry point to routine used to:

1. Get the extent limit information.

2. Validity check the extent limits. This routine is generalized so that it

can be used for both lower limit and upper limit extents.

XTOP5 ............... LZ

Entry point to the routine used to:

1. Get and check the serial number and store it in the label area DSECT (I/O area).

2. Convert the SYSXXX field of the extent to class and displacement.

3. Get the B2 field of an extent, convert it to binary, and store it in the label area, DSECT (I/O area).

XTOP12 .............. MA

Entry point to a subroutine used to extract and validity-check the first two operands (type and sequence number) of an XTENT statement. It converts the operand to binary, and stores it in the label area, DSECT (I/O area).

XTOP34 .............. MA

Entry point to a subroutine used to extract limit information from the XTENT statement, perform initial validity checks, convert the numeric EBCDIC limit data to binary, and put the limits into the label area, DSECT (I/O area).

XTOUT ............... MB

XTUNIT .............. MB

## Phase $$BATTNM EXEC and UCS Statement Processor, Charts MC-MF

EXEC ................ MC

Entry point to the execute (EXEC) processor. This phase is the last processing phase of the foreground initiator. The foreground program will be loaded when this phase has finished executing and the foreground program has been chosen by the task selection mechanism of the supervisor.

FINISH .............. MF

INITL ............... MG

MOVLOP .............. MF

Start of a repetitive sequence of code to move the last two routines of the EXEC processor to the main storage area occupied by the root phase, $$BATTNA. The root phase resides in the logical transient area of main storage. The two routines are moved 256 bytes at a time. The last time the move is executed, some remaining number of bytes (less than 256) are moved to the logical transient area.

MOVRTN .............. MF

Entry point to the subroutine used to:

1. Move any label information from the temporary label storage area to the label storage area.

2. Clear the remainder of main storage to initialize it for the foreground program being initiated.

MVCLRT .............. MF

NOLBPR .............. ME

PNPERR .............. MC

Exit to the subroutine used to send error messages (ERRRTN). The subroutine eventually exits to CONTROL (Chart KB).

RSRMCP .............. MD

UCS ................. MG

UCSDN ............... MG

UCSSCN .............. MG

UCS1 ................ MG

UCS2 ................ MG

UCS3 ................ MG

UCS4 ................ MG

## Phase $$BATTNN Timer Statement Processor, Chart MH

CHKOWN .............. MH

Test for timer ownership.

TIMER ............... MH

Entry point to the TIMER processor.

TIMLNK .............. MH

Entry point to a subroutine used to assign the system timer to the program specified in the TIMER statement.

TNAERR .............. MH

Error exit.

## Phase $$BEOJ (Supervisor B-transient), Charts NA-NC

ARCANCEL ............ NA

B-transient $$BTERM is fetched to perform functions of an Attention Routine cancel.

CNCLTEST ............ NA

Routine which determines cause of cancel and selects appropriate subsequent program to be fetched.

DKTYPE .............. NC

If output device to receive dump is disk, the extents for disk device assigned to SYSLST (recorded in DIB table) are checked to see if extents filled. If they are, the dump is bypassed and the next program in the terminating sequence is fetched.

ENT1 .................. NB
Entry point for subroutine which tests for user Program Check and/or Operator Communication Option table entries - if there are any, they are reset to zeroes.
ENT2 .................. NB
Same as ENT1 for Interval Timer Option.
ENT3 .................. NB
EOJSTEP ............... NB
Routine if BG program is canceled due to reaching normal end-of-job.
FGJOB ................. NC
FGLST ................. NC
IJBEJ20+8 ............. NA

INTERR ................ NA

LOGLIST ............... NC
Entry to subroutine to output message regarding canceled program.

MVZ ................... NC
Identification is made of physical device associated with SYSLST or SYS000.

OCTEST ................ NA
Test is made for existence of Operator Communication Option.
ONLIST ................ NC
Determination is made whether message should be outputted on SYSLST or SYS000.
OTHERS ................ NA

PCTEST ................ NA
Test is made for existence of Program Check Option entry indicating user routine will handle program check errors.
PROGCHK ............... NA
PUT ................... NB
Entry to subroutine which uses PIOCS to output message.

RLCCB ................. NC
Depending upon device to be used for output, the appropriate CCW address is placed in a common CCB.

SETLOGUN .............. NC
Routine which sets logical unit address for either SYSLST or SYS000 in CCB after determination has been made as to which symbolic device will be used for message output.
SUPPRIO ............... NB
This routine is entered if an abnormal end-of-job condition exists and the B-transient $$BTERM is executing. Should an I/O unrecoverable error occur which would then cause a cancel of $$BTERM itself, an unending loop would result. Therefore I/O operation is bypassed and $$BTERM is recalled.
SVCERR ................ NA
SVC0 .................. NB
Entry point to subroutine which uses PIOCS to output message.
SVC2 .................. NA

TPTYPE ................ NC
TPTYPE1 ............... NC

UNNORM ................ NA
Routine entered when abnormal end-of-job condition exists. Investigation will be made as to cause of cancel and type of program executing to determine which B-transient of the terminating phases to call next.

## Phase $$BEOJ3 Supervisor B-Transient, Chart ND

CLI ................... ND

GARY .................. ND
GO .................... ND

HALTIO ................ ND

ITERATE ............... ND

LASTPUB ............... ND

PUT ................... ND

QUEUE ................. ND

TM .................... ND

## Phase $$BTERM Supervisor B-transient, Charts NE-NH

ADDLST ....... NF
The pointer from FAVP byte which was pointing to the first available JIB before this terminating phase began is put in the chain byte of the last-dequeued JIB (using register 8 as an intermediate storage). The second byte of the LUB has a pointer to the first JIB associated with that LUB; this pointer is now put in the FAVP byte.

CHAIN ........ NF
Routine which zeroes additional JIB entries if a LUB has more than one JIB.
CONTSCAN ..... NG
Program ownership (F1, F2, or BG) of devices is determined.

DEQUEUE ...... NF
The JIB pointer from the LUB is temporarily stored at label FRLSTBEG. The JIB pointed at by the LUB is addressed and its first 3 bytes are zeroed. The chain byte (4th byte) of the JIB is checked for additional JIBs in the chain; if there are any, they too have their first 3 bytes zeroed until the end of the chain is reached.

DEQUEUED ..... NF
The LUB being unassigned is made X'FFFF'. The next LUB is then addressed and tested for associated JIBs. This continues until all the LUBs have been checked.

DONE ......... NE
Registers are prepared to do a PUB table scan.

ENDPUB ....... NE
Test is made to determine if end of PUB table has been reached.

FG1 .......... NG

GETBYTE ...... NG
Tape Error Block (TEB) data is retrieved and prepared for logging; the first 3 bytes of the TEB entry are zeroed and the statistics are logged.

GETNXT ....... NG
Entry point to subroutine for actual printing of TEB statistics on SYSLOG.

LOG .......... NH

MOD .......... NE
PUB Job Control flags are reset for the devices which are owned by the program being terminated.

MTAPE ........ NG
The device type from the PUB table entry for the device is examined. If the device is not a tape drive, the PUB scan proceeds to the next entry in the table; if it is a tape drive, the Tape Error Block (TEB) for that particular drive is addressed and checked for any record of tape errors. Should this tape drive have experienced no errors, the PUB scan resumes and the next device in the PUB table is investigated.

MVI .......... NE
A detach flag is posted in the PIB for the terminated program; the portion of core occupied by this program is now available for overlay. An End-of-Termination switch is set in the PIBPUBAS flag byte, an SVC 22 releases control of the system from this program and an SVC 11 returns the system to the Task Selection routine of the supervisor.

NOP .......... NG
Switch to enter or bypass the routine which prints headings prior to logging the Tape Error Block (TEB) statistics. Since only one set of headings is needed, this routine will be used only for the first TEB statistics logged; after being entered once, this routine is subsequently bypassed by making this switch an unconditional branch.

OUTAR ........ NE
Entry point to this program phase after it is fetched. The output area address is loaded into a CCW; register 13 is loaded as a link register to the unassign routine. The terminated program is identified as being an F2 job or otherwise; if F2, the ownership flags will be reset in the PUB entries of devices owned by this program.
If the program is not an F2 program, it is therefore an F1 because the $$BTERM phase is called to terminate foreground programs. The PIB assign flag byte is checked to see if the cancel switch is on which indicates cancel occurred while in a terminator phase due to an I/O malfunction. To prevent a repetitive cancel-within-cancel loop, a branch is set in the switch at label LOG to suppress further I/O operations.

PHYSEIZE ..... NE
Further I/O operations are disabled and an SVC 22 is issued which disables multiprogramming and gives this program control over the system to complete its desired functions until another SVC 22 is issued to release control.

PRCOMPL ...... NE
Routine which gets the job name from the foreground save area, to identify the job and writes a message "PROGRAM COMPLETED."

SETUP ........ NE
If it is not a normal end-of-job step and is not the Attention Routine which is being canceled, the Tape Error Block (TEB) statistics are obtained and logged.

SKIPHDR ...... NG
Routine prepares to print channel and unit number of the device, the permanent read error count, and the number of times the read error routine is entered; this data is obtained from the Tape Error Block (TEB) for the tape drive.

TEB .......... NG
Entry point to the subroutine which scans the PUB table for tape devices and logs the tape error statistics.

TEST ......... NF
LUB entries are checked to see if they have any associated JIBs.

UNASSGN ...... NE
Check is made if symbolic device assignments should be reset.

UNASSGN+8 .... NF
The LUBNDX from the PIB of this program (F1 or F2) is inserted in register 5. In the case of F1, for example, LUBNDX is equal to the sum of the LUBs assigned to devices owned by the system programs, the background program, and the Foreground 2 program. This index is doubled because there are 2 bytes per LUB entry. The result is the

displacement, from the LUB table starting address, where this foreground program's LUBs begin. By adding this displacement to the LUB table starting address, the actual address for the first LUB is obtained in register 5.
The number of LUBs assigned to this type of foreground program is obtained from the NICL (Number-in-Class); this value is adjusted and doubled.

Phase $$BEOJ1 Supervisor B-transient, Charts NJ-NK

Label Chart
BALR14 ....... NK
Exit and return to PRINT subroutine for output message when cancel of program occurs during execution of a logical B-transient phase.

CAUSE1 ....... NJ

DKTYPE ....... NK
Routine which checks the extents of SYSLST from the SYSLST disk information block (DIB) once the I/O device is identified as a disk. CCWs and a CCB are prepared for disk use.

FGJOB ........ NJ
Canceled program has been identified as a foreground program so message output will be on SYS000. The SYS000 LUB is located and tested to see if it is assigned. If yes, register 4 is set to use the SYSLST LUB when reference is made to SYS000 so that the output message actually occurs on the device assigned to SYSLST.
FGLST ........ NK

LOGGER ....... NJ
Routine which prepares to output message on SYSLOG device, providing that the device is present, assigned, and not the same physical device as SYSLST. If they are the same, SYSLST is used.

NAMED ........ NJ
This label indicates task of moving job name to output area has been accomplished; now the cause of program cancellation is to be investigated.

ONLIST ....... NJ
Determination has been made that the output message will occur on SYSLST.

PRINT ........ NK
Entry to subroutine which uses PIOCS to write the output message.
PROG ......... NJ
When a program check has been found as the cause of cancellation, the appropriate message code and message are placed in the output area, and the total length of the output message is stored in the CCW byte count field.

SETLOGUN ..... NK
The logical unit number is stored in the CCB, and the PUB entry for the output device is located.
STH .......... NK
TERM ......... NK
A supervisor call is issued to fetch the $$BPSW program.
TPTYPE ....... NK
TPTYPE1 ...... NK
Routine which investigates the physical I/O output device.

Phase $$BEOJ2 Supervisor B-Transient, Charts NL-NM

Label Chart
A ............ NL
Cancel code from PIB of canceled program is compared with codes entered in a look-up table until the code is identified, or the last entry is reached which covers unrecognized codes.

B ............ NL
Routine which uses pointer from lookup table to displace into an actual message code table to obtain code for output message. Name of background job is retrieved from communications region and moved to message output area.

C ............ NL
Name of foreground job is retrieved from save area and moved to message output area.
CAUSE1 ....... NL
Entry point to start of $$BEOJ B-transient.

D ............ NL
Routine which obtains address of desired message and moves actual message to output area after having first cleared storage where previously used instructions resided for use as output area. Also sets count value in count field of CCW to be used.
DKTYPE ....... NM
Routine which checks DIB for SYSLST to see if extents are full; sets current address from DIB into CCW for outputting message.

FGJOB ........ NM
FGLST ........ NM

LOGGER ....... NM
Entry point to subroutine which outputs message on SYSLOG, SYSLST or SYS000. If

these symbolic devices are unassigned no message is written. The next B-transient terminating program is fetched.

MVC .......... NL

ONLIST ....... NM

PRINT ........ NM
Entry point to subroutine which uses PIOCS to output message.

RLCCB ........ NM
Address of CCW for the output device is placed in the CCB.

SETLOGUN ..... NM
Logical unit address is placed in CCB.

TPTYPE ....... NM
TPTYPE1 ...... NM

## Phase $$BILSVC Supervisor B-Transient, Charts NN-NQ

Note: Labels used in this program which have the same function as those of $$BEOJ1 are discussed in the label list of program phase $$BEOJ1. Only those labels which are significantly different or are unique to this program phase are discussed here.

Label Chart
DKTYPE ....... NP
FGJOB ........ NN

FGLST ........ NP
FGTAPE ....... NP

L ............ NN
The address of the instruction issuing the illegal supervisor call is obtained and translated to hex; the illegal SVC code is also translated to hex and both become part of the output message.
LA ........... NP
MVC .......... NP

NOP .......... NP

ONLIST ....... NN
OVLAY ........ NN
Job name taken from foreground save area will be overlaid by the job name from the communication region if the program is found to be a background type.

PNFORSVC ..... NN
Determination is made whether the cause of program cancel was due to an illegal SVC or a phase-not-found condition.
PRINT ........ NQ

RELOC ........ NN
RLCCB ........ NP

SETLOGUN ..... NN

TERM ......... NP
TPTYPE ....... NP
TPTYPE1 ...... NP

## Phase $$BPSW Supervisor B-Transient, Charts NR-NS

Note: Labels for this program are identical to those of $$BEOJ1. Please refer to discussion of labels on $$BEOJ1 label list.

Label Chart

BALR14 ....... NS

DKTYPE ....... NS

FGJOB ........ NR
FGLST ........ NS

LOGGER ....... NR

ONLIST ....... NR

PRINT ........ NR

SETLOGUN ..... NR

STH .......... NS

TERM ......... NS
TPTYPE ....... NS
TPTYPE1 ...... NS

## Phase $$BPCHK Supervisor B-Transient, Charts NT-NU

Label Chart
DKTYPE ....... NT
Routine which locates disk information block owned by SYSLST, checks the SYSLST extents for room remaining, and stores the current address in the seek CCW.

FGJOB ........ NT
Indicates program canceled was a foreground program, and symbolic device SYS000 must be checked for assignment.
FGLST ........ NU
Canceled program has been identified as foreground.

LA ........... NU
Data address for disk output area is placed in write count, key and data CCW.

MVC .......... NU
CCWs are prepared for disk use.

NOP .......... NU
After first line of message is output this switch is set to branch and the second line of the message will be output to the disk; the branch condition causes the B-transient $$BDUMP to be fetched.

NOTBG ........ NT
Routine which recovers the address of the instruction which caused the program check so it can be identified in the output message.

ONLIST ....... NT
Message will be output on SYSLST rather than SYSLOG.

PRINT ........ NU
Subroutine which uses PIOCS to output message.

RLCCB ........ NT

SETLOGUN ..... NT
SYSLST logical unit number is set in CCB and PUB entry address for the device is obtained.

START ........ NT
Entry point to this program to start execution.

TERM ......... NT
Exit from this program is a supervisor call to fetch $$BDUMP.

TPTYPE ....... NT

TPTYPE1 ...... NT
SYSLST is identified as a tape drive, and the CCWs and CCB are prepared for tape output.

Phase $$BDUMP Supervisor B-Transient, Charts NV-NW

BAL1 ......... NW

DSKRT ........ NV

FETCH ........ NW

FGJOB ........ NW
The upper storage limit of the program to be dumped is calculated by: obtaining from the PIB of the canceled program the number of 2K blocks of storage the program occupies, multiplying it by 2048, adding the result to the address of the end of the supervisor area (= BG save area address), and subtracting one byte. Register 8 will communicate this value to the dump program fetched.

NOCHNG ....... NW
Routine used when a foreground program is to be dumped, to identify the physical I/O device associated with SYS000. The type of device determines which B-transient dump program will be fetched to perform the actual dump.

PRINTER ...... NV

PUT .......... NW

SETCODE ...... NV

SYSTST ....... NV
Routine similar to the NOCHNG routine. Identifies the physical device assigned to SYSLST for a background program dump.

TAPE1 ........ NW
When SYS000 is found to be a tape drive, the CCB and CCW are modified accordingly to perform a sense operation for a file-protect condition. Register 12 signals the fetched dump program that a tape drive receives the storage dump.

TERM ......... NW

TPTYPE ....... NW

Phase $$BDUMPF Supervisor B-Transient Charts NX-PA

| Label | Chart |
|---|---|

ALTER ........ NZ
Switch to enter or bypass SPECIAL routine that blanks printing of the first two storage data words. To illustrate the use of this SPECIAL routine, consider the example where the beginning address of a problem program or parameter dump falls between 3F8 and 3FF. To begin print of the dump at the nearest lower double-word boundary, it is necessary to blank out data from 3F0 through 3F7.
In the case of a parameter dump, an additional calculation is made to determine the number of additional blanks needed, should the desired starting address be 3FC; this number is put in register 2 by the $$BPDUMP monitor phase and passed to the phase actually performing the dump. This switch will therefore be a NOP only once (if needed) at the outset of the problem program portion of a dump, or a parameter dump and will normally be set to a branch.

ALTER1 ....... NZ
Routine that puts an extra 2 spaces between groups of 4 words, making a total of 3 spaces. This makes the dump easier to read since storage locations such as 1B0, 1C0, 1D0, etc., stand out clearly. The word counter, register 0, used for this grouping function is reset to 4.

ALTER2 ....... NZ
This routine increments register 6 that points to locations along the print line where data information is being

assembled. It is incremented by 9 for each new word to be printed; one for the space between words and 8 for the print positions of each unpacked word.

ALTER3 ....... NZ
Switch to enter or bypass instructions that create 2 blank spaces between the location counter and first word of storage data. Switch will be set to branch except when preparing the first word of each new print line.

BLNK2 ........ NZ
Sublabel of routine discussed under ALTER3.

BTSTCR ....... PA
Branch and link to TSTCOR subroutine is followed by comparing characters of the next line to be printed with those of the line just printed. If the next line is identical, a switch is set to branch to the CLRLIN routine that will suspend printing the identical line and prints---SAME---instead.

CLRLIN ....... PA
See discussion of label BTSTCR.

CMPCOR ....... PA
Register 5 contains the highest storage location that prints for any single line. Register 5 is compared to register 8 (which contains the upper storage limit of the dump) to see if limit of dump will be exceeded should the entire line be printed. If register 5 is higher than register 8, the value in register 8 is then loaded into register 5 and the printing ceases at the dump limit.

CORE ......... NX
Register 7, containing the beginning storage address of the problem program area, is tested for proper boundary alignment. If it is not on a boundary that is a multiple of 16, it is adjusted to a boundary such as 1B0, 1C0, 1D0, etc., and the switch at ALTER is set to NOP. See label ALTER.

CORE1 ........ NX
Preparation of problem program identification.

CORE2 ........ NX
Problem program portion of main storage is dumped up to limit address contained in register 8.

CORE3 ........ NX
Routine that obtains and prints information about the length of the label area of main storage.

ENDLIN ....... PA
See discussion of ENDLIN1 label.

ENDLIN1 ...... PA
Instructions at ENDLIN and ENDLIN1 are used to index the location counter, register 7, and identify the storage limits to be printed as the last line of the dump when printing of identical portions of storage had been suspended.

FPSW ......... NX
Switch used to bypass routine for printing floating-point registers if this feature is not present on the system.

LST .......... NZ
Switch used to return from REGPNT subroutine, when last word of a printline has been unpacked and printed, to prepare the next line. For register printing and user's communication region printing it will be a NOP; this permits entry to a routine which blanks out unneeded high order positions of the printline.

LSTLN ........ PA
The location counter, register 7, is set and translated to identify the storage locations being printed on each line of the dump. This label is also used to enter the PRNTLN subroutine on a last line condition, thereby bypassing the TSTCOR subroutine.

NOTEST ....... NX
An area of storage used for phase initialization instructions is blanked out to be used as an output area for the dump. If needed, a branch will be taken past the end of the cleared area to the next instruction.

OUT .......... PA
Switch made a NOP when supervisor portion of dump is completed. During the problem program portion of the dump, the switch permits exit from dump phase by fetching $$BEOJ when the dump limit is reached.

OUT1 ......... PA
Switch set to branch if SYS000 is a tape drive, to write a tape mark following the record of the last line of the dump.

OUT2 ......... PA
Exit from foreground program dump $$BDUMPF by fetching B-transient phase $$BTERM.

PAGHED ....... NY
Entry to subroutine that prepares and prints page headings.

PRINT ........ NY
Routine that uses PIOCS to perform I/O operations.

PRINTL ....... NZ
Entry to subroutine that loads address of CCB into register 1 and goes to PRINT label discussed previously.

PRNTLN ....... PA
Entry to subroutine that defines an area of storage to be printed on a line, obtains, edits, and prints the data.

REGPNT ....... NZ

Entry to subroutine that obtains, edits, arranges and prints: register data, user's portion of the communication region information and label length information.
REGPNT1 ...... NZ
REGPNT5 ...... NZ
Entry to subroutine REGPNT which bypasses the blanking of the I/O area.
RELOCF ....... NX
If output device is not a tape drive, the CCB is supplied with the CCW address of the alternate device.

SPECIAL ...... NZ
See discussion of this label under ALTER.
START ........ NX
Entry point to the program phase fetched into the logical-transient area.
START1 ....... NX
SUPV ......... NX
Beginning of routine that dumps supervisor program portion of main storage.
SUPV1 ........ NX
Return to program mainline after supervisor program has been dumped.

TAPNOP ....... NY
Switch set to branch if SYS000 is a tape drive.
TAPRTN ....... NX
Data address is stored in the tape CCW and CCB is furnished with the CCW address. Switches at OUT1 and TAPNOP are set to branch to perform functions necessary for output on tape drive.
TAPSYS ....... NY
TAPSYS1 ...... NY
Switch to a NOP if an end-of-volume condition is detected on the tape drive receiving the dump.
TPMARK ....... PA
Routine for writing a tape mark following last record of dump.
TSTCOR ....... PA
Entry to subroutine that tests whether storage area to be printed on a line is in dump limits and whether the next line will be the last line. Register 3 is a pointer to the storage address of the first byte of a line to be printed and register 5 points to the last byte of the line. See CMPCOR label discussion which is part of this subroutine.
TSTLST ....... PA
The storage address of the last byte of the next intended print line is tested to determine if it is the last line of the dump. If it is, the program enters routines to end the dump.
TSTLST1 ...... PA
Switch that is set to a branch on last line of dump. If a portion of core is found that is identical to the previous line, this switch is set to a NOP and the identical data is shown by printing a line with --SAME--.
TSTPRT ....... PA
Switch that determines if data will be edited and printed as a normal line, or if --SAME-- will be substituted for consecutive identical lines.

UNPK ......... NZ
An entry point into the subroutine REGPNT discussed previously.
UNPK1 ........ NZ

## Phase $$BDUMPB Supervisor B-transient, Charts PB-PF

Note: The labels for this program that are identical to those of the $$BDUMPF program are discussed in the label list of $$BDUMPF. Only those labels which differ significantly or are unique to this program are expanded here.

Label          Chart
ALTER ........ PE
ALTER1 ....... PE
ALTER2 ....... PE
ALTER3 ....... PE

BAL1 ......... PB
Routine that blanks out initializing instructions of this phase so this portion of storage can be used as an I/O output area.
BLANKS ....... PE
Blanks are used to blank out the unneeded high-order positions of the printline area when the registers and user's part of the communication region are printed.
BLNKST ....... PE
Switch that determines if BLANKS instruction will be used; switch will be set to branch except for conditions given under BLANKS label.
BLNK2 ........ PE
BTSTCR ....... PF

CLRLIN ....... PF
CMPCOR ....... PF
COMM ......... PC
Routine for preparing user's portion of communication region data for printing.
CORE ......... PC
CORE1 ........ PC
CORE2 ........ PC
CORE3 ........ PC

ENDLIN ....... PF
ENDLIN1 ...... PF

FPSW ......... PC

LST .......... PE
LSTLN ........ PF

NOTEST ....... PB

OUT .......... PF

PAGHED ....... PD
PRINT ........ PD
PRNTL ........ PE
PRNTLN ....... PF

REGPNT ....... PE
REGPNT1 ...... PE
REGPNT5 ...... PE
RELOC ........ PB
   Same as RELOCF label in $$BDUMPF program.

SPECIAL ...... PE
START ........ PB
SUPV ......... PC
SUPV1 ........ PC

TPRTN ........ PB
   CCWs and CCB are prepared for use with tape drive as output device to receive dump.
TRANS ........ PE
   Data is translated into printable characters for dump print-out. Translate operation is defined in the <u>IBM System/360 Principles of Operation</u> manual.
TSTCOR ....... PF
TSTLST ....... PF
TSTLST1 ...... PF
TSTPRT ....... PF

UNPK ......... PE
UNPK1 ........ PE

<u>Phase $$BDUMPD Supervisor B-Transient, Charts PG-PK</u>

<u>Note</u>: The labels for this program that are identical to those of the $$BDUMPF program are discussed in the label list of $$BDUMPF. Only those labels that differ significantly or are unique to this program are expanded here.

<u>Label</u> <u>Chart</u>
ALTER ........ PJ
ALTER1 ....... PJ
ALTER2 ....... PJ
ALTER3 ....... PJ

BAL1 ......... PG
   Routine that blanks out initializing instructions of this phase so this portion of storage can be used as an I/O output area.
BLANKS ....... PJ
   Blanks are used to blank out the unneeded high-order positions of the print line area when the registers and user's part of the communication region are printed.
BLANKST ...... PJ
   Switch that determines if BLANKS instruction will be used; switch will be set to a NOP except for conditions given under BLANKS label.
BLNK2 ........ PJ
BTSTCR ....... PK

CLRLIN ....... PK
CMPCOR ....... PK
COMM ......... PG
   Routine for preparing user's portion of communication region data for printing.
CORE1 ........ PG
CORE3 ........ PG

DUMP ......... PG

ENDLIN ....... PK
ENDLIN1 ...... PK

LOAD ......... PH
   The data address for the output area is loaded into the disk CCWs, and the address of the first CCW of the chain is put in the CCB.
LST .......... PJ

LSTLN ........ PK

MOVE ......... PH
   Current address taken from the Disk Information Block (DIB) for the appropriate symbolic disk device is put in output area to serve as the count ID information when count, key and data are written. The current address record number is then reduced by 1 and put in the search CCW for writing the first dump record.
MVBLNK ....... PJ
   Sublabel of SPECIAL routine. Refer to label SPECIAL in $$BDUMPF label list.

NOTEST ....... PG

OUT .......... PK
OUT1 ......... PK

PAGHED ....... PH
PDUMP2 ....... PG
   Switch set to branch if it is a parameter dump; will bypass printout of all parts of core except the area specified in the parameter limits.
PRINT ........ PH
PRINT1 ....... PH
   Routine that uses PIOCS to seek, search ID equal, write count, key and data, verify, and wait for completion of the I/O operation.
PRNTL ........ PJ
PRNTLN ....... PK

REGPNT ....... PJ
REGPNT5 ...... PJ

SPECIAL ...... PJ
SUPV1 ........ PG

TSTCOR ....... PK
TSTLST ....... PK
TSTLST1 ...... PK
TSTPRT ....... PK

UNPK ......... PJ
UNPK1 ........ PJ

<u>Phase $$BPDUMP Supervisor B-Transient, Chart PL</u>

<u>Label</u> <u>Chart</u>

BAL1 ......... PL
Sense data is tested for file-protect condition if SYS000 is a tape drive; if it is protected, the dump cannot be taken and this program phase returns to supervisor for selection of next task. If not protected, B-transient $$BPDUM1 is fetched to perform the actual parameter dump.

DISKRT ....... PL
Register 3 is loaded with a 1 prior to fetching B-transient $$BDUMPD to signal that this is a parameter dump.

EXIT ......... PL

FGJOB ........ PL
Routine that identifies SYS000 device type.

FIX7 ......... PL
See discussion of next label.

MAIN1 ........ PL
The starting address for the parameter dump, entered in register 6, is shifted right double logical 4 positions so that any value not a multiple of 16 will now be in register 7. If value in register 7 is now zero, it indicates that the starting value in register 6 is on a double-word boundary. Register 6 is then restored by shifting left to the next lower double-word boundary nearest the value specified by the dump parameter (label FIXT). If register 7 was not zero when tested, the value now in it is used to calculate the number of blank print positions needed so printout starts at desired starting byte.

MAIN2 ........ PL
The upper parameter address is incremented by a word length and tested against system's main storage capacity to see if requested dump is a valid address within core. If not, the upper storage limit is put in register 8 to impose a valid dump end limit.

OUT .......... PL

PRINT ........ PL
Entry to subroutine that uses PIOCS to test for a tape file-protect condition in the event a foreground program dump is taken and SYS000 is a tape drive.

PROCED ....... PL

START ........ PL
Entry point to B-transient $$BPDUMP.

<u>Phase $$BPDUM1 Supervisor B-transient, Charts PM-PQ</u>

<u>Note</u>: The labels for this program that are identical to those of the $$BDUMPF program are discussed in the label list of $$BDUMPF. Only those labels that differ significantly or are unique to this program are expanded here.

<u>Label</u> <u>Chart</u>

ALTER ........ PP
ALTER1 ....... PQ
ALTER2 ....... PQ

BAL .......... PM
Routine that blanks out initializing instructions of this program phase so this portion of storage can be used as an I/O output area.

BTSTCR ....... PN

CLRLIN ....... PN
CMPCOR ....... PP

DATE ......... PP

ENDLIN ....... PN
ENDLIN1 ...... PN

EOVFG ........ PQ
End-of-volume has been detected and the program decides whether a foreground or background program is being dumped. Foreground programs dump on symbolic device SYS000 while background programs use SYSLST.

EOVFGN ....... PQ
End-of-volume has occurred on the tape drive receiving a foreground program dump (symbolic device SYS000). The class and unit number for SYS000 is retrieved and stored in register 0 to communicate this information to the end-of-volume routine $$BCMT07.

EOVMV ........ PQ
An end-of-volume switch in the communications region is turned on and used by the phase $$BCMTO7 to indicate its function.

FGHED ........ PP
FGNAME ....... PP
FGTST ........ PM
The logical transient key (LTK) is compared with the LTK for a background

program, to determine if the program to be dumped is a background or foreground program.

FIX .......... PQ
When word counter reaches zero, 2 extra blanks are inserted between words so that locations such as 1B0, 1C0, 1D0, etc., will stand out, thus making the dump easier to read.

LST .......... PQ
LSTLN ........ PN
LST1 ......... PQ

MVBLNK ....... PQ

PAGHED ....... PP
PDUMP ........ PM
Register 14 is tested to see if it was loaded by $$BPDUMP phase to indicate a tape drive will receive the parameter dump. If register 14 is zero, the physical I/O device is a printer.
PDUMP1 ....... PM
Routine that prepares the CCWs and the CCB for a printer operation rather than tape operation.
PRINT ........ PQ
PRINTL ....... PQ
PRNTLN ....... PN
PRNTLN1 ...... PN

REGPNT ....... PQ
REGPNT1 ...... PQ
REGPNT5 ...... PQ
REGPNT6 ...... PQ
RELOC ........ PM

SPECIAL ...... PQ
SPECIAL1 ..... PQ
START ........ PM

TPRTN ........ PM
TSTCOR ....... PP
TSTLST ....... PN
TSTLST1 ...... PN
TSTPRT ....... PN

UNPK ......... PP
UNPK1 ........ PP

## LINKAGE EDITOR PROGRAM (SECTION 5)

### Linkage Editor Program ($LNKEDT, $LNKEDT0, $LNKEDT2, $LNKEDT4, $LNKEDT6, $LNKEDT8, $LNKEDTA, $LNKEDTC), Charts QA-SW

The first portion of the linkage editor label list contains labels that can be referred to by more than one phase. These labels do not appear in any flowchart. However, the explanation of these labels describes critical areas of the listing.

### L/E Common Labels

ESDNOO
Listing Only: Contains the disk address of the first ESD card image if this card image is on SYS001.
ESDOOO
Listing Only: Contains the disk address of the first ESD card image if this card image is on SYSLNK.

FCHPHS
Listing Only: Contains the phase number of the linkage editor phase to be fetched next.

NMELST
Listing Only: A list of control sections that have been specified by the name list operand of an INCLUDE card. The list is blank except when a named submodular is still being processed.
NMSBSW
Listing Only: Supplies the information in byte 4 of location PERIDA during INCLUDE card processing in the $LNKEDT6 phase. Resets to zero during the execution of the control card scan in the $LNKEDT4 phase. Bit 6 is set to one during initial INCLUDE card processing and bit 1 is set during phase post processing (autolink mode).
NOBLOK
Listing Only: Third subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the number of blocks required for the specified phase.
NOBYTE
Listing Only: Fourth subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the number of bytes in the last block.
NXPHRG
Listing Only: Seventh and last subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the next available load address. This is a function of phase length.

ONSOOO
Listing Only: Contains the address of the next card to be processed.
ORPHDA
Listing Only: Fifth subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the disk address of the first block of the phase specified in PHEADR (see label list this section).
ORPHRG
Listing Only: Sixth subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the load address of this phase.

PERIDA

Listing Only: The location labeled PERIDA is a 30-byte input control area used by the linkage editor program to:

- Obtain the address of the next card image to be processed after the END card.
- Determine the point at which processing is finished for an object module.
- Maintain control over the nesting of include statements by functioning as last in-first out list to establish processing priorities.

Location PERIDA is used in conjunction with either location ESD000 or ESDN00 (see label list, this section) depending on the input device being used at this time. ESD000 or ESDN00 is loaded with the disk address of the first ESD card image of the object module. PERIDA is loaded with the disk address of card image that follows the control card image. The linkage editor program compares the disk address in location PERIDA with the address in either ESD000 or ESDN00. Input control is based on the result of the comparison that is made at END card time. Possible results and corresponding input control actions are:

- The address in PERIDA is equal to or higher than the address in ESD000. Process the card image sequentially following the END card.
- The address in PERIDA is lower than the address in ESD000. Get the address of next card image to be processed from PERIDA.
- The address in PERIDA is lower than the address in ESDN00. Get the address of the next card image to be processed from PERIDA.
- The address in PERIDA is equal to or higher than the address in ESDN00. Effectively shift PERIDA left five bytes. Get the address of the next card image to be processed from the updated PERIDA.

Before the comparison is made and the appropriate actions are taken at END card time, the linkage editor program ensures a value is available for PERIDA (see RECF00, phase 1 section of the label list).

Location PERIDA establishes processing priority by functioning as a last in - first out list for up to five levels of include (nest depth). The list is built during the execution of the include card processor (Chart KN). Figure 80 illustrates the physical structure of PERIDA and Figure 81 illustrates how this location functions as a last in - first out list.

Note: If all five levels of include are used, the last 5-byte segment of PERIDA contains the address of the card image following the first include statement.

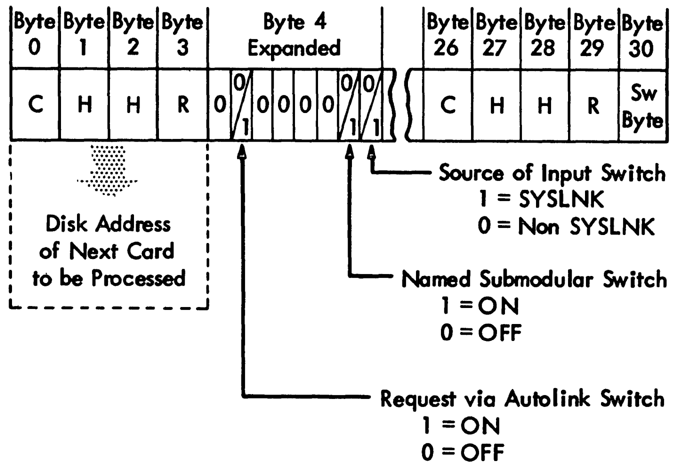



Figure 80. PERIDA Layout

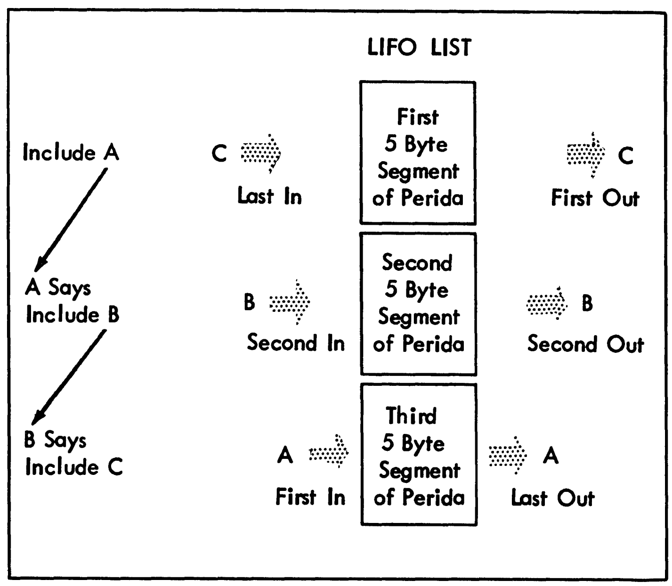



Figure 81. Last In - First Out List (LIFO)

PHEADR

Listing Only: First subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains the 8-byte phase name. This is the beginning of a special purpose I/O area used when phase information is reinitialized.

ROOTNO

Listing Only: This location contains a zero when the first phase is not specified root and a one when it is a root. The value in ROOTNO is either added to or subtracted from the control dictionary number.

- Subtracted - When the control dictionary number is being used to obtain a control dictionary entry address.

  (C/D NO - ROOTNO) X 16 = DISPLACEMENT
  CDENT1 + DISPLACEMENT = ENTRY ADDRESS

- Added - When control dictionary entries are built, an adjustment of one is necessary for each entry when a root phase has been specified.

SBMDST

Listing Only: A program switch that indicates when NMELST should be cleared. SBMDST is turned on when a named submodular [INCLUDE NAME, (CSECT)] is found. At the same time, a bit switch in location PERIDA (see label list this section) is turned on. The apparent duplication of switches is necessary because the first 5-byte segment of PERIDA is a variable, depending on nested levels of include. At END card time, a test is made of the bit switch in location PERIDA.

If bit 6 of byte 4 is on, turn SBMDST off. The switch in PERIDA can then be tested.

If bit 6 of byte 4 is off, do not change the status of SBMDST. The bit switch has already been moved to some other 5-byte segment of PERIDA.

The linkage editor program can then test SBMDST to determine if the END card being processed is part of the module named in the include statement (SBMDST - OFF). The name list (NMELST) is always cleared at END card time, except when a named submodular is still being processed.

TRFRAD

Listing Only: Second subfield of the 28-byte phase header built in the $LNKEDT6 phase. Contains a transfer address.

## Phase 1 ($LNKEDT), Charts QA-QU

Label Chart

ABTERR ...... QM
Loads $LNKEDTA so that abort error processing can continue. Entered when non-recoverable errors are found.

ACTCAN ...... QU

ACTCLR ...... QU
Tests for CLEAR operand.

ACTERR ...... QC
Program switch. It is set to load address (LA) if an error was found on an action card and the main storage available is 14K or more. If the switch is NOP, there is no error and normal processing continues. If the switch has been modified to LA by the action processor (Chart QU), an error handling subroutine is executed after the $LNKEDT0 phase is loaded.

ACTLOP ...... QU
Zeros a core image record.

ACTNMP ...... QU
Tests for NOMAP operand.

ACTNPR ...... QU
Beginning of the action card processor routine.

ACTNTO ...... QU
Tests for NOAUTO operand.

ACTRET ...... QC
Entry point from the action processor (Chart QU).

ACTR16 ...... QC
Program switch. It is set to branch when an error is found on an action card and the main storage available is less than 14K. If 14K or more main storage is available, or if no error has been found, testing continues. The switch is modified by the action processor (Chart QU).

AD1DSK ...... QK
Starting label of the update address subroutine.

ALNKCD ...... QQ
Tests for more ER's left to process.

ALNKGT ...... QQ
Entry point to the autolink routine when autolink is NOT to be done. Provides an exit to load $LNKEDT6.

ALNKOF ...... QP
Entry point to the read input stream subroutine from the autolink processor (Chart QQ).

ALNKPR ...... QQ
Starting address of the autolink processor, which is entered whenever a phase has finished processing and autolink has been requested.

ALNKSC ...... QQ
Tests for end of control dictionary.

ALNKVL ...... QQ
Initially, loads register 6 with the first control dictionary address. Subsequently, ensures the lowest ER in the collating sequence is in register 6.

CDSIZE ...... QN
Starting label of a subroutine to check for control dictionary-linkage table overlap.

CHKSYM ...... QP

CLRLOP ...... QU
Loop established to clear the core image library.
CNVAHX ...... QH
Tests for more characters to convert.
CNVHEX ...... QH
Label at the beginning of a subroutine used to convert hexadecimal to binary notation.
CNVHSW ...... QH
Tests for a hexadecimal number in the range A - F.
CNVSHF ...... QH
Converts hexadecimal to binary notation.
CTLSKP ...... QP
Submodular structure causes skip of cards except an entry card.

DECONT ...... QJ
Instruction that can be modified within the print subroutine. Set to BCTR to decrease the lines-remaining-count by 1 extra line.
DERCAL ...... QT
DERDAD ...... QT
Beginning of subroutine to build core image blocks. Basically, this subroutine:

1. Ensures the text is within the limits of the phase.
2. Finds the core image block the text belongs in.
3. Reads the core image block required by the text into the I/O area.

DERDOK ...... QT
DERDSW ...... QT
Program switch used to force continued processing when a zero length control section is found by the ESD processor (Chart RJ).
DERITE ...... QT
DERLOP ...... QT
Loop to find correct core image block for the text being processed.
DERSW1 ...... QT
DISKIO ...... QM
Entry point to the read/write subroutine when the operation code is already set to the desired operation, the disk search address is set up, and both the CCW and the CCB are already correctly set.
DMPHSW ...... QP
Program switch initialized as a MVC instruction. Modified to an effective NOP by the ESD processor (Chart RA) when a dummy phase card is to be built. By the NOP modification, the disk address of the ESD card not yet processed is retained in location COMNRF for use after the phase processing is finished. (Dummy phase cards are treated as actual phase cards.)

ERRACT ...... QU
Common error exit from the action processor. Error messages are initialized in the action processor, but are actually issued during the execution of another initialization routine (see Chart QC).
ERRO ........ QA
ERROR ....... QR
Beginning of the error handling subroutine.

ERR002 ...... QH
ERR035 ...... QU
ERR036 ...... QU
ERR044 ...... QN
ERR044 ...... QE
ERR044A ..... QE
ERR044C ..... QN
ERR050 ...... QT
ERR070 ...... QE
ERR091B ..... QA
ERR093 ...... QT
ERR094 ...... QM
ERR112 ...... QU
EXLOAD ...... QP

1. Entry point from the autolink routine (Chart CQ) when the control card processing phase is to be loaded.
2. Entry point from the ESD processor when a dummy phase card has been built and a control card processing phase is to be loaded.
3. Entry point from the initialize $LNKEDT2 routine when the ESD processing phase is to be loaded.

FNDENT ...... QP
FNDOP ....... QU
Loop control label.
FNDVRB ...... QC
Finds the operation field of the card image in the input stream.

HDGCAN ...... QC
HDGMAP ...... QC
HDGMSW ...... QC
HDGOVR ...... QC
HDGSW2 ...... QC
HDNGSP ...... QJ
Tests to determine if a heading line was just printed. If a heading line was just printed, spaces one extra line.

IJBINL ...... QA
INLEXT ...... QC
Program switch. When 14K or more main storage is available, it is set to NOP by part one of the initialization routine.
INS000 ...... QD
Entry point when the read SYSLNK subroutine tests for the presence of a record. Also used internally by the RDS000 subroutine.
INTPT1 ...... QB
Begins part one of initialization. Executed after part two.
INTPT2 ...... QA

Begins part two of initialization.
Executed before part one.

INTS01 ...... QA

LOADSW ...... QP
Program switch used to cancel individual phase loading. It is set to NOP by the initialization routine (Chart QB) when all phases are in main storage.

LOGPRT ...... QA

LSETB ....... QE

LTCDAD ...... QE
Entry point to the control dictionary search subroutine from the extract phase subroutine (Chart QL). Computes address of the control dictionary.
(Control dictionary number - ROOT number) x 16 = Control dictionary displacement.
CDENT1 + displacement = desired entry address
This entry point is used when the calling routine already has a usable control dictionary number.

LTCDNO ...... QE
Entry point to the control dictionary search subroutine when a test for control dictionary number assignment is required by the calling routine. The control dictionary number was available but because its status was undetermined, entry is made to this subroutine.

LTCDRF ...... QE
Finds the relocation factor (sign controlled). Also used as an entry point from the phase processor (Chart SC).

LTESID ...... QE
Label at the beginning of the subroutine that finds control dictionary number, control dictionary address, or relocation factor using the language translator supplied ESID number and the linkage editor constructed linkage table. Entry point when the ESID number is supplied.

NDESLP ...... QR

NOLNPG ...... QB

NOTACT ...... QC
Entry point from the action processor when an error has occurred in the action card operand. Also the normal exit when no action card is found.

NOTCTL ...... QR

NTESLP ...... QR

OPEN ........ QA
Opens both SYSLNK and SYS001.

OVFHDG ...... QJ

OVFLOW ...... QJ
Resets lines per page count to 56. Also used as an entry point to the print subroutine. If used as entry point and the first part of the print/carriage control subroutine has been overlaid by the overlay subroutine, exit is provided by the first instruction of the overlay subroutine.
Note: Only the first part of the print/carriage control subroutine is overlaid.

OVRLAY ...... QS
Beginning of the subroutine that overlays the print/carriage control subroutine (Chart QJ), when a NOMAP option is found. OVRLAY types error messages on SYSLOG.

PRERR ....... QR

PRINT ....... QJ
Beginning of the print/carriage control subroutine. If the NOMAP option is selected on an action control card, the overlay subroutine (Chart QS) begins to overlay the print subroutine at this point.

PRLCNT ...... QJ
Instructions at PRLCNT adjust the remaining line count.

PRNEOF ...... QJ
Clears the print line area.

PRTLOG ...... QJ
Entry point from the error processor.

PRTLST ...... QJ
Sets up the print area for map.

RDEXEC ...... QP
Entry point to the read input stream subroutine from the initialize routine (Chart QC).

RDNEXT ...... QP
Starting label of the read input stream subroutine. Normal entry point for this subroutine.

RDPHAS ...... QP
Sets up fetch of desired linkage editor processing phase.

RDS000 ...... QD
Beginning of read input subroutine.

RDUNIT ...... QP

RDWRMV ...... QM

READ ........ QM
Beginning of the read/write subroutine if a read operation is to be performed.

RECFOO ...... QD
Program switch set to branch and reset to NOP by either the ESD processor (Chart RA) or the END card processor (Chart RQ). The ESD processor ensures the correct disk address is in location PERIDA (see L/E Common Label section, PERIDA) by branching to the read SYSLNK subroutine at location INS000 after the CCW has been set to NOP. The disk address of the next card image is located by updating the disk until a record is found. Register 2 supplies the correct disk address for location PERIDA. The END card processor used the same technique to locate the disk address of the next card to be processed (put into location NDS000). RECFOO is set to branch to prevent updating beyond the proper disk address.

RELBSW ...... QP
Program switch that tests for input from

the relocatable library. Sets (branch) in the include processor and resets (NOP) in the END card processor.

SPACE1 ...... QJ
Entry point to the print subroutine when an extra space is required.
SRCHCD ...... QF
Label at the beginning of a subroutine used to find the last duplicate label in the control dictionary.
SRLABL ...... QF
Tests for a duplicate label.
SRPCOD ...... QF
Tests for end of control dictionary. Entry point for the label search subroutine.

TAPE01 ...... QA
If SYS001 is a tape, the DTF table is saved by:
1. Writing the DTF table as the first record on SYS001.
2. Writing a tapemark after the DTF table on SYS001.

At retrieval time (closing of SYSLNK and SYS001), the DTF table is found by first backspacing the file (if RLD's on SYS001) and then backspacing the record.
TISESD ...... QR
TSTMAP ...... QJ
Tests for the MAP option by testing MAPSW.
TYPEVB ...... QQ

VALDVE ...... QM
Tests for a valid symbolic unit (SYSLNK - SYS001).

WRITE ....... QM
Beginning of the read/write subroutine if a write operation is to be performed.

XTPHGT ...... QL
Builds an 11-bit phase number in a register.
XTPHNO ...... QL
Starting label of a subroutine that extracts the phase number from the control dictionary.

## Phase 2 $LNKEDT0, Charts RA-RJ

Label            Chart

CHKNPH ...... RA
Beginning of the ESD processor phase, $LNKEDT0. (See Appendix E for a detailed description of this phase.)
CNCALK ...... RC
The instructions starting at this label cancel autolink by moving a hexadecimal 'FF' into the variable field of the card image. This indicates that no autolink should be attempted on this ER either because autolink has already been tried or because autolink should never be performed.

EISDPC ...... RC
ELBCER ...... RD
ELBCM ....... RD
Beginning of CM (common) processing. Whenever both the ESD item in the card image and the control dictionary entry are commons with matching labels, puts the common with the longest length in the control dictionary. If commons with different labels are found, posts the new common in the control dictionary. The map processor of the linkage editor calculates a total length of commons. This total, or cumulative, length is used to adjust the phase origin.
ELBDSD ...... RF
ELBELR ...... RE
ELBER ....... RE
Beginning of terminal ER processing.
ELBGSD ...... RF
ELBINT ...... RH
ELBLD ....... RG
Beginning of terminal LD/LR processing.
ELBLDR ...... RG
ELBNAS ...... RG
ELBNCD ...... RH
ELBNLR ...... RG
ELBPC ....... RD
ELBSD ....... RF
Beginning of terminal SD processing.
ENDLD ....... RB
EPHLOP ...... RJ
Tests the control dictionary entry to determine if it is an unassigned LD/LR.
EPHSCD ...... RJ
EPHSCN ...... RJ
Loop control label.
EPHULD ...... RJ
ERR040 ...... RB
ERR040 ...... RD

ERR041 ...... RH

ERR042 ...... RB
ERR043 ...... RF
ERR043 ...... RG
ERR045 ...... RC
ERR046 ...... RD
ESCNCD ...... RJ
Sets up for control dictionary scan. Whenever a new ESD entry is posted, the ESD processor tries to assign any unassigned LD/LRs in the control dictionary. It sets up a loop to search for the unassigned LD/LRs. If the ESID has been processed, the ESD processor flags the LD/LR as assigned.
ESDNXT ...... RA
ESDRET ...... RB
Common entry point when the processing of an ESD item is finished. Instructions starting at this label

determine if any more ESD items are in the variable field of the card image.
ESDSBM ...... RC
Loop control label for SD label/name list test.
ESD1ST ...... RA
Tests for the first ESD/SYM record.
ESLBCD ...... RD
Beginning at this label, the ESD processor searches the control dictionary for a label that corresponds to the label of the ESD item from the variable field of the card image. See Appendix E.
EUPDCN ...... RH
Stores the control dictionary number in the linkage table, thereby completing the update procedure.
EUPDLT ...... RH
Linkage table update begins at this label. Entry point whenever the ESD processor requires the update only.
EUPDOK ...... RH
Moves the type field from the card image into the type field of the linkage table.
EUPDSW ...... RH
Tests for a non-process SD.
EUPDXT ...... RJ
Tests for control dictionary/linkage table overlap (see label ELBNCD).
EUPTRY ...... RH
Sets up for exit to a subroutine. Goes to LTESID to get the control dictionary number.

PRSDPC ...... RC
Beginning of SD/PC processing within the ESD processor.

Phase 3 $LNKEDT2, Charts RK-RS

| Label | Chart |
|---|---|

ACSLTH ...... RR
Some compilers supply control section length in the END card. This routine processes the control section length for this special case.

CLREXT ...... RS

EISCSL ...... RR
At this location a length was found in the END card. The next possible phase origin can now be calculated.
EISXFR ...... RQ
ENCRLT ...... RR
Beginning of a loop to clear the linkage table. The table is first built during initialization. Whenever a module is finished processing (signaled by an END card), the linkage table is cleared and the starting address becomes the next available table address.
ENDPRC ...... RQ
Beginning of the end card processor.
ENDRTN ...... RQ
Tests SBMDST to determine if the name list should be cleared. (See SBMDST in label list.)
ENDSBM ...... RR
Program switch providing an exit from the END card processor. The switch is assembled in the NOP state, initialized to branch by the END card processor, and reset to NOP by the END card processor if the module being processed has been autolinked.
ENDSCD ...... RR
Searches control dictionary for unassigned LD/LRs and ensures the control dictionary number for such items is negative.
ENDTOO ...... RQ
ENDXFR ...... RQ
ENOXFR ...... RQ
ENUNAS ...... RR
ERR000 ...... RK
ERR013A ..... RM
ERR051 ...... RM
ERR055 ...... RN
ERR058 ...... RR
ERR091 ...... RS
ERR091A ..... RS

IJBOTH ...... RK
Beginning of $LNKEDT2 phase. This phase processes TXT, REP, RLD, and END cards.

MOVPER ...... RQ

OTHINC ...... RK
OTHTYP ...... RK
Loop control label.

REPROC ...... RM
Beginning of the REP card processor.
REPTXT ...... RM
Loop control label.
RLBYWR ...... RP
RLCONS ...... RN
RLDPRC ...... RN
Beginning of RLD pass 1 processing.
RLRET ....... RN
RLSTP ....... RN
RLSW1 ....... RN
Program switch set and reset within $LNKEDT2 phase. Setting (NOP or branch) determines if the R and P pointers are to be processed. Several RLD items can have the same R and P pointers. Only the first set of identical R and P pointers are processed.
RLWRIT ...... RP
Program switch set to NOP in the pass 1 RLD processor when the RLD is to be processed in the pass 2 RLD processor. If the switch is set to branch (initial condition), the RLD is ignored.

TSTESD ...... RQ
Reference for execute instruction.
TXTALL ...... RL
Moves test to the work area and provides an exit for the text processor.
TXTGET ...... RL
Loop control label.
TXTPRC ...... RL
Beginning of the text processor.

WRST01 ...... RS
Label of the instructions that perform I/O on SYS001 when SYS001 is a tape unit.
WRS001 ...... RS
Beginning of the write SYS001 subroutine.

Phase 4 $LNKEDT4, Charts RT-RZ

| Label | Chart |
|---|---|

CHKRP ....... RU
CHKRPN ...... RY
CMDEL ....... RZ
Beginning of a subroutine that checks for a comma.
CRDEND ...... RY

DECDSP ...... RX
Beginning of decimal displacement processing for the operand of a phase card.
DELEXT ...... RW
DSPRTN ...... RW
Beginning of displacement operand processing.

ENTALK ...... RY
Provides an exit from the $LNKEDT4 phase of the linkage editor when a blank ENTRY card is found.
ENTCRD ...... RV
Beginning of the entry card processor.
Note: At this time, a blank operand field would have been detected in the skip blanks subroutine. (See SKIPB and ENTALK in the label list.)
ENTPRT ...... RV
ERRNML ...... RT
Program switch initialized to branch. However, if an error occurs on an include card and the include card has created a name list, ERRNML is set to NOP. This allows the name list to be cleared prior to card error processing.
ERR010 ...... RT
ERR012 ...... RZ
ERR013 ...... RZ
ERR014 ...... RU
ERR015 ...... RY
ERR025 ...... RW
ERR033 ...... RU
EXTRCT ...... RZ
Beginning of the extract subroutine. Used to put the operands into a work area.

FINDEL ...... RZ
Beginning of a loop to find the delimiter that is adjacent to an operand.
FINDND ...... RU
FNDDEL ...... RY
Tests for a qualifier in the operand.
FONDEL ...... RZ

GETVRB ...... RT

IJBSCN ...... RT
Beginning of the $LNKEDT4 phase, the first control card processing phase.
INCCRD ...... RU
Beginning of the include card processor.

LABST ....... RU
Builds the name list of control sections from the operand field of an include card. These control sections are subsequently used to build a phase (see NMELST).
LKQUO ....... RW

NAUTO ....... RY
Tests for a NOAUTO option.
NTABS ....... RY

PHCRD ....... RW
Beginning of the phase processor in the $LNKEDT4 phase. This part of the phase processor performs two basic functions:
1. Determines which optional operand has been used.
2. Validity checks the phase card image.

QUAPRO ...... RY
Beginning of qualifier processing for the operand of a phase card.

RESET ....... RZ
RUNNING ..... RW

SAVCTL ...... RV
Provides exit to the autolink processor (Chart QQ). Saves the disk address of the control card before exiting. (See $LNKEDT label list entry ALNKPR.)
SEEBLK ...... RT
SETMDS ...... RU
SKIPB ....... RZ
Beginning of the skip blanks subroutine.
STORR5 ...... RX
SUBMOD ...... RU

TSTLIM ...... RX
TSTNEG ...... RX

UPDATE ...... RZ

Beginning of a subroutine to point to the next character in the input stream.

Phase 5 $LNKEDT6, Charts SA-SG

Label          Chart

ALKERR ...... SG
ALKFND ...... SG
CHEKQU ...... SC
CHQUAL ...... SC

CIMBLK ...... SE
Phase size check.
CINOBL ...... SE
Number of core image blocks required by this phase is equal to number of bytes loaded divided by block size. Add one block for any remainder.

ERR020 ...... SB
ERR021 ...... SB

ERR022 ...... SC

ERR023 ...... SA

ERR024 ...... SD

ERR025 ...... SB
ERR031 ...... SG
ERR081 ...... SA
ERR082 ...... SF
ERR092 ...... SE

IJBCTL ...... SA
Beginning of $LNKEDT6 phase (control card terminal processing).
INCERR ...... SG
INCFND ...... SG
INCGET ...... SG
Entry point to the terminal include card processor when an autolink is in progress.
INCLOP ...... SG
Beginning of the relocatable directory scan used for autolink, include, and terminal processing. The scan looks for a module with a name that matches the card image name field.
INCLPR ...... SG
Beginning of include card terminal processing.
INCRED ...... SG
ISDISP ...... SD
Processing when the phase card operand is a displacement.
ISROOT ...... SD
Processing when the phase card operand is ROOT.

LABINV ...... SA

NEWPHS ...... SF
Beginning of processing for the first phase card.
NOT1ST ...... SB
NODUPL ...... SC
NTROOT ...... SC

PHSPRC ...... SB
Phase name processing.
PHXADD ...... SA

SCNSYM ...... SC
SCSYM1 ...... SC

SETPHS ...... SA

WRPHCD ...... SA
WRTHDR ...... SA
WRTRFR ...... SA
Saves the transfer address.

Phase 6 $LNKEDT8, Charts SH-SL

This phase is the linkage editor MAP processor. Figure 82 is a sample MAP printout.

Label          Chart

ADMDSW ....... SH
Program switch to indicate first time through the MAP processor.
CLRCMN ....... SH
CNVBIN ....... SL
Beginning of subroutine to convert binary to hexadecimal.
CNVLOP ...... SL
COMCHK ...... SH
Beginning of common (CM) processing in the MAP processor. The cumulative length of discretely named commons is calculated at this point in the program.
CSCAN ....... SJ
End of control dictionary test.

DUPLAB ...... SK
Tests for duplicate label.

ERR085 ...... SK
ESIXTA ...... SH
ESIXTY ...... SH
Sets a switch when a transfer label is found.
EXECWR ...... SK
I/O for error.
EXTNLP ...... SK
EXTNSW ...... SK
EXTSCN ...... SK
End of control dictionary test.

FCHRLD ...... SK
Entry point when NOMAP option is found.

IJBMAP ...... SH

Beginning of the $LNKEDT8 phase of the MAP processor.

LDRGO ....... SJ
LDRSCN ...... SJ
End of control dictionary scan.
LDRTSD ...... SJ

MAPCST ...... SJ
Sets up for a scan of the control dictionary, searching for control sections belonging to the phase previously identified in this routine.
MAPHAS ...... SJ
Beginning of a control dictionary scan searching for phase entries.
MAPHNM ...... SJ
Sets up MAP print area with the proper phase name.
MAPLDR ...... SJ
Sets up for scan of control dictionary searching for LD/LRs.
MPLDSW ...... SJ

NOTRFR ...... SH
OVRLSW ...... SK
Tests for overlay of root phase.

PHADMD ...... SJ
Adjusts load address of the phase by the cumulative length of the discretely named commons.
PHSCAN ...... SJ
End of control dictionary test.
PHSTOR ...... SH
Reinitializes the phase information in location PHEADR (see common label section).
PREXTN ...... SK
PRTMAP ...... SH

SCNCMN ...... SH
End of control dictionary test.
SETJCS ...... SL
Sets Job Control flag to show good linkage editor output.

TERSXY ...... SK
TRYROT ...... SJ
Tests for root phase prior to setting up root message in MAP print area.

UNRSPC ...... SK

```
JOB   BURHANS    03/02/66   DISK LINKAGE EDITOR DIAGNOSTIC OF INPUT

ACTION TAKEN     MAP
LIST             PHASE    $LNKEDT,S,NOAUTO               OVERHEAD/INITIALIZATION     OHD 0003
LIST             INCLUDE ,%IJBLNK10,IJBINL10¤                                        OHD 0005
LIST             PHASE    $LNKEDT0,IJBINL,NOAUTO         ESD/SYM                     OHD 0007
LIST             INCLUDE ,%IJBESD10¤                                                 OHD 0009
LIST             PHASE    $LNKEDT2,IJBINL,NOAUTO         TXT/REP/RLD/END             OHD 0011
LIST             INCLUDE ,%IJBOTH10¤                                                 OHD 0013
LIST             PHASE    $LNKEDT4,IJBINL,NOAUTO         CONTROL CARD SCANNER        OHD 0015
LIST             INCLUDE ,$IJBSCN10¤                                                 OHD 0017
LIST             PHASE    $LNKEDT6,IJBINL,NOAUTO         PHASE/INCLUDE POST PROCESSOOHD 0019
LIST             INCLUDE ,%IJBCTL10¤                                                 OHD 0021
LIST             PHASE    $LNKEDT8,IJBLOV,NOAUTO         MAP                         OHD 0023
LIST             INCLUDE ,%IJBMAP10¤                                                 OHD 0025
LIST             PHASE    $LNKEDTA,IJBLOV,NOAUTO         PASS 2 - RLD RESOLUTION     OHD 0027
LIST             INCLUDE ,%IJBRLD10¤                                                 OHD 0029
LIST    ENTRY
```

```
03/02/66    PHASE    XFR-AD   LOCORE   HICORE   DSK-AD    ESD TYPE   LABEL      LOADED  REL-FR

         $LNKEDT    002170   001800   002867   22 1 2    CSECT      IJBLNK10   001800 -000100
                                                         * ENTRY    IJBLNK     001800
                                                         * ENTRY    IJBLOV     001F20

                                                         CSECT      IJBINL10   002170 -000100
                                                         * ENTRY    IJBINL     002170
                                                           ENTRY    IJJCPD3    002848
                                                           ENTRY    IJJCPD1    002848

         $LNKEDT0   002170   002170   0025CF   22 3 1    CSECT      IJBESD10   002170 -0007F8
                                                         * ENTRY    IJBESD     002170
                                                         * ENTRY    IJBESDND   0025D0

         $LNKEDT2   002170   002170   002667   22 3 2    CSECT      IJBOTH10   002170 -000C58
                                                         * ENTRY    IJBOTH     002170
                                                         * ENTRY    IJBOTHND   002668

         $LNKEDT4   002170   002170   00266F   22 4 1    CSECT      IJBSCN10   002170 -001150
                                                         * ENTRY    IJBSCN     002170
                                                         * ENTRY    IJBSCNND   002670

         $LNKEDT6   002170   002170   002497   22 4 2    CSECT      IJBCTL10   002170 -001650
                                                         * ENTRY    IJBCTL     002170
                                                         * ENTRY    IJBCTLND   002498

         $LNKEDT8   001F20   001F20   002557   22 5 1    CSECT      IJBMAP10   001F20 -001BC8
                                                         * ENTRY    IJBMAP     001F20
                                                         * ENTRY    IJBMAPND   002558

         $LNKEDTA   001F20   001F20   0024EF   22 5 2    CSECT      IJBRLD10   001F20 -002200
                                                         * ENTRY    IJBRLD     001F20
                                                         * ENTRY    IJBRLDND   0024F0
```

Figure 82. Linkage Editor Map

Phase 7 $LNKEDTA, Charts SM-ST

Label Chart

ABORT ....... SQ
Beginning of non-recoverable error handling subroutine.

BLKHDR ...... SR
Beginning of a routine to build 20-byte core image directory headers.
BLKLOP ...... SR
Beginning of a loop to read 28-byte phase headers from the system work area, and to build 20-byte core image directory headers.

CANCLE ...... SQ
CLOSE ....... ST
Beginning of a subroutine to close SYSLNK. To perform a close operation, the DTF information is required. The DTF table was stored during initialization as the first record on SYS001. If SYS001 is a disk, the DTF information can be directly accessed. If SYS001 is a tape, the close subroutine must backspace to the first tapemark, and then backspace to the beginning of the first record in order to get the DTF information. Backspace file - backspace record - read DTF - close SYSLNK is the I/O sequence for tape.

EXCP01 ...... ST
Backspace either file or record subroutine.

IJBRLD ...... SM
Beginning of the $LNKEDTA phase, pass 2 RLD processor.
ISCLOS ...... ST
Issues close to SYSLNK.

RADD4 ....... SM
RDOK01 ...... ST
RDS001 ...... ST
Beginning of the read from SYS001 subroutine. Used primarily to get the pass 1 RLD information.
RDTP01 ...... ST
RECF01 ...... ST
Updates the disk address after the I/O operation if SYS001 is a disk device.
REP001 ...... ST
Beginning of a subroutine to reposition SYS001.
RESDCN ...... SP
RLADCN ...... SP
RLCTER ...... SN
Counts the number of unresolved ADCONS.
RLDCON ...... SP
Beginning of pass 2 RLD constant processor.
RLDOP ....... SM
RLDOR ....... SN
RLDRAG ...... SM
Reads RLD information supplied by pass 1 from SYS001.
RLDRET ...... SM
Common entry point used during pass 2 whenever an RLD item has finished processing. A test is made for more RLD items on the card image followed by a test for more RLDs on SYS001.
RLDSW1 ...... SM
Program switch set to branch within this phase, whenever pointer processing is finished.
RLDSW2 ...... SN
Program switch initialized to branch, calculates load address (assembled origin of control dictionary entry plus relocation factor) when set to NOP in this phase.
RLDSW3 ...... SP
Initialized to NOP. Set to branch within this phase whenever R and P pointers point to wrong phase.
RLDSW4 ...... SP
Program switch initialized to NOP indicating the ADCON is to be extracted from the core image block. If the switch is set to branch, the ADCON is to be replaced in the core image block.
RNXTRN ...... SN
Tests for unresolved ADCON.
ROTSID ...... SP
Counts ADCONS outside the phase limits.

TPREAD ...... ST
TSTCNT ...... SQ
Tests to determine if any error diagnostic information is available. If error diagnostic information is found, it is converted to unpacked decimal and printed on the MAP.
TSTSID ...... SQ
Sets up MAP information (number of ADCONS outside the phase limits) in a test register. If the register is zeroed, there is no MAP information. If the content is nonzero, MAP information will be printed.
TSTUNR ...... SQ

VERLOP ...... SR
Beginning of a loop to read and verify all core image blocks written by linkage editor. All verification occurs at this point rather than after each individual write operation.

WRTLOP ...... SR
Beginning of a loop to write the 20-byte core image directory headers in the directory.

Phase 8 $LNKEDTC, Charts SU-SW

| Label | Chart |
|---|---|
| BEGINN | SU |
| CATSUP | SV |
| CHKSUP | SV |
| CHKTAG | SW |
| CHKWA | SU |
| CHKWAR | SU |
| CHLENT | SV |
| CNCMSS | SU |
| COSTRT | SV |

Program switch. Branch equal instruction until the last phase entry in the system work area is detected (* in the 1st byte). At this time, it is changed to a NOP.

| | |
|---|---|
| DIRSCN | SV |
| EOBWA | SW |
| GOONL | SW |
| INCRWR | SW |
| KWITT | SW |
| LCANCL | SU |
| LSTPH | SU |
| MODLR | SV |
| MVEWA | SW |
| MVEWB | SV |
| NBLOCK | SV |
| NEXCID | SU |
| NODRUP | SV |
| NOMTCH | SV |
| OUTWA | SU |
| PHNMSS | SU |
| READWA | SV |
| REPLRE | SV |
| RSYSDR | SW |
| SETADDR | SU |
| START | SU |
| WCIDRC | SW |
| WRTSYD | SW |

## LIBRARIAN MAINTENANCE PROGRAMS (SECTION 6)

### Common Library Maintenance Program (MAINT), Charts TA-TH

## Core Image Library Maintenance Program (MAINTC2), Charts TJ-TK

Label Chart

DELETE ...... TJ

ERRMES ...... TK
 Program switch. Set to NOP if no match to a program (first four characters of operand) is found in the CI directory. Set to branch if a match is found.
ERRME1 ...... TK
 Program switch. Set to NOP if operation is delete. Set to branch if operation is rename and a scan for the new name is performed.
ERRPHS ...... TK
 Program switch. Set to NOP if operation is delete. Set to branch equal if operation is rename and a scan for the new name is performed.
ERRPRS ...... TJ

GONCOM ...... TK

INCR ........ TK
 Entry to subroutine to continue scanning the CI directory from the point where a match has been found.

MREOPM ...... TJ
MREOPN ...... TJ

NERR1 ....... TJ
NINLIB ...... TK

OPALL ....... TJ

RNXTBL ...... TJ

SCNALL ...... TJ
SCNCID ...... TK
SCNCIN ...... TK
SKPOND ...... TK

## Relocatable Library Maintenance Program (MAINTR2), Charts TL-TX

Label Chart

ACATAR ...... TL
ADELER ...... TL
AINIT ....... TL
AINITC ...... TL
AINITO ...... TL
AINITP ...... TL
AINITT ...... TL
AREA1
 Refer to Figure 65. ESID save area.
AREA2
 Refer to Figure 65. ESID save area.

AREA3
 Refer to Figure 65. ESID save area.
AREA4
 Refer to Figure 65. ESID save area.

ARENAR ...... TL

BUFCCD
 Listing Only: Input area for records from SYSIPT (160 bytes).
BUFFER
 Listing Only: Output area for physical blocks to be written in the relocatable library on SYSRES (322 bytes).
BUFREC
 Listing Only: Area where one ESD, TXT, or RLD logical record for the relocatable library is assembled (160 bytes).

CANCEL ...... TL, TM, TX
CATALR ...... TM
CATNEW ...... TM
CATNMD ...... TM
CATPFD ...... TM
CATREP ...... TW
CATRIP ...... TM
CATRIQ ...... TM
CATRLB ...... TN
CATSER ...... TN
CATSIC ...... TN
CATSYE ...... TM
 Error message 3M52I: RELOCATABLE DIRECTORY IS FULL.
CATSYR ...... TX
 Error message 3M53I: RELOCATABLE LIBRARY IS FULL.
COMCAT ...... TN
COMCWD ...... TN

DEERNM ...... TP
 Error message 3M33I (module name): NOT IN LIBRARY.
DELCMP ...... TQ
DELET ....... TP
DELETA ...... TP
DELETM ...... TP
DELETO ...... TP
DELINA ...... TQ
DELNER ...... TP
DELNEX ...... TQ
DELPAL ...... TQ
DELPFD ...... TQ
DELPFX ...... TQ
DELPND ...... TQ
DELPNE ...... TQ
DELSW ....... TQ
 Switch is set when an entry is deleted in the RL directory block being worked on. Only blocks being changed will be written on SYSRES.
DRCTRY
 Listing Only: I/O area for RL directory blocks (322 bytes).

ENDERR
Listing Only: EOFADDR specified in DTF for SYSIPT. Error message 3M34I: EOF ON SYSIPT - END STATEMENT MISSING.

ENTMAI ...... TA
Entry in MAINT root phase to get next operand or next control statement from SYSRDR or SYSIPT when all operands are processed.

ERRRTN ...... TF
Label of the error routine. The address of this routine is ERRREG ( register 9).

INVCRD ...... TN
Error message 3M11D: INVALID CARD IN MODULE.

INVOPD ...... TW
Error message 3M21I: INVALID OPERAND IN CONTROL STATEMENT.

NEWRD ....... TA
Reads next control card from SYSRDR or SYSIPT. A module has been bypassed on SYSIPT because of an error.

NOCYL ....... TL
Error message 3M43I: NO RELOCATABLE LIBRARY.

NOTIN ....... TW
Error message 3M33I (module name): NOT IN LIBRARY.

NXTOPR ...... TE
Entry in MAINT root phase to extract the second operand of a RENAME control statement.

RCESD ....... TR
RCESDB ...... TR
RCESDC ...... TR
RCESDR ...... TR
RCESDS ...... TR
RCESDT ...... TR
Compares to see if ESID numbers in input records are in sequence. If not, a new library record must be started.

RCESDU ...... TR
RCESLC ...... TR
Figure 65 is a table showing how AREA1, AREA2, AREA3, and AREA4 (ESID save areas) are set up on Chart TS when the ESD output record is full and there are more entries to be moved from the input record. Refer to Figure 65 to aid in finding the ESID number for this and the next output record.

RCESMC ...... TT
One or two must be the number of entries moved to the output ESD record from the input card at this point in the program because ESD input cards have a maximum of three entries. Refer to Figure 65.

RCESSW ...... TT
RCESTE ...... TT
RCESWB ...... TT
RCESWR ...... TR
RCMVB1 ...... TS
RCMVB2 ...... TS
RCMVB3 ...... TS
RCMVB4 ...... TS
RCMVB5 ...... TS
RCMVB6 ...... TS
RCRLD ....... TU
RCRLDB ...... TU
RCRLDC ...... TU
RCRLDE ...... TU
RCRLDL ...... TU
RCRLDM ...... TU
RCRLDN ...... TU
RCRLDR ...... TU
RCRLDS ...... TU
RCRLDU ...... TU
RCTXT ....... TV
RCTXTB ...... TV
RCTXTC ...... TV
RCTXTR ...... TV
RCTXTS ...... TV
RCTXTU ...... TV
RCTXTV ...... TV
RCTXTW ...... TV
RCWRBK ...... TX
Subroutine used to write blocks in the relocatable library on SYSRES.

RCWRBL ...... TX
RCWRBM ...... TX
RCWRBN ...... TX
RCWSUR ...... TR
RENAME ...... TW
RENCON ...... TW
RENCSO ...... TW
RENCST ...... TW
RENERN ...... TW
Error message 3M54I (module name): ALREADY IN LIBRARY.

RENERO ...... TW
RMAINT ...... TL
Initial entry to MAINTR2 is RMAINT to catalog, RMAINT+4 to delete, or RMAINT+8 to rename.

RSKIPT ...... TX
Routine used to skip records on SYSIPT to the end of the module.

RSKIPU ...... TX
END statement on SYSIPT indicates that the end of the module has been reached. SYSIPT is positioned at the record which follows the END statement.

SEARCH ...... TW
Subroutine used to find a name in the relocatable directory.

SEARRD ...... TW
SRCCMP ...... TW
SYSDIR
Listing Only: I/O area for system directory record number 2 (80 bytes).

UPDAT ....... TX
Subroutine used to update disk address.

UPDAT1 ...... TX
UPDATE ...... TX
UPDEND ...... TX

ZNMNFD ...... TW
ZNMNON ...... TW

Source Statement Library Maintenance Program (MAINTS2), Charts UA-UN

Label Chart

ALLTHT ...... UJ
Starting address of the all through processing routine.
ALOERR ...... UM

BEGADD
Listing Only: Register 2 contains the operand length from the MAINT root phase.
BERR1 ....... UH
BKCMPR ...... UD
BKCPRS ...... UE
BKNDCK ...... UE
BKNDPR ...... UH
Routine used to check the book-end header. It is not written into the source statement library.
BKOSE1 ...... UA
BKOSET ...... UA
BKOTST ...... UD
BKWAIT ...... UC
Entry into I/O routine to read first card of book.
BNDERR ...... UH

CATALS ...... UA
Starting address of the catalog routine.
CATENT ...... UA
The initial entry to MAINTS2 for catalog operation.
CDCTPR ...... UH
CDNDT1 ...... UF
CDVSW ....... UJ
CLGETB ...... UB
CMXBLK ...... UJ
CNTERR ...... UG
CPRDPR ...... UH
CTLCHT ...... UD
CTLBY1
Listing Only: Switch used to indicate whether initialization has been done. Switch is initially on, turned off when initialization is done.

DCSW
Listing Only: Switch to delete a book from the source statement library.
DECRLG ...... UH
DELE1 ....... UN
DELE2 ....... UN
DELENT ...... UA
The initial entry to MAINTS2 for a delete operation.
DELERR ...... UN
Subroutine that issues error message for any delete errors.
DELET ....... UK
DELETS ...... UA
Starting address of the delete book routine.

DINIRT ...... UA
DIRUP ....... UK
System directory update routine.
DIRDI1 ...... UK
DLINC ....... UJ
DNEWT ....... UK
DNTCHT ...... UK
DOUCAL ...... UJ
DOURST ...... UJ
DOUTST ...... UJ
DOVERR ...... UL
DRENM1 ...... UK
DRPROC ...... UK
DTSTC ....... UL

EODO1 ....... UL
EODIRO ...... UL
ERRBY
Listing Only: Current error switch.
ERRSEQ ...... UC

FILOU ....... UF
Output buffer full switch that branches to the address contained in register 10. The address is FILOU1 until the output block is full. When the output block is filled, the branch address is changed to RELDIN, which is a save area for the remainder of the card in compressed form.
FILOU1 ...... UF
FINENT ...... UJ
Entry to MAINTS2 to set a switch indicating that the system directory is updated and written back on SYSRES.
FINSHS ...... UJ
(See FINENT in this section.)
FLGBLN ...... UF
FLGINC ...... UF
On a compressed card, the high order 4 bits of column two contain the length, in bytes, of the following non blank field.
FNDOP ....... UE
FRESET ...... UJ

GETBKN ...... UM
GET1CE ...... UD
Logical IOCS used to read input from SYSRDR.
GETNNM ...... UB

INIDAD ...... UM
INITBK ...... UC
Two instructions are assembled at this location. When the program is loaded, the branch instruction is effective. The first time the branch is executed, it is overlaid with a wait that is effective thereafter.
INITS ....... UM
Entry to the initialization routine that is used to read the system directory into storage from SYSRES.

LASLI2 ...... UM
LASLID ...... UM
Subroutine that checks the book name

operands for length and determines that the first character is alphabetic.
LCDPR1 ...... UG
Entry to last card processing routine if last card was a MEND card.
LCDPRC ...... UG
Entry to last card processing routine if last card was a BKEND card.
LCDSW1 ...... UF
LCDTST ...... UE
LCDTSW ...... UE
LCDXIT ...... UG
LDCDND ...... UE
LDSCAD ...... UE

MVBCAT ...... UB
The starting address of a section of the CATALS routine that is common to all the routines. It is used to get the book name from the operand field of the control card.
MVBKN1 ...... UB
MVBNMC ...... UM
Subroutine that saves the book name of the book to be cataloged.
MVSTMT ...... UJ

NDLERR ...... UC
NMCK ........ UA
NNM2 ........ UB
NOBHDR ...... UE
NOCPRS ...... UF
NOFULL ...... UF
NRELDI ...... UF

OFULLB ...... UF
Output buffer full switch that branches to the address contained in register 10. The address is NOFULL until the output block is full. When the output block is filled, the branch address is changed to NRELDI, which is a test for the end of valid input data in compressed form.
OPRED1 ...... UN
OPRED2 ...... UN
OPREI1 ...... UB, UM, UN
OPRERS ...... UN
OPRERT ...... UN
Subroutine that determines if an I or D type message is to be issued and then issues the message.
OPRRTN ...... UH
OPRSCN ...... UH

PARLGT
Listing Only: Register 1 contains the operand length from the MAINT root phase.

RELDIN ...... UF
RENAMS ...... UB
Starting address of the rename a book routine.
RESET ....... UJ
RNMENT ...... UB
Initial entry to MAINTS2 for a rename operation.
RSETSW ...... UA
RSTORC ...... UH
RSTSWS ...... UG

SBLMSG ...... UA
SCNBLK ...... UK
SCNBLN ...... UF
SCNCD ....... UF
SCNMF ....... UE
SEQNPR ...... UH
SEQSW ....... UC
SETCTM ...... UG
SLNMC2 ...... UA
SLNMCK ...... UB
SNETST ...... UB
SPLPRT ...... UJ
SPRTT ....... UJ
STEPB1 ...... UK
STEPB2 ...... UL
STEPL1 ...... UK
STEPL2 ...... UL
STEPL5 ...... UL
SWBOA ....... UC

TSTD5 ....... UL
TSTOP ....... UK

UPSTAT ...... UG
UPSYSD ...... UJ

WAIO1 ....... UC
WAITBI ...... UC
Entry into I/O input control
WAITBO ...... UC
Starting address of the I/O output routine.
WRBLK ....... UL
WRCMPL ...... UK
WRDIR ....... UJ
WRTE ........ UJ
WRVRD ....... UK

## System Reallocation Program (MAINTA), Charts VA-VM

Label Chart

ALLERR ...... VC
Displays error message 3M62I: TRACKS FOR DIR EXCEED CYL FOR LIB.

BLDTB ....... VD
BLNKD1 ...... VG
BLNKD2 ...... VG

CANCEL ...... VA, VB, VC, VD, VF, VL
CCDERR ...... VA, VB, VL
Displays error message 3M21I: INVALID OPERAND in ALLOC CTRL STMNT.
CDOSA
Listing Only: Beginning of reallocation tables. Refer to Figure 58 for an example of one of the three tables at

this label.
CLOSA
Listing Only: Beginning of CI library reallocation table. Refer to Figure 58.
CLSWT ....... VA
CNVRTK ...... VB
COMPAR
Listing Only: The second operand of this compare instruction is a one during pass 1 on Chart VJ and a two during pass 2 on Chart VJ. A compare is made to the update code in the reallocation table.
COMPA1 ...... VM
COMPA2 ...... VM
CONVRT ...... VL
Subroutine to convert the fields in the ALLOC control statement from decimal to hexadecimal.
COPYLB ...... VM
Subroutine used to save the first track of the label cylinder in the system work area and to restore this track to the relocated label cylinder after reallocation.
CYL ......... VB

DIRERR ...... VD
Displays error message 3M63I: xxDIRECTORY ALLOCATION IS TOO SMALL.
DIRUP ....... VH
DIRUP1 ...... VH
DIRUP2 ...... VH
DIRUP3 ...... VH
DIRUP4 ...... VH
DIRUP5 ...... VH
DIRUP6 ...... VH
DIRUP7 ...... VH
DIRUP8 ...... VH
DIRUP9 ...... VH
DIRUP10 ..... VH
DSKERR ...... VC, VG, VH, VJ, VK
Displays error message 3M50I: POTENTIAL DISASTER ERROR. REBUILD SYSTEM.
DSPLN ....... VE
DSPL1 ....... VE
DSPL2 ....... VE
DSPL3 ....... VE
DSPL4 ....... VE

END ......... VK
ENDJOB ...... VK

GETAL ....... VB
GOON1 ....... VC

INTL1 ....... VD
INTL2 ....... VD
INTL3 ....... VD
INVALL ...... VD

LABEL2 ...... VK
LIBERR ...... VD
Displays error message 3M64I: xx LIBRARY ALLOCATION IS TOO SMALL.

MAINTA ...... VA
MOV ......... VB
MOVE ........ VJ
MOVE1 ....... VJ
MOVE2 ....... VJ
MOVE3 ....... VJ
MOVE4 ....... VJ
MOVE5 ....... VJ
MOVE6 ....... VJ
MOVE7 ....... VJ
MOVE8 ....... VJ
MOVE9 ....... VJ
MOVE10 ...... VJ
MOVE11 ...... VJ
MOV1 ........ VB
MOV2 ........ VB
MVALDC ...... VL

NEXTAL ...... VC
NXTL ........ VL

QUIT ........ VC, VG, VH, VJ, VK
An SVC 7 is issued on a non-existent CCB.

RDOSA
Listing Only: Beginning of relocatable directory reallocation table. Refer to Figure 59.
RDSYSD ...... VC
RLOSA
Listing Only: Beginning of relocatable library reallocation table. Refer to Figure 59.
RLSWT ....... VA
RTRN ........ VM

SDOSA
Listing Only: Beginning of source statement directory reallocation table. Refer to Figure 59.
SLIB ........ VL
SLOSA
Listing Only: Beginning of source statement library reallocation table. Refer to Figure 59.
SUBIT ....... VM
SUBIT2 ...... VM

TKCOMP ...... VM
Subroutine used to increment or decrement the disk address by a displacement in register 3 (DISREG).
TKFMT ....... VK
TKFMT1 ...... VK
TKFMT2 ...... VK
TKFMT3 ...... VK
TKFMT4 ...... VK
TKFMT5 ...... VK
TKFMT6 ...... VK
TKRETN ...... VM
TRCKS ....... VB
TSTB ........ VB
TSTNUM ...... VL
Subroutine to test the fields of the ALLOC control statement. Each character must be numeric (0-9).
TYPEL ....... VA

UPDATE ...... VL

Subroutine used to increment an address to the next character of the ALLOC control statement.

UPD1 ........ VB
UPD2 ........ VB
UPSYSN ...... VF
UPSYS1 ...... VF
UPSYS2 ...... VF
UPSYS3 ...... VF

WRITE ....... VG
WRITE1 ...... VG
WRSYSD ...... VG

XTNERR ...... VF
Displays error message 3M65I: ALLOCATION EXCEEDS SYSRES EXTENT.

## Library Condense Program (MAINTCN), Chart VN-VT

| Label | Chart |
|---|---|
| AUTOCN ...... | VR |
| BLDCCW ...... | VQ |
| CANNOT ...... | VR |
| CHAST ....... | VP |
| CHBLNK ...... | VP |
| CHDEOB ...... | VQ |
| CHGCCW ...... | VT |
| CHGON ....... | VT |
| CHKEOB ...... | VQ |
| CHKOFL ...... | VT |
| CH2 ......... | VT |
| CH3 ......... | VT |
| CH4 ......... | VT |
| CNDCL ....... | VN |
| CNDRL ....... | VN |
| CNRSL ....... | VN |
| DISERR ...... | VS |
| DSMES ....... | VS |
| ENDROU ...... | VR |
| FTCHMA ...... | VR |
| FTCHME ...... | VR |
| GOWRT1 ...... | VQ |
| GOWRT2 ...... | VR |
| ICRDAD ...... | VT |
| ICREND ...... | VT |
| ICRNOP ...... | VT |
| ILLEC ....... | VR |
| IMUP ........ | VQ |
| INCRID ...... | VP |
| INCRIL ...... | VQ |
| IODISK ...... | VS |
| MAICON ...... | VP |
| MODHR ....... | VP |
| MODOFL ...... | VT |
| MON ......... | VT |
| MVECHN ...... | VQ |
| MVREC ....... | VP |
| MVREC1 ...... | VP |
| MVSLCD ...... | VR |
| MVSLCN ...... | VQ |
| NOCND ....... | VR |
| NTATRK ...... | VQ |
| NXTTRA ...... | VN |
| RDDIR ....... | VP |
| RDLIB ....... | VQ |
| RESTNA ...... | VR |
| SKIPWR ...... | VT |
| SWBYTE | |

*Listing Only*: Bit 0 of CI LIB is condensed. Bit 7 of SS LIB is being condensed.

TESTAU ...... VR
TRNOFF ...... VS

WIRTDR ...... VQ
WRTEDR ...... VS

## Set Condense Limits Program (MAINTCL), Chart VU

ADDRA ....... VU

BEGINN ....... VU
Starting address of the MAINTCL program.
BLOPER ...... VU

CRDDNE ...... VU

GENPRO ....... VU
Test the operand for the correct length; then put the limits into binary to be written into the System Directory.

MAINTCL ..... VU
Program used to write the automatic condense limits into the Source Statement Library

OPRPRS ...... VU

PROCCL ...... VU
PROCRL ...... VU
PROCSL ...... VU

## Update Transient, Library Routine, and Foreground Directories Program ($MAINEOJ), Charts VV-VZ

| Label | Chart |
|---|---|
| AUNOP | VW |
| AUTONO | VX |
| BEGINN | VV |
| BLGOON | VX |
| BRIFRL | VV |
| BRTLB | VW |
| COLMN | VV |
| COMDOL | VX |
| DSERV | VX |
| ENDJCE | VX |
| ENDTBL | VW |
| ENTCOM | VX |
| EOJROU | VX |
| GNLR | VY |
| GNOP | VZ |
| GNTP | VZ |
| GNTRA | VY |
| INCRPT | VW |
| INCRTR | VX |
| INCRTT | VX |
| MESLIB | VY |
| MESOP | VZ |
| MESTP | VZ |
| MESTRA | VY |
| NOWRT1 | VX |
| NOWRT2 | VX |
| NOWRT3 | VX |
| NOWRT4 | VX |
| PROLIB | VY |
| PROOP | VZ |
| PROTEL | VZ |
| PROTRA | VY |
| RDCID | VX |
| REIPL | VX |

A new supervisor has been cataloged. The system must be started over with the IPL procedure.

| Label | Chart |
|---|---|
| RELD | VZ |
| RLSL | VW |
| SYSDIR | VX |
| TRNON | VW |

## LIBRARIAN ORGANIZATION PROGRAMS (SECTION 7)

### Copy System Program (CORGZ, CORGZ2), Charts WA-WY

#### Phase I CORGZ, Charts WA-WV

| Label | Chart |
|---|---|
| ABORT | WU |
| ALOCAT | WD |
| AROUND | WC |
| ARUD | WC |
| BLDDE | WH |
| BLKLUP | WN |
| BLNKT | WF |
| BUILD | WM |
| CCARD | WB |
| CI | WE |
| CISON | WH |
| CLSWT | WD |
| COMCMP | WF |
| CONVRT | WS |
| CONVRTK | WD |
| COPYRT | WE |
| CORGZ | WA |
| CPYALL | WQ |
| CYL | WD |
| DIRGET | WS |
| DLIMIT | WG |
| DOTCMP | WF |
| DOTF | WH |
| DOTFND | WF |
| END | WL |
| ENDIND | WT |
| EOFT | WB |
| ERETRN | WV |
| ERRORA | WU |
| ERRORB | WU |
| ERRORC | WU |
| ERRORD | WU |
| ERRORE | WU |
| ERROR1 | WU |
| ERROR2 | WU |
| ERROR3 | WU |
| ERROR4 | WU |
| ERROR5 | WU |
| ERROR6 | WU |
| ERROR7 | WU |
| ERROR8 | WU |
| ERROR9 | WU |
| ERRRTN | WV |
| EXCMP | WR |
| EXIT | WL |
| EXP | WT |
| FINDIT | WH |
| FINDRL | WJ |
| FRMDIR | WC |

Phase 2 CORGZ2, Charts WW-WY

Label Chart



LIBRARIAN SERVICE PROGRAMS (SECTION 8)

Directory Service Program (DSERV), Charts XA-XJ

Label Chart

ADD01 ....... XG
ADD02 ....... XG
ADD03 ........ XG

BBB ......... XD

CCC ......... XE
CLEAR ....... XJ
CNT
   Listing Only: 576 is the maximum number of entries in the transient directory.
COBL ........ XE
CONTIN ...... XE
COUNTING .... XC

DATACHEK .... XG
DATERR ...... XG
DDD ......... XE
DECRT ....... XF
DIRTYP ...... XA
DSERV ....... XB
DSPBYT
   Listing Only: Initially X'00'.
   Bit 4 X'08' = TRANSIENT DIR SW
   Bit 5 X'04' = SS DIR SW
   Bit 6 X'02' = RL DIR SW
   Bit 7 X'01' = CI DIR SW
DTFCPPUT .... XB
   Subroutine used to print a line on SYSLST using CPMOD LIOCS module.

ENDTD ....... XC
ENTMAI ...... XA
EOJ ......... XB, XE
EOVLST ...... XB
ERRINV ...... XA
   Displays error message 3D10D: INVALID OPERATION.

FRSTCH ...... XH
   Subroutine used to determine the position and length of the first operand of the DSERV control statements.

GETAD ....... XG
GETADA ...... XG
GETCD ....... XC
GETNXRD ..... XD
GETNXSD ..... XE
GETNXTD ..... XC
GETRD ....... XD
GETSD ....... XE
GETSUT ...... XF
   This is the entry label to the SDMOD LIOCS module when a GET macro is issued in the program.
GETTD ....... XC
GETTD1 ...... XC
GETXIT ...... XF

HDRRD ....... XD
HDRSD ....... XE
HDRTD ....... XJ

INITSHL ..... XH
   Subroutine used to determine the position and length of the operation field in the DSERV control statements.
IORUT ....... XF

LIBRPRES .... XB
   Subroutine used to determine if the relocatable and/or source statement libraries are present.

NEWRD ....... XA
NOFOND ...... XH
NOREAD ...... XA
NOSL ........ XB
NOTBLK ...... XA
NXTOPR ...... XH
   Subroutine used to determine the position and length of the next operand in the DSERV control statements.

OFF ......... XD
OFFSD ....... XE
OPAERR ...... XA
   Displays error message 3D20D: INVALID OPERAND.

RDCONV ...... XD
RDERU ....... XG
READ1 ....... XF
RELSEA ...... XF
RET ......... XB

SCANR1 ...... XH
SCANR2 ...... XH
SDCONV ...... XE
SKIPA ....... XG
SW
   Listing Only: Indicator for directory being displayed.
   0 - not transient directory
   T - transient directory
SYSDIR ...... XB

TDCONV ...... XC
TDWORK
   Listing Only: Input buffer for one 20-byte directory entry.
TEST ........ XF
TESBYT
   Listing Only: Initially X'80' bit 0, X'80' - First control statement switch.
TESTSL ...... XB
TRYCD ....... XB
TRYRD ....... XB
TRYSD ....... XB
TSTLCH ...... XA
TURNOFF ..... XJ

USERSK ...... XG

WLRERR ...... XG
WRHDR ....... XC

XXX ......... XC

YYY ......... XC

ZZZ ......... XC

## Relocatable Library Service Program (RSERV), Charts YA-YL

Label / Chart

AINTIS ...... YA
ALLSW2 ...... YC

CANCEL ...... YA
CHP3 ........ YC
CMPDIR ...... YC
CMP2 ........ YC
CNVORG ...... YJ
CPLSOP ...... YB
CPSLSH ...... YA
CRDSWT
  Listing Only: Program flags.
  Bit 0 = first time switch in RLDPRT routine.
  Bit 2 = 2540 reader/punch.

EDCPGP ...... YE, YG, YH
EDPCBA ...... YE
EDPCCS ...... YE
EDPCIA ...... YE
EDPCIR ...... YE
EDPCLA ...... YE
  The number of entries is computed by dividing by 16 the variable field byte count of the ESD record.
EDPCLI ...... YE
EDPCLT ...... YE
  Only non-LD items are included in the ESID number count.
EDPCNA ...... YE
EDPCRT ...... YE
EDPCSC ...... YE
EDPCSI ...... YE
EOD1 ........ YC
EOMTST ...... YD
EOMTS1 ...... YD
ENDRTN ...... YA
  EOFADDR specified in the DTF for SYSRDR.
ENDRT1 ...... YA
ERILOP ...... YA
  Error message 3R10D: INVALID OPERATION.
ESDPBP ...... YF
ESDPCA ...... YF
ESDPCH ...... YE
ESDPIA ...... YF
ESDPRT ...... YF
ESDPSB ...... YF
ESDPTL ...... YF
ESDPUD ...... YF
EXTK3 ....... YD
EXTRCT ...... YB
EXTRT1 ...... YB

GETCTL ...... YA
  Subroutine to read control cards from SYSRDR.
GETFIA ...... YL
GETFIL ...... YL
GETFLD ...... YL
GETFR1 ...... YL
GETFR2 ...... YL
GETFSC ...... YL
GETFSI ...... YL
GETFSN ...... YL
GETFSR ...... YL
GETFTS ...... YL
GETFWM ...... YL

IGNORE ...... YC
ILOPRD ...... YB
  Error message 3R21I: INVALID OPERAND.
INCRIT ...... YH

NOLIB ....... YA
  Error message 3R43I: NO RELOCATABLE LIBRARY.
NOPCH ....... YD
NOTHR1 ...... YC
  Error message 3R27I: (module name) NOT FOUND.

PCHSUB ...... YK
PCHSWT ...... YA
PPSWT ....... YA
PRTHCM ...... YD
PRTHCN ...... YD
PRTHDR ...... YD
PRTHGS ...... YD
PRTHG1 ...... YD
PRTHG2 ...... YD
PRTHUD ...... YD
PRTSRA ...... YK
PRTSSK ...... YK
PRTSUB ...... YK
PRTSWT ...... YA
PRT1 ........ YD
PSUB2 ....... YK

RDCD ........ YA
RDDISK ...... YK
RDPCAI ...... YH
RDPCBB ...... YH
RDPCCT ...... YH
RDPCS4 ...... YH
RDPCS8 ...... YH
RDPCTS ...... YH
RDRD1 ....... YC
RDRD2 ....... YC
RDRD4 ....... YC
RDSD ........ YA
REINT1 ...... YE, YG, YH
REPPL ....... YJ
  When a replace card is printed, entries are divided by blanks instead of commas.
REPPRT ...... YJ
RLDPBP ...... YJ
RDLPCH ...... YH
RLDPCO ...... YJ
RLDPDR ...... YJ
RLDPHE ...... YJ
RLDPRR ...... YJ
RLDPRT ...... YJ
RLDPSW ...... YJ
RLDPUD ...... YJ
RLDPXA ...... YJ

SS2 ......... YA
SYMPRT ...... YG

TSTDAL ...... YC
TSTPCH ...... YD
TSTPCI ...... YD
TSTPRT ...... YD
TXPCIM ...... YG
TXPCSC ...... YG
TXTPCH ...... YG
TXTPIA ...... YG
TXTPRT ...... YG
TXTPSA ...... YG
TXTPT2 ...... YG

UPDT3 ....... YC

## Source Statement Library Service Program (SSERV), Charts ZA-ZL

**Label** **Chart**

ALLSW ....... ZD
   Switch set if operand is ALL.
ALLTST ...... ZB
ALOTST . ZB

BEOVRT ...... ZK
BKENDO ...... ZD
BKNDOA ...... ZF
BKNDOS ...... ZF
BKNDOU ...... ZF
   Subroutine for header and trailer control.
BKNDO1 ...... ZF
BLNENT ...... ZJ
   Entry into find operand subroutine used to find the first operand on the control statement.

CALLCS ...... ZA
   Entry into SSERV control statement analysis used to read another control statement.
CHNGSB ...... ZB
CLRCO ....... ZC
CMPRST ...... ZD
COSET ....... ZE
CPPCH2 ...... ZE
CPRPCH ...... ZE
CPRSC1 ...... ZA
CPRSC3 ...... ZA
CPRSCN ...... ZA
CPRSW ....... ZC
   Switch set to -FF- if output is compressed.
CSERR1 ...... ZL
CSERR2 ...... ZL
CSERR3 ...... ZL

DIRCK1 ...... ZG

EOBK ........ ZD
EOBKT ....... ZD
EOVLST ...... ZK
   Entry to end-of-volume routine when an EOV condition has been encountered on SYSLST.
EOVPCH ...... ZK
   Entry to end-of-volume routine when an EOV condition has been encountered on SYSPCH.
EXIT ........ ZH
EXP ......... ZC
EXPCD1 ...... ZC
EXPCD ....... ZC
EXPLP ....... ZC
EXPOUT ...... ZE
   Entry into routine that controls the expanded output.
EXPTIN ...... ZC

FNDBK ....... ZG
   Entry to subroutine used to find the book to be serviced in the source statement library.
FNDBK1 ...... ZG
   Entry to find book subroutine.
FNDBK2 ...... ZG
FNDL ......... ZG
   The two instructions at this address are overlayed the first time they are executed (fetching $$BOPNLB). $$BOPNLB is called to find the source statement library. $$BOPNLB replaces the calling instructions with two execute instructions.

GETALL ...... ZC
   Entry point to get next book if the sublibrary qualifier is a period.
GETBK1 ...... ZB
GETOUT ...... ZK

HIBKSW ...... ZL
HIBKT ....... ZD

IBUFPT ...... ZC
INIT1 ....... ZA
   The initial entry point to the SSERV program.

MVBKN ....... ZC
MVBKN1 ...... ZC

NBLENT ...... ZJ
   Entry into find operand subroutine to find the second and all remaining operands.
NBLE1 ....... ZJ
NFERR ....... ZL
NOLIB ....... ZH

PCHESW ...... ZE
PRTCCD ...... ZL
PRTCC1 ...... ZL
PRTNF ....... ZL
PRTNF1 ...... ZL
PRTNF2 ...... ZL
PRTSW ....... ZC
PRTTST ...... ZE
PTRNT ....... ZD

RDBLK ....... ZG

RDBLK1 ...... ZG
RDBLK2 ...... ZD
RDBLK3 ...... ZD
RDRDR ....... ZJ
Entry to read control statement subroutine. Reads a card and checks for EOF.
RELOC ....... ZH
RELOC1 ...... ZH
REPCAL ...... ZH
RESCN ....... ZA
RSTBPT ...... ZG
RSTBP1 ...... ZG
RSTDA ....... ZG
RSTPRT ...... ZE
RTRNT1 ...... ZD
RTURN ....... ZG
RTURN1 ...... ZG

SCNNBL ...... ZJ
SCNTSP ...... ZJ
SKIPBO ...... ZD
SLASHO ...... ZK
Entry into end-of-file subroutine that completes job and calls job control.
SLASHY ...... ZK
SLASH1 ...... ZK
START ....... ZH
Starting address of $$BOPNLB transient, which is used to find the source statement library directory in the SYSRES pack.

TSTCM1 ...... ZF
TSTND ....... ZG
TSTPC1 ...... ZF
TSTQUA ...... ZB
Entry point used to test for more operands on the control statement being serviced.

| | |
|---|---|
| ABS | Absolute |
| ACC | Accumulator |
| ACCT | Account |
| ACT | Actual |
| ADDR | Address |
| ADJ | Adjust |
| ADV | Advance |
| ALG | Algebraic |
| ALL BND | All Bound |
| ALLOC | Allocation |
| ALPHA | Alphabetic |
| ALT | Alternate, Alteration |
| APROX | Approximate |
| ARITH | Arithmetic |
| ASDNG | Ascending |
| ASMBL | Assemble |
| ASGN | Assign |
| ATT | Attention |
| AUX | Auxiliary |
| AVAIL | Availability |
| | |
| BFR | Buffer |
| BI | Binary |
| BKSP | Backspace |
| BLK | Block |
| BLKCNT | Block Count |
| BLNK | Blank |
| BR | Branch |
| BM | Buffer Mark |
| | |
| CALC | Calculate, Calculator |
| CARR | Carriage |
| CC | Card Column |
| CD | Card |
| CHAN | Channel |
| CHAR | Character |
| CHG | Change |
| CHK | Check |
| CHKPT | Checkpoint |
| CLR | Clear |
| CLS | Close |
| CMND | Command |
| CMP | Compare |
| CMPL | Complement |
| CMPRSD | Compressed |
| CNCL | Cancel |
| CNSL | Console |
| CNT | Count |
| COL | Column |
| COMM | Communication |
| COMP | Compute |
| CON | Constant |
| COND | Condition |
| CONT | Continue |
| CONV | Convert |
| CORR | Correction |
| CPLD | Coupled |
| CPSW | Current PSW |
| CTR | Counter |
| CTRL | Control |
| CTRL DICT | Control Dictionary |
| CU | Control Unit |

| | |
|---|---|
| CURR | Current |
| CYL | Cylinder |
| | |
| DCMT | Document |
| DCML | Decimal |
| DEC | Decision |
| DECR | Decrement |
| DEL | Delete |
| DESCG | Descending |
| DEV | Device |
| DIM | Dimension |
| DIR | Directory |
| DR | Drive |
| DSK | Disk |
| DSPLT | Displacement |
| DSPY | Display |
| | |
| ELIM | Eliminate |
| ENT | Entry |
| EOF | End of File |
| EOJ | End of Job |
| EOPSW | External Old PSW |
| EOR | End of Reel |
| EQ | Equal |
| EQUIP | Equipment |
| ERP | Error Recovery Procedure |
| ERR | Error |
| ES | Electronic Switch |
| EXEC | Execute |
| EXT | External |
| | |
| FIG | Figure |
| FLD | Field |
| FLDL | Field Length |
| FLT | Floating |
| FMT | Format |
| FR | From |
| FREQ | Frequency |
| FUNC | Function |
| FWD | Forward |
| FXD | Fixed |
| | |
| GEN | Generator |
| GENL | General |
| GM | Groupmark |
| | |
| HDR | Header |
| HEX | Hexadecimal |
| HI | High |
| HLT | Halt |
| HSK | Housekeeping |
| HYPER | Hypertape |
| | |
| I/O | Input/Output |
| IC | Instruction Counter |
| ID | Identification |
| INCR | Increment |
| IND | Indicate |
| INDN | Indication |
| INDR | Indicator |
| INFO | Information |
| INIT | Initialize |

| | |
|---|---|
| INQ | Inquire |
| INST | Instruction |
| INT | Initial |
| INTERV | Intervention |
| INTRPT | Interrupt |
| INVAL | Invalid |
| IOOPSW | I/O Old PSW |
| IT | Interval Timer |
| IW | Index Word |
| LA | Load Address |
| LBL | Label |
| LD | Load |
| LDG | Leading |
| LGL | Logical |
| LIT | Literal |
| LNG | Length |
| LOC | Location |
| LT | Less Than |
| LTK | Logical Transient Key |
| LTR | Letter |
| LUB | Logical Unit Block |
| MACH | Machine |
| MAX | Maximum |
| MCOPSW | Machine Check Old PSW |
| MIN | Minimum |
| MISC | Miscellaneous |
| MOD | Modification |
| MPXR | Multiplexor |
| MPS | Multiprogramming System |
| MPY | Multiply |
| MSG | Message |
| NEG | Negative |
| NO | Number |
| NUM | Numeric |
| NXT | Next |
| OC | Operator Communication |
| OP | Operation |
| OPN | Open |
| OPND | Operand |
| OPTR | Operator |
| ORD | Order |
| OVFLO | Overflow |
| OVLP | Overlap |
| OVLY | Overlay |
| OVRN | Overrun |
| P.PROG | Problem Program |
| PARAM | Parameter |
| PAREN | Parenthesis |
| PC | Program Check |
| PCI | Program Controlled Interrupt |
| PCOPSW | Program Check Old PSW |
| PG | Page |
| PGLIN | Page and Line |
| PH | Phase |
| PKD | Packed |
| PNCH | Punch |
| PNDG | Pending |
| PNTR | Pointer |
| POS | Position |
| PR | Print |
| PREC | Precision |
| PREV | Previous |
| PRI | Priority |
| PROB | Problem |
| PROC | Process |
| PROG | Program |
| PROT | Protect, Protection |
| PRT | Print |
| PT | Point |
| PTR | Printer |
| PUB | Physical Unit Block |
| Q | Queue |
| R+S | Reset+Start |
| R/W | Read/Write |
| RCD | Record |
| RCV | Receive |
| RD | Read |
| RDR | Reader |
| RDY | Ready |
| RECVY | Recovery |
| REF | Reference |
| REG | Register |
| REJ | Reject |
| REL | Release |
| RELOC | Relocatable |
| REQ | Request, Require |
| RES | Residual |
| RET | Return |
| RGN | Region |
| RI | Read In |
| RLS | Reels |
| RM | Record Mark |
| RO | Read Out |
| RPT | Report |
| RSLT | Result |
| RST | Reset |
| RSTR | Restore |
| RSTRT | Restart |
| RTE | Route |
| RTN | Routine |
| RWD | Rewind |
| R0 | Record Zero |
| SCHED | Schedule, Scheduler |
| SCN | Scan |
| SCTR | Sector |
| SECT | Section |
| SEG | Segment |
| SEL | Select |
| SEN | Sense |
| SEQ | Sequence |
| SER | Serial |
| SIG | Signal |
| SILI | Suppress Incorrect Length Indication |
| SIM | Simulator |
| SK | Seek |
| SM | Storage Mark |
| SNGL | Single |
| SP | Space |
| SPEC | Specification, Specify |
| SRCH | Search |
| ST | Store |
| STG | Storage |
| STMNT | Statement |
| STRD | Stored |
| STRTG | Starting |
| SUB | Subtract |

| | |
|---|---|
| SUMM | Summarize |
| SUP | Suppress |
| SUPVR | Supervisor |
| SV | Save |
| SVCOPSW | SVC Old PSW |
| SYM | Symbol, Symbolic |
| SYNC | Synchronize, Synchronizer |
| SYST | System |
| SW | Switch |
| | |
| TBL | Table |
| TEMP | Temporary |
| TM | Tapemark |
| TMN | Transmission |
| TMT | Transmit |
| TOT | Total |
| TP | Tape |
| TR | Transfer |
| TRANS | Transient |
| TRK | Track |
| TRLR | Trailer |
| TST | Test |
| TU | Tape Unit |
| TW | Typewriter |
| | |
| UNC | Unconditional |
| UNLD | Unload |
| UNPK | Unpack |
| UNPKD | Unpacked |
| UNUSL | Unusual |
| UPD | Update |
| | |
| VAR | Variable |
| VER | Verify |
| VOL | Volume |
| | |
| WA | Work Area |
| WD | Word |
| WLR | Wrong Length Record |
| WM | Wordmark |
| WR | Write |
| WRK | Work |
| | |
| XPL | Explain, Explanation |
| XTR | Extra |
| | |
| Z | Zero |
| ZN | Zone |

| ITEM | IDENTIFICATION |
|---|---|
| 1 | TERMINAL. Beginning, end or point of interruption in the program. |
| 2 | DECISION. Decision function dependent on existing conditions. |
| 3 | SUBROUTINE. Label of routine corresponds to listing label. |
| 4 | SUBROUTINE CHART IDENTIFICATION. Identifies chart where further explanation is found. |
| 5 | LABEL. Corresponds to listing label. |
| 6 | PROCESS BLOCK. Notation within the block defines the function being performed. |
| 7 | BLOCK COORDINATE. Can anpear on the left or right hand side of the block. |
| 8 | I/O BLOCK. Notation within the block defines the function being performed. |
| 9 | MODIFICATION BLOCK. Notation within the block defines the program modification being performed. |
| 10 | PREDEFINED PROCESS BLOCK. Notation within the block defines processing not included in this set of flow charts. |
| 11 | OFF-PAGE CONNECTOR. Notation within this block defines the flowchart and block coordinate to which control is being transferred. |
| 12 | OFF-PAGE CONNECTOR. Notation within this block defines the flowchart and block coordinate from which control is being transferred. |
| 13 | Same as 12 except the * defines a multiple entry. |
| 14 | ON-PAGE CONNECTOR. Exit from this block to another block on the same flowchart. |
| 15 | ON-PAGE CONNECTOR. Entry from another block on the same flowchart. |
| 16 | SUBROUTINE. Notation within the block defines the function being performed by the subroutine. |

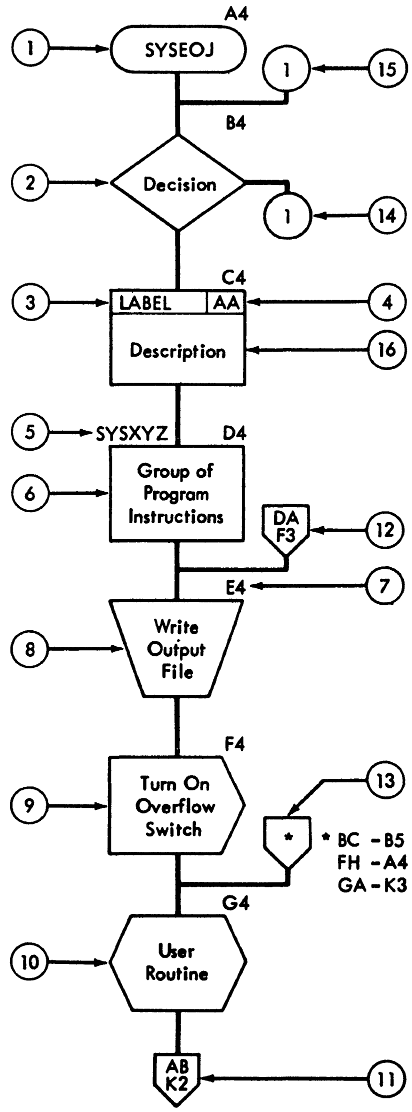



Figure 83. Description of Flowchart Symbols

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

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

1
```
// LISTIO SYS

              *** SYSTEM ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYSRDR            0       0C
 SYSIPT            0       0C
 SYSPCH            0       0D
 SYSLST            0       0F
 SYSLOG            0       1F
 SYSLNK            ** UA **
 SYSRES            1       90
```

2
```
// LISTIO PROG

              *** PROGRAM ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYS000            1       91
 SYS001            1       91
 SYS002            1       91
 SYS003            1       91
 SYS004            1       92
 SYS005            1       92
 SYS006            1       92
 SYS007            1       92
 SYS008            ** UA **
 SYS009            ** UA **
 SYS010            ** UA **
 SYS011            ** UA **
```

3
```
// LISTIO F1

              *** FOREGROUND 1 ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYS000            ** UA **
 SYS001            ** UA **
 SYS002            ** UA **
 SYS003            ** UA **
 SYS004            ** UA **
```

4
```
// LISTIO ALL

              *** SYSTEM ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYSRDR            0       0C
 SYSIPT            0       0C
 SYSPCH            0       0D
 SYSLST            0       0F
 SYSLOG            0       1F
 SYSLNK            ** UA **
 SYSRES            1       90

              *** PROGRAM ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYS000            1       91
 SYS001            1       91
 SYS002            1       91
 SYS003            1       91
 SYS004            1       92
 SYS005            1       92
 SYS006            1       92
 SYS007            1       92
 SYS008            ** UA **
 SYS009            ** UA **
 SYS010            ** UA **
 SYS011            ** UA **

              *** FOREGROUND 2 ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYS000            ** UA **
 SYS001            ** UA **
 SYS002            ** UA **
 SYS003            ** UA **
 SYS004            ** UA **

              *** FOREGROUND 1 ***

I/O UNIT   CMNT   CHNL    UNIT   MODE

 SYS000            ** UA **
 SYS001            ** UA **
 SYS002            ** UA **
 SYS003            ** UA **
 SYS004            ** UA **
```

Figure 84. Sample LISTIO Printouts (Part 1 of 2)

5
```
// LISTIO F2

                 *** FOREGROUND 2 ***

I/O UNIT    CMNT    CHNL    UNIT    MODE

 SYS000              ** UA **
 SYS001              ** UA **
 SYS002              ** UA **
 SYS003              ** UA **
 SYS004              ** UA **
```

6
```
// LISTIO SYSRDR

                 *** SYSTEM ***

I/O UNIT    CMNT    CHNL    UNIT    MODE

 SYSRDR               0      0C
```

7
```
// LISTIO UNITS

CHNL    UNIT    OWNER   I/O UNIT    CMNT   MODE

  0      0F              SYSLST
  0      0C              SYSRDR
  0      0C              SYSIPT
  0      1F              SYSLOG
  0      0D              SYSPCH
  1      90              SYSRES
  1      91      BG      SYS000
  1      91      BG      SYS001
  1      91      BG      SYS002
  1      91      BG      SYS003
  1      92      BG      SYS004
  1      92      BG      SYS005
  1      92      BG      SYS006
  1      92      BG      SYS007
```

8
```
// LISTIO UA

        *** UNASSIGNED ***

             CHNL    UNIT

               1      90
```

9
```
// LISTIO DOWN

             *** DOWN ***

             CHNL    UNIT

              ** NONE **
```

10
```
// LISTIO X'01F'

CHNL    UNIT    OWNER   I/O UNIT    CMNT   MODE

  0      1F              SYSLOG
```

Figure 84. Sample LISTIO Printouts (Part 2 of 2)

| CONTROL DICTIONARY ENTRY | ESD ITEM FOR PROCESSING | | | | |
|---|---|---|---|---|---|
| | SD | LD | ER | PC | CM |
| | • Origin on a doubleword boundary<br>• Current phase number saved in the input area for later use in the control dictionary.<br>• SD length saved so that it can be used in calculating the next phase origin at the end of ESD processing.<br>• Relocation factor for this SD computed. | • Check the linkage table to determine if a corresponding SD entry has been processed.<br>a. Processed-negative control dictionary number-ignore this LD.<br>b. Not processed - make this LD unassigned.<br>c. Processed-positive control dictionary number-assign the LD and save the control dictionary number in the input area.<br>• LD must point to a SD or CM or an error exists. | • If this phase is not to be autolinked, set the first byte of the origin field to a X'FF'. | • Origin on a doubleword boundary.<br>• PC length saved so that it can be used in calculating the next phase origin at the end of ESD processing.<br>• Relocation factor for this PC computed.<br>• Name field of PC must be blank or an error exists. | No comment |
| LD | Note A | Note D | Note G | A PC item cannot be matched against the control dictionary because it has a blank name field. Post any PC as a new control dictionary entry. | Error |
| LR | Note A | Note D | Note G | | Error |
| SD | Note B | Note E | Note H | | Error |
| PC | Not Possible | Not Possible | Not Possible | | Not Possible |
| CM | Error | Error | Note J | | Note L |
| ER | Note C | Note F | Note K | | Note M |
| NO MATCH | Post the SD to the control dictionary. | Post the LD to the control dictionary. | Post the ER to the control dictionary. | | Post the CM to the control dictionary. |

Figure 85. Description of ESD Processing

Notes For Figure 85.

A. An SD that matches an assigned LD/LR entry is an error. An SD that matches an unassigned LD/LR entry requires that:

1. The ESD number of the SD equal the LD/LR ESD number.

2. The assembled origin of the ESD item (SD) equal the assembled origin of the control dictionary entry (LD/LR).

If either requirement is not met, an error exists. If both requirements are met, replace the LD/LR entry with the SD item just processed.

After the linkage table has been updated, and if the SD is not to be bypassed, calculate the next available phase origin. Try to resolve any unassigned LD/LRs. If the ESD number is negative, skip the LD/LR. If the ESD number is positive and a control dictionary number exists in the linkage table for the LD/LR, put this number into the control dictionary. If the control dictionary number is negative, leave the LD/LR unassigned. If it is positive, assign the LD/LR entry in the control dictionary. The reason an attempt is made to assigh unassigned LD/LRs at this time is that a new SD has just been posted and it might define the unassigned LD/LR.

B. If the SD entry in the control dictionary is not for the current phase, or for the root phase, post the ESD-SD to the control dictionary as a new entry.

If the SD entry is for the current phase, or for the root phase, make the control dictionary number in the linkage table a negative value so that all LD references to this SD will be bypassed.

C. Replace the ER entry in the control dictionary with the ESD-SD item.

D. The three possible conditions and actions taken are as follows:

1. The control dictionary entry is an unassigned LD/LR. Set a switch to indicate a possible duplicate entry. Replace the control dictionary entry with the ESD-LD. (Change the input LD to an LR if the old control dictionary entry was an LR.)

2. The ESD input is an assigned LD and the control dictionary entry is an assigned LD/LR.

   a. The assembled origin of the input LD must equal the assembled origin of the control dictionary entry or an error exists.

   b. If the assembled origins agree and the LDs point to the same SD, ignore the input.

   c. If the input and control dictionary entry do not point to the same SD, the labels of the SD entries pointed to must match or an error exists.

   d. If the labels are the same, test the phase numbers. If equal phase numbers are found, set a switch to indicate a possible duplicate entry and ignore the input. If the phase numbers are different, post the input LD as a new control dictionary entry.

3. The ESD input is an unassigned LD and the control dictionary entry is an assigned LD/LR.

   a. The assembled origin of the input LD must equal the assembled origin of the control dictionary entry, or an error exists.

   b. Compare the phase numbers of the LDs. If the phase numbers are equal, set a switch to indicate a possible duplicate entry and ignore the input. If the phase numbers are different, post the input LD as a new control dictionary entry.

E. The two possible conditions and actions taken are as follows:

1. The ESD input is an assigned LD.

   a. The assembled origin of the input LD must equal the assembled origin of the control dictionary entry or an error exists.

   b. Compare the control dictionary number of the input with the control dictionary number of the control dictionary entry. If the control dictionary numbers are equal, ignore the input. If the numbers are different, an error exists (invalid duplication of a label).

2. The ESD input is an unassigned LD.

   a. Same as item 1 above.

   b. Compare the phase numbers of the input and the control dictionary entry. If the phase numbers are equal, set a switch to indicate a possible duplicate entry and ignore the input. If the phase numbers are different, post the input LD as a new control dictionary entry.

F. Make the LD entry an LR and replace the control dictionary entry (ER) with the input LR.

G. The two possible conditions and actions taken are as follows:

1. The input ER matches an unassigned LD/LR. Continue the scan of the control dictionary.

2. The input ER matches an assigned LD/LR or an SD.

   a. If an IJ prefix is found, autolink is specified, and the phase number of the control dictionary entry is not the current phase or the root phase, post the ER to the control dictionary. This enables the ER to be autolinked.

   b. If the prefix is not IJ, NOAUTO is specified, and the control dictionary entry is an assigned LD/LR, force the control dictionary entry to be an LR. If the control dictionary entry is either an SD or an LR, do not post the input as a new control dictionary entry.

H. Same as item G2 above.

J. Do not change the control dictionary entry (CM). Put the control dictionary number of the CM control dictionary entry into the linkage table entry of the ER, thereby assigning the ER.

K. Replace the control dictionary entry with the input ER for autolink.

L. Put the common (CM) with the longest length into the control dictionary.

M. Replace the ER entry in the control dictionary with the ESD-CM item.

## APPENDIX F: ERROR MESSAGE CROSS REFERENCE

| Message | Phase | Chart ID |
|---|---|---|
| 0I00I | $$A$IPL2 | AB |
| 0I01A | $$A$IPL2 | AD |
| 0I10A | $IPLRT2 | AJ |
| 0I11I | $IPLRT2 | AP |
| 0I12I | $IPLRT2 | AP |
| 0I13I | $IPLRT2 | AN |
| 0I14I | $IPLRT2 | AR |
| 0I15I | $IPLRT2 | AN |
| 0I16A | $IPLRT2 | AL |
| 0I17A | $IPLRT2 | AL |
| 0I18A | $IPLRT2 | AJ |
| 0I20I | $IPLRT2 | AM |
| 0I22I | $IPLRT2 | AY |
| 0I23I | $IPLRT2 | AP |
| 0P08 | $$A$$UP1 (Disk) | GV |
| | $$ANERRE (Tape) | HG |
| | $$ANERRG (Data Cell) | HP |
| | $$ANERRU (Unit Record) | JL |
| | $$ANERRX (Paper Tape) | JP |
| | $$ANERR9 (Optical Reader) | JR |
| 0P09 | $$A$SUP1 (Disk) | GV |
| | $$ANERRD (Tape) | HE |
| | $$ANERRG (Data Cell) | HP |
| | $$ANERRV (Unit Record) | JM |
| | $$ANERRX (Paper Tape) | JP |
| | $$ANERR9 (Optical Reader) | JR |
| 0P10 | $$A$$UP1 (Disk) | GV |
| | $$ANERRD (Tape) | HE |
| | $$ANERRG (Data Cell) | HN |
| | $$ANERRU (Unit Record) | JL |
| | $$ANERRX (Paper Tape) | JP |
| 0P11 | $$A$SUP1 (Disk) | GV |
| | $$ANERRD (Tape) | HE |
| | $$ANERRG (Data Cell) | HP |
| | $$ANERRU (Unit Record) | JM |
| | $$ANERRX (Paper Tape) | JP |
| 0P12 | $$ANERRB (Disk) | HB |
| | $$ANERRK (Data Cell) | HV |
| 0P13 | $$A$SUP1 (Disk) | GV |
| | $$ANERRG (Data Cell) | HN |
| 0P14 | $$A$SUP1 (Disk) | GV |
| | $$ANERRE (Tape) | HG |
| | $$ANERRG (Data Cell) | HQ |
| | $$ANERRV (Unit Record) | JM |
| | $$ANERR9 (Optical Reader) | JR |
| 0P15 | $$A$SUP1 (Disk) | GV |
| | $$ANERRG (Data Cell) | HN, HT |
| 0P16 | $$ANERRB (Disk) | HD |
| | $$ANERRK (Data Cell) | HV |
| 0P17 | $$ANERRB (Disk) | HC |
| | $$ANERRG (Data Cell) | HQ |
| | $$ANERRL (Tape) | HH |
| 0P18 | $$A$SUP1 (Disk) | GV |
| | $$ANERRE (Tape) | HG |
| | $$ANERRG (Data Cell) | HQ |
| | $$ANERRU (Unit Record) | JL |
| | $$ANERRX (Paper Tape) | JP |
| | $$ANERR9 (Optical Reader) | JR |
| 0P19 | $$ANERRB (Disk) | HC |
| | $$ANERRE (Tape) | HJ |
| | $$ANERRG (Data Cell) | HQ, HE |
| | $$ANERRU (Unit Record) | JK |
| | $$ANERRX (Paper Tape) | JP |
| 0P20 | $$A$SUP1 (Sense Error) | FU |
| | $$ANERRD (Tape) | HE |
| | $$ANERRJ (Data Cell) | HT |
| 0P21 | $$A$SUP1 (Disk) | GV |
| | $$ANERRJ (Data Cell) | HT |
| 0P22 | $$ANERRG | HN |
| 0P23 | $$ANERRJ | HT |
| 0P24 | $$ANERRA | HA |
| 0P25 | $$ANERRA | HA |
| 0P26 | $$ANERRA (Disk) | HB |
| | $$ANERRG (Data Cell) | HN |
| 0P27 | $$ANERRC | HA |
| | $$ANERRU | JK |
| 0P28 | $$A$SUP1 (Disk) | GV |
| | $$ANERRD (Tape) | HE |
| | $$ANERRG (Data Cell) | HN |
| | $$ANERRU (Unit Record) | JK |
| | $$ANERRX (Paper Tape) | JP |
| | $$ANERR9 (Optical Reader) | JR |
| 0P29 | $$ANERRE | HG |
| 0P30 | $$ANERRE | HG |
| 0P31 | $$ANERRA | HA |
| 0P32 | $$ANERRE | HJ |
| 0P33 | $$ANERRV | JN |
| 0P35 | $$ANERR9 (Optical Reader) | JR |
| 0P60D | $$ANERRY | JT |
| 1I40D | $$ANERR0 | JW |
| 1A00D | $$BATTNI | LG |
| 1A10D | $$BATTNI | LG |
| 1A20D | $$BATTNI | LG |
| 1A30D | $$BATTNI | LG |
| 1A40D | $$BATTNI | LG |
| | $$BATTNK | LV |
| 1A50D | $$BATTNI | LG |
| 1A60D | $$BATTNI | LG |
| 1A70D | $$BATTNI | LG |
| 1C20D | $$BATTNH | KU |
| 1C30A | $$BATTNM | MC |
| 1L00D | $$BATTNK | LV |
| 1L10D | $$BATTNK | LV |
| 1P00D | $$BATTNF | KQ |
| 1P10D | $$BATTNG | KS |
| 1S00D | $$BATTNB | KE |
| | $$BATTNG | KS |
| | $$BATTNE | KN, KP |

| Message | Phase | Chart ID |
|---|---|---|
| | $$BATTNI | KV,KX,KY |
| | | KZ,LE,LF |
| | | MJ,MK,ML |
| | $$BATTNJ | LH |
| | $$BATTNK | LP,LQ,LS |
| | | LT, LU |
| | $$BATTNL | LZ, MA |
| | $$BATTNM | MC |
| | $$BATTNN | MG |
| 0P70I | $$BEOJ2 | NL |
| 0P71I | $$BEOJ2 | NL |
| 0P72I | $$BEOJ2 | NL |
| 0P73I | $$BEOJ2 | NL |
| 0P74I | $$BEOJ2 | NL |
| 0P75I | $$BEOJ2 | NL |
| 0P76I | $$BEOJ2 | NL |
| 0P77I | $$BEOJ2 | NL |
| 0P78I | $$BEOJ2 | NL |
| 0S00I | $$BPCHK | NT |
| | $$BILSVC | NN |
| 0S01I | $$BEOJ2 | NL |
| 0S02I | $$BEOJ2 | NL |
| 0S03I | $$BPCHK | NT |
| 0S04I | $$BILSVC | NN |
| 0S05I | $$BILSVC | NN |
| 0S07I | $$BPSW | NR |
| 0S08I | $$BEOJ | NB |
| 0S09I | $$BEOJ1 | NJ |
| 0S10I | $$BTERM | NE |
| 1A0ND | $JOBCTLD | CY |
| 1A1ND | $JOBCTLD | CY |
| 1A2ND | $JOBCTLD | CY |
| 1A20D | $JOBCTLG | DY |
| 1A20D | $JOBCTLJ | ET |
| 1A3ND | $JOBCTLD | CY |
| 1A4ND | $JOBCTLD | CY |
| 1A40D | $JOBCTLJ | ET |
| 1A5ND | $JOBCTLD | CY |
| 1A50D | $JOBCTLJ | ET |
| 1A6ND | $JOBCTLD | CY |
| 1A7ND | $JOBCTLD | CY |
| 1A70D | $JOBCTLJ | ET |
| 1A80D | $JOBCTLD | CY |
| 1A90D | $JOBCTLD | CY |
| 1C00A | $JOBCTLA | BL |
| 1C10A | $JOBCTLA | BL |
| 1C10A | $JOBCTLG | DY |
| 1C10A | $JOBCTLJ | ET |
| 1C30A | $JOBCTLG | DY |
| 1C30A | $JOBCTLJ | ET |
| 1C80D | $JOBCTLA | BL |
| 1L00D | $JOBCTLJ | ET |
| 1L10D | $JOBCTLG | DY |
| 1L10D | $JOBCTLJ | ET |
| 1P00D | $JOBCTLG | DY |
| 1S00D | $JOBCTLA | BL |
| 1S10D | $JOBCTLG | DY |
| 1S10D | $JOBCTLJ | ET |

| Message | Phase | Chart ID |
|---|---|---|
| 2100I | $LNKEDT2 | RK |
| 2101I | $LNKEDT4 | RT |
| 2102I | $LNKEDT | QH |
| | $LNKEDT4 | RX |
| 2110I | $LNKEDT4 | RT |
| 2112I | $LNKEDT | QC, QU |
| | $LNKEDT4 | RZ |
| 2113I | $LNKEDT2 | RM |
| | $LNKEDT4 | RZ |
| 2114I | $LNKEDT4 | RU |
| 2115I | $LNKEDT4 | RY |
| 2116I | $LNKEDT4 | RT |
| 2120I | $LNKEDT6 | SB |
| 2122I | $LNKEDT6 | SC |
| 2123I | $LNKEDT6 | SA |
| 2124I | $LNKEDT6 | SD |
| 2125I | $LNKEDT4 | RW |
| | $LNKEDT6 | SB |
| 2131I | $LNKEDT6 | SG |
| 2133I | $LNKEDT4 | RU |
| 2135I | $LNKEDT | QC, QU |
| 2136I | $LNKEDT | QC, QU |
| 2140I | $LNKEDT0 | RB, RD |
| 2141I | $LNKEDT0 | RH |
| 2142I | $LNKEDT0 | RB |
| 2143I | $LNKEDT0 | RF, RG |
| 2144I | $LNKEDT | QE, QN |
| 2145I | $LNKEDT0 | RC |
| 2146I | $LNKEDT0 | RD |
| 2147I | $LNKEDT2 | RR |
| 2150I | $LNKEDT | QT |
| 2151I | $LNKEDT2 | RM |
| 2155I | $LNKEDT2 | RN |
| 2156I | $LNKEDT2 | RP |
| 2158I | $LNKEDT2 | RR |
| 2170I | $LNKEDT2 | RL,RN,RQ |
| 2181I | $LNKEDT6 | SA |
| 2182I | $LNKEDT6 | SF |
| 2185I | $LNKEDT8 | SK |
| 2191I | $LNKEDT | QA |
| | $LNKEDT2 | RS |
| 2192I | $LNKEDT6 | SE |
| 2193I | $LNKEDT | QT |
| 2194I | $LNKEDT | QM |
| 3C10I | CORGZ | WU |
| 3C20I | CORGZ | WU |
| 3C21I | CORGZ | WU |
| 3C30I | CORGZ | WU |
| 3C33I | CORGZ | WU |
| 3C40I | CORGZ | WU |
| 3C60I | CORGZ | WU |
| 3C61I | CORGZ | WU |
| 3C62I | CORGZ | WU |
| 3C63I | CORGZ | WU |
| 3C64I | CORGZ | WU |
| 3C65I | CORGZ | WU |
| 3C66I | CORGZ | WU |
| 3D10D | DSERV | XA |
| 3D20D | DSERV | XA |

| Message | Phase | Chart ID |
|---|---|---|
| 3D43I | DSERV | XB |
| 3D47I | DSERV | XB |
| 3M10D | MAINT | TB |
| 3M11D | MAINTR2 | TN |
| 3M21I | MAINT | TD |
| | MAINTC2 | TJ |
| | MAINTR2 | TM, TW |
| | MAINTS2 | UB, UN |
| | MAINTCN | VN |
| | MAINTCL | VU |
| | MAINTA | VA,VB,VL |
| 3M22I | $LNKEDTC | SU |
| 3M23D | MAINTS2 | UH |
| 3M25D | MAINTS2 | UC |
| 3M26D | MAINTS2 | UG |
| 3M33I | MAINTC2 | TK |
| | MAINTR2 | TP,TW |
| | MAINTS2 | UN |
| 3M34I | MAINTR2 (Listing Only) | |
| 3M35D | MAINTS2 | UE |
| 3M43I | MAINTR2 | TL |
| | MAINTS2 | UM |
| 3M52I | $LNKEDTC | SU |
| | MAINTR2 | TM |
| | MAINTS2 | UC |
| 3M54I | MAINTC2 | TJ |
| | MAINTR2 | TW |
| | MAINTS2 | UN |
| 3M61I | MAINTA | VD |
| 3M62I | MAINTA | VC |
| 3M63I | MAINTA | VD |
| 3M64I | MAINTA | VD |
| 3M65I | MAINTA | VF |
| 3M66I | MAINTA | VF |
| 3M67I | MAINTA | VK |
| 3M68I | MAINTCN | VR |
| 3M69I | MAINTCN | VR |
| 3M70A | MAINTCN | VS |
| | MAINTA | VC,VG,VH |
| | | VJ, VK |
| 3R10D | RSERV | YA |
| 3R21I | RSERV | YB |
| 3R27I | RSERV | YC |
| 3R43I | RSERV | YA |
| 3S10D | SSERV | ZL |
| 3S21I | SSERV | ZJ, ZL |
| 3S33I | SSERV | ZL |
| 3S43I | SSERV | ZB, ZC |

## PIK (Program Interrupt Key)

The PIK is a halfword in length and consists of a zero value in the high-order byte and the key value in the low-order byte. The key value is the key of the program that was last enabled for interrupts.

When an interrupt occurs, the value in the PIK indicates to the supervisor which program was interrupted. It can also be used by transient programs and problem programs to determine if they are running as BG, F1, or F2.

The value of the PIK equals the displacement from the beginning of the PIB table to the PIB entry for the program (task). For BG, F2, and F1 tasks, this value equals the storage protect key multiplied by 16.

| Task | PIK Value |
|---|---|
| All Bound* | X'00' |
| BG | X'10' |
| F2* | X'20' |
| F1* | X'30' |
| AR | X'40' |
| Quiese I/O | X'50' |
| Supervisor | X'60' |

*Multiprogramming generation option only.

The PIK is set by task selection within the general exit routine. The fetch routine sets the PIK to X'60' because it enables itself for interrupts and because it gets control directly from the SVC interrupt routines. The SVC interrupt routines, like other completely disabled supervisor routines, do not change the PIK from the value it had when the interrupt occurred that transferred control.

## LTK (Logical Transient Key)

The LTK has the same value as the PIK when the logical transient area is in use. When the transient area is free, the LTK equals zero. The SVC 2 routine sets the LTK and the SVC 11 routine resets it to zero.

## RIK (Requestor I/O Key)

When a supervisor routine (fetch or physical transient) issues a SVC 0 or SVC 15, the routine puts the value to be used in the CAW storage protect key into the high-order digit of the second byte of the RIK halfword. When this value is zero, the low order digit can have these special meanings:

| RIK | Meaning |
|---|---|
| X'01' | This is a SYSLOG I/O request. The channel scheduler is not to type a SYSLOG ID prefix. |
| X'02' | This has been a fetch I/O request. This special code is required by ERP to recognize fetch requests. |

Fetch always sets a X'02' in the RIK. ERP transients put the key of the program requiring ERP into the RIK, when the ERP is a retry of a user EXCP and the ERP transient requires control to return to itself.

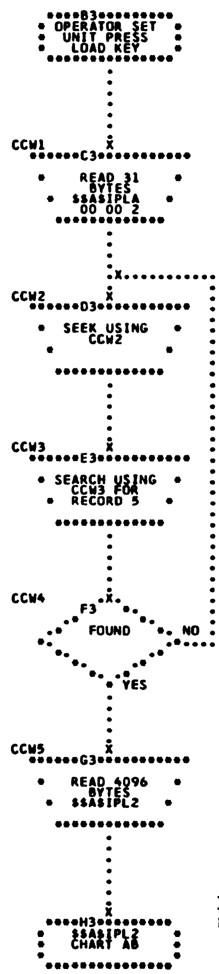
Physical transients put a X'01' into the RIK when they are doing a SYSLOG I/O. The PIK for physical transients has a value of X'06', therefore the channel scheduler would type "SP" (supervisor ID) as the SYSLOG ID. The physical transients put the ID of the program referred to by the message into the message.

## FIK (Fetch I/O Key)

Used by the fetch to validate the phase name address and load address. FIK has the following values:

| | |
|---|---|
| 1 | Key of the problem program requestor. |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 if the transient issued the SVC4. Key of the problem program if not a transient. |

## Chart AA. BOOTSTRAP-- $$A$IPLA; Refer to IPL, Chart 01




BY PRESSING THE LOAD KEY, THE OPERATOR CAUSES MICROPROGRAMMING TO READ THE $$A$IPL1 RECORD FROM SYSRES (CYLINDER 0, TRACK 0, RECORD 1) INTO MAIN STORAGE LOCATIONS 0-17 (HEXADECIMAL). THIS RECORD CONSISTS OF:

| HEX LOC | 0 | 8 | 10 | 18 |
|---|---|---|---|---|
| CONTENT | PSW | CCW1 | CCW2 | |

06 00 00 18 60 00 00 28
CCW1

READ DISK INTO STORAGE LOCATION HEX 18. (40-BYTES OF DATA, CHAINED CCW, AND SUPPRESSED WLR.)

MICROPROGRAMMING INITIATES A SIO COMMAND SPECIFYING SYSRES AS THE UNIT AND CCW1 (HEX LOC 8) AS THE FIRST CCW. THIS COMMAND CAUSES THE $$A$IPLA RECORD TO BE READ INTO MAIN STORAGE LOCATIONS 18-3F (HEXADECIMAL). THIS RECORD CONSISTS OF:

| HEX LOC | 18 | 20 | 28 | 30 | 37 - 3F |
|---|---|---|---|---|---|
| CONTENT | CCW3 | CCW4 | CCW5 | SEEK ADDRESS | UNUSED |

- THE CCWS (1-5) ARE CHAINED TOGETHER.
- THE SEEK ADDRESS (HEX LOC 30) CONTAINS THE SYSRES DISK ADDRESS OF THE $$A$IPL2 PROGRAM. THE FORMAT OF THE SEEK ADDRESS FIELD IS:

| HEX LOC | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
|---|---|---|---|---|---|---|---|
| DECIMAL CONTENT | 0 | 0 | 0 | 0 | 0 | 1 | 5 |
| MEANING | B | B | C | C | H | H | R |

WHERE: BB IS BIN NO.
CC IS CYLINDER NO.
HH IS TRACK NO.
R IS RECORD NO.

07 00 00 30 60 00 00 06
CCW2

SEEK THE DISK ADDRESS SPECIFIED BY THE FIRST 6-BYTES OF DATA IN THE SEEK ADDRESS LOCATED AT STORAGE LOCATION HEX 30. (CHAINED CCW AND SUPPRESSED WLR.)

31 00 00 32 60 00 00 05
CCW3

SEARCH FOR THE DISK RECORD SPECIFIED BY THE 5-BYTES OF DATA IN STORAGE LOCATION HEX 32 (SEEK ADDRESS PLUS 2-BYTES). THE CCW IS CHAINED AND WLR IS SUPPRESSED.

AT THE COMPLETION OF THIS BOOTSTRAP OPERATION, $$A$IPL2 HAS BEEN LOADED INTO MAIN STORAGE AND MICROPROGRAMMING TRANSFERS CONTROL TO IT BY LOADING THE PSW FROM MAIN STORAGE LOCATION 0 (HEXADECIMAL).

08 00 00 18 60 00 00 00
CCW4

TRANSFER IN CHANNEL (TIC) TO CCW3 UNTIL THE SEARCH ADDRESS SPECIFIED BY CCW3 IS FOUND. (CHAINED CCW AND SUPPRESSED WLR.)

06 00 30 00 20 00 10 00
CCW5

READ DISK INTO STORAGE LOCATION HEX 3000. (4096-BYTES OF DATA, SUPPRESSED WLR.) THIS CCW IS NOT CHAINED.

THE PSW AT LOCATION 0 CONTAINS THE ADDRESS OF THE FIRST EXECUTABLE INSTRUCTION OF THE $$A$IPL2 PROGRAM.

Chart AB. Clear Storage and Load Supervisor--
$$A$IPL2, Refer to IPL, Chart 01

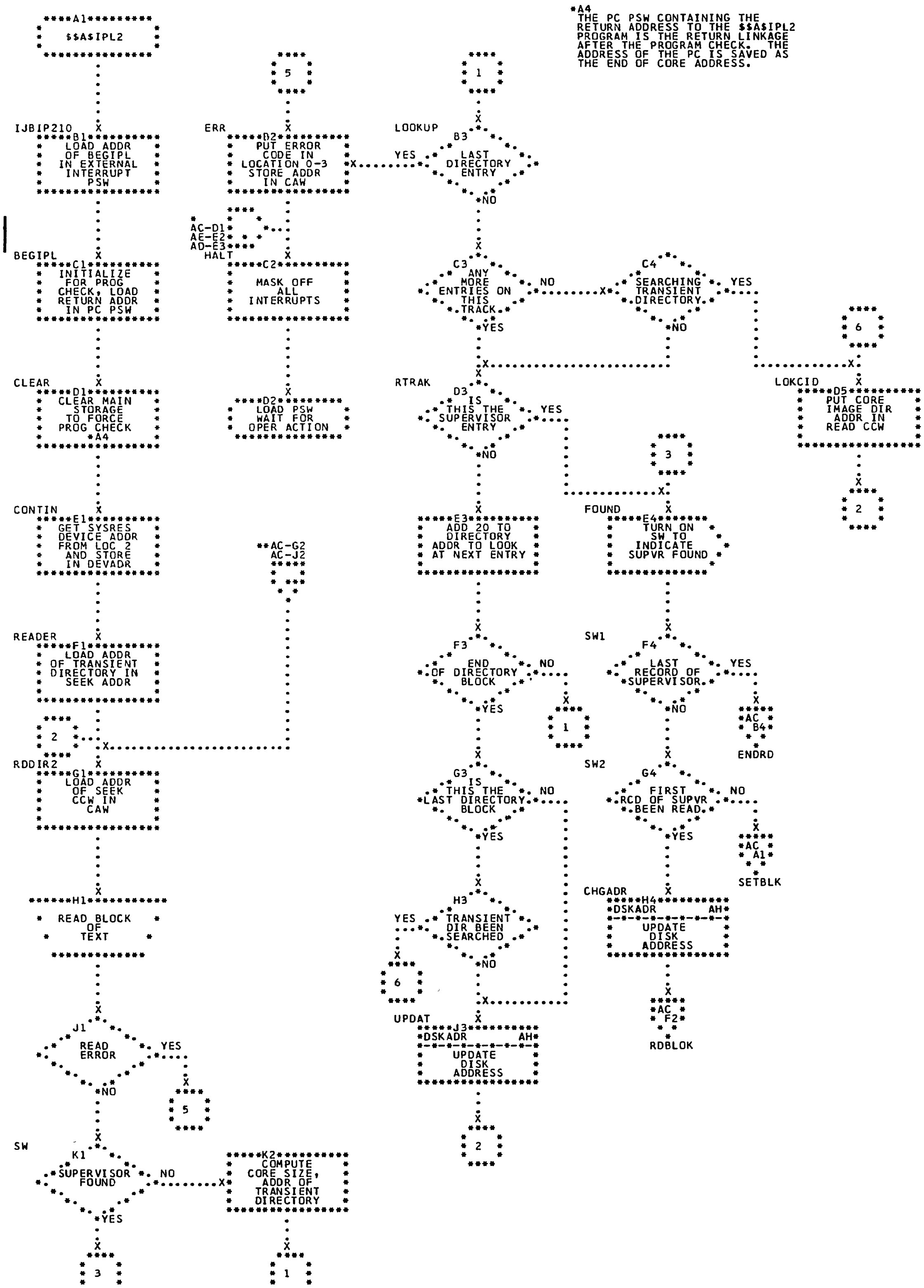


Chart AC. Build Two Device System (Part 1 of 2)- $$A$IPL2 ;
Refer to IPL, Chart 01

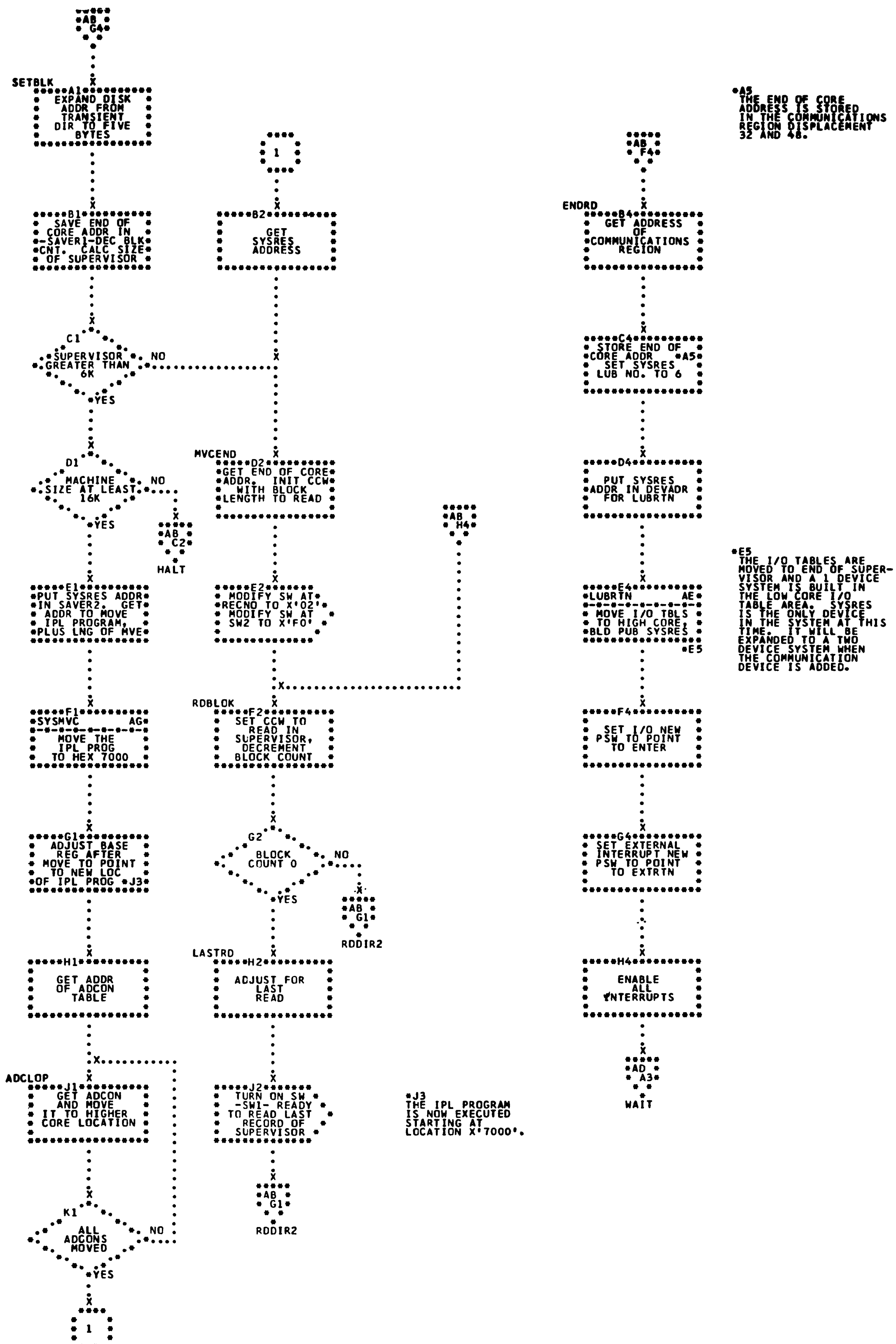


Chart AD. Build Two Device System (Part 2 of 2)- $$A$IPL2 ;
Refer to IPL, Chart 01

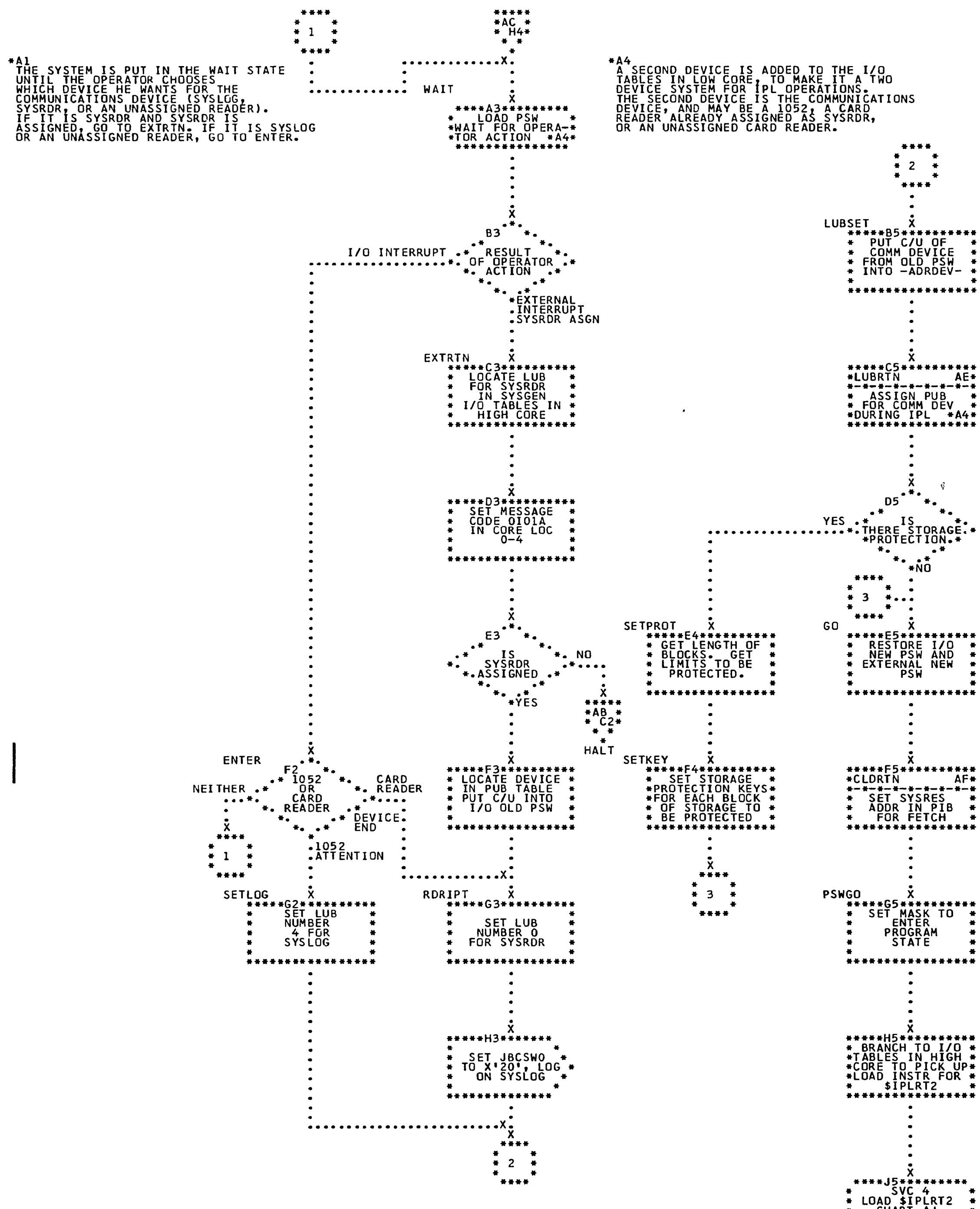


Chart AE. Move I/O Tables-- $$A$IPL2; Refer to IPL, Chart 01

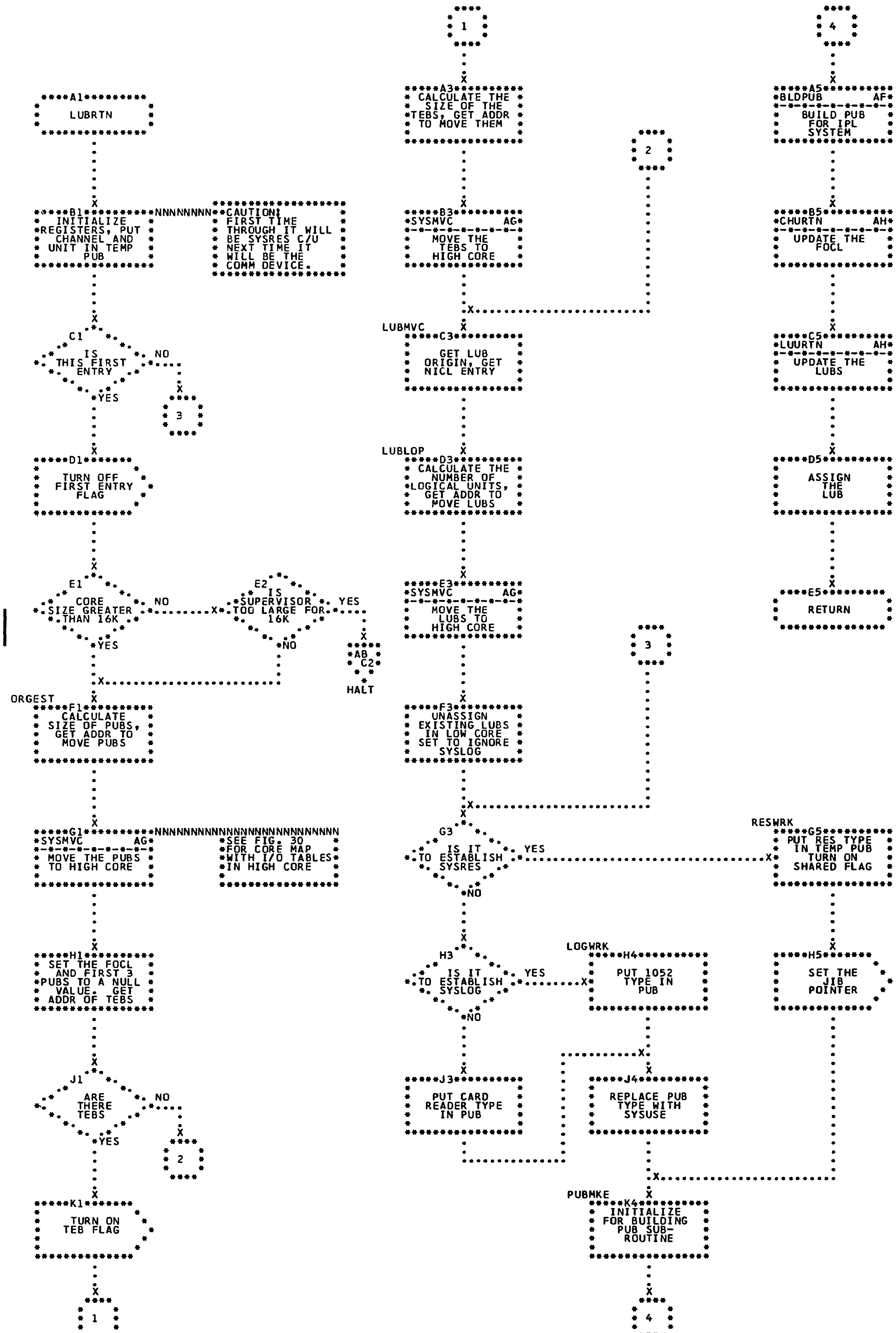


Chart AF. Build PUB Table-- $$A$IPL2; Refer to IPL, Chart 01

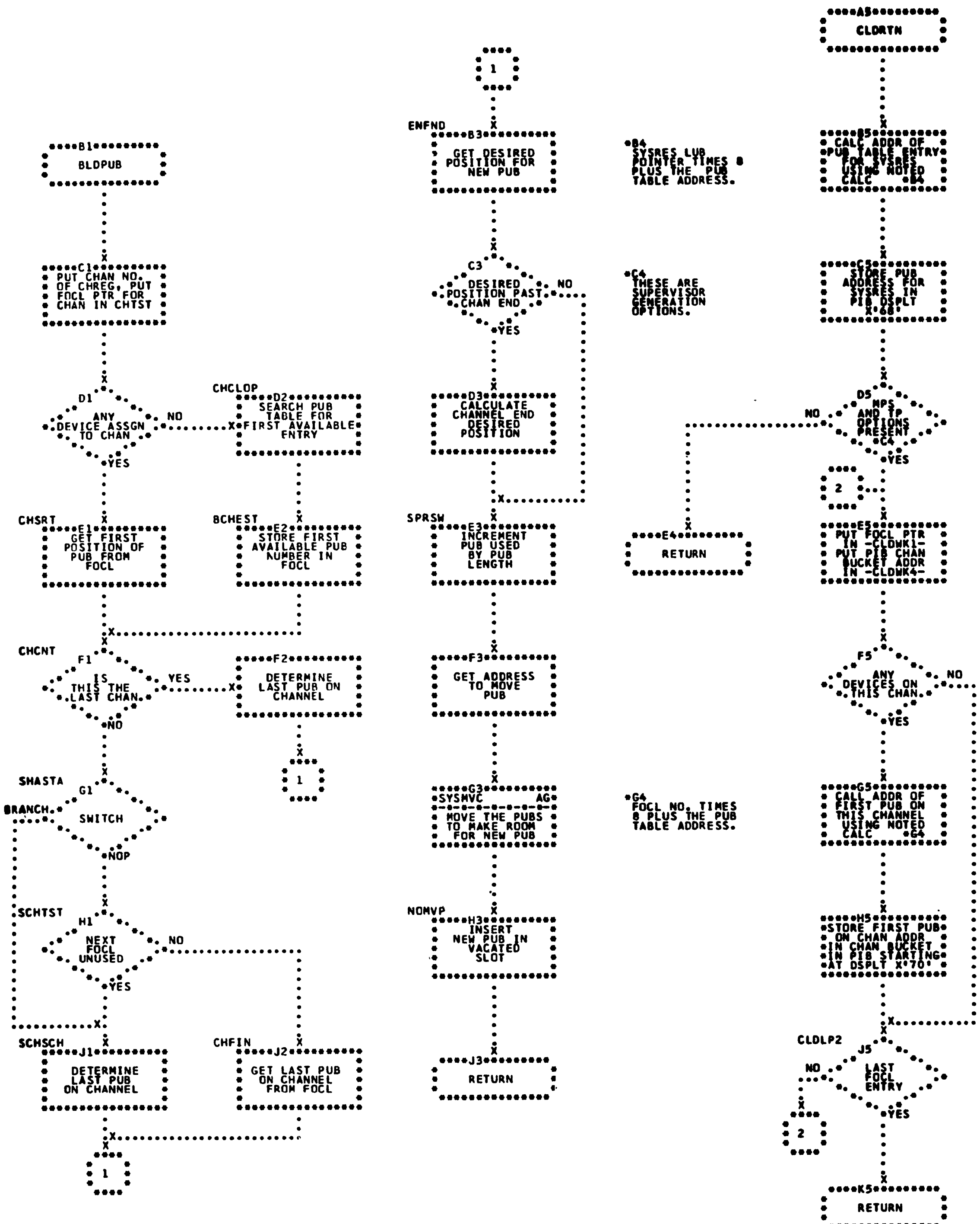


Chart AG. Common Move Subroutine-- $$A$IPL2; Refer to IPL, Chart 01

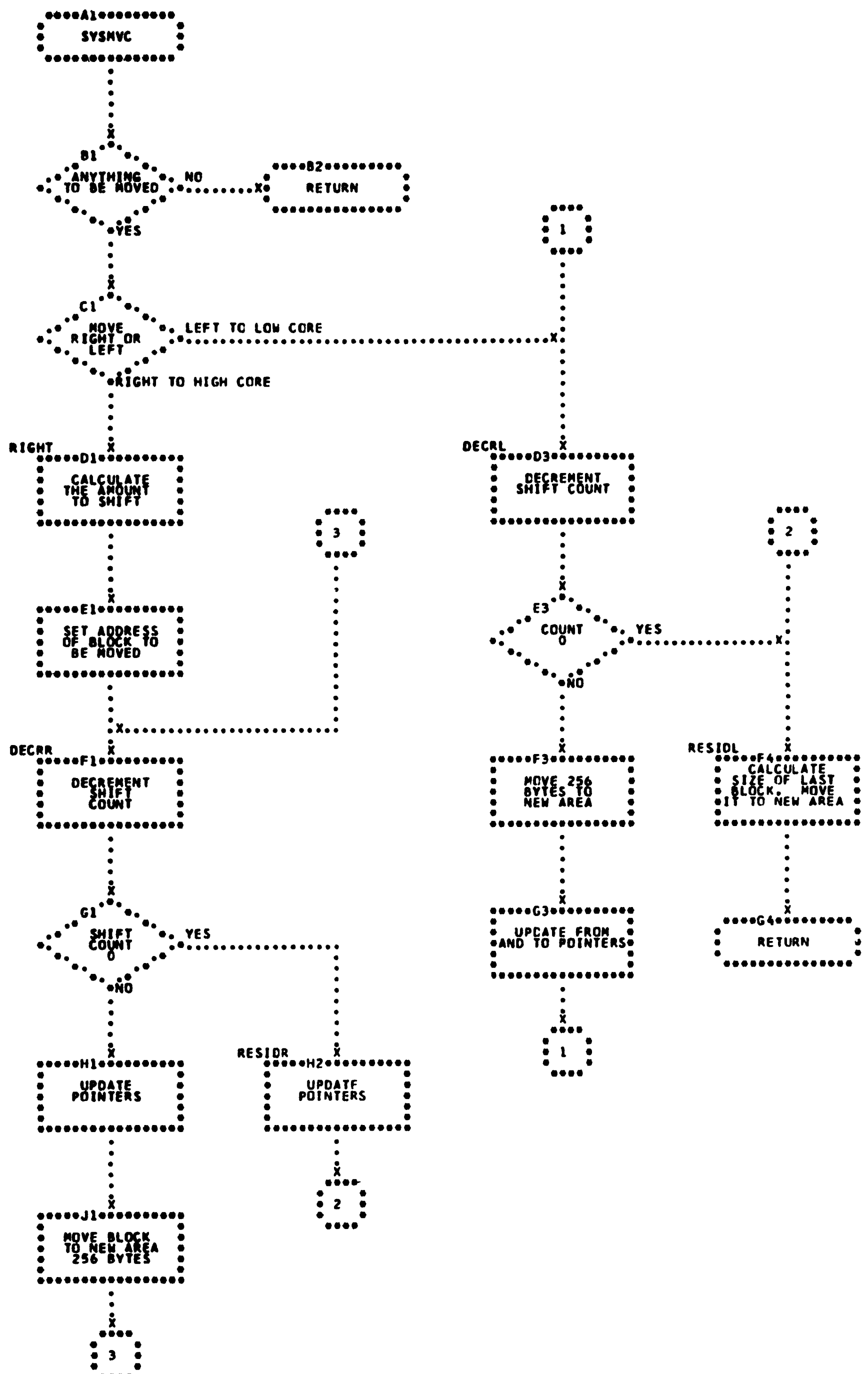


Chart AH. Update Disk Address-- $$A$IPL2; Refer to IPL , Chart 01

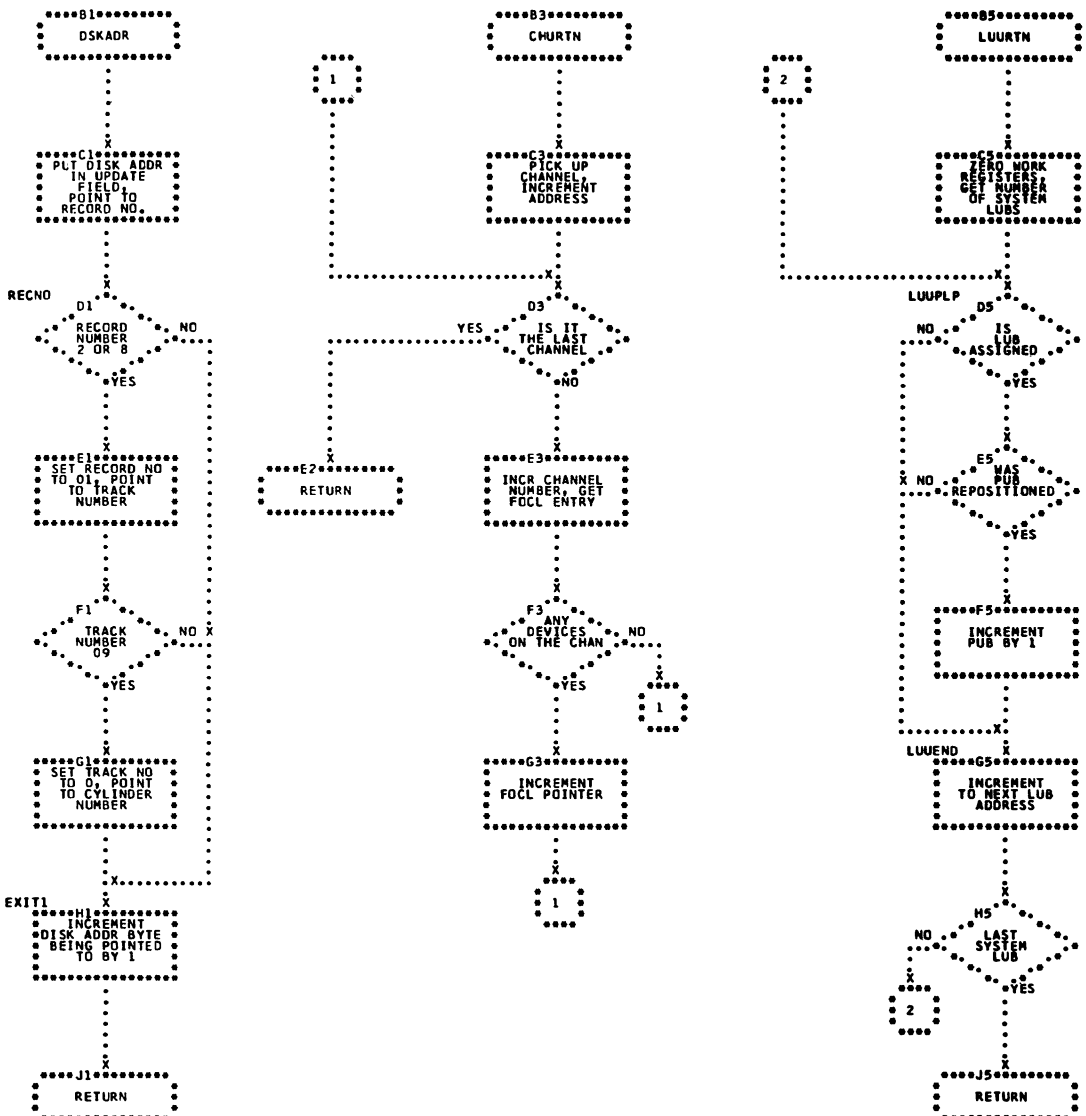


Chart AJ. Initialization and Read Control Cards-- $IPLRT2; Refer to IPL, Chart 02

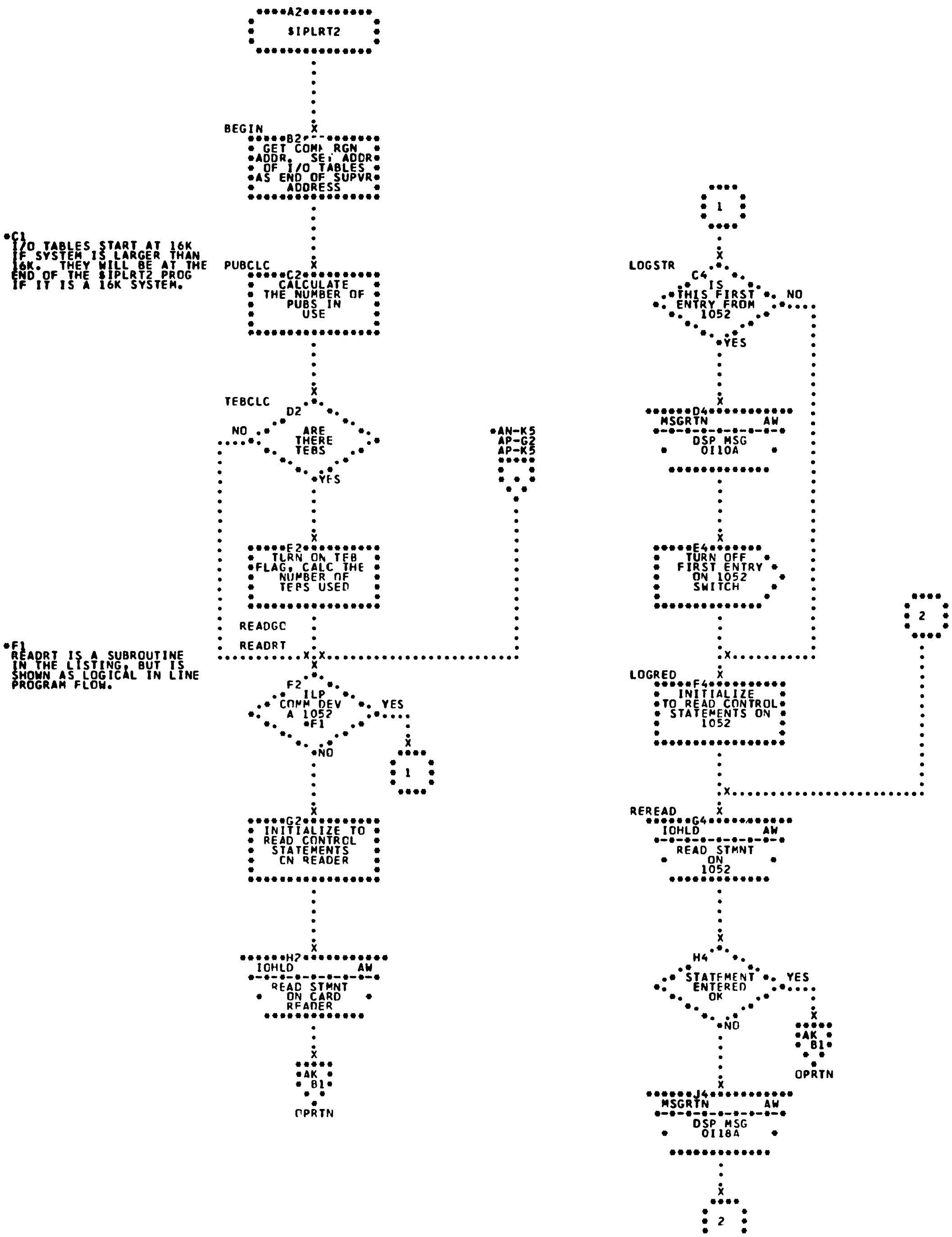


Chart AK. Evaluate Control Statement and Check Time of Day-- $IPLRT2; Refer to IPL, Chart 02

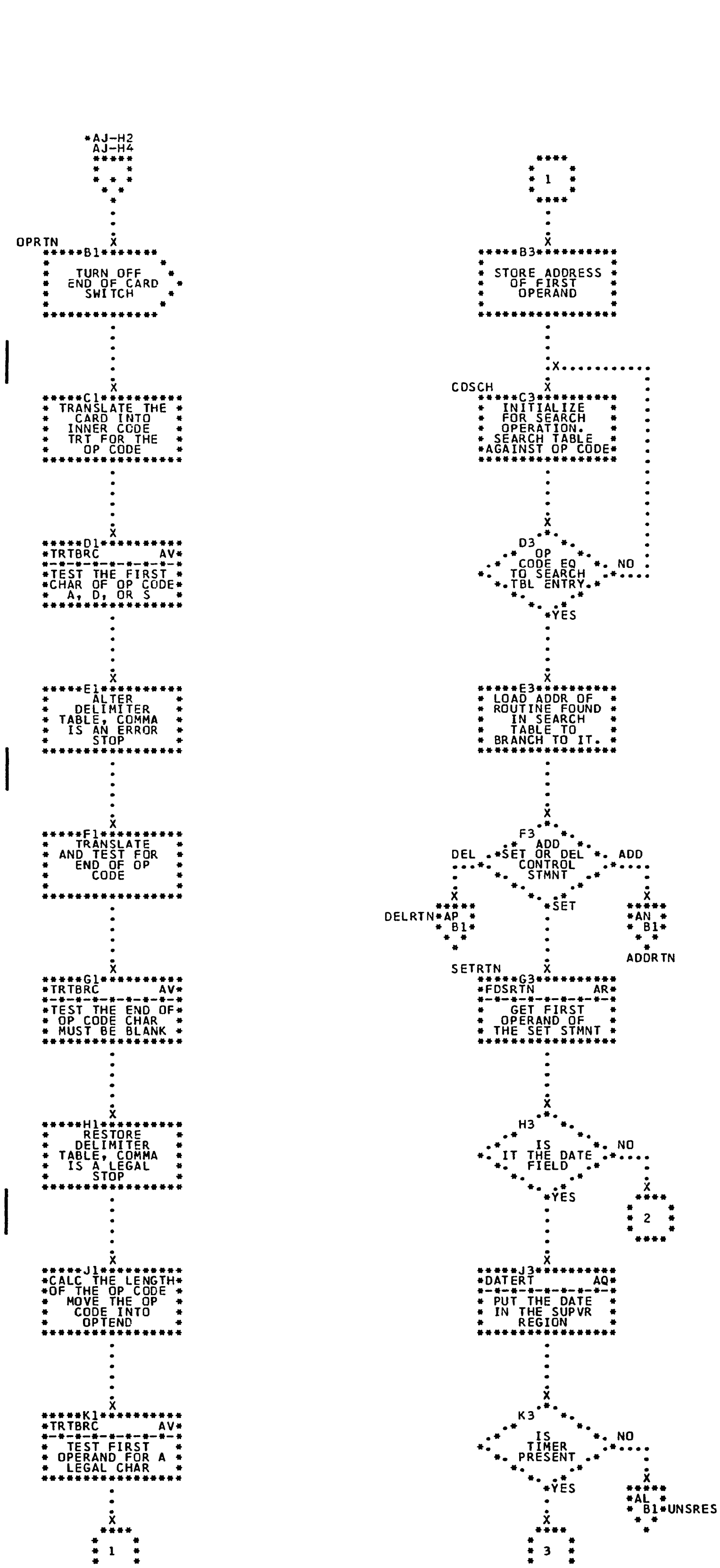


Chart AL. Assign SYSRES and SYSLOG-- $IPLRT2; Refer to IPL, Chart 02

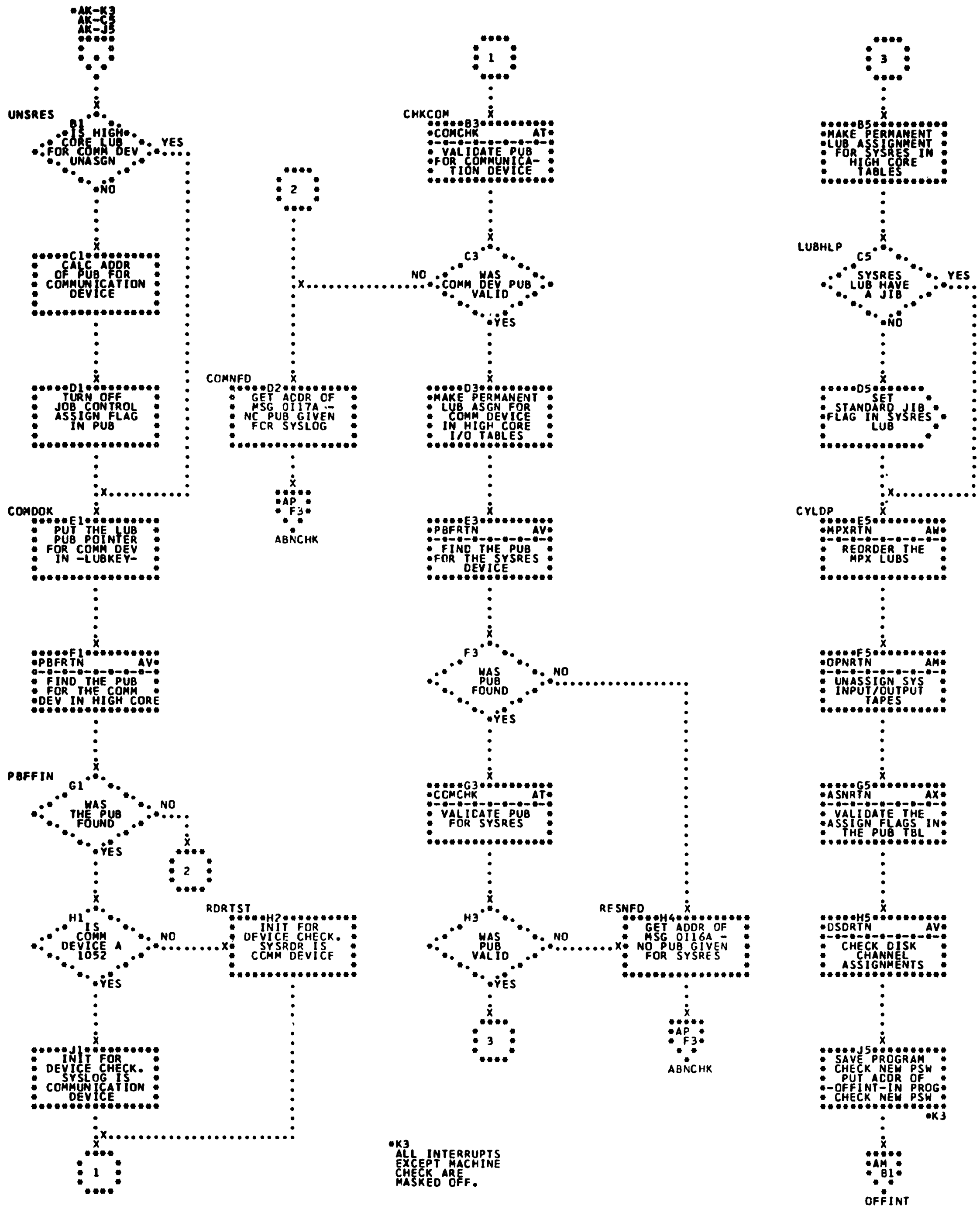


Chart AM. Move I/O Tables to Low Main Storage-- $IPLRT2;
Refer to IPL, Chart 02

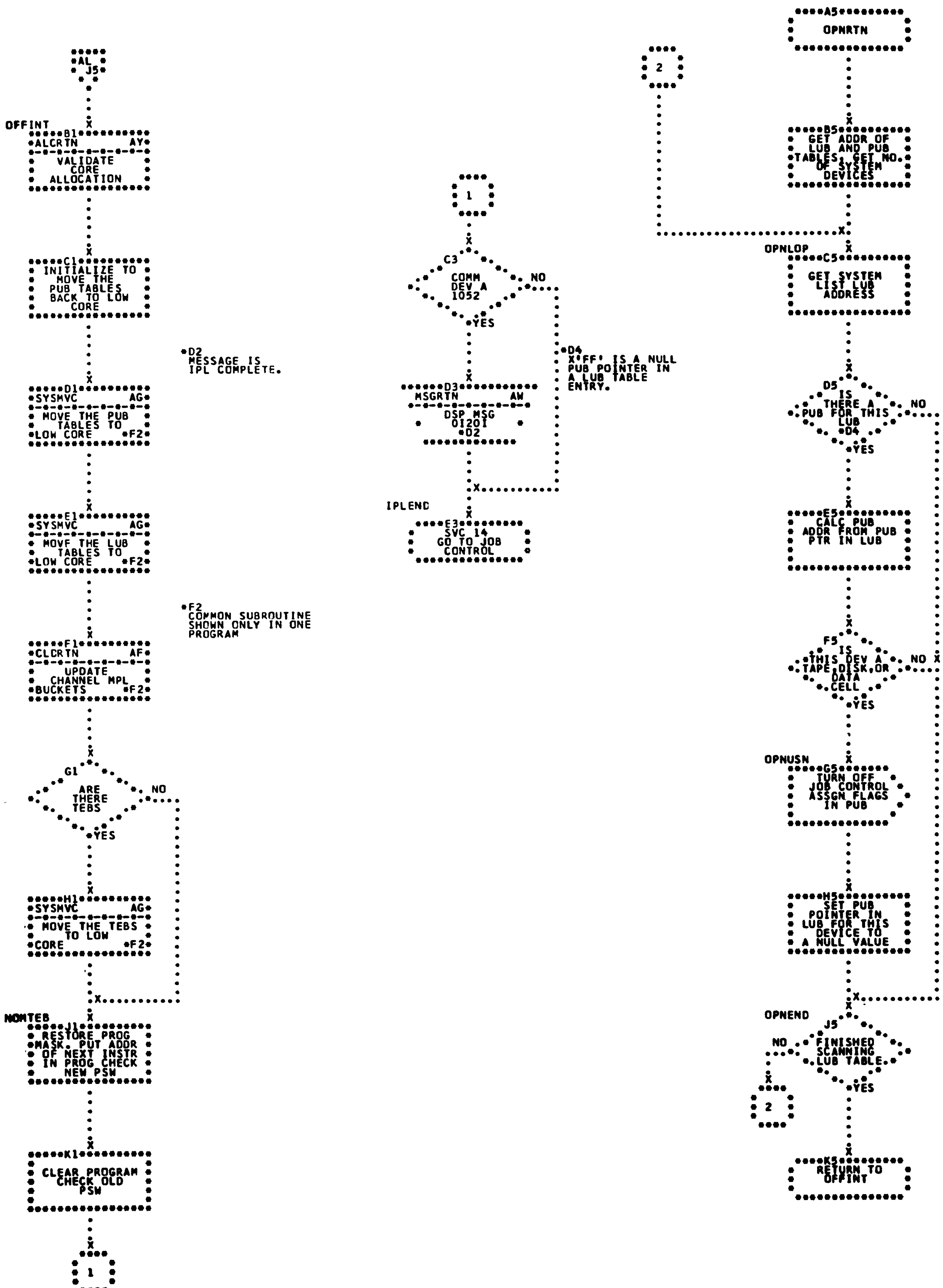


Chart AN. Add a Device-- $IPLRT2; Refer to IPL, Chart 02

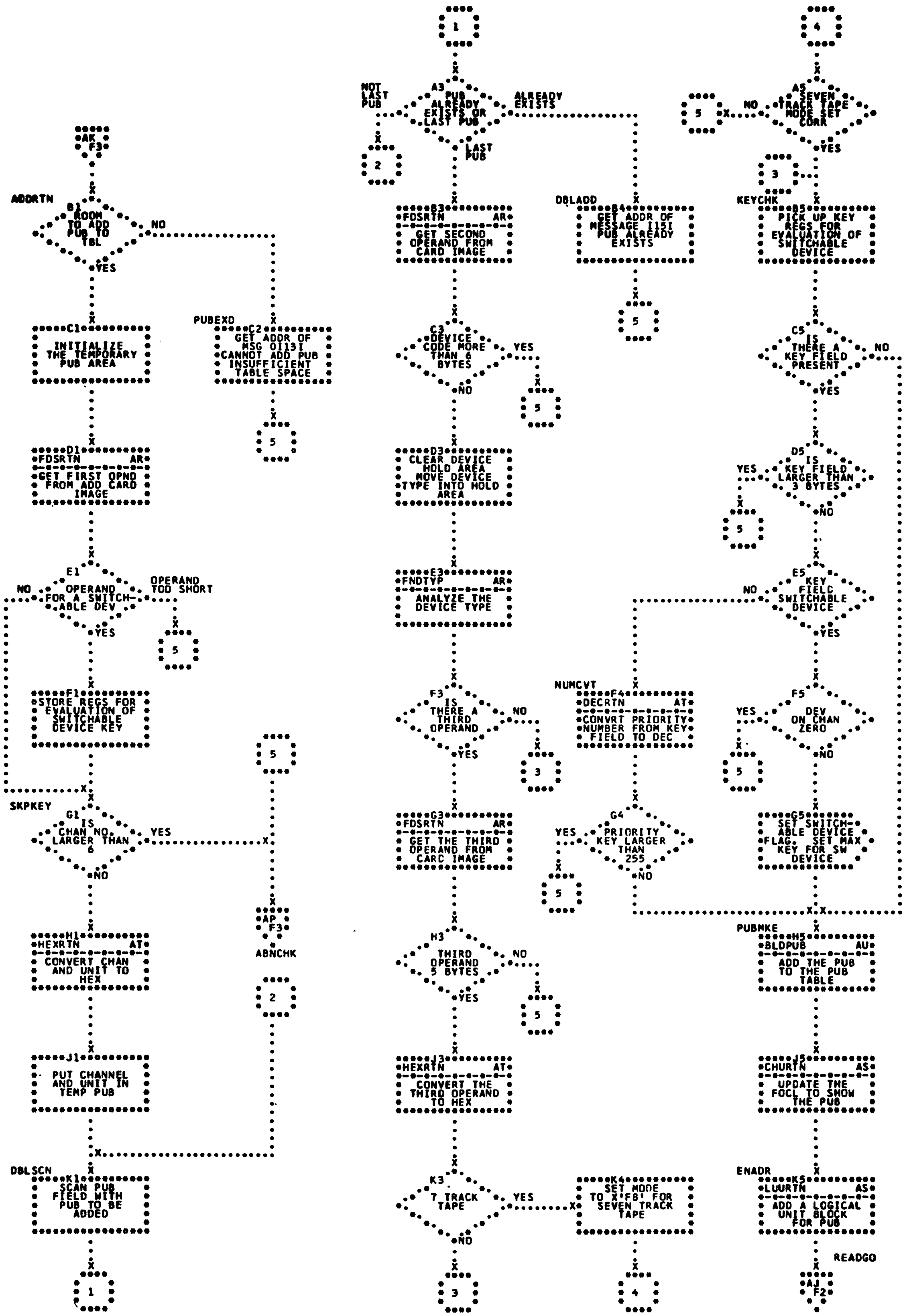


Chart AP. Delete a PUB-- $IPLRT2; Refer to IPL, Chart 02

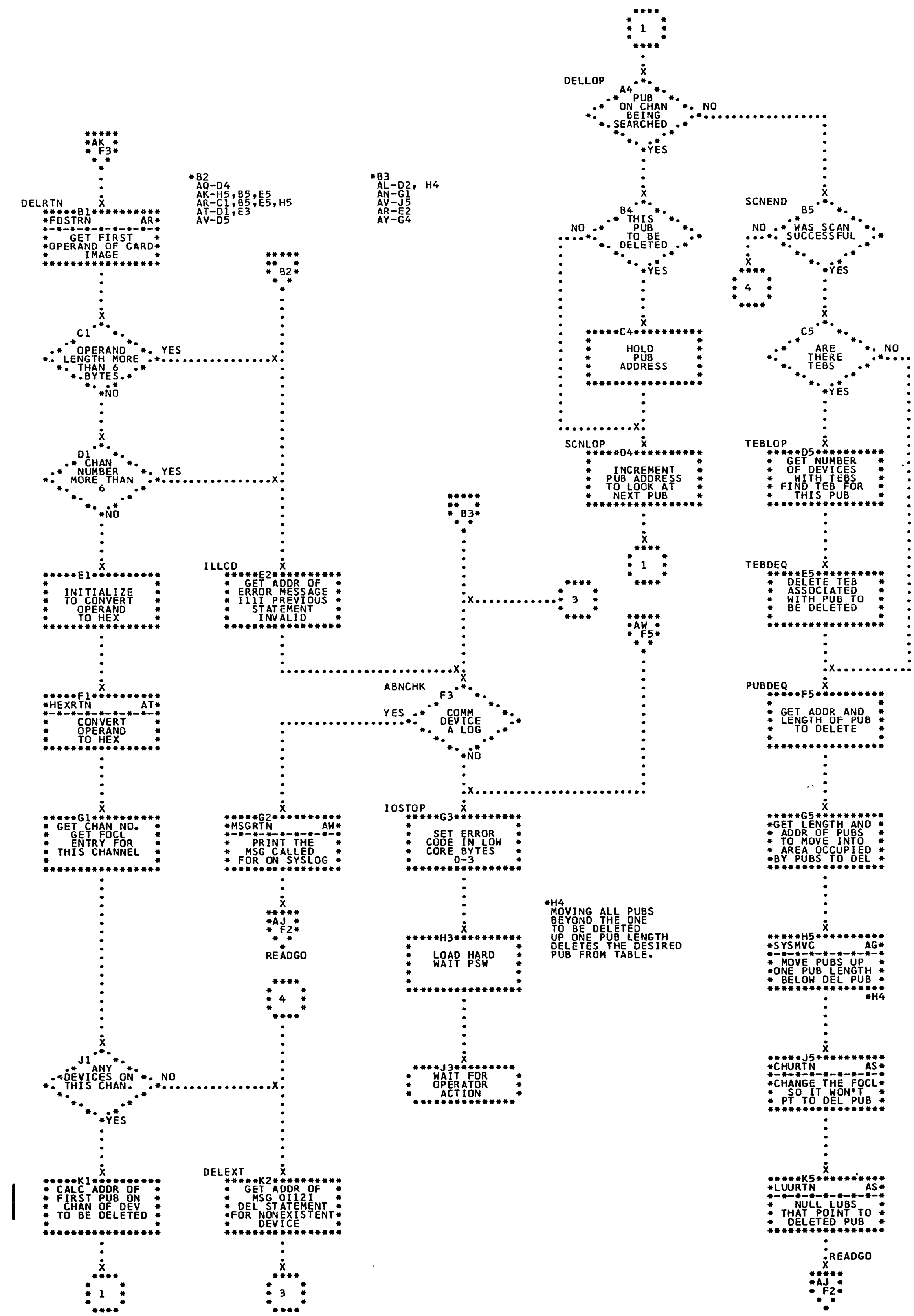


Chart AQ. Date and Time Subroutines-- $IPLRT2; Refer to IPL, Chart 02

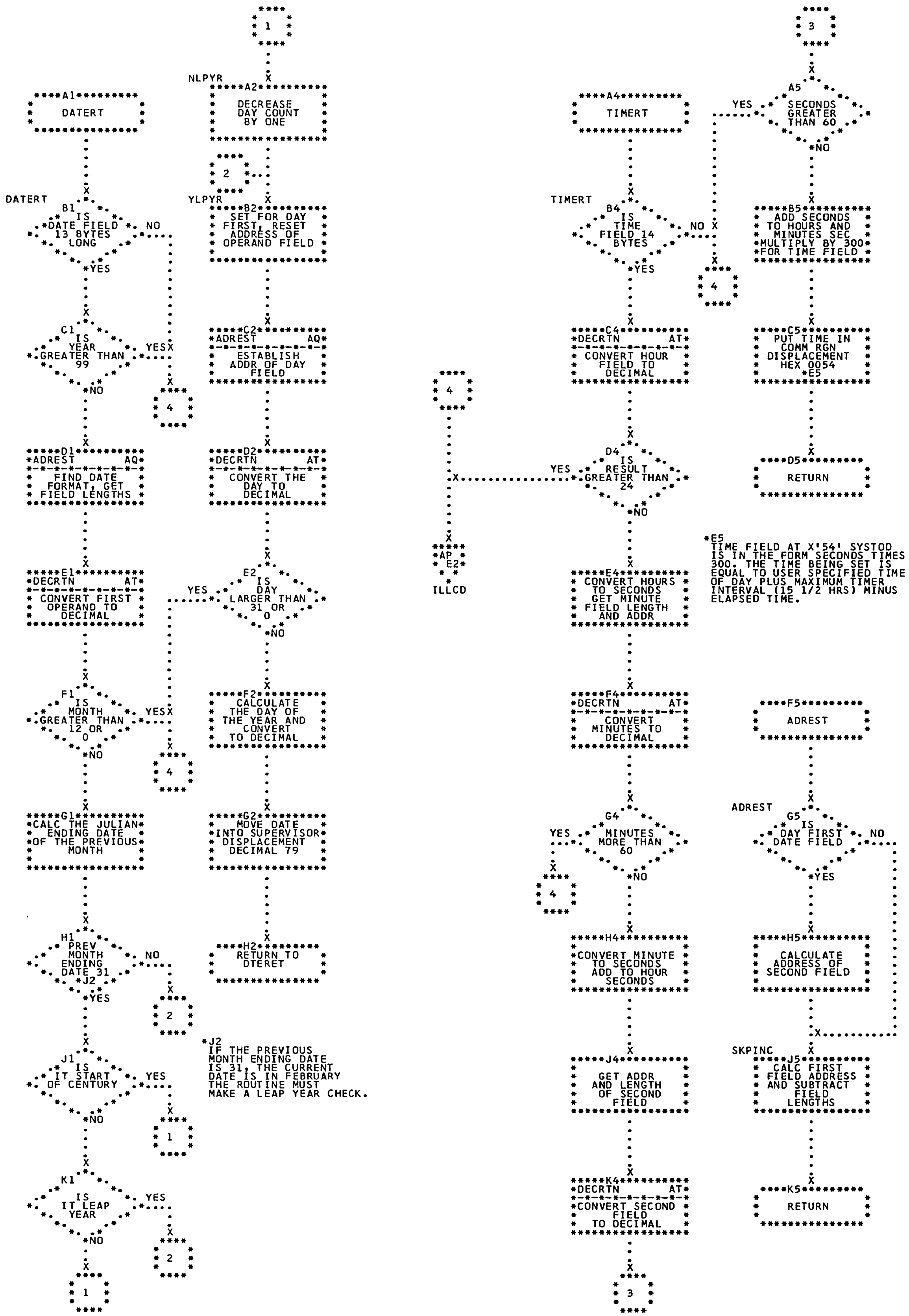


Chart AR. Analyze Device Type-- $IPLRT2; Refer to IPL, Chart 02

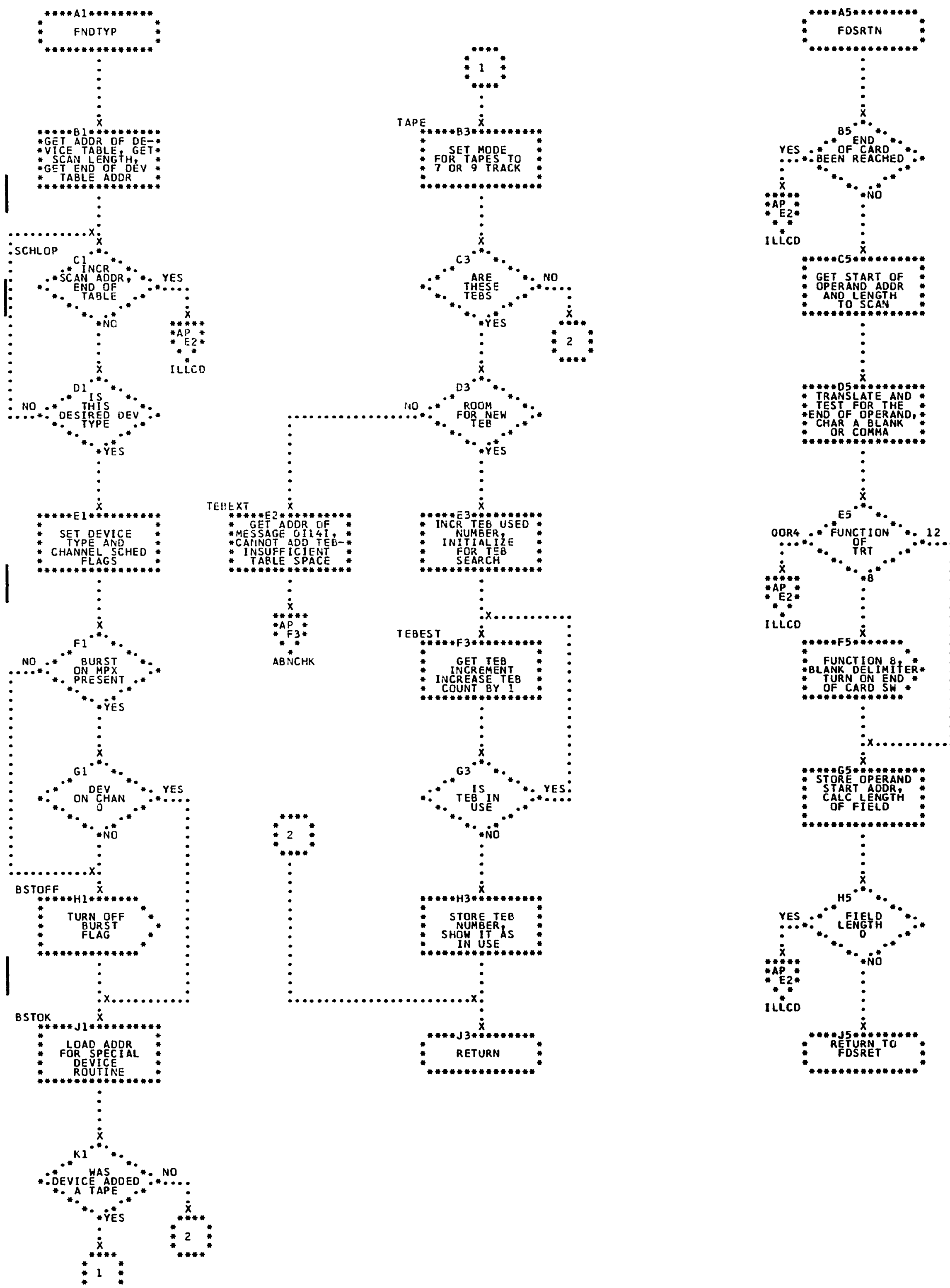


Chart AS. Update FOCL and LUB Entry-- $IPLRT2; Refer to IPL, Chart 02

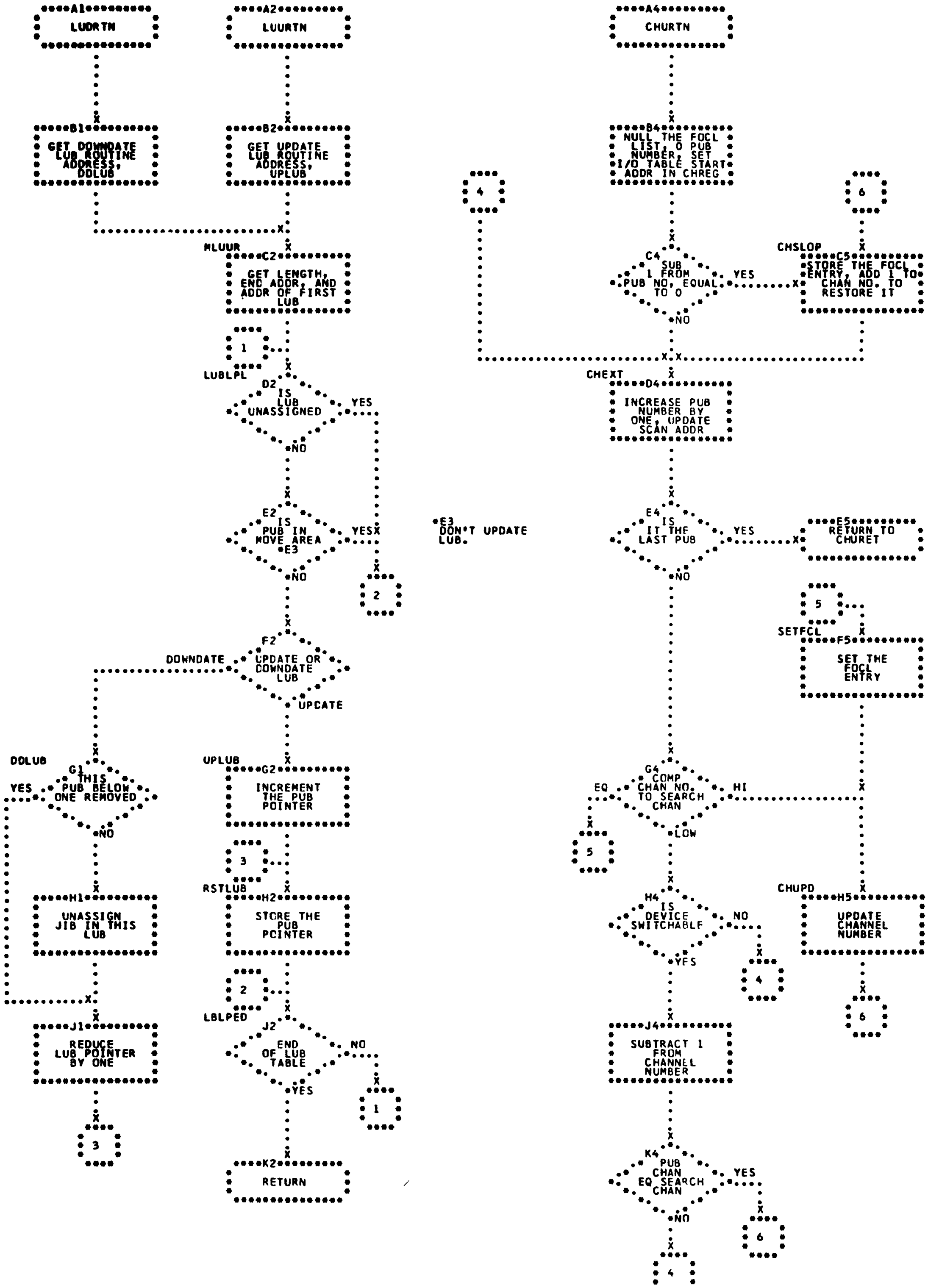


Chart AT. Check Device Assignment and Convert Decimal to Hexadecimal-- $IPLRT2; Refer to IPL, Chart 02

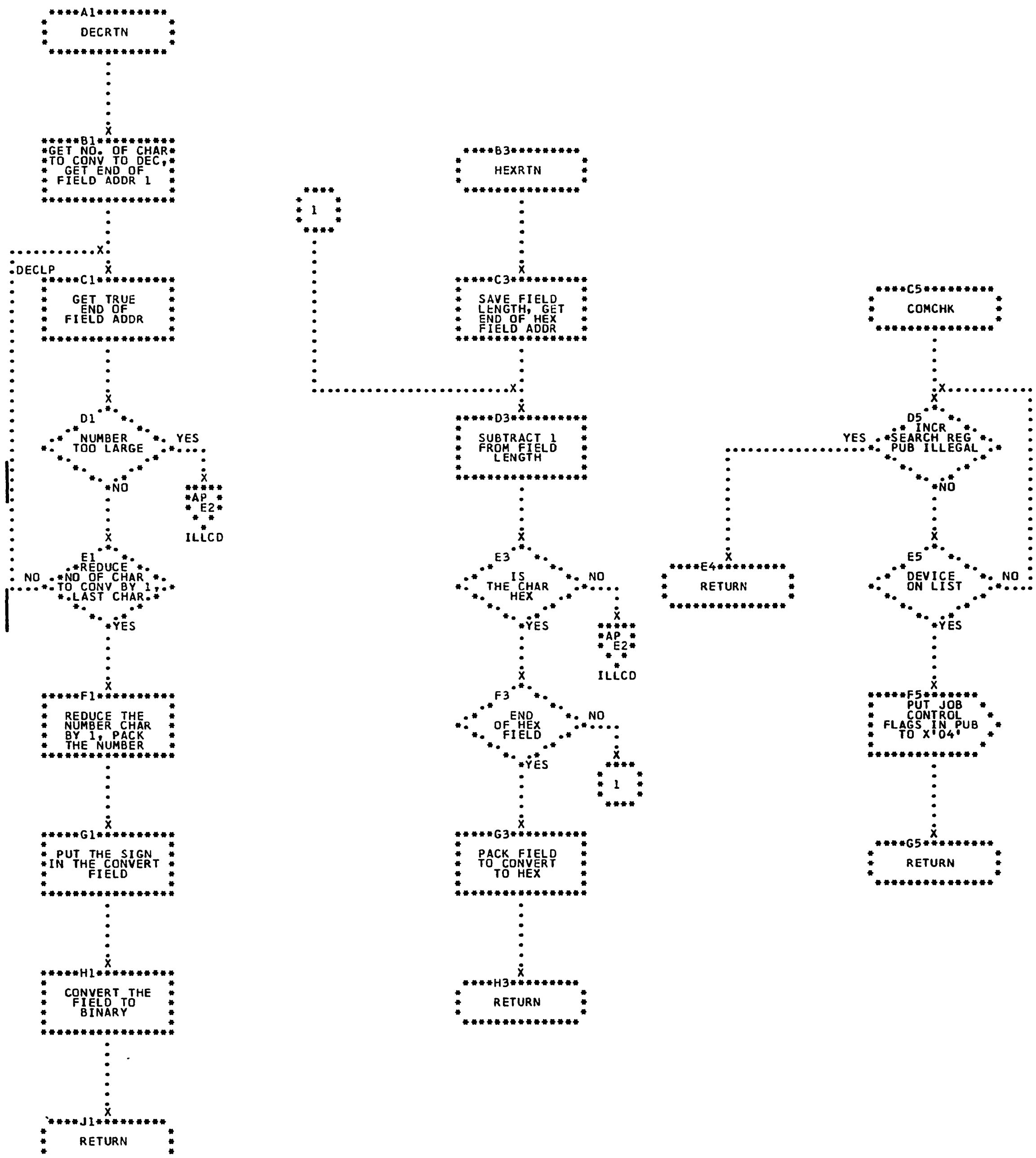


Chart AU. Build PUB Table-- $IPLRT2; Refer to IPL, Chart 02

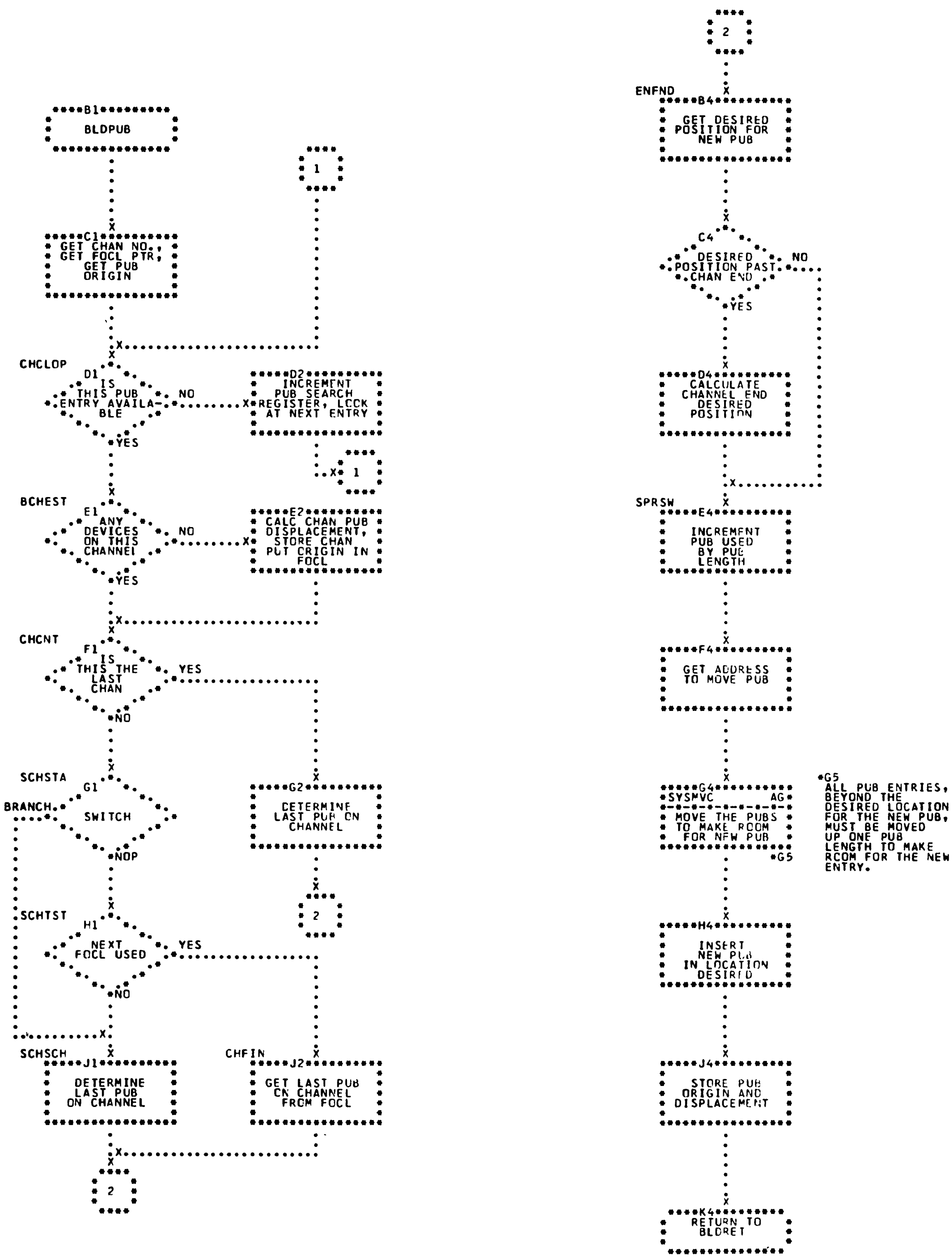


Chart AV. Find PUB and Test Delimiter Subroutines--
$IPLRT2; Refer to IPL, Chart 02

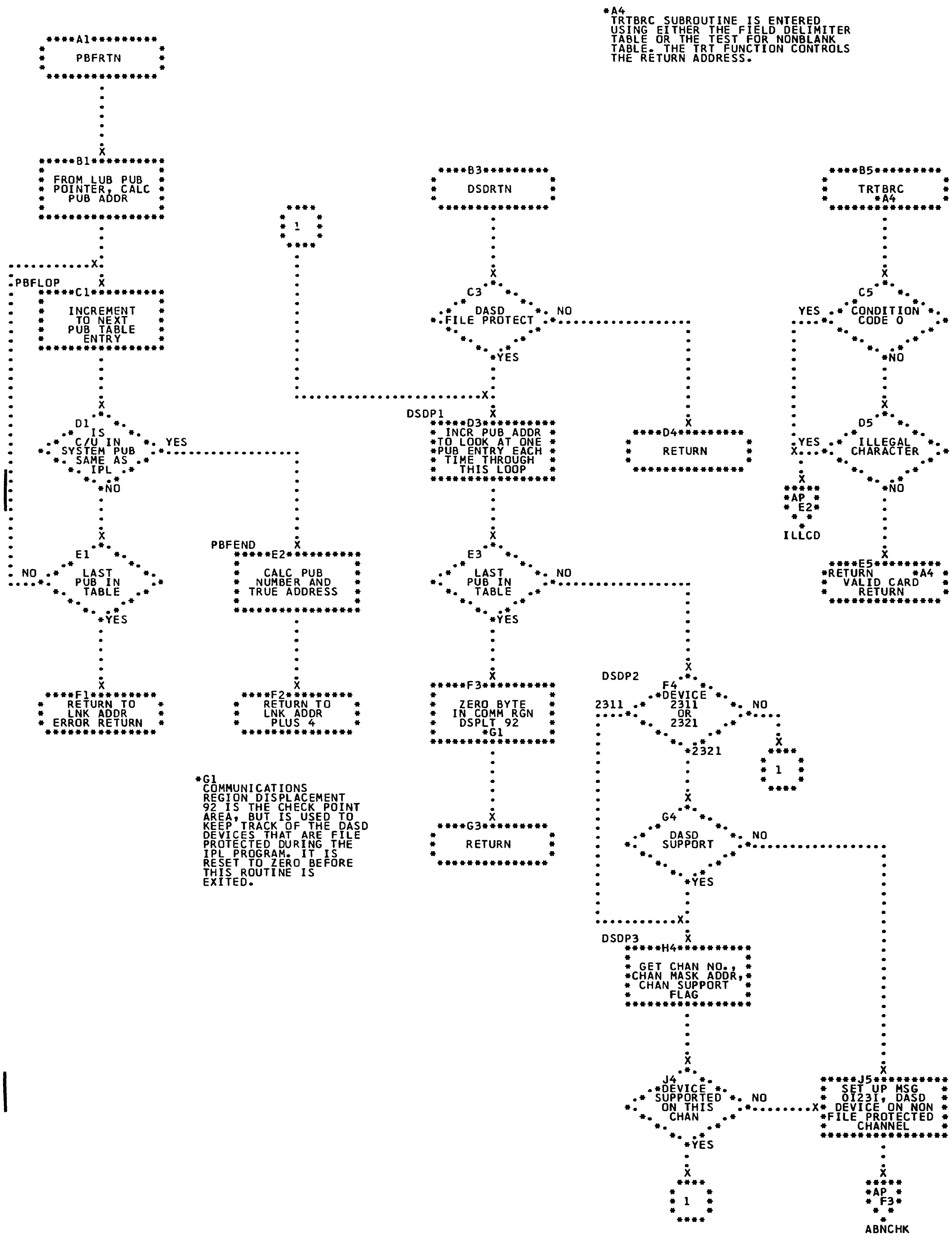


Chart AW. Reorder MPX Channel LUB's and PUB's and 1052 I/O Subroutines -- $IPLRT2; Refer to IPL, Chart 02

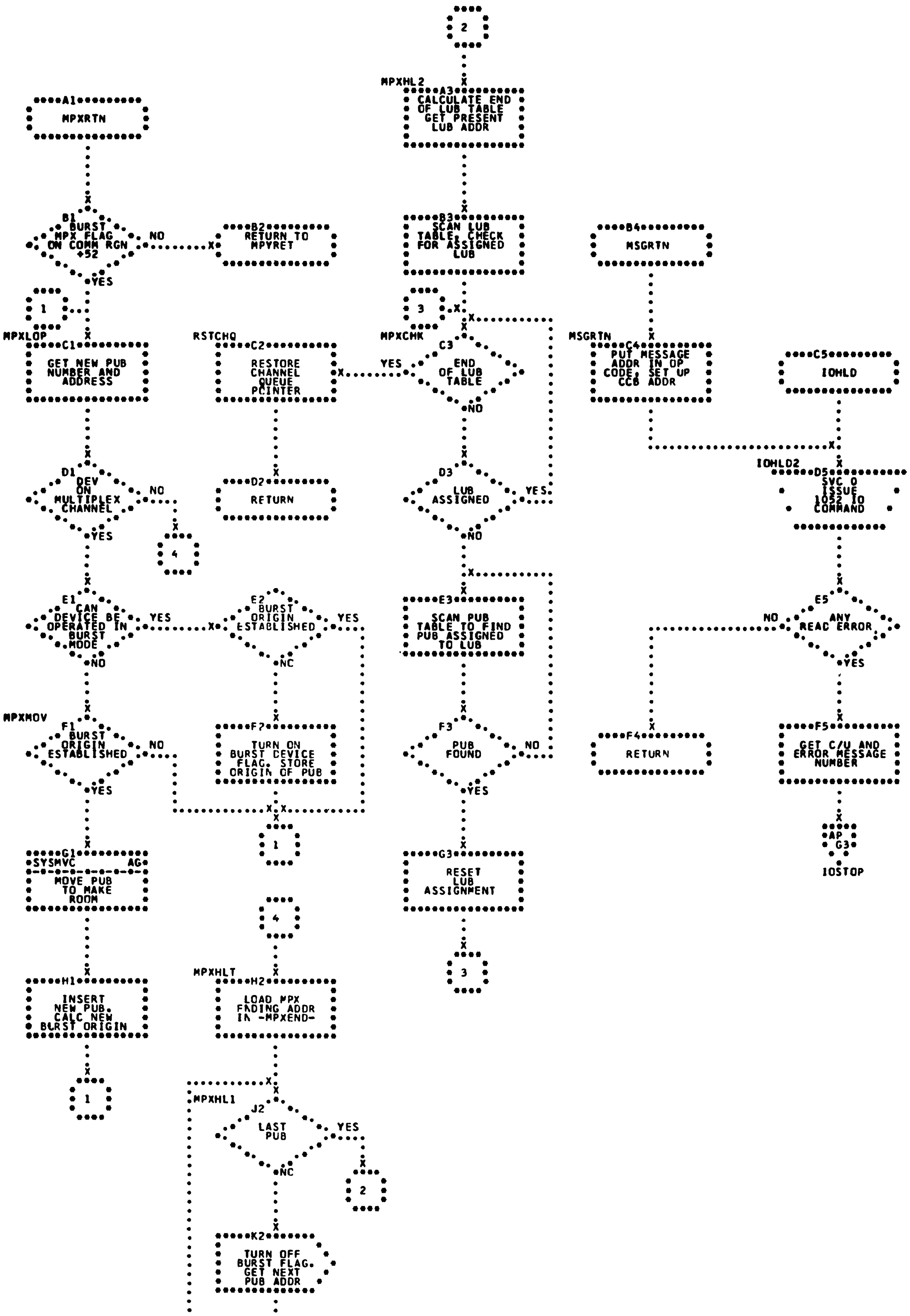


Chart AX. Set Job Control Flags-- $IPLRT2; Refer to IPL, Chart 02

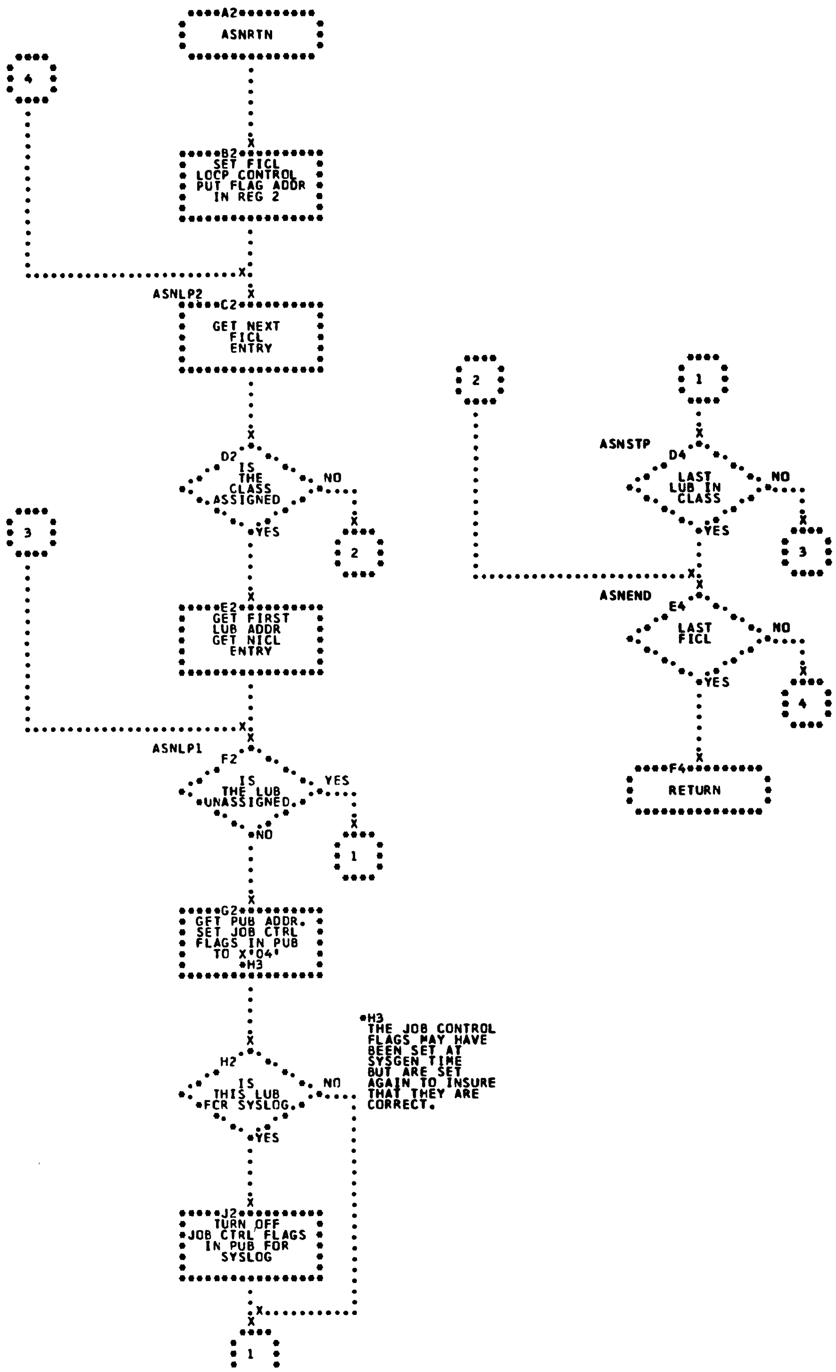
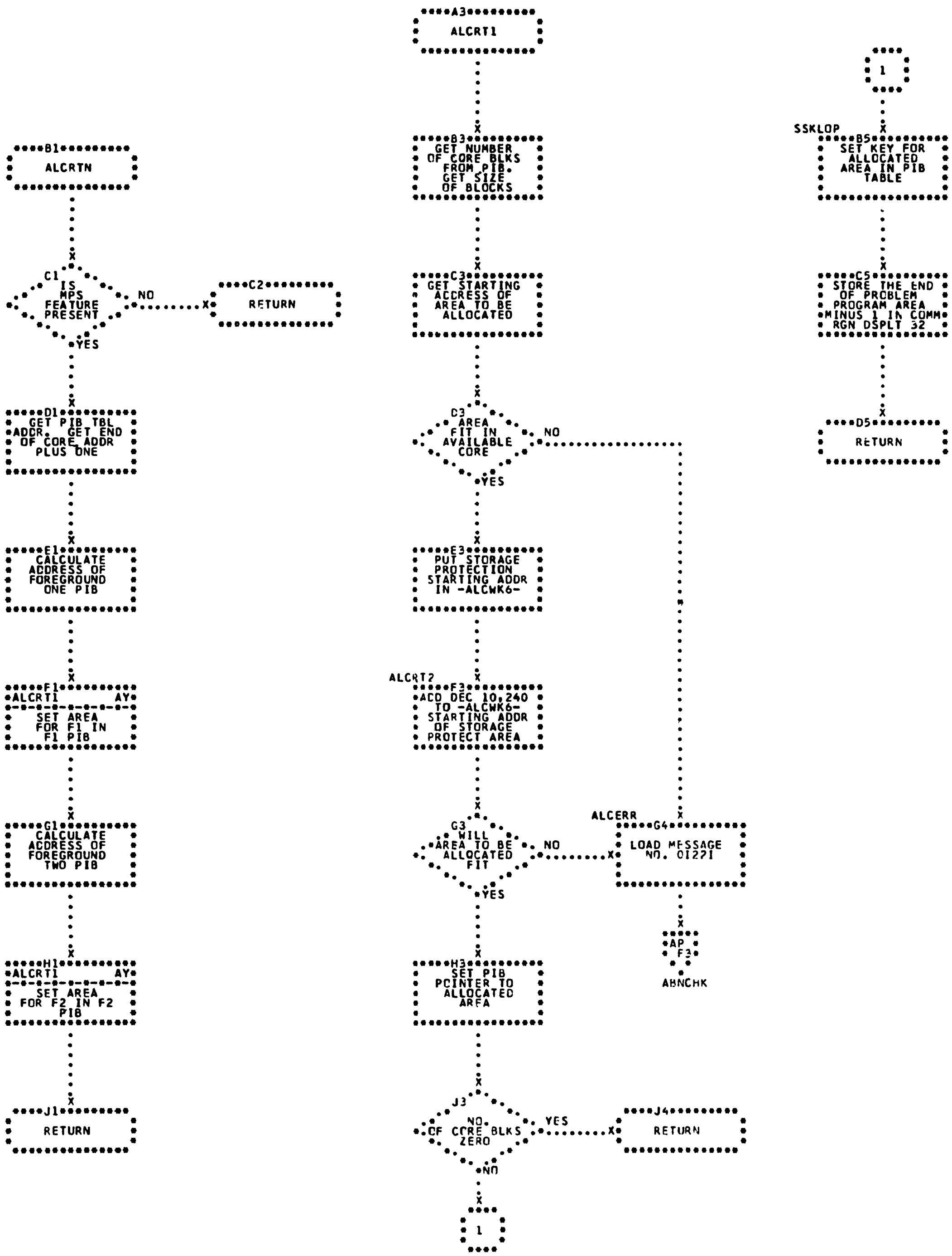
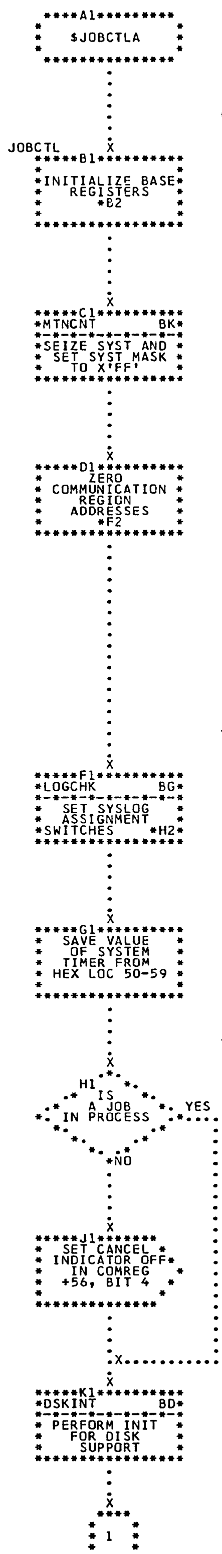


Chart AY. Allocate Main Storage Subroutine-- $IPLRT2; Refer to IPL, Chart 02



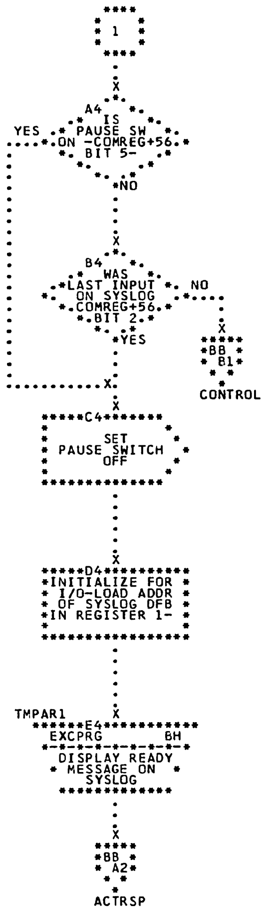
## Chart BA. Initialization-- $JOBCTLA; Refer to Job Control, Chart 03




*B2
1. BASRG1-SET TO THE JOB CONTROL ORIGIN -BASVCT-. THIS REG ALSO USED AS A COMMON BRANCH REG TO THE CONTROL STATEMENT READ ROUTINE.
2. BASRG2-POINTS TO BEGINNING ADDRESS OF OVERLAY AREA -OVRVCT-. THIS IS AREA WHERE JOB CONTROL PHASES $JOBCTLD, $JOBCTLG, AND $JOBCTLJ ARE LOADED.
3. BASRG3-SET TO BASRG2 +4096.
4. BASRG4-POINTS TO THE COMMUNICATION REGION.

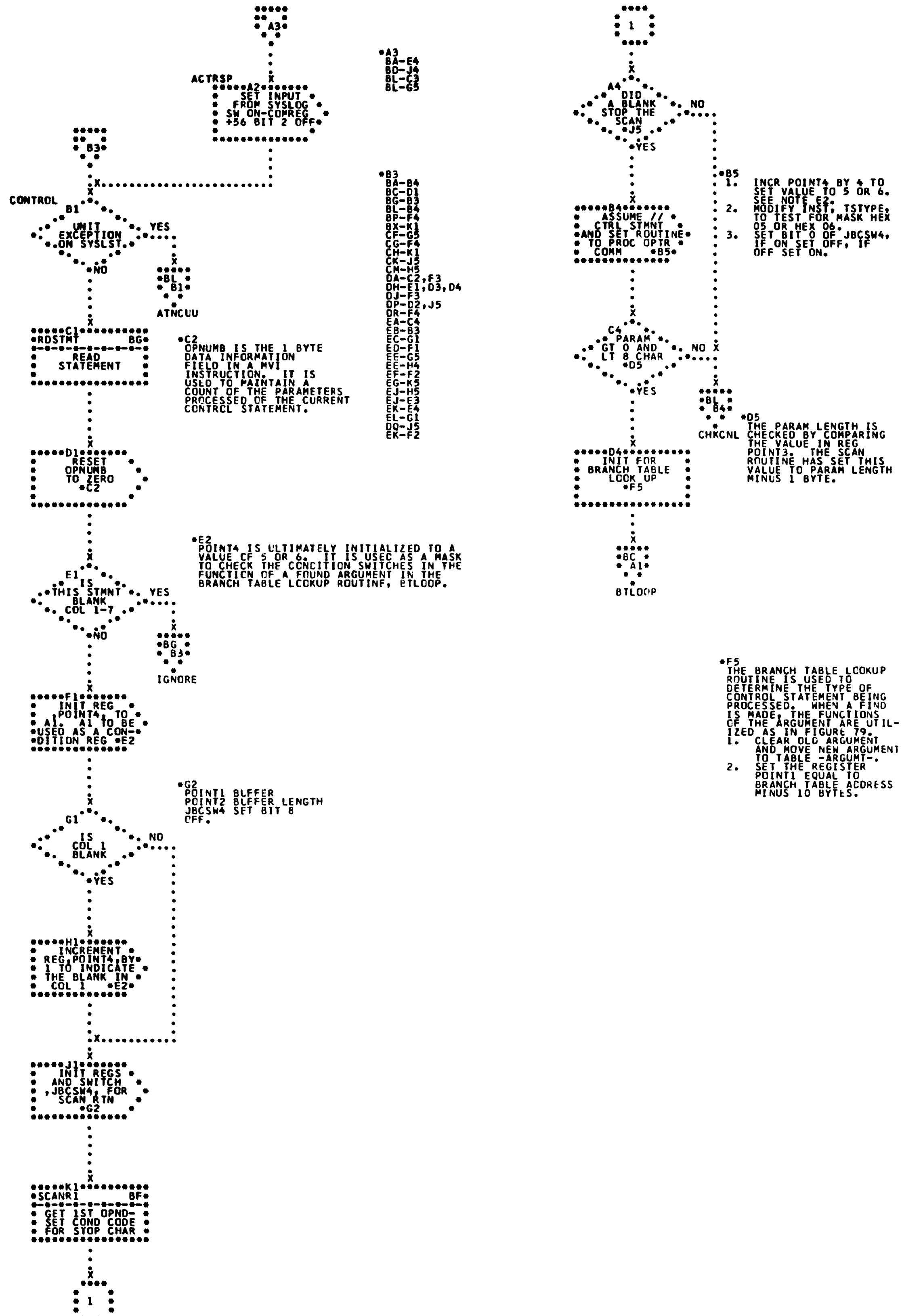
*F2
1. LENGTH OF PROBLEM PROGRAM LABEL AREA -COMREG+44-.
2. LAST CHECKPOINT NUMBER -COMREG+92-.
3. HIGHEST CORE ADDRESS OF LONGEST PROBLEM PROGRAM PHASE -COMREG+40-.

*H2
THE ASSIGNMENT FOR SYSLOG IS DETERMINED AND SWITCHES IN COMREG +56 ARE SET.

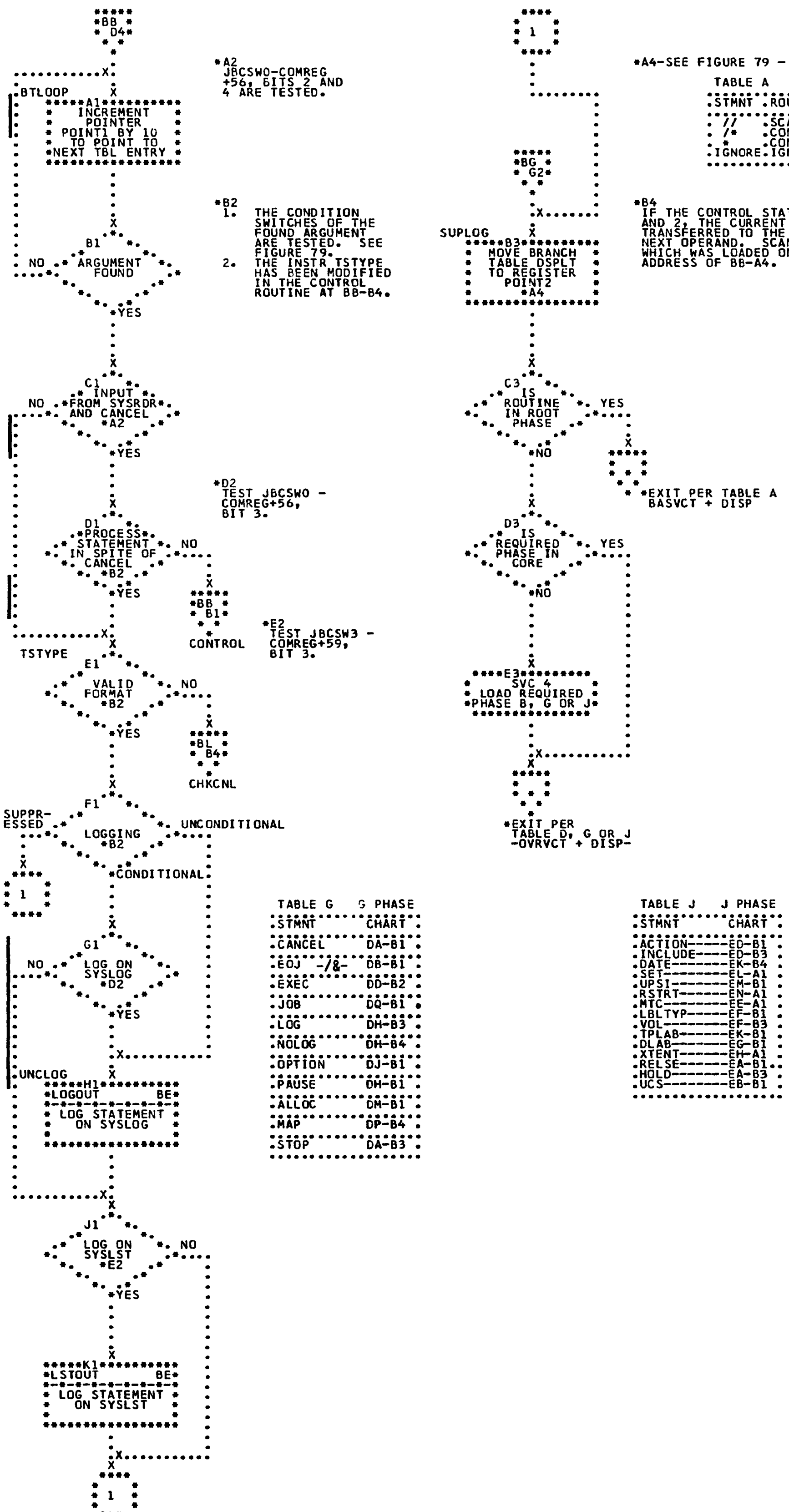



NOTE
----
THE MAIN STORAGE AREA THAT CONTAINS THIS CODING IS ALSO USED AS A 120 BYTE I/O BUFFER AREA LABELED BUFFER. CONTROL STATEMENTS ARE READ INTO THIS BUFFER AREA.

## Chart BB. Control Statement Read $JOBCTLA; Refer to Job Control, Chart 03



## Chart BC. Phase Vector Table Lookup-- $JOBCTLA; Refer to Job Control, Chart 03




*A4-SEE FIGURE 79 - FIELD 3

TABLE A ROOT PHASE(A)

| STMNT | ROUTINE | CHART |
|---|---|---|
| // | SCANR2 | BF----------SEE NOTE *B4 |
| /* | CONTROL | BB |
| * | CONTROL | BB |
| IGNORE | IGNORE | BG |

*B4
IF THE CONTROL STATEMENT STARTED WITH // IN COL 1 AND 2, THE CURRENT ARGUMENT IS //. CONTROL IS TRANSFERRED TO THE SCANR2 SUBROUTINE TO GET THE NEXT OPERAND. SCANR2 EXITS BY BRANCHING ON LINKR1 WHICH WAS LOADED ON CHART BB-K1 AND CONTAINS THE ADDRESS OF BB-A4.

TABLE D D PHASE

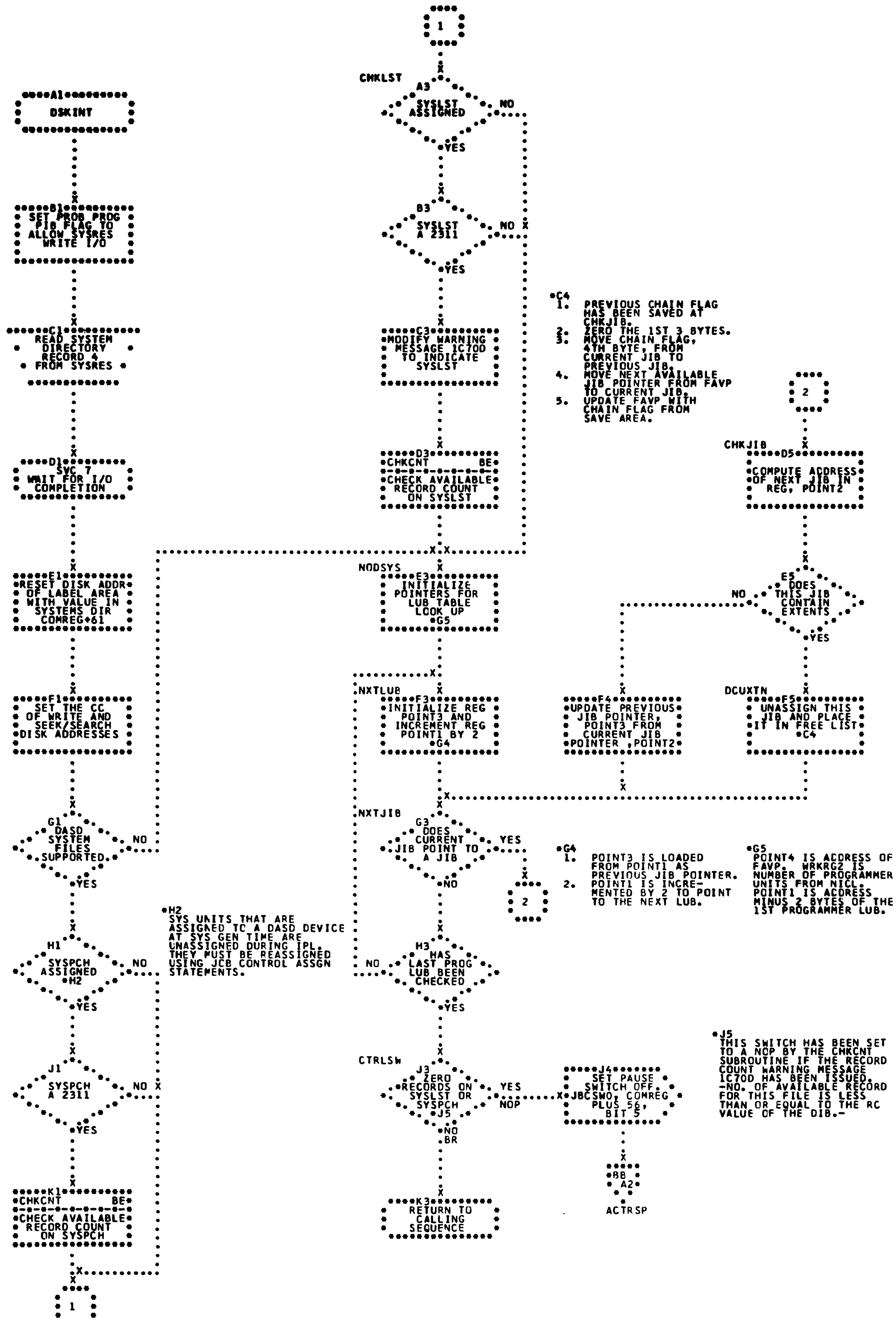
| STMNT | CHART |
|---|---|
| ASSGN | BY-B2 |
| CLOSE | BN-A1 |
| DVCDN | BU-B1 |
| DVCUP | BX-B1 |
| LISTIO | BP-B1 |
| RESET | CJ-B1 |
| UNA | BM-B1 |

TABLE G G PHASE

| STMNT | CHART |
|---|---|
| CANCEL | DA-B1 |
| EOJ -/&- | DB-B1 |
| EXEC | DD-B2 |
| JOB | DQ-B1 |
| LOG | DH-B3 |
| NOLOG | DH-B4 |
| OPTION | DJ-B1 |
| PAUSE | DH-B1 |
| ALLOC | DM-B1 |
| MAP | DP-B4 |
| STOP | DA-B3 |

TABLE J J PHASE

| STMNT | CHART |
|---|---|
| ACTION | ED-B1 |
| INCLUDE | ED-B3 |
| DATE | EK-B4 |
| SET | EL-A1 |
| UPSI | EM-B1 |
| RSTRT | EN-A1 |
| MTC | EE-A1 |
| LBLTYP | EF-B1 |
| VOL | EF-B3 |
| TPLAB | EK-B1 |
| DLAB | EG-B1 |
| XTENT | EH-A1 |
| RELSE | EA-B1 |
| HOLD | EA-B3 |
| UCS | EB-B1 |



## Chart BE. Subroutines-- $JOBCTLA (LOGOUT, MSGOUT, LSTOUT, and CHKCNT); Refer to Job Control, Charts 03-11

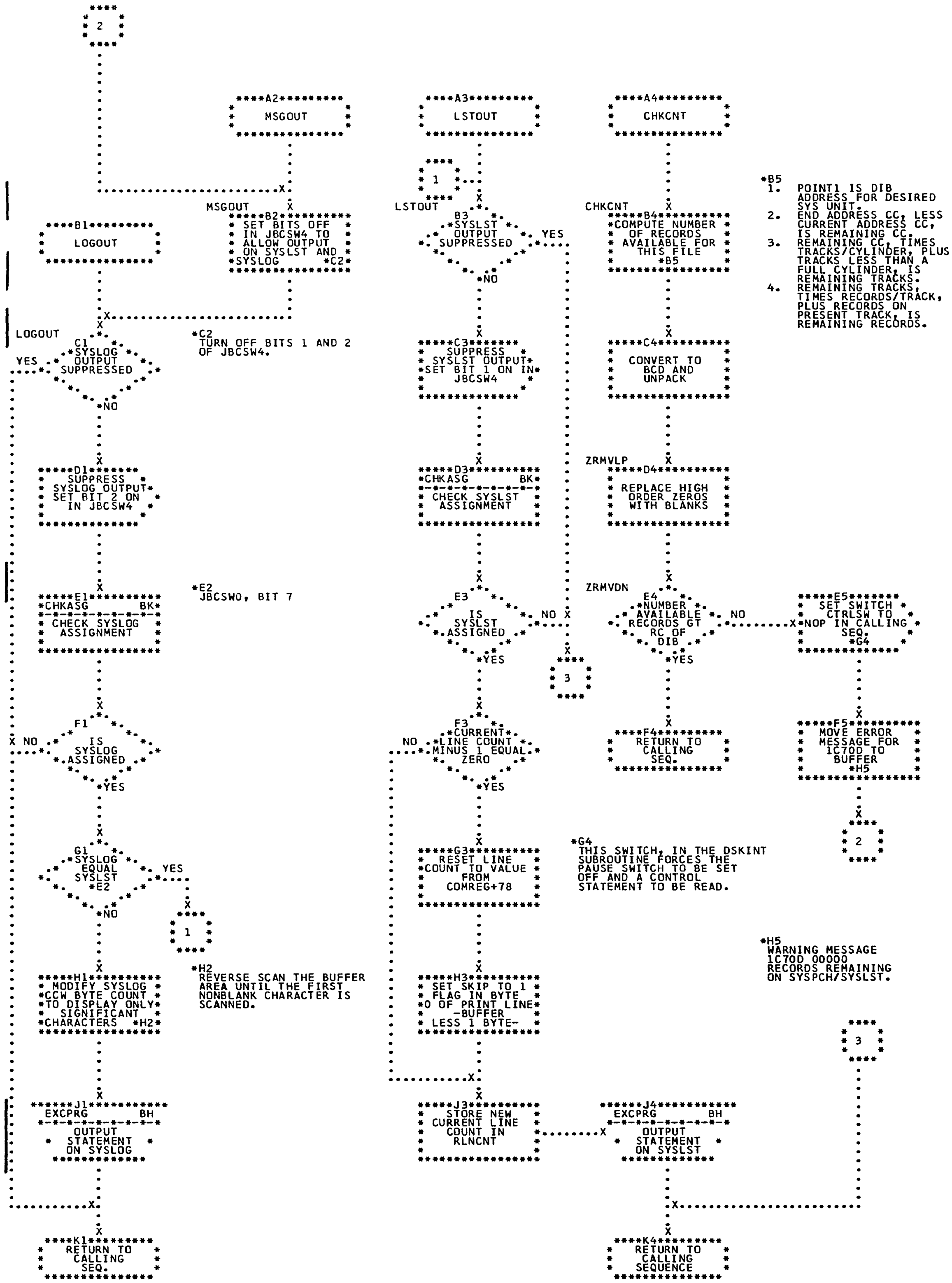


Chart BF. Subroutines-- $JOBCTLA (SCANR1, SCANR2, and SCANR3); Refer to Job Control Charts 03-11

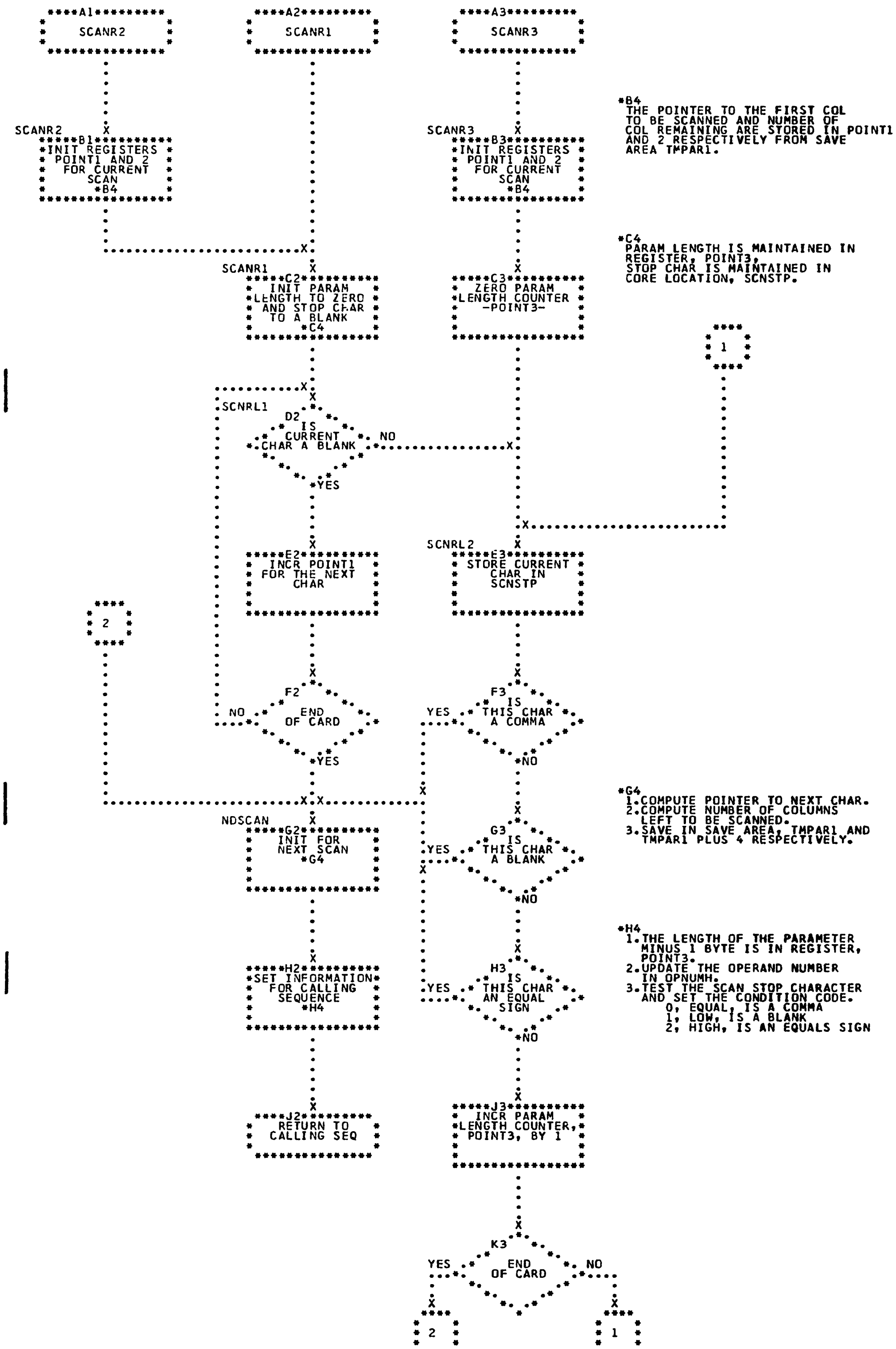
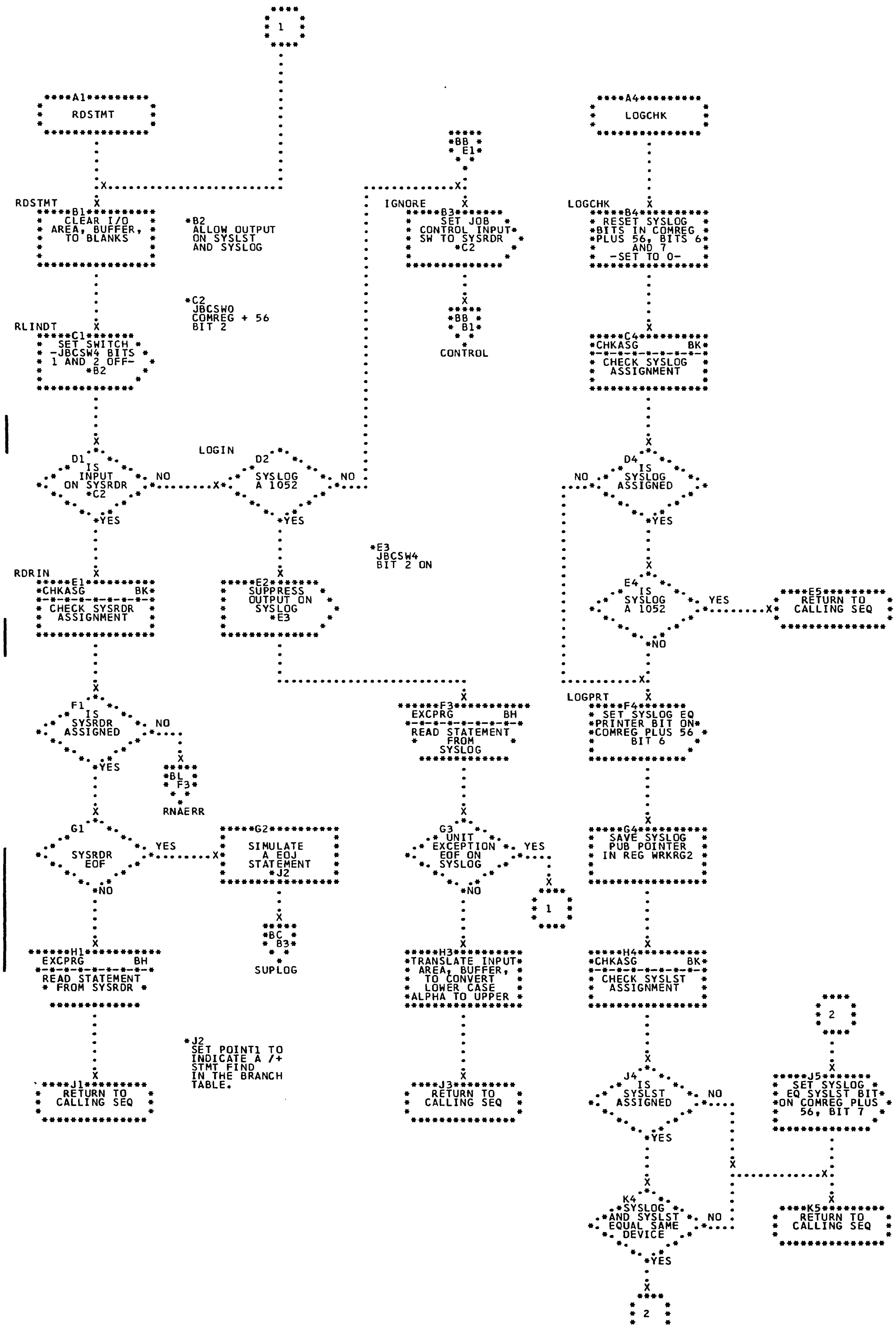


Chart BG. Subroutines-- $JOBCTLA (RDSTMT, LOGCHK); Refer to Job Control, Charts 03-11



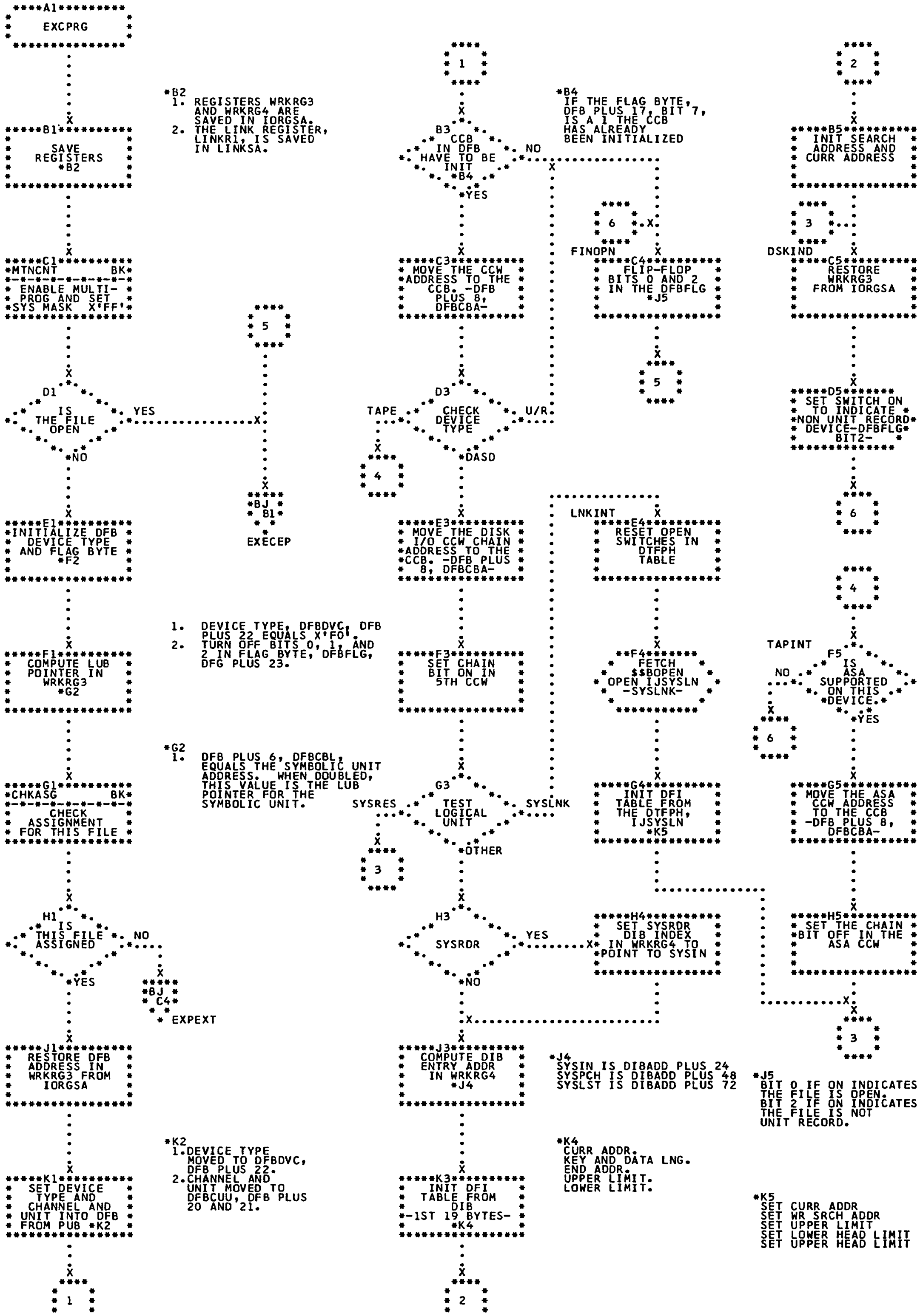

A1
EXCPRG
B1
SAVE REGISTERS *B2
*B2
1. REGISTERS WRKRG3 AND WRKRG4 ARE SAVED IN IORGSA.
2. THE LINK REGISTER, LINKR1, IS SAVED IN LINKSA.
C1
MTNCNT BK
ENABLE MULTI-PROG AND SET SYS MASK X'FF'
5
D1
IS THE FILE OPEN
YES
NO
BJ B1
EXECEP
E1
INITIALIZE DFB DEVICE TYPE AND FLAG BYTE *F2
1. DEVICE TYPE, DFBDVC, DFB PLUS 22 EQUALS X'F0'.
2. TURN OFF BITS 0, 1, AND 2 IN FLAG BYTE, DFBFLG, DFG PLUS 23.
F1
COMPUTE LUB POINTER IN WRKRG3 *G2
*G2
1. DFB PLUS 6, DFBCBL, EQUALS THE SYMBOLIC UNIT ADDRESS. WHEN DOUBLED, THIS VALUE IS THE LUB POINTER FOR THE SYMBOLIC UNIT.
G1
CHKASG BK
CHECK ASSIGNMENT FOR THIS FILE
H1
IS THIS FILE ASSIGNED
NO
YES
BJ C4
EXPEXT
J1
RESTORE DFB ADDRESS IN WRKRG3 FROM IORGSA
*K2
1.DEVICE TYPE MOVED TO DFBDVC, DFB PLUS 22.
2.CHANNEL AND UNIT MOVED TO DFBCUU, DFB PLUS 20 AND 21.
K1
SET DEVICE TYPE AND CHANNEL AND UNIT INTO DFB FROM PUB *K2
1
1
B3
CCB IN DFB HAVE TO BE INIT *B4
NO
YES
*B4
IF THE FLAG BYTE, DFB PLUS 17, BIT 7, IS A 1 THE CCB HAS ALREADY BEEN INITIALIZED
C3
MOVE THE CCW ADDRESS TO THE CCB. -DFB PLUS 8, DFBCBA-
D3
CHECK DEVICE TYPE
TAPE
U/R
DASD
4
E3
MOVE THE DISK I/O CCW CHAIN ADDRESS TO THE CCB. -DFB PLUS 8, DFBCBA-
F3
SET CHAIN BIT ON IN 5TH CCW
G3
TEST LOGICAL UNIT
SYSRES
SYSLNK
OTHER
3
H3
SYSRDR
YES
NO
J3
COMPUTE DIB ENTRY ADDR IN WRKRG4 *J4
*J4
SYSIN IS DIBADD PLUS 24
SYSPCH IS DIBADD PLUS 48
SYSLST IS DIBADD PLUS 72
K3
INIT DFI TABLE FROM DIB -1ST 19 BYTES- *K4
*K4
CURR ADDR.
KEY AND DATA LNG.
END ADDR.
UPPER LIMIT.
LOWER LIMIT.
2
6
FINOPN
C4
FLIP-FLOP BITS 0 AND 2 IN THE DFBFLG *J5
5
LNKINT
E4
RESET OPEN SWITCHES IN DTFPH TABLE
F4
FETCH $$BOPEN OPEN IJSYSLN -SYSLNK-
G4
INIT DFI TABLE FROM THE DTFPH, IJSYSLN *K5
H4
SET SYSRDR DIB INDEX IN WRKRG4 TO POINT TO SYSIN
2
B5
INIT SEARCH ADDRESS AND CURR ADDRESS
3
DSKIND
C5
RESTORE WRKRG3 FROM IORGSA
D5
SET SWITCH ON TO INDICATE NON UNIT RECORD DEVICE-DFBFLG BIT2-
6
4
TAPINT
F5
IS ASA SUPPORTED ON THIS DEVICE
NO
YES
6
G5
MOVE THE ASA CCW ADDRESS TO THE CCB -DFB PLUS 8, DFBCBA-
H5
SET THE CHAIN BIT OFF IN THE ASA CCW
3
*J5
BIT 0 IF ON INDICATES THE FILE IS OPEN.
BIT 2 IF ON INDICATES THE FILE IS NOT UNIT RECORD.
*K5
SET CURR ADDR
SET WR SRCH ADDR
SET UPPER LIMIT
SET LOWER HEAD LIMIT
SET UPPER HEAD LIMIT

Chart BJ. Subroutine-- $JOBCTLA (EXCPRG) (Part 2 of 2); Refer to Job Control, Charts 03-11

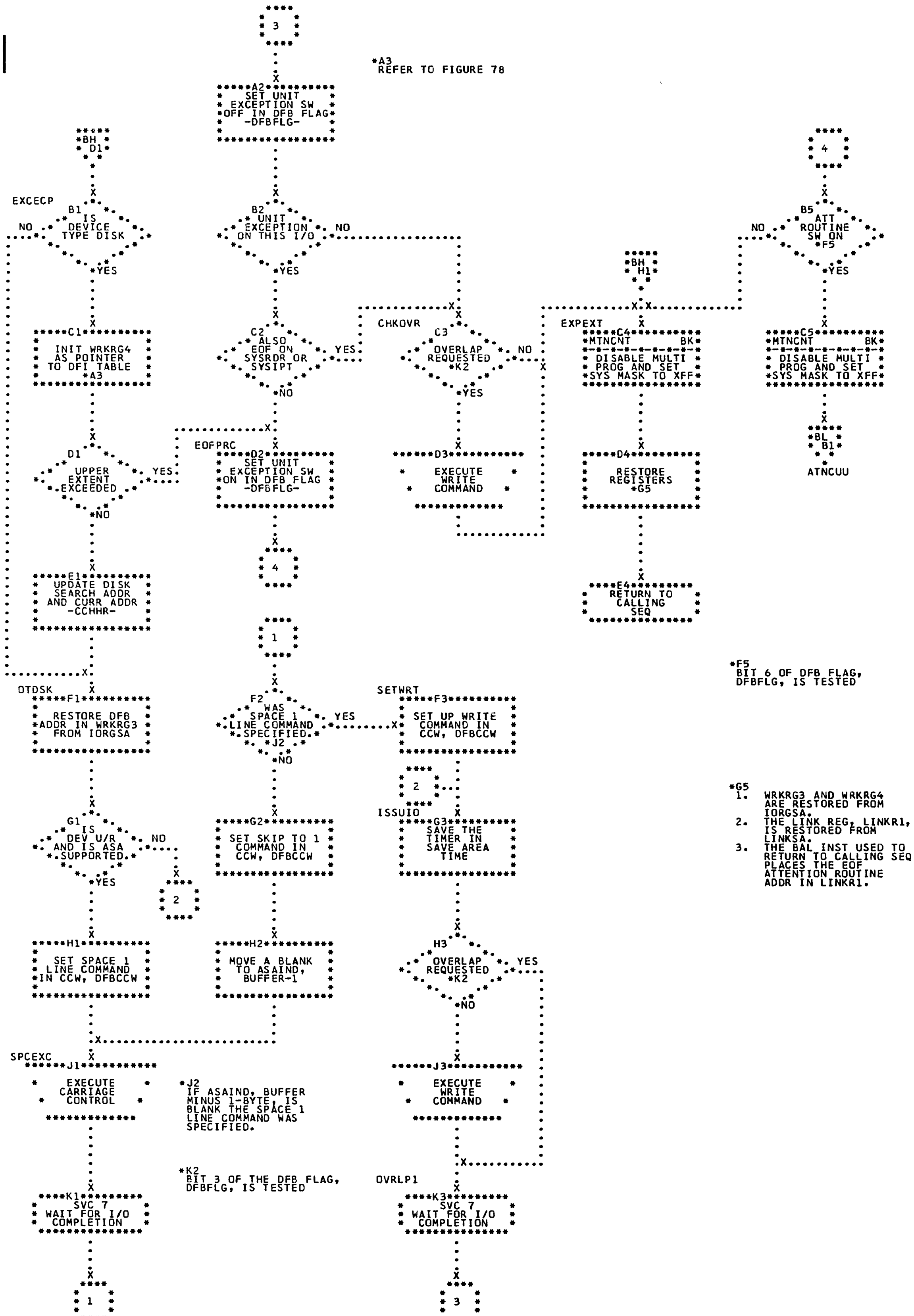


Chart BK. Subroutines-- $JOBCTLA (MTNCNT, CHKASG, CHKASG3); Refer to Job Control, Charts 03-11

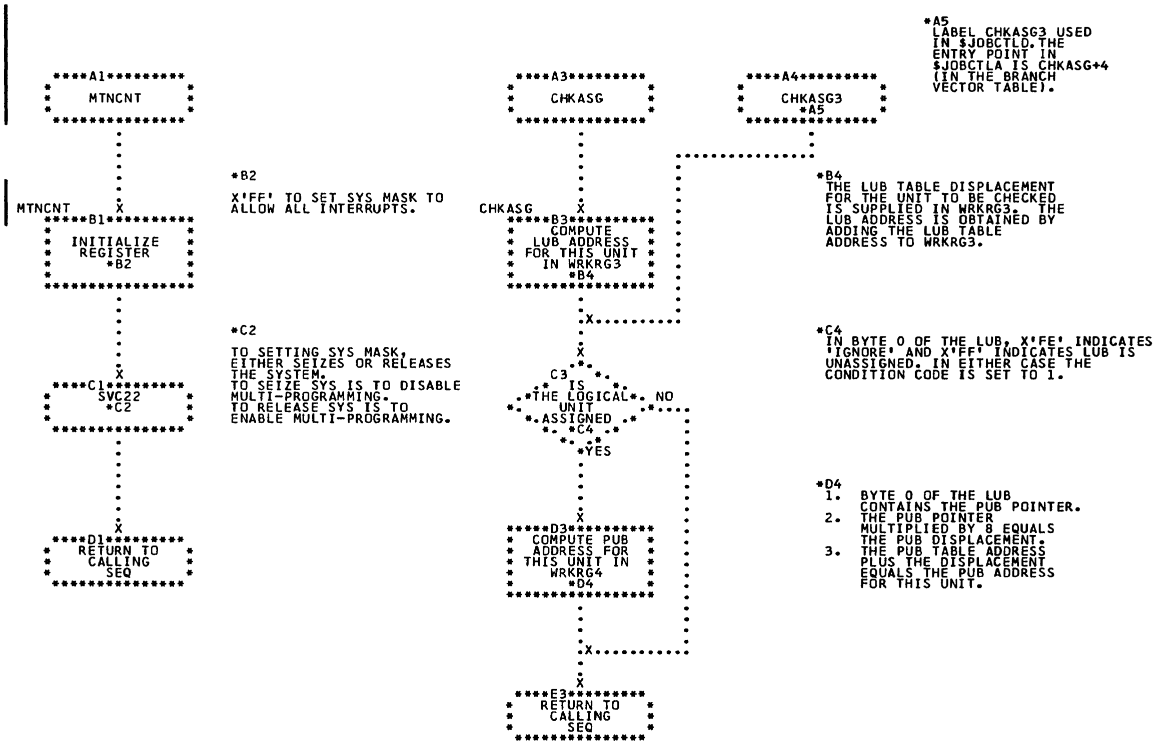


Chart BL. Error Routines-- $JOBCTLA (ATNCUU, NOEERR, OERRTN, RNAERR, NVSERR, and ERRRTN); Refer to Job Control, Charts 03-11

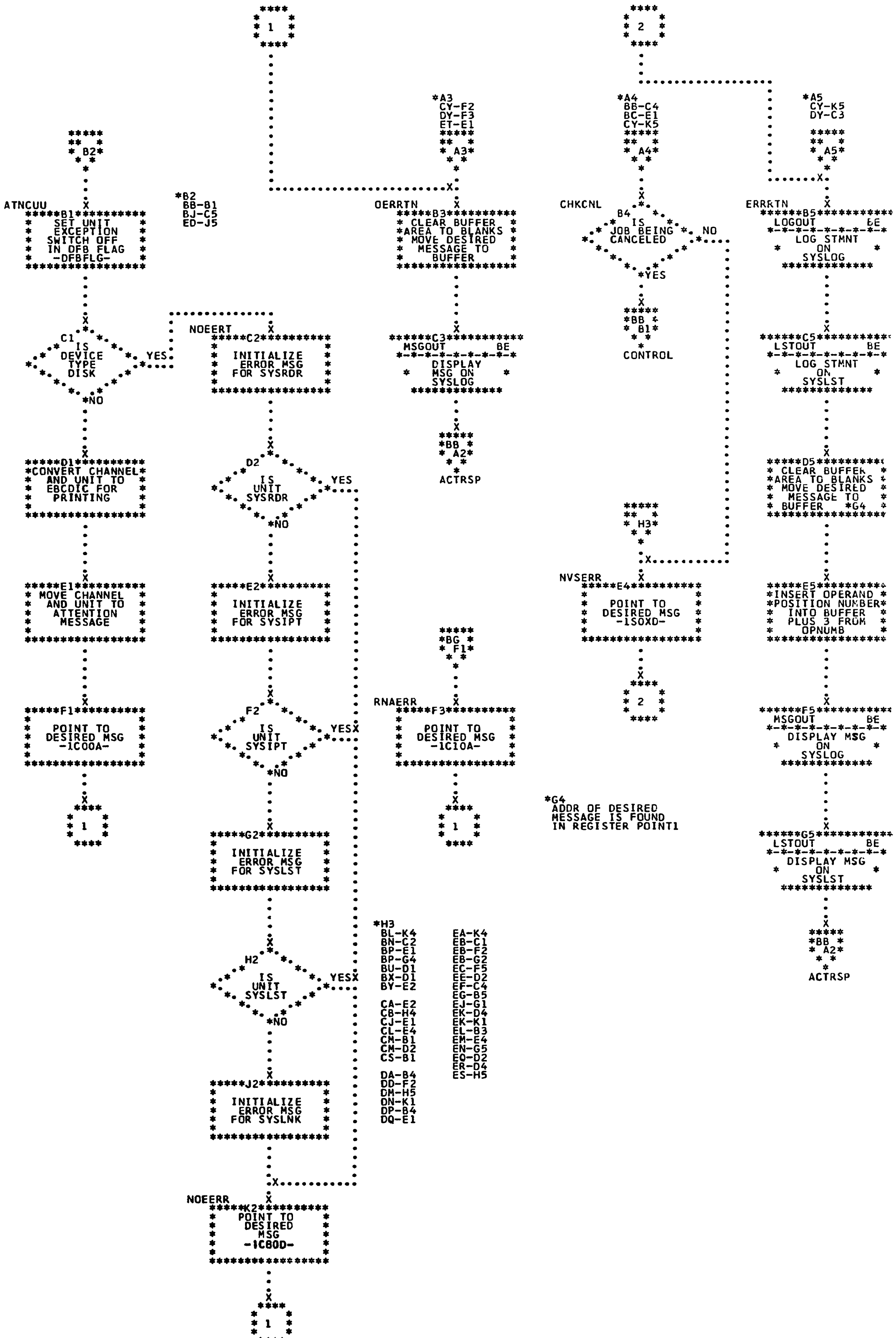


Chart BM. UNA Statement Processor-- $JOBCTLD; Refer to Job Control, Chart 05

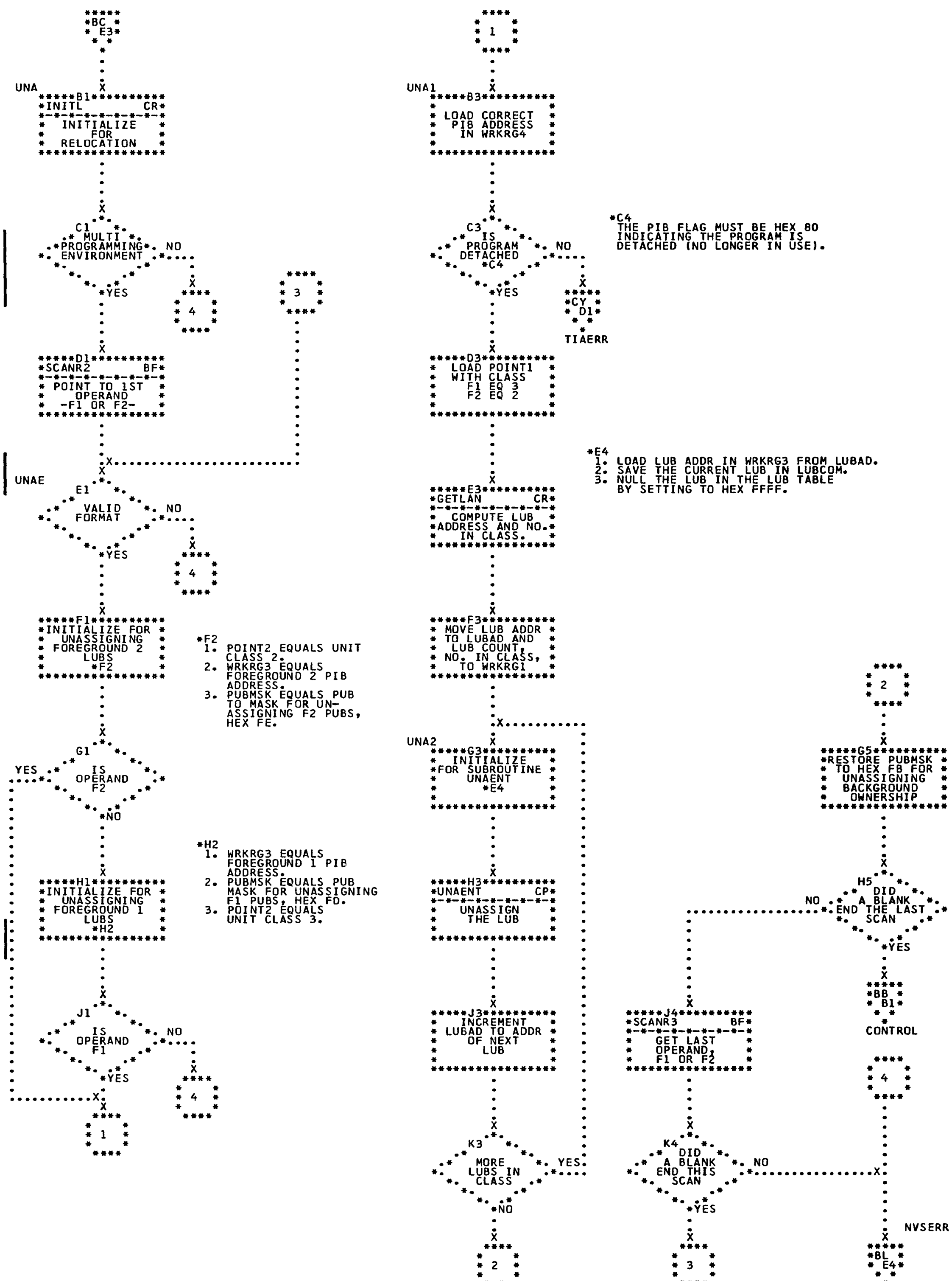
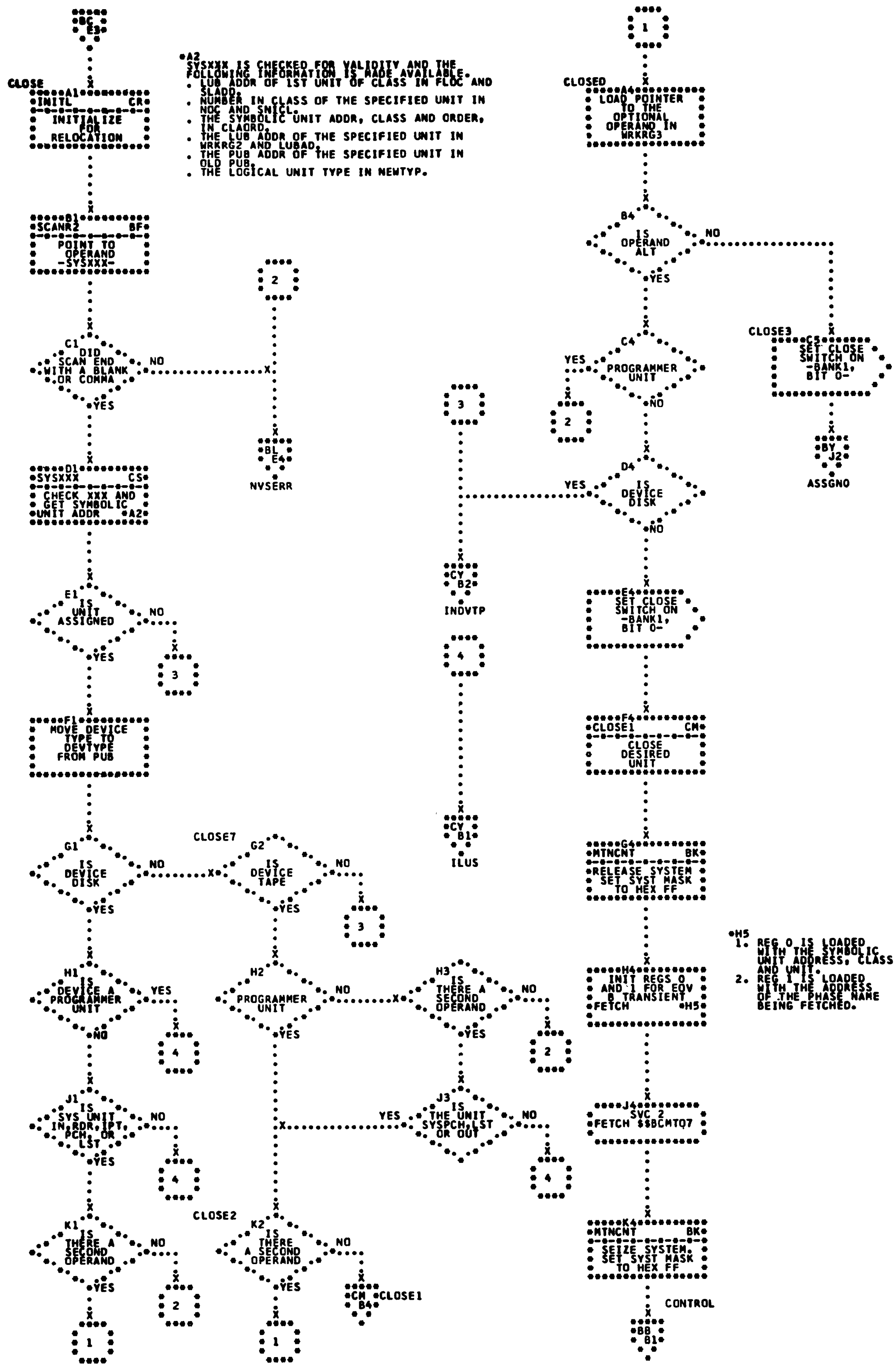


Chart BN. CLOSE Statement Processor-- $JOBCTLD; Refer to Job Control, Chart 05



## Chart BP. LISTIO Statement Processor- $JOBCTLD Scan and Terminate Routines (Part 1 of 5); Refer to Job Control, Chart 05

Connector 1

A2: MOVE THE ADDR OF THE SYS UNIT TABLE,SYSTABLE, TO THE TRANS LOC. TASYSTAB

*A3 BQ-K3 BR-G2 BT-H4 BV-J5

B2: FNDARG EX — DETERMINE TYPE OF OPERAND

→ D2

BC E3 →

LISTIO B1: INITL CR — INITIALIZE FOR RELOCATION

C1: LOAD POINT4 WITH ERROR EXIT TO BE USED IF $$BLSTIO IS LOADED *F2

*D2
1. THE OPERAND IS ANALIZED AND CONTROL IS TRANSFERRED TO THE CORRECT PROCESSING ROUTINE USING TABLE A.
2. REGISTER POINT1 CONTAINS THE LOGICAL UNIT CLASS
3. REGISTER WRKRG2 CONTAINS THE LUB COUNT.

D1: SCANR2 BF — POINT TO OPERAND

E1: DID A BLANK END THE SCAN — NO → BL E4 NVSERR; YES ↓

F1: SAVE POINT1 IN WRKFLD3. POINT1 IS THE POINTER TO THE OPERAND

*F2
IF THE $$BLSTIO TRANSIENT HAS BEEN LOADED, LIOEOJ IS THE NORMAL EXIT FROM THIS PHASE AND TNVSERR IS THE INVALID STATEMENT ERROR EXIT.

*G2
LINKR2 IS USED BY THE $$BLSTIO TRANSIENT TO RETURN CONTROL TO THE CALLING ROUTINE AT THE LABEL RETADD.

G1: LOAD LINKR2 WITH RETURN ADDRESS TO BE USED BY $$BLSTIO *G2

H1: SVCBTRNS CT — LOAD AND INIT. $$BLSTIO *G2

RETADD J1: MTNCTN BK — SEIZE SYSTEM AND SET SYSTEM MASK TO HEX FF

K1: MOVE THE OPERAND TO THE TRANSIENT AREA IN LOCATION TARG

→ 1

TABLE A

| OPERAND | ROUTINE | CHART |
|---|---|---|
| SYS | LIOL | BQA1 |
| PROG | LIOL | BQA1 |
| F1 | LIOL | BQA1 |
| F2 | LIOL | BQA1 |
| ALL | LIOL | BQA1 |
| UNITS | UNITS | BRA1 |
| CUU | LIOCUU | BSA1 |
| DOWN | DOWN | BTB3 |
| UA | UA | BTB1 |
| SYSXXX | LIOSYX | BSB3 |

A3 →

LIOEOJ A4: ZERO WRKRG3 *F2

*A5
ENTERED FROM $$BLSTIO SUBROUTINES AS AN ERROR EXIT. POINT4 IS USED AS A BRANCH REG.

A5 →

LIOEOJ1 B4: SAVE THE BRANCH TABLE DISPLACEMENT IN LOC COUNT FROM WRKRG3

TNVSERR B5: LOAD WRKRG3 WITH THE BRANCH TABLE DISPLACEMENT OF THE NVSERR RTN

C4: SET THE B-TRANSIENT ACTIVE SWITCH, TRANSN,OFF

D4: MTNCTN BK — RELEASE SYSTEM

E4: SVC 11 RELEASE THE TRANSIENT AREA

F4: MTNCTN BK — SEIZE SYSTEM

*G5
EXIT BY TABLE B

G4: LOAD WRKRG3 WITH DISPLACEMENT FROM SAVE AREA, COUNT

→ G5

TABLE B

| DISPLACE-MENT | ROUTINE | CHART |
|---|---|---|
| HEX00 | CONTROL | BB-B1 |
| HEX1A | NVSERR | BL-E4 |

Chart BQ. LISTIO Statement Processor- $JOBCTLD (SYS, PROG, F1, F2, or ALL Operand Routine; Part 2 of 5) Refer to Job Control, Chart 05

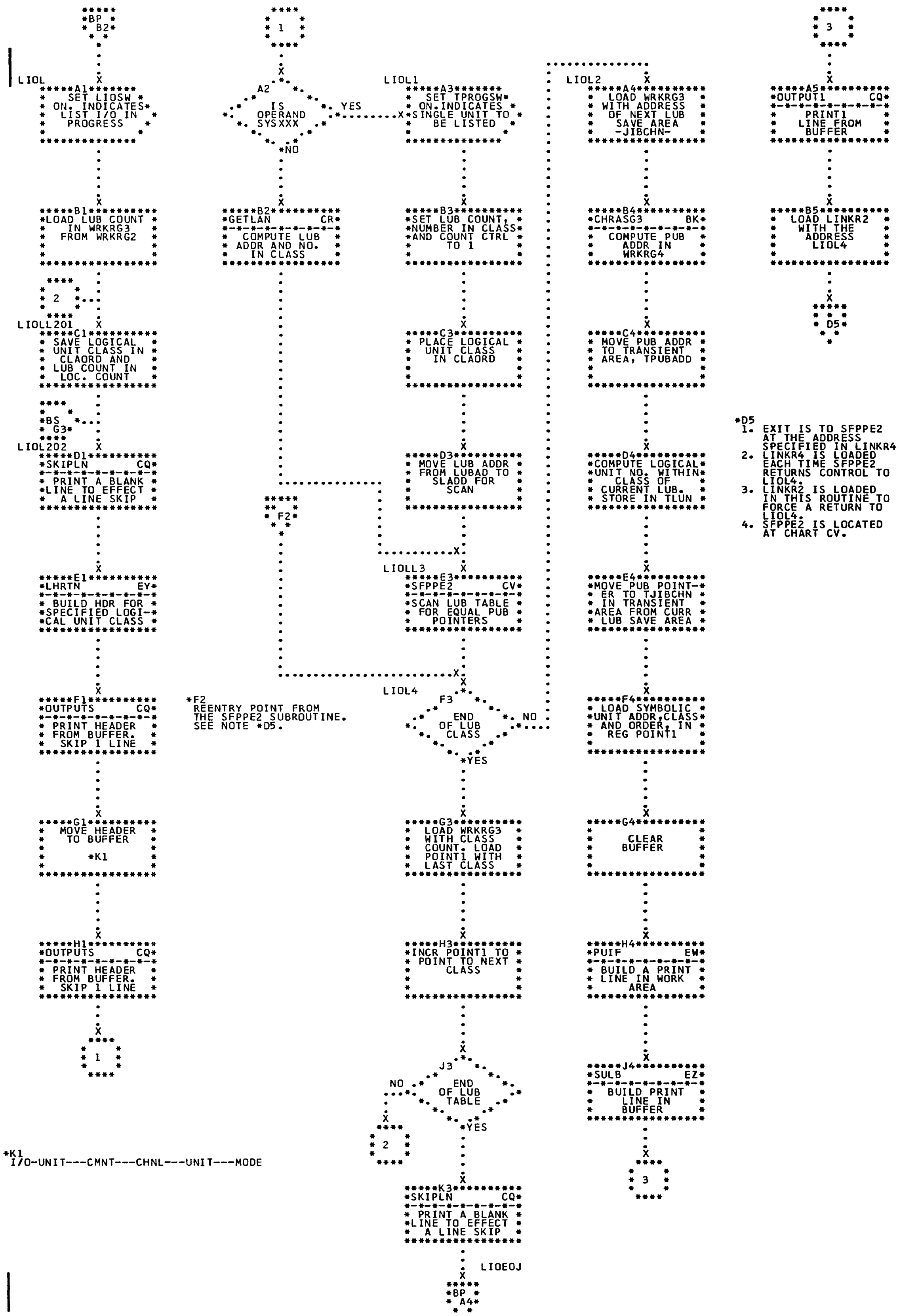


Chart BR. LISTIO Statement Processor- $JOBCTLD UNIT Operand Routine (Part 3 of 5); Refer to Job Control, Chart 05

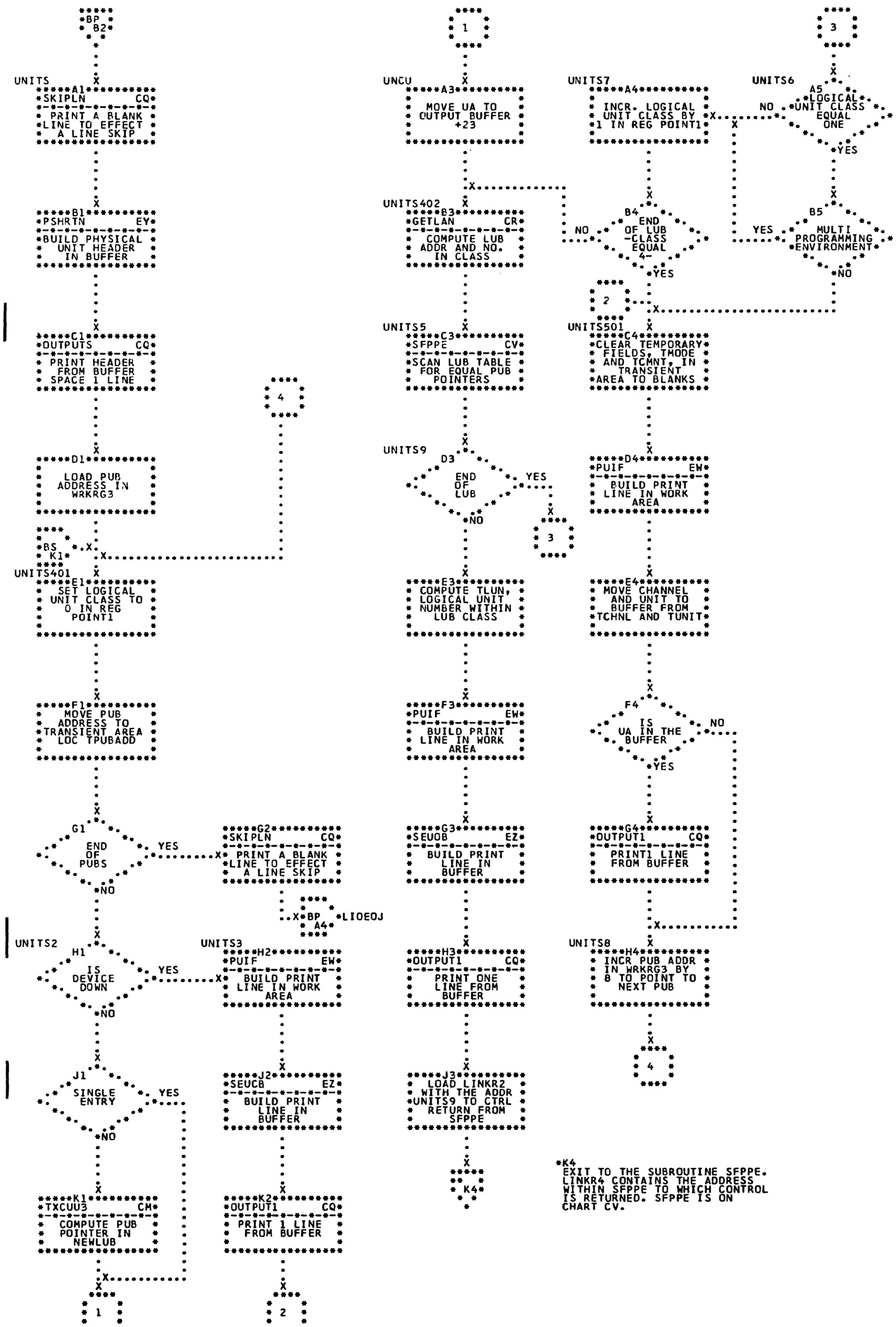


Chart BS. LISTIO Statement Processor- $JOBCTLD (CUU or SYSXXX Operand Routine; Part 4 of 5); Refer to Job Control, Chart 05

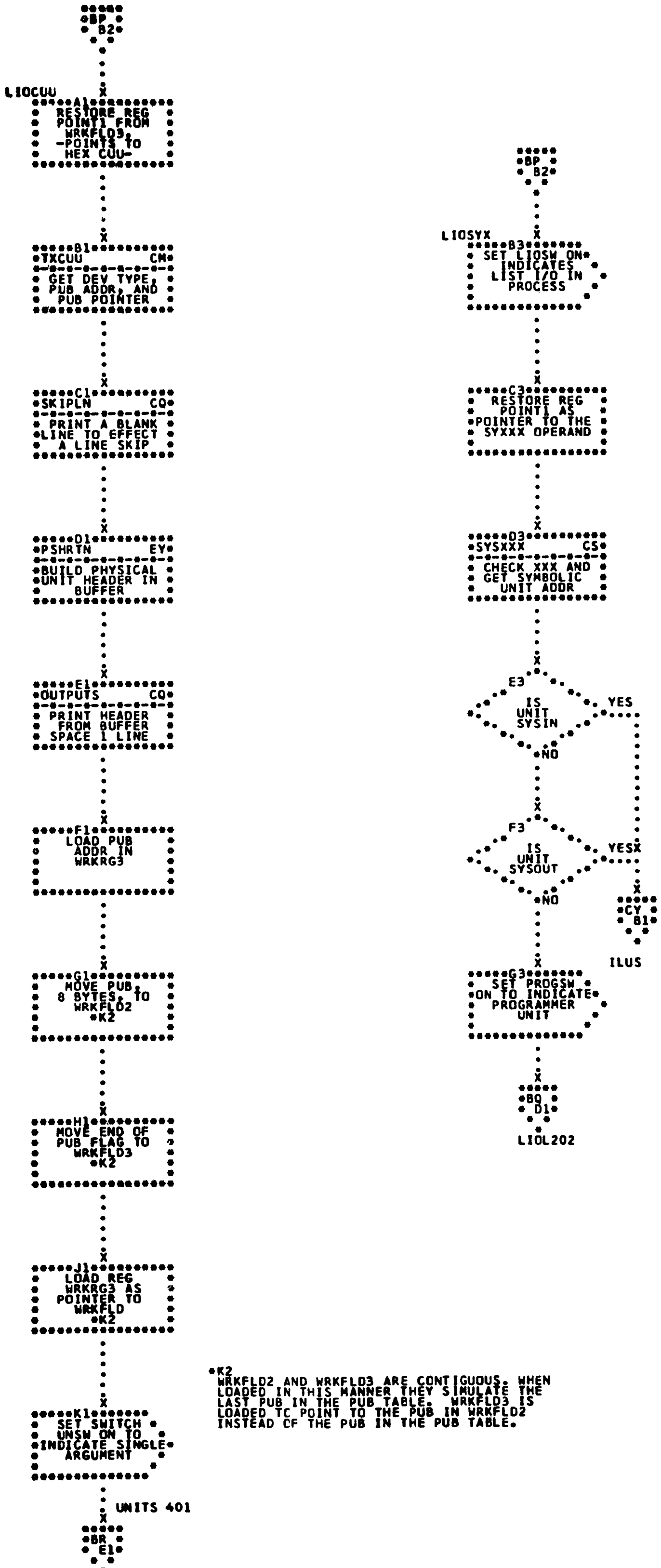


Chart BT. LISTIO Statement Processor- $JOBCTLD (UA and Down Operand Routines; Part 5 of 5) Refer to Job Control, Chart 05

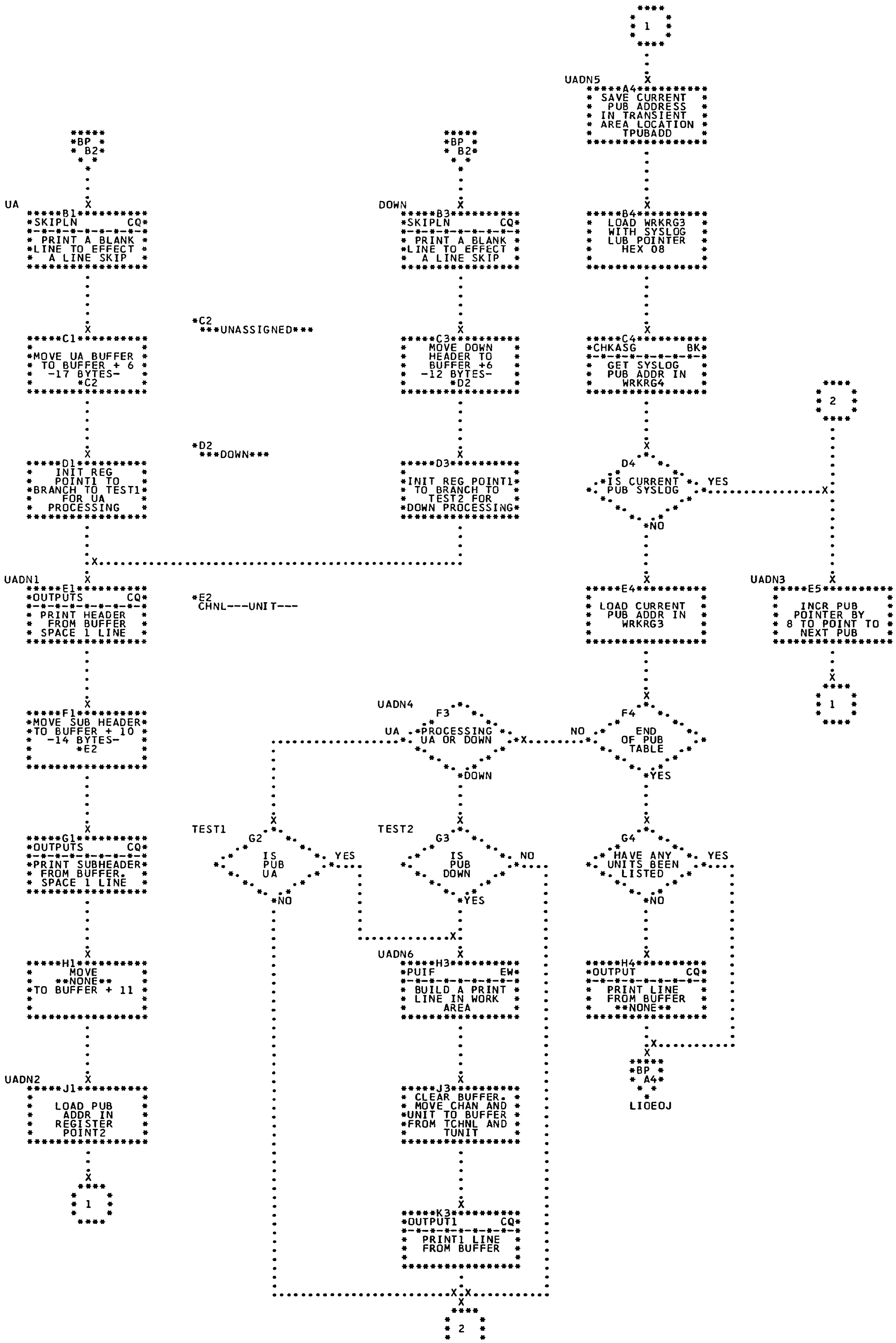


Chart BU. DVCDN Statement Processor- $JOBCTLD (Part 1 of 3); Refer to Job Control, Chart 05

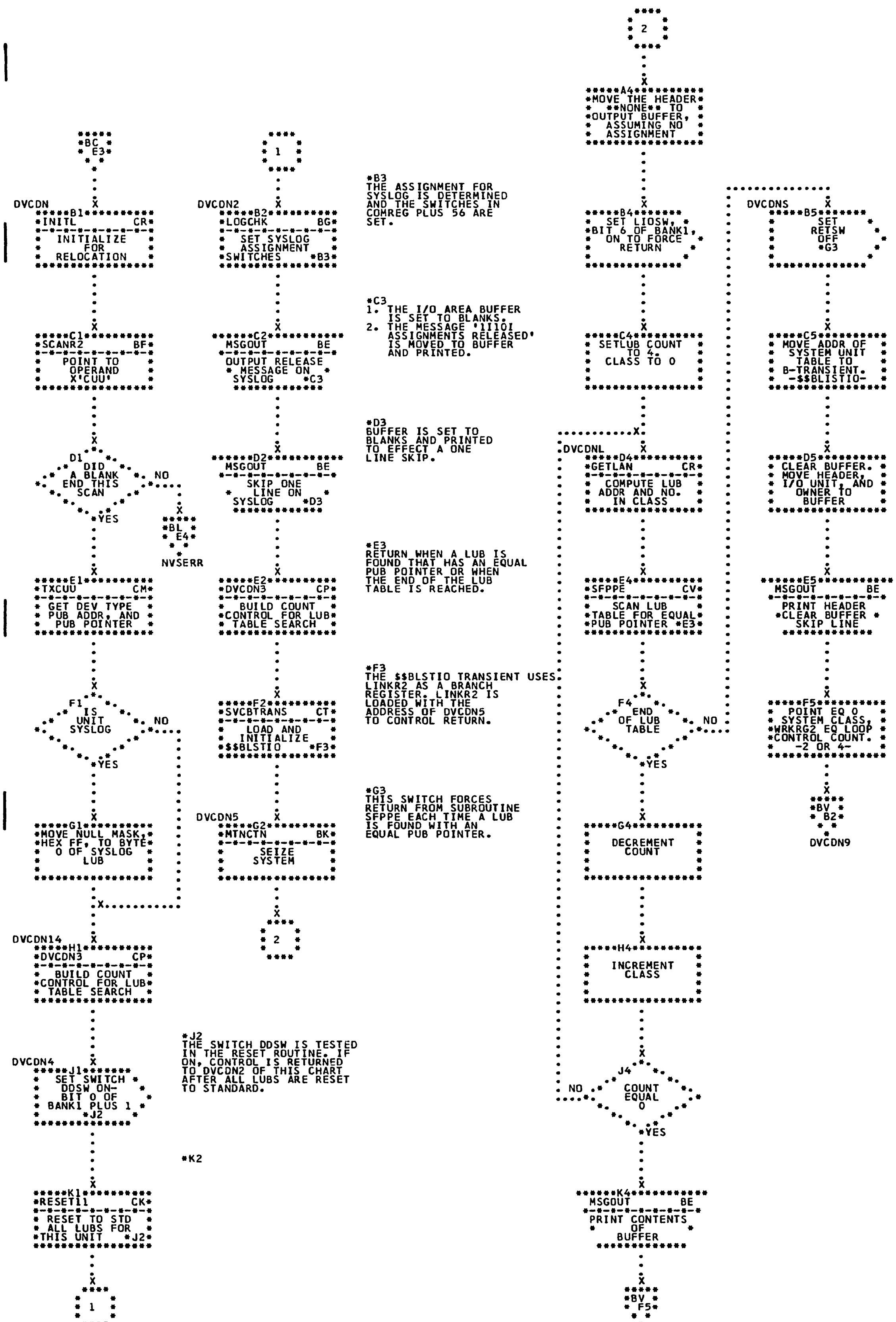


Chart BV. DVCDN Statement Processor- $JOBCTLD (Part 2 of 3); Refer to Job Control, Chart 05

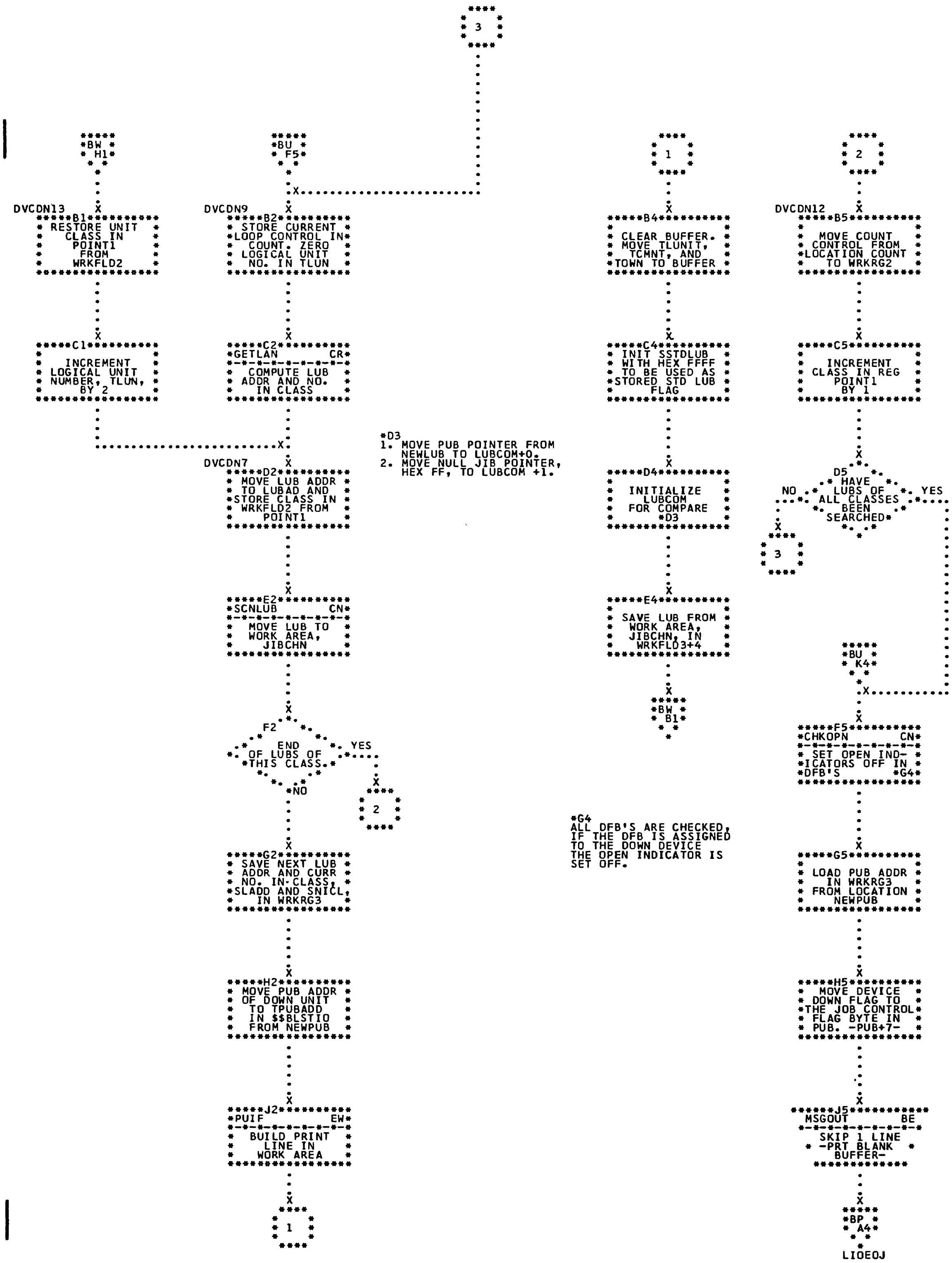


Chart BW. DVCDN Statement Processor- $JOBCTLD (Part 3 of 3); Refer to Job Control, Chart 05

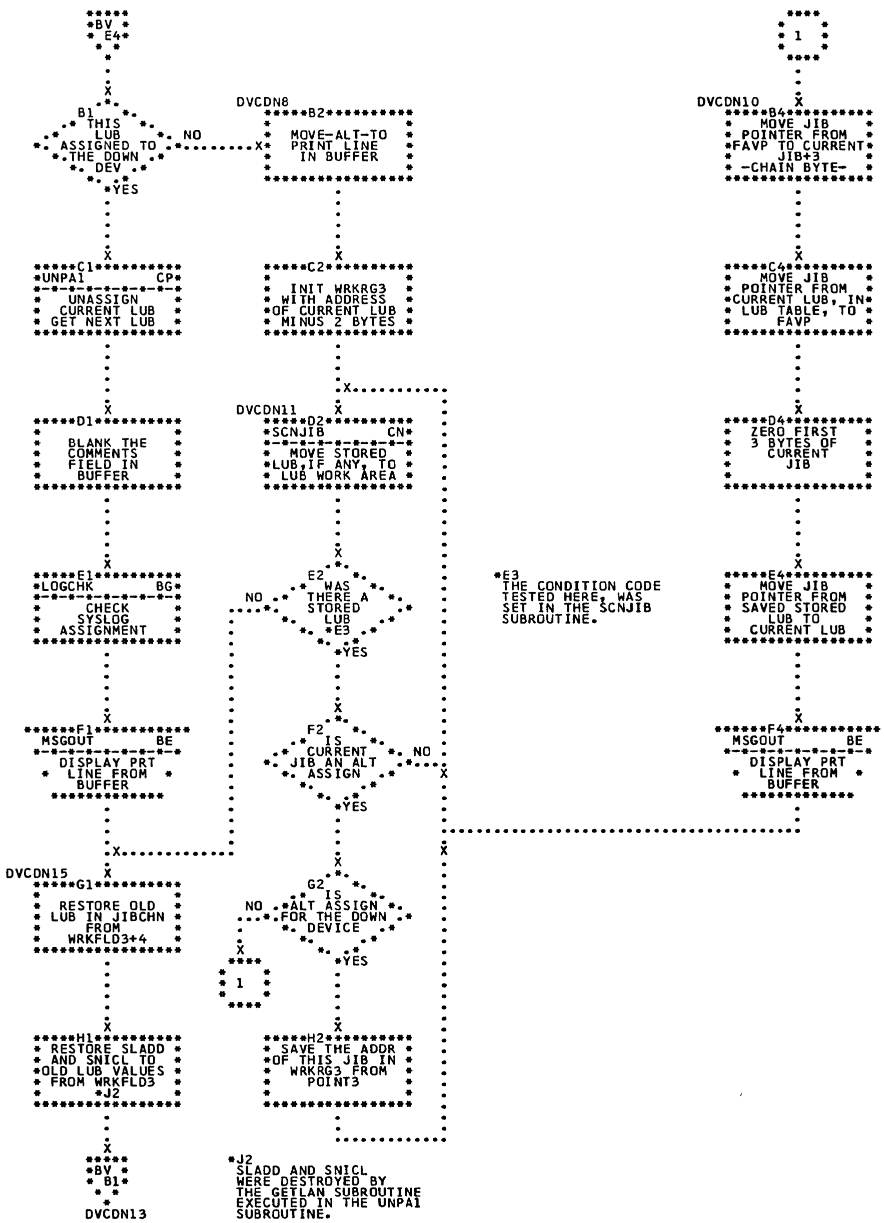


Chart BX. DVCUP Statement Processor-- $JOBCTLD; Refer to Job Control Chart 05

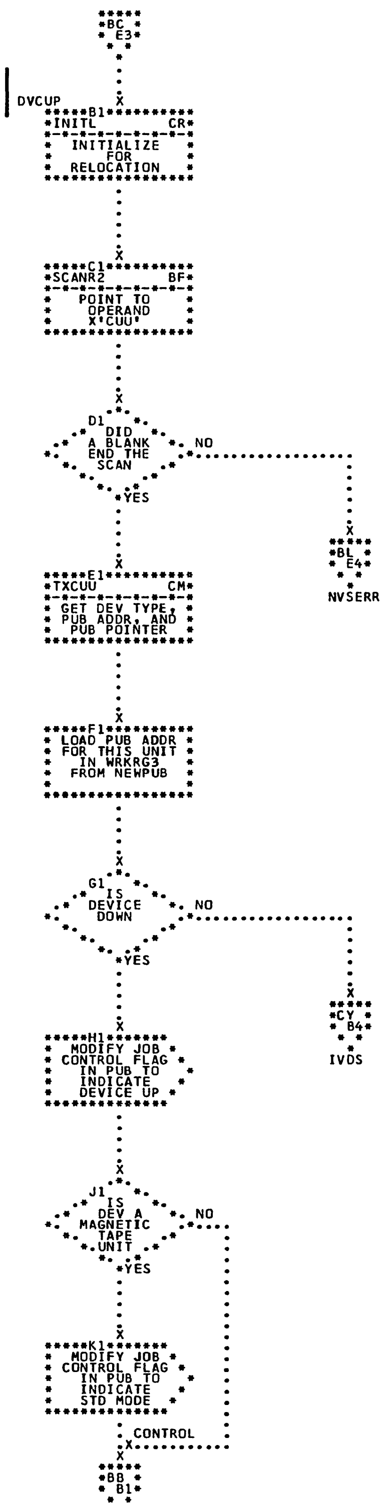


Chart BY. ASSGN Statement Processor- $JOBCTLD (Scan and Check First and Second Operand; Part 1 of 10); Refer to Job Control, Chart 04

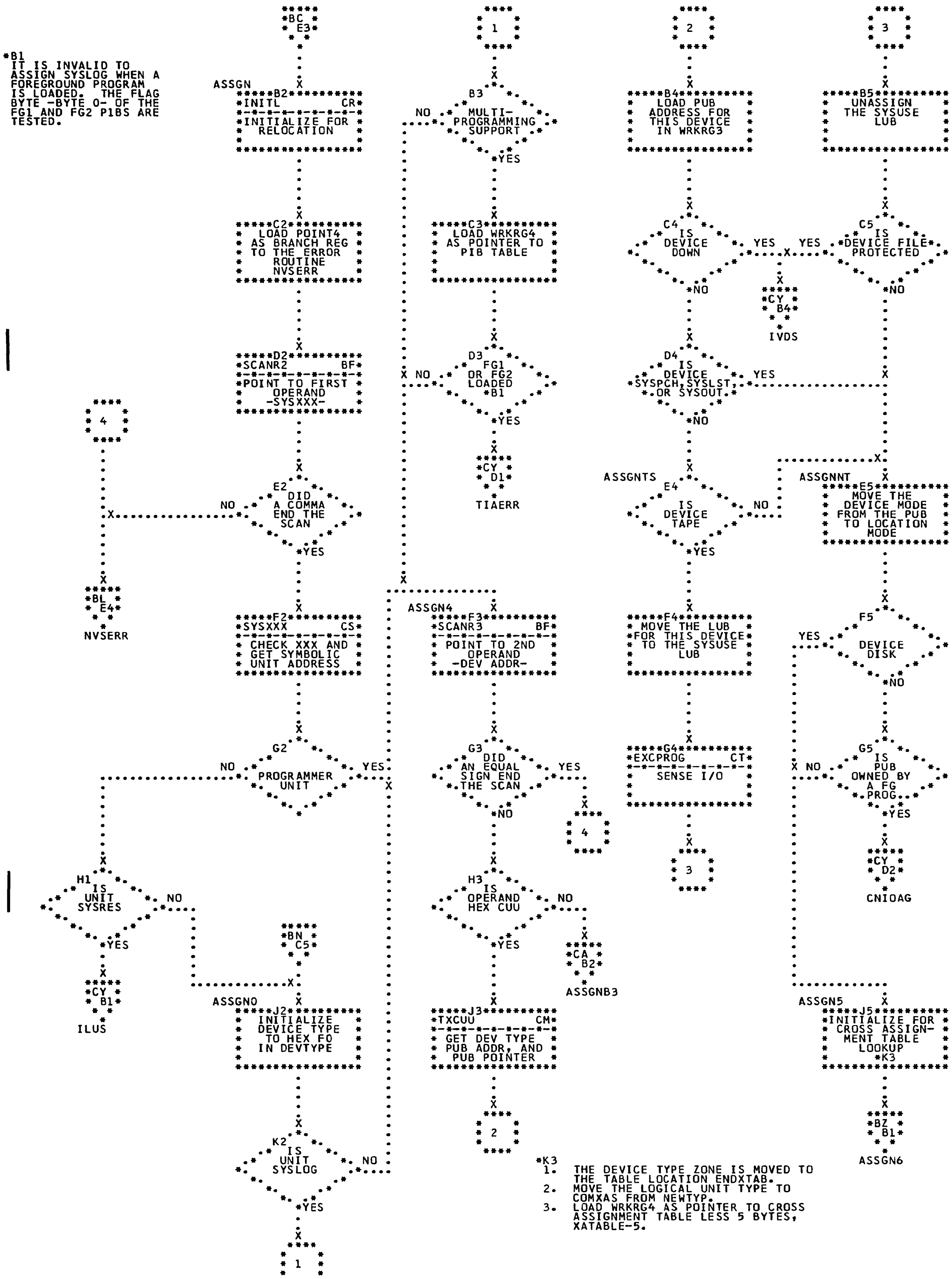
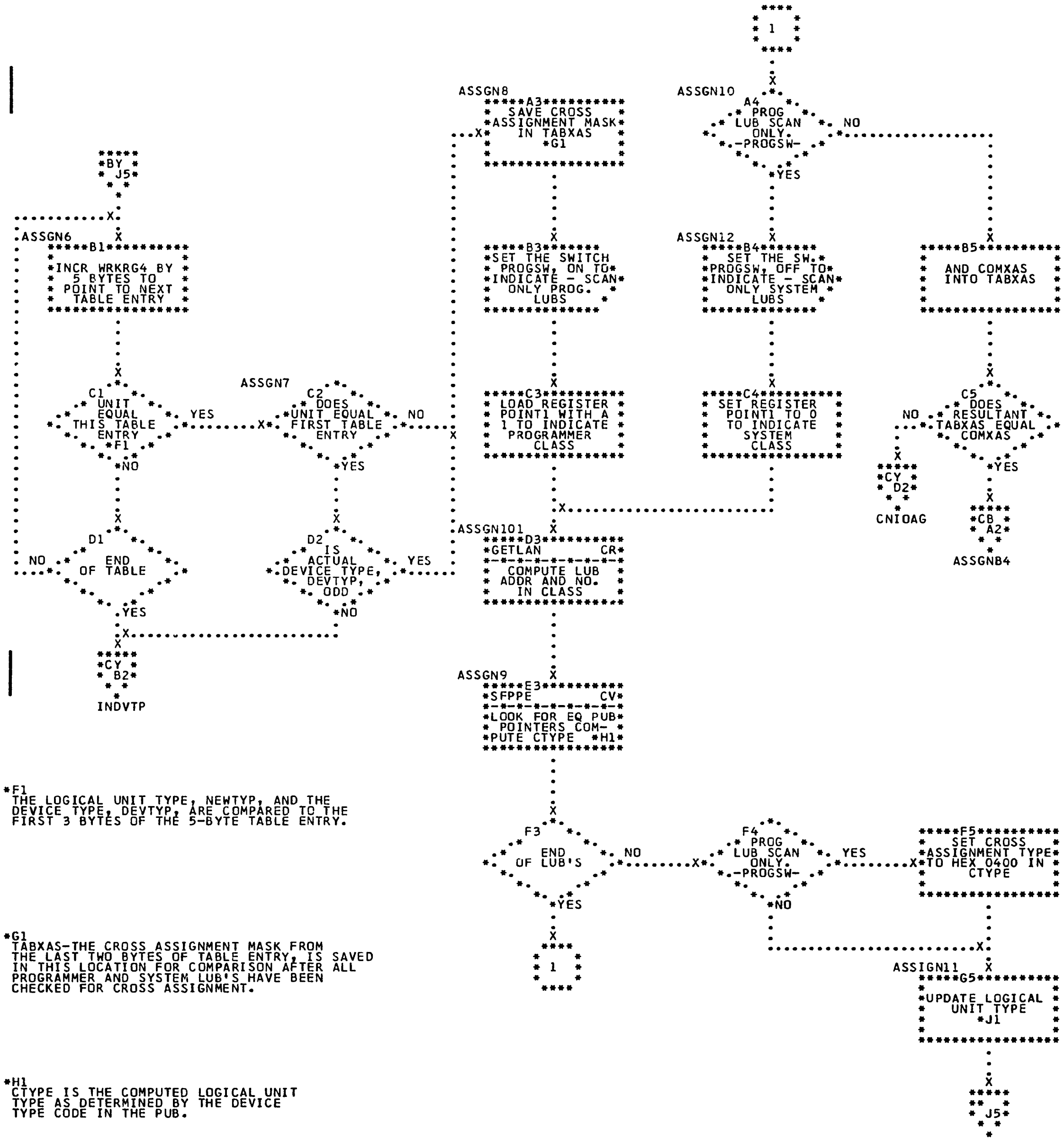


Chart BZ. ASSGN Statement Processor- $JOBCTLD (Cross Assignment Verification); (Part 2 of 10); Refer to Job Control, Chart 04




*F1
THE LOGICAL UNIT TYPE, NEWTYP, AND THE DEVICE TYPE, DEVTYP, ARE COMPARED TO THE FIRST 3 BYTES OF THE 5-BYTE TABLE ENTRY.

*G1
TABXAS-THE CROSS ASSIGNMENT MASK FROM THE LAST TWO BYTES OF TABLE ENTRY, IS SAVED IN THIS LOCATION FOR COMPARISON AFTER ALL PROGRAMMER AND SYSTEM LUB'S HAVE BEEN CHECKED FOR CROSS ASSIGNMENT.

*H1
CTYPE IS THE COMPUTED LOGICAL UNIT TYPE AS DETERMINED BY THE DEVICE TYPE CODE IN THE PUB.

*J1
EACH TIME AN EQUAL PUB POINTER IS FOUND, THE VALUE IN CTYPE IS ORED INTO COMXAS TO BUILD A LOGICAL ACCUMULATION OF BITS TO BE COMPARED TO THE VALUE IN TABXAS. COMXAS CONTAINED THE LOGICAL UNIT TYPE AS IT WAS SET IN NEWTYP BY THE SUBROUTINE SYSXXX. CTYP CONTAINS THE LOGICAL UNIT TYPE HEX 04 IF PROGRAMMER LUBS ARE BEING SCANNED, OR THE COMPUTED VALUE SET IN THE SUBROUTINE SFPPE IF SYSTEM LUBS ARE BEING SCANNED.

*J5
RETURN TO THE SFPPE SUBROUTINE BY BRANCHING TO THE ADDRESS IN LINKR4.

Chart CA. ASSGN Statement Processor- $JOBCTLD (Verify and Store UA or IGN Assignment; (Part 3 of 10); Refer to Job Control, Chart 04

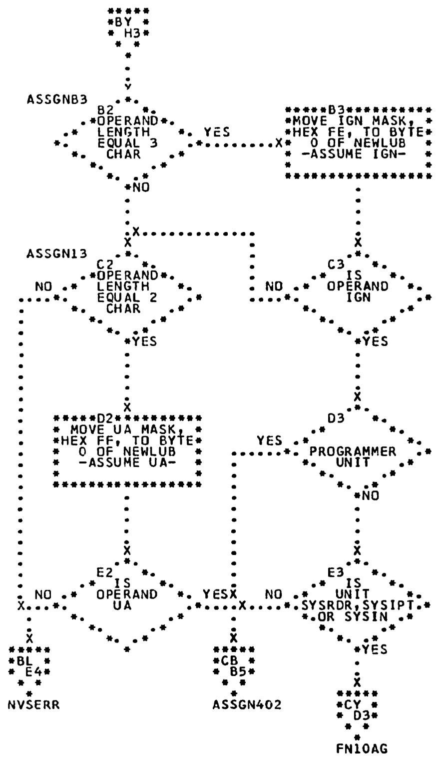


Chart CB. ASSGN Statement Processor- $JOBCTLD (Complete Scan of Operands; Part 4 of 10); Refer to Job Control, Chart 04

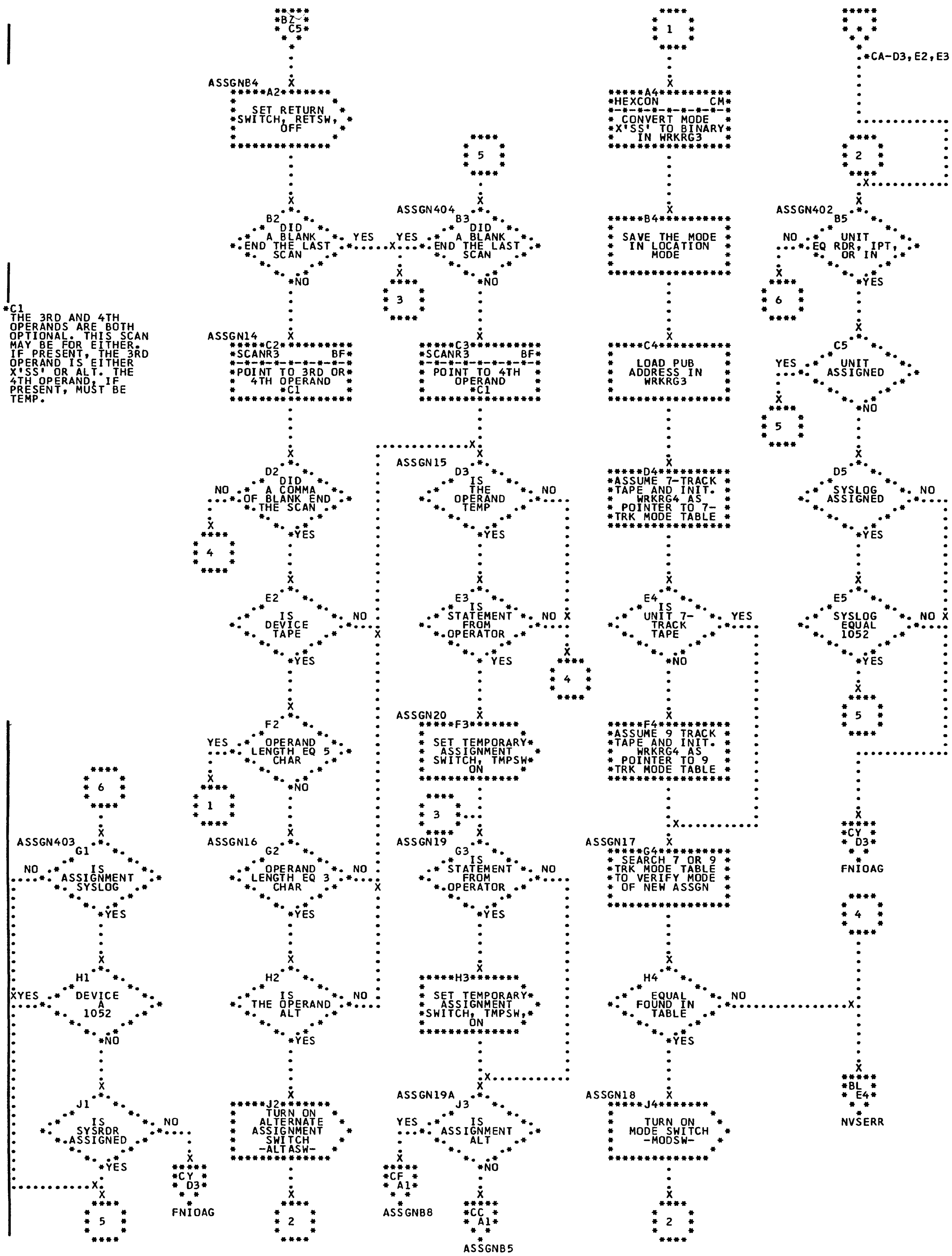


Chart CC. ASSGN Statement Processor- $JOBCTLD (Final Verification for Normal Assignment; Part 5 of 10); Refer to Job Control, Chart 04

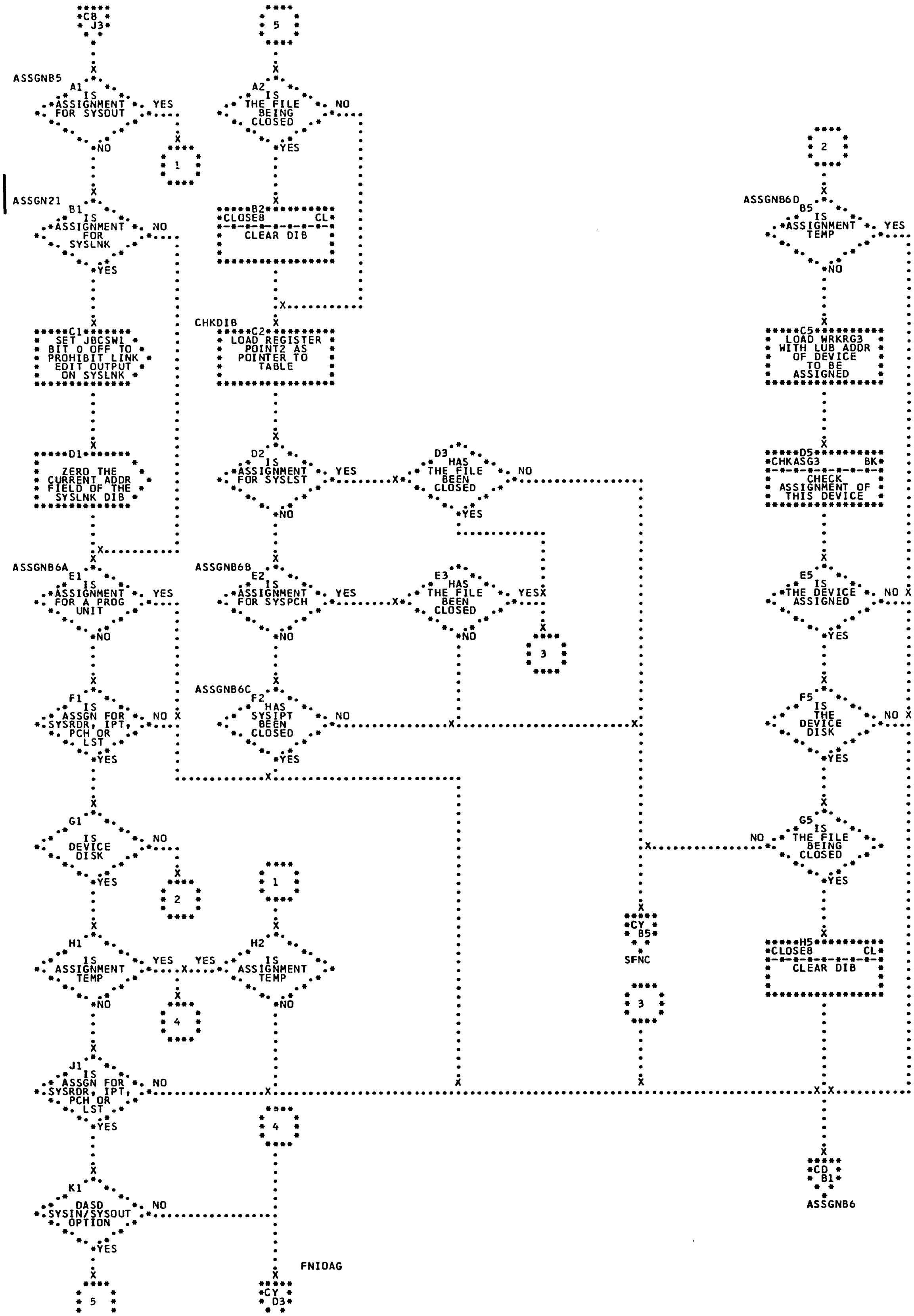


Chart CD. ASSGN Statement Processor- $JOBCTLD (Make Normal Standard Assignment; Part 6 of 10); Refer to Job Control, Chart 04

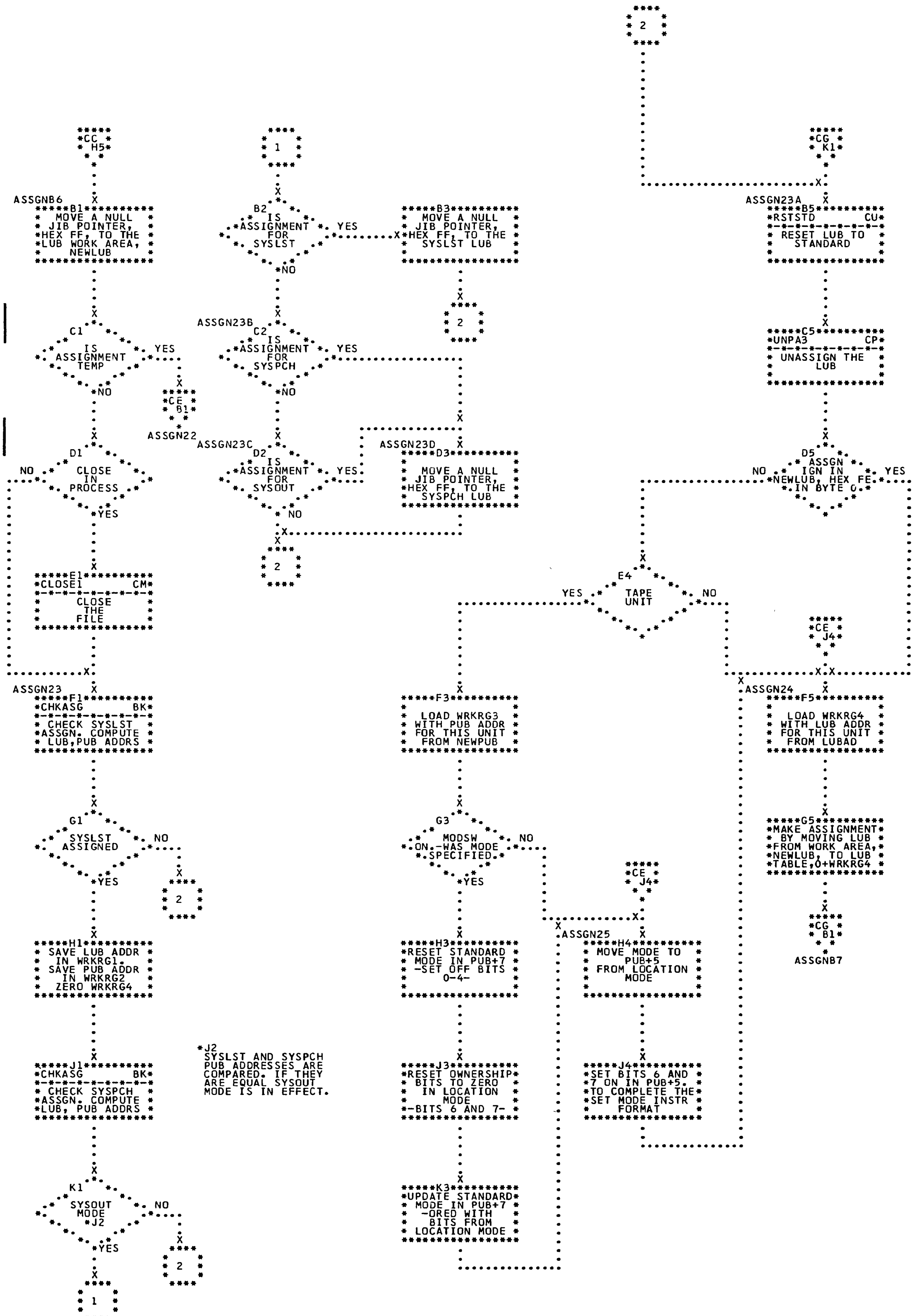


Chart CE. ASSGN Statement Processor- $JOBCTLD (Make Normal Temporary Assignment; Part 7 of 10); Refer to Job Control, Chart 04

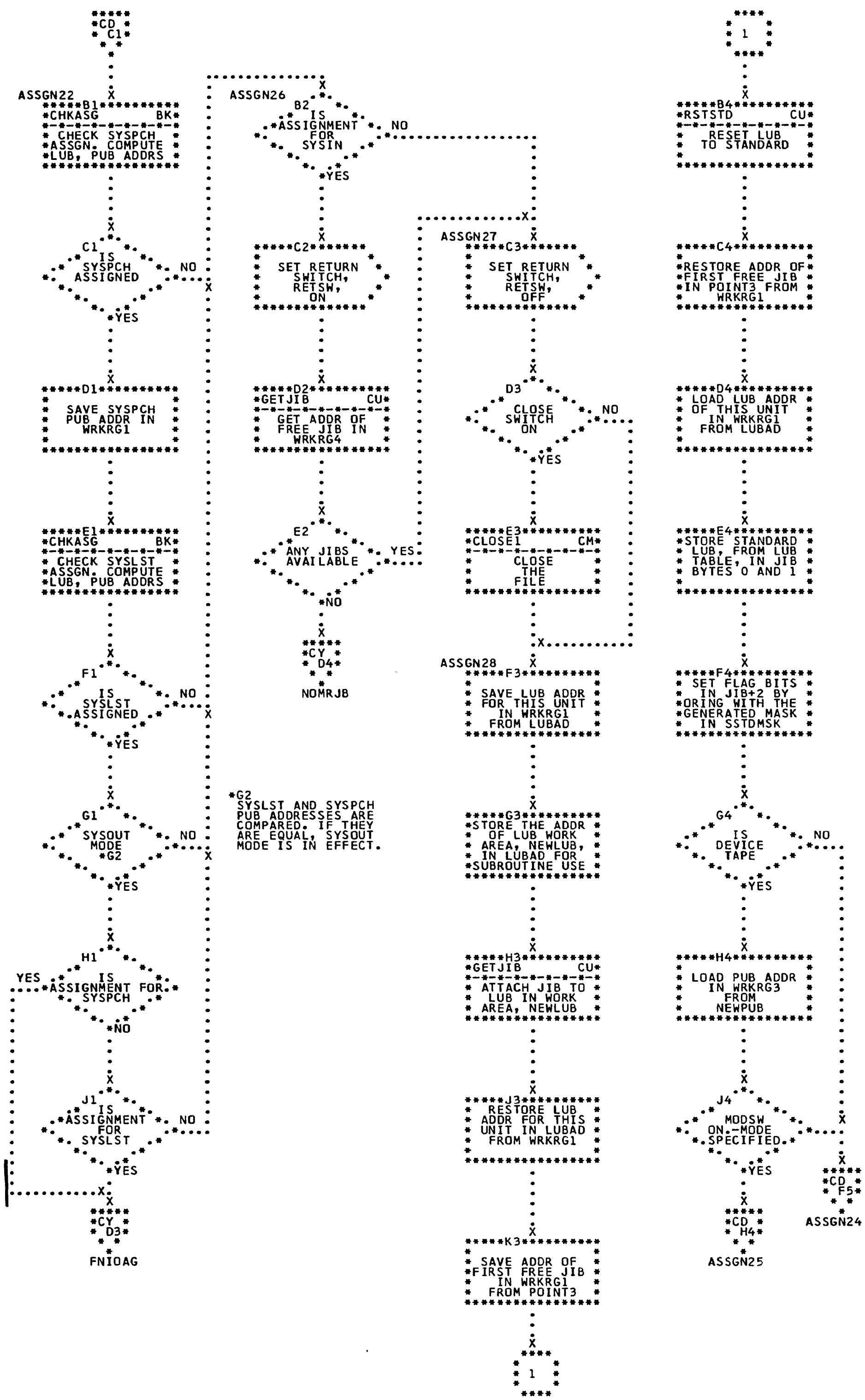


Chart CF. ASSGN Statement Processor- $JOBCTLD (Make Alternate Assignment; Part 8 of 10); Refer to Job Control, Chart 04

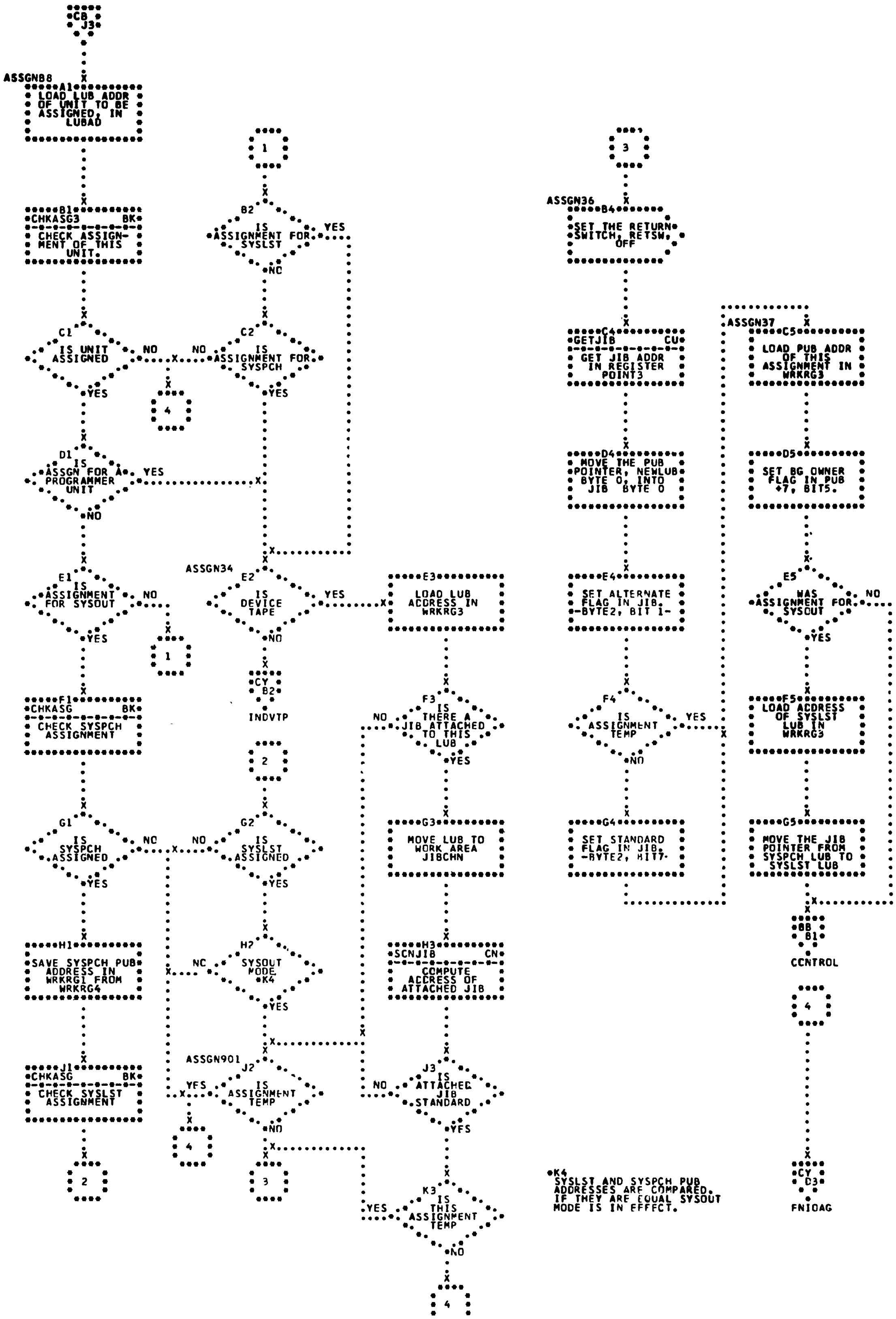


Chart CG. ASSGN Statement Processor- $JOBCTLD (Terminate Assignment and Open Files--Part 1 of 2); Refer to Job Control, Chart 04 (Part 9 of 10)

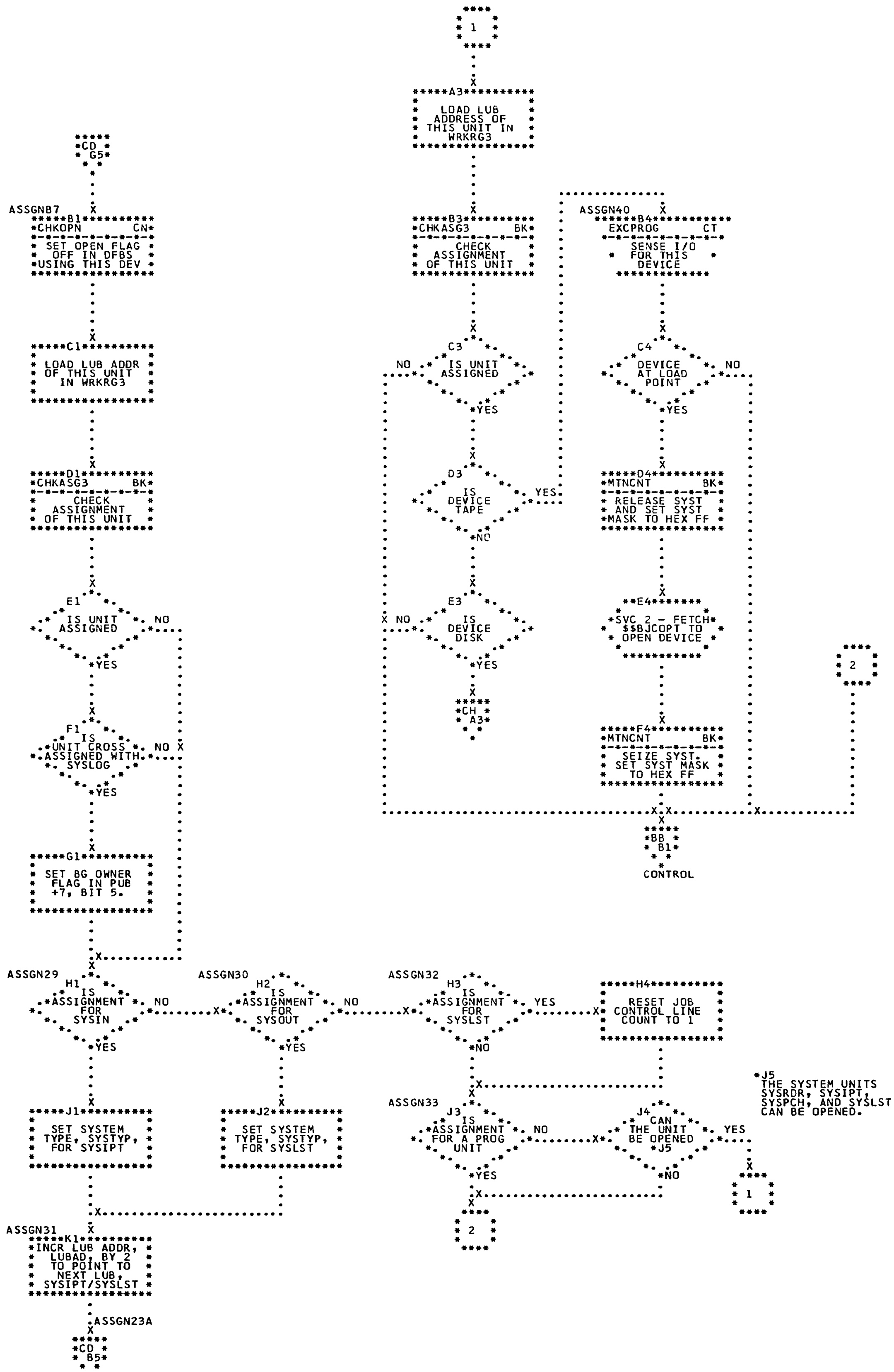


Chart CH. ASSGN Statement Processor- $JOBCTLD (Terminate Assignment and Open Files--Part 2 of 2); Refer to Job Control, Chart 04 (Part 10 of 10)

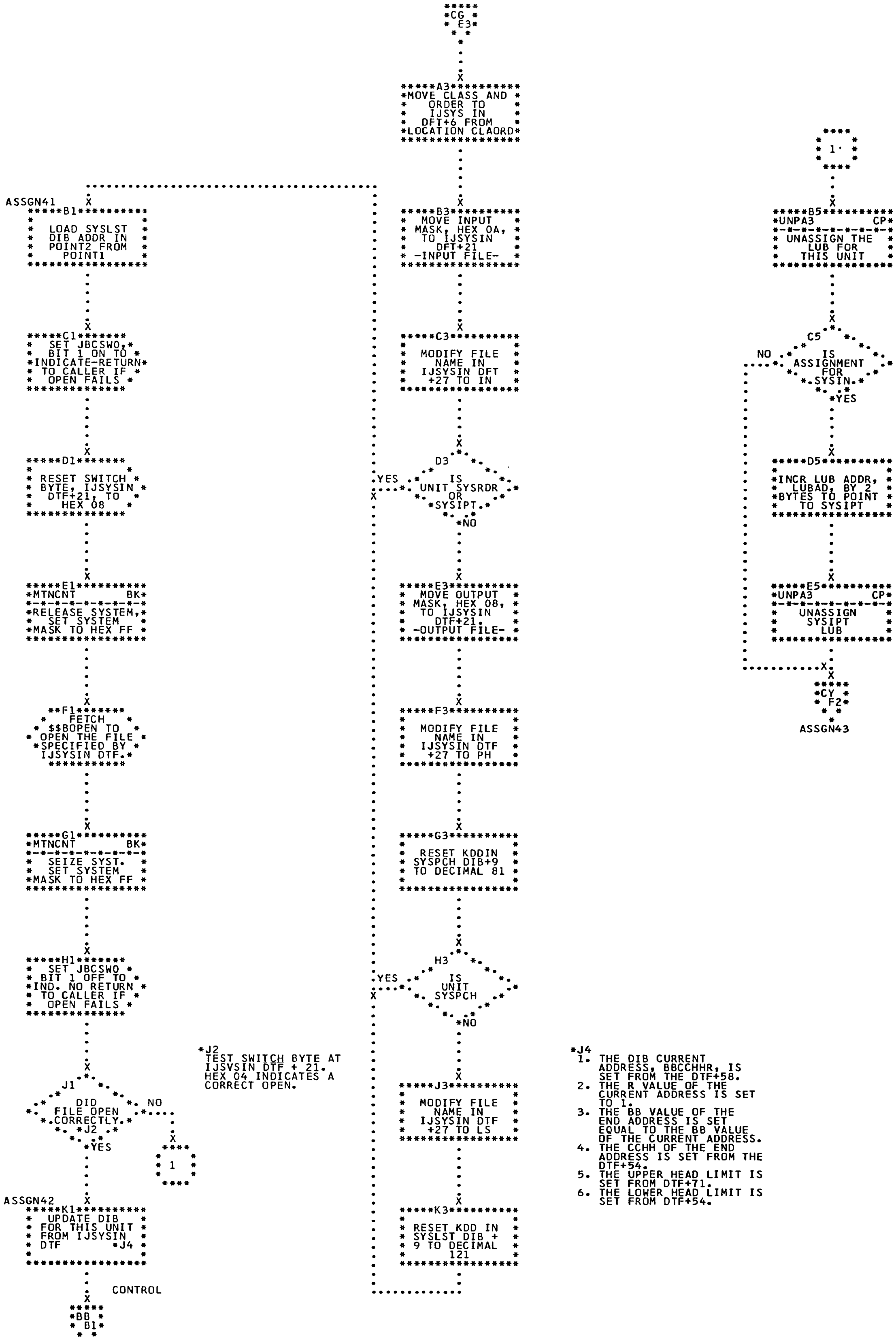


Chart CJ. RESET Statement Processor- $JOBCTLD (Part 1 of 2); Refer to Job Control, Chart 05

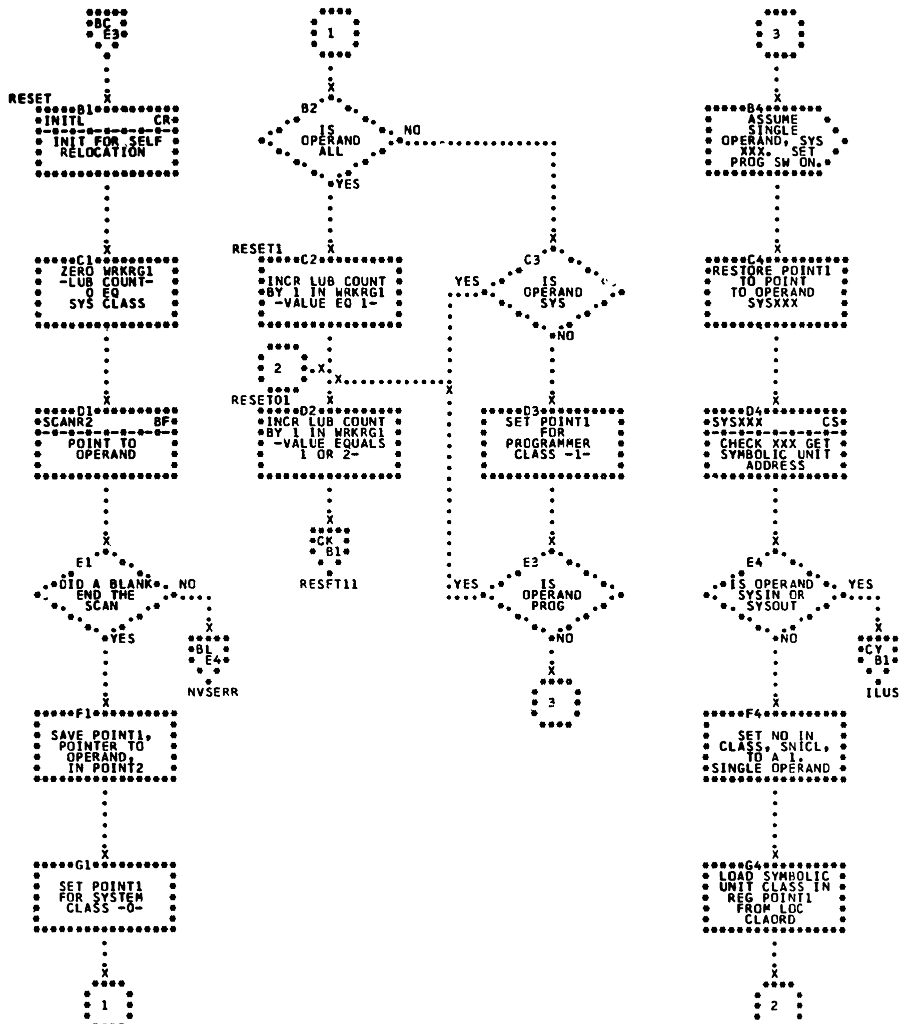


Chart CK. RESET Statement Processor- $JOBCTLD (Part 2 of 2); Refer to Job Control, Chart 05

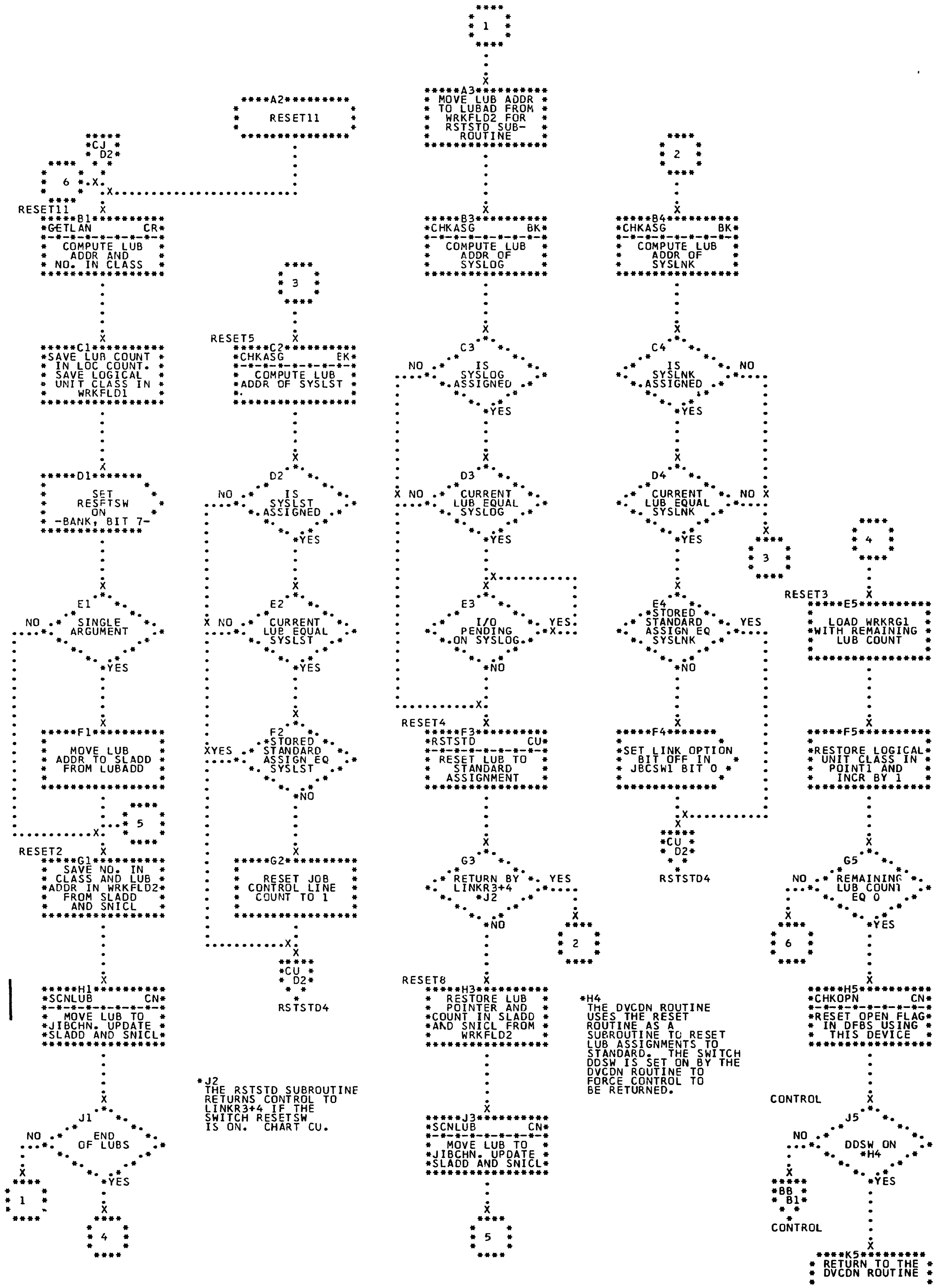


Chart CL. Subroutine-- $JOBCTLD (CLOSE8); Refer to Job Control, Charts 04, 05

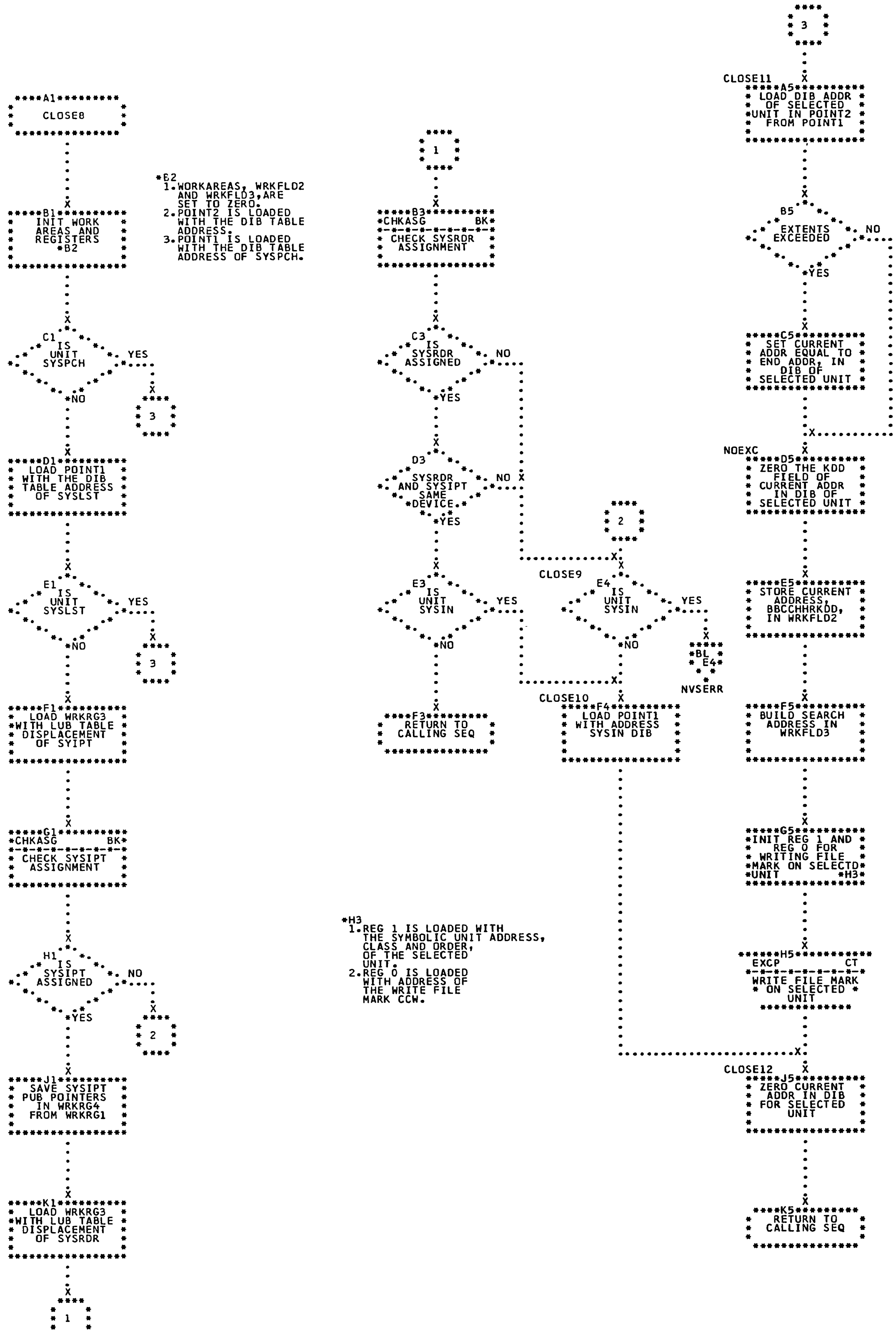


Chart CM. Subroutines-- $JOBCTLD (TXCUU, TXCUU3, HEXCON and CLOSE1); Refer to Job Control, Charts 04, 05

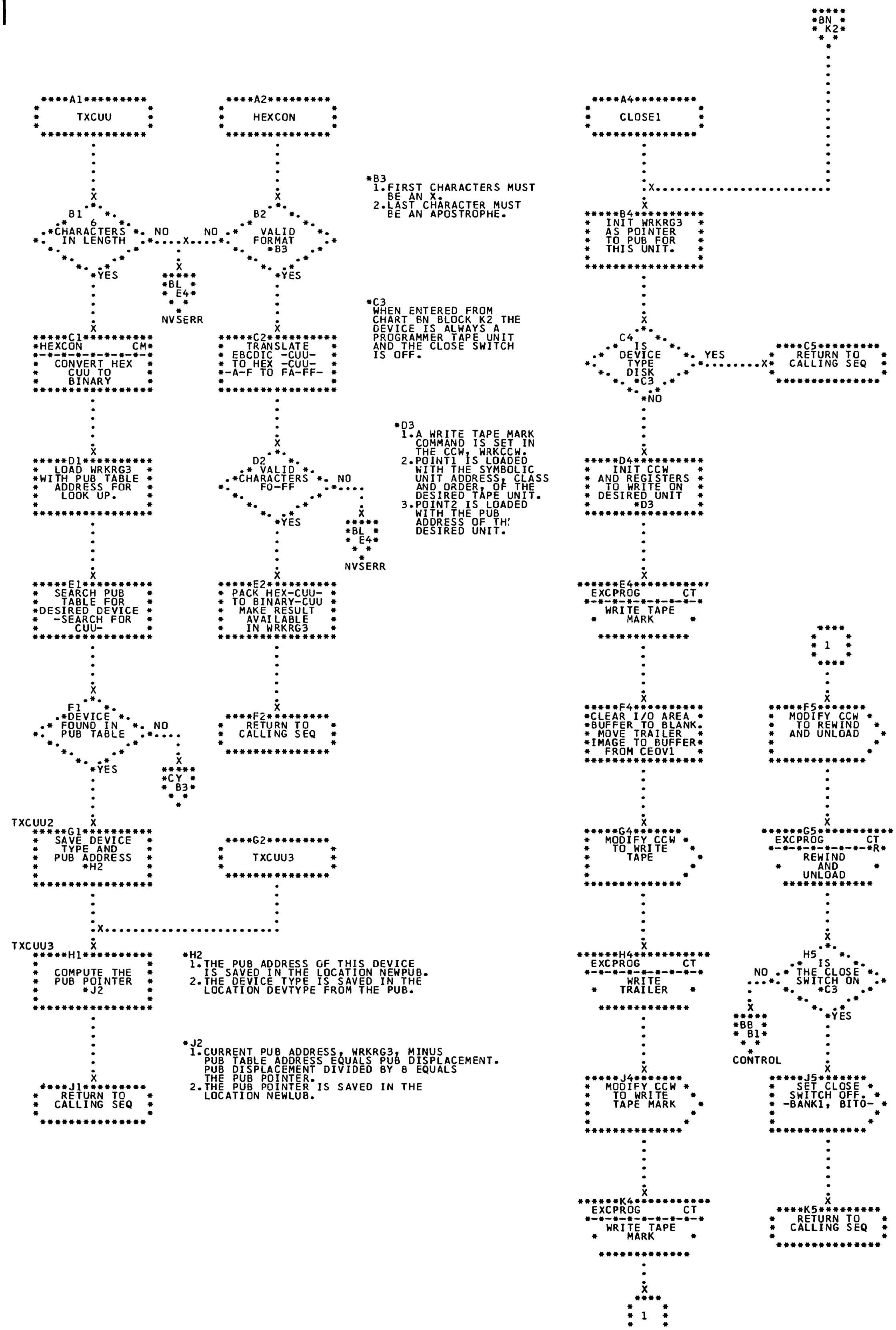
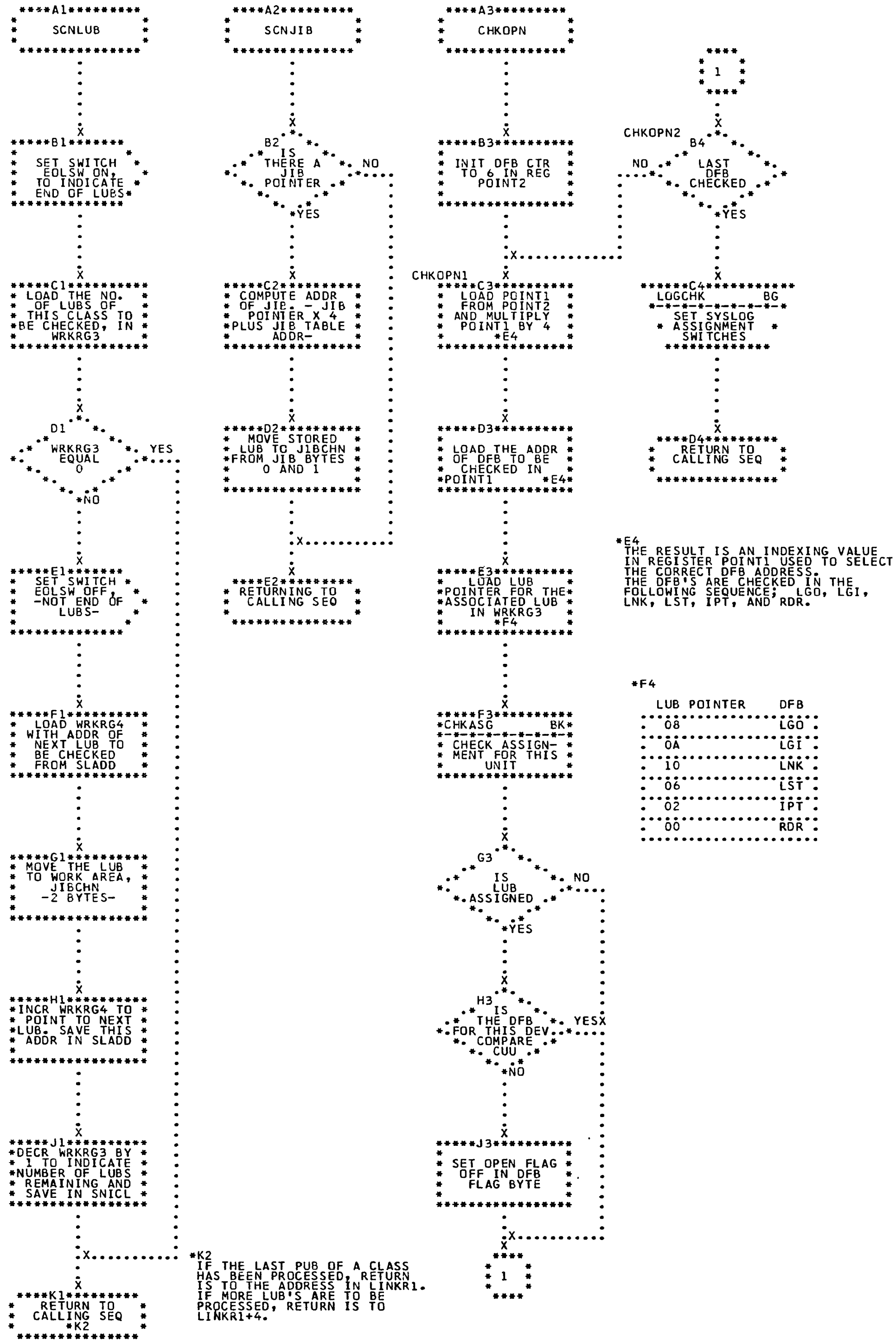


Chart CN. Subroutines-- $JOBCTLD (SCNLUB, SCNJIB, and CHKOPN); Refer to Job Control, Charts 04, 05



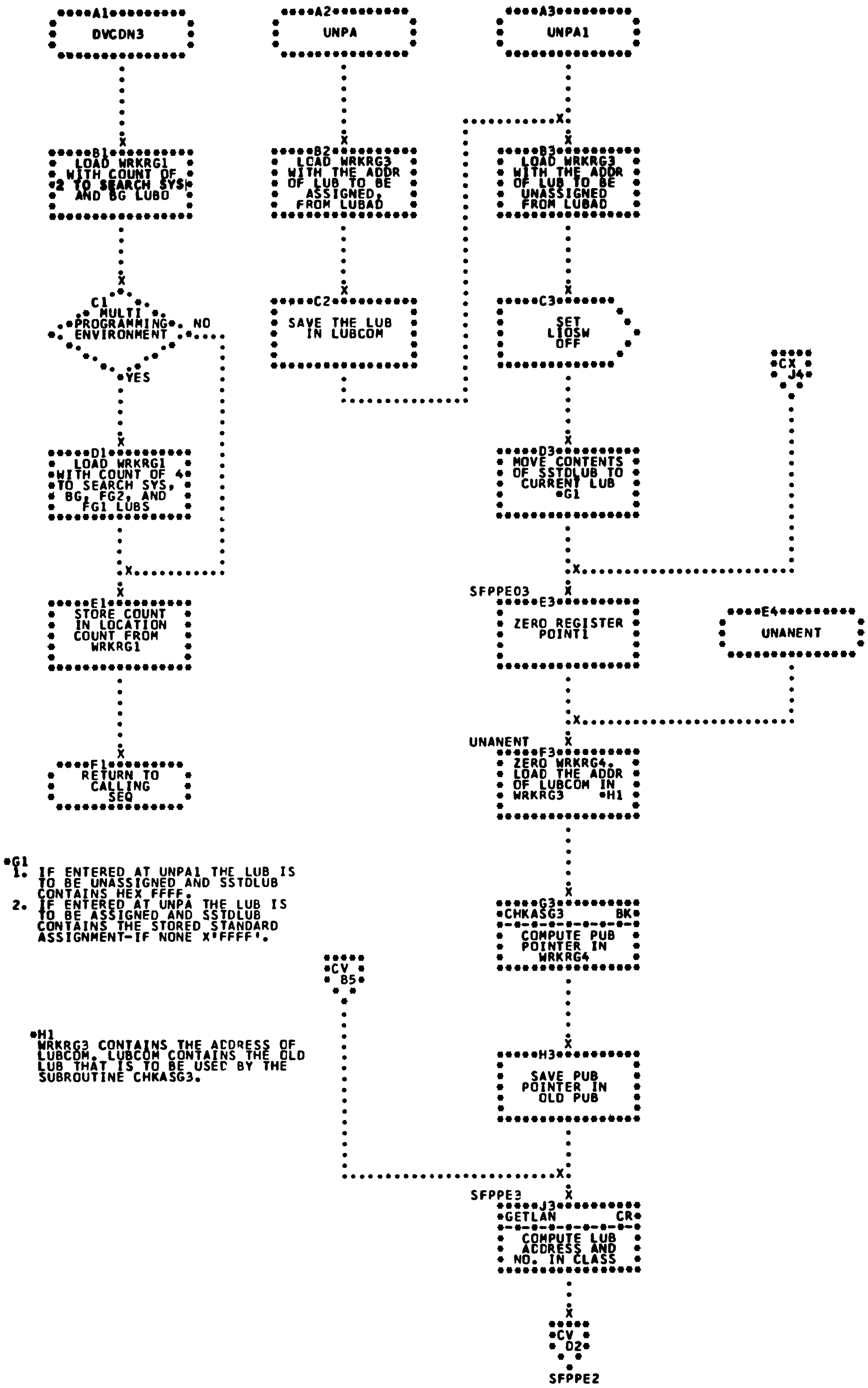
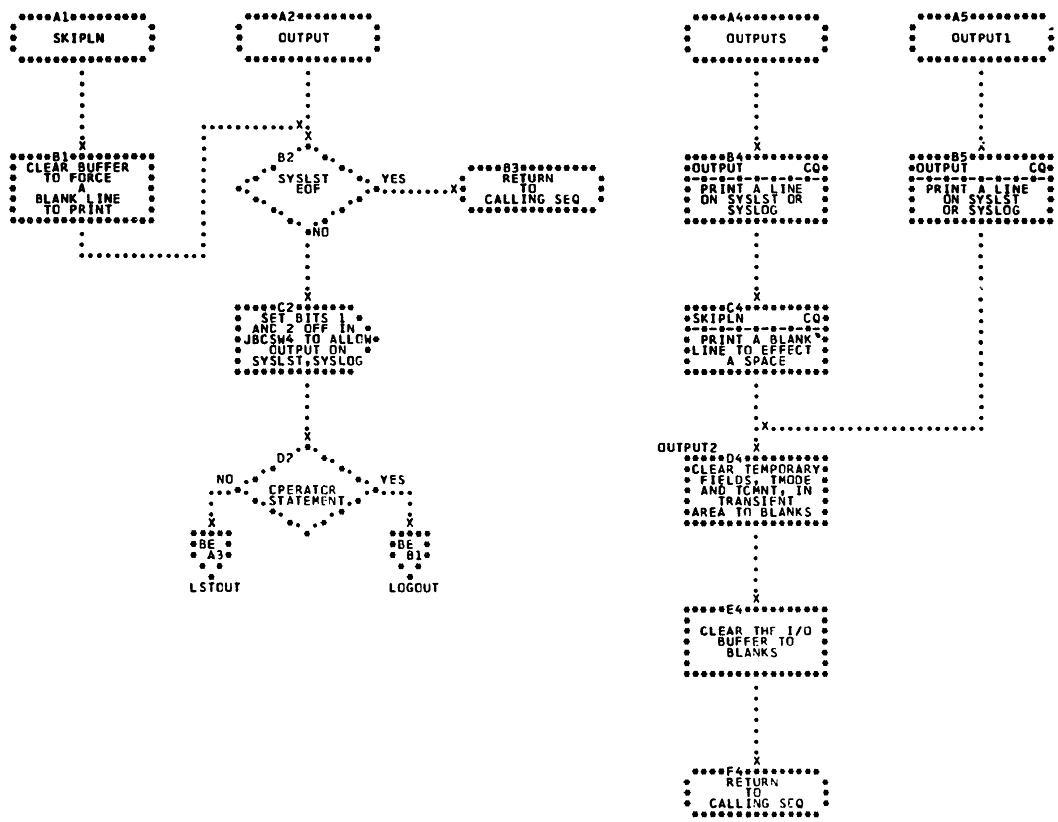
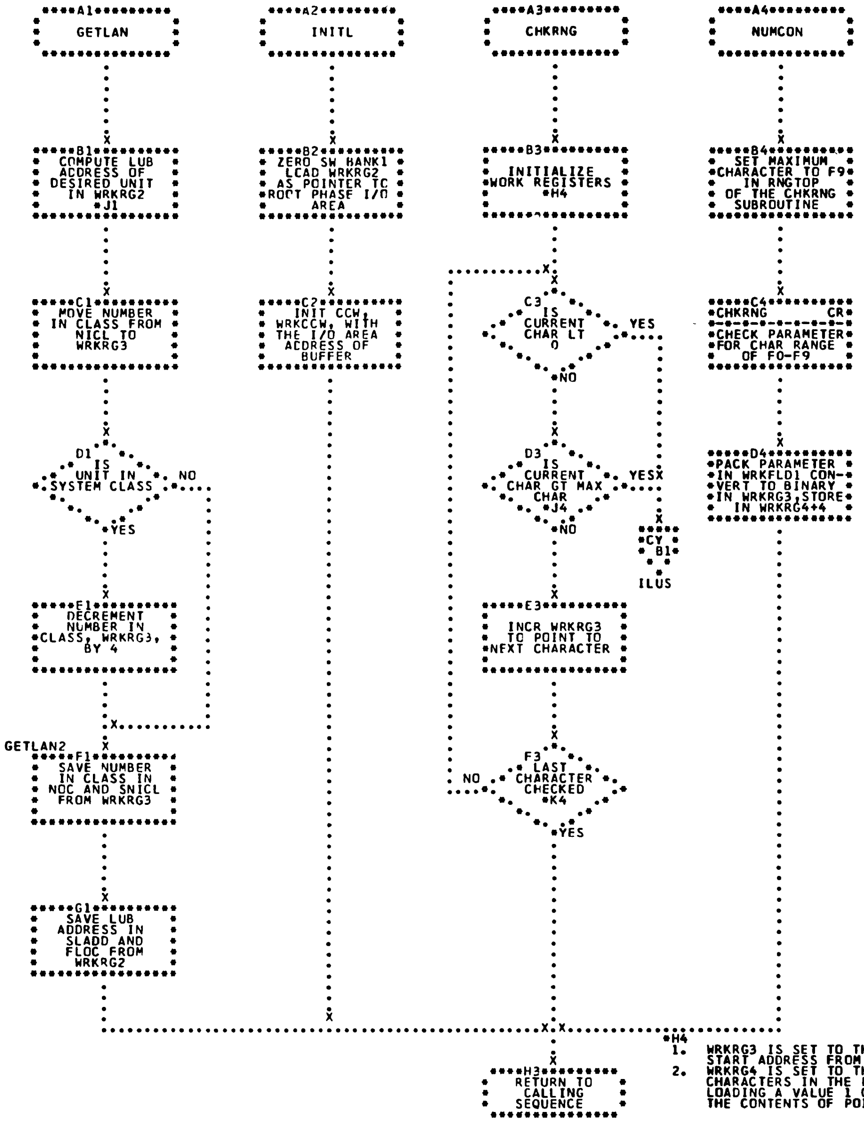

A1
DVCDN3
A2
UNPA
A3
UNPA1
B1
LOAD WRKRG1 WITH COUNT OF 2 TO SEARCH SYS AND BG LUBO
B2
LOAD WRKRG3 WITH THE ADDR OF LUB TO BE ASSIGNED, FROM LUBAD
B3
LOAD WRKRG3 WITH THE ADDR OF LUB TO BE UNASSIGNED FROM LUBAD
C1
MULTI PROGRAMMING ENVIRONMENT
NO
YES
C2
SAVE THE LUB IN LUBCOM
C3
SET LIOSW OFF
CX
J4
D1
LOAD WRKRG1 WITH COUNT OF 4 TO SEARCH SYS, BG, FG2, AND FG1 LUBS
D3
MOVE CONTENTS OF SSTDLUB TO CURRENT LUB *G1
SFPPE03
E1
STORE COUNT IN LOCATION COUNT FROM WRKRG1
E3
ZERO REGISTER POINT1
E4
UNANENT
UNANENT
F1
RETURN TO CALLING SEQ
F3
ZERO WRKRG4. LOAD THE ADDR OF LUBCOM IN WRKRG3 *H1
*G1
1. IF ENTERED AT UNPA1 THE LUB IS TO BE UNASSIGNED AND SSTDLUB CONTAINS HEX FFFF.
2. IF ENTERED AT UNPA THE LUB IS TO BE ASSIGNED AND SSTDLUB CONTAINS THE STORED STANDARD ASSIGNMENT-IF NONE X'FFFF'.
G3
CHKASG3 BK
COMPUTE PUB POINTER IN WRKRG4
CV
B5
*H1
WRKRG3 CONTAINS THE ADDRESS OF LUBCOM. LUBCOM CONTAINS THE OLD LUB THAT IS TO BE USED BY THE SUBROUTINE CHKASG3.
H3
SAVE PUB POINTER IN OLD PUB
SFPPE3
J3
GETLAN CR
COMPUTE LUB ADDRESS AND NO. IN CLASS
CV
D2
SFPPE2

Chart CO. Subroutines-- $JOBCTLD (SKIPLN, OUTPUT, OUTPUTS, and OUTPUT1); Refer to Job Control, Charts 04, 05



## Chart CR. Subroutines-- $JOBCTLD (GETLAN, INITL, CHKRNG, and NUMCON); Refer to Job Control, Charts 04, 05




*H4
1. WRKRG3 IS SET TO THE PARAMETER START ADDRESS FROM POINT1.
2. WRKRG4 IS SET TO THE NUMBER OF CHARACTERS IN THE PARAMETER BY LOADING A VALUE 1 GREATER THAN THE CONTENTS OF POINT3.

*J1
1. UNIT CLASS IS IN POINT1. 0=SYSTEM, 1=PROGRAMMER.
2. FICL ADDRESS + UNIT CLASS EQUALS LUB POINTER TO 1ST UNIT OF A CLASS, SYSTEM OR PROGRAMMER.
3. LUB POINTER X 2 EQUALS LUB TABLE DISPLACEMENT.
4. LUB TABLE ADDRESS PLUS DISPLACEMENT EQUALS LUB ADDRESS OF THE FIRST UNIT OF THE DESIRED CLASS.

*J4
RNGTOP IS A COMPARE IMMEDIATE INSTRUCTION. THE CALLING SEQUENCE MOVES THE MAXIMUM CHARACTER INTO THIS INSTRUCTION BEFORE BRANCHING TO THIS ROUTINE.

*K4
THE CHARACTER COUNT IN WRKRG4 IS DECREMENTED BY 1 AND CHECKED FOR ZERO. THE ROUTINE LOOPS UNTIL WRKRG4 BECOMES ZERO.

Chart CS. Subroutine-- $JOBCTLD (SYSXXX); Refer to Job Control, Charts 04, 05

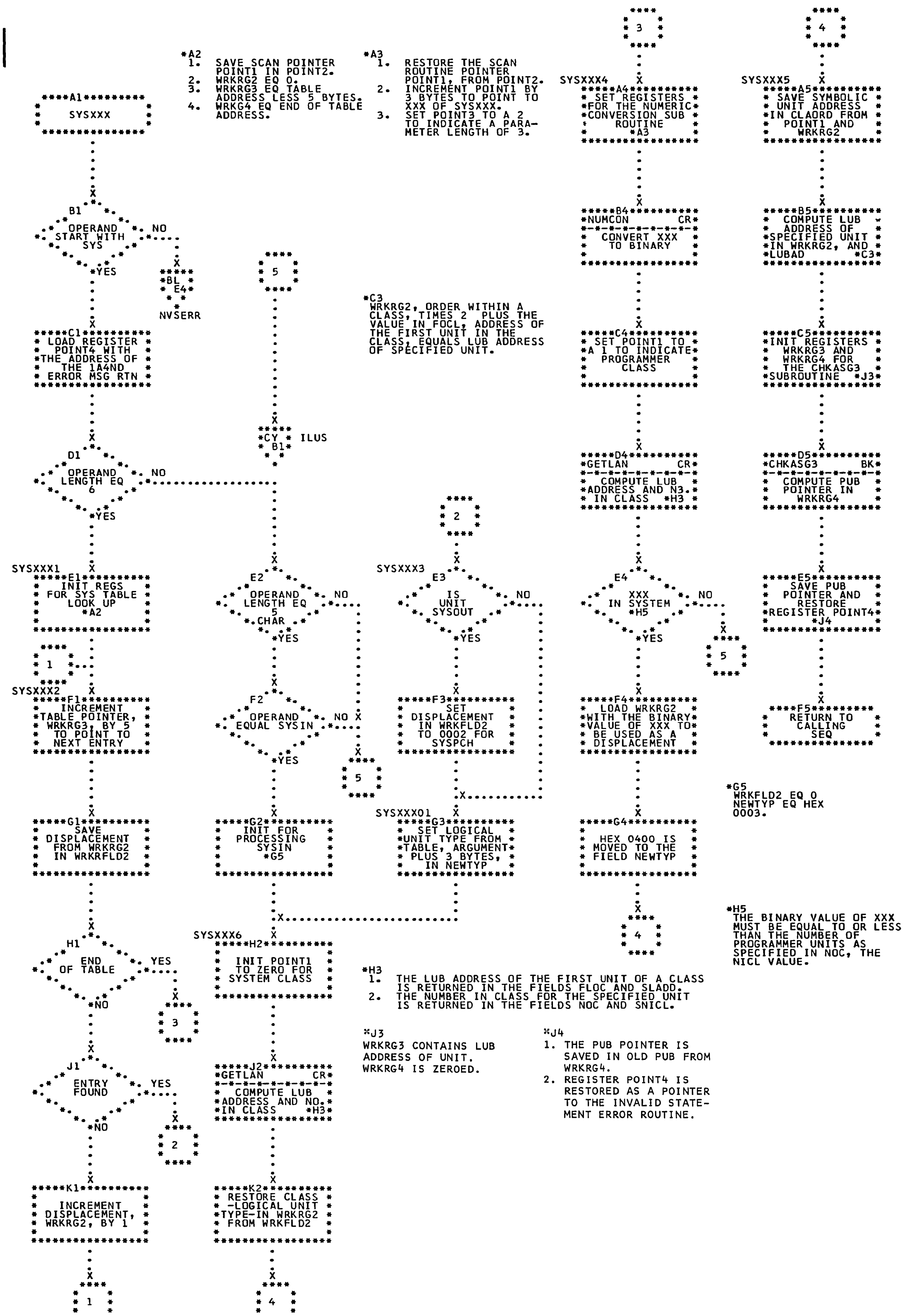


Chart CT. Subroutines-- $JOBCTLD (EXCP, EXCPROG, EXCPROG1, and SVCBTRANS); Refer to Job Control, Charts 04, 05

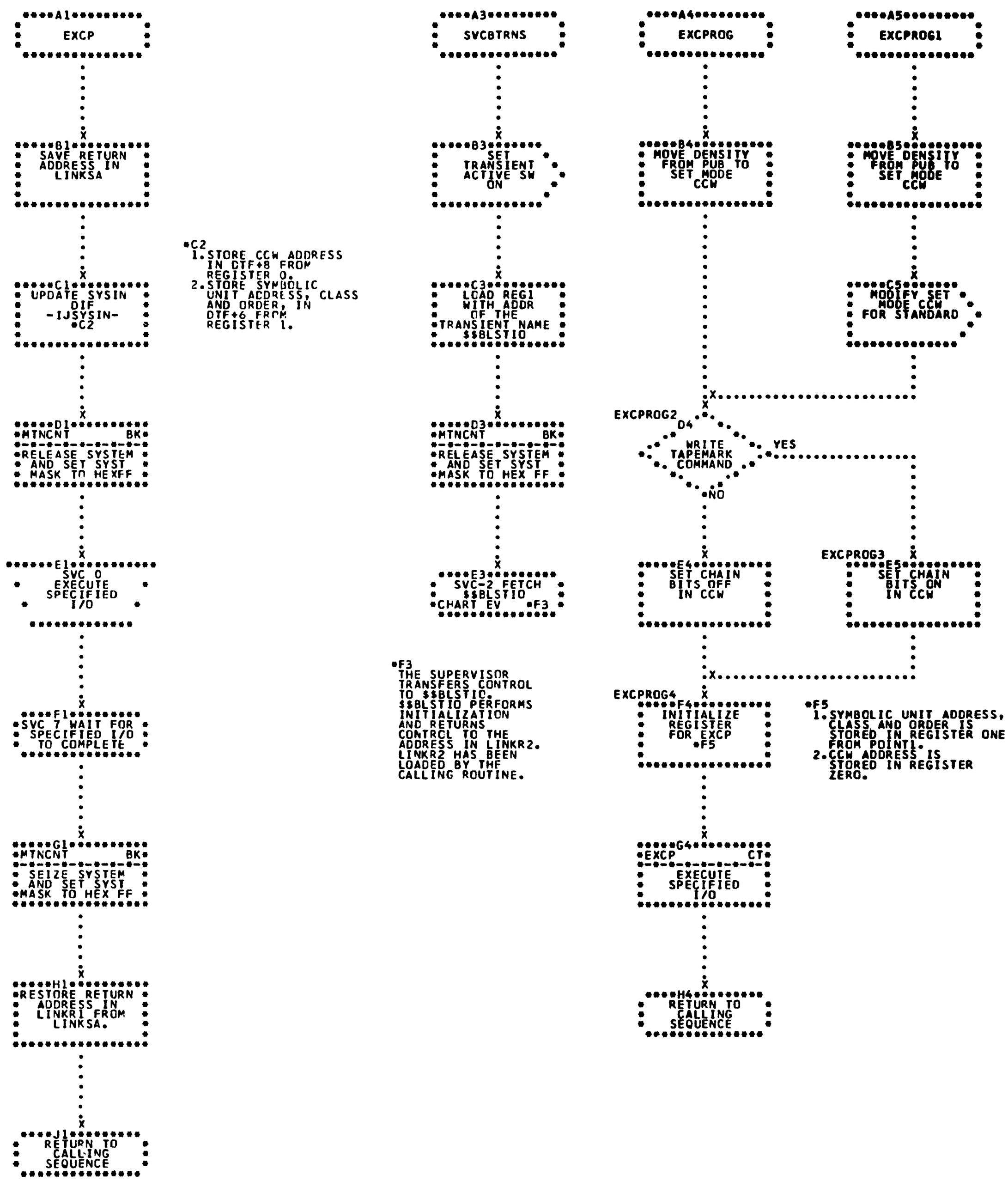


Chart CU. Subroutines-- $JOBCTLD (RSTSTD, and GETJIB); Refer to Job Control, Charts 04, 05

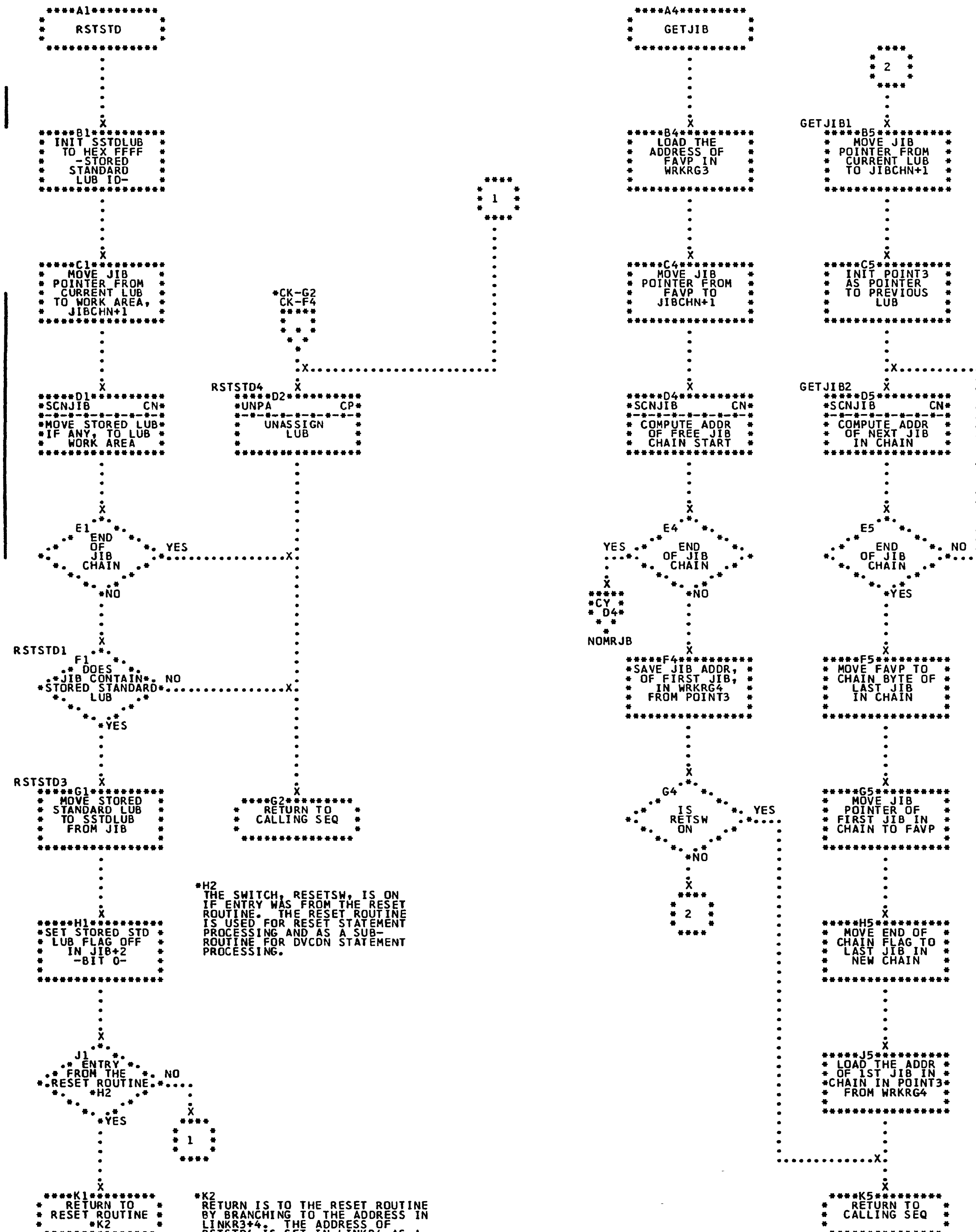
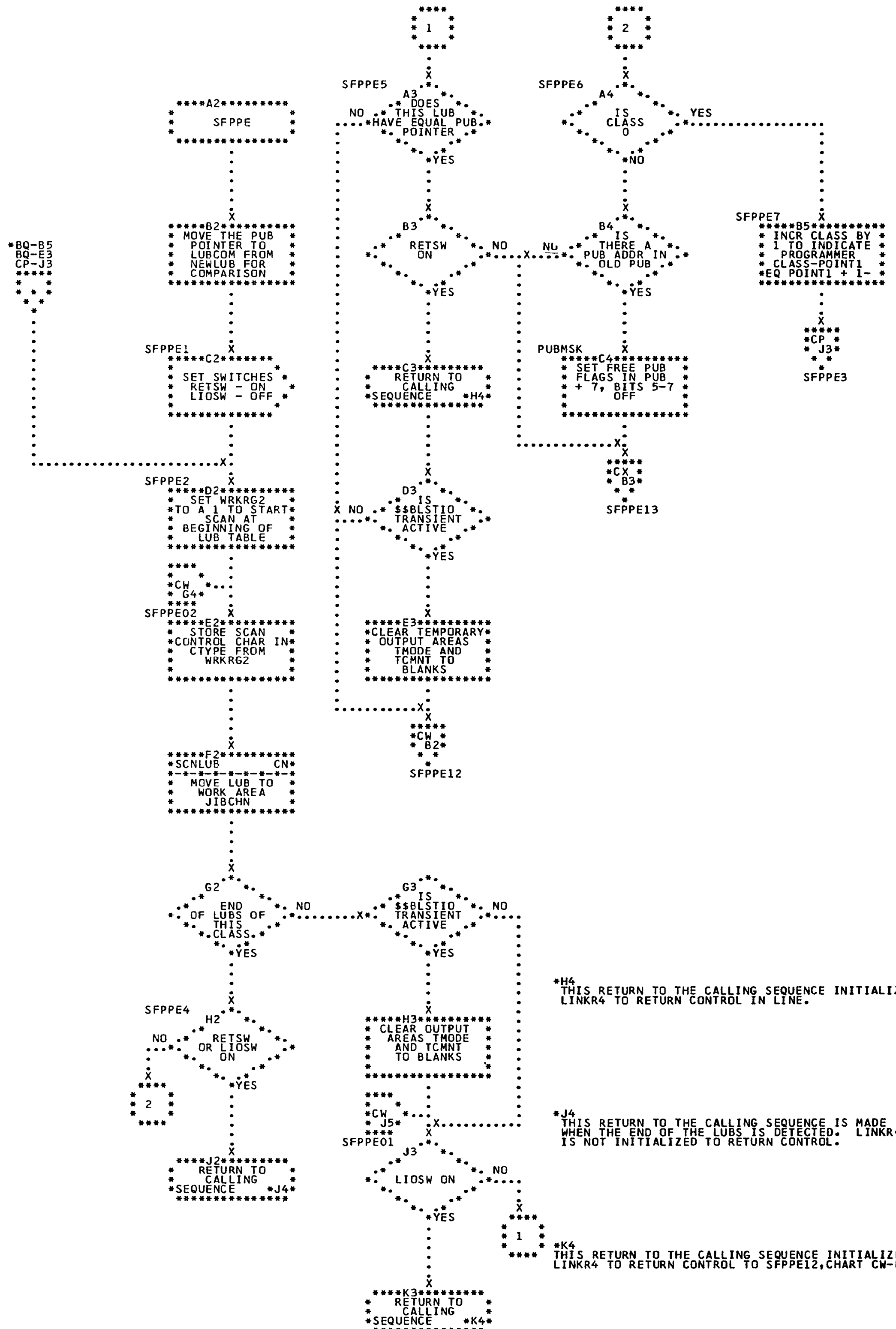


Chart CV. Subroutine-- $JOBCTLD (SFPPE; Part 1 of 3); Refer to Job Control, Charts 04, 05

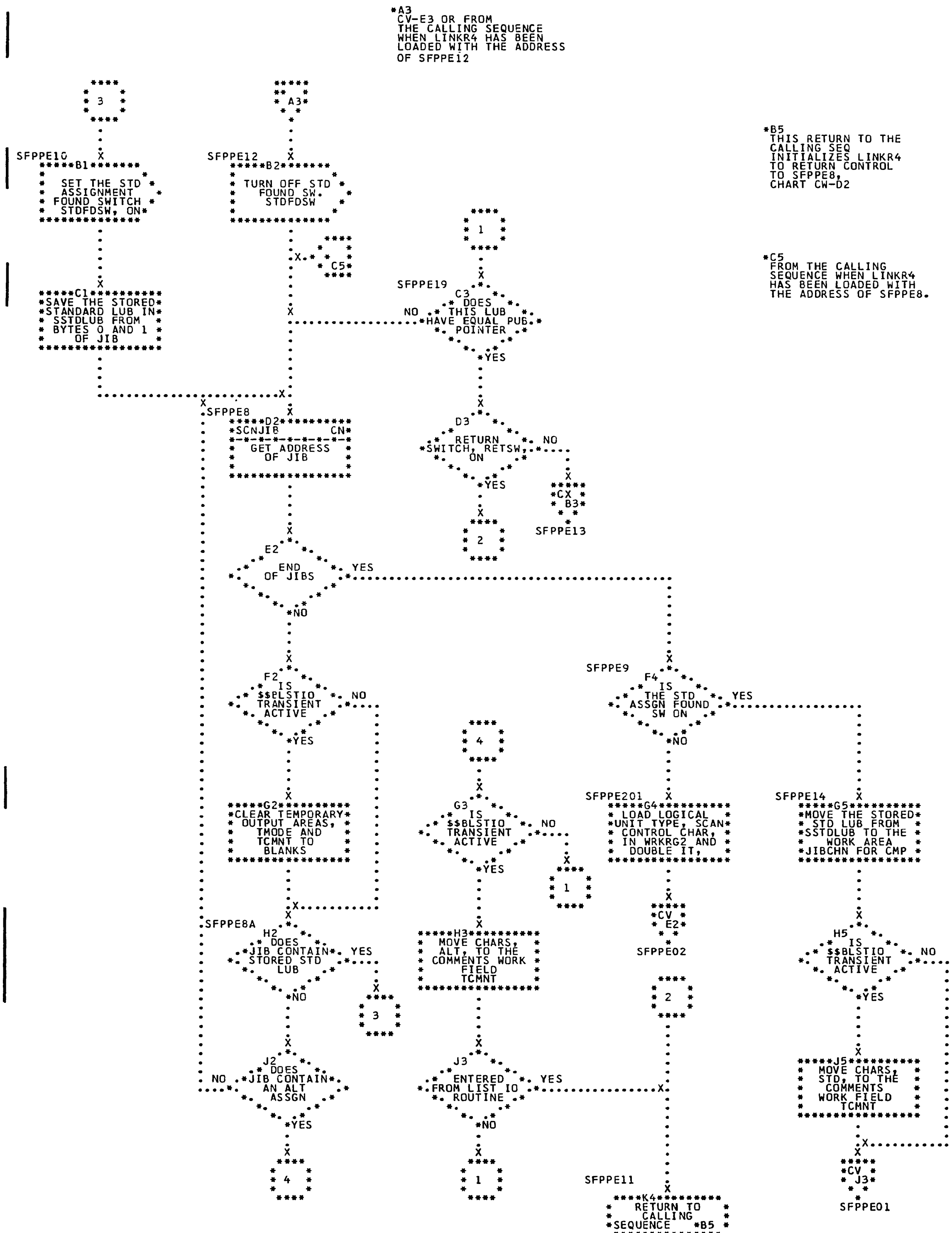


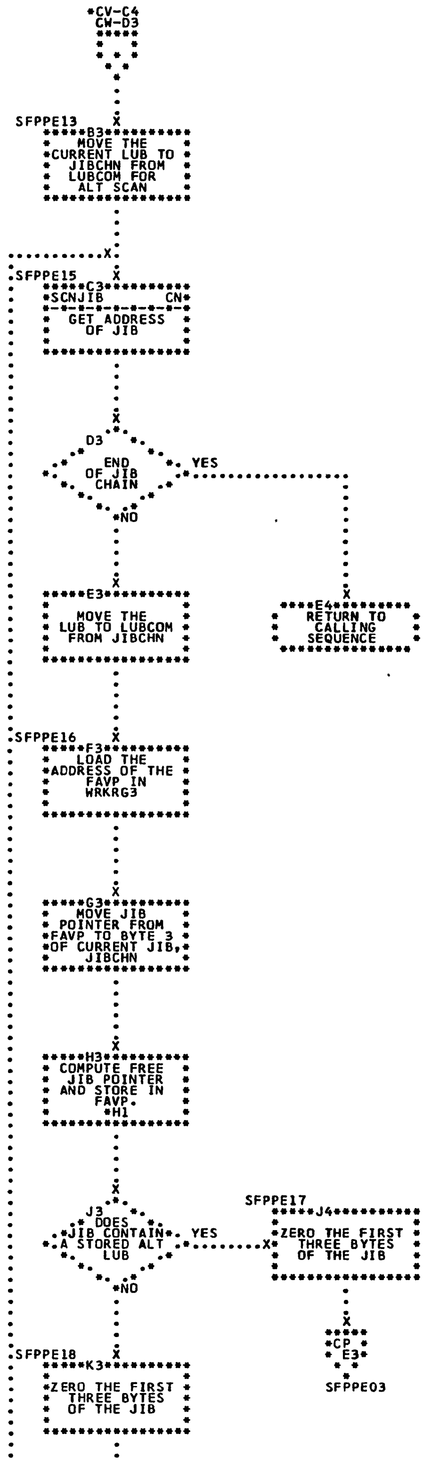

*H4 THIS RETURN TO THE CALLING SEQUENCE INITIALIZES LINKR4 TO RETURN CONTROL IN LINE.

*J4 THIS RETURN TO THE CALLING SEQUENCE IS MADE WHEN THE END OF THE LUBS IS DETECTED. LINKR4 IS NOT INITIALIZED TO RETURN CONTROL.

*K4 THIS RETURN TO THE CALLING SEQUENCE INITIALIZES LINKR4 TO RETURN CONTROL TO SFPPE12,CHART CW-B2

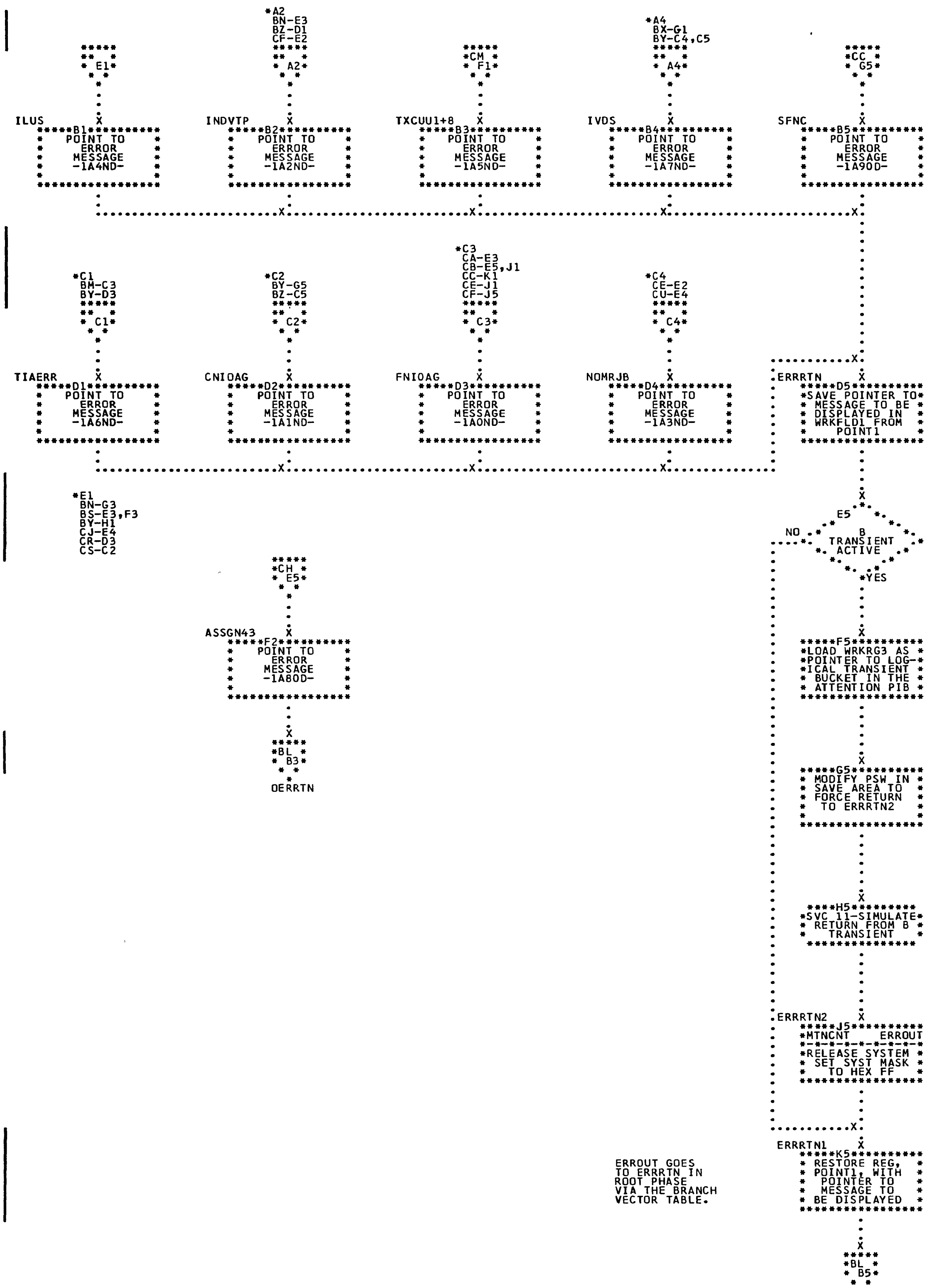
Chart CW. Subroutine-- $JOBCTLD (SFPPE; Part 2 of 3); Refer to Job Control, Charts 04, 05






*H1
1. THE FREE JIB ADDRESS IS LOADED IN WRKRG4 FROM POINT3.
2. THE JIB TABLE ADDRESS IS SUBTRACTED FROM WRKRG4 LEAVING THE FREE JIB DISPLACEMENT IN WRKRG4.
3. THE DISPLACEMENT IN WRKRG4 IS DIVIDED BY 4 TO DEVELOP THE FREE JIB POINTER.
4. THE RESULTANT POINTER IS STORED IN FAVP FROM WRKRG4.

Chart CY. Error Routines-- $JOBCTLD (ILUS, INDVTP, TXCUU1+8, IVDS, SFNC, TIAERR, CNIOAG, FNIOAG, NOMRJB, ASSGN43, and ERRRTN); Refer to Job Control, Charts 04, 05



# Chart DA. CANCEL, and STOP Statement Processors-- $JOBCTLG; Refer to Job Control, Charts 07, 08

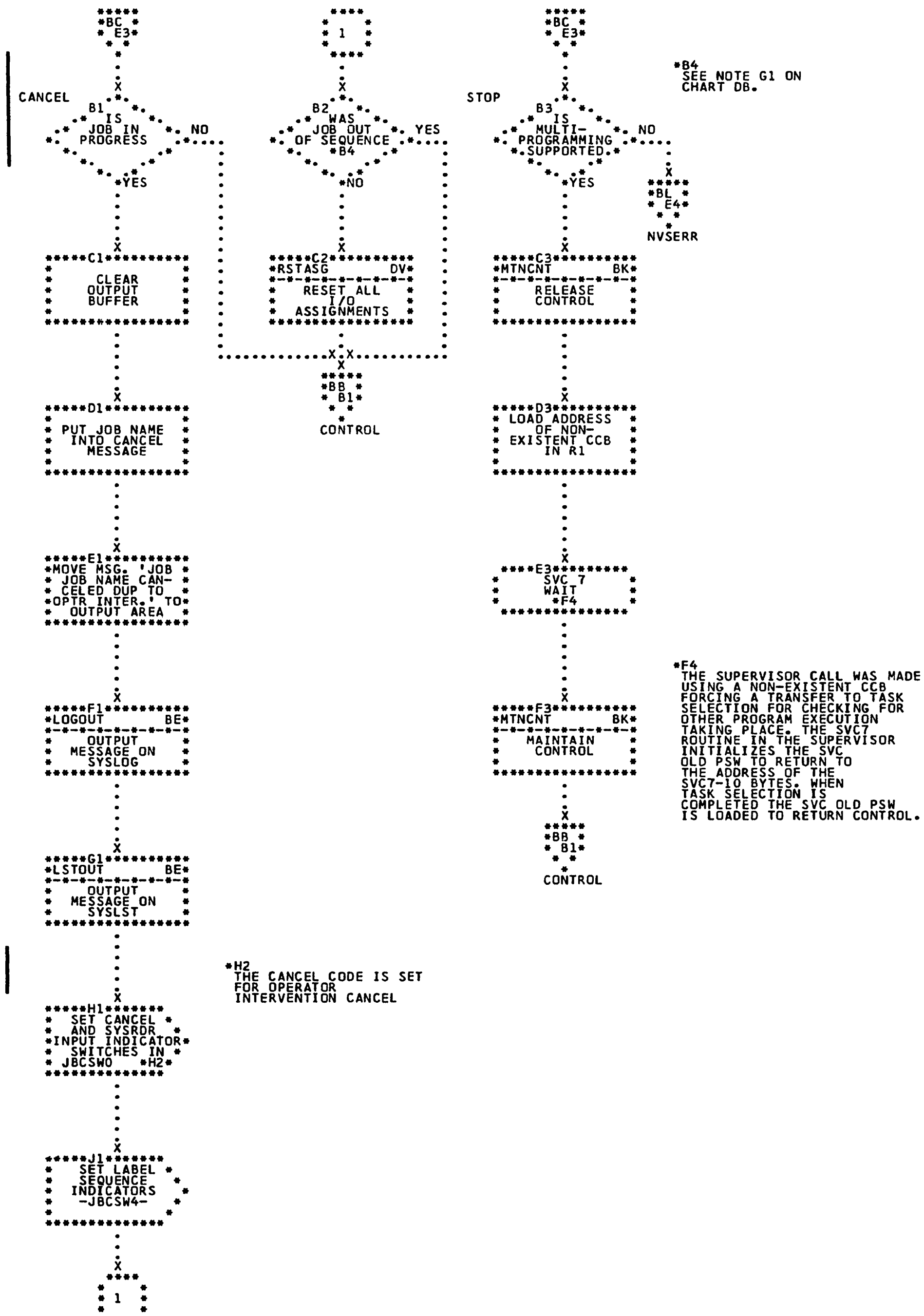


Chart DB. EOJ (/&) Statement Processor- $JOBCTLG (Part 1 of 2); Refer to Job Control, Chart 07

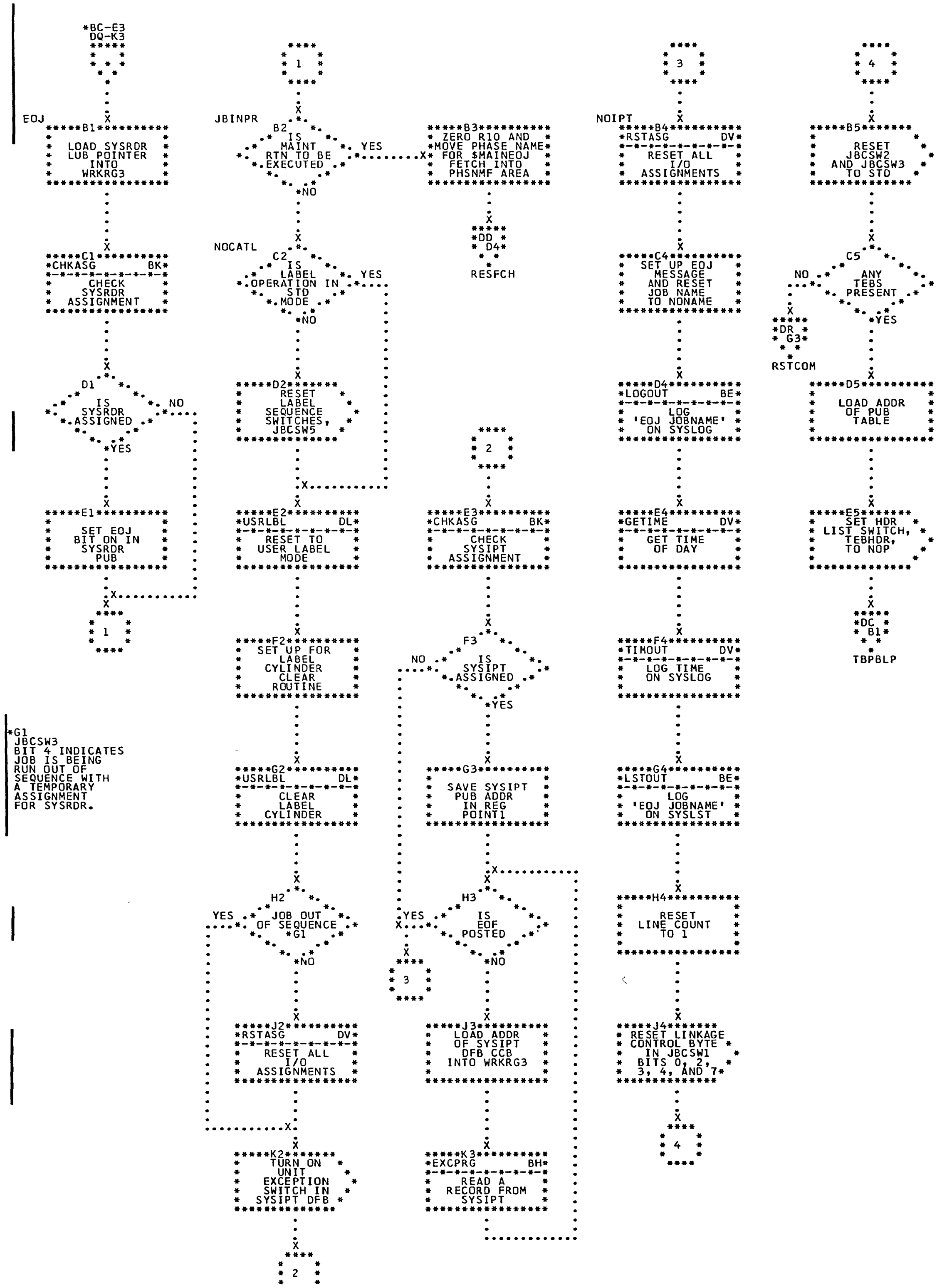


Chart DC. EOJ (/&) Statement Processor- $JOBCTLG (Part 2 of 2); Refer to Job Control, Chart 07

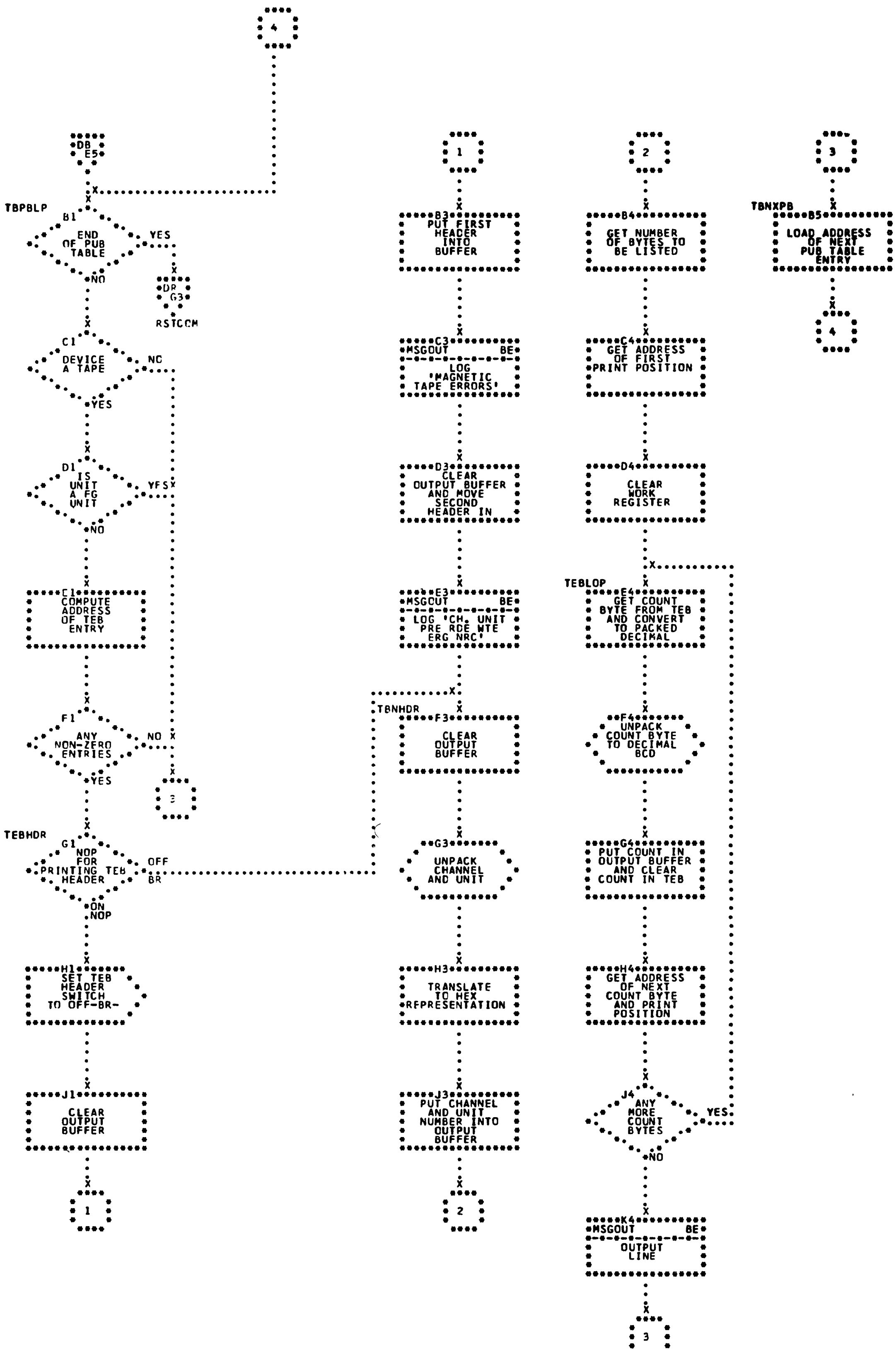
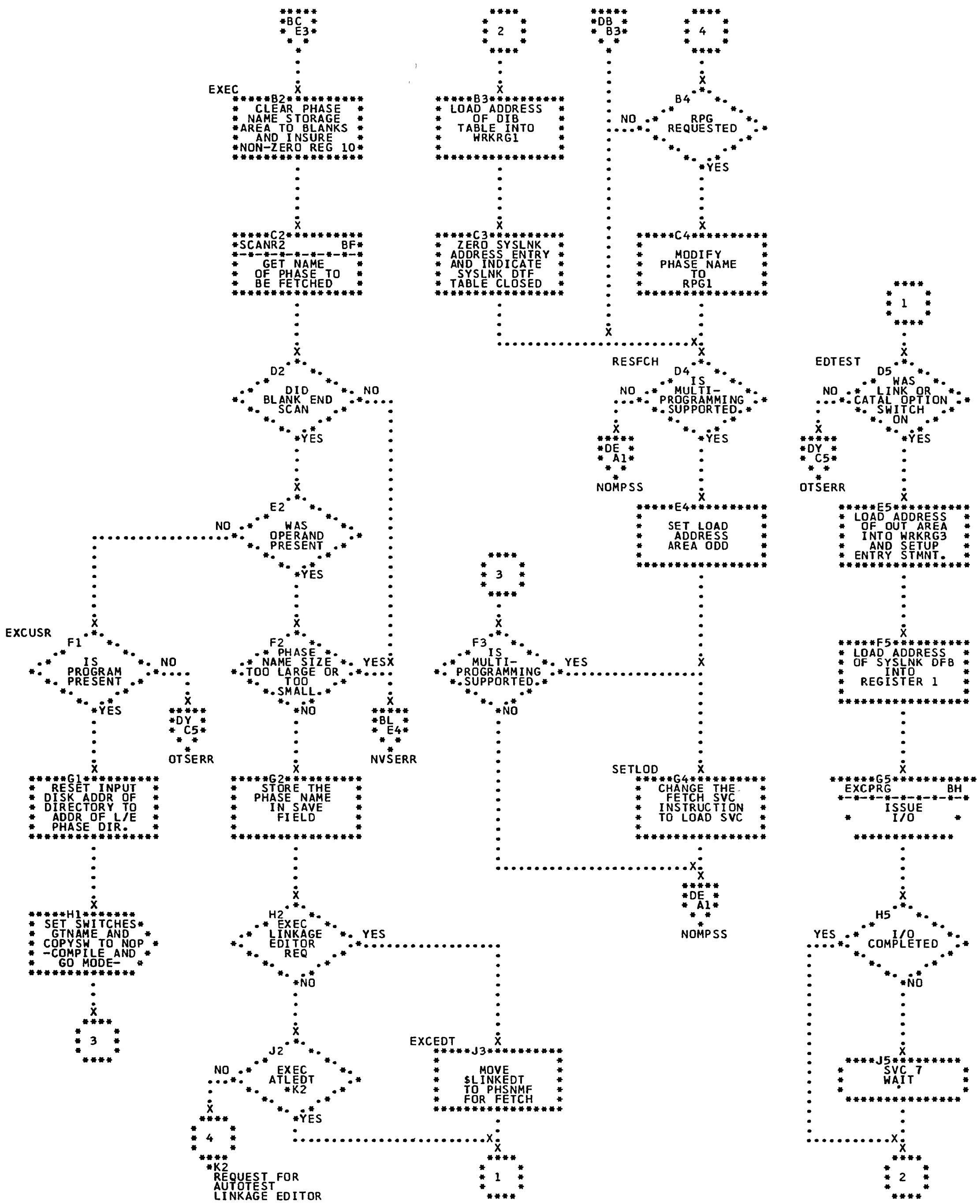


Chart DD. EXEC Statement Processor- $JOBCTLG (Part 1 of 4); Refer to Job Control, Chart 08



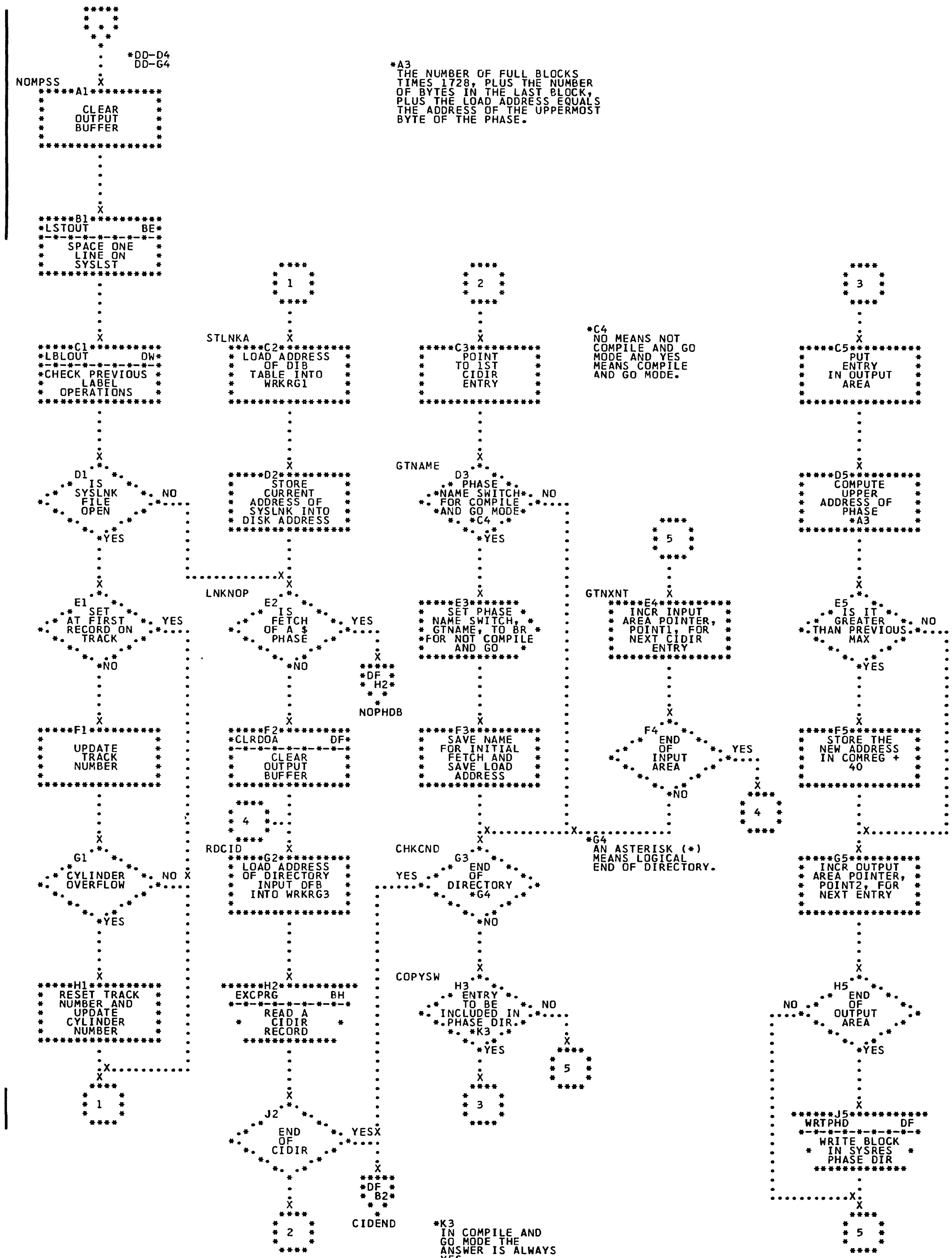
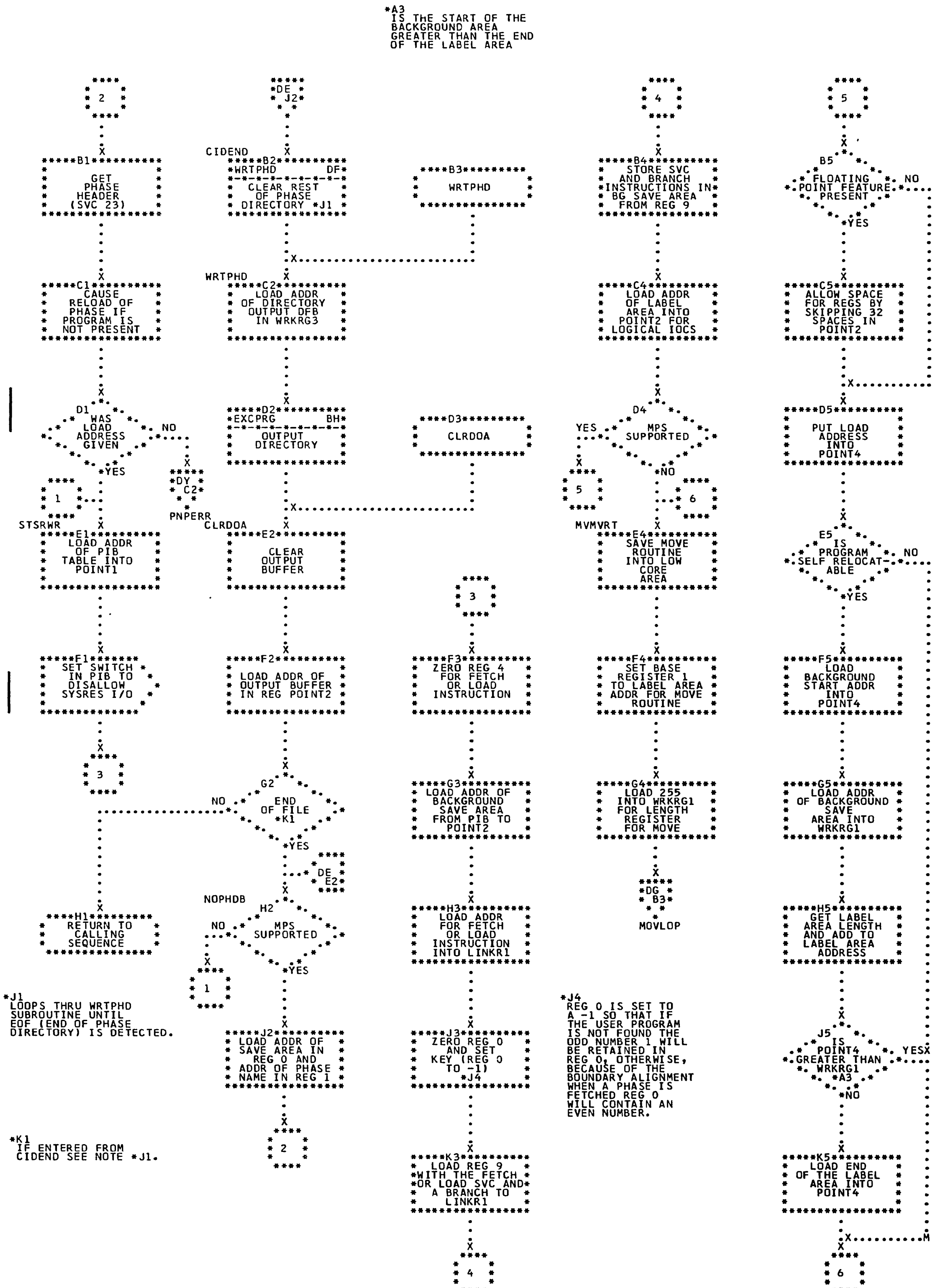

NOMPSS
A1 CLEAR OUTPUT BUFFER
B1 LSTOUT BE SPACE ONE LINE ON SYSLST
C1 LBLOUT DW CHECK PREVIOUS LABEL OPERATIONS
D1 IS SYSLNK FILE OPEN
E1 SET AT FIRST RECORD ON TRACK
F1 UPDATE TRACK NUMBER
G1 CYLINDER OVERFLOW
H1 RESET TRACK NUMBER AND UPDATE CYLINDER NUMBER
STLNKA
C2 LOAD ADDRESS OF DIB TABLE INTO WRKRG1
D2 STORE CURRENT ADDRESS OF SYSLNK INTO DISK ADDRESS
LNKNOP
E2 IS FETCH OF A $ PHASE
NOPHDB
F2 CLRDOA DF CLEAR OUTPUT BUFFER
RDCID
G2 LOAD ADDRESS OF DIRECTORY INPUT DFB INTO WRKRG3
H2 EXCPRG BH READ A CIDIR RECORD
J2 END OF CIDIR
CIDEND
*A3 THE NUMBER OF FULL BLOCKS TIMES 1728, PLUS THE NUMBER OF BYTES IN THE LAST BLOCK, PLUS THE LOAD ADDRESS EQUALS THE ADDRESS OF THE UPPERMOST BYTE OF THE PHASE.
C3 POINT TO 1ST CIDIR ENTRY
GTNAME
D3 PHASE NAME SWITCH FOR COMPILE AND GO MODE *C4
E3 SET PHASE NAME SWITCH, GTNAME, TO BR FOR NOT COMPILE AND GO
F3 SAVE NAME FOR INITIAL FETCH AND SAVE LOAD ADDRESS
CHKCND
G3 END OF DIRECTORY *G4
COPYSW
H3 ENTRY TO BE INCLUDED IN PHASE DIR. *K3
*C4 NO MEANS NOT COMPILE AND GO MODE AND YES MEANS COMPILE AND GO MODE.
GTNXNT
E4 INCR INPUT AREA POINTER, POINT1, FOR NEXT CIDIR ENTRY
F4 END OF INPUT AREA
*G4 AN ASTERISK (*) MEANS LOGICAL END OF DIRECTORY.
*K3 IN COMPILE AND GO MODE THE ANSWER IS ALWAYS YES.
C5 PUT ENTRY IN OUTPUT AREA
D5 COMPUTE UPPER ADDRESS OF PHASE *A3
E5 IS IT GREATER THAN PREVIOUS MAX
F5 STORE THE NEW ADDRESS IN COMREG + 40
G5 INCR OUTPUT AREA POINTER, POINT2, FOR NEXT ENTRY
H5 END OF OUTPUT AREA
J5 WRTPHD DF WRITE BLOCK IN SYSRES PHASE DIR

Chart DF. EXEC Statement Processor- $JOBCTLG (Part 3 of 4); Refer to Job Control, Chart 08



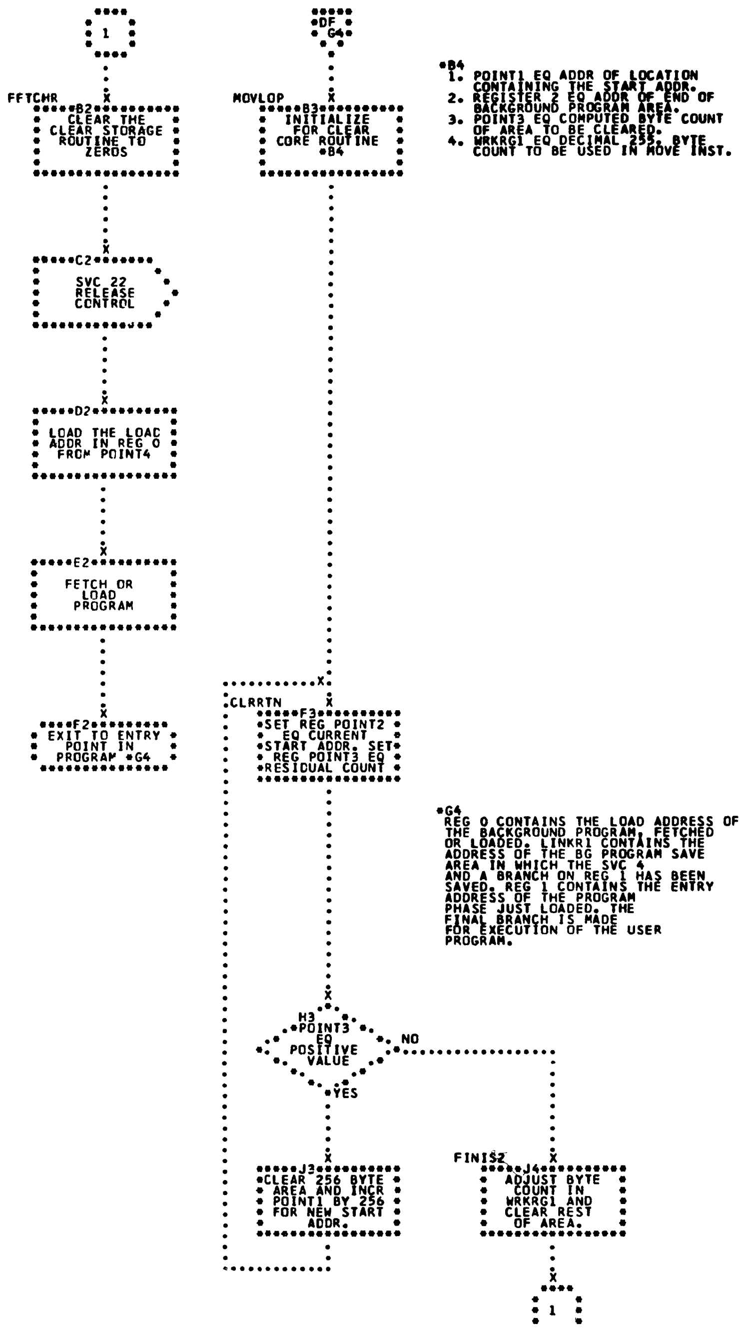

1
DF G4
FFTCHR
B2
CLEAR THE CLEAR STORAGE ROUTINE TO ZEROS
MOVLOP
B3
INITIALIZE FOR CLEAR CORE ROUTINE *B4
*B4
1. POINT1 EQ ADDR OF LOCATION CONTAINING THE START ADDR.
2. REGISTER 2 EQ ADDR OF END OF BACKGROUND PROGRAM AREA.
3. POINT3 EQ COMPUTED BYTE COUNT OF AREA TO BE CLEARED.
4. WRKRG1 EQ DECIMAL 255, BYTE COUNT TO BE USED IN MOVE INST.
C2
SVC 22 RELEASE CONTROL
D2
LOAD THE LOAD ADDR IN REG 0 FROM POINT4
E2
FETCH OR LOAD PROGRAM
F2
EXIT TO ENTRY POINT IN PROGRAM *G4
CLRRTN
F3
SET REG POINT2 EQ CURRENT START ADDR. SET REG POINT3 EQ RESIDUAL COUNT
*G4
REG 0 CONTAINS THE LOAD ADDRESS OF THE BACKGROUND PROGRAM, FETCHED OR LOADED. LINKR1 CONTAINS THE ADDRESS OF THE BG PROGRAM SAVE AREA IN WHICH THE SVC 4 AND A BRANCH ON REG 1 HAS BEEN SAVED. REG 1 CONTAINS THE ENTRY ADDRESS OF THE PROGRAM PHASE JUST LOADED. THE FINAL BRANCH IS MADE FOR EXECUTION OF THE USER PROGRAM.
H3
POINT3 EQ POSITIVE VALUE
NO
YES
J3
CLEAR 256 BYTE AREA AND INCR POINT1 BY 256 FOR NEW START ADDR.
FINIS2
J4
ADJUST BYTE COUNT IN WRKRG1 AND CLEAR REST OF AREA.
1

Chart DH. PAUSE, LOG and NOLOG Statement Processors-- $JOBCTLG; Refer to Job Control, Chart 08

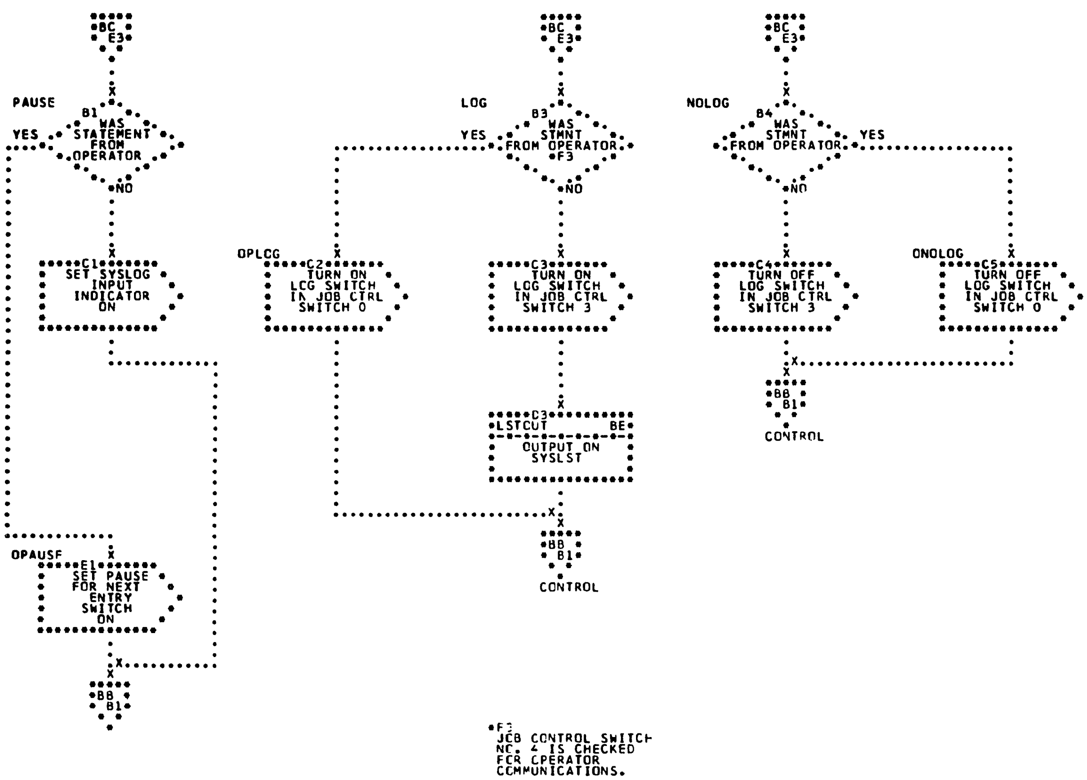
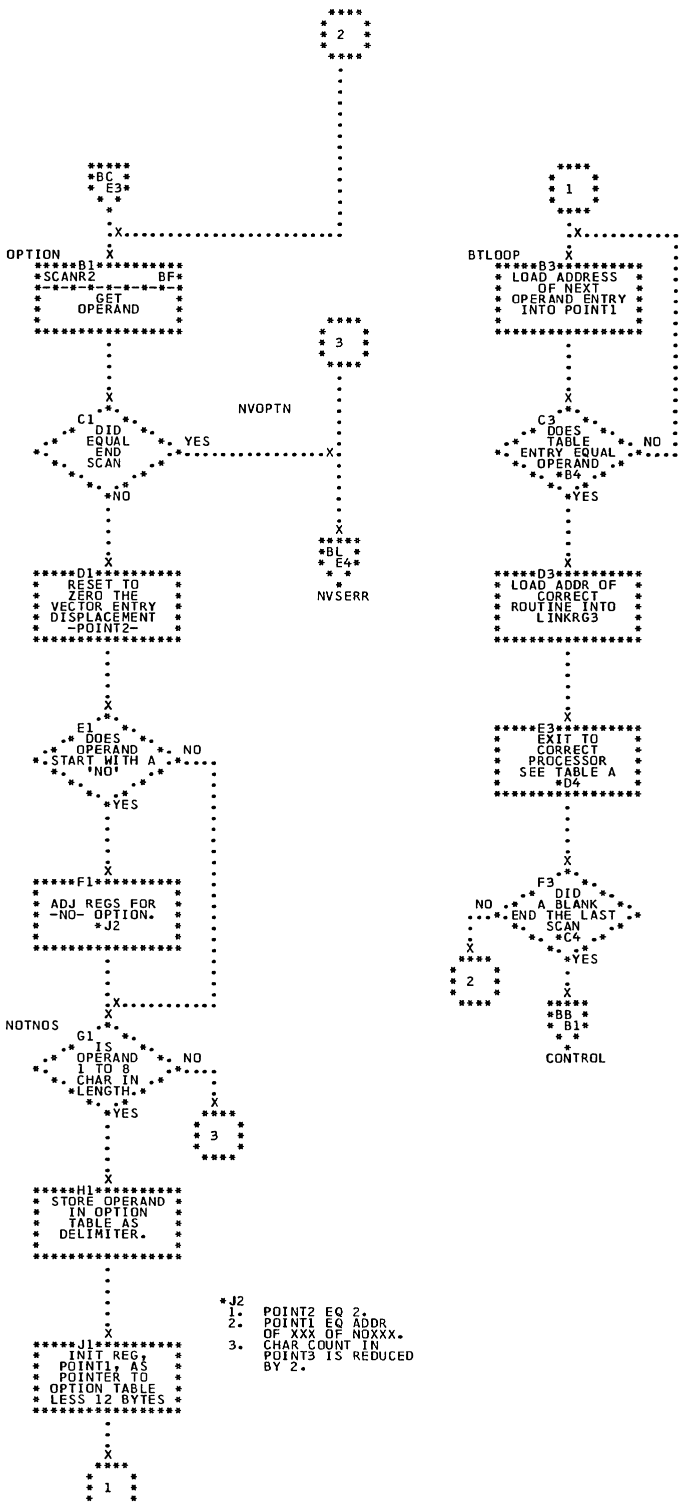


Chart DJ. OPTION Statement Processor- $JOBCTLG (Part 1 of 3); Refer to Job Control, Chart 06




*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.

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

TABLE A

| OPERAND | CHART |
|---|---|
| DECK | DK |
| NODECK | DK |
| LIST | DK |
| NOLIST | DK |
| LISTX | DK |
| NLISTX | DK |
| SYM | DK |
| NOSYM | DK |
| XREF | DK |
| NOXREF | DK |
| ERRS | DK |
| NOERRS | DK |
| 48C | DK |
| 60C | DK |
| NOLOG | DK |
| LOG | DL |
| USRLABEL | DL |
| STDLABEL | DL |
| DUMP | DL |
| NODUMP | DL |
| LINK | DL |
| NOLINK | DL |
| CATAL | DL |

Chart DK. OPTION Statement Processor- $JOBCTLG (Part 2 of 3); Refer to Job Control, Chart 06

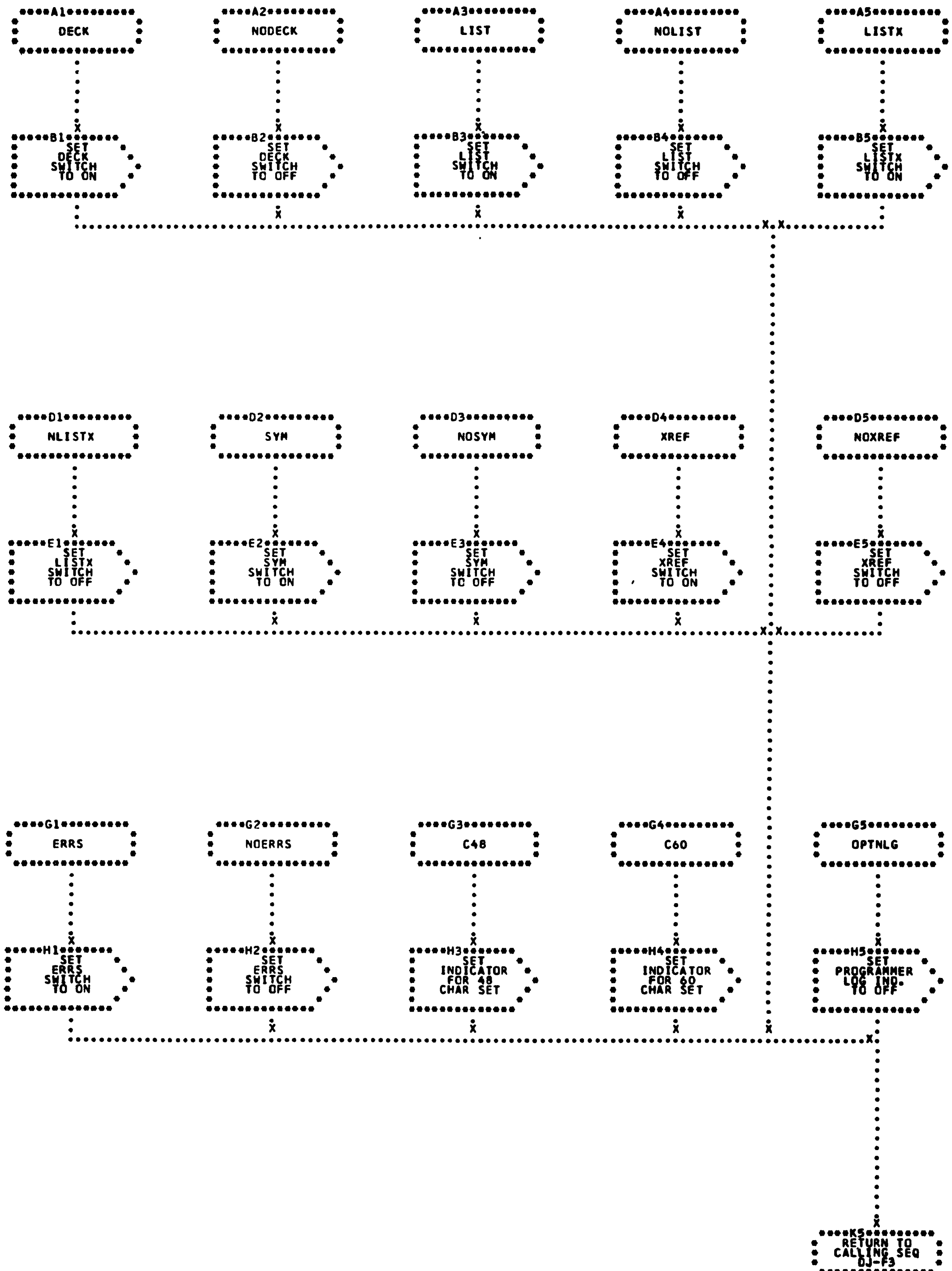


Chart DL. OPTION Statement Processor- $JOBCTLG (Part 3 of 3); Refer to Job Control, Chart 06

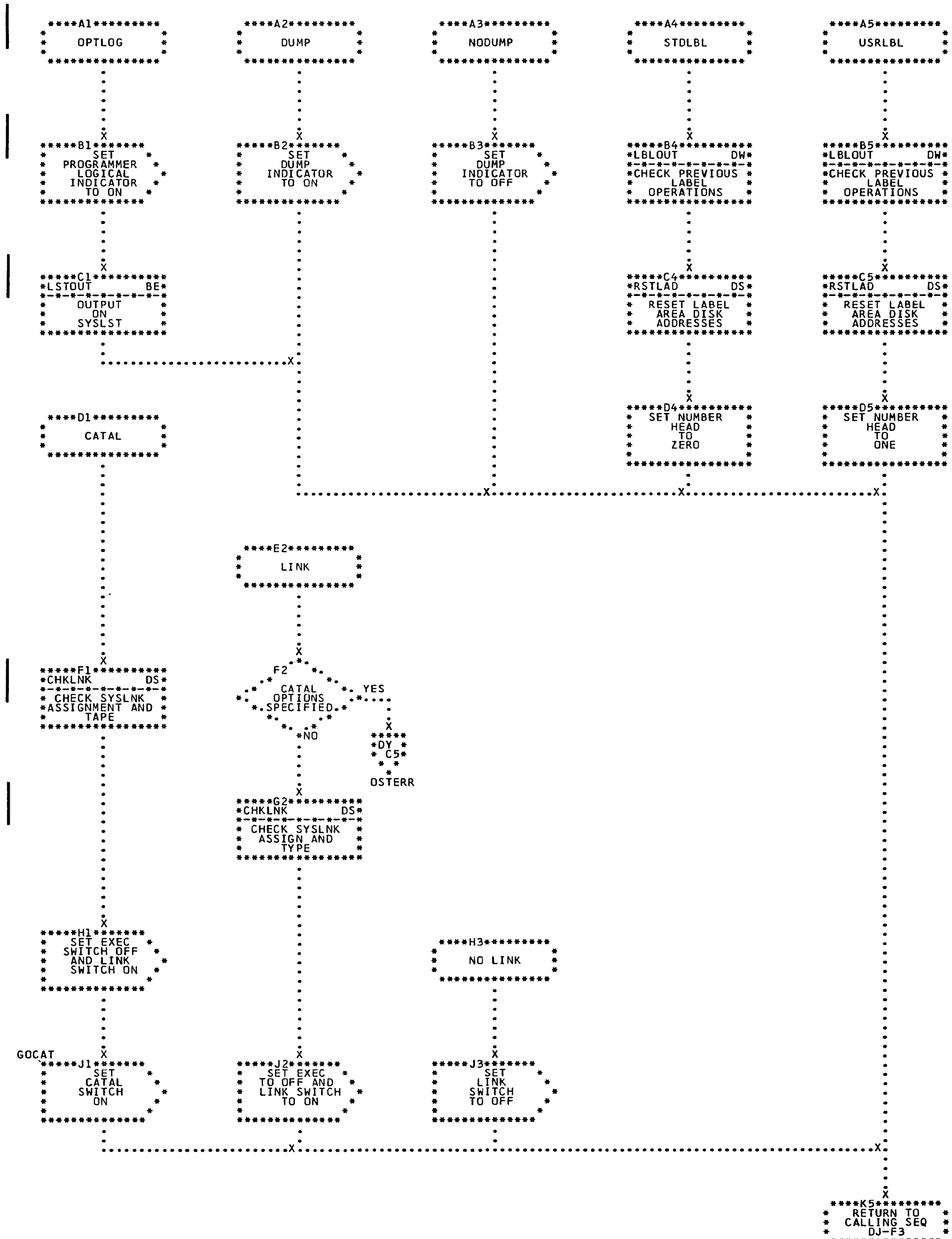
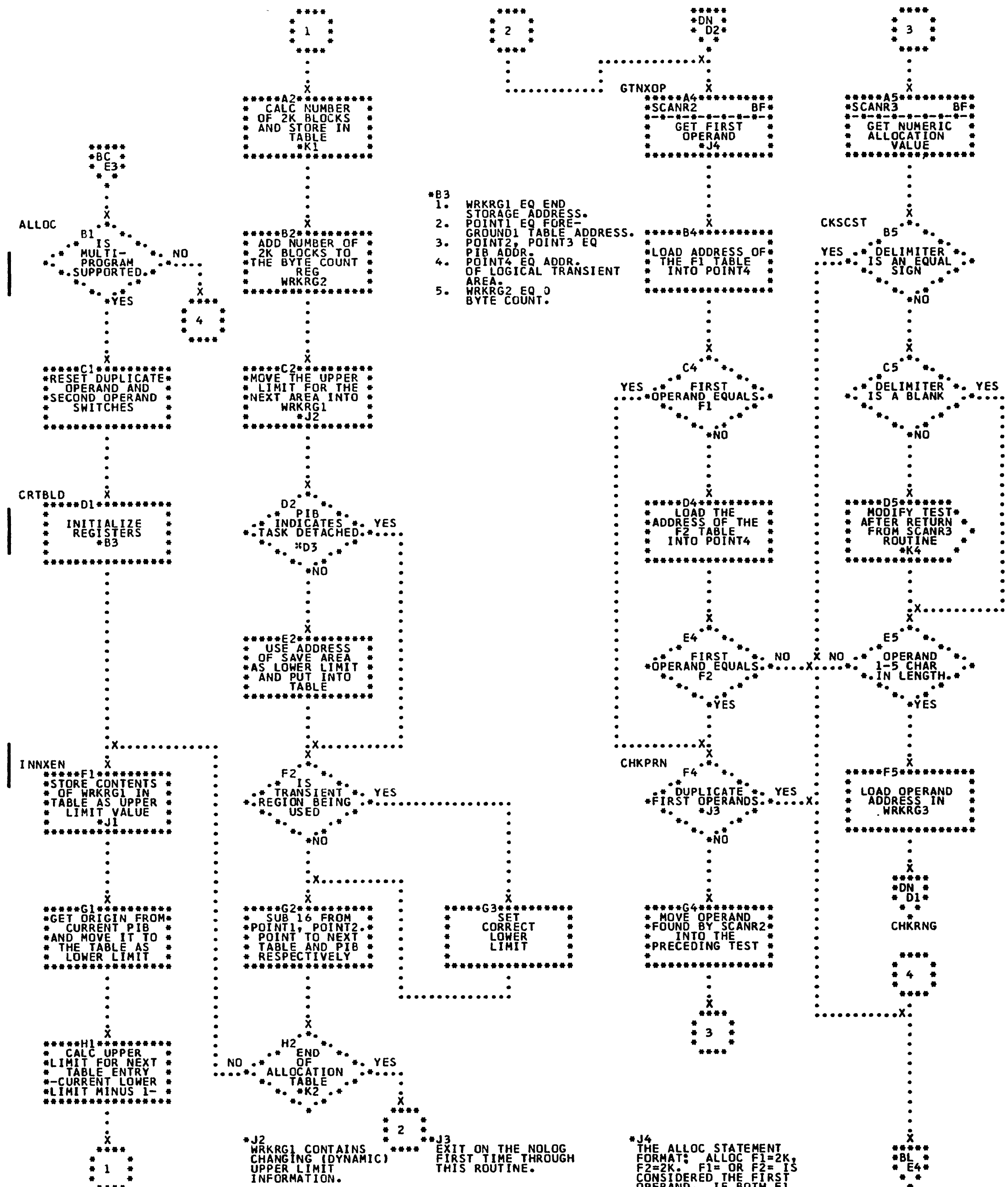


Chart DM. ALLOC Statement Processor- $JOBCTLG (Part 1 of 3); Refer to Job Control, Chart 08




*B3
1. WRKRG1 EQ END STORAGE ADDRESS.
2. POINT1 EQ FOREGROUND1 TABLE ADDRESS.
3. POINT2, POINT3 EQ PIB ADDR.
4. POINT4 EQ ADDR. OF LOGICAL TRANSIENT AREA.
5. WRKRG2 EQ 0 BYTE COUNT.

*J1
THIS VALUE IS ALSO STORED IN THE ACTIVE LOWER LIMIT FIELD INDICATING THIS FIELD IS INACTIVE. IT CONTAINS A HEX FF IN THE LAST BYTE.

*J2
WRKRG1 CONTAINS CHANGING (DYNAMIC) UPPER LIMIT INFORMATION.

*J3
EXIT ON THE NOLOG FIRST TIME THROUGH THIS ROUTINE.

*J4
THE ALLOC STATEMENT FORMAT: ALLOC F1=2K, F2=2K. F1= OR F2= IS CONSIDERED THE FIRST OPERAND. IF BOTH F1 AND F2 ARE SPECIFIED SCANR2 IS USED TWICE.

*K2
THIS LOOP IS EXECUTED UNTIL 3 TABLES, ONE FOR EACH TYPE OF PROBLEM PROGRAM, ARE BUILT. THE TABLES ARE BUILT IN THE SEQUENCE F1, F2, AND BG. THE SEQUENCE IS A REVERSAL OF THE PHYSICAL ORDER. SEE LABEL LIST FOR ATABLE, F2TBEN AND F1TBEN EXPANSIONS.

*K4
THE INSTRUCTION AT LOCATION CKSCST IS MODIFIED TO BRANCH TO THE ERROR HANDLING ROUTINE IF THE SECOND OPERAND IS DELIMITED BY AN EQUAL SIGN OR A COMMA. THE COMMA IS ALLOWED TO DELIMIT THE FIRST OPERAND.

*K1
CURRENT UPPER LIMIT MINUS NEXT UPPER MLIMIT DIVIDED BY 2048.

Chart DN. ALLOC Statement Processor- $JOBCTLG (Part 2 of 3); Refer to Job Control, Chart 08

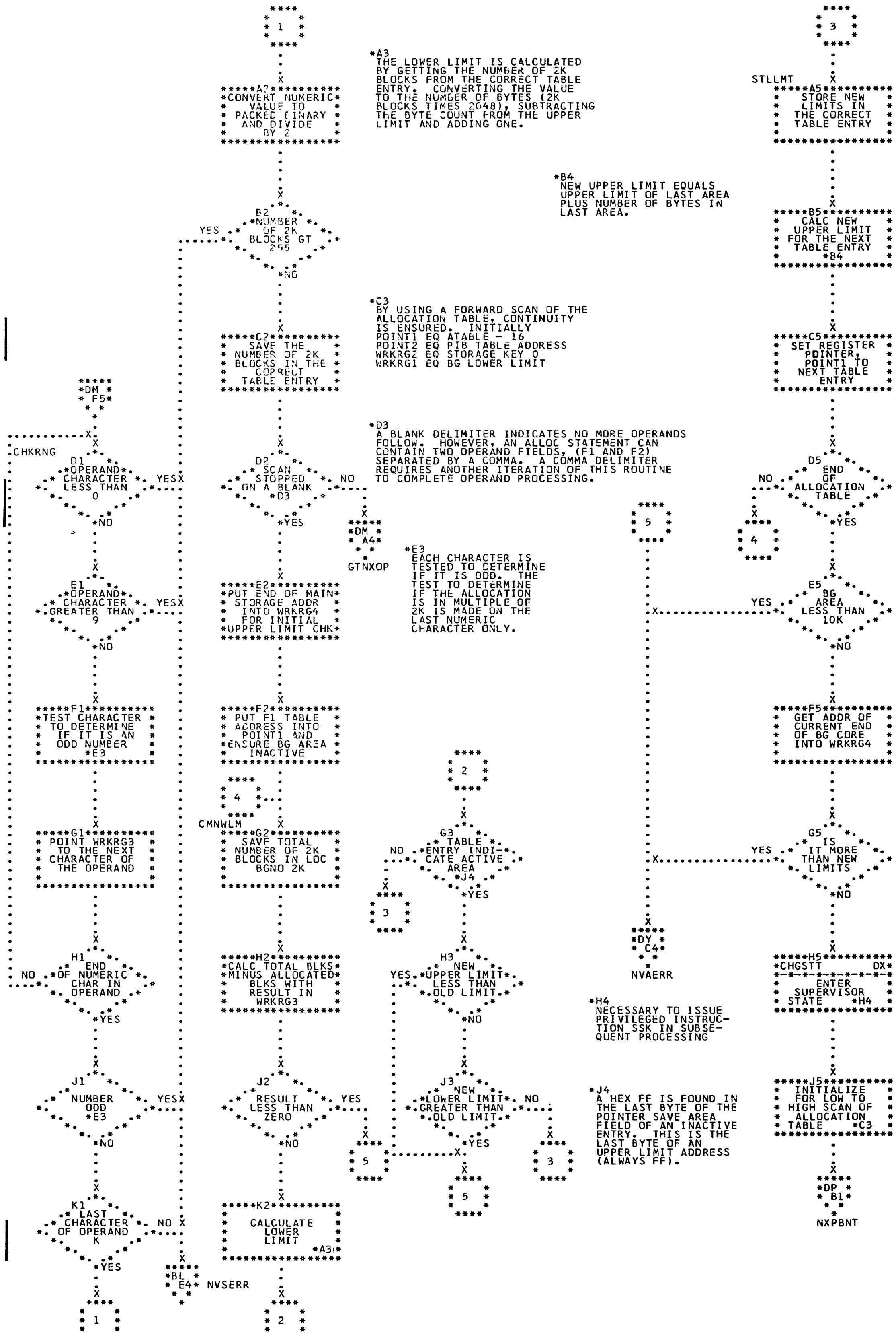


Chart DP. ALLOC Statement Processor (Part 3 of 3) and MAP Statement Processor-- $JOBCTLG; Refer to Job Control, Chart 08

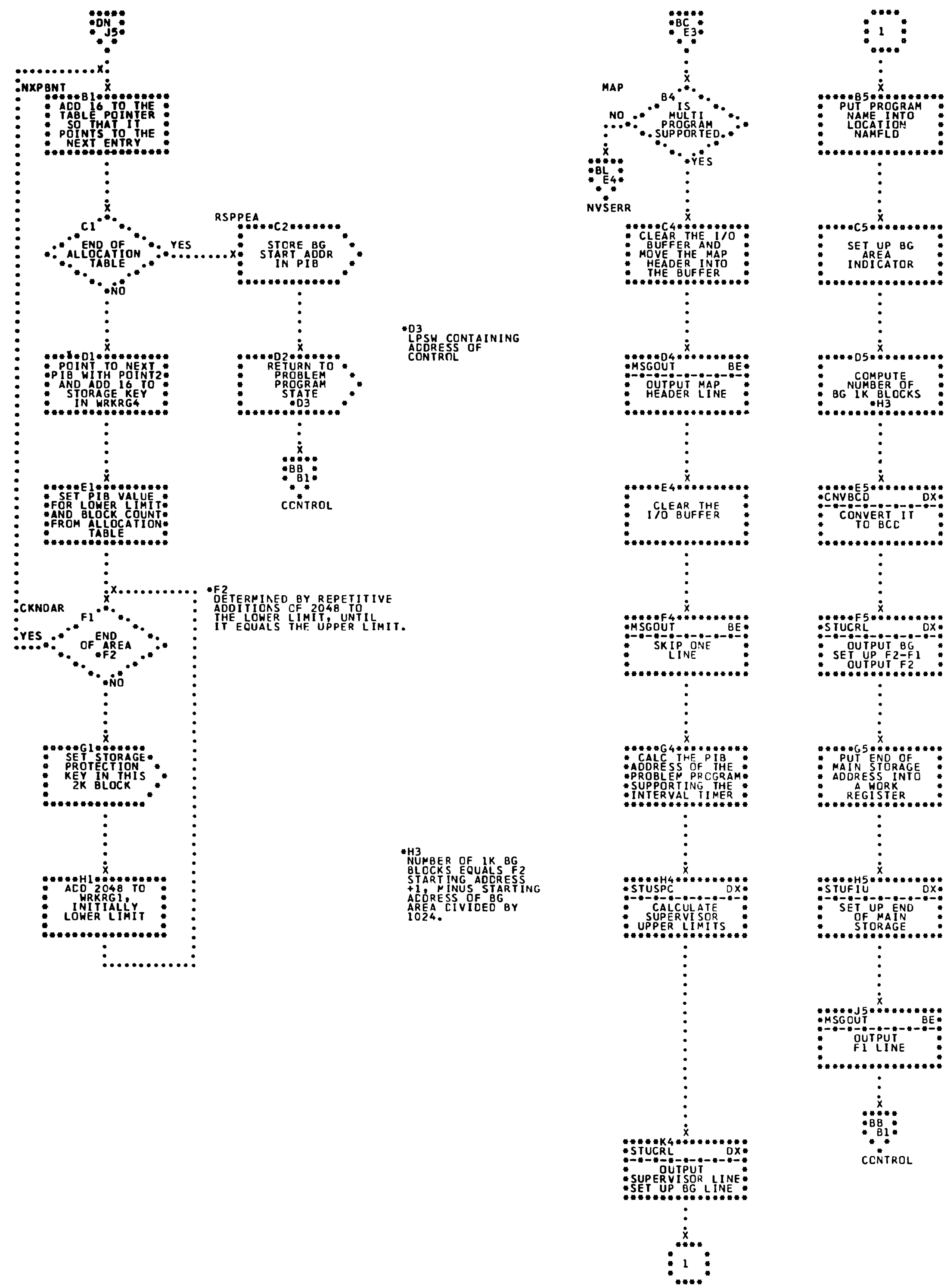


Chart DQ. JOB Statement Processor- $JOBCTLG (Part 1 of 2);
Refer to Job Control, Chart 07

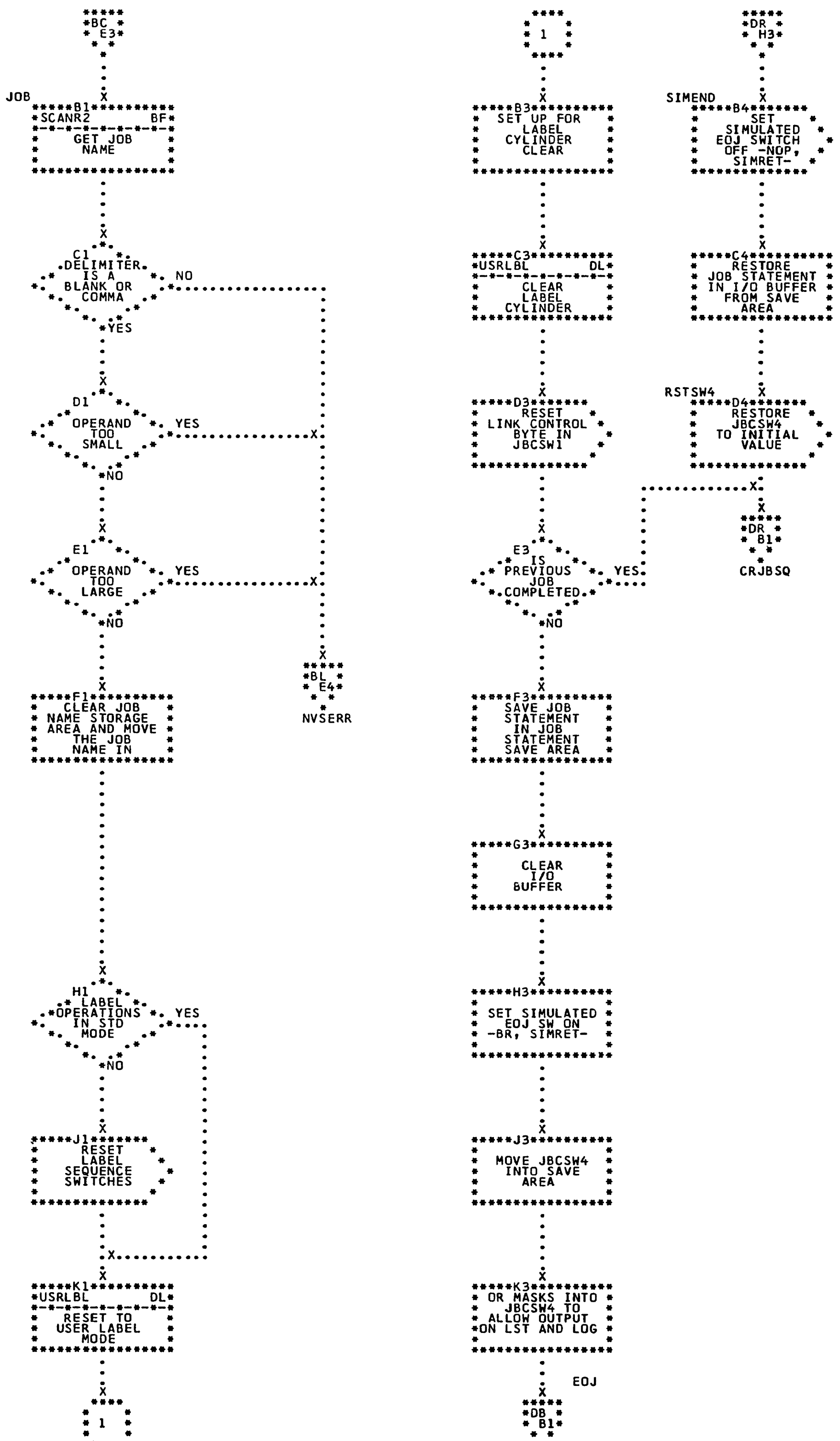


Chart DR. JOB Statement Processor- $JOBCTLG (Part 2 of 2);
Refer to Job Control, Chart 07

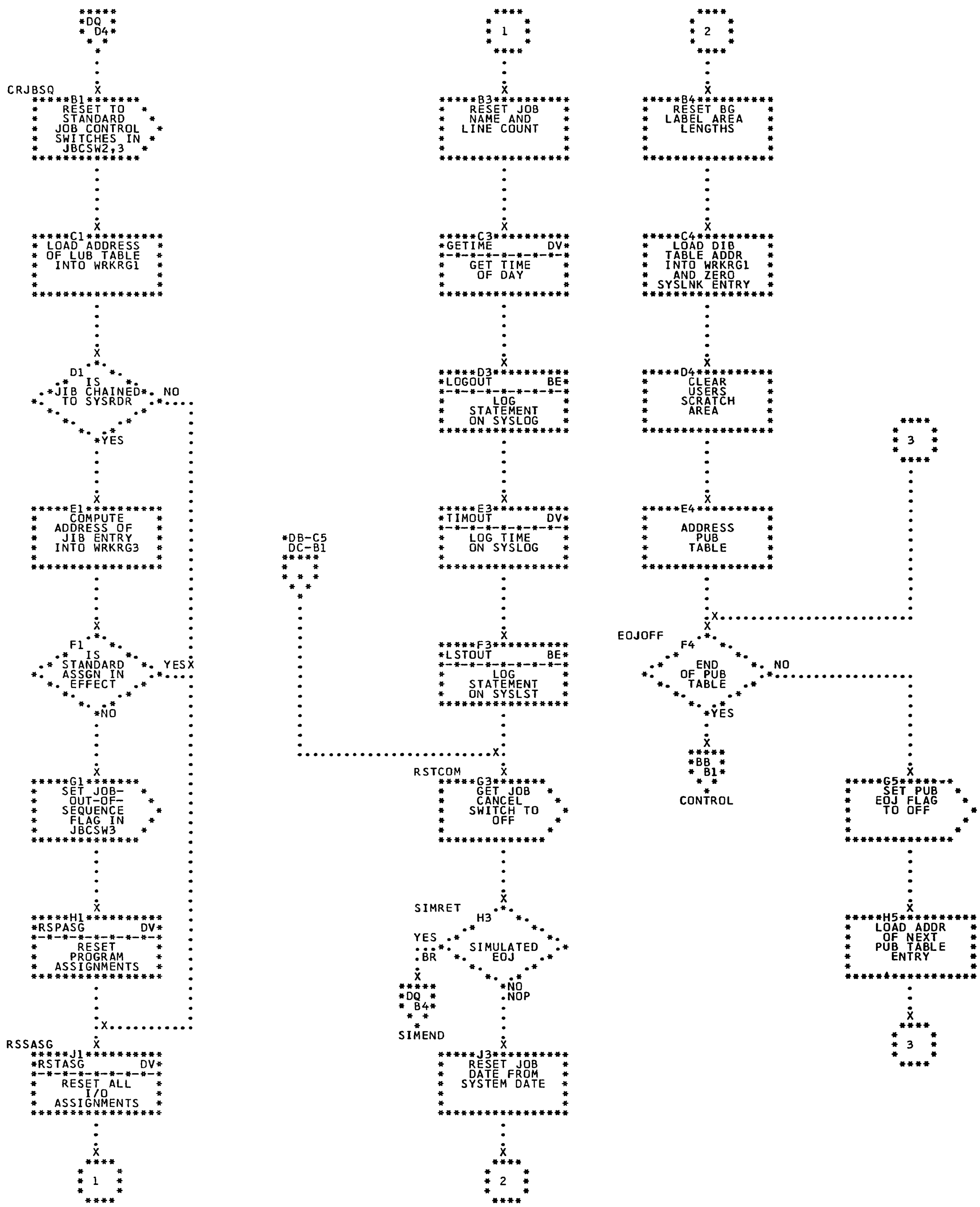


Chart DS. Subroutines-- $JOBCTLG (OPNLNK, RSTLAD, CHKLNK, IOROUT, and GTMXHN); Refer to Job Control, Charts 06-08

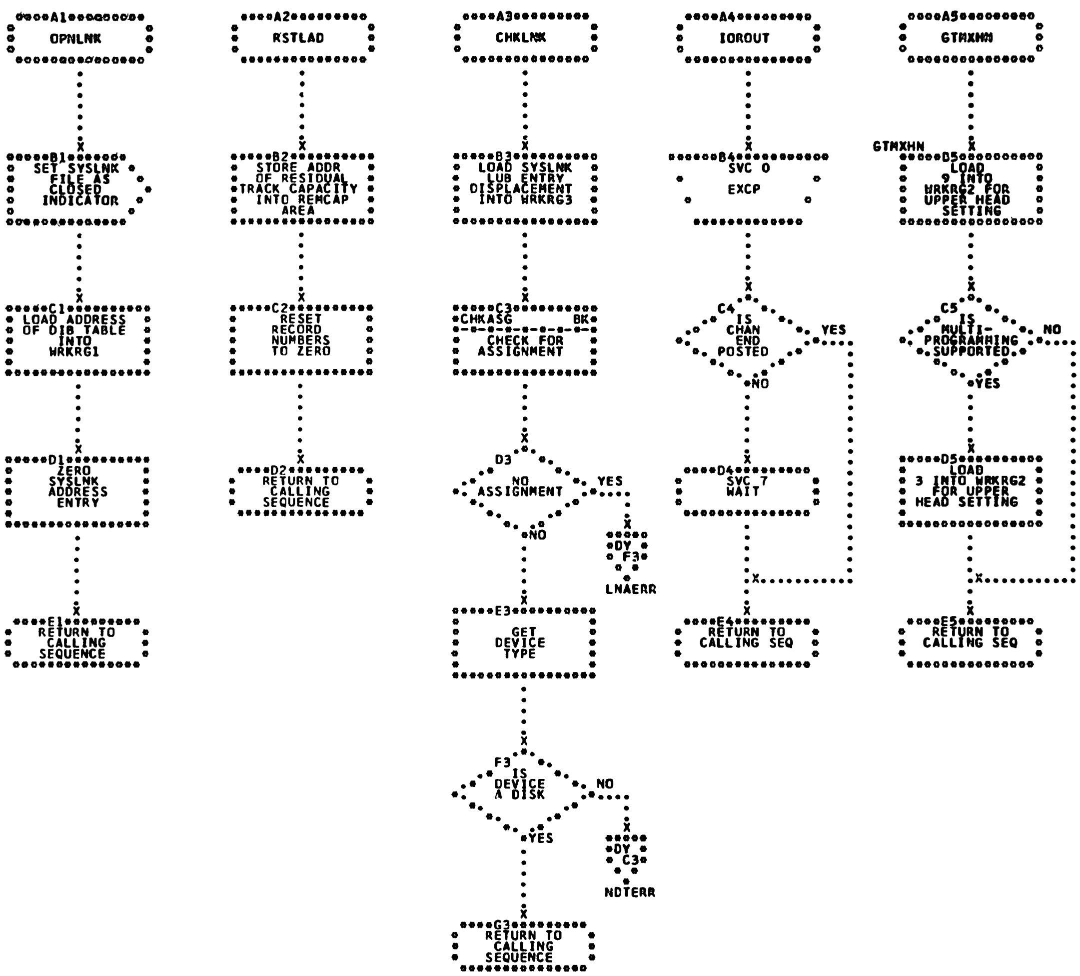


Chart DT. Subroutines-- $JOBCTLG (RASCAN, LUBSCN, GETPUB, and JIBSCN); Refer to Job Control, Charts 06-08

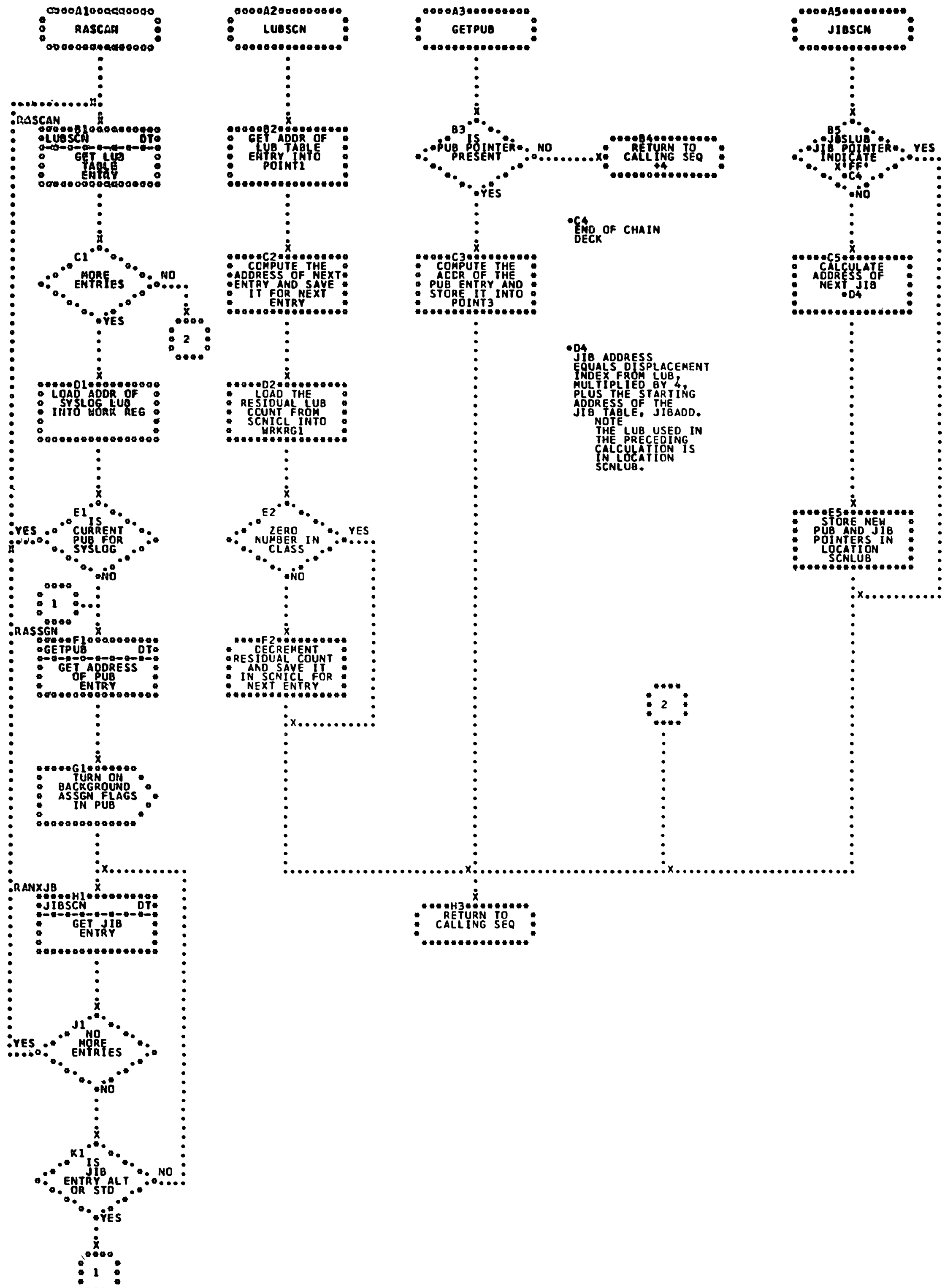


Chart DU. Subroutines-- $JOBCTLG (SCNINT, and UASCAN); Refer to Job Control, Charts 06-08

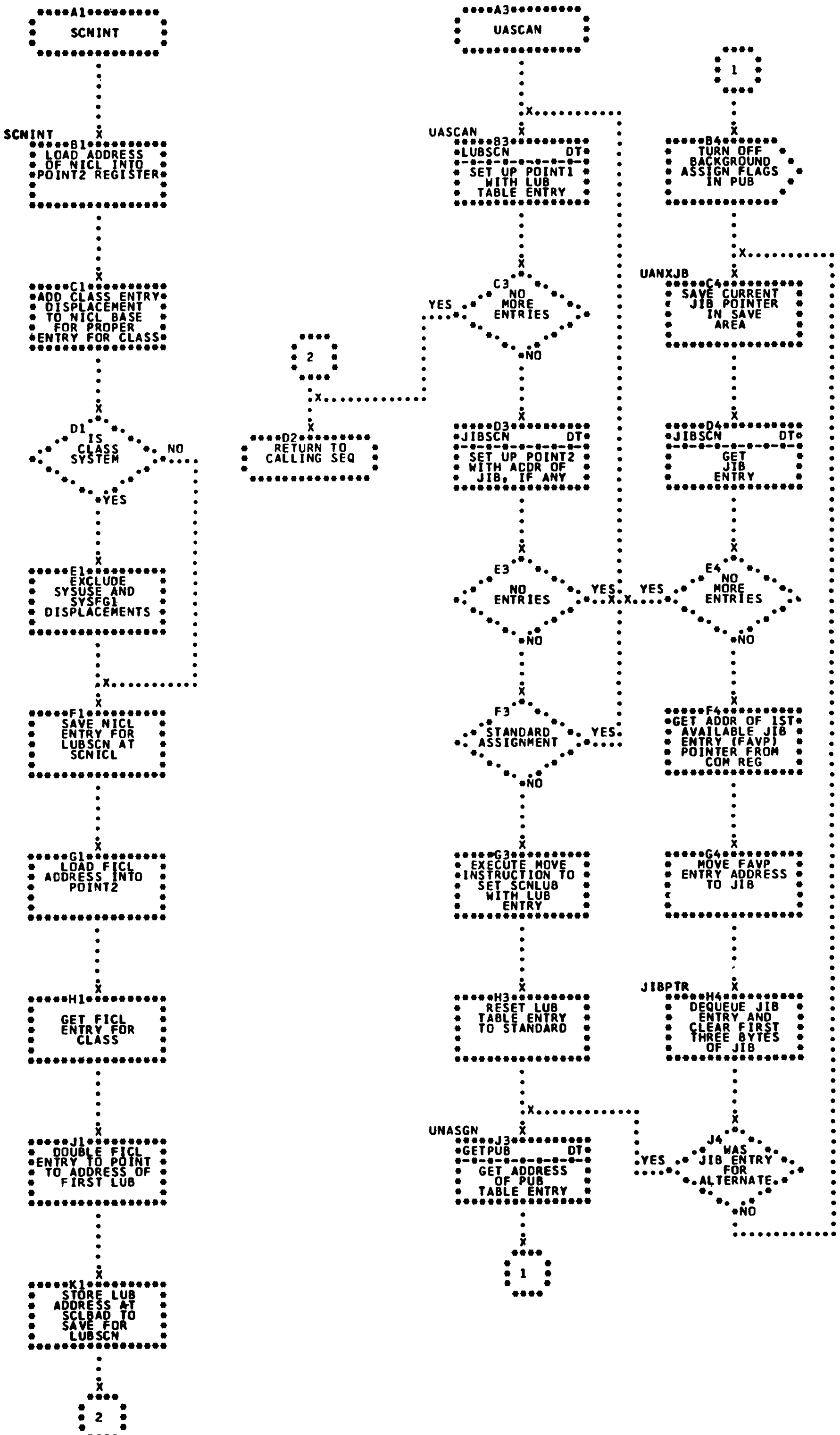


Chart DV. Subroutines-- $JOBCTLG (GETIME, TIMOUT, RSTASG, and RSPASG); Refer to Job Control, Charts 06-08

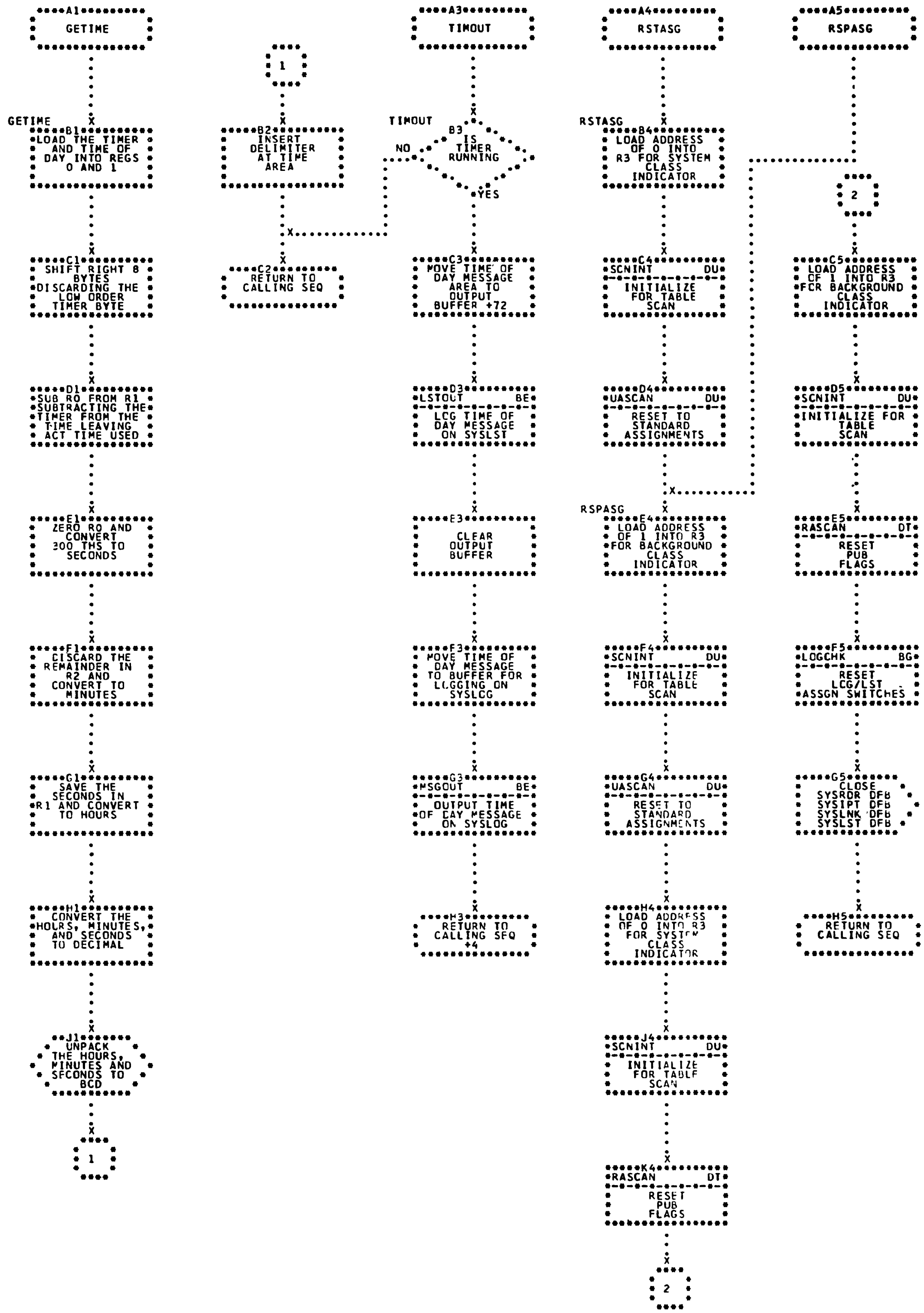


Chart DW. Subroutine-- $JOBCTLG (LBLOUT); Refer to Job Control, Charts 06-08

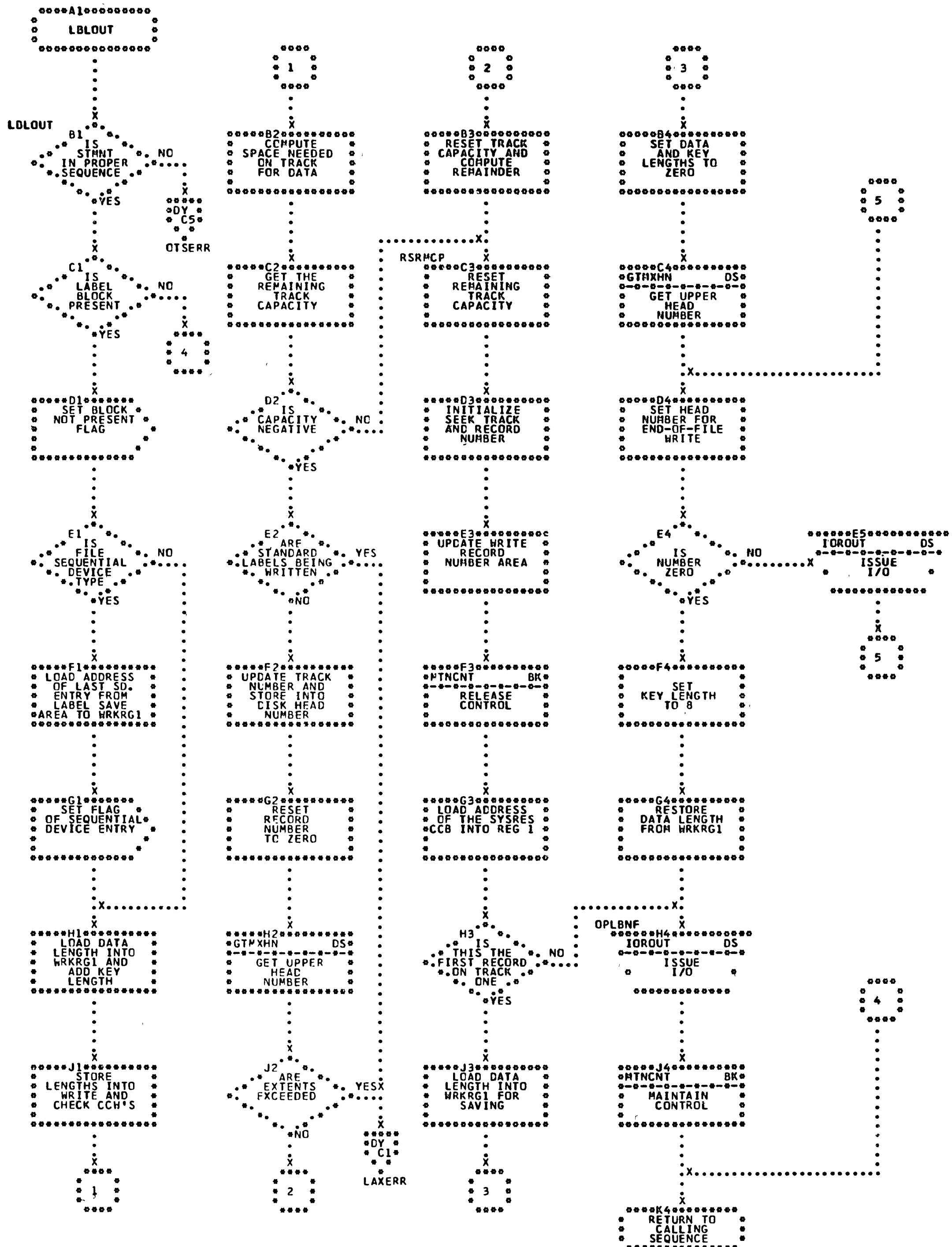


Chart DX. Subroutines-- $JOBCTLG (CNVBCD, CHGSTT, STUCRL, STUSPC, and STUFIU); Refer to Job Control, Charts 06-08

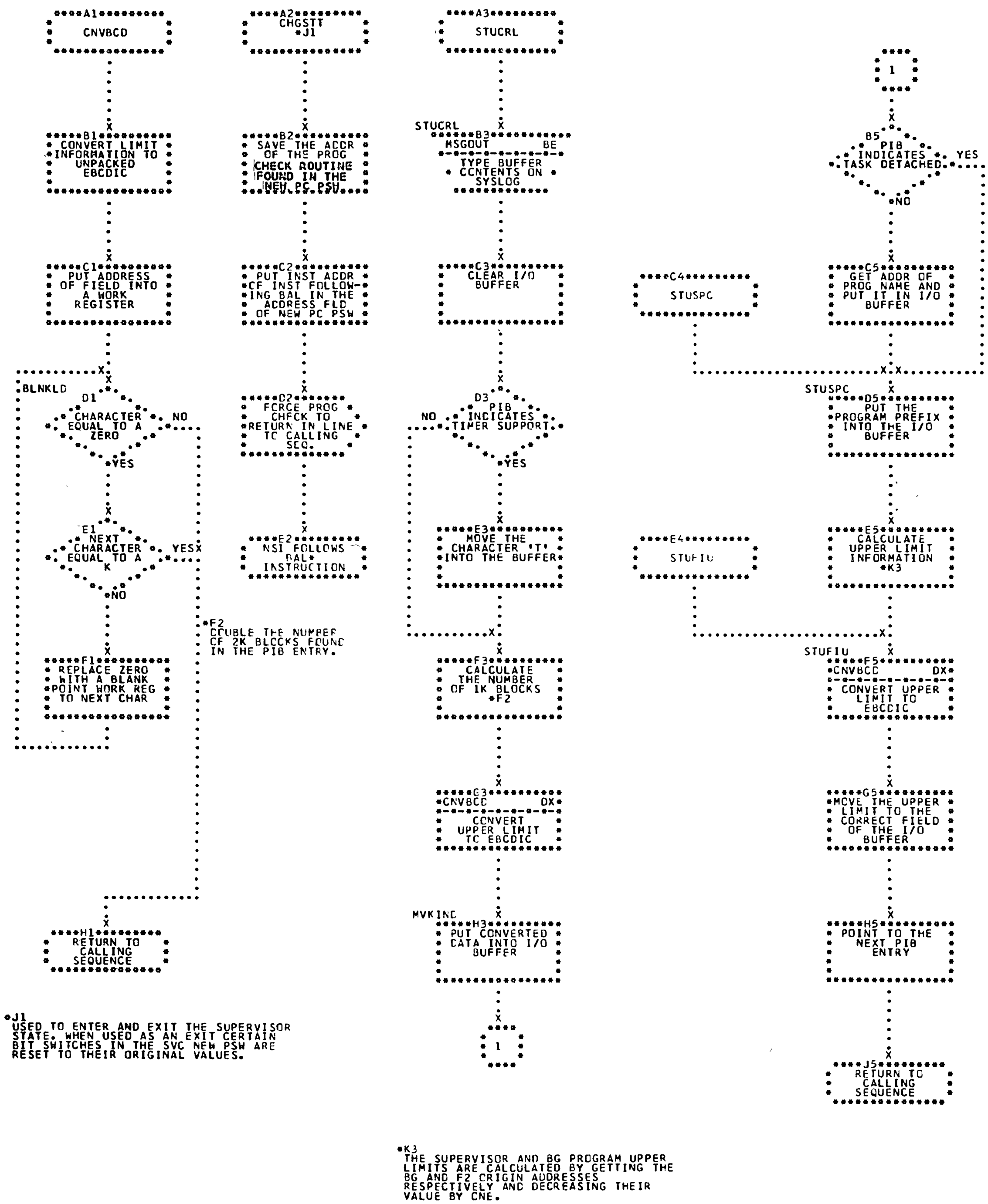


Chart DY. Error Routines-- $JOBCTLG (LAXERR, PNPERR, NDTERR, NVAERR, OTSERR, and LANERR); Refer to Job Control, Charts 06-08

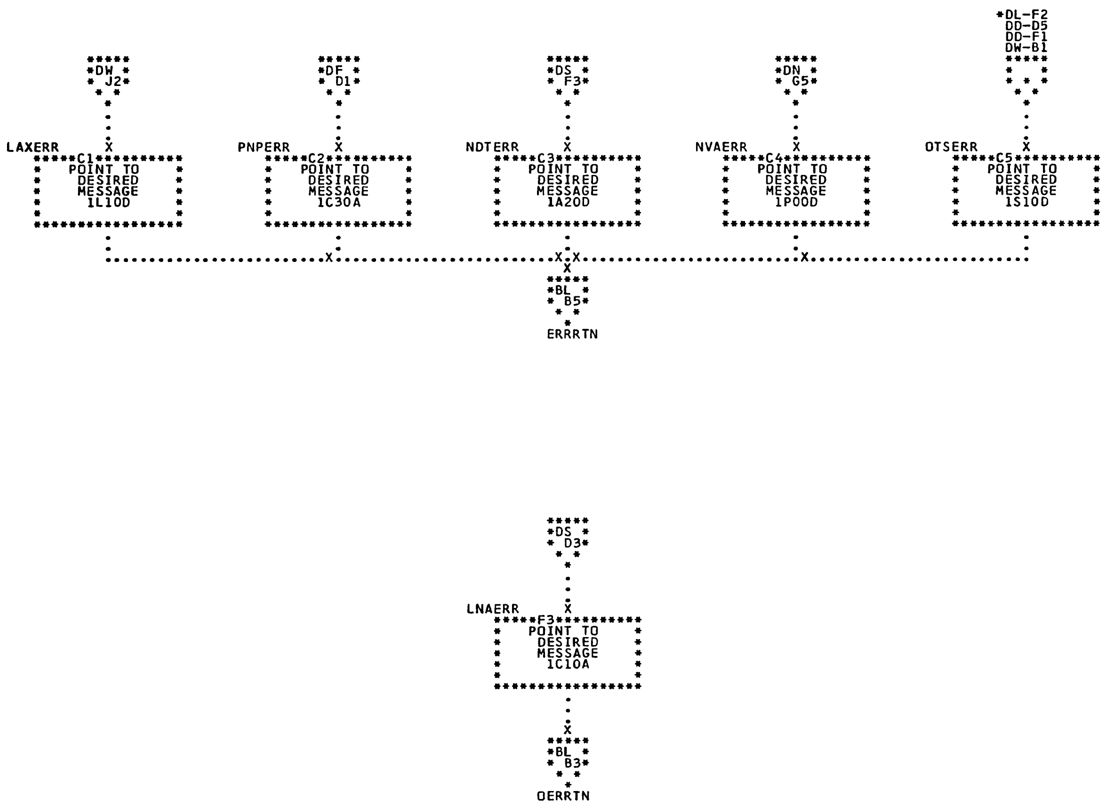


Chart EA. RELSE, and HOLD Statement Processors-- $JOBCTLJ; Refer to Job Control, Chart 11

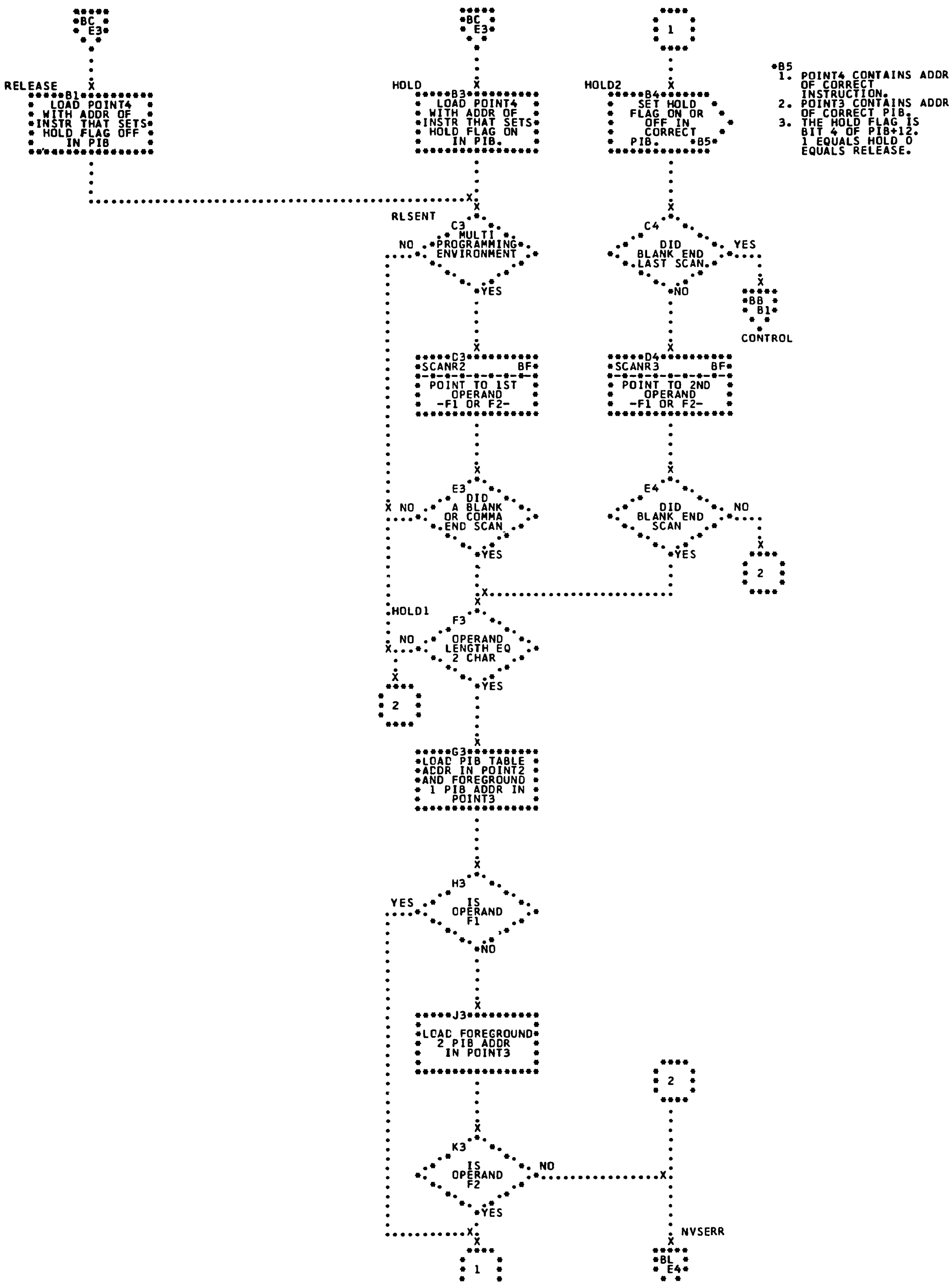
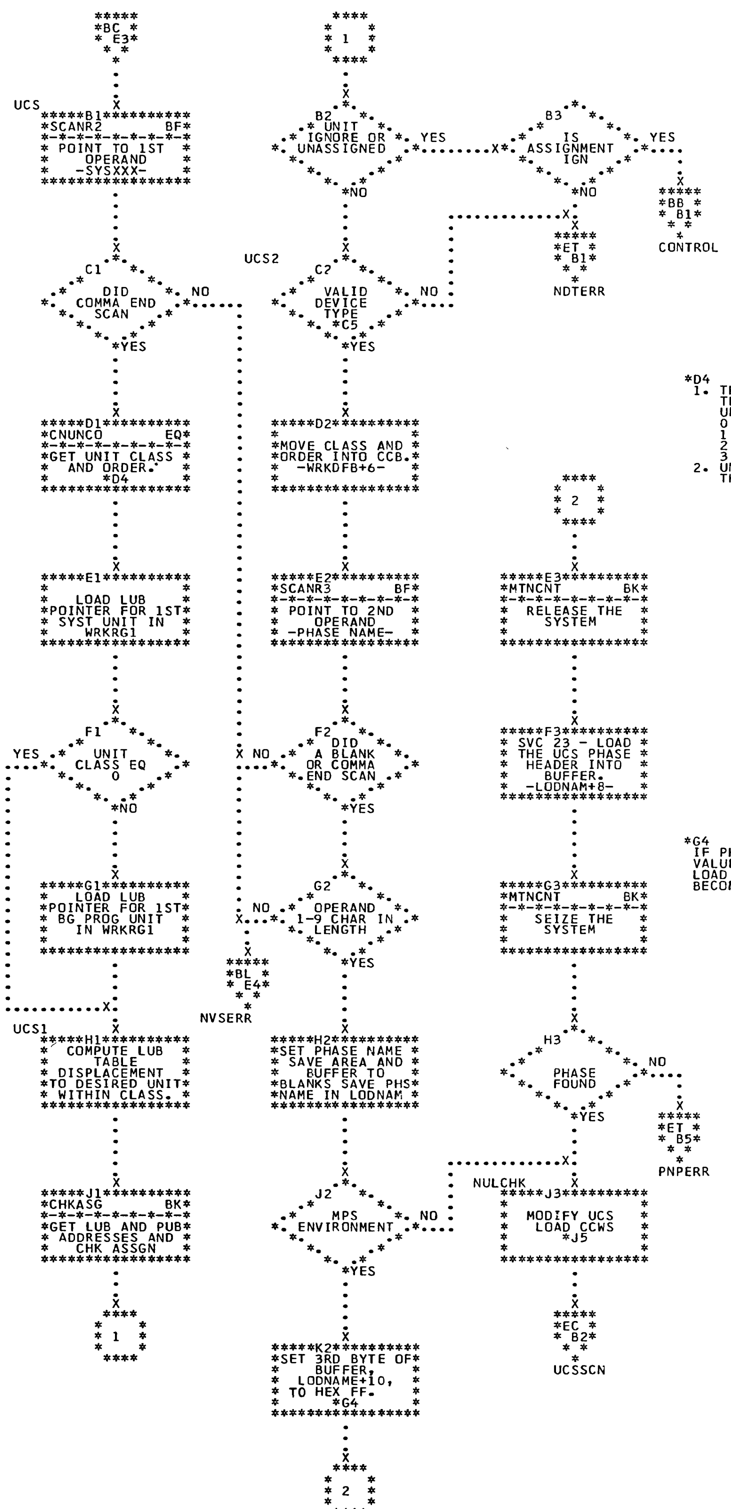


Chart EB. UCS Statement Processor- $JOBCTLJ (Part 1 of 2); Refer to Job Control, Chart 11



*C5
DEVICE MUST BE A 1403
WITH UCS FEATURE.
-DEVICE TYPE CODE HEX 42-

*D4
1. THE SUBROUTINE CNUNCO RETURNS THE UNIT CLASS IN THE LOCATION UNCLOR, BYTE 0.
0 EQUALS SYSTEM
1 EQUALS BG
2 EQUALS FG1
3 EQUALS FG2
2. UNCLOR, BYTE 1, CONTAINS THE UNIT ORDER -PUB POINTER-.

*G4
IF PHASE IS CATALOGED THIS VALUE CHANGES TO THE PHASE LOAD ADDRESS. BIT 7 ALWAYS BECOMES A ZERO

*J5
LODCCW2-SET TO NONFOLD OPCODE -HEX FB-.
LODCCW2+4-SET CHAIN FLAG TO CHAIN WRITE CCW -HEX 60-.
LODCCW3-SET ALLOW DATA CHECK OPCODE -HEX 7B-

Chart EC. UCS Statement Processor- $JOBCTLJ (Part 2 of 2); Refer to Job Control, Chart 11

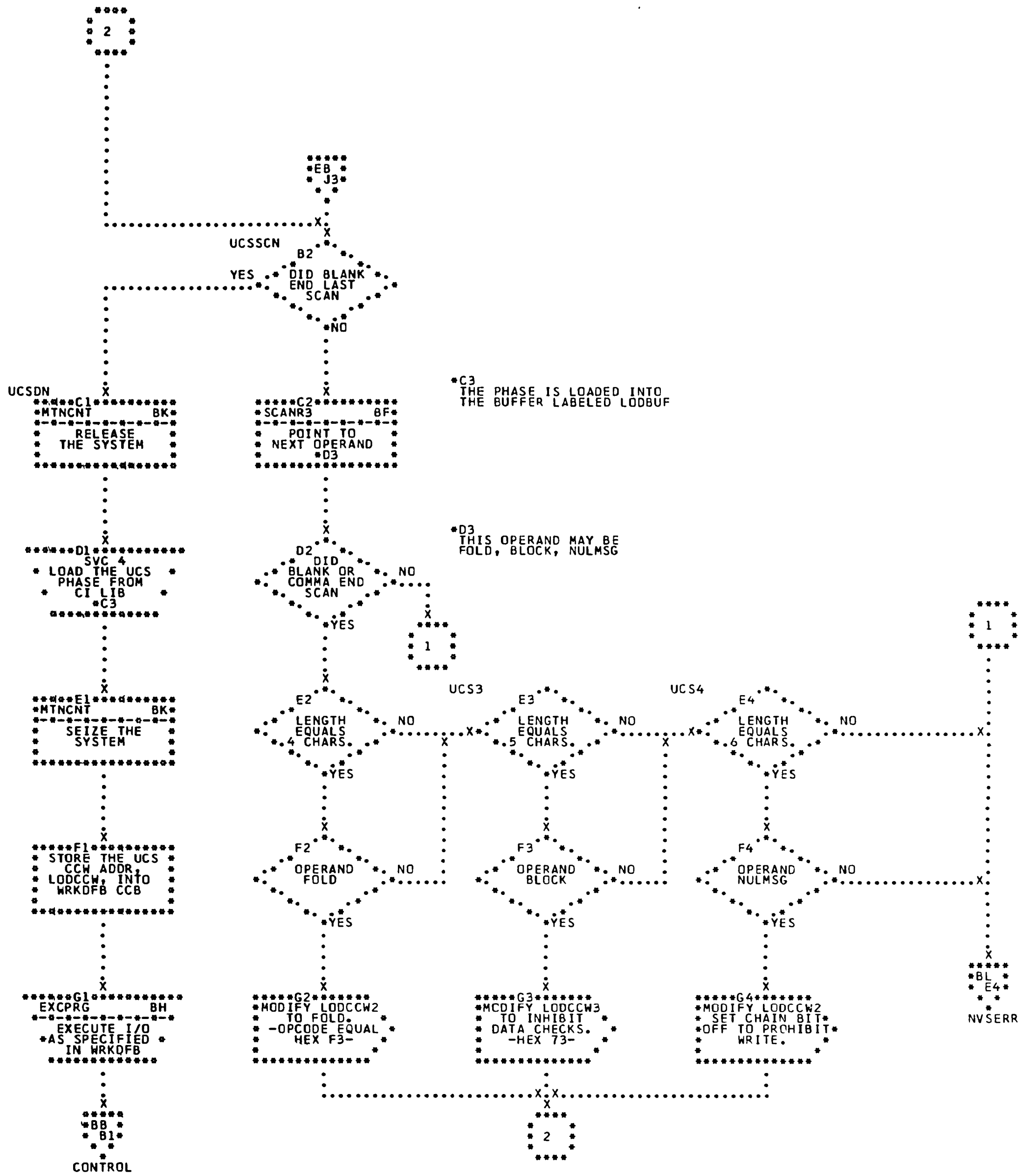


Chart ED. ACTION, and INCLUDE Statement Processors--
$JOBCTLJ; Refer to Job Control, Chart 09

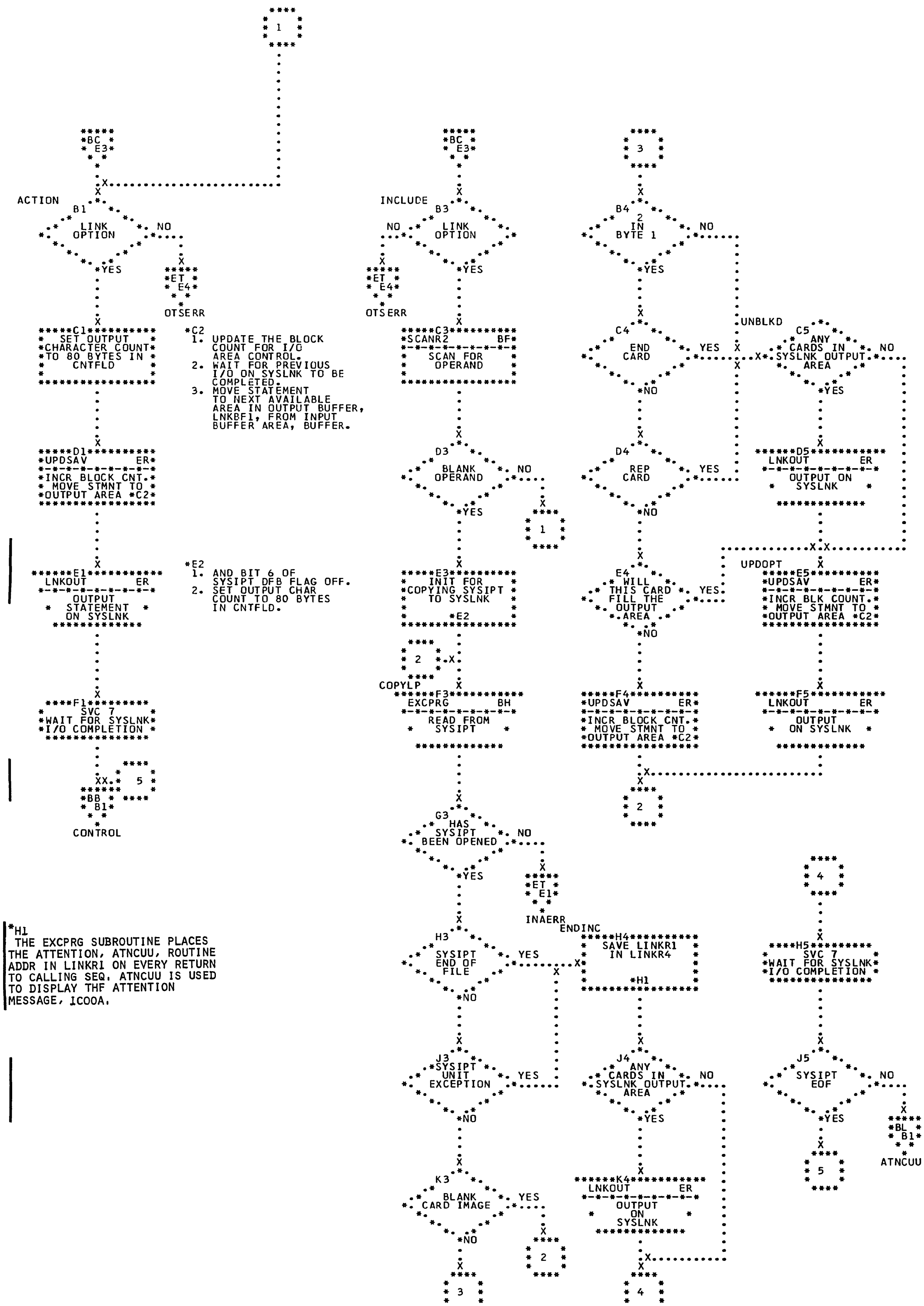


Chart EE. MTC Statement Processor-- $JOBCTLJ; Refer to Job Control, Chart 09

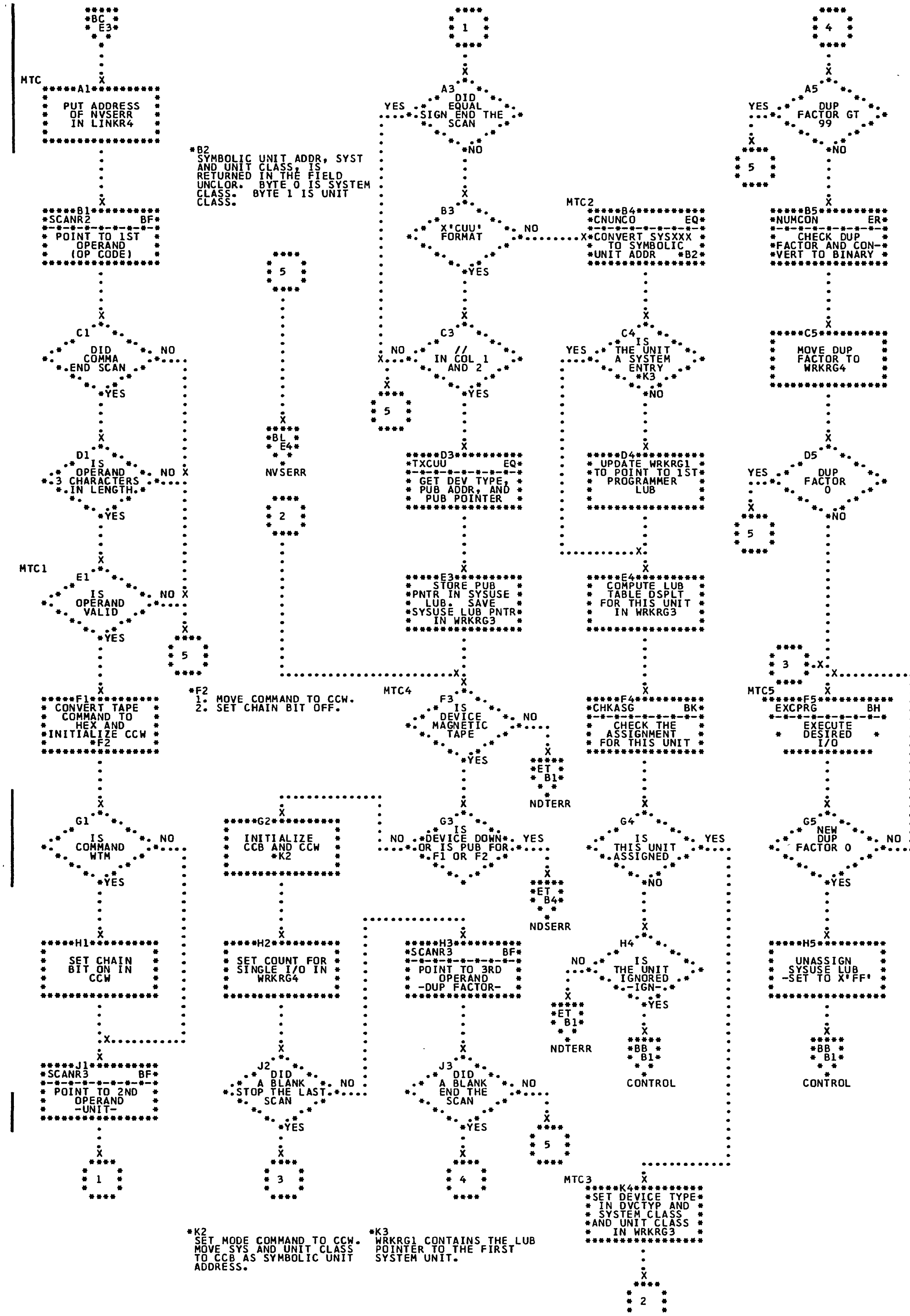


Chart EF. LBLTYP, and VOL Statement Processors-- $JOBCTLJ; Refer to Job Control, Chart 10

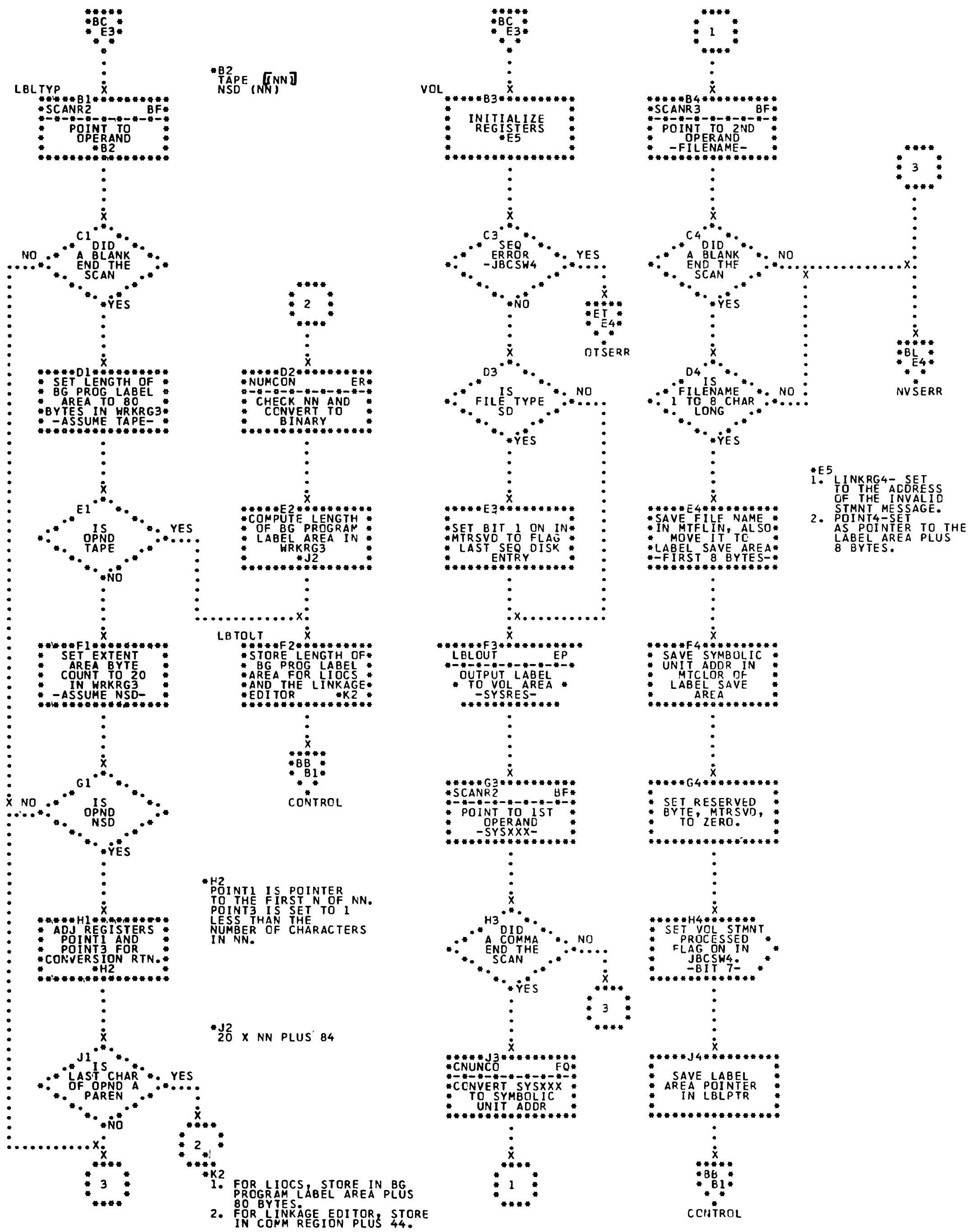


Chart EG. DLAB Statement Processor-- $JOBCTLJ; Refer to Job Control, Chart 10

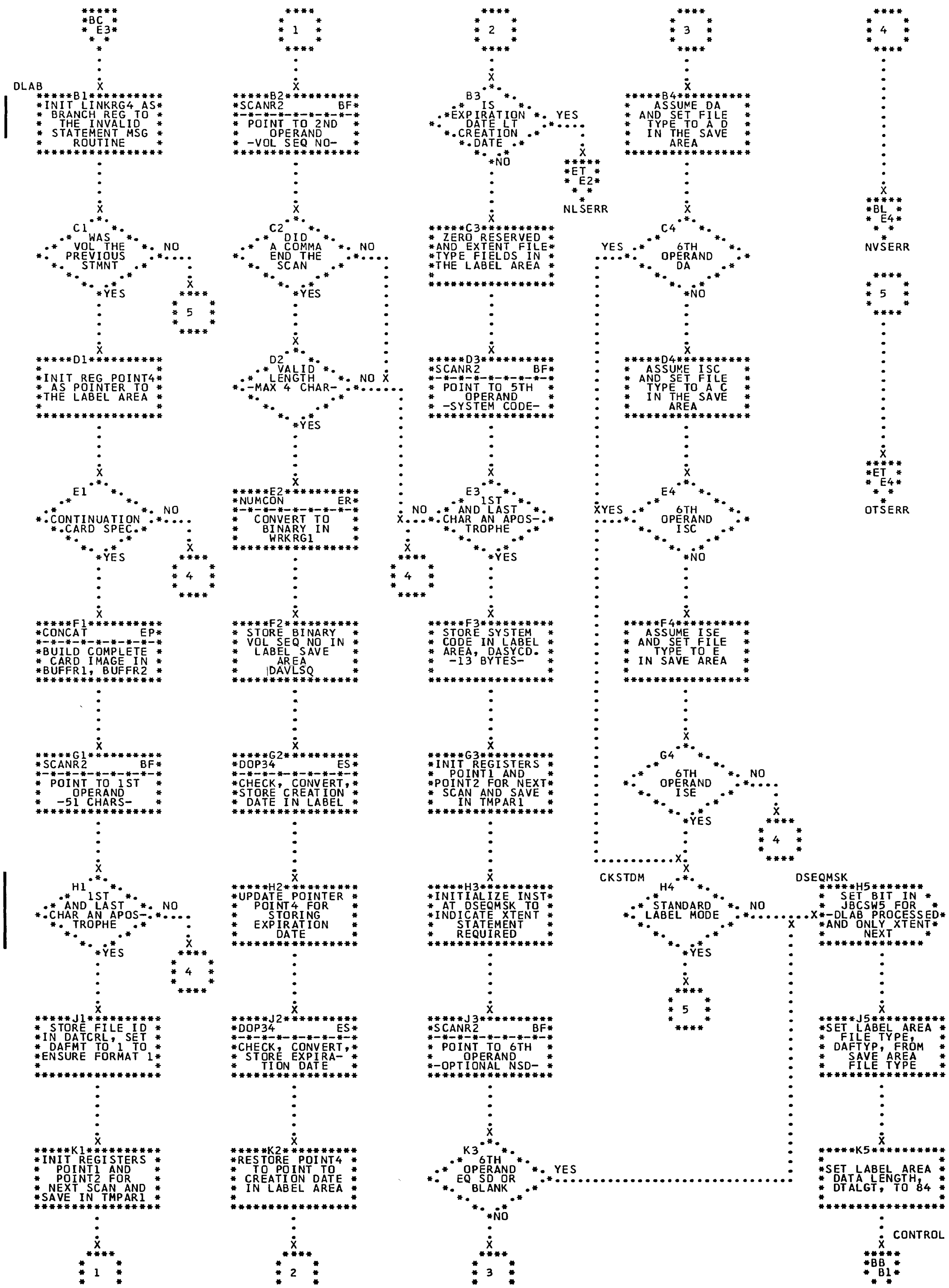


Chart EH. XTENT Statement Processor- $JOBCTLJ (Part 1 of 2); Refer to Job Control, Chart 10

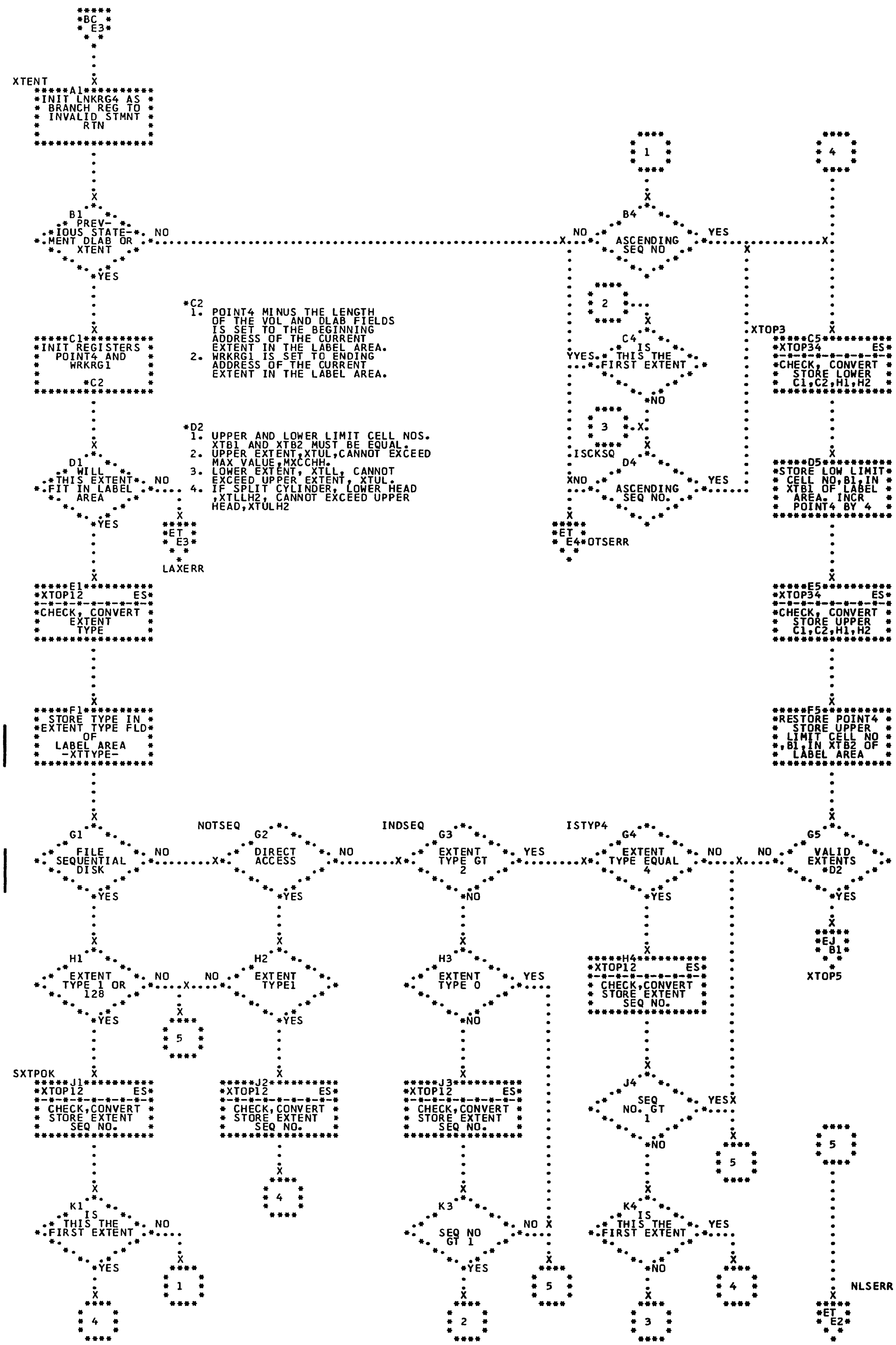


Chart EJ. XTENT Statement Processor- $JOBCTLJ (Part 2 of 2); Refer to Job Control, Chart 10

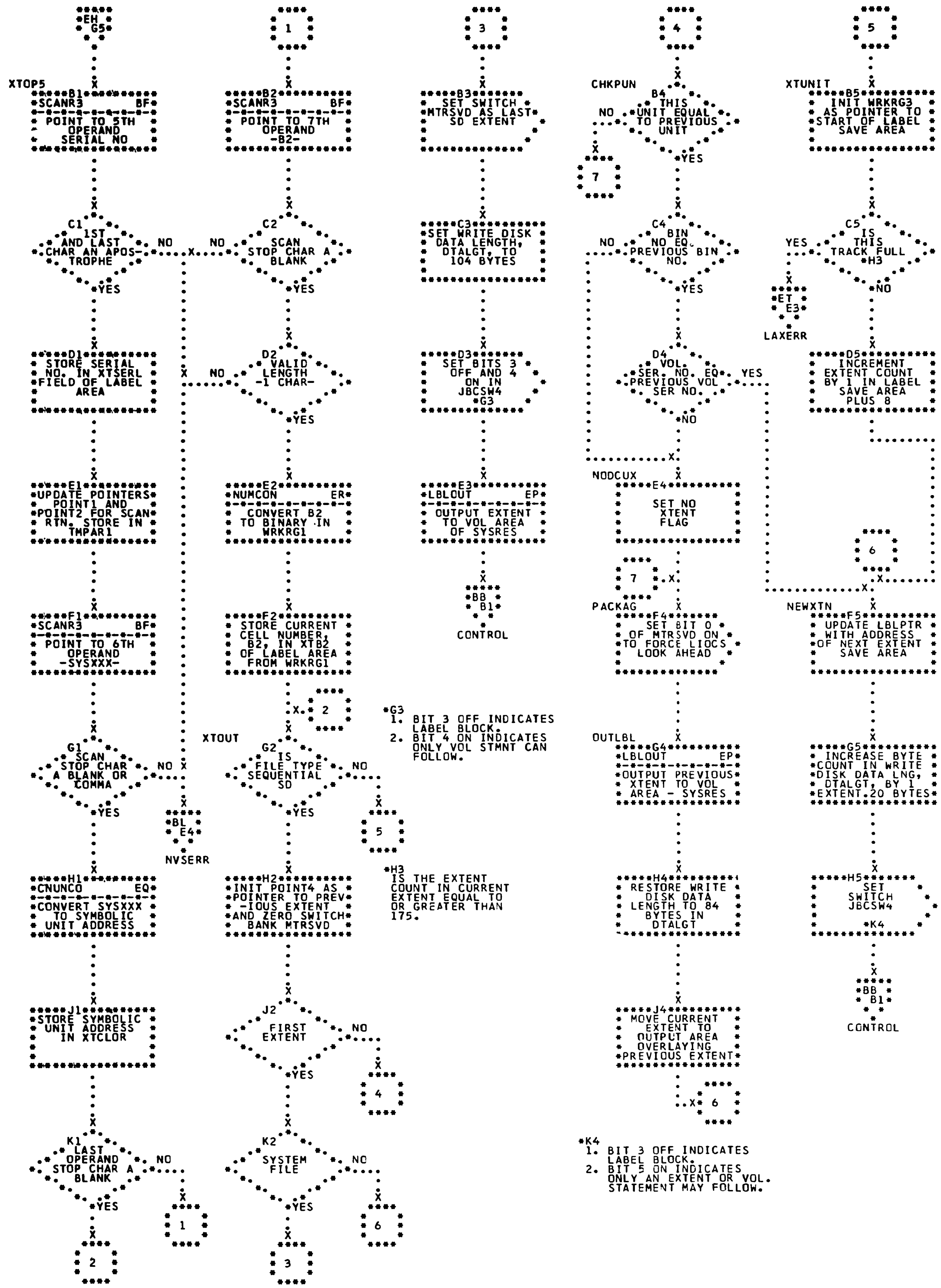


Chart EK. TPLAB, and DATE Statement Processors-- $JOBCTLJ; Refer to Job Control, Charts 09, 10

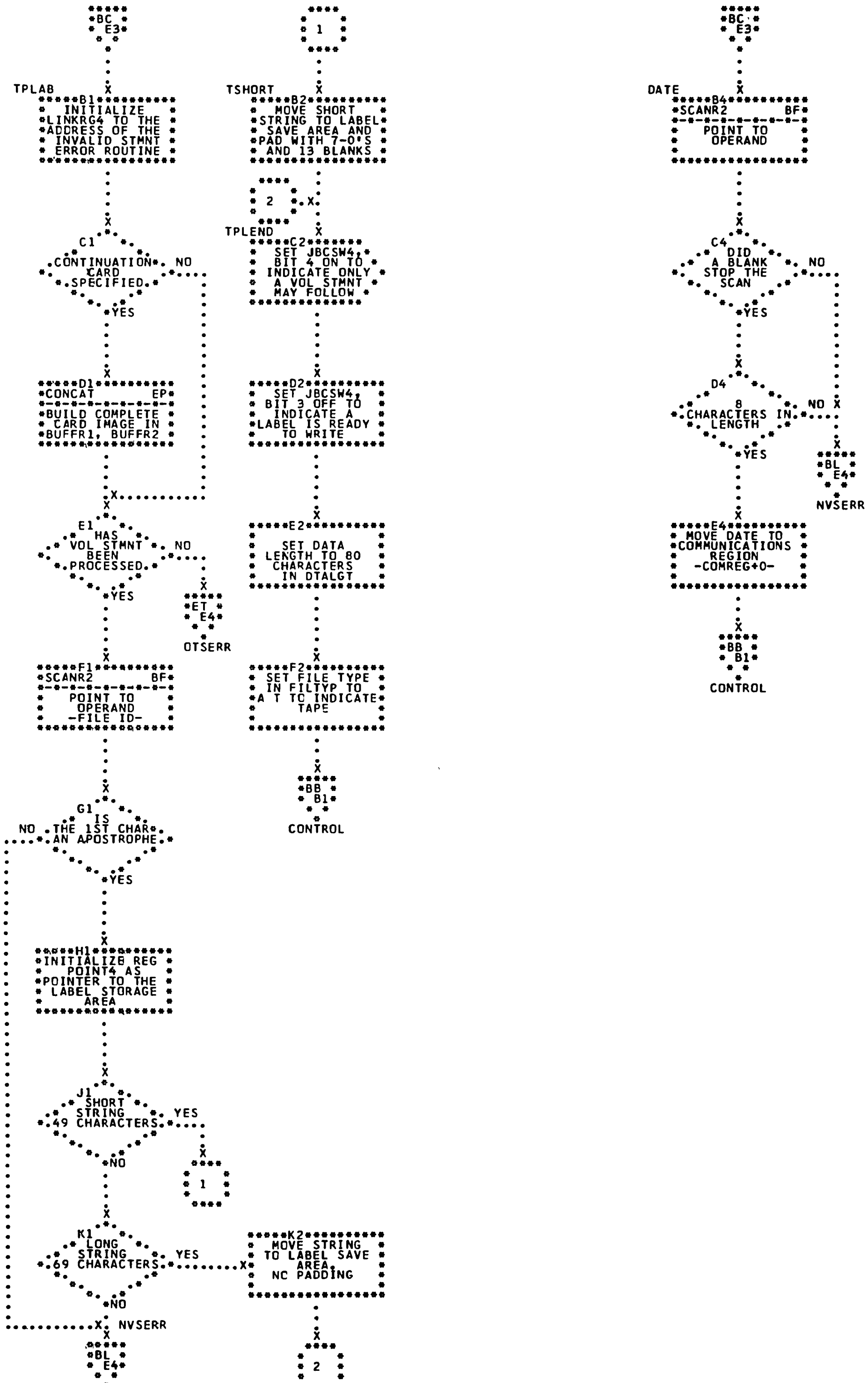


Chart EL. SET Statement Processor-- $JOBCTLJ; Refer to Job Control, Chart 09

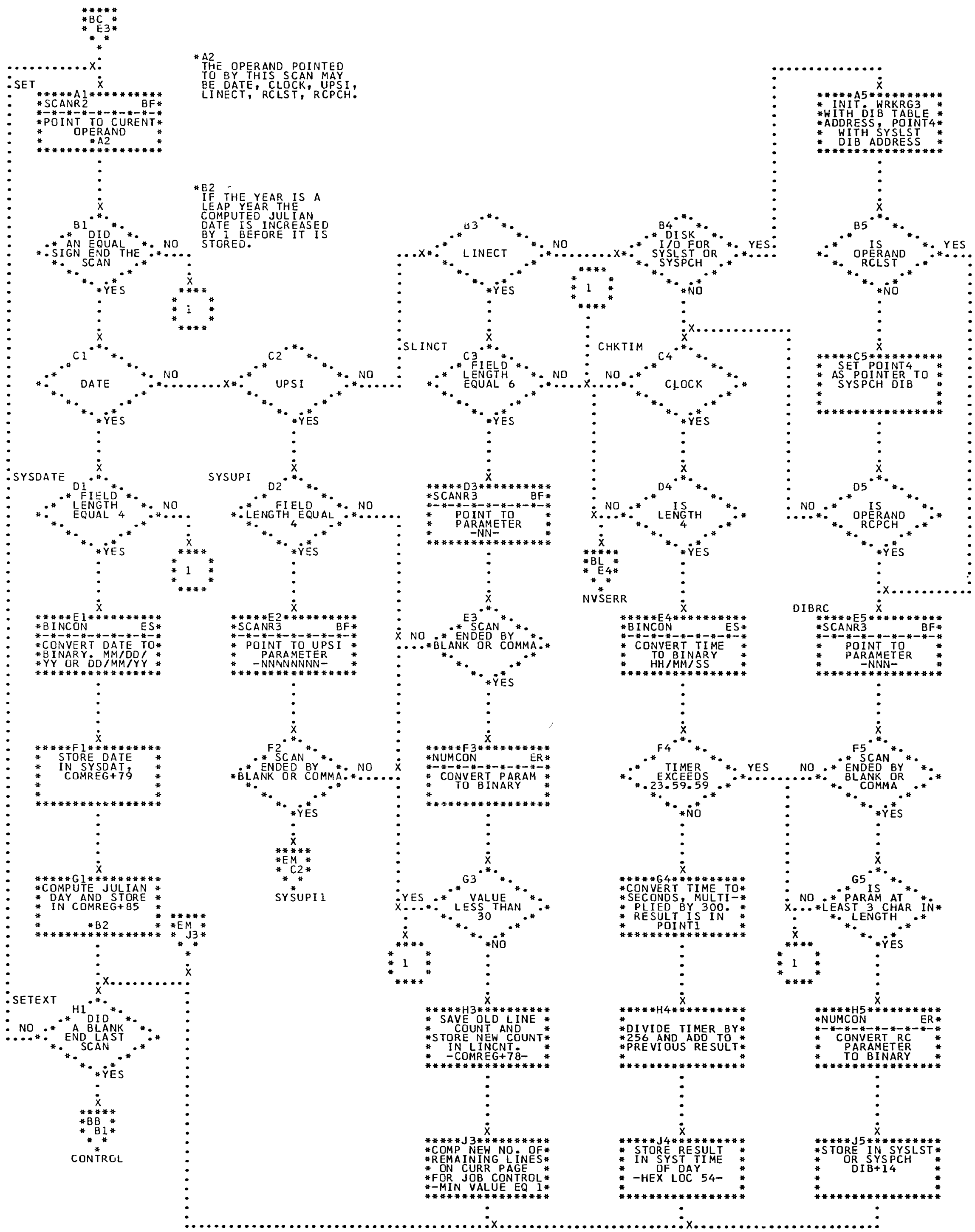


Chart EM. UPSI Statement Processor-- $JOBCTLJ; Refer to Job Control, Chart 09

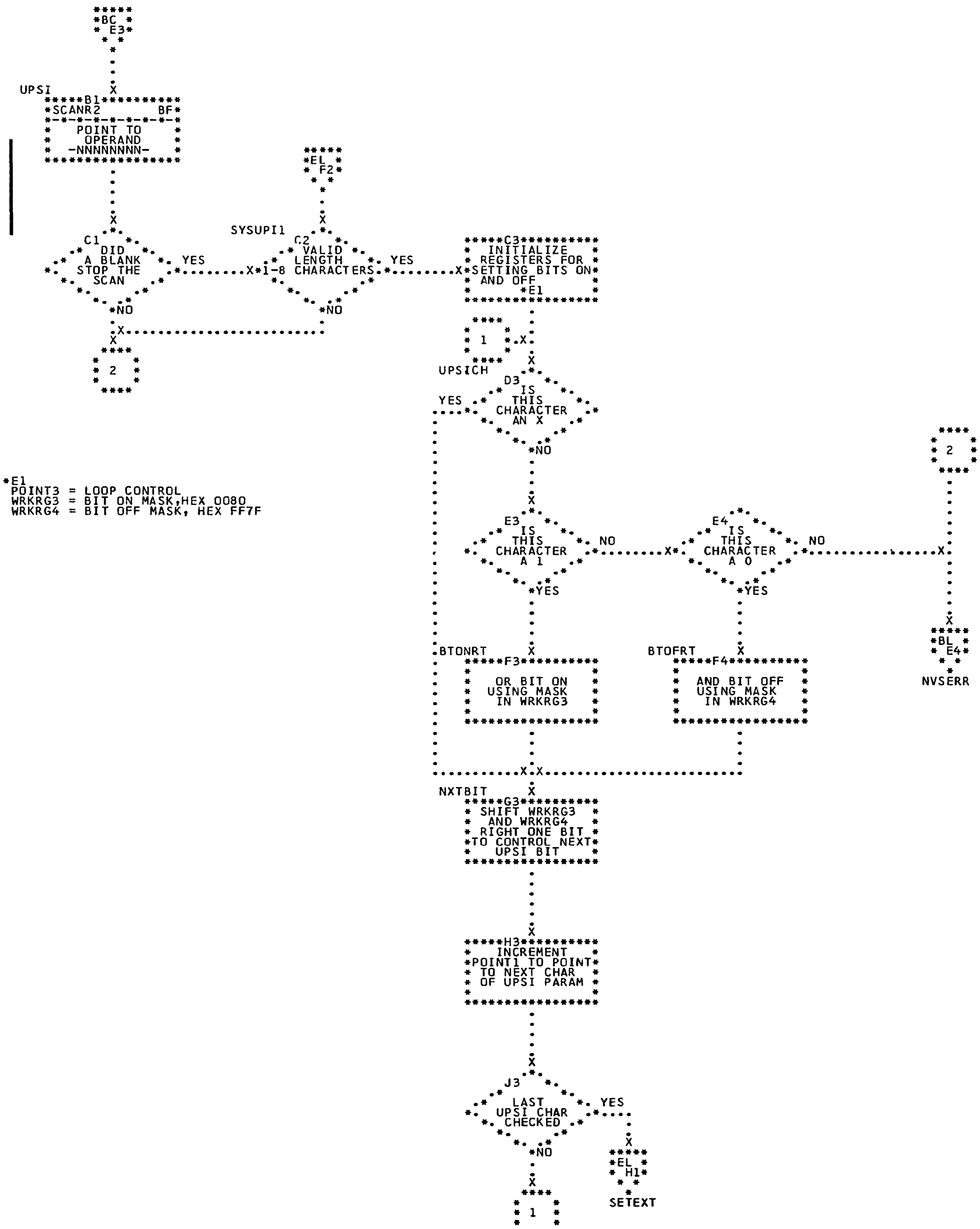


Chart EN. RSTRT Statement Processor-- $JOBCTLJ; Refer to Job Control, Chart 09

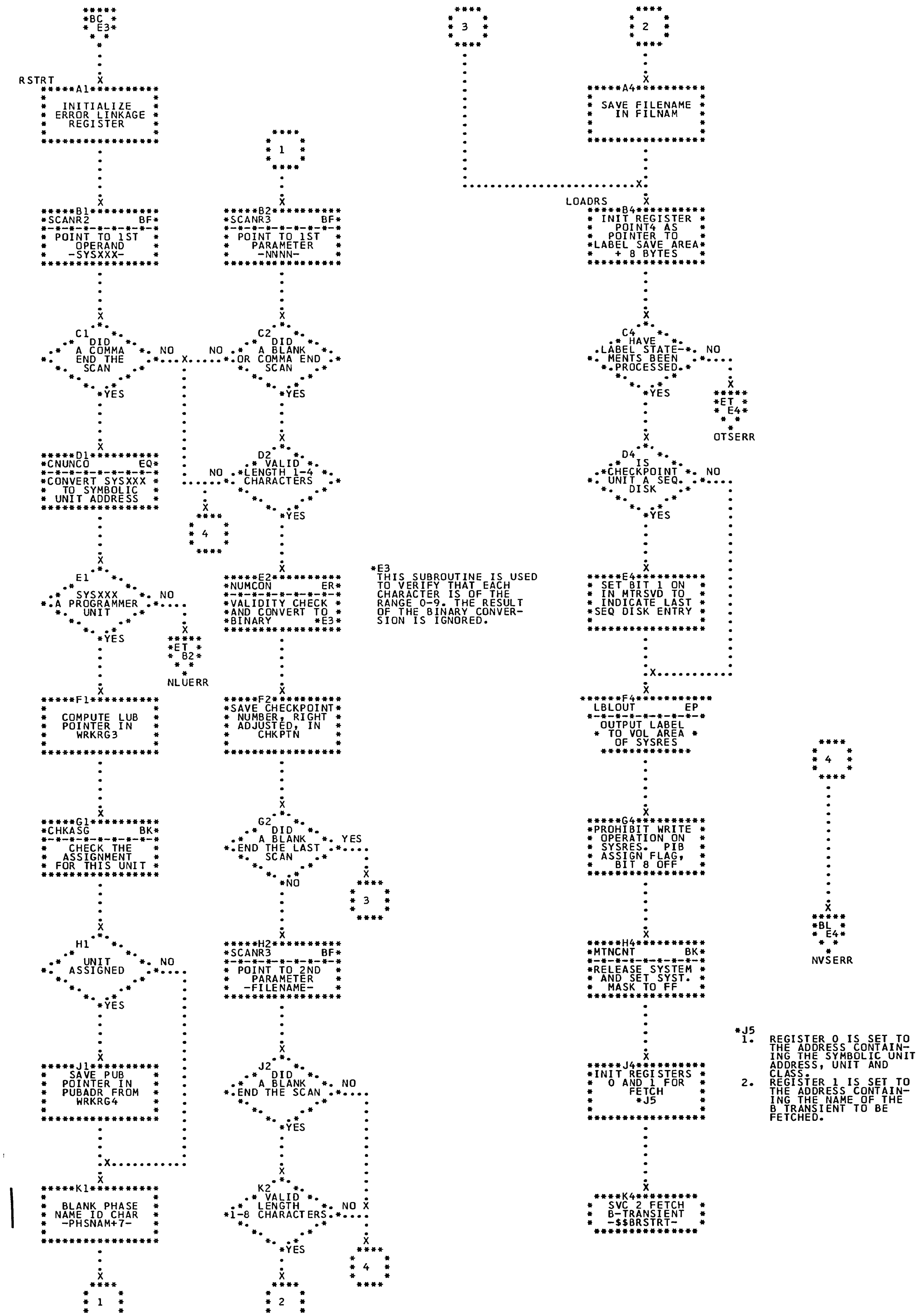


Chart EP. Subroutines-- $JOBCTLJ (LBLOUT, and CONCAT); Refer to Job Control, Charts 09-11

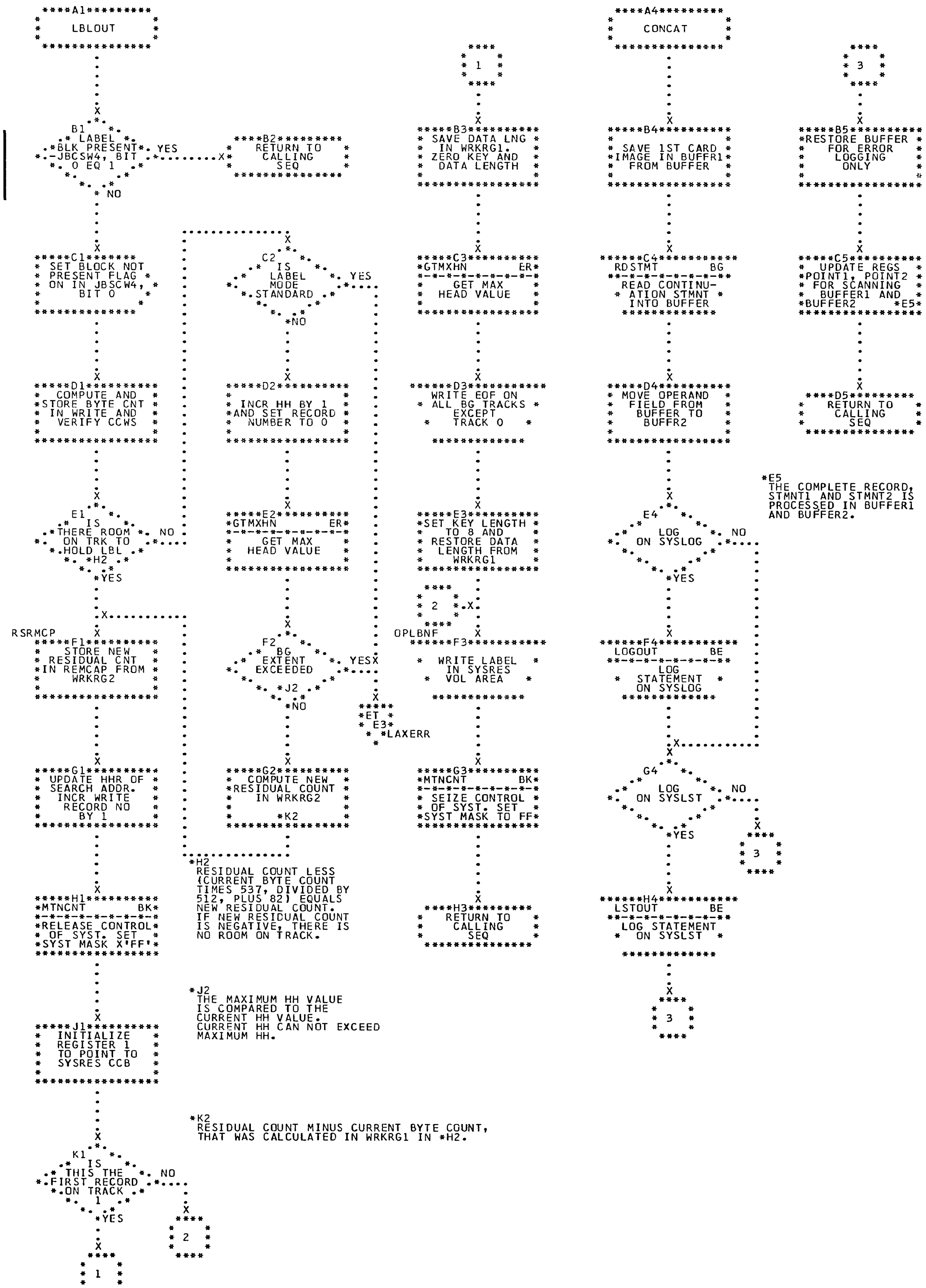


Chart EQ. Subroutines-- $JOBCTLJ (TXCUU, HEXCON, and CNUNCO); Refer to Job Control, Charts 09-11

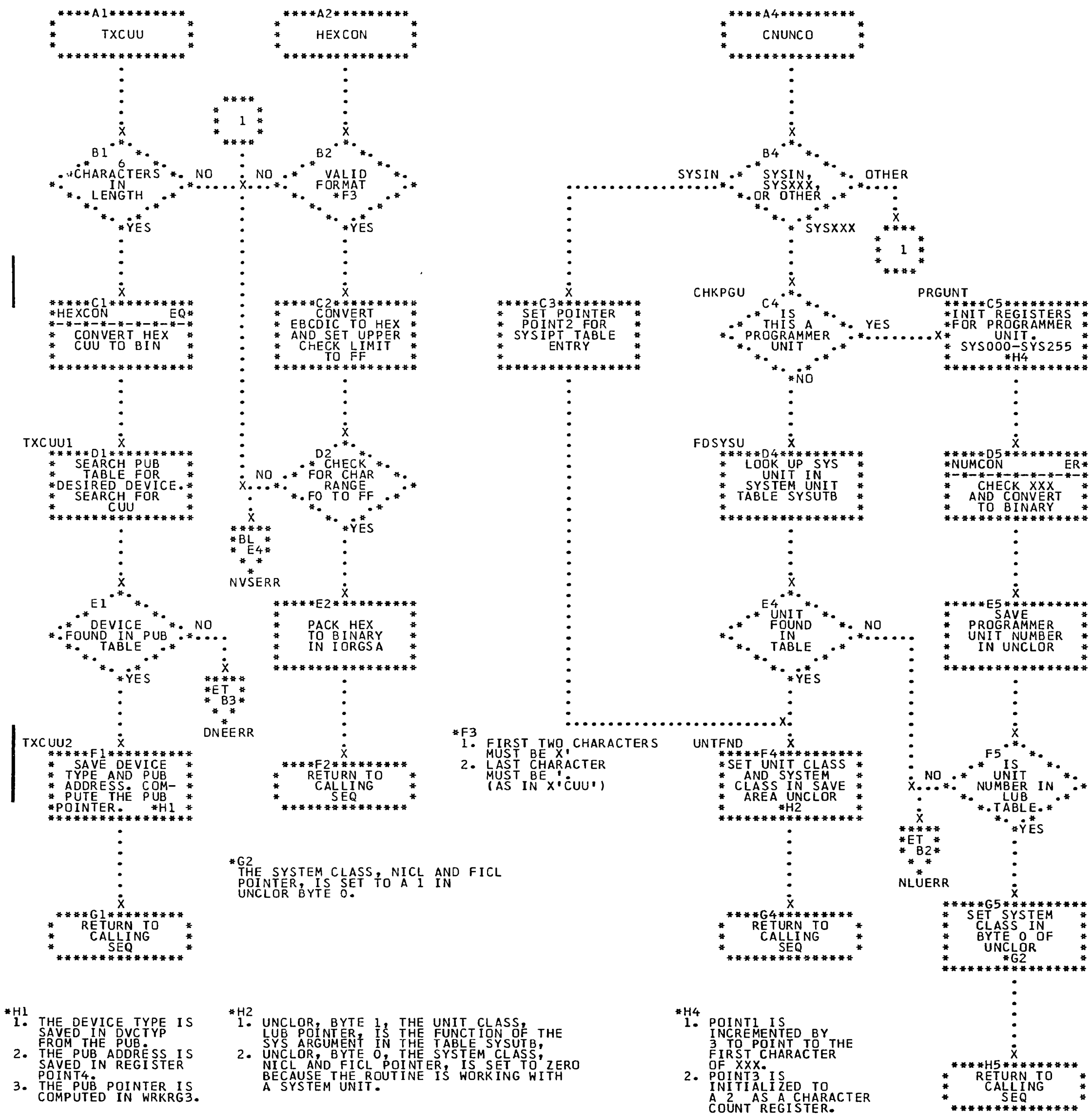


Chart ER. Subroutines-- $JOBCTLJ (UPDSAV, LNKOUT, NUMCON, and GTMXHN); Refer to Job Control, Charts 09-11

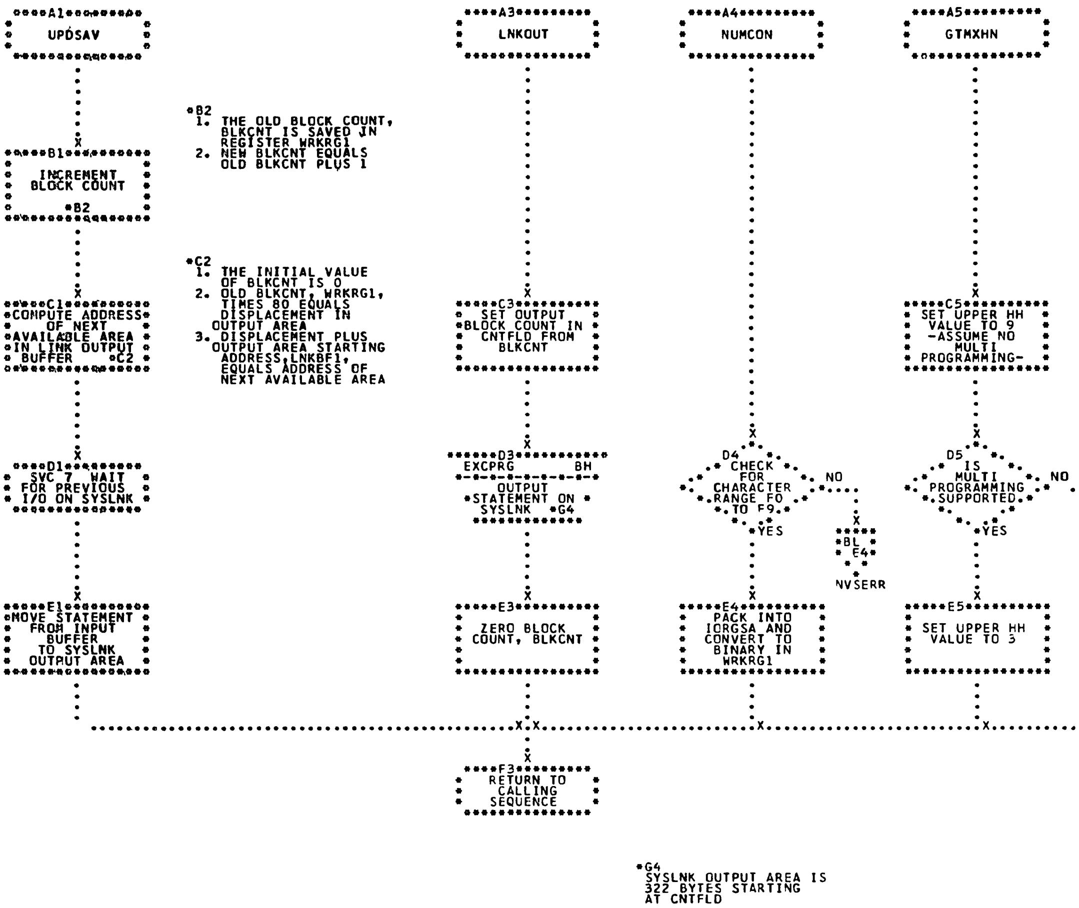
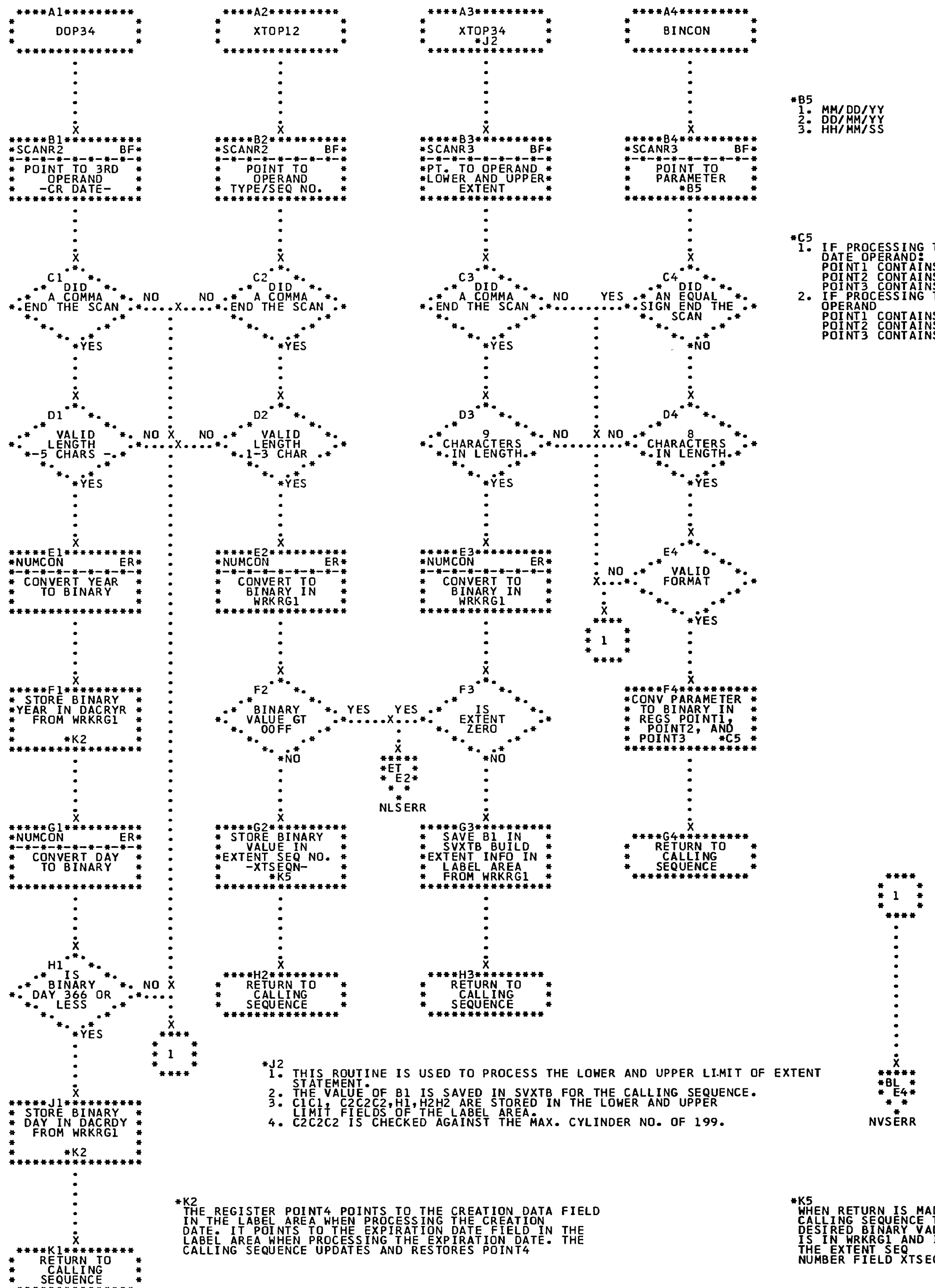


Chart ES. Subroutines-- $JOBCTLJ (DOP34, XTOP12, XTOP34, and BINCON); Refer to Job Control, Charts 09-11




*B5
1. MM/DD/YY
2. DD/MM/YY
3. HH/MM/SS

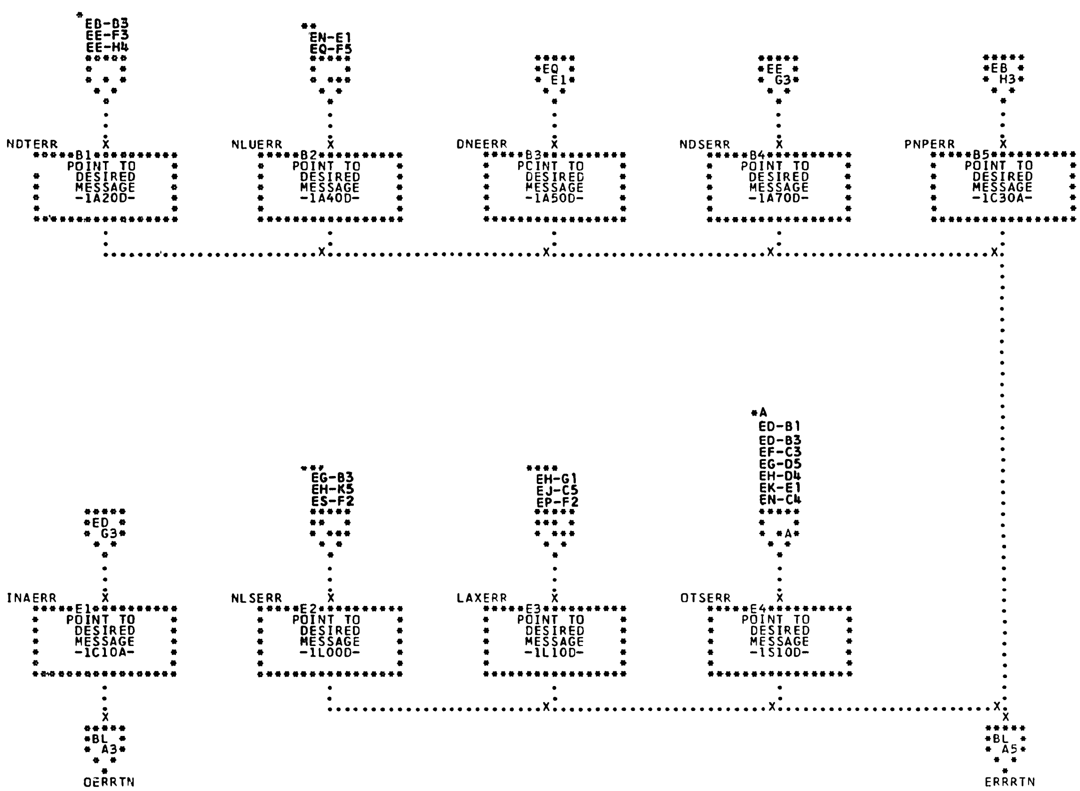
*C5
1. IF PROCESSING THE DATE OPERAND:
   POINT1 CONTAINS MM OR DD
   POINT2 CONTAINS DD OR MM
   POINT3 CONTAINS YY
2. IF PROCESSING THE CLOCK OPERAND
   POINT1 CONTAINS HH
   POINT2 CONTAINS MM
   POINT3 CONTAINS SS

*J2
1. THIS ROUTINE IS USED TO PROCESS THE LOWER AND UPPER LIMIT OF EXTENT STATEMENT.
2. THE VALUE OF B1 IS SAVED IN SVXTB FOR THE CALLING SEQUENCE.
3. C1C1, C2C2C2,H1,H2H2 ARE STORED IN THE LOWER AND UPPER LIMIT FIELDS OF THE LABEL AREA.
4. C2C2C2 IS CHECKED AGAINST THE MAX. CYLINDER NO. OF 199.

*K2
THE REGISTER POINT4 POINTS TO THE CREATION DATA FIELD IN THE LABEL AREA WHEN PROCESSING THE CREATION DATE. IT POINTS TO THE EXPIRATION DATE FIELD IN THE LABEL AREA WHEN PROCESSING THE EXPIRATION DATE. THE CALLING SEQUENCE UPDATES AND RESTORES POINT4

*K5
WHEN RETURN IS MADE TO THE CALLING SEQUENCE THE DESIRED BINARY VALUE IS IN WRKRG1 AND IN THE EXTENT SEQ NUMBER FIELD XTSEQN.

Chart ET. Error Routines-- $JOBCTLJ (NDTERR, NLUERR, DNEERR, NDSERR, INAERR, NLSERR, LAXERR, and OTSERR); Refer to Job Control, Charts 09-11



## Chart EV. Initialize and Return to Fetching Routine-- $$BLSTIO; Refer to Job Control, Chart 05

```
A3
$$BLSTIO

B3
INITIALIZE
TRANSIENT SW
BANKT TO OFF
STATUS
-HEX 00-

C3
GET
COMMUNICATION
REGION ADDR
FROM LOC
HEX 16

D3
GET PIB TABLE
ADDR FROM
COMMUNICATION
REGION PLUS 90

E3
GET ADDR OF
LOGICAL TRANS
BUCKET, L.T.B.,
FROM PIB TABLE
PLUS 72        *E4

*E4
L.T.B. CONTAINS THE ADDRESS OF
SAVE AREA WHERE REGISTERS
HAVE BEEN SAVED BY THE
SUPERVISOR.

F3
RESTORE REG
9-14 FROM SAVE
AREA PLUS 16
*F4

*F4
REGISTER 15 IS THE BASE REGISTER
FOR THIS TRANSIENT. IT IS NOT
RESTORED BY THIS ROUTINE.

G3
RESTORE
REGISTERS
0-8
FROM SAVE
PLUS 44

H4

*H4
RETURN TO THE ROUTINE THAT FETCHED
$$BLSTIO BY BRANCHING ON REG LINKR2.
LINKR2 WAS LOADED WITH THE RETURN
ADDRESS BY THE FETCHING
ROUTINE. IT WAS SAVED BY
THE SUPERVISOR. IT WAS
RESTORED BY THIS ROUTINE.
```

Chart EW. Build Printline in Workarea Subroutine-- $$BLSTIO (PUIF); Refer to Job Control, Chart 05

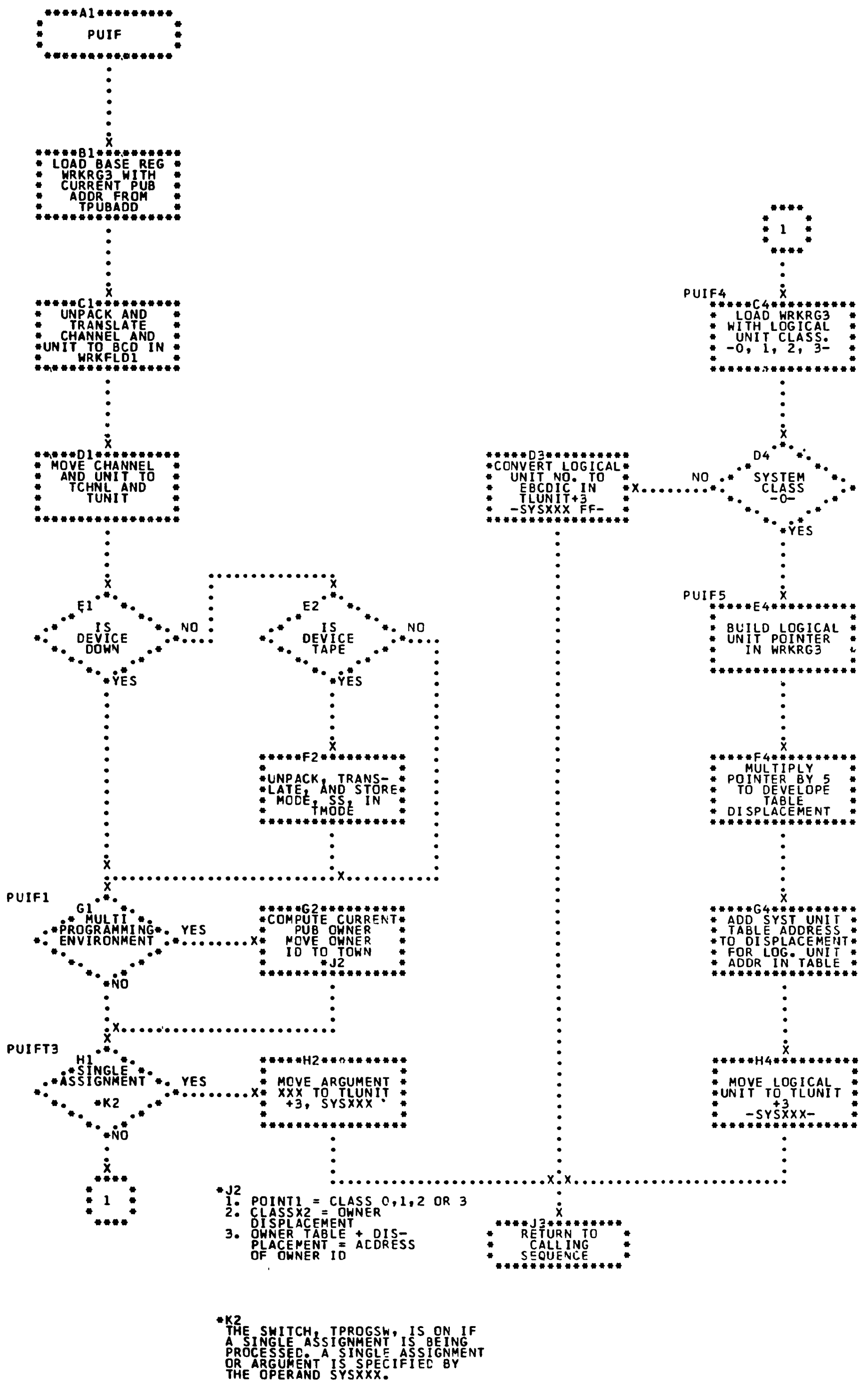


Chart EX. Identify the LISTIO Operand Subroutine-- $$BLSTIO (FNDARG); Refer to Job Control, Chart 05

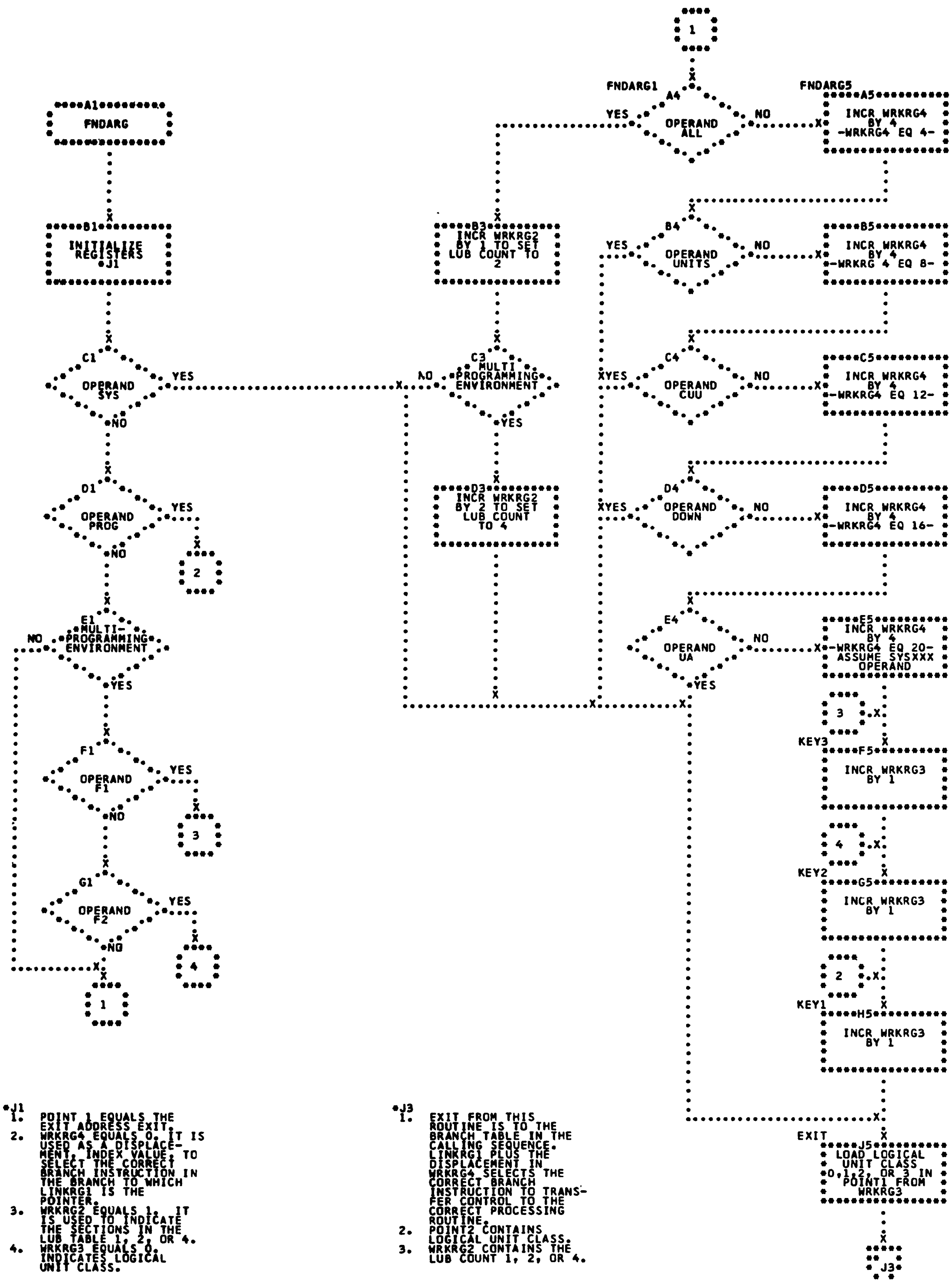
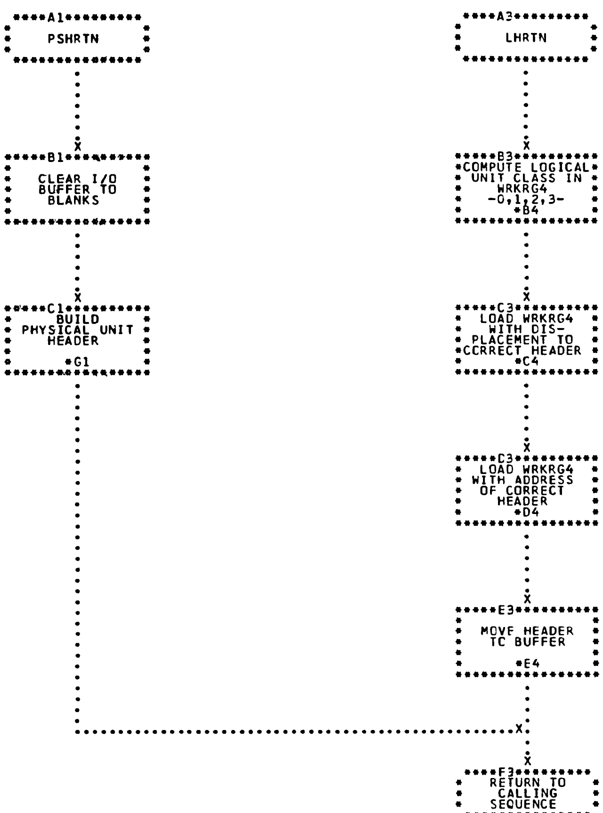


Chart EY. Build Header Subroutines-- $$BLSTIO (PSHRTN, and LHRTN); Refer to Job Control, Chart 05




*B4
SYS EQUAL 0
BG EQUAL 1
F2 EQUAL 2
F1 EQUAL 3

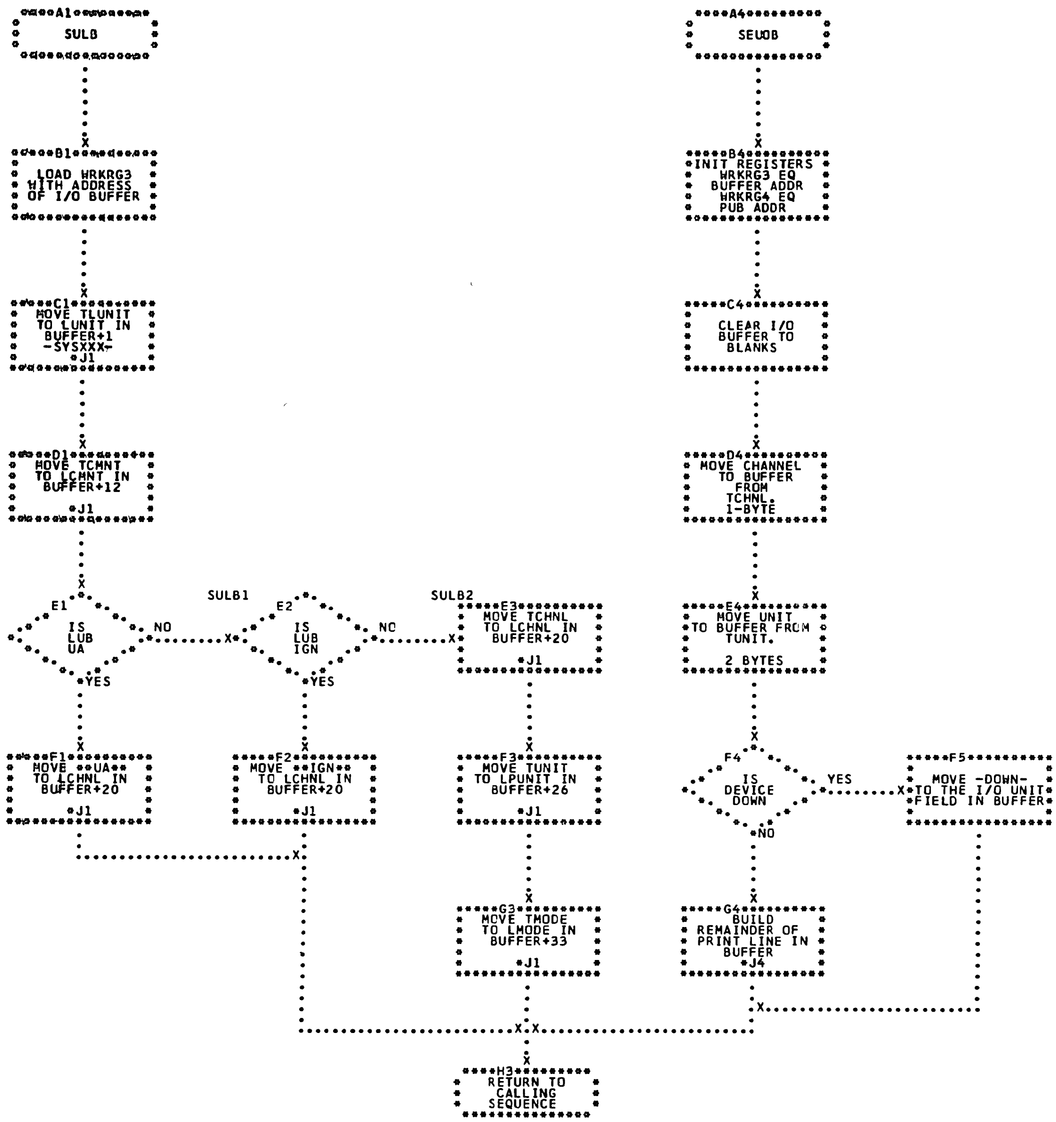
*C4
SYS EQUAL HEX 70
BG EQUAL HEX 84
F2 EQUAL HEX 98
F1 EQUAL HEX A9

*D4
HEADER ADDRESS EQUALS 0 PLUS DISPLACEMENT, *C5 PLUS THE CONTENTS OF REGISTER 15. REG. 15 IS THE BASE REGISTER OF THE $$BLSTIO TRANSIENT.

*E4
1 SYS HEADER***-SYSTEM-***
2 BG, HEADER***-PROGRAM-***
3 F2, HEADER***-FOREGROUND2-***
4 F1, HEADER***-FOREGROUND1-***

*G1
CHNL---UNIT----OWNER--I/O UNIT----------MODE

Chart EZ. Build Print Line Subroutines-- $$BLSTIO (SULB, and SEUOB); Refer to Job Control, Chart 05




*J1
```
LUNIT       LCMNT     LCHNL      LPUNIT LMODE
-SYSXXX-----XXX-------**-UA-**     XX     XX
                      **-IGN-**
```

*J4
1. MOVE OWNER TO OWNER FIELD
2. MOVE MODE FROM TMODE TO MODE FIELD
3. MOVE LOGICAL UNIT FROM TLUNIT TO THE I/O UNIT FIELD
4. MOVE THE COMMENT FROM TCMNT TO THE COMMENT FIELD-MODE LESS 6 BYTES

Chart FA. SUPVR Macro-- General Entry; Refer to Supervisor, Chart 12

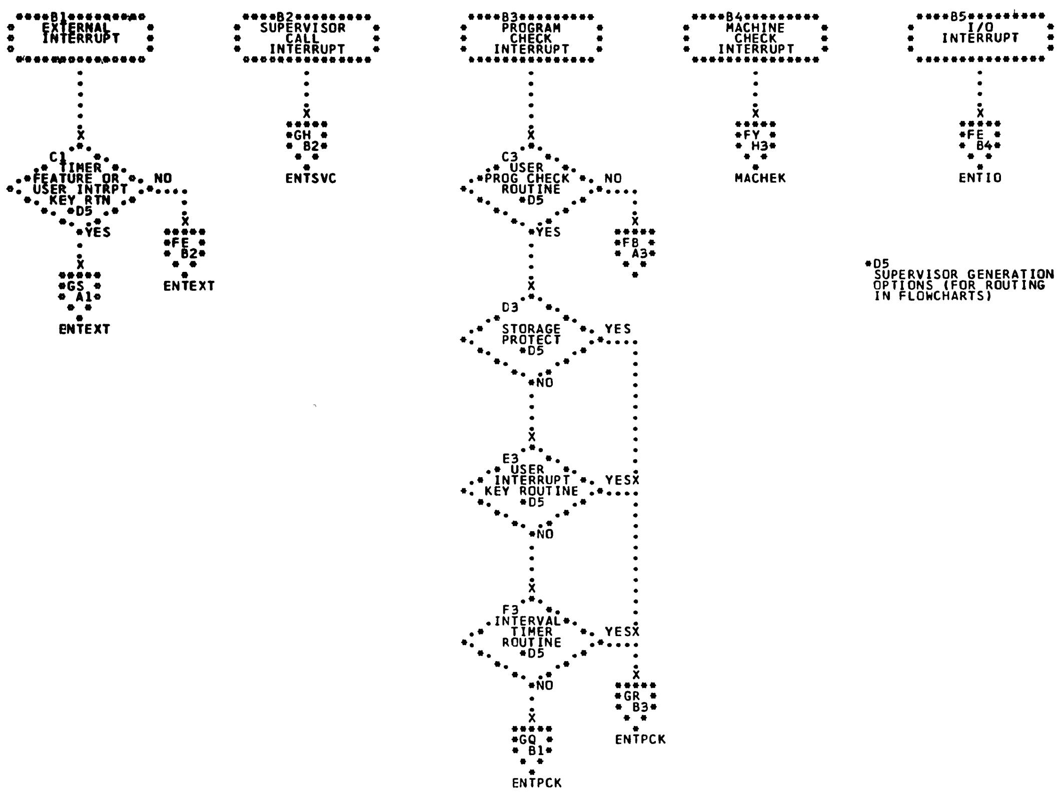


Chart FB. FOPT Macro-- General Cancels and Program Check without User PC Routine; Refer to Supervisor, Charts 14 and 16

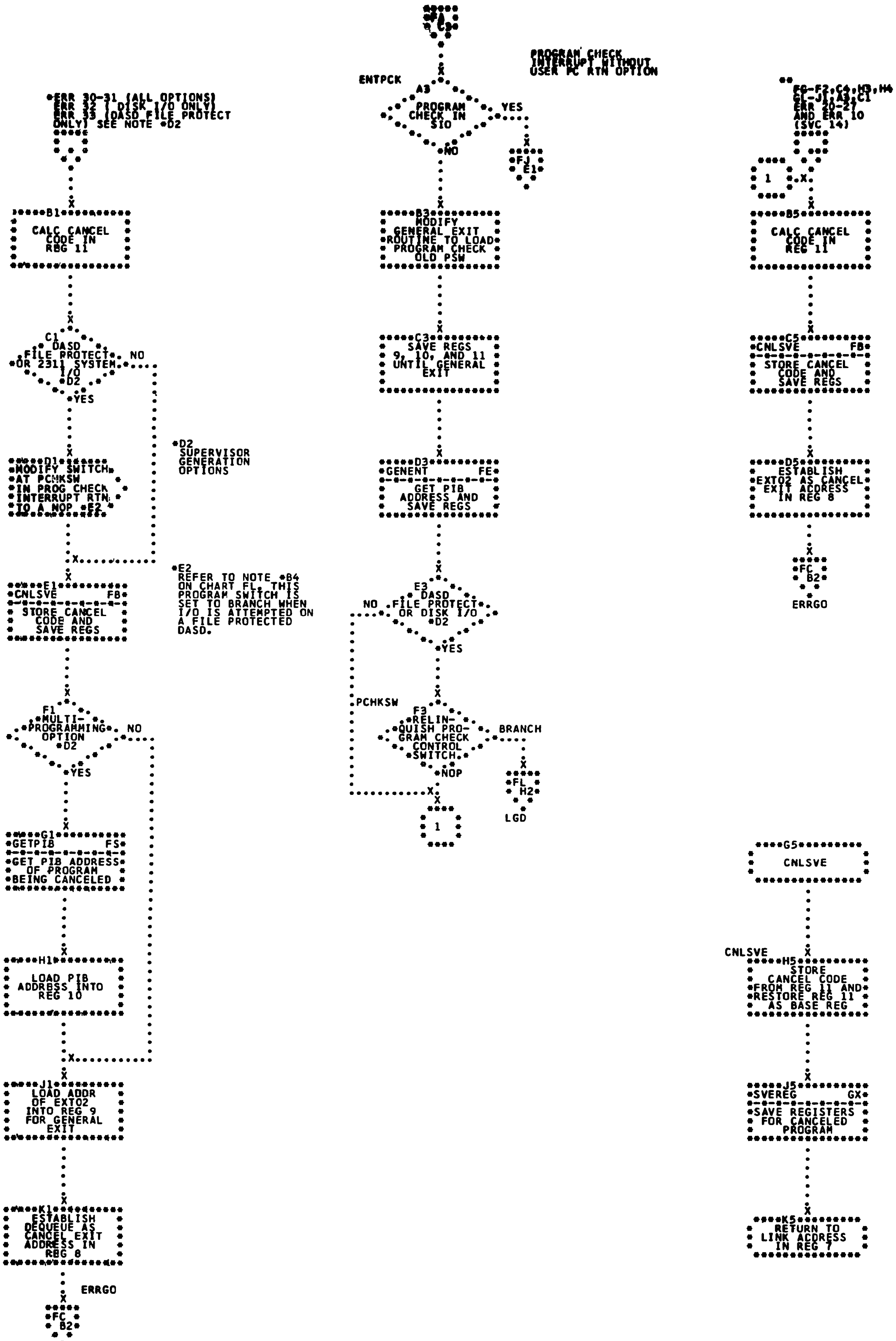


Chart FC. FOPT Macro--General Cancel Subroutine; Refer to Supervisor, Chart 14

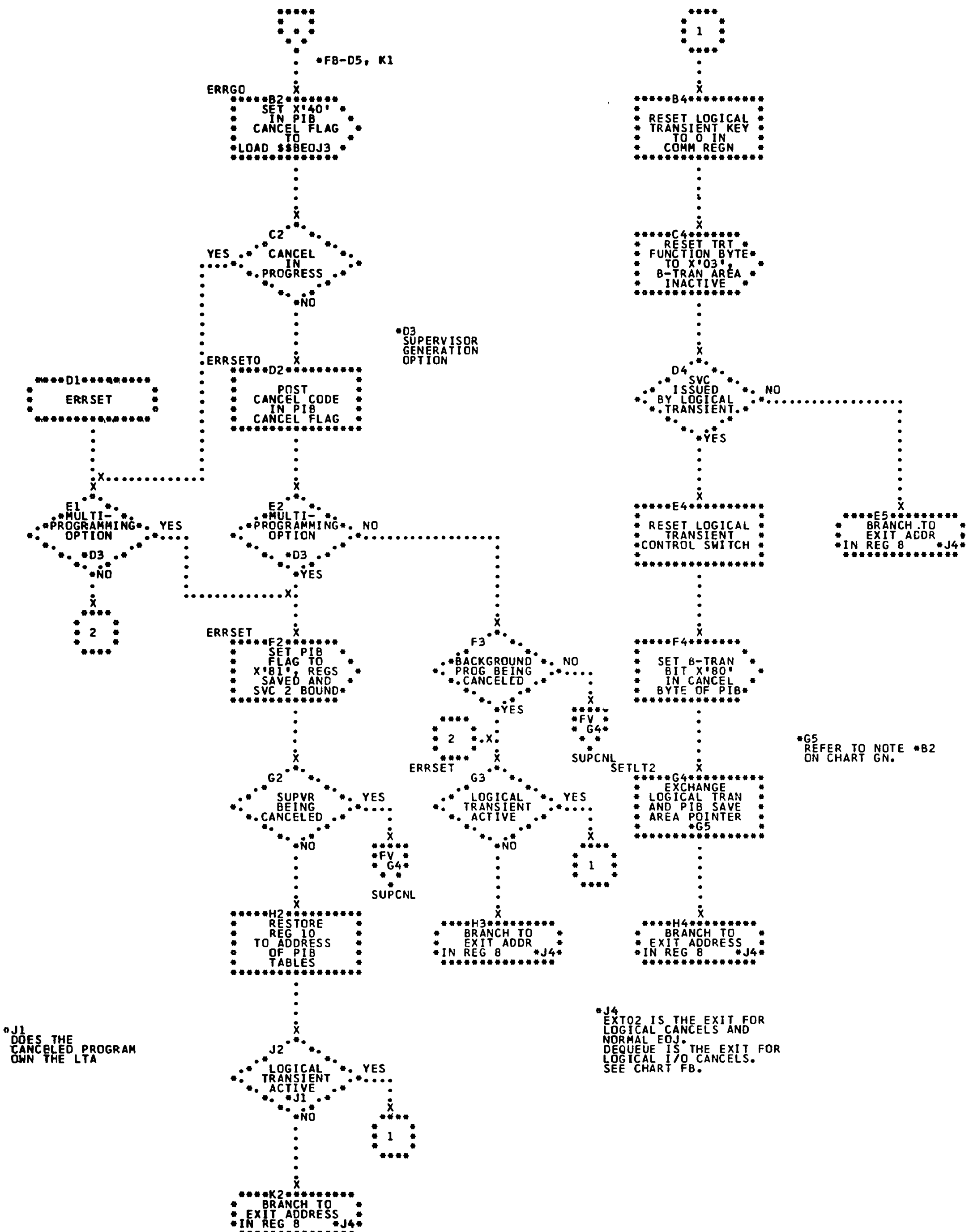


Chart FD. FOPT Macro-- General Exits; Refer to Supervisor, Chart 12

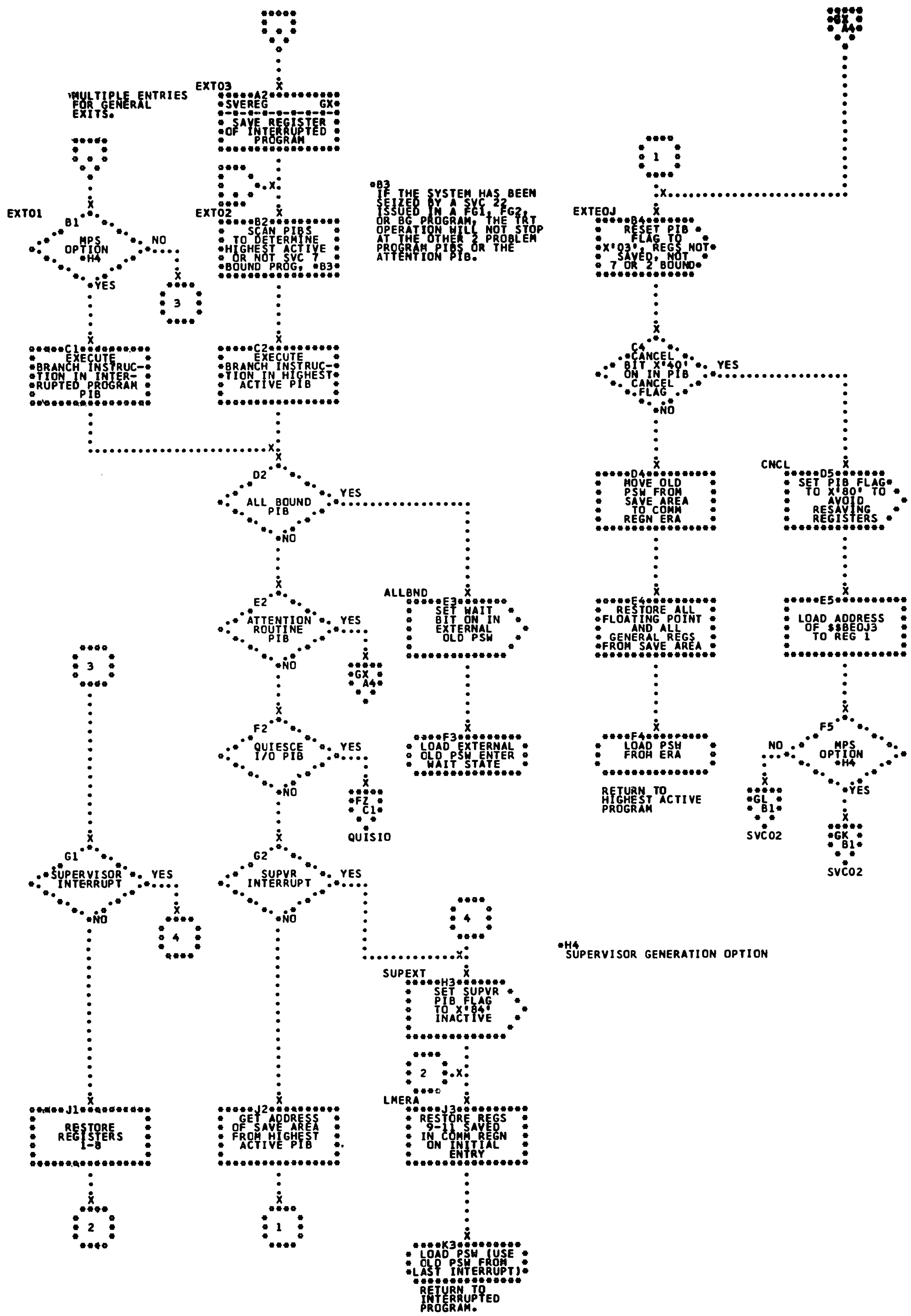


Chart FE. FOPT Macro-- General Entry; Refer to Supervisor, Chart 16

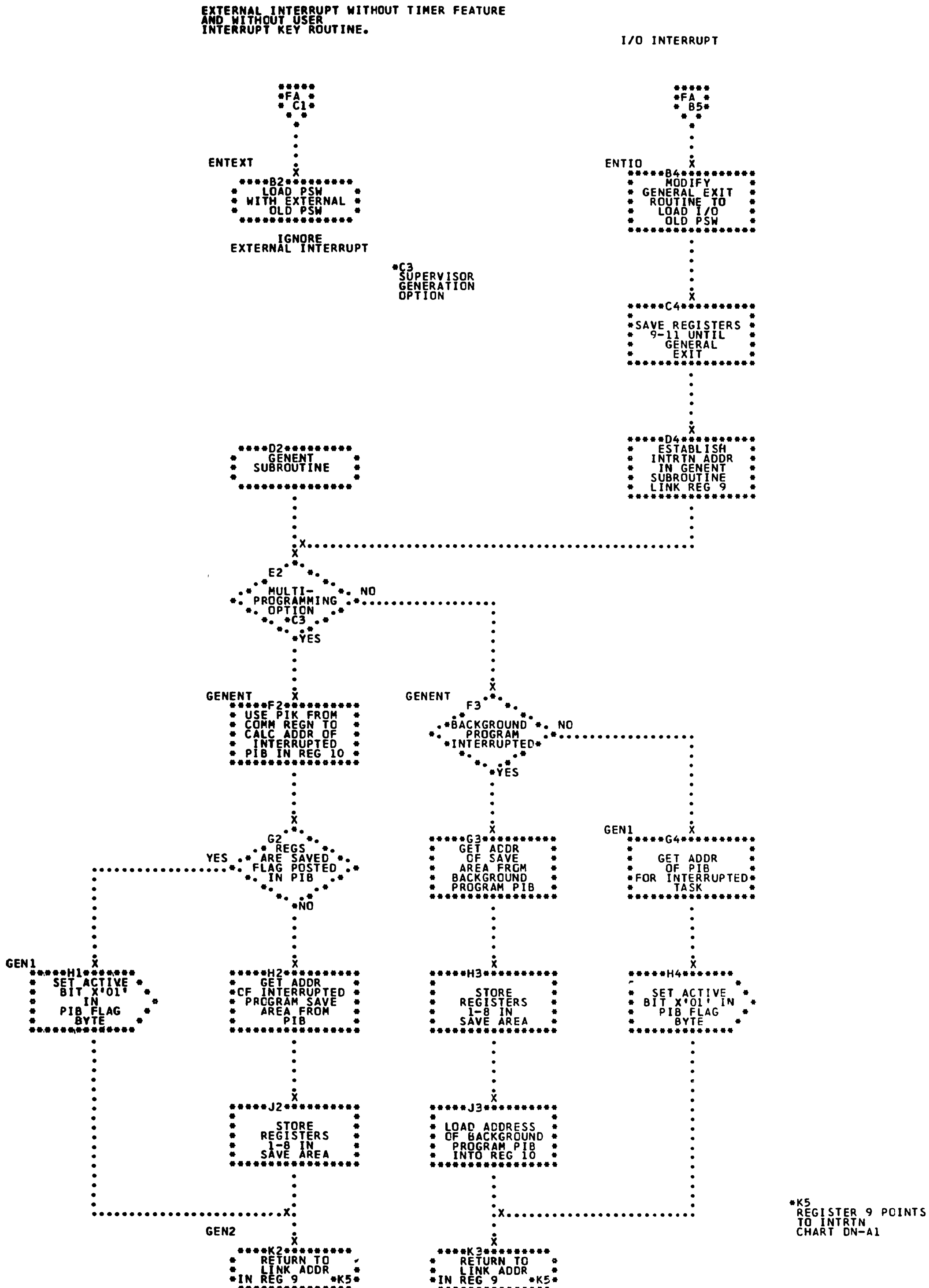


Chart FF. SGTCHS Channel Scheduler (Part 1 of 3); Refer to Supervisor, Chart 15

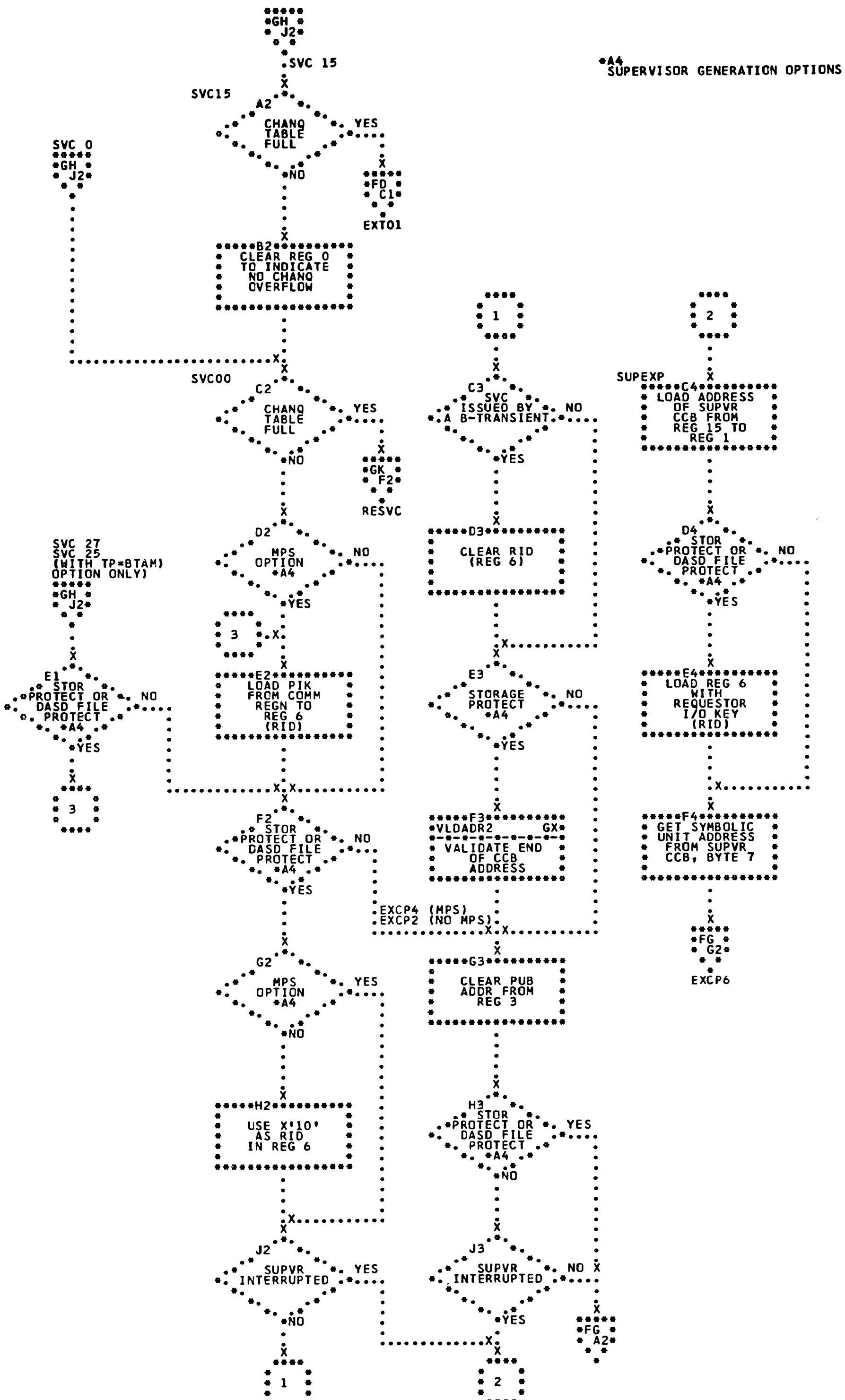


Chart FG. SGTCHS Channel Scheduler (Part 2 of 3); Refer to Supervisor, Chart 15

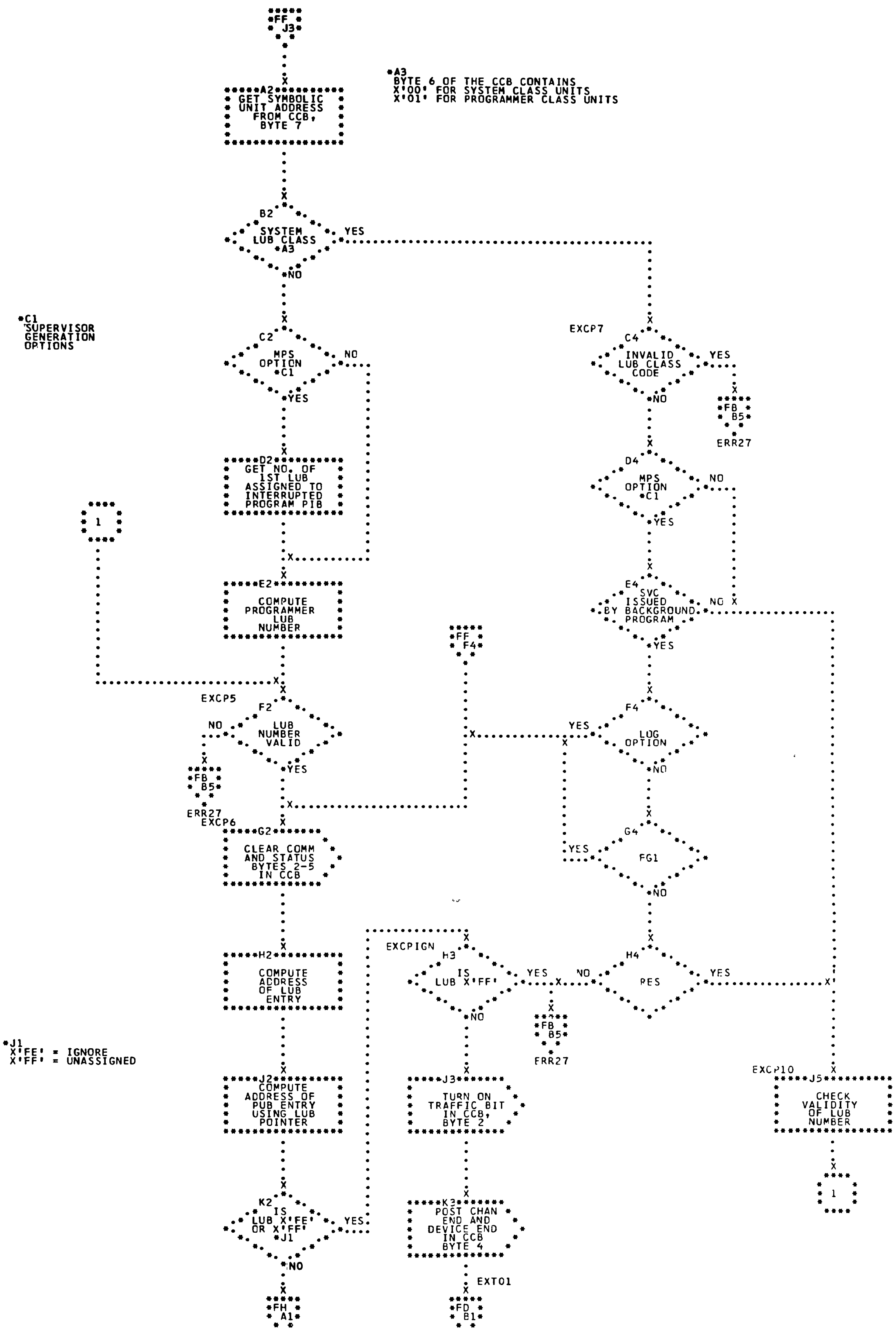


Chart FH. SGTCHS Channel Scheduler (Part 3 of 3); Refer to Supervisor, Chart 15

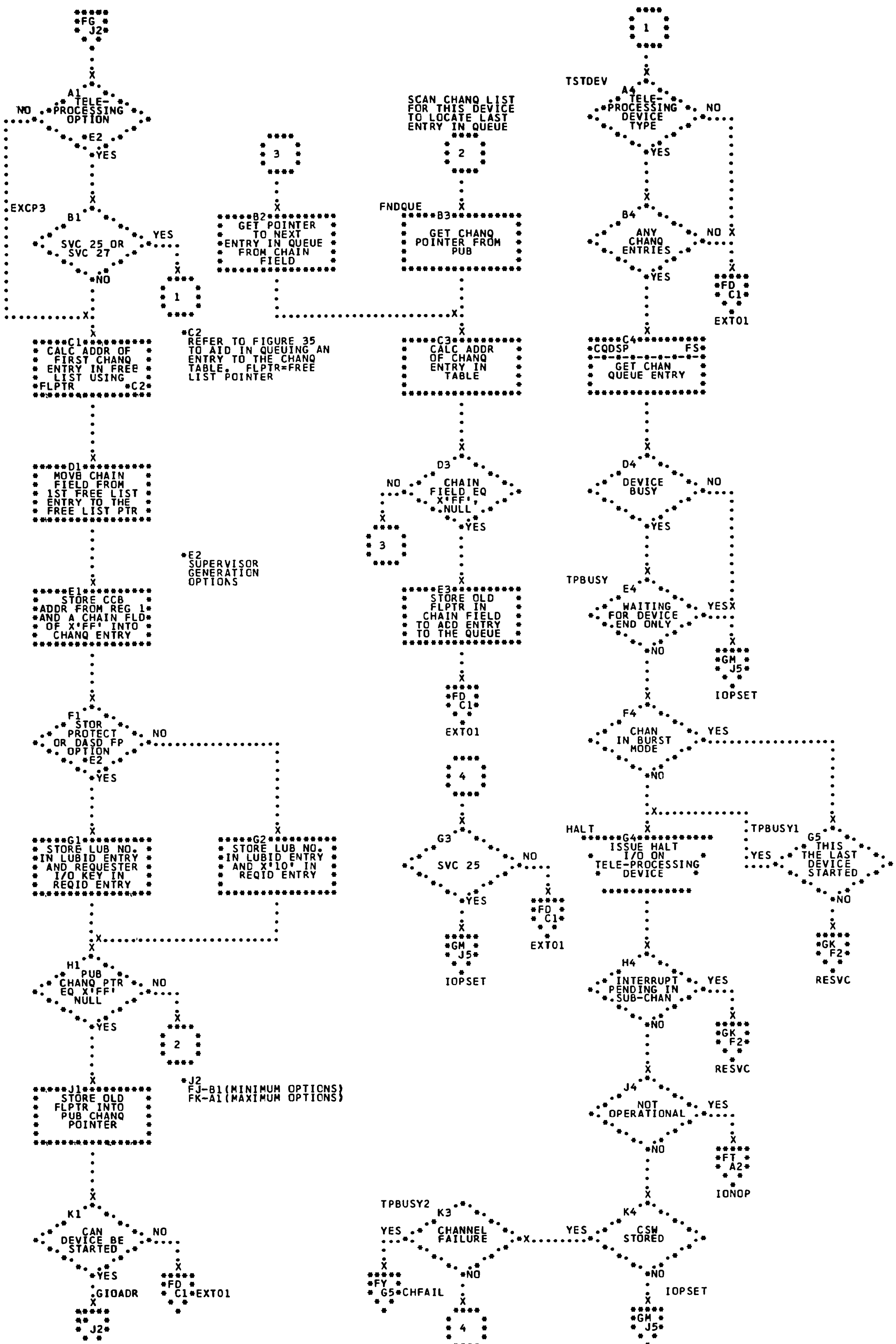


Chart FJ. SGTCHS Start I/O-- No Options; Refer to Supervisor, Chart 15

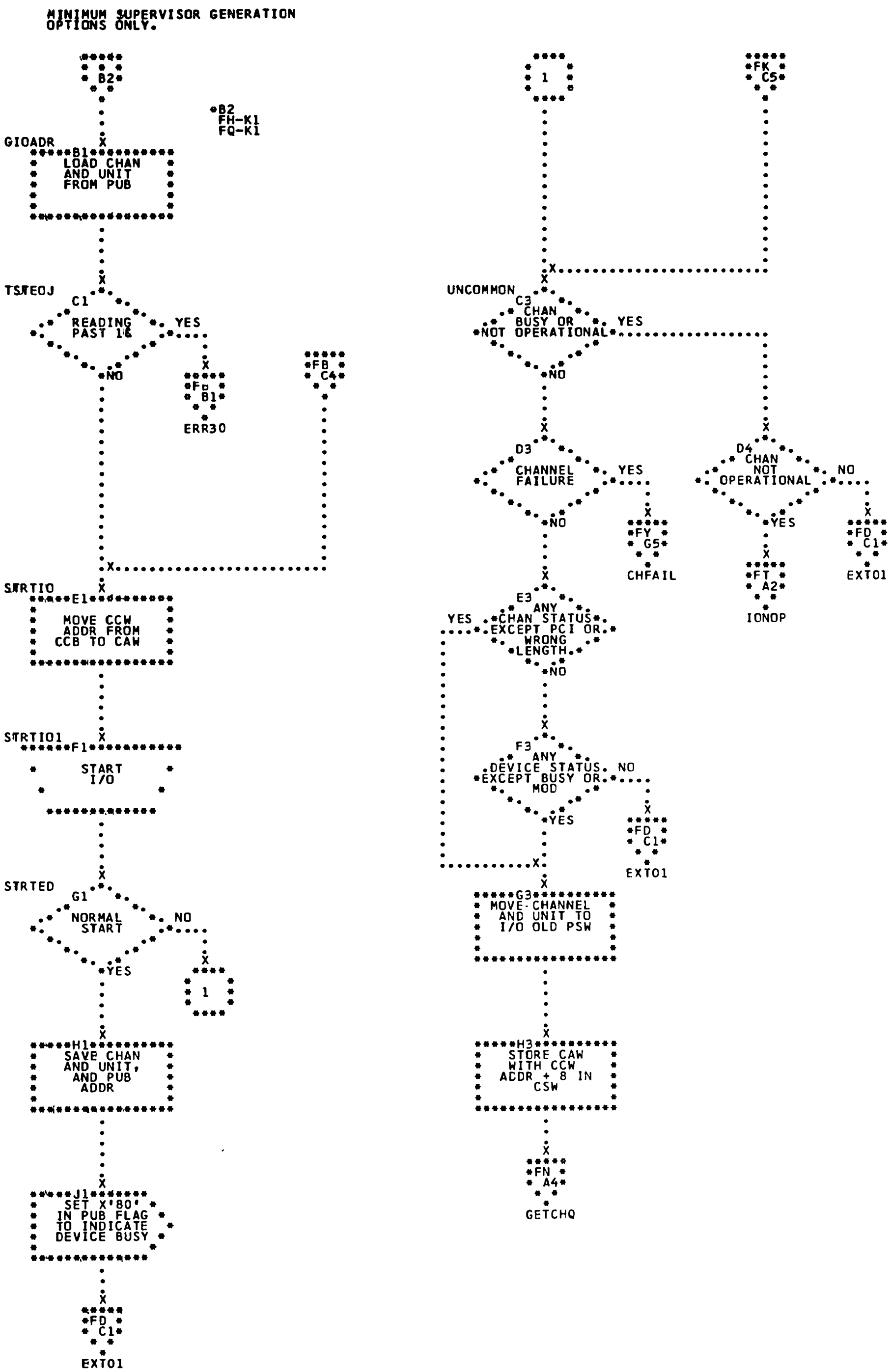


Chart FK. SGTCHS Start I/O-- Maximum Options (Part 1 of 3); Refer to Supervisor, Chart 15

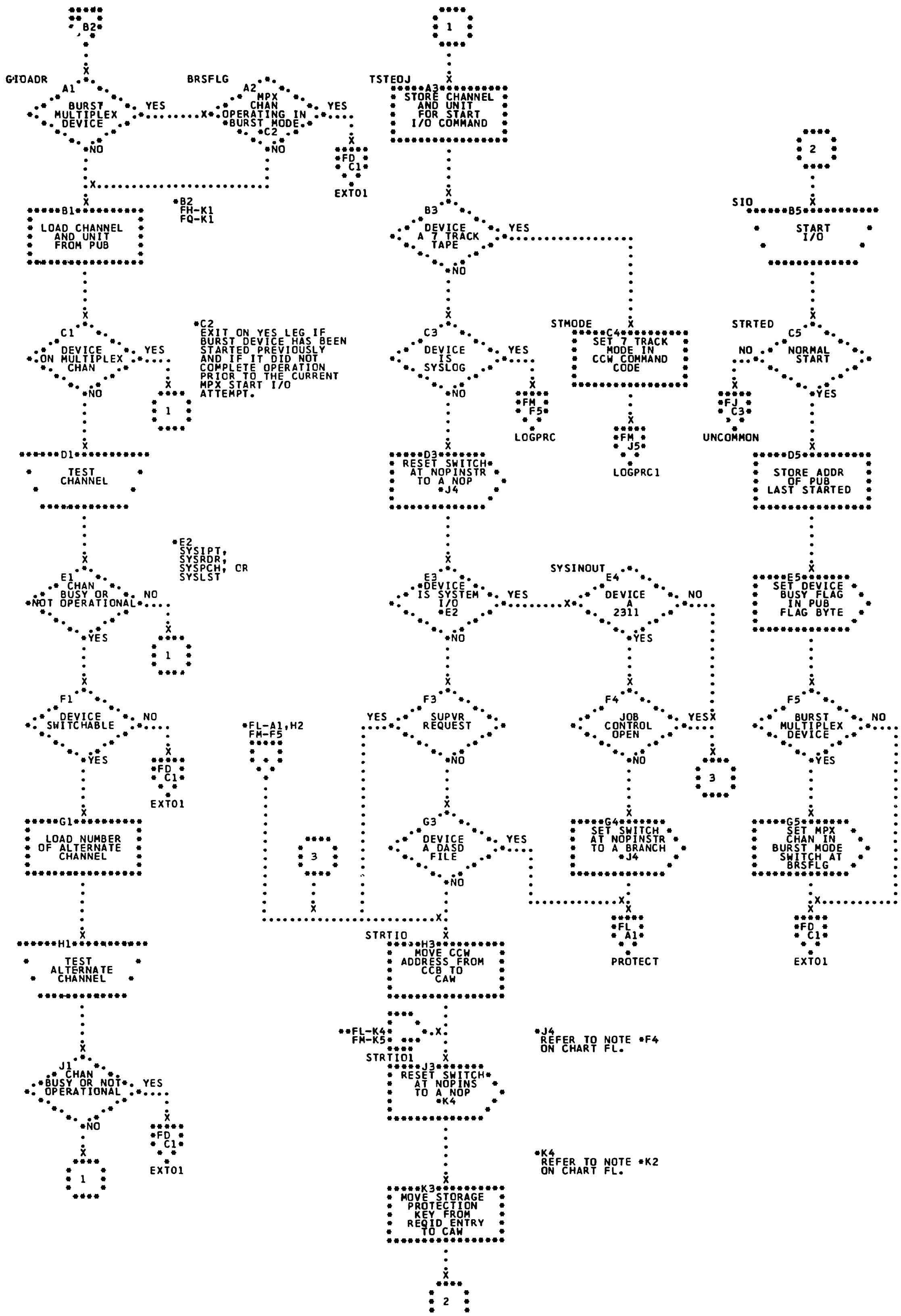


Chart FL. SGTCHS Start I/O-- Maximum Options (Part 2 of 3); Refer to Supervisor, Chart 15

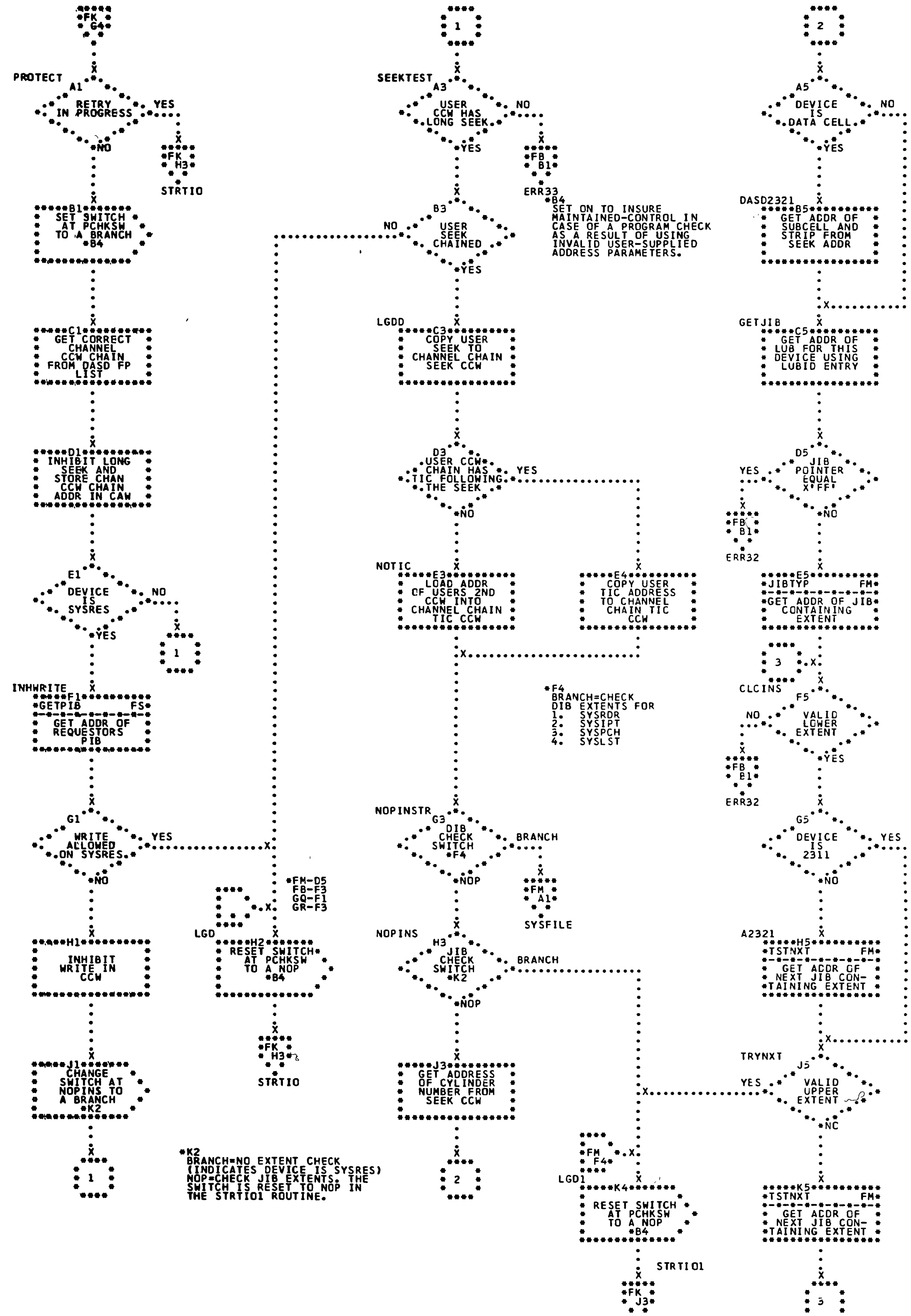


Chart FM. SGTCHS Start I/O-- Maximum Options (Part 3 of 3); Refer to Supervisor, Chart 15

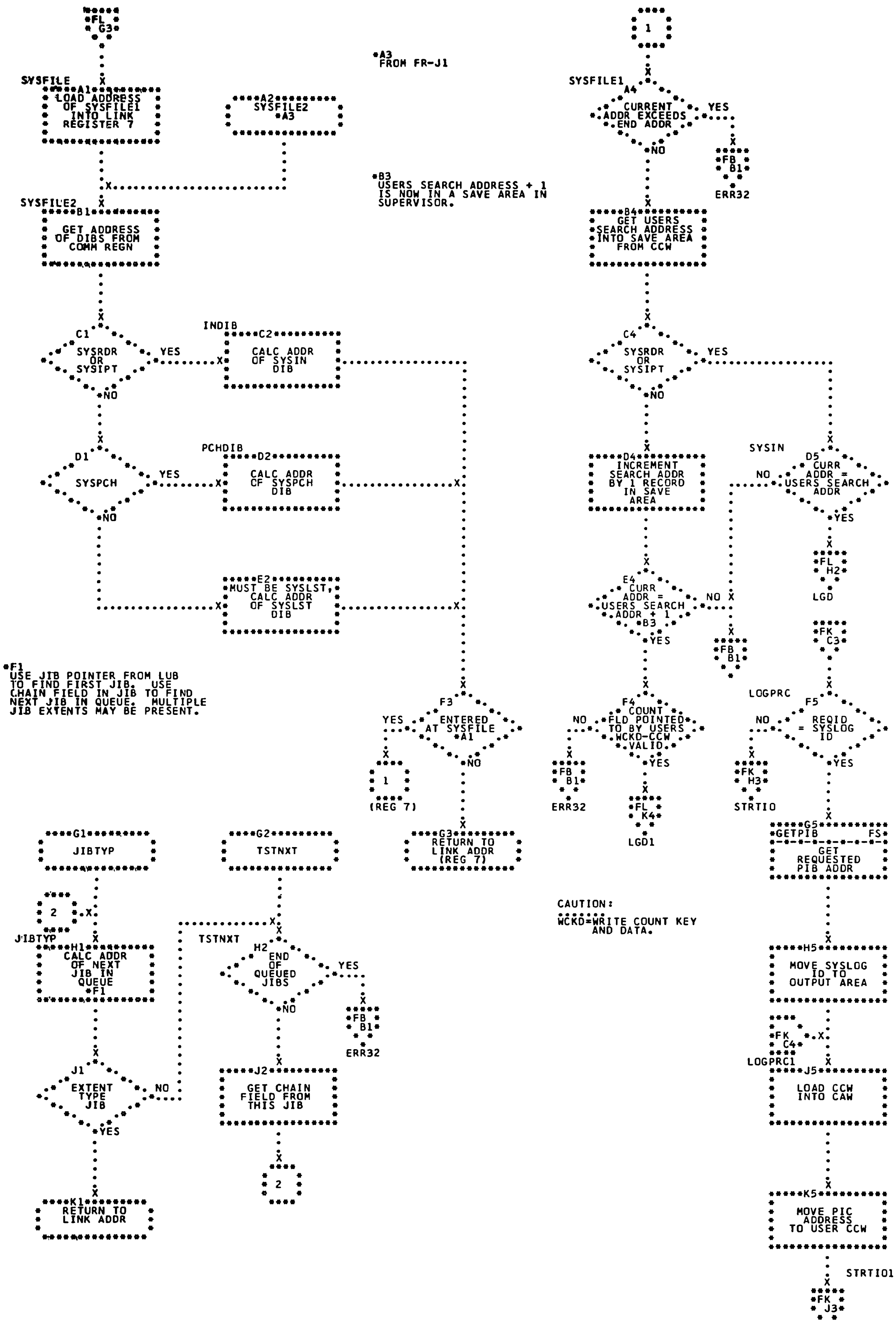


Chart FN. SGTCHS Macro-- I/O Interrupt (Part 1 of 5); Refer to Supervisor, Chart 15

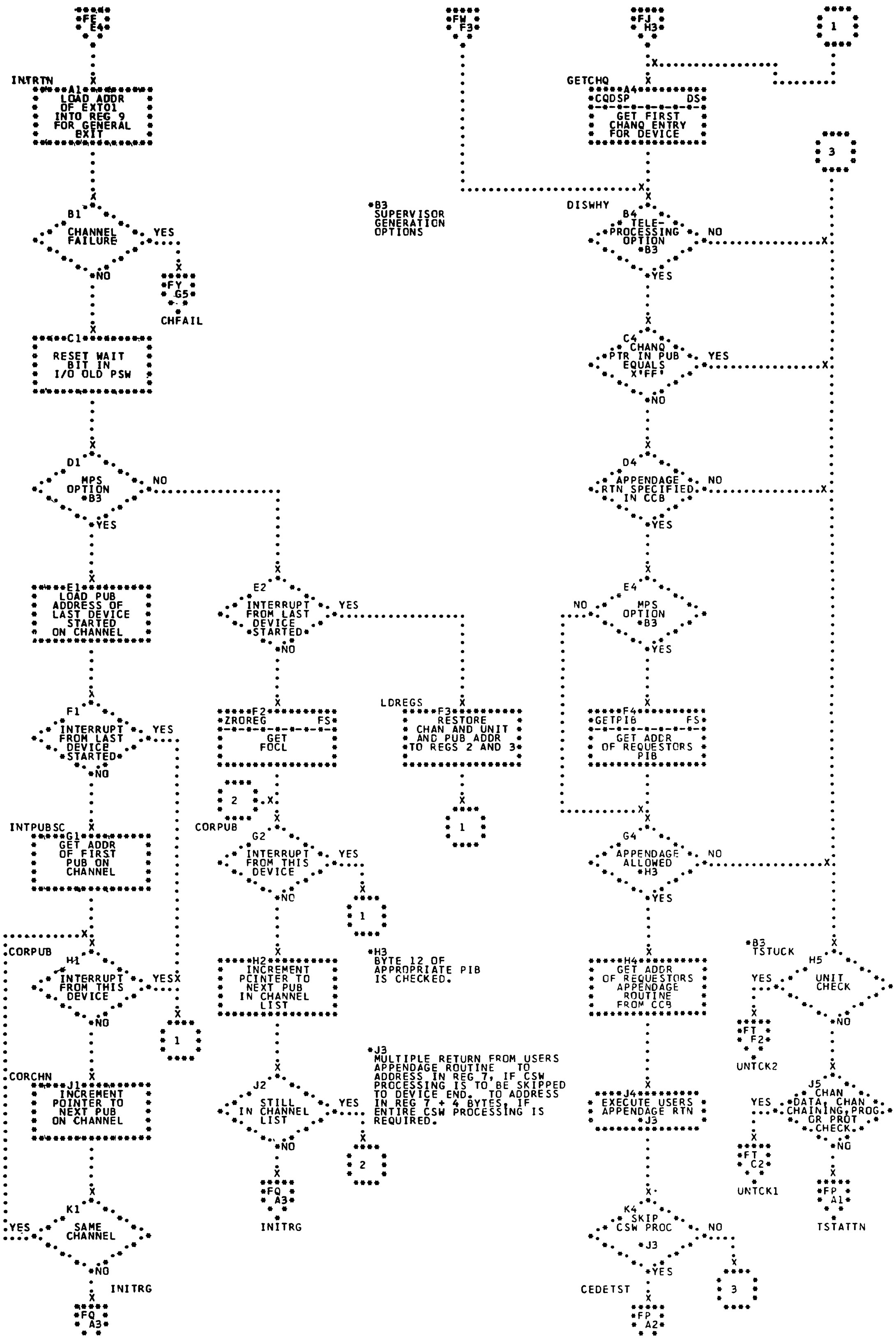


Chart FP. SGTCHS Macro-- I/O Interrupt (Part 2 of 5); Refer to Supervisor, Chart 15

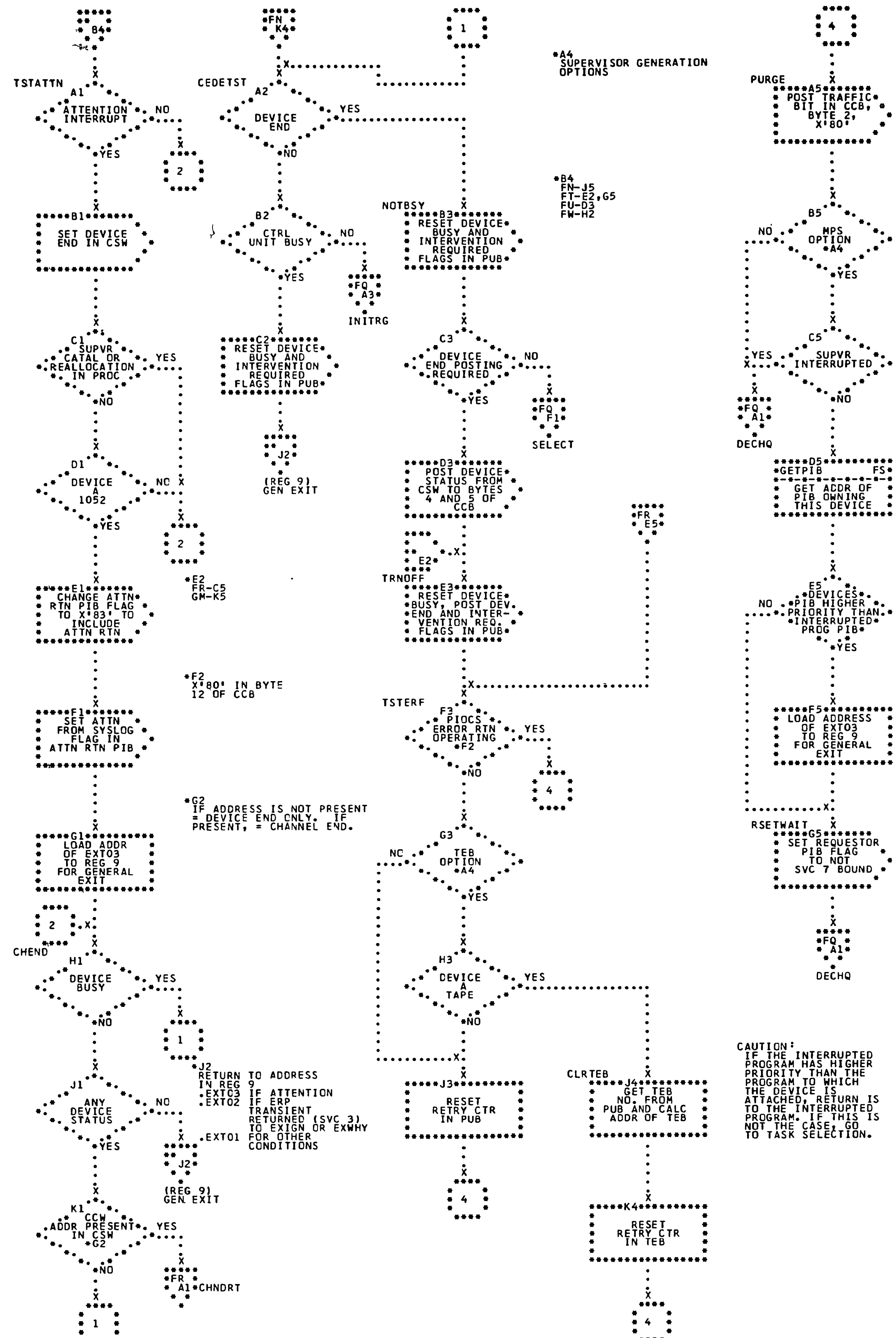


Chart FQ. SGTCHS Macro-- I/O Interrupt (Part 3 of 5); Refer to Supervisor, Chart 15

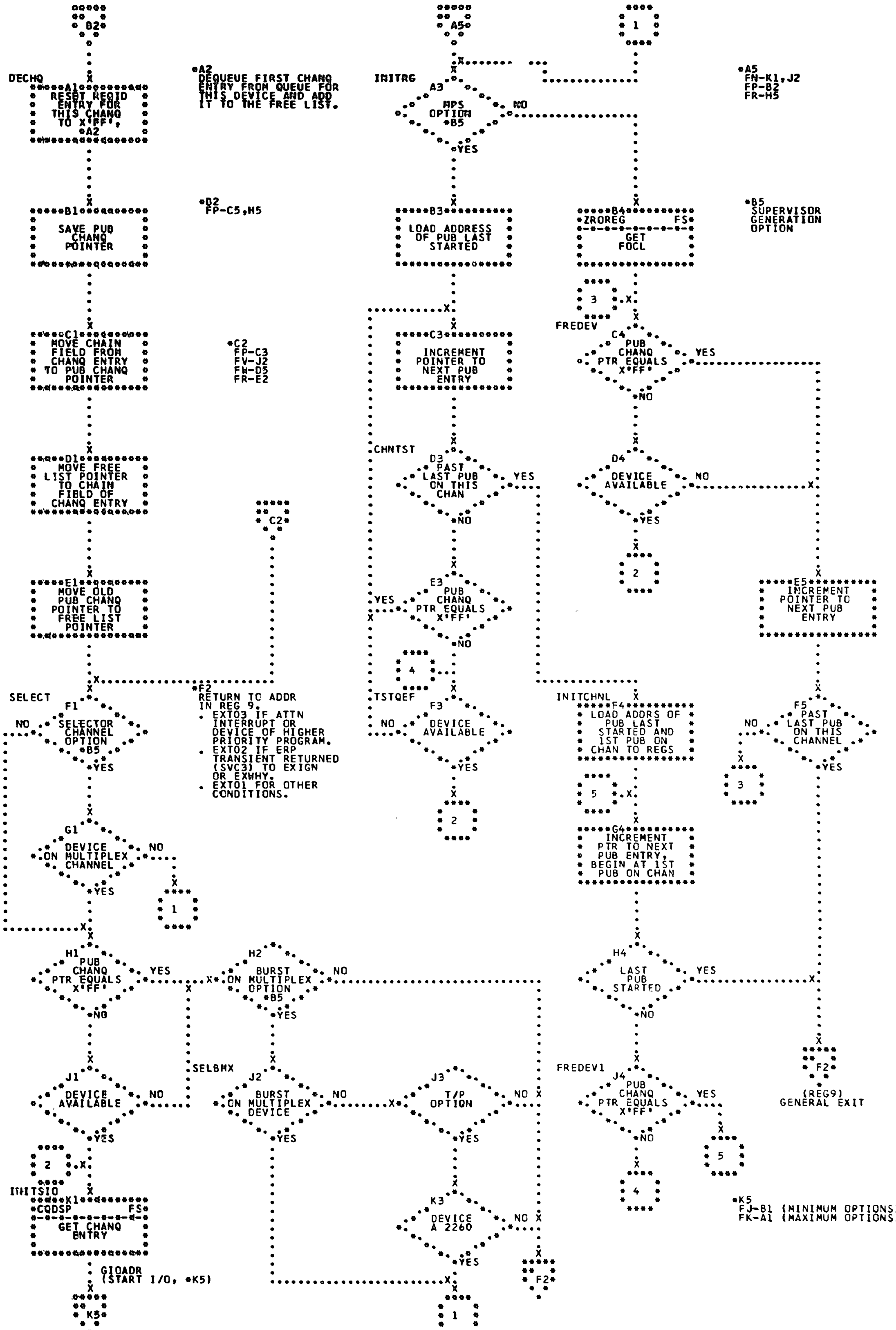


Chart FR. SGTCHS Macro-- I/O Interrupt (Part 4 of 5); Refer to Supervisor, Chart 15

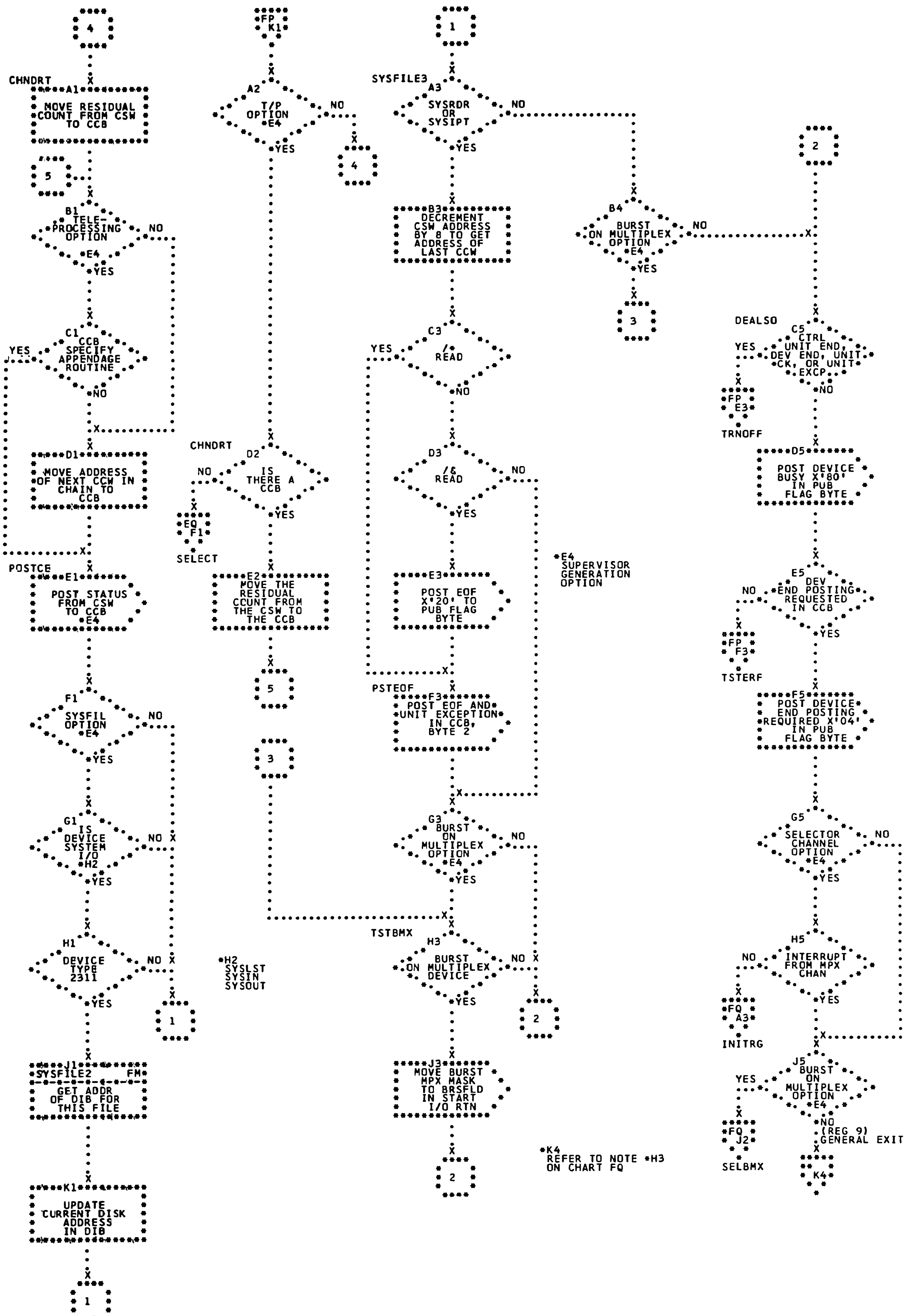


Chart FS. SGTCHS Macro-- I/O Interrupt (Part 5 of 5); Refer to Supervisor, Chart 15

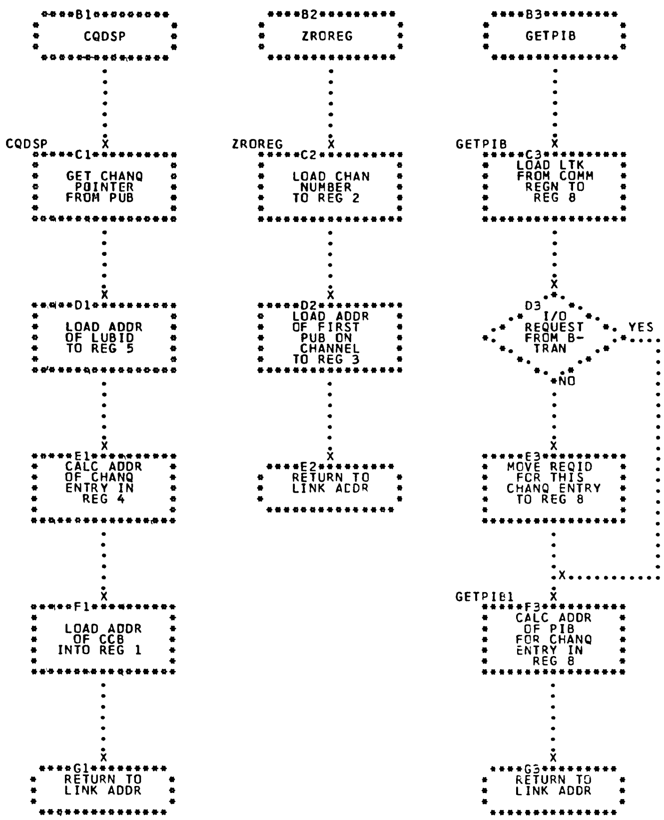


Chart FT. SGUNCK Macro-- Unit Check Routine Entries; Refer to Supervisor, Chart 17

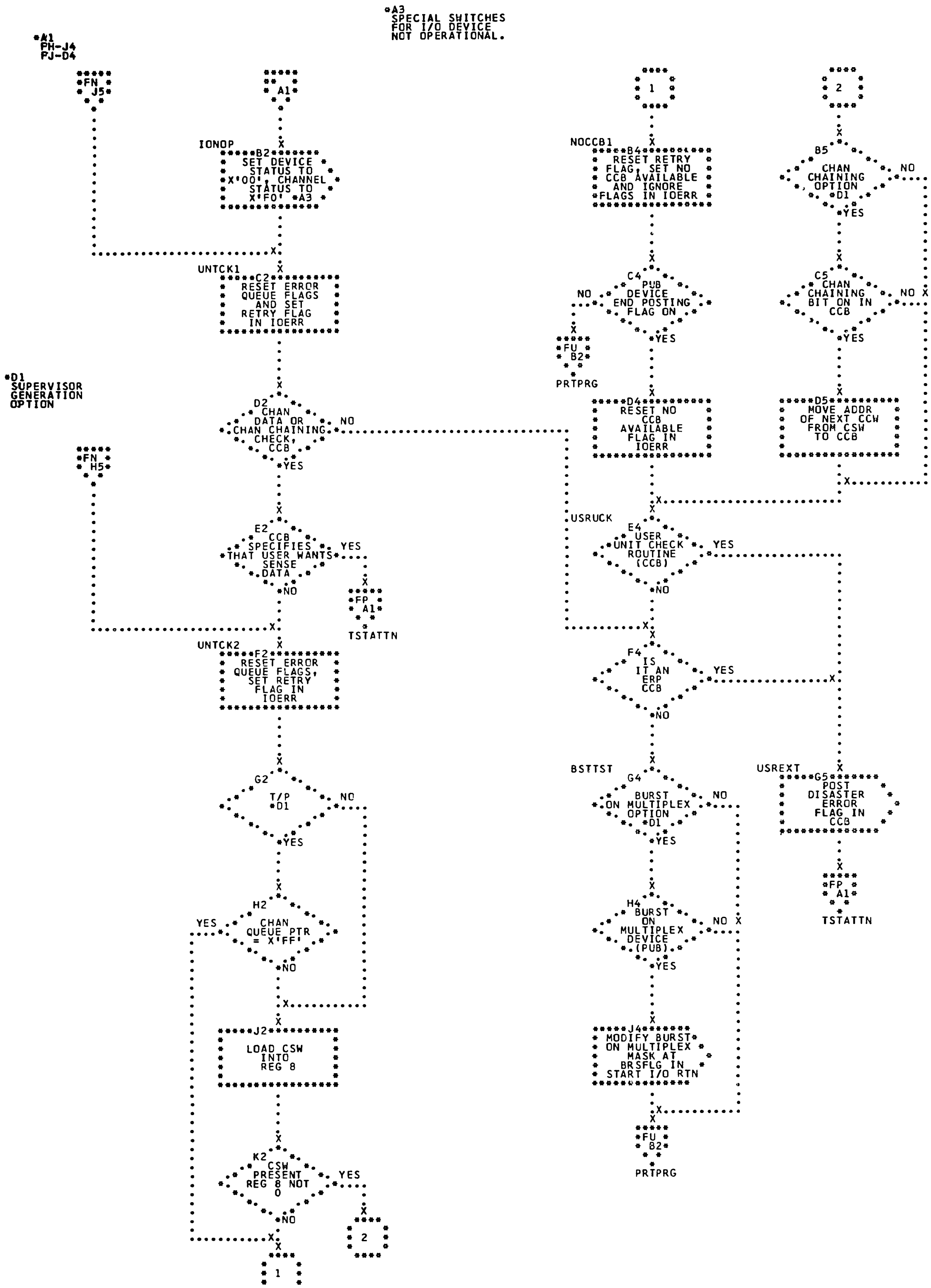


Chart FU. SGUNCK Macro-- Unit Check Routine Build Error Queue Entry; Refer to Supervisor, Chart 17

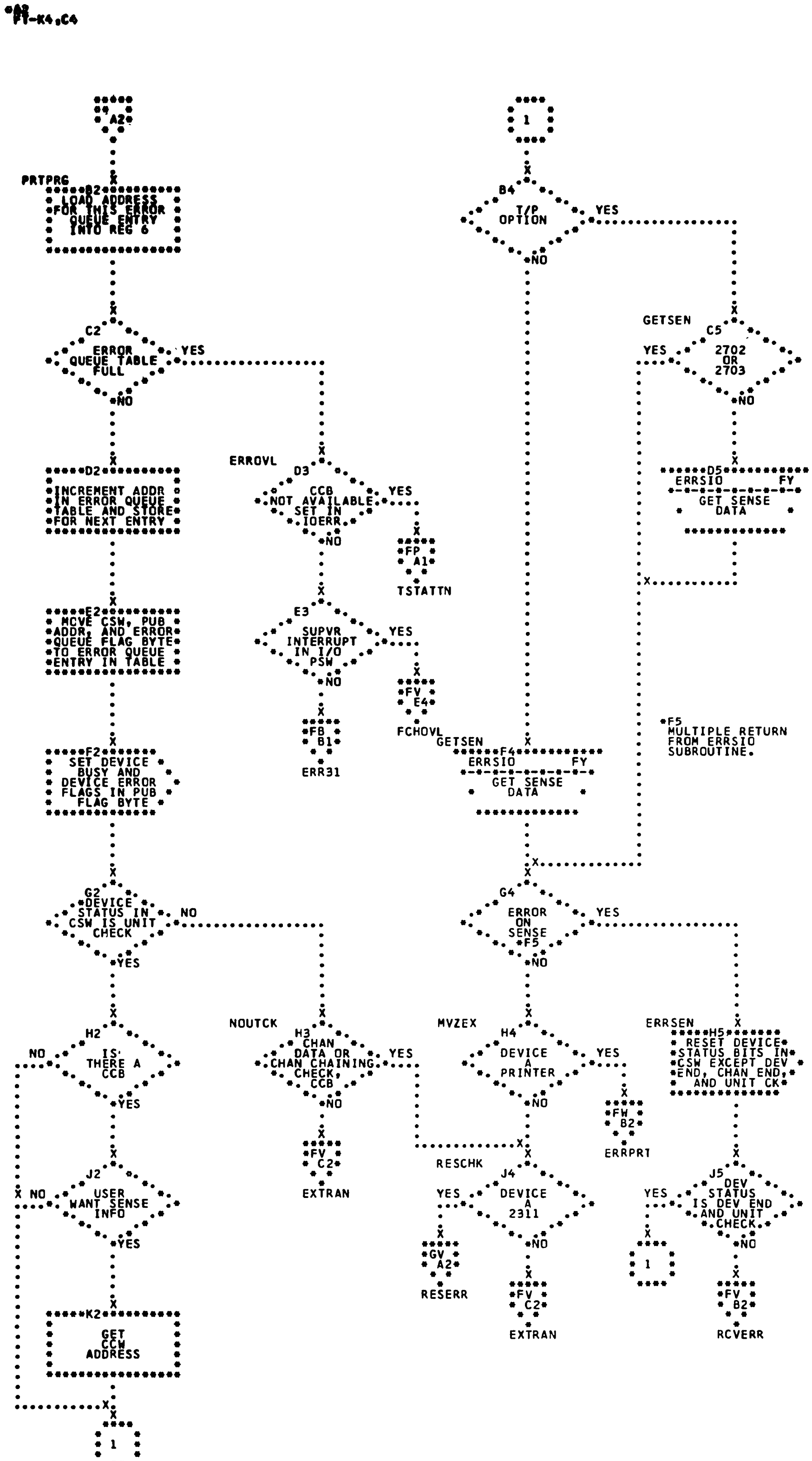


Chart FV. SGUNCK Macro Error Recovery Exits (Part 1 of 2); Refer to Supervisor, Chart 17

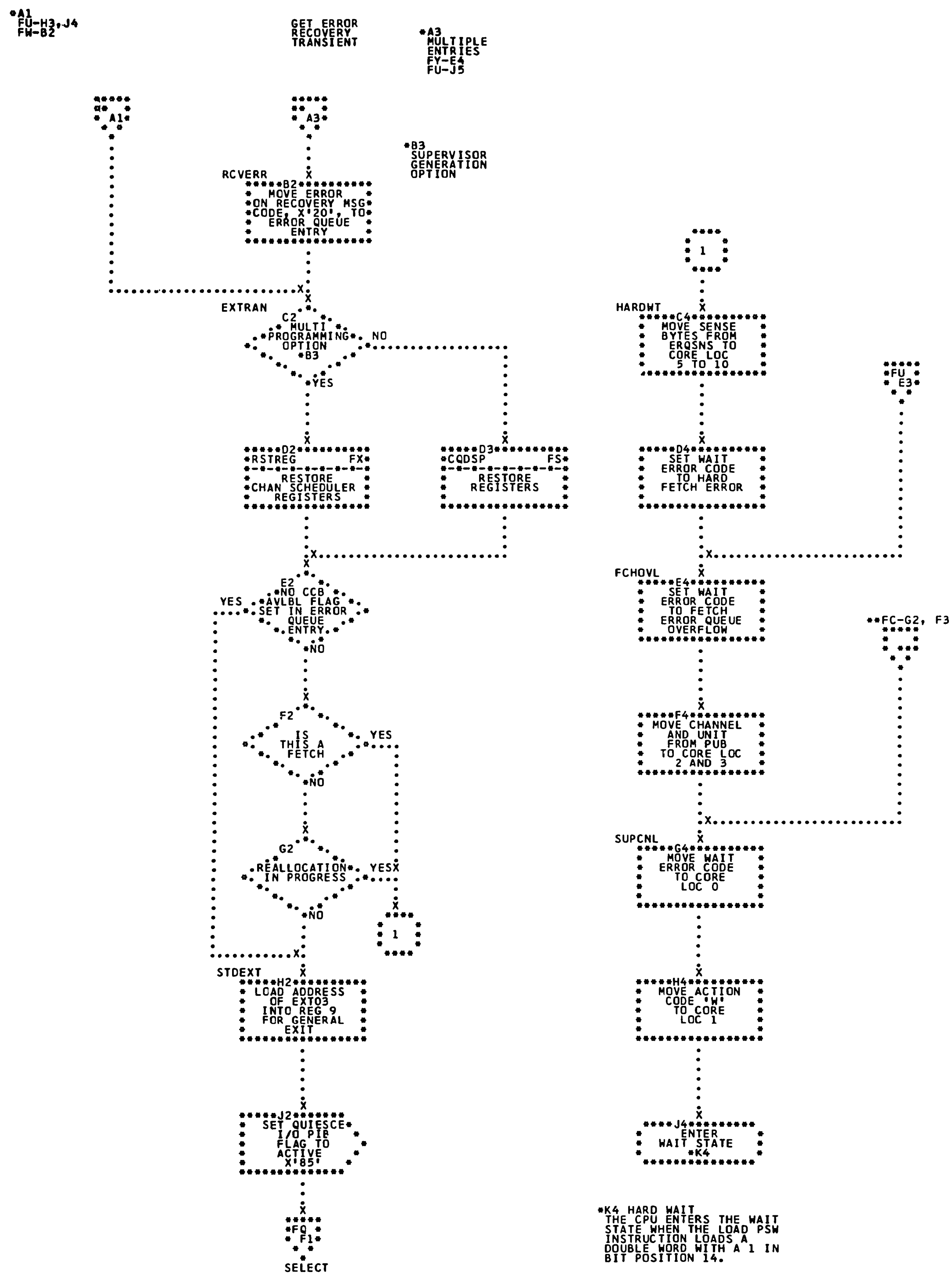


Chart FW. SGUNCK Macro Error Recovery Exits (Part 2 of 2); Refer to Supervisor, Chart 17

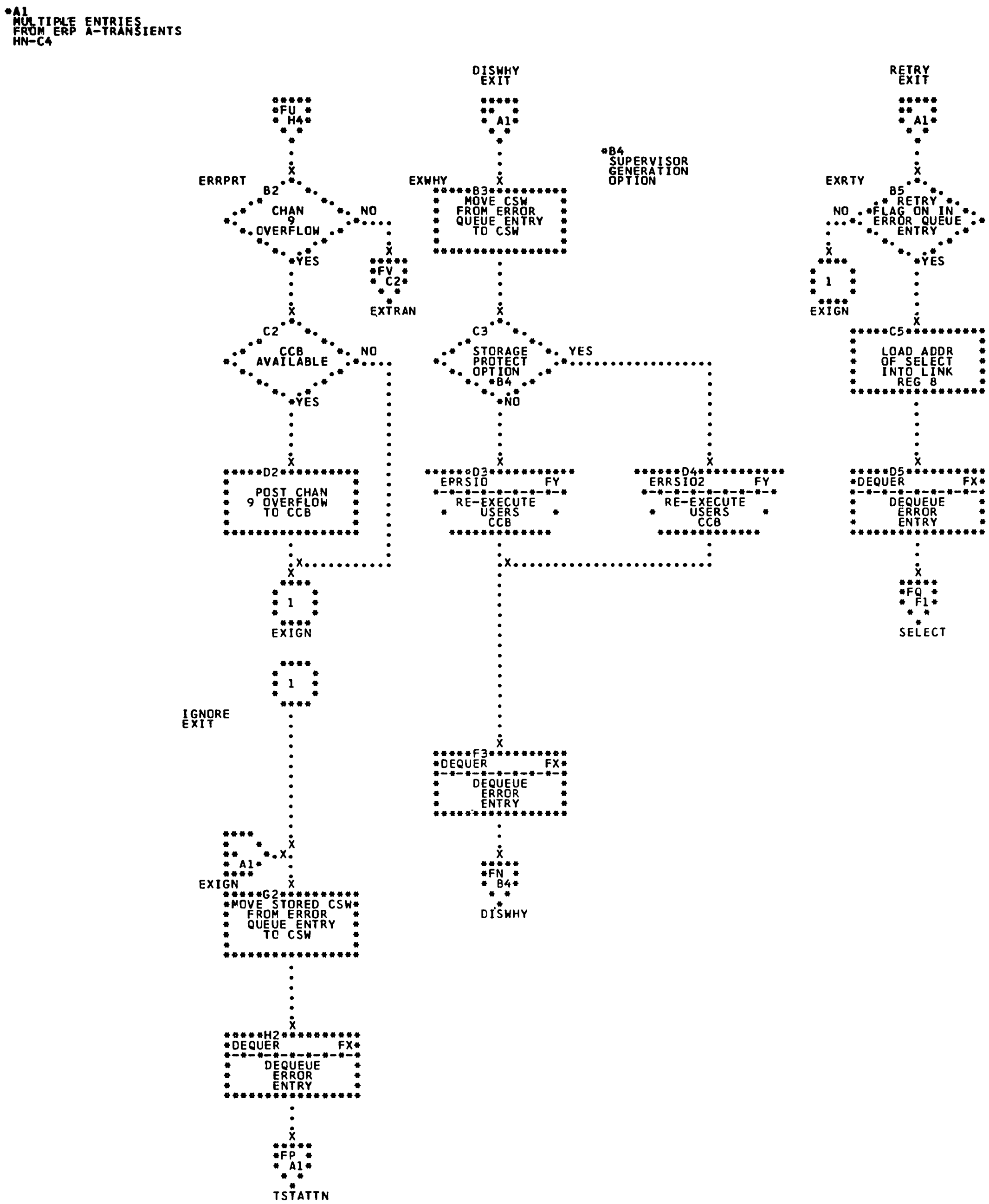


Chart FX. SGUNCK Macro-- DEQUER and RSTREG Subroutines; Refer to Supervisor, Chart 17

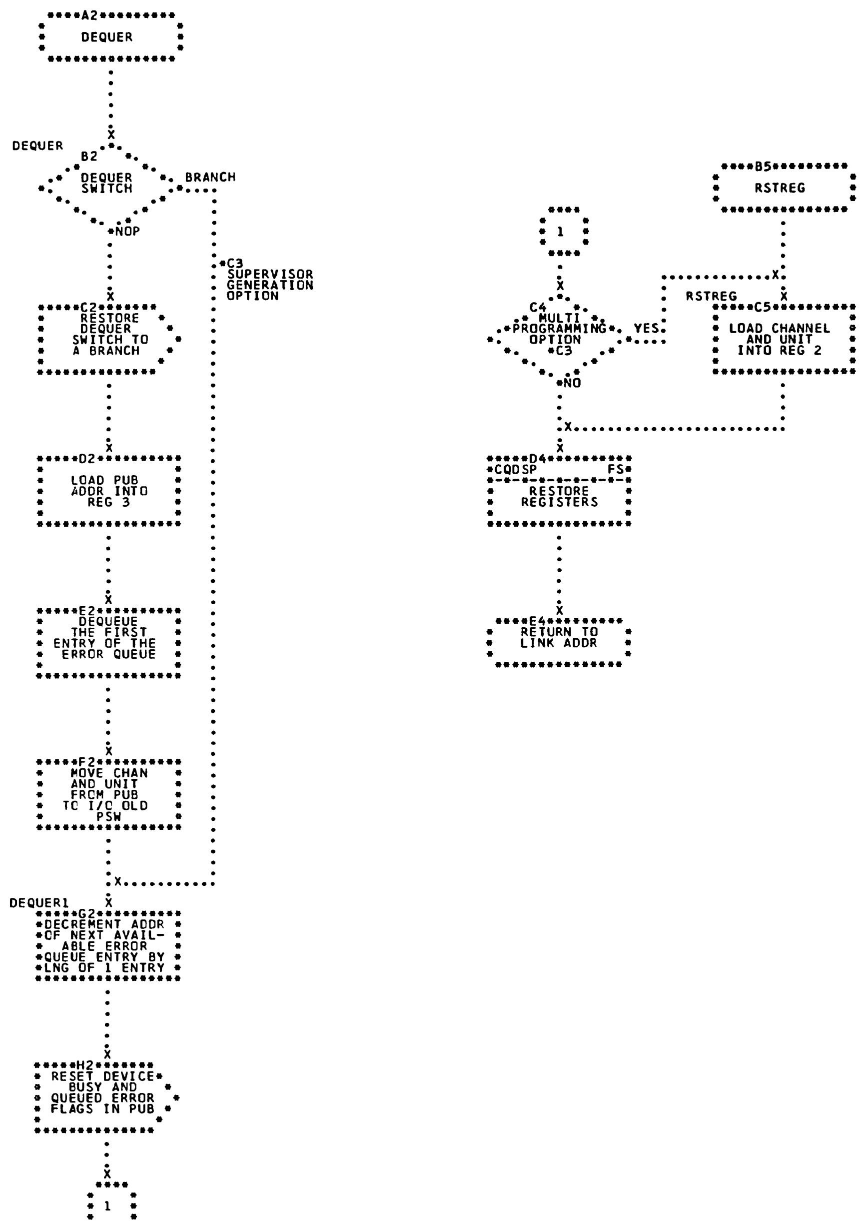


Chart FY. SGUNCK Macro-- Error Start I/O Subroutine; Refer to Supervisor, Chart 17

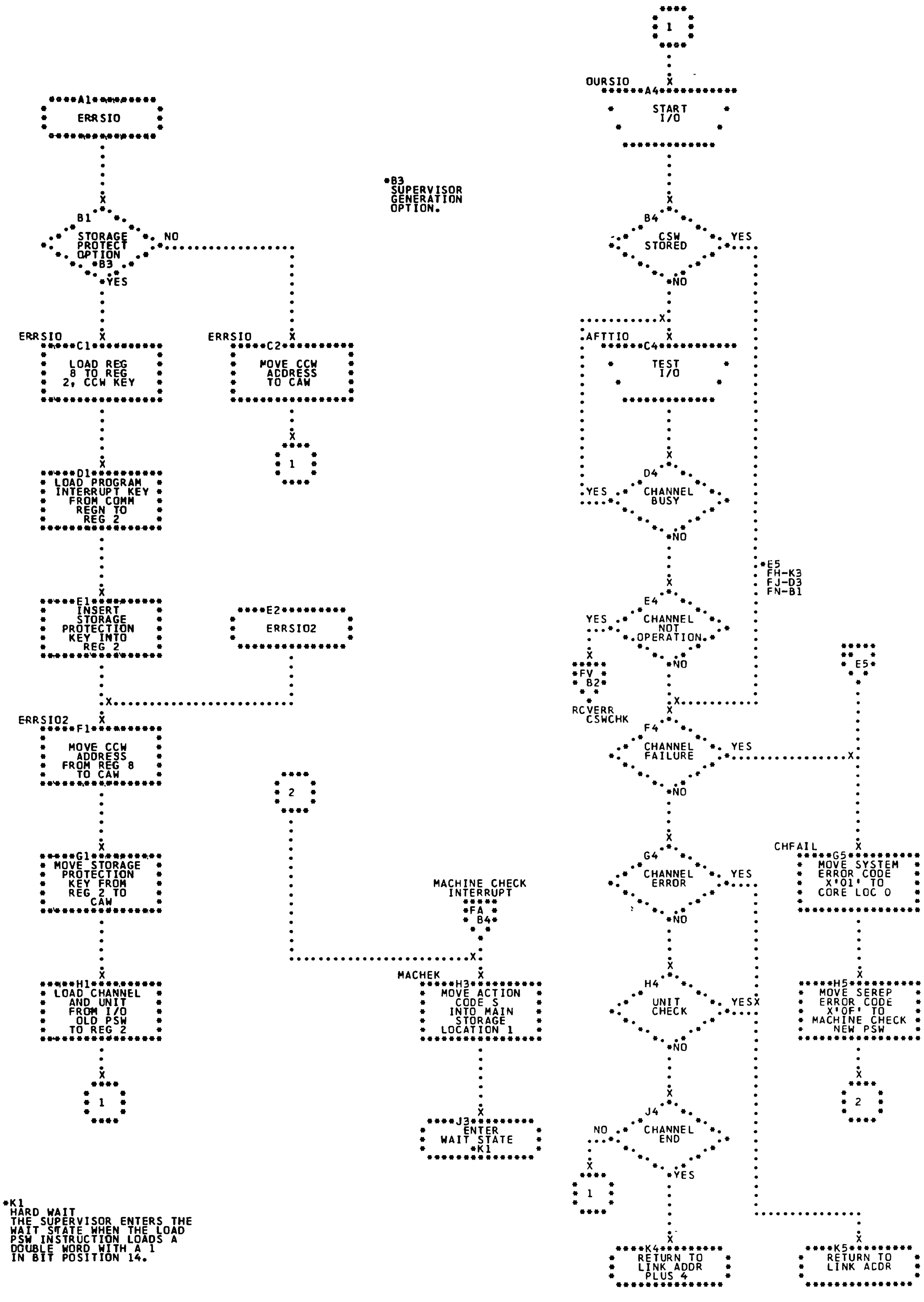


Chart FZ. SGUNCK Macro-- Quiesce I/O Task; Refer to Supervisor, Chart 17

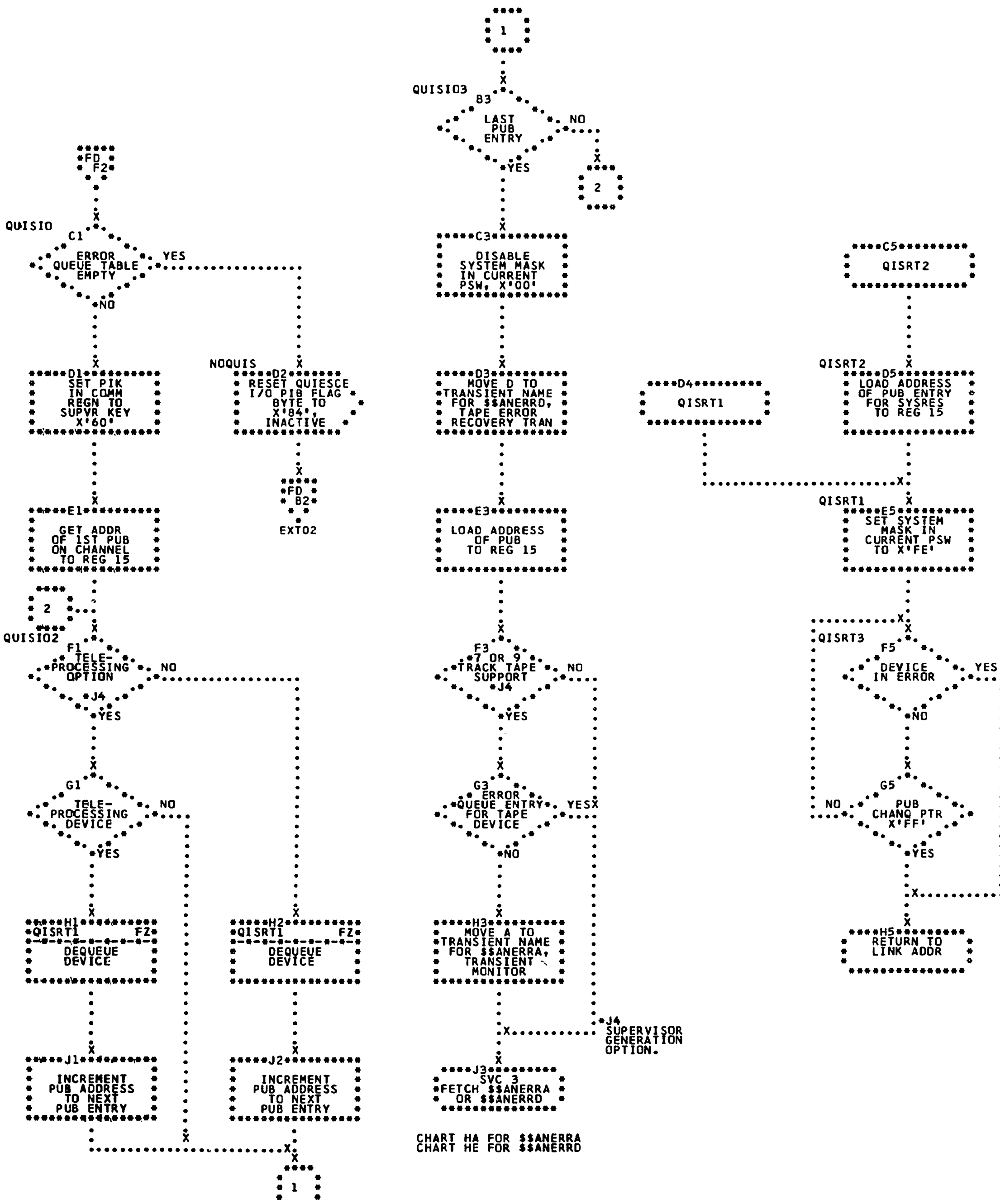


Chart GA. SGDFCH Macro-- Fetch with MPS Option (Part 1 of 3); Refer to Supervisor, Chart 14

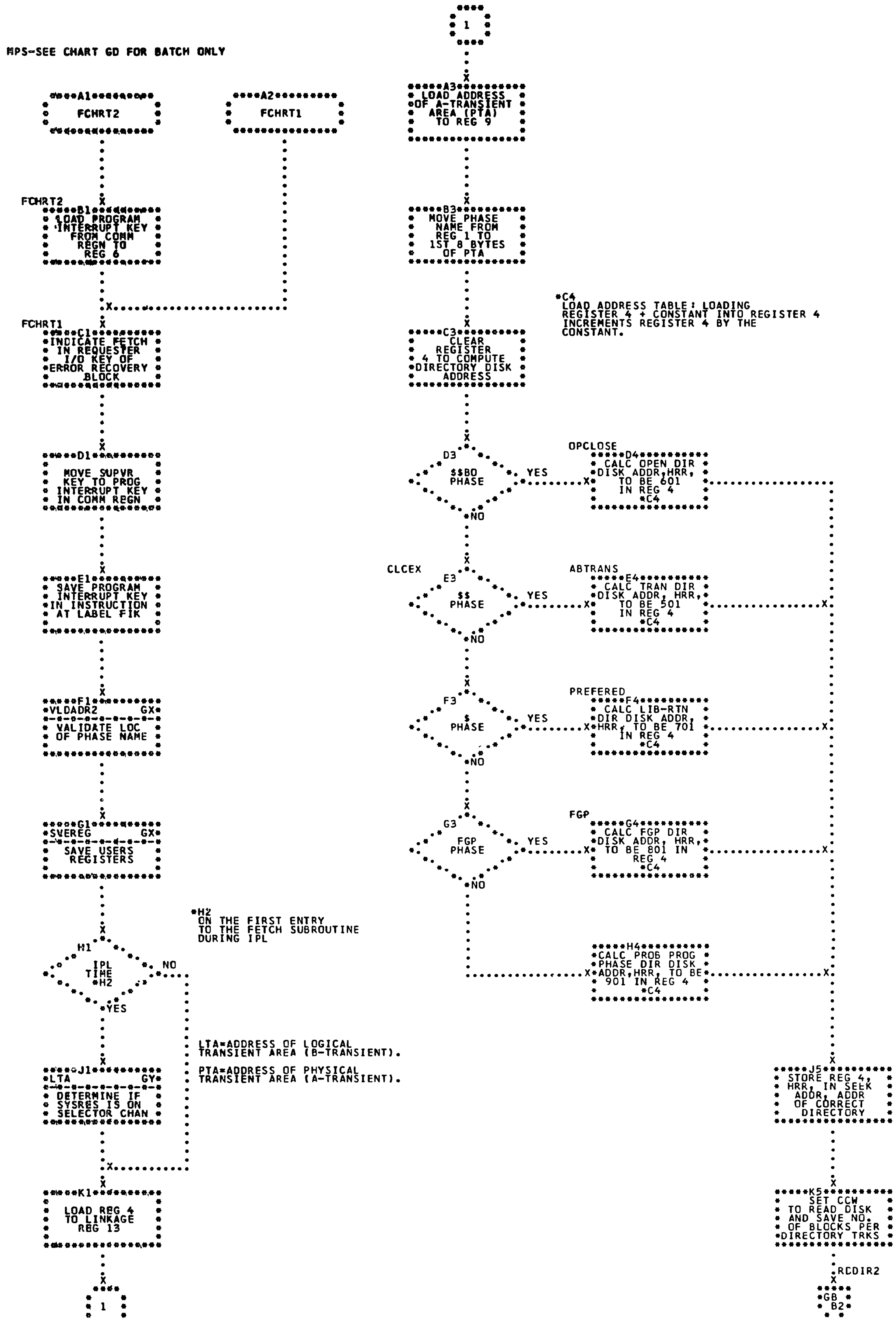
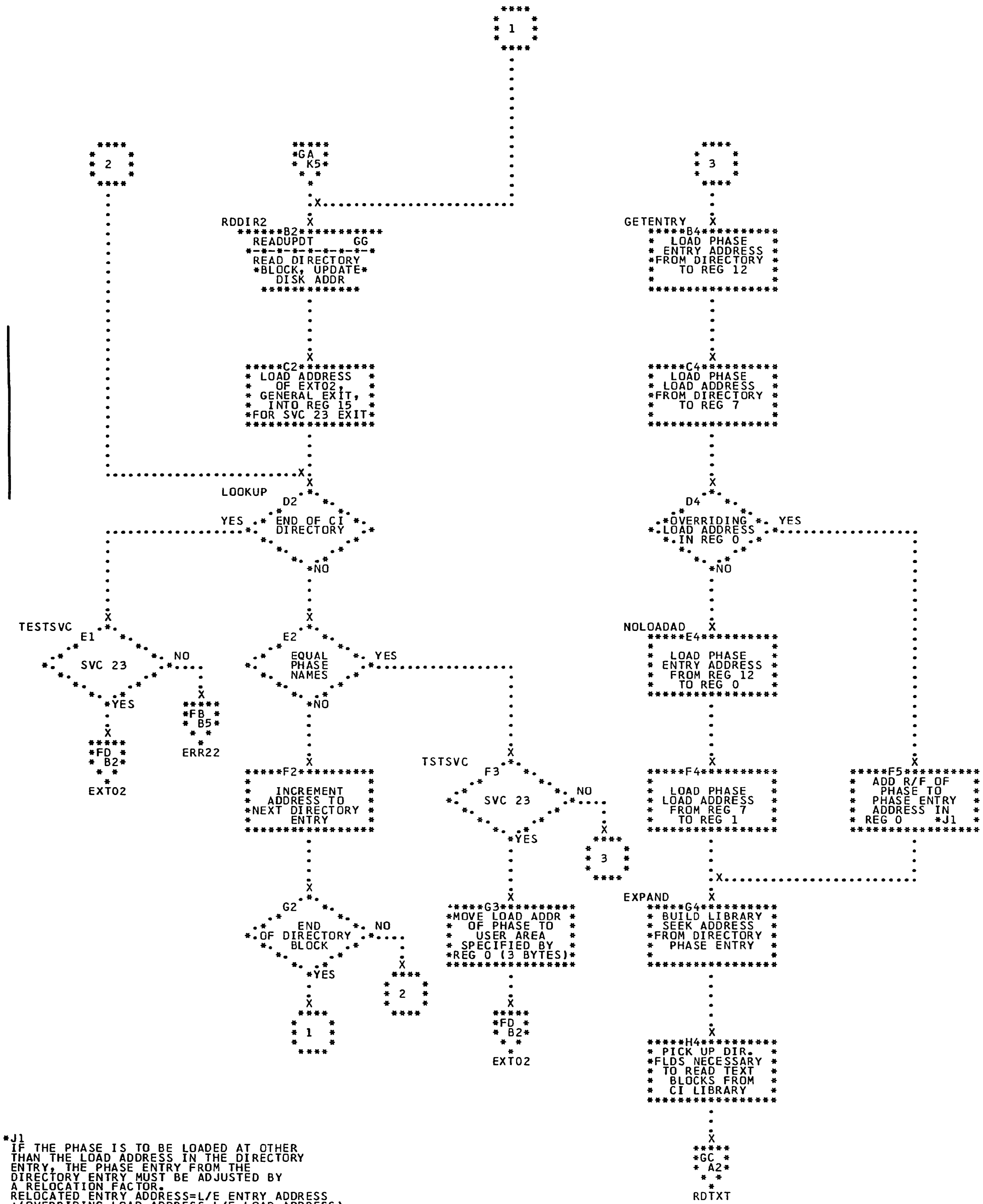


Chart GB. SGDFCH Macro-- Fetch with MPS Option (Part 2 of 3), Refer to Supervisor, Chart 14




*J1
IF THE PHASE IS TO BE LOADED AT OTHER THAN THE LOAD ADDRESS IN THE DIRECTORY ENTRY, THE PHASE ENTRY FROM THE DIRECTORY ENTRY MUST BE ADJUSTED BY A RELOCATION FACTOR.
RELOCATED ENTRY ADDRESS=L/E ENTRY ADDRESS +(OVERRIDING LOAD ADDRESS-L/E LOAD ADDRESS).

Chart GC. SGDFCH Macro-- Fetch with MPS Option (Part 3 of 3); Refer to Supervisor, Chart 14

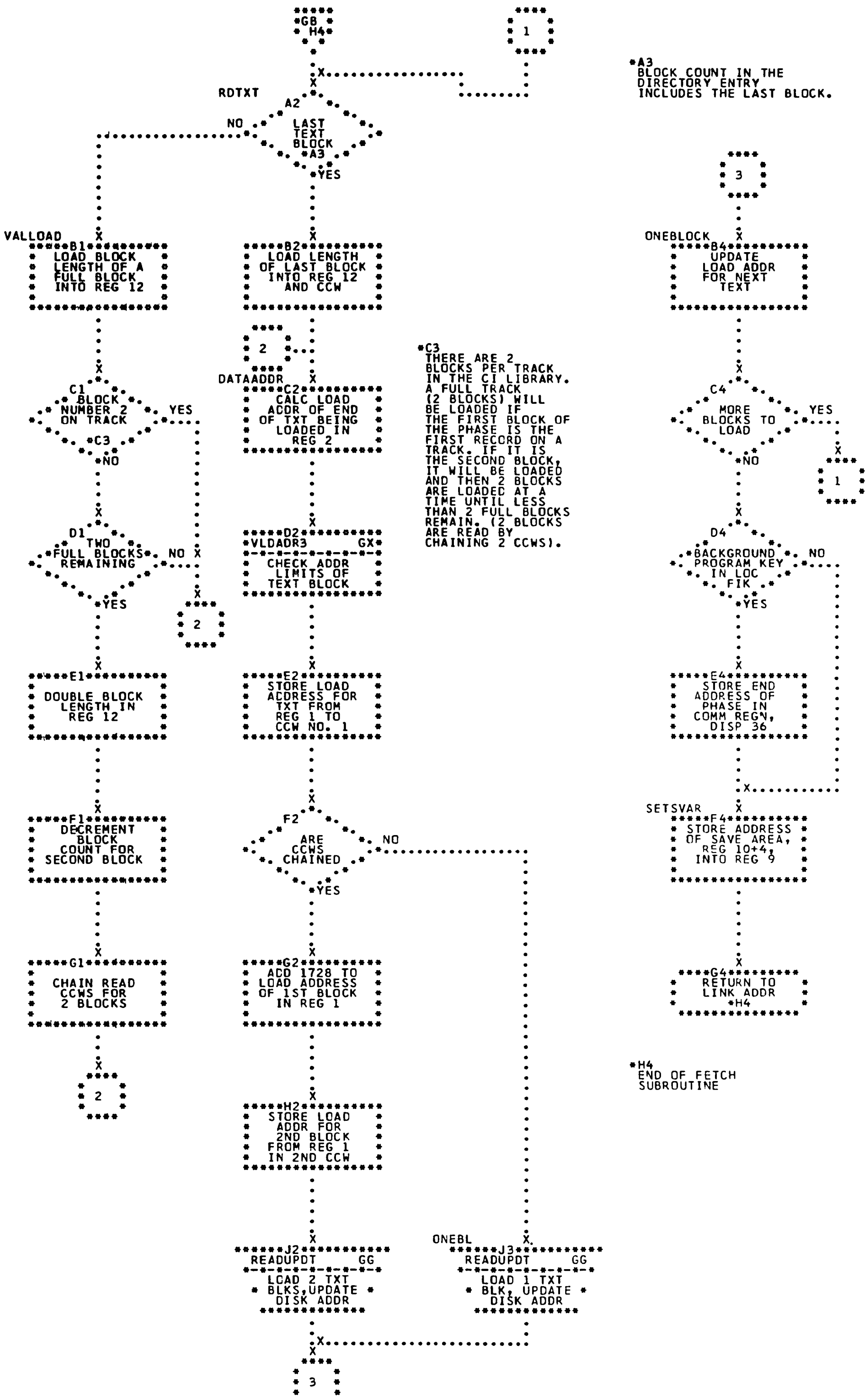


Chart GD. SGDFCH Macro-- Fetch with Batch Only Option (Part 1 of 2); Refer to Supervisor, Chart 14

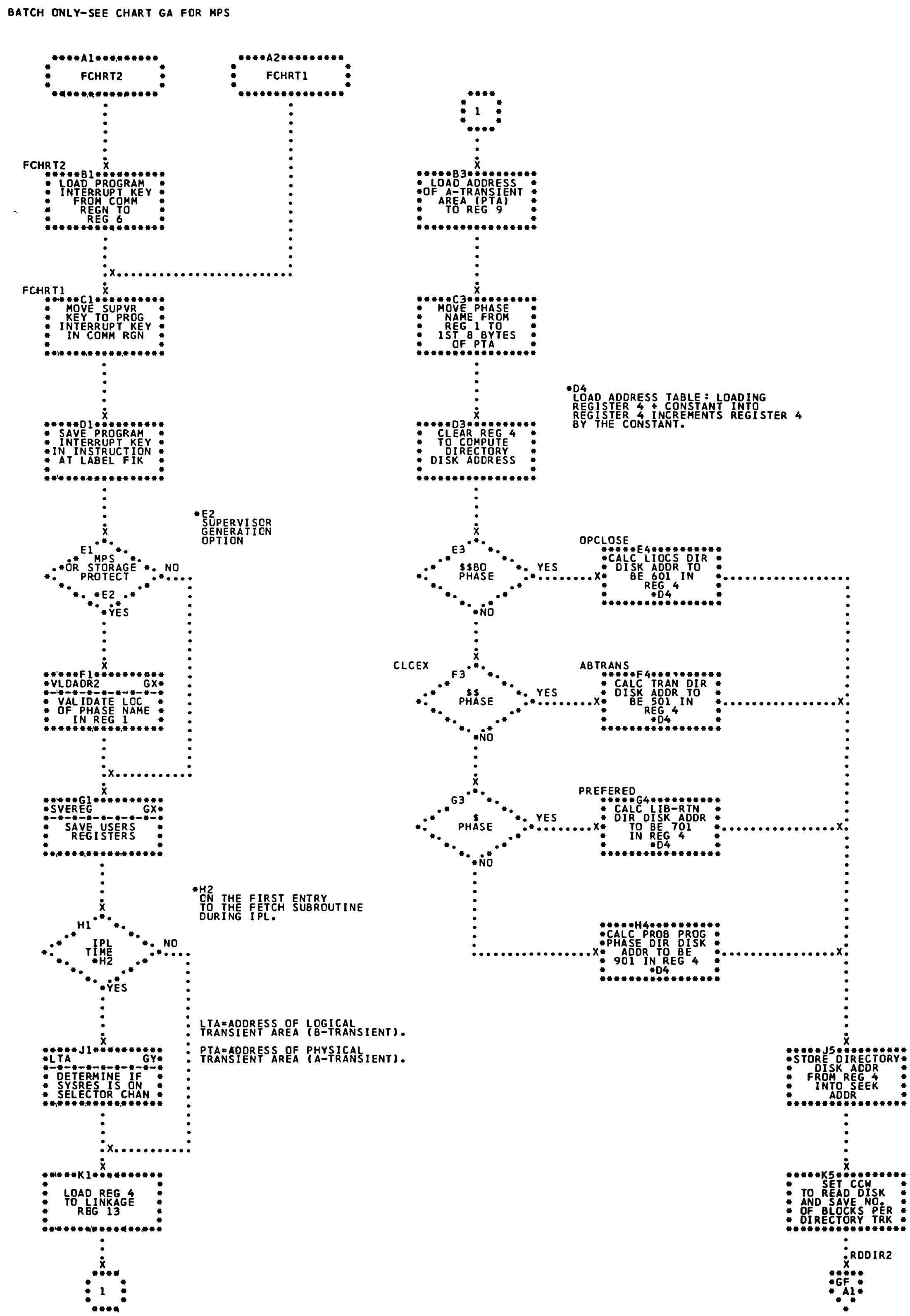


Chart GF. SGDFCH Macro-- Fetch with Batch Only Option (Part 2 of 2); Refer to Supervisor, Chart 14

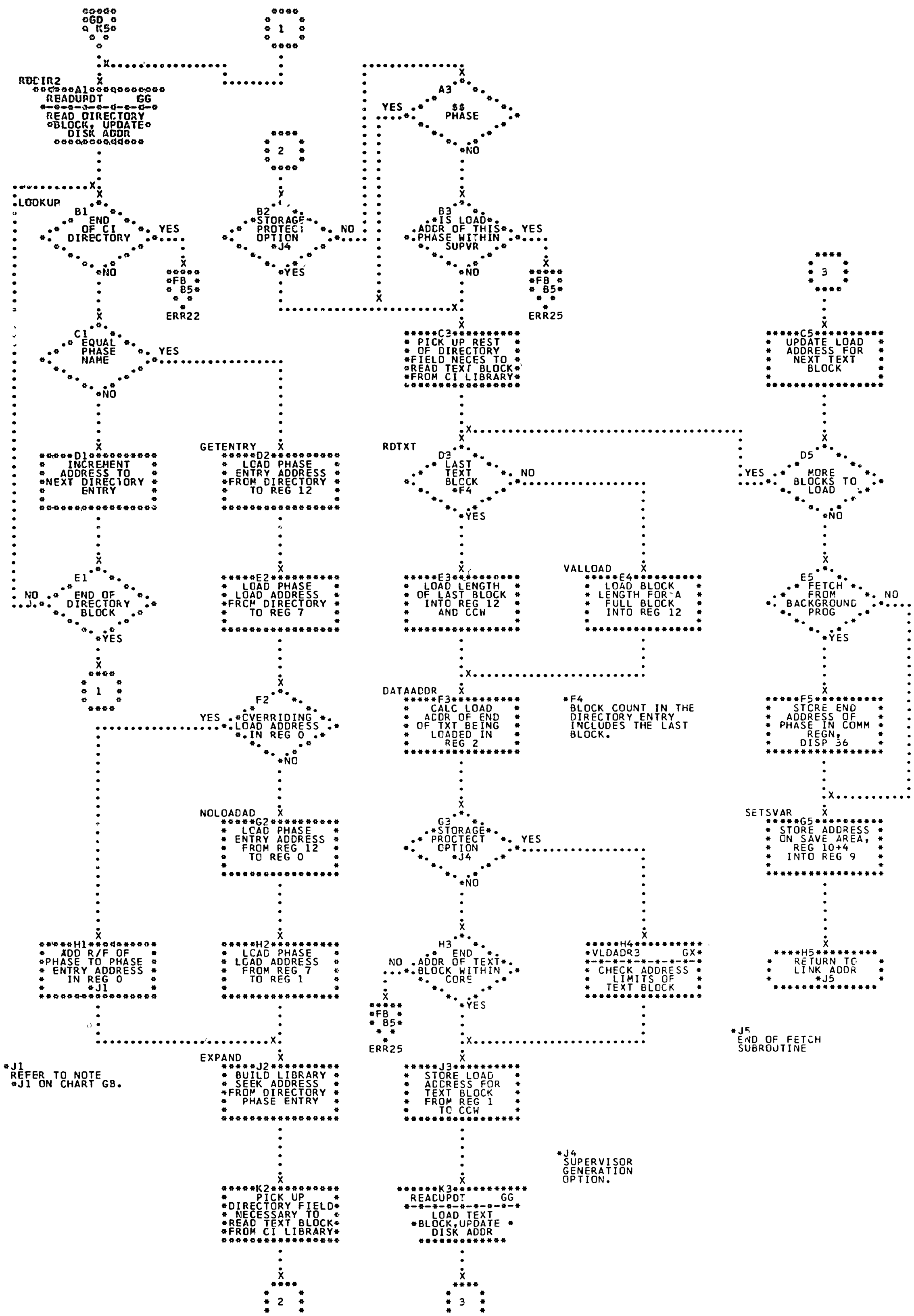


Chart GG. SGDFCH Macro-READUPDT and RSTPUB Subroutine; Refer to Supervisor, Chart 14

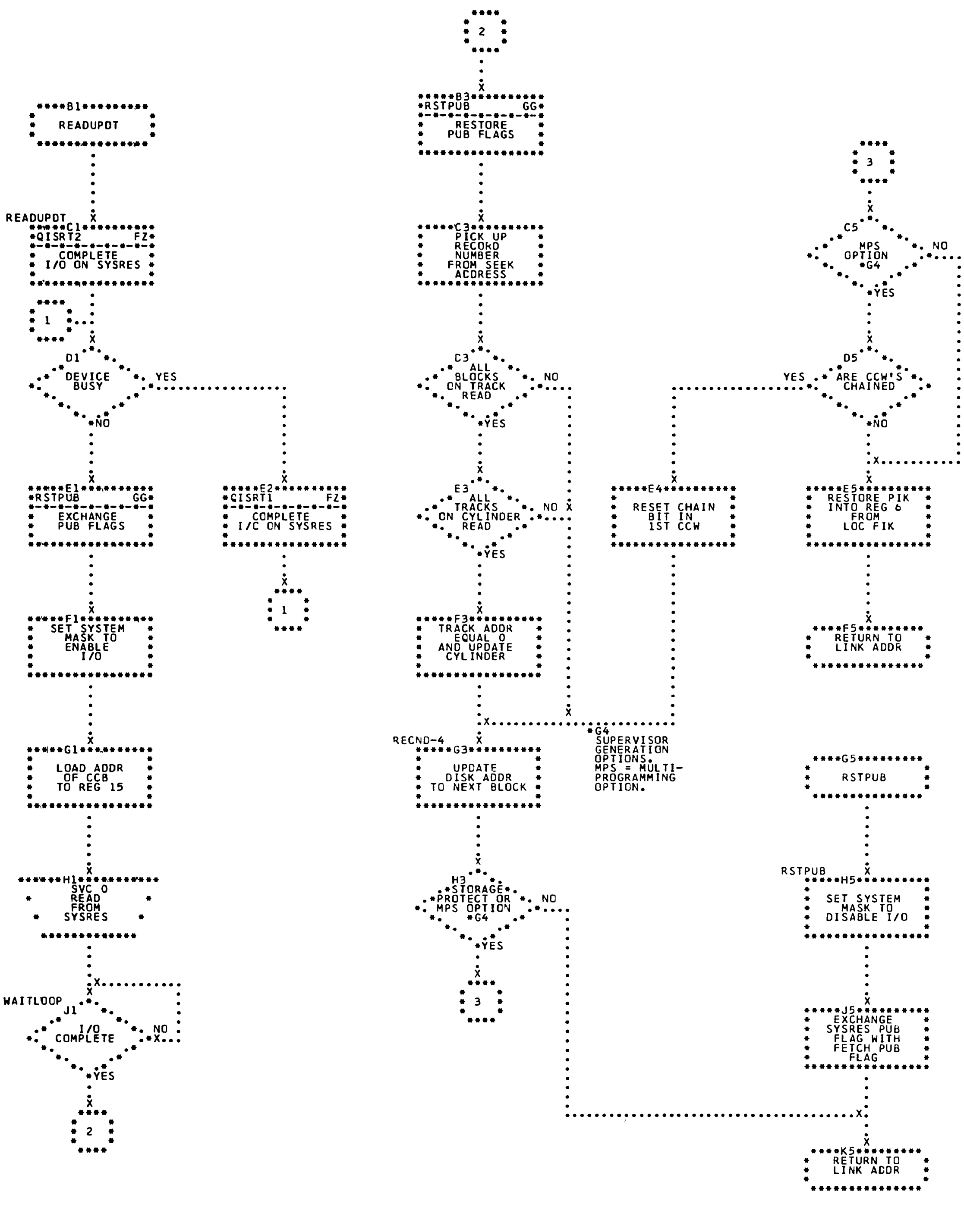
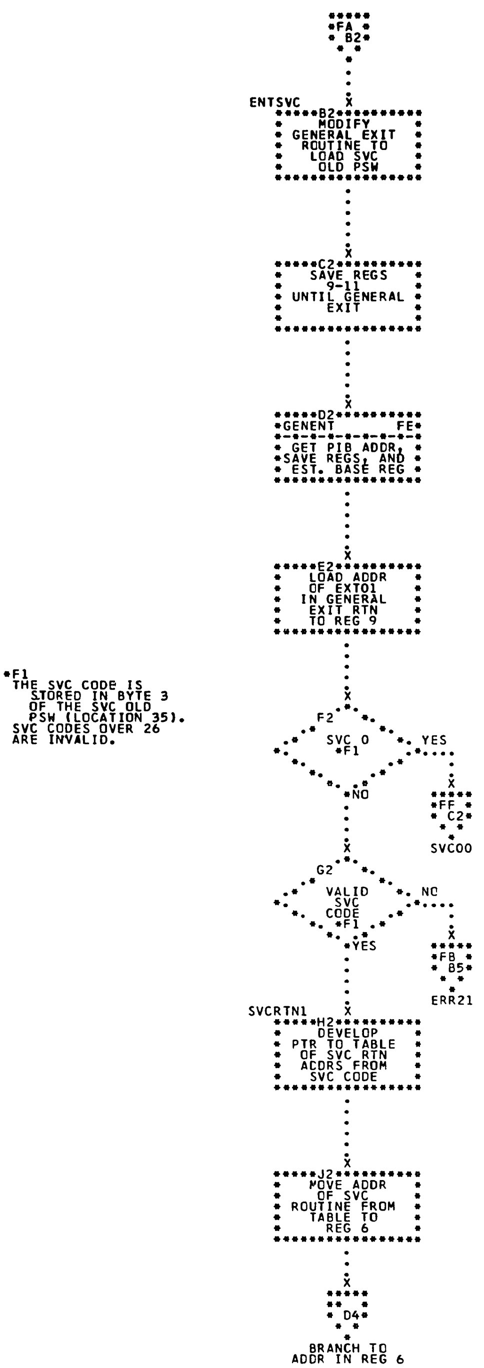


Chart GH. SGSVC Macro-- SVC Interrupt Handler; Refer to Supervisor, Chart 14




*D4

| SVC | OPTION | LABEL | CHART |
|---|---|---|---|
| 0 | ALL | SVC00 | FF-C2 |
| 1 | ALL | SVC01 | GJ-B1 |
| 2 | MPS | SVC02 | GK-B1 |
| | NO MPS | SVC02 | GL-B1 |
| 3 | ALL | SVC03 | GM-B1 |
| 4 | ALL | SVC04 | GM-B3 |
| 5 | ALL | SVC05 | GJ-D3 |
| 6 | ALL | ERR23 | FB-B5 |
| 7 | ALL | SVC07 | GM-B4 |
| 8 | ALL | SVC08 | GN-B1 |
| 9 | ALL | SVC09 | GN-B3 |
| 10 | TIMER | SVC10 | GN-B5 |
| | NO TIMER | ERR21 | FB-A5 |
| 11 | MPS | SVC11 | GK-B4 |
| | NO MPS | SVC11 | GL-B3 |
| 12 | ALL | SVC12 | GJ-F4 |
| 13 | ALL | SVC13 | GJ-H5 |
| 14 | ALL | ERR10 | FB-B5 |
| 15 | ALL | SVC15 | FF-A2 |
| 16 | PC,NO IT OR OC | SXTRT1 | GQ-B5 |
| | PC AND IT OR OC | SXTRT1 | GU-B4 |
| | NO PC | ERR21 | FB-B5 |
| 17 | IC,NO IT OR OC | EXTRT1 | GQ-B4 |
| | PC AND IT OR OC | EXTRT1 | GU-B2 |
| | NO PC | ERR21 | FB-B5 |
| 18 | TIMER | SVC18 | GU-B5 |
| | NO TIMER | ERR21 | FB-B5 |
| 19 | TIMER | SVC19 | GU-B3 |
| | NO TIMER | ERR21 | FB-B5 |
| 20 | OC RTN | SXTRT1 | GU-B4 |
| | NO OC RTN | ERR21 | FB-B5 |
| 21 | OC RTN | EXTRT1 | GU-B2 |
| | NO OC RTN | ERR21 | FB-B5 |
| 22 | MPS | SVC22 | GP-B1 |
| | NO MPS | EXT01 | FP-C1 |
| 23 | MPS | SVC23 | GP-B3 |
| | NO MPS | ERR21 | FB-B5 |
| 24 | TIMER | SVC24 | GP-B5 |
| | NO TIMER | ERR21 | FB-B5 |
| 25 | BTAM | SVC25 | FF-F2 |
| | NO BTAM | ERR21 | FB-B5 |
| 26 | STOR PROT | SVC26 | GP-H4 |
| | NO STOR PROT | EXT01 | FD-C1 |
| 27 | BTAM | SVC27 | FF-F2 |
| | NO BTAM | EXT01 | |

Chart GJ. SGSVC Macro-- SVC's 1, 5, 12, and 13; Refer to Supervisor, Chart 14

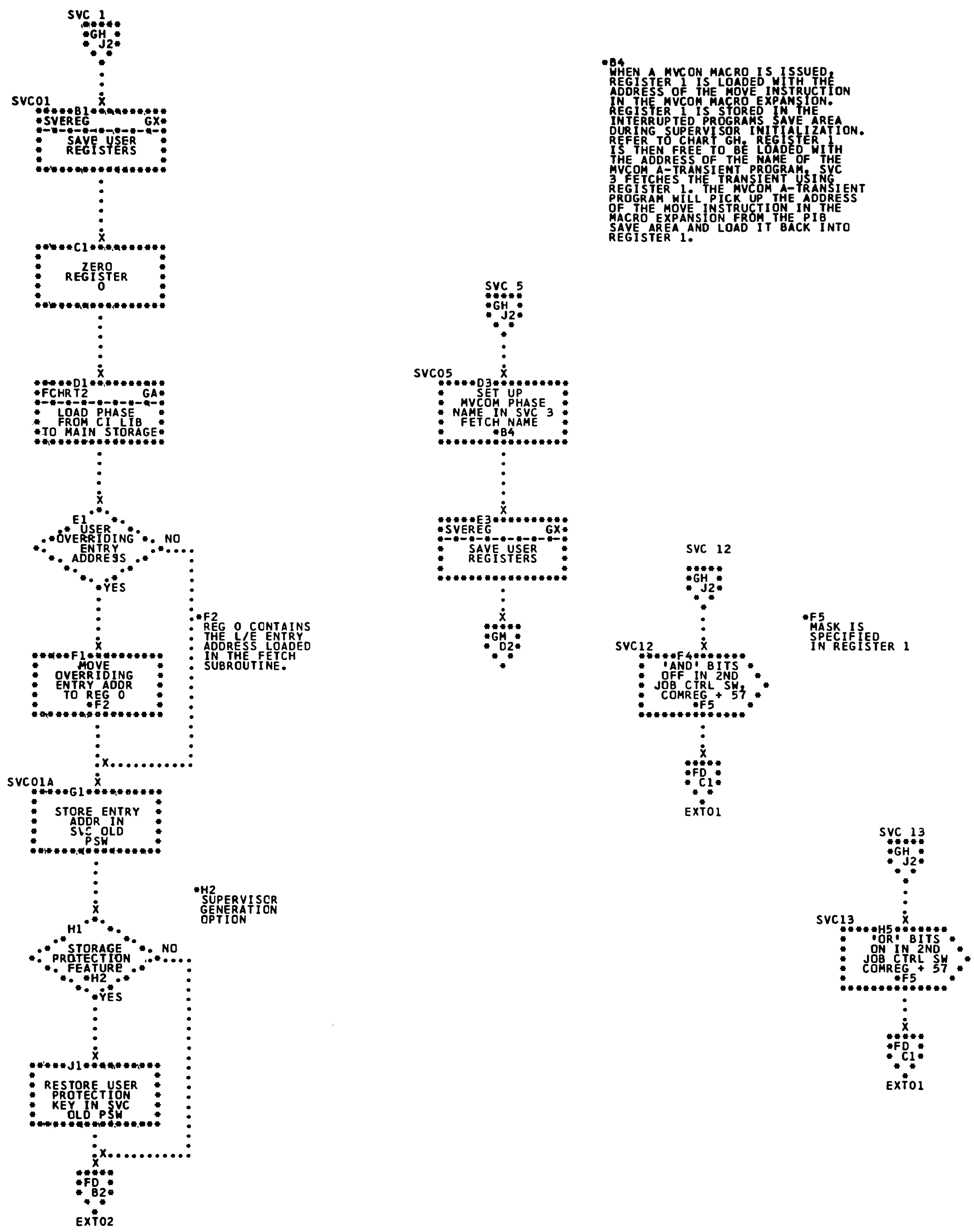


Chart GK. SGSVC Macro-- SVC's 2 and 11 with MPS Option; Refer to Supervisor, Chart 14

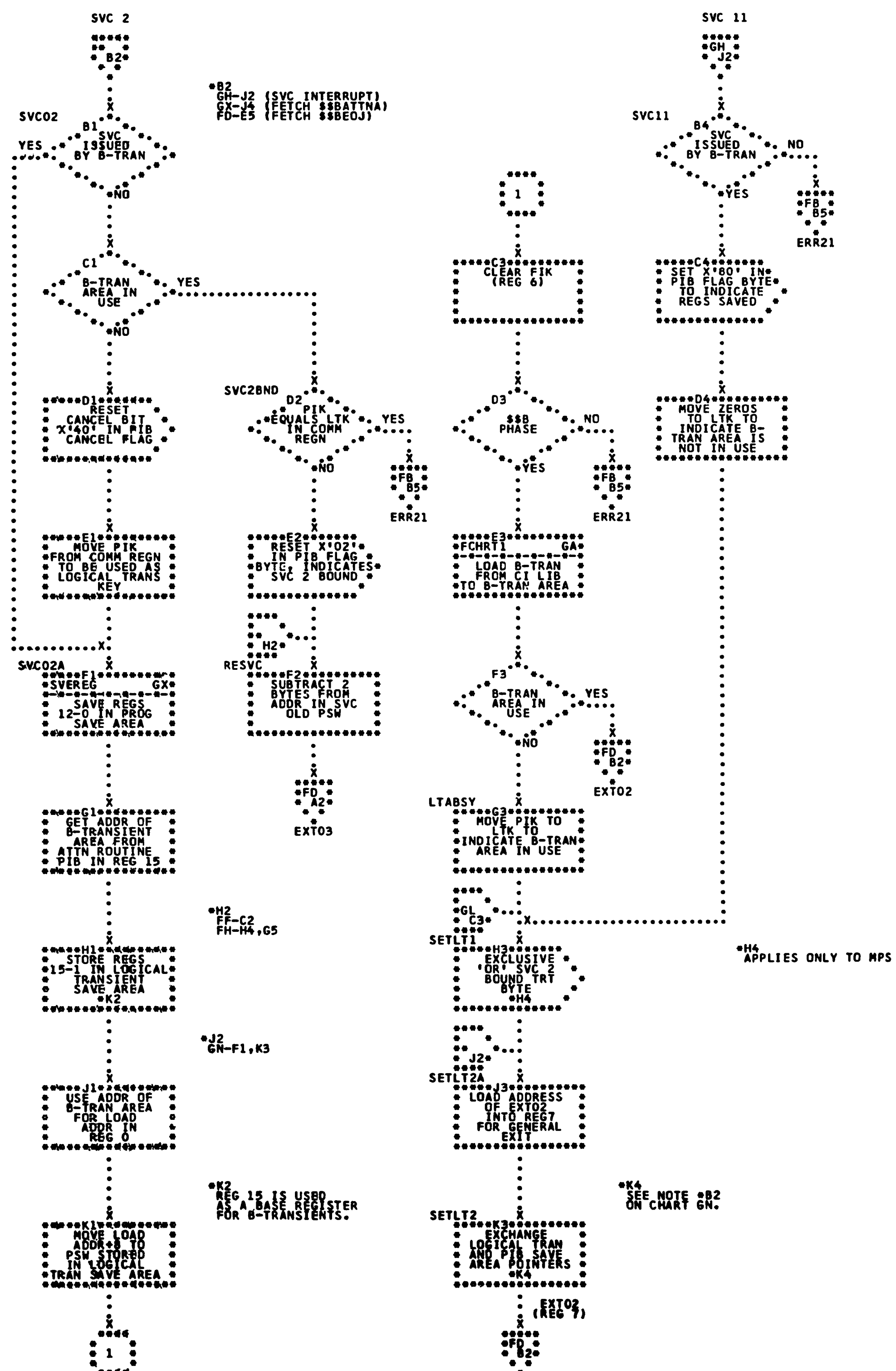


Chart GL. SGSVC Macro-- SVC's 2 and 11 with Batch Only Option; Refer to Supervisor, Chart 14

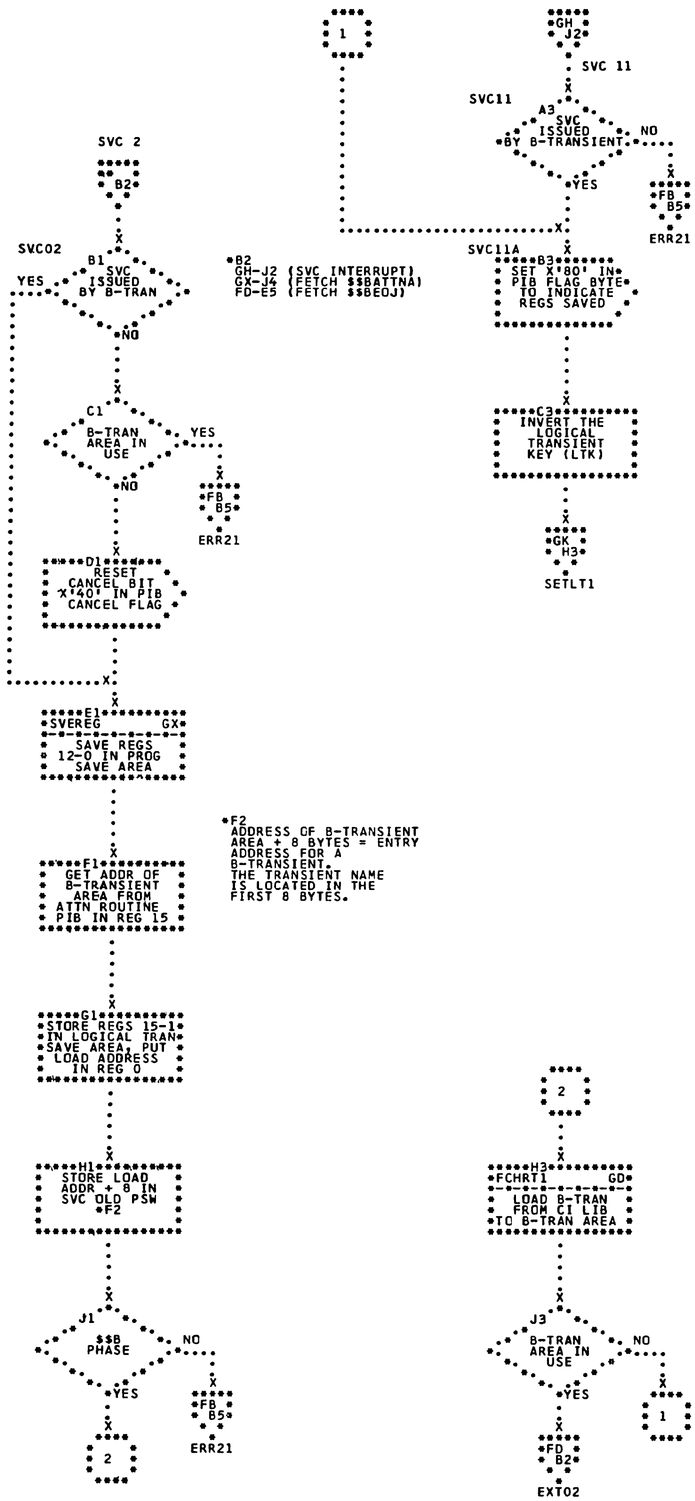


Chart GM. SGSVC Macro-- SVC's 3, 4, and 7; Refer to Supervisor, Chart 14

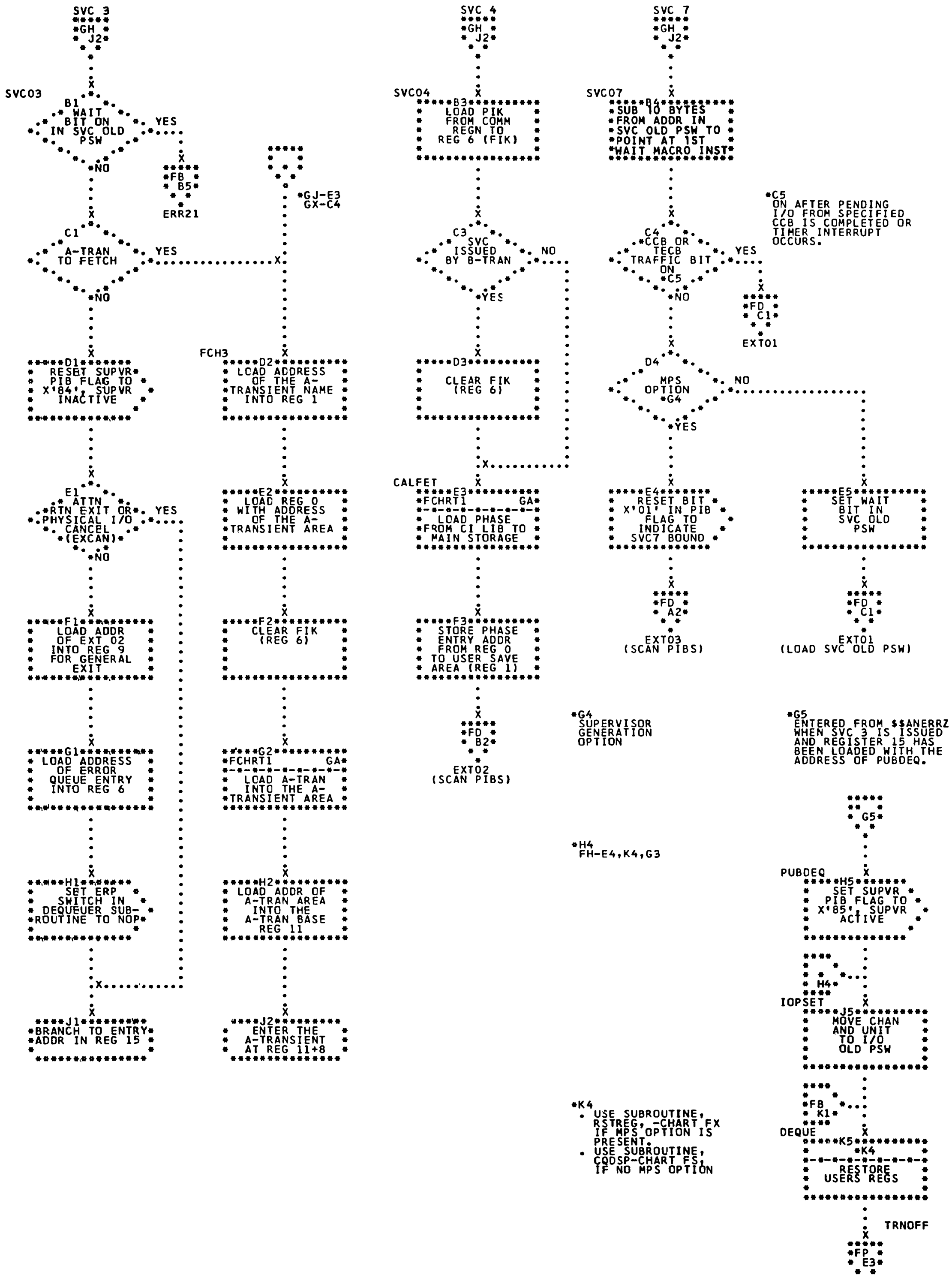


Chart GN. SGSVC Macro--SVC's 8, 9, and 10; Refer to Supervisor, Chart 14

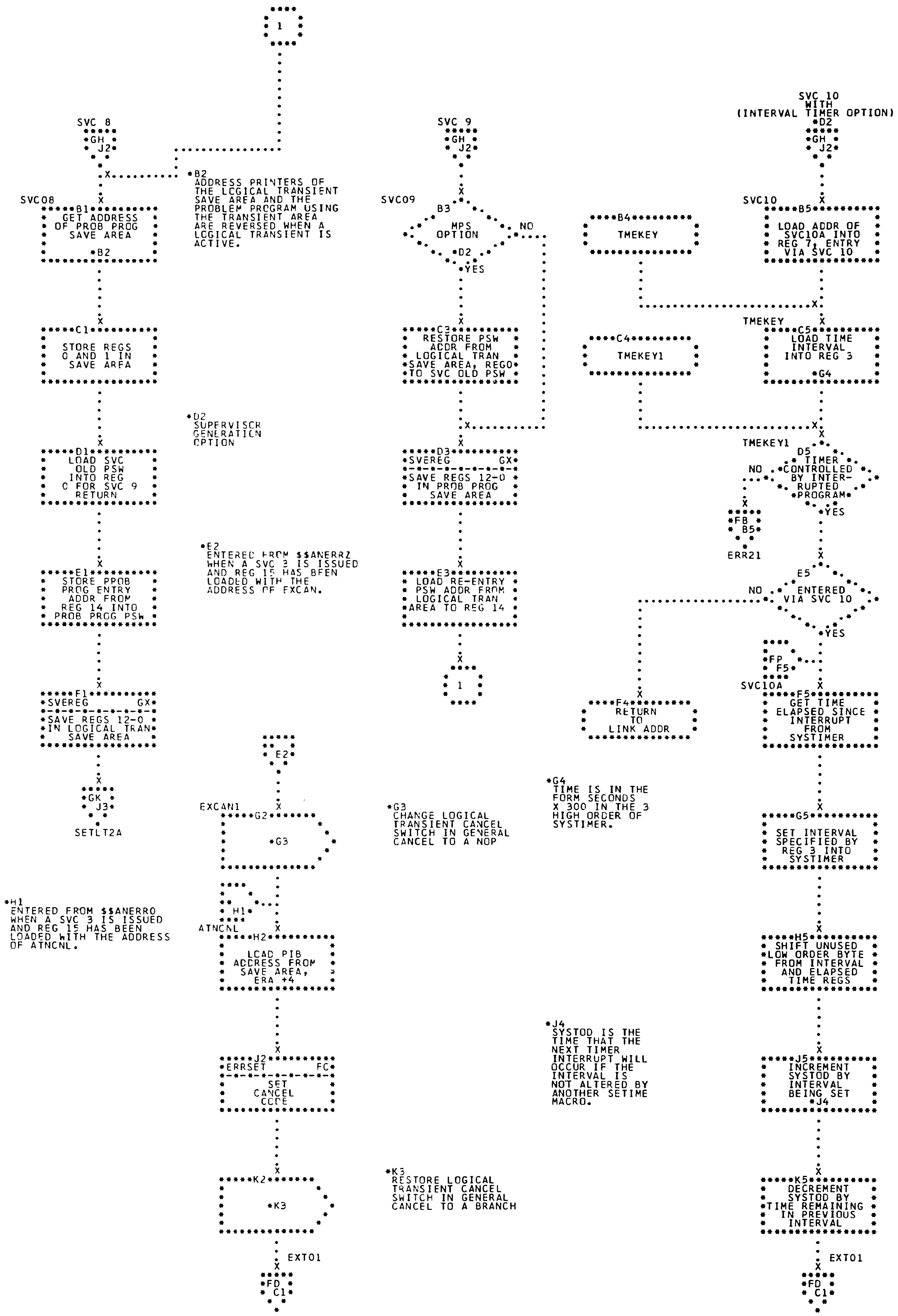


Chart GP. SGSVC Macro--SVC's 22, 23, 24, and 26; Refer to Supervisor, Chart 14

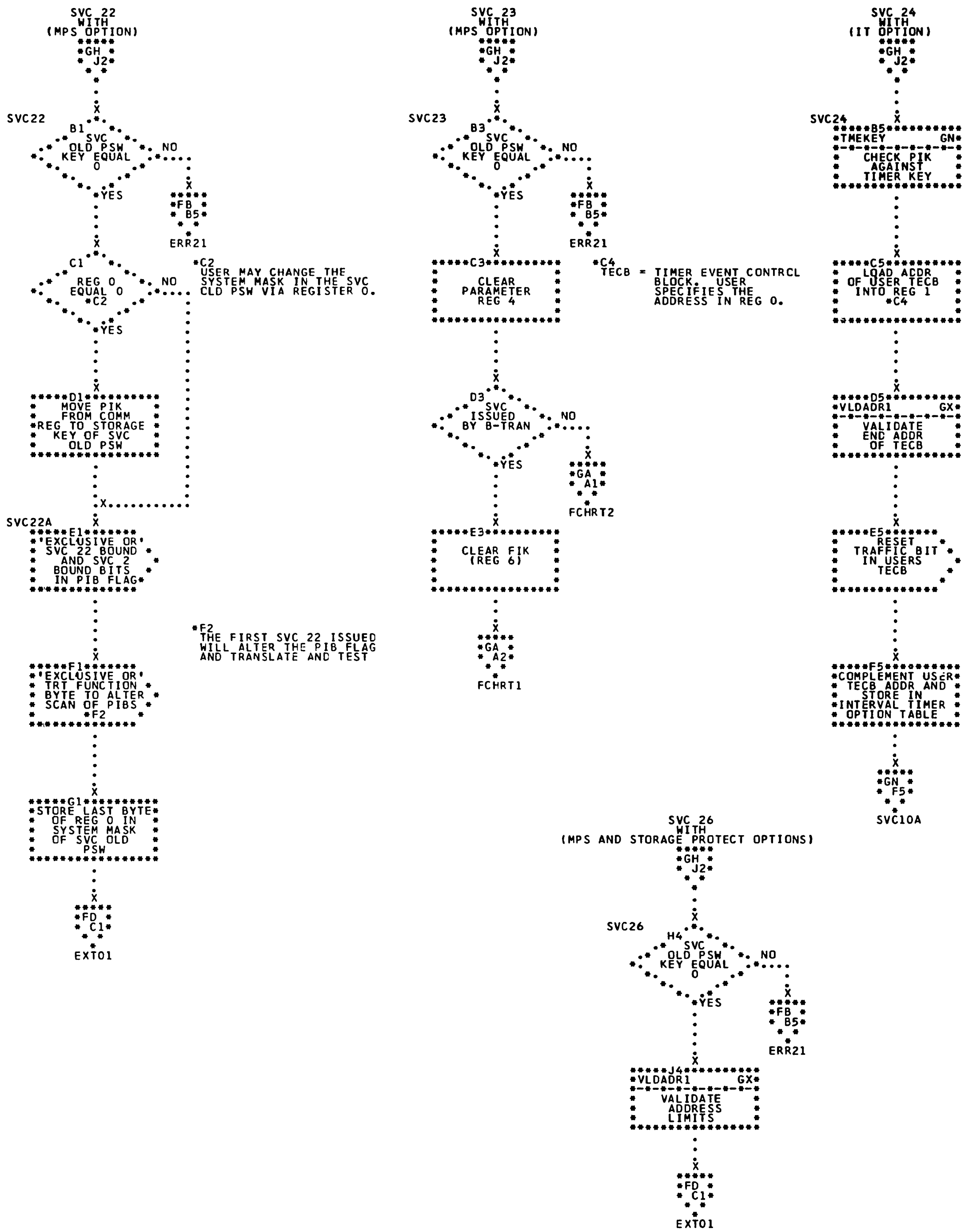


Chart GQ. SGSVC Macro-- Program Check Interrupt, SVC's 17 and 18; Refer to Supervisor, Chart 14

OPTIONS USER PC ROUTINE WITHOUT (STORAGE PROTECT, INTERVAL TIMER, AND USER OC ROUTINE).

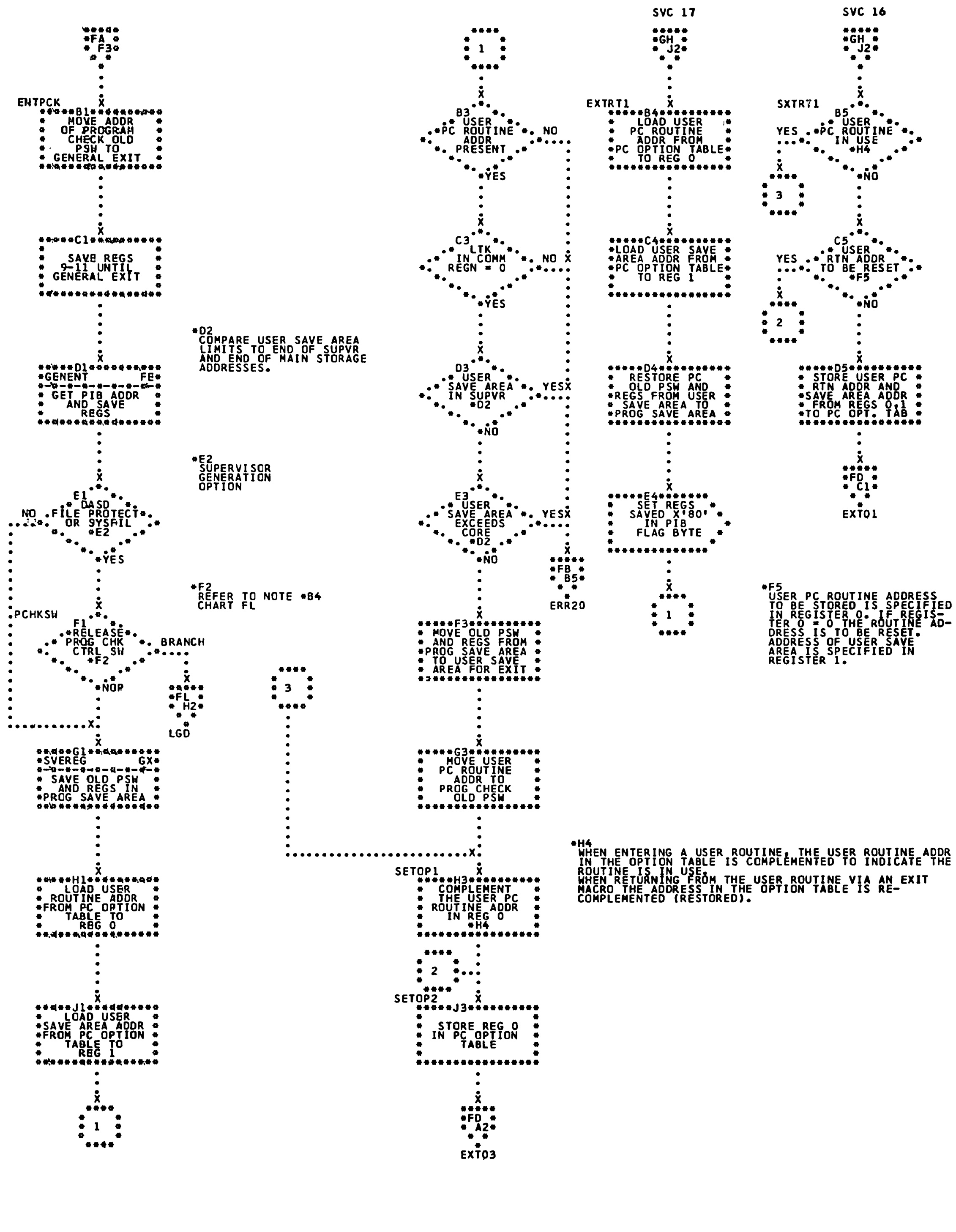


Chart GR. SGSVC Macro-- Program Check Interrupt; Refer to Supervisor, Chart 16

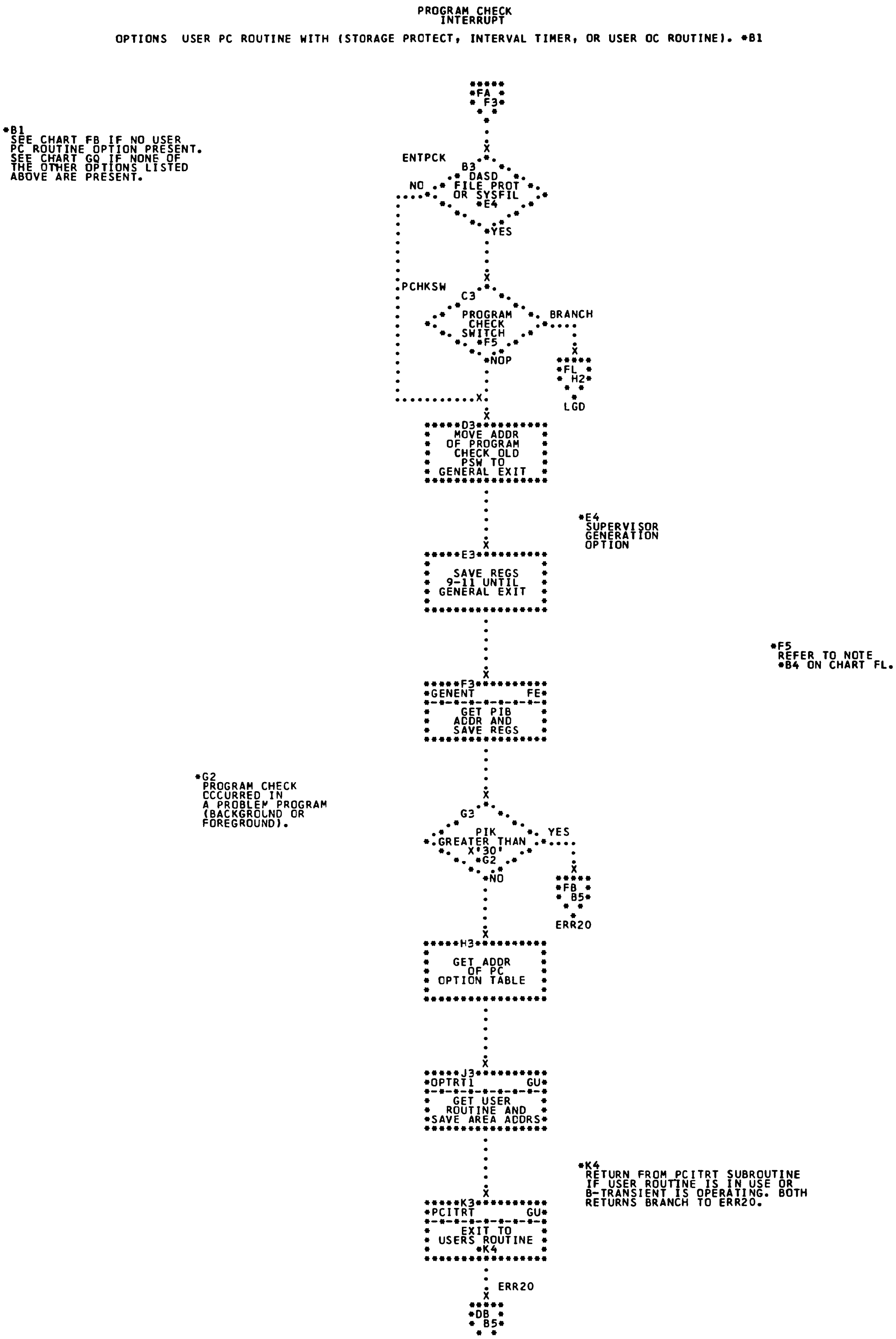


Chart GS. SGSVC Macro-- External Interrupt with User OC or IT Routines; Refer to Supervisor, Chart 16

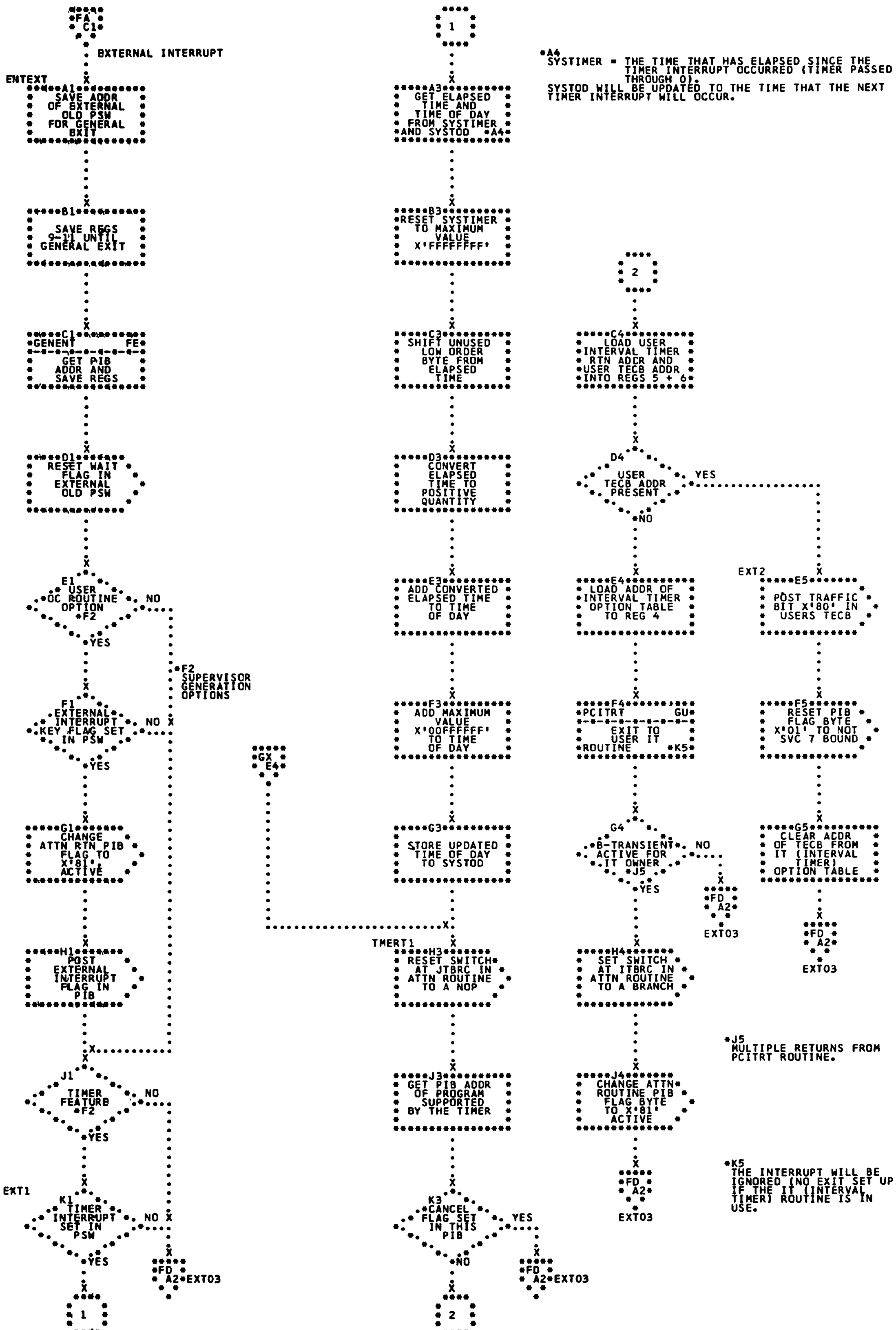


Chart GT. SGSVC Macro-- External Interrupt Subroutines; Refer to Supervisor, Chart 16

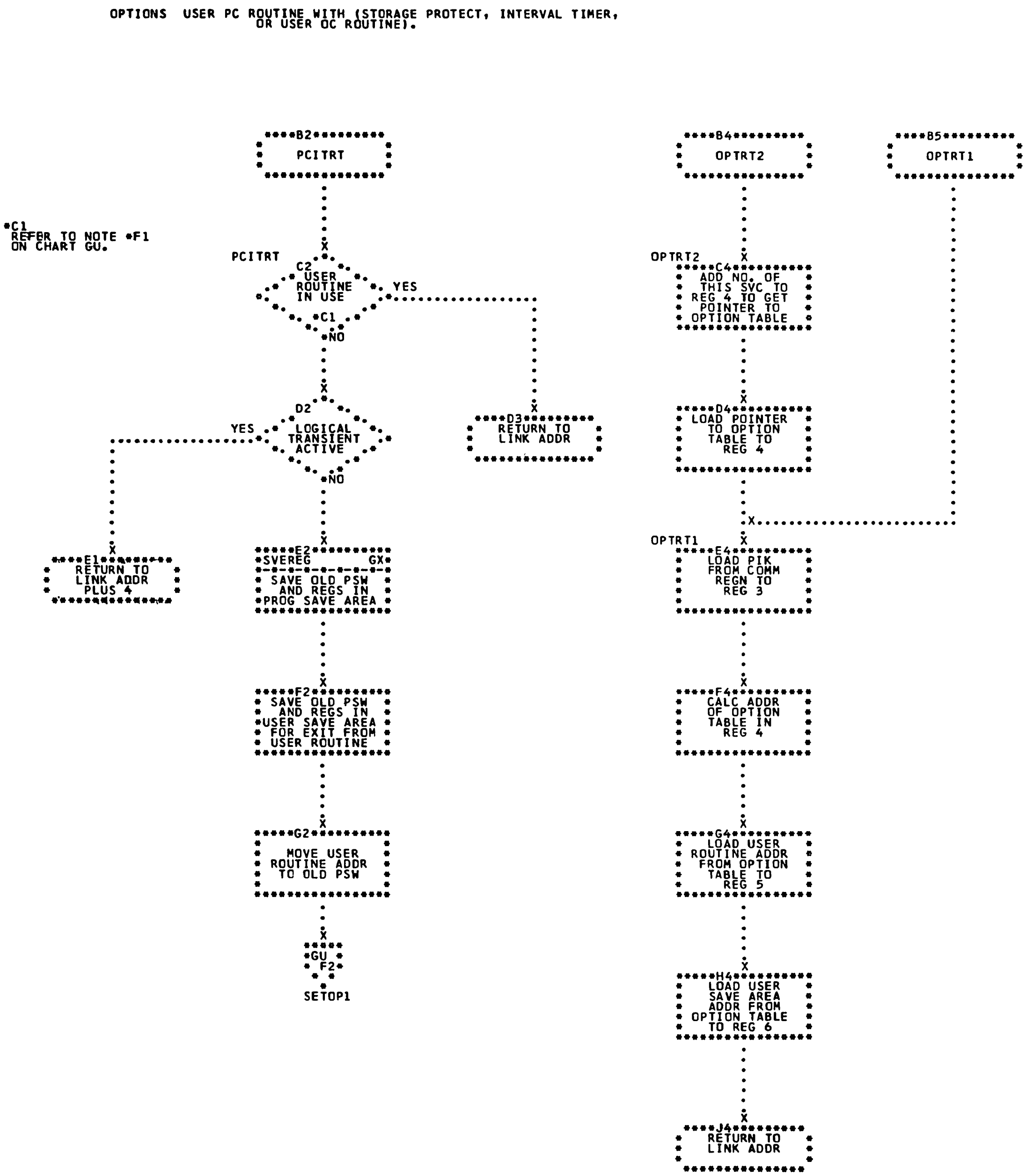


Chart GU. SGSVC Macro-- Program Check Interrupt; Refer to Supervisor, Chart 14

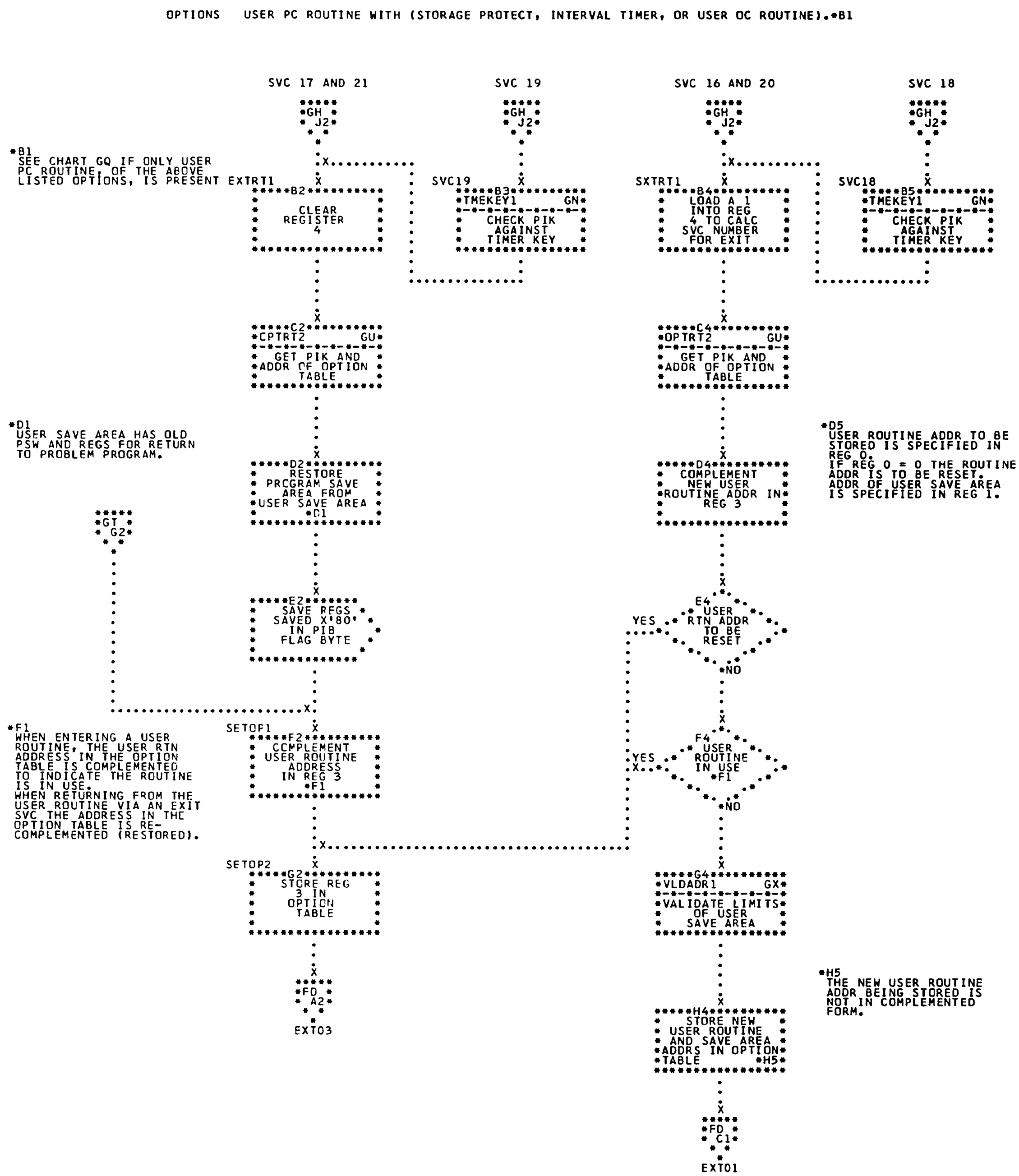


Chart GV. SGDSK Macro-Resident Disk Error Recovery (Part 1 of 2); Refer to Supervisor, Chart 17

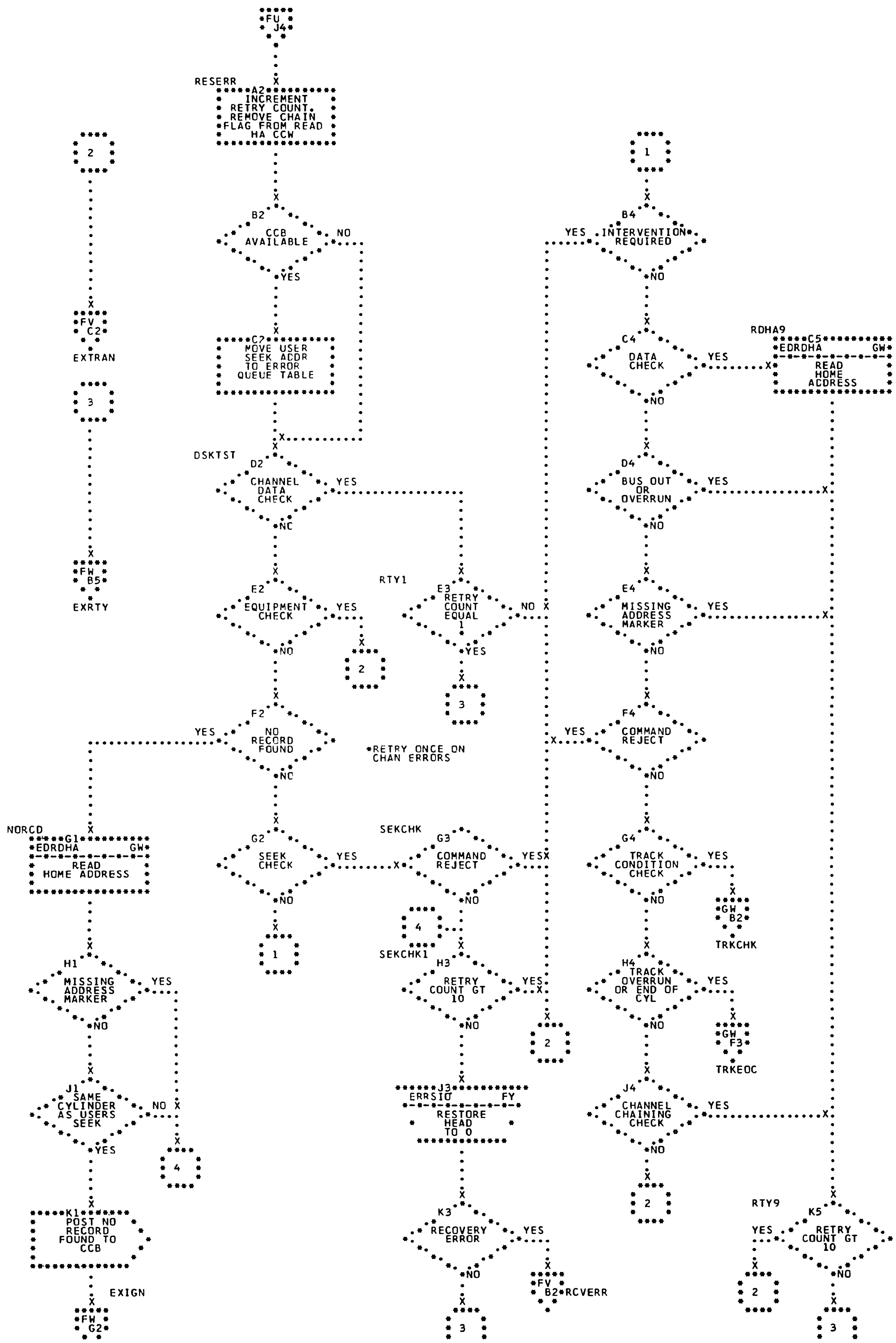


Chart GW. SGDSK Macro-- Resident Disk Error Recovery (Part 2 of 2); Refer to Supervisor, Chart 17

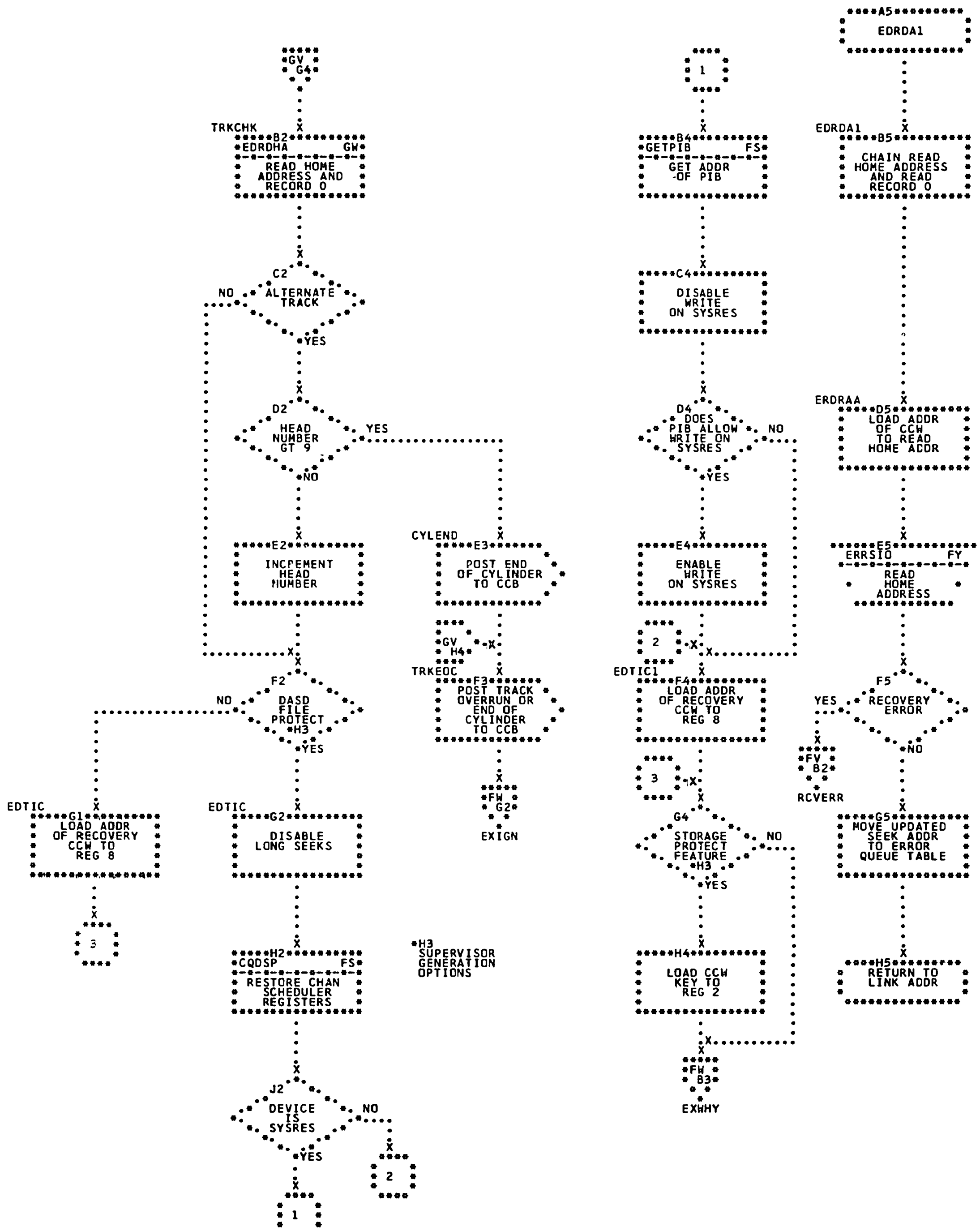


Chart GX. SGTCON Macro-- Resident Attention and SVEREG-VLDADR Subroutines; Refer to Supervisor, Chart 13

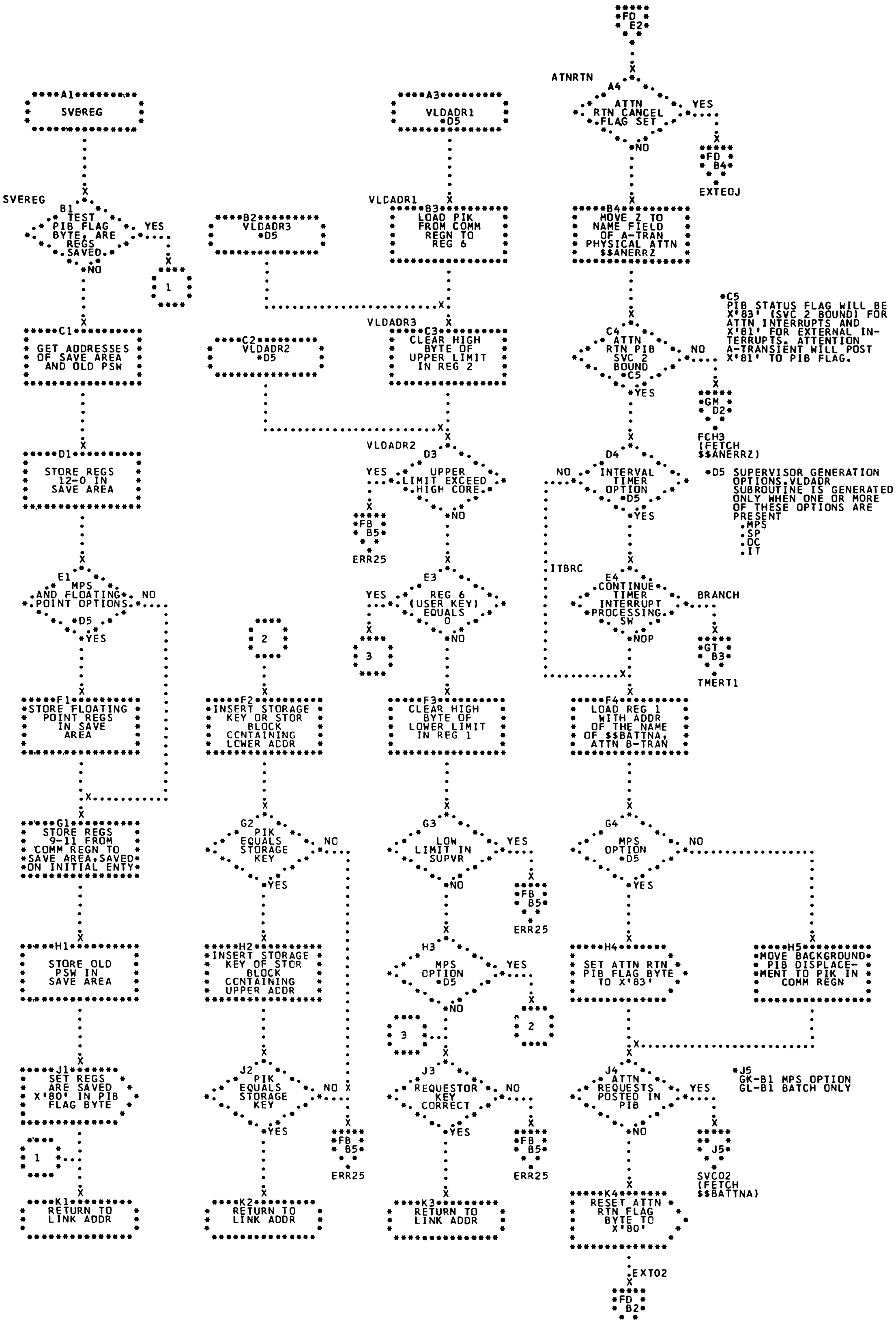


Chart GY. SEND Macro-- LTA Subroutine

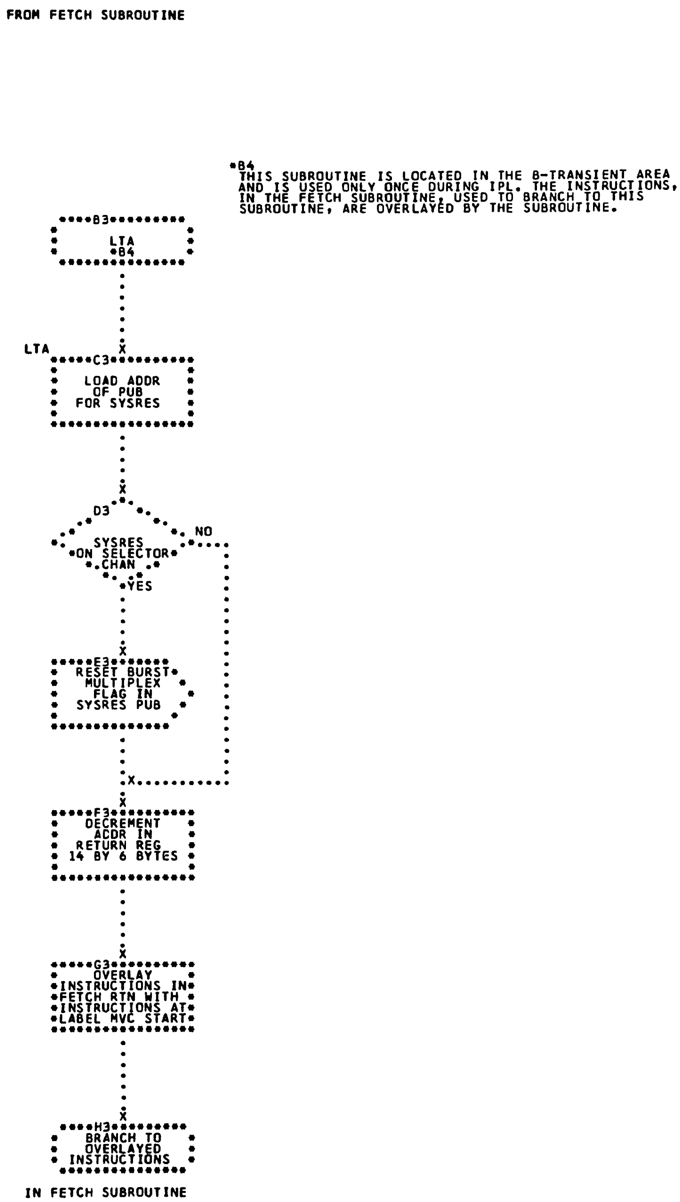


Chart HA. ERP Monitor (Part 1 of 2); ($$ANERRA); Refer to Supervisor, Chart 18

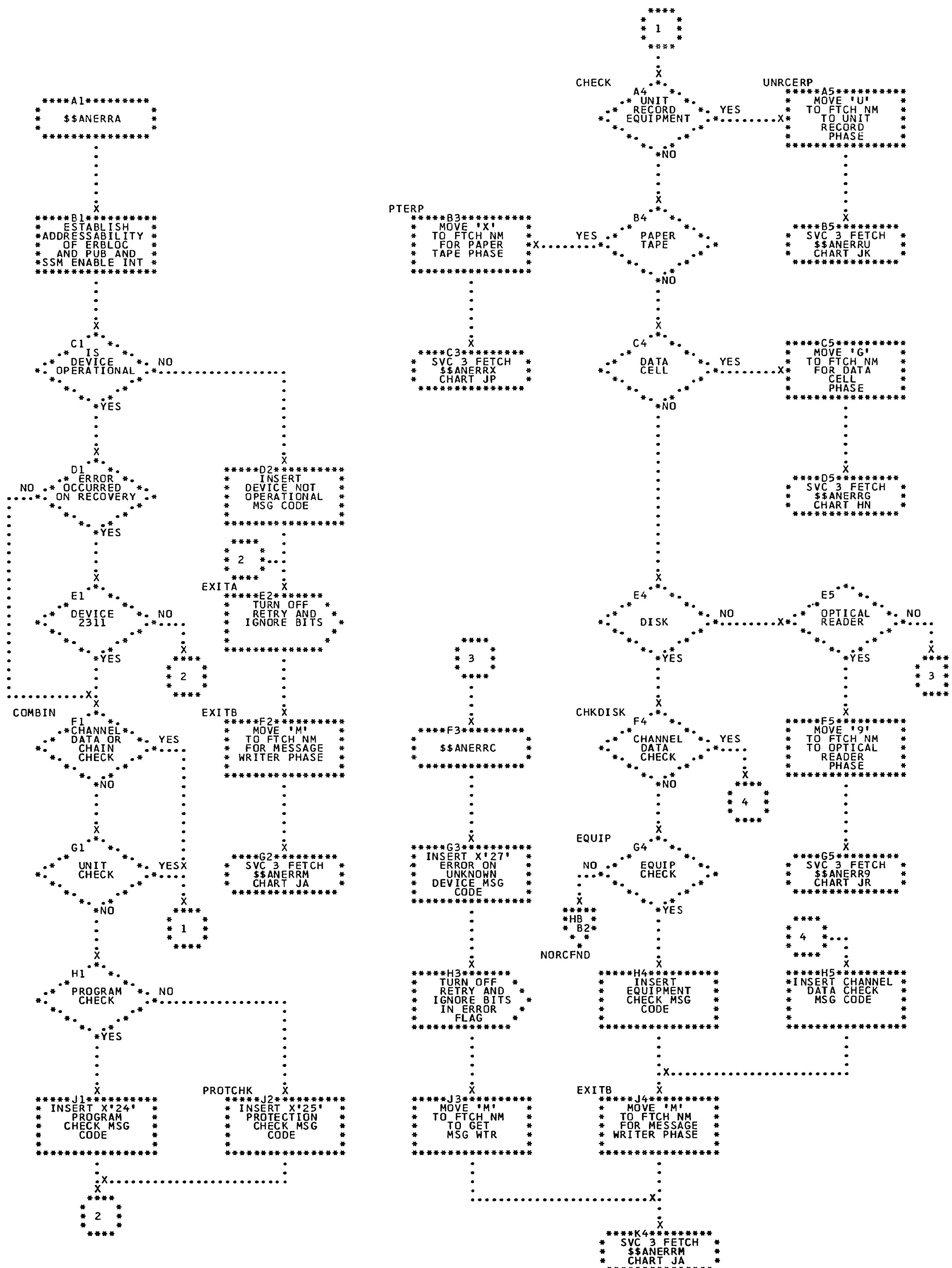


Chart HB. ERP Monitor (Part 2 of 2; $$ANERRA); Refer to Supervisor, Chart 18

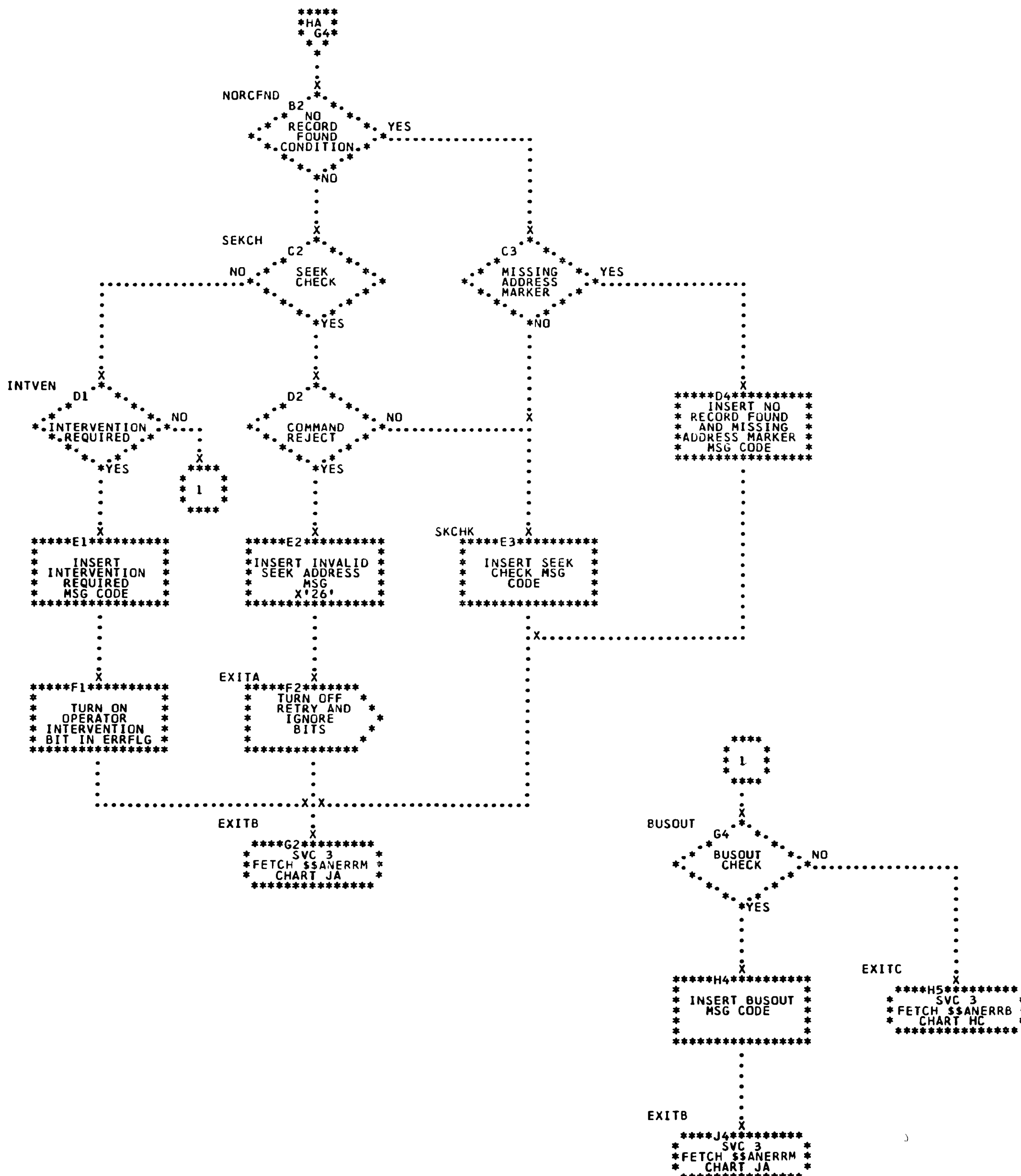


Chart HC. 2311 Nonresident ERP (Part 1 of 2) $$ANERRB;Refer to Supervisor, Chart 18

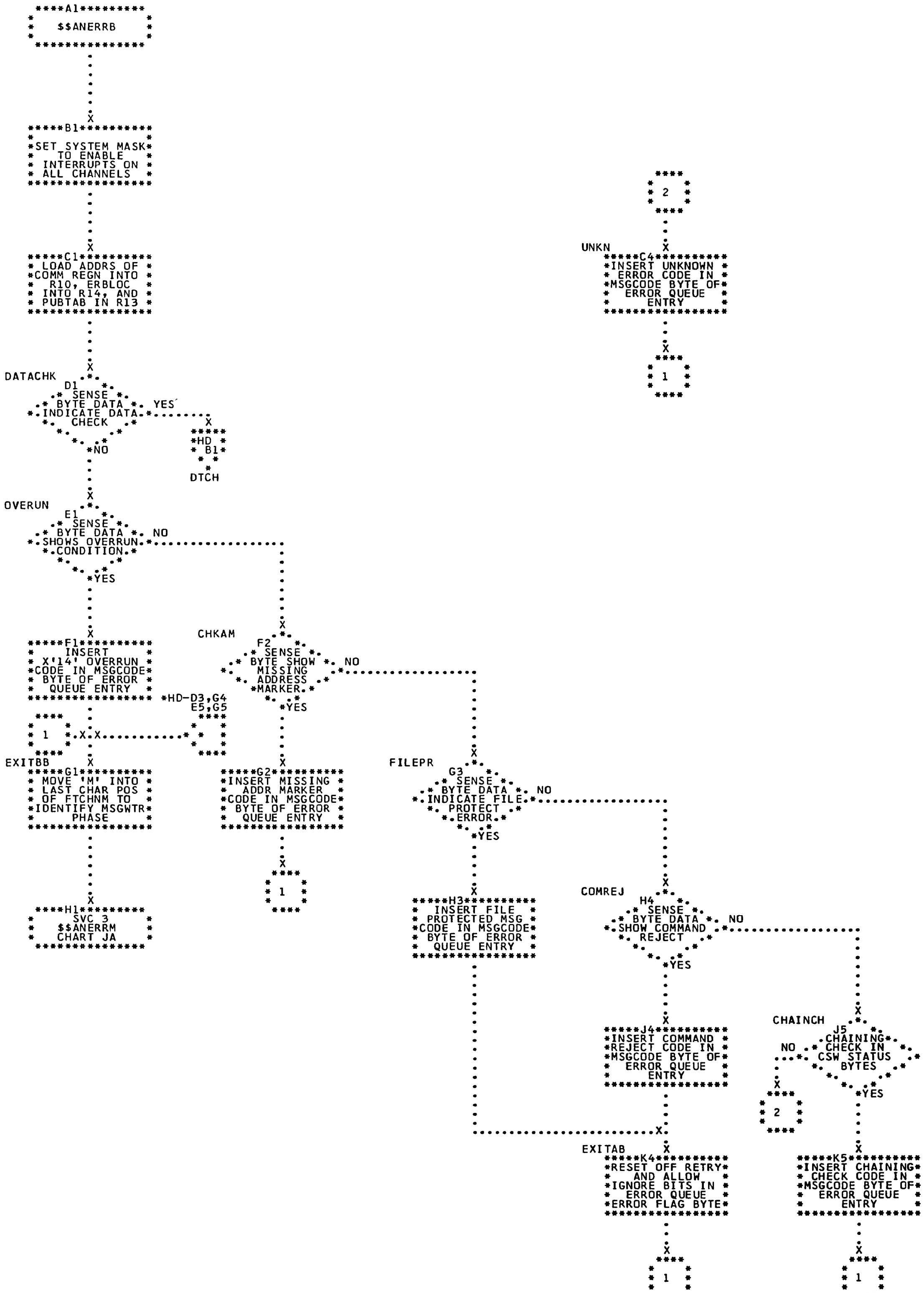


Chart HD. 2311 Nonresident ERP (Part 2 of 2) $$ANERRB;
Refer to Supervisor, Chart 18

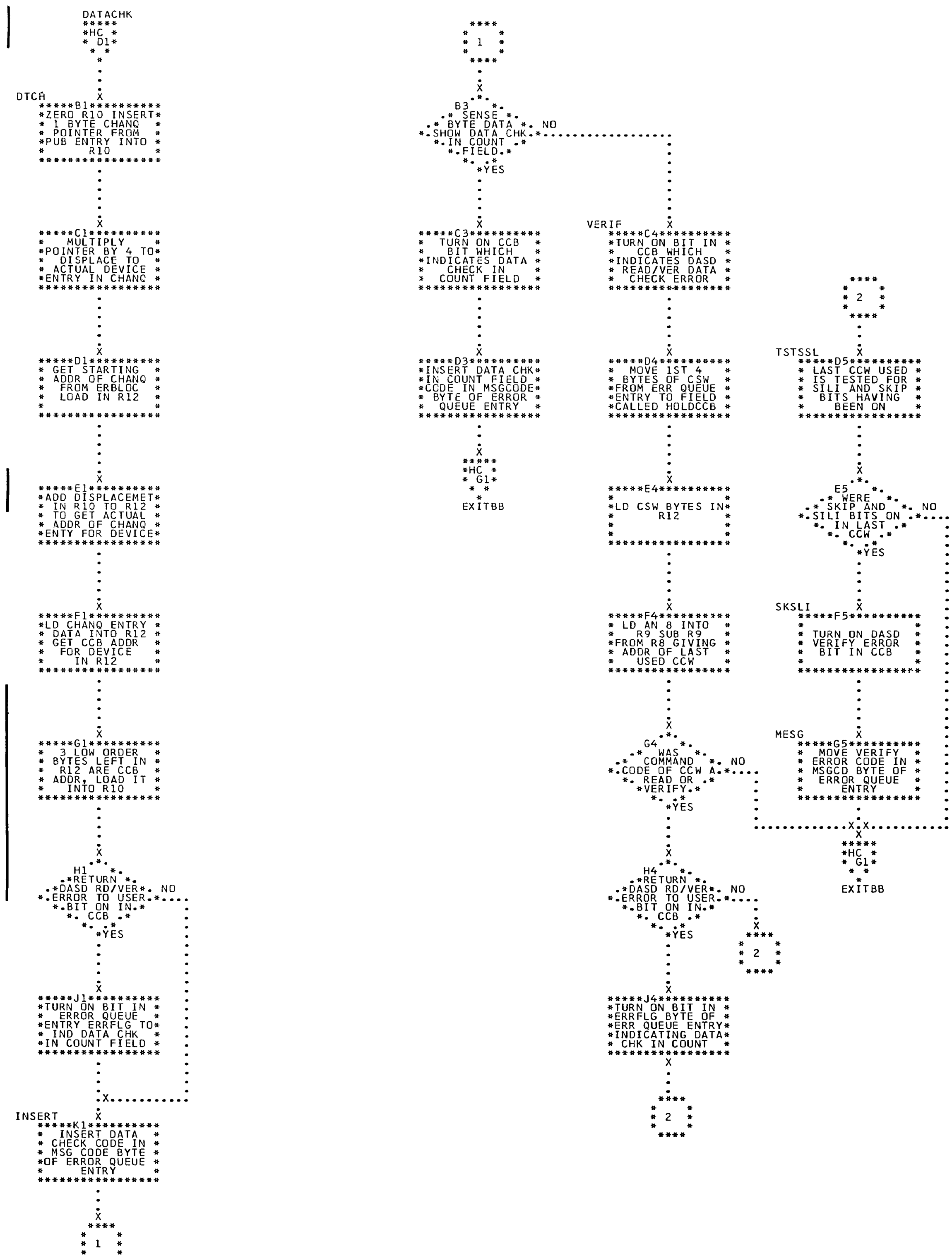


Chart HE. 2400 ERP-- Error Analysis and Selected Errors (Part 1 of 2) $$ANERRD; Refer to Supervisor, Chart 18

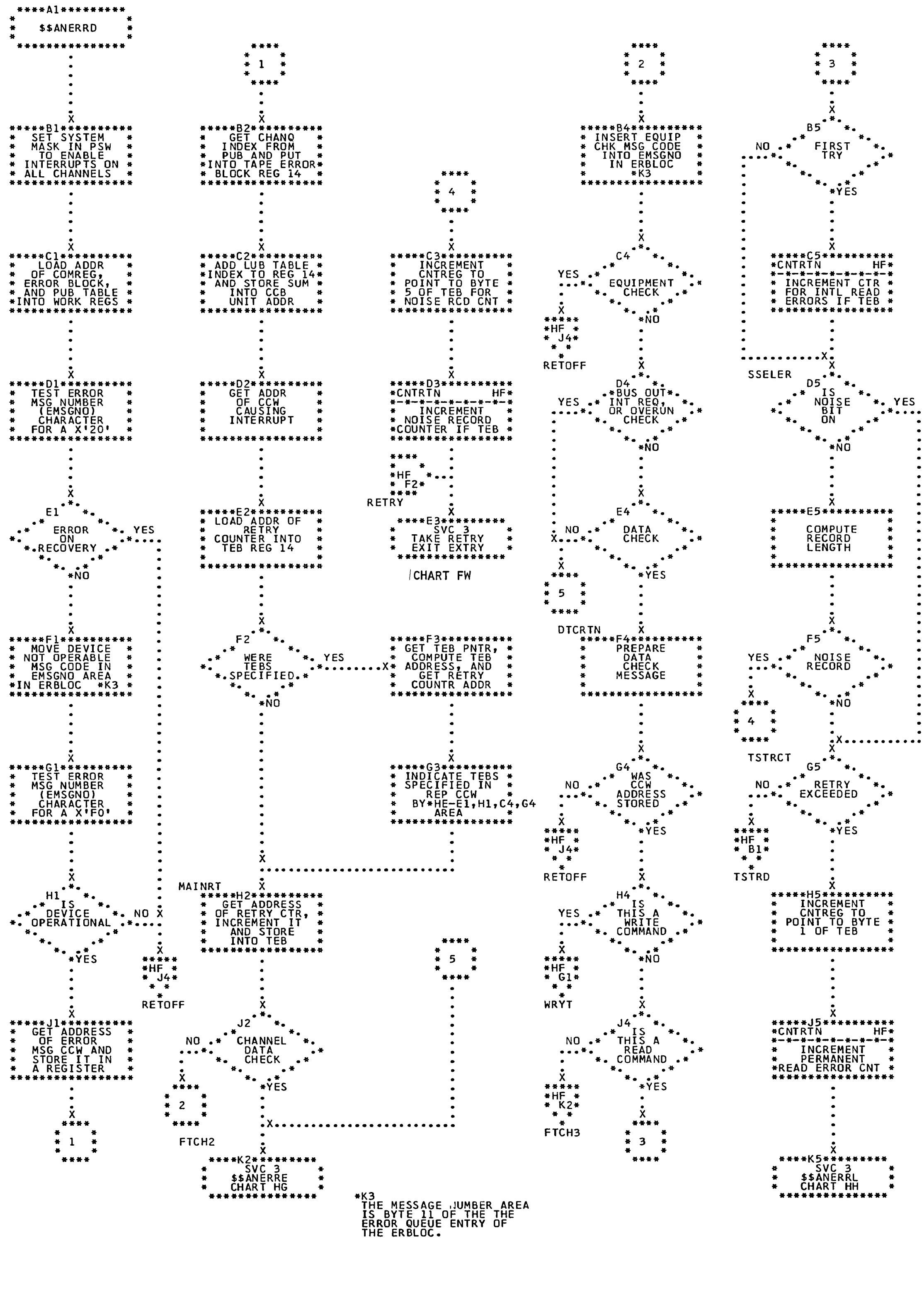


Chart HF. 2400 ERP-- Error Analysis and Selected Errors (Part 2 of 2) $$ANERRD; Refer to Supervisor, Chart 18

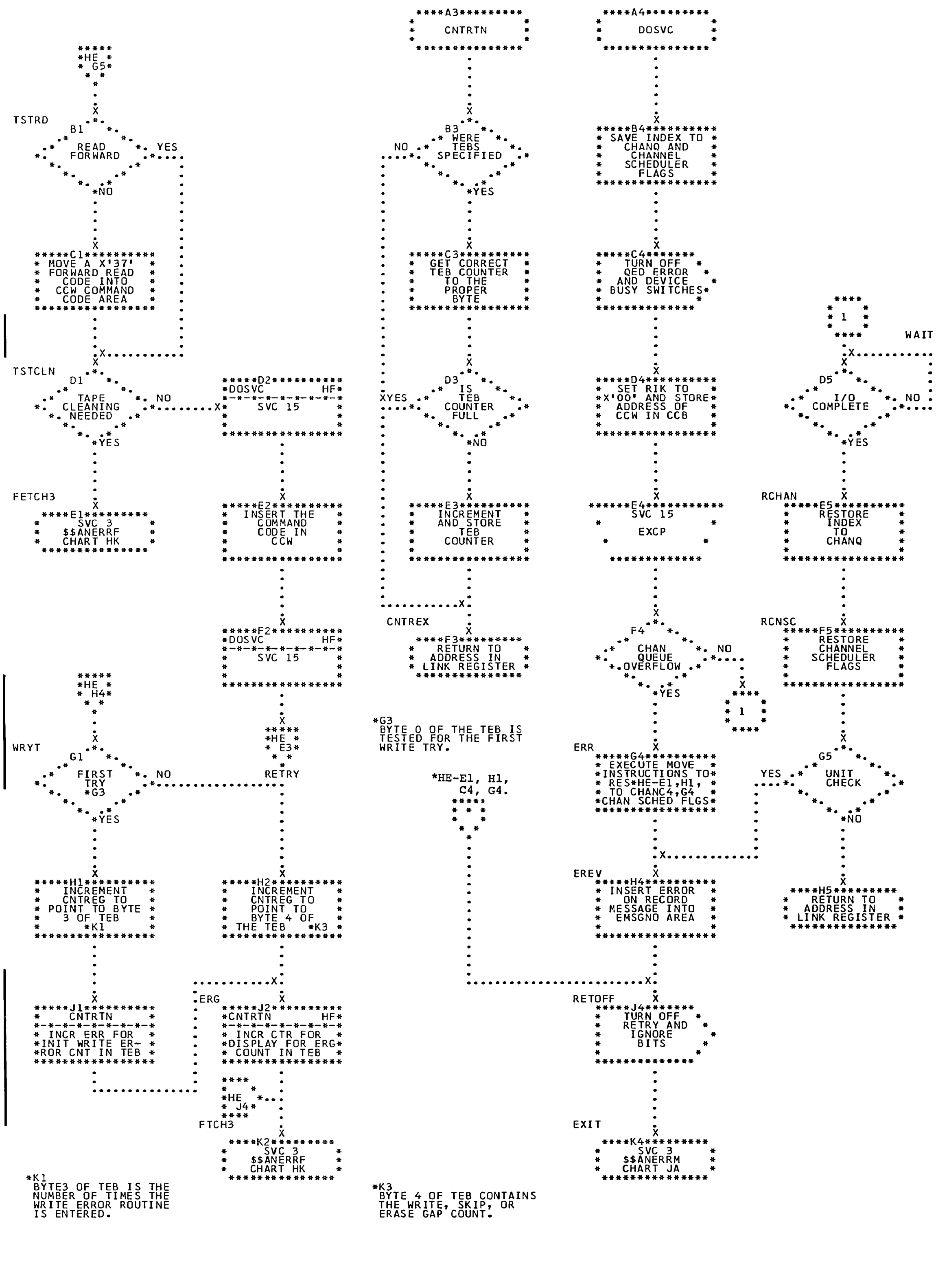


Chart HG. 2400 ERP Selected Errors (Part 1 of 3) $$ANERRE; Refer to Supervisor, Chart 18

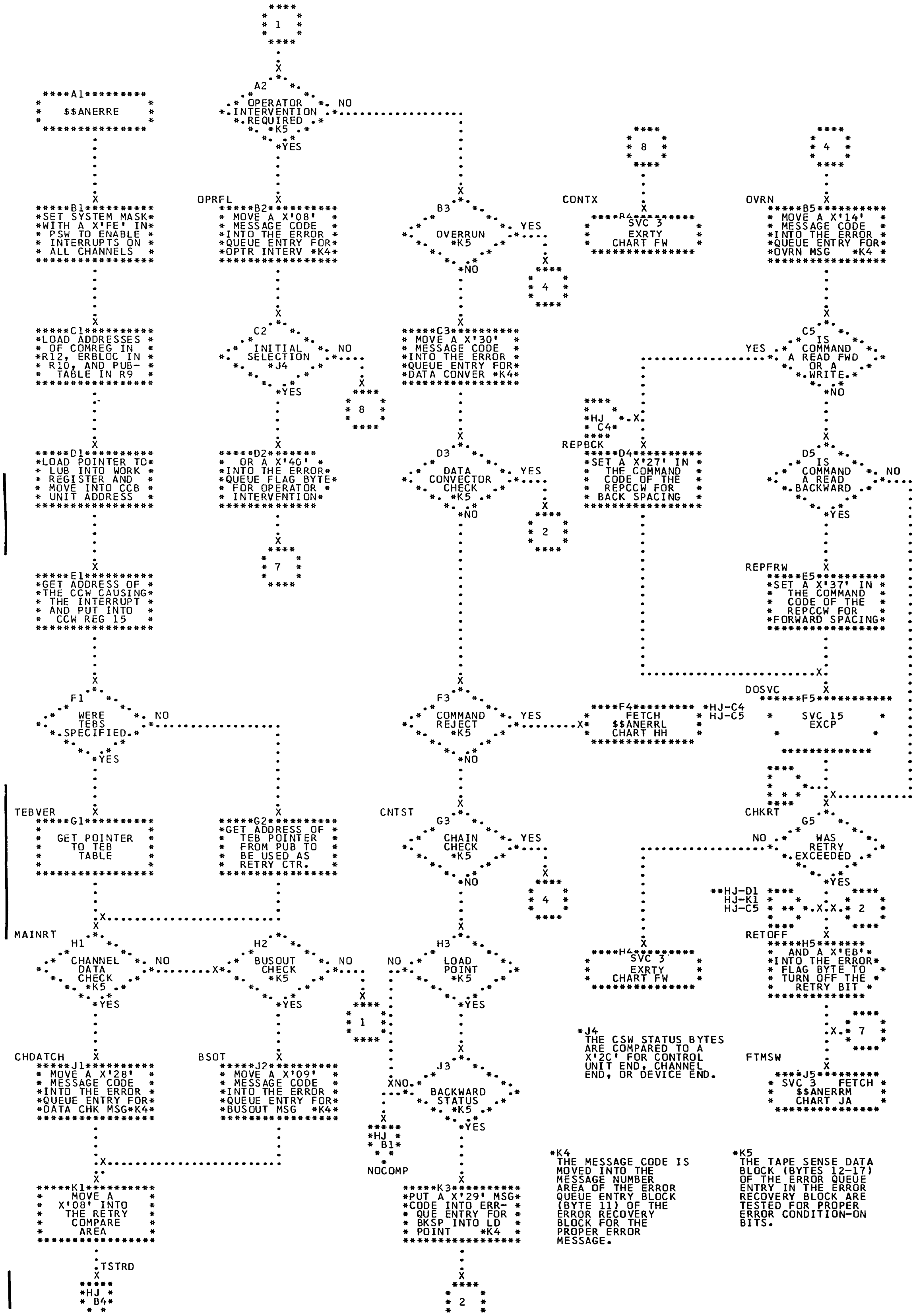


Chart HH. 2400 ERP Selected Errors (part 2 of 3) $$ANERRE; Refer to Supervisor, Chart 18

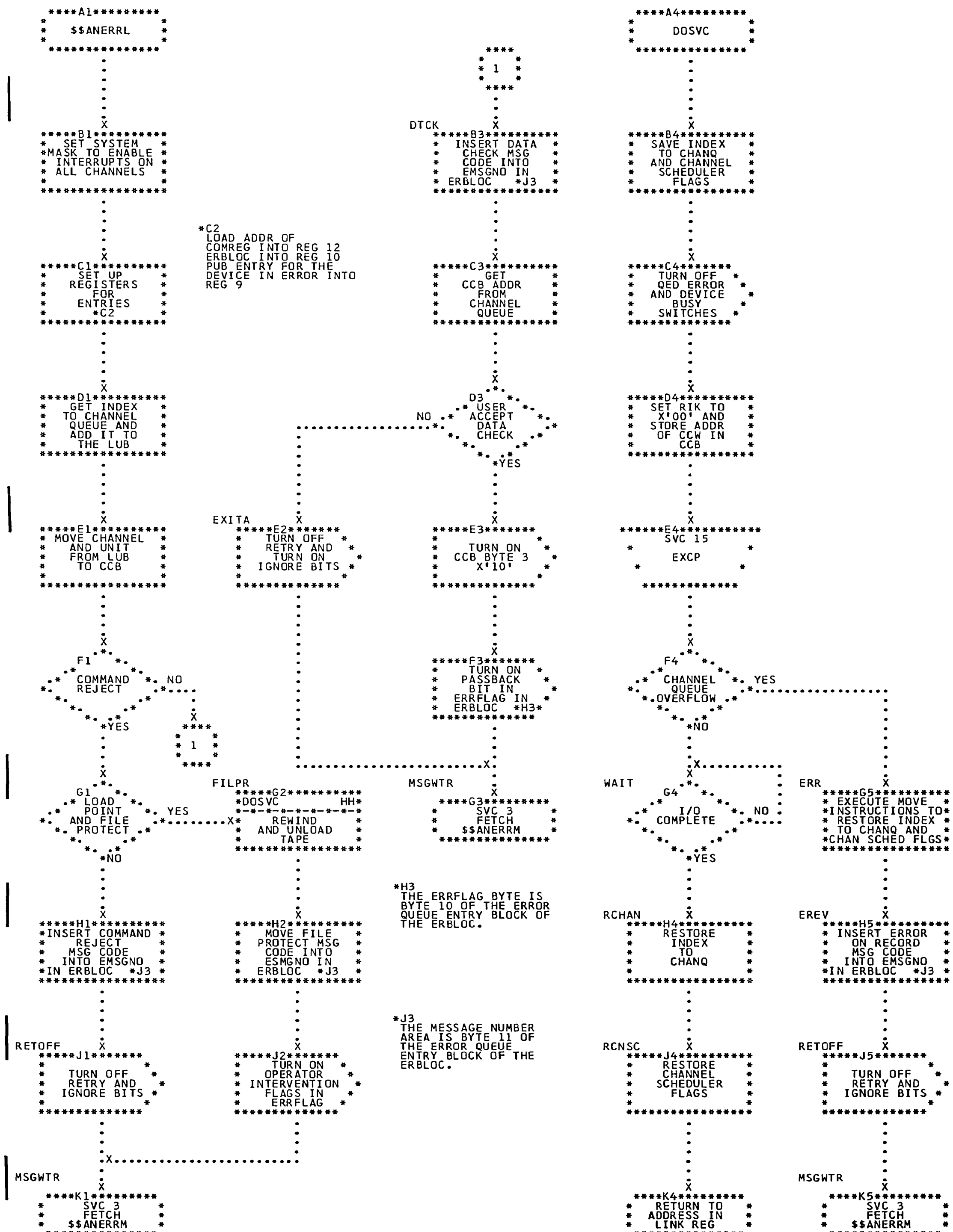


Chart HJ. 2400 ERP Selected Errors (Part 3 of 3) $$ANERRE; Refer to Supervisor, Chart 18

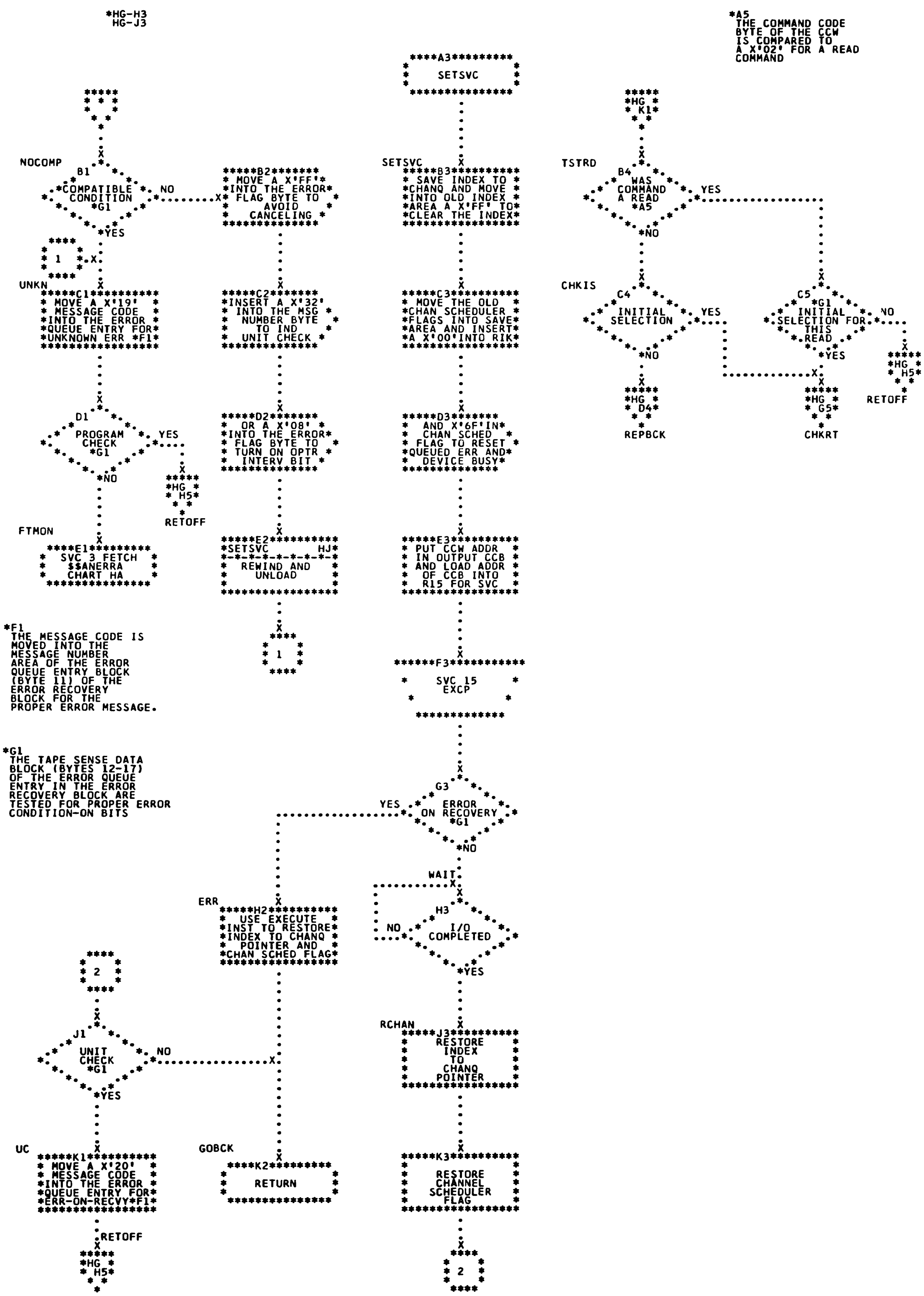


Chart HK. 2400 ERP Data Check (Part 1 of 3) $$ANERRF; Refer to Supervisor, Chart 18

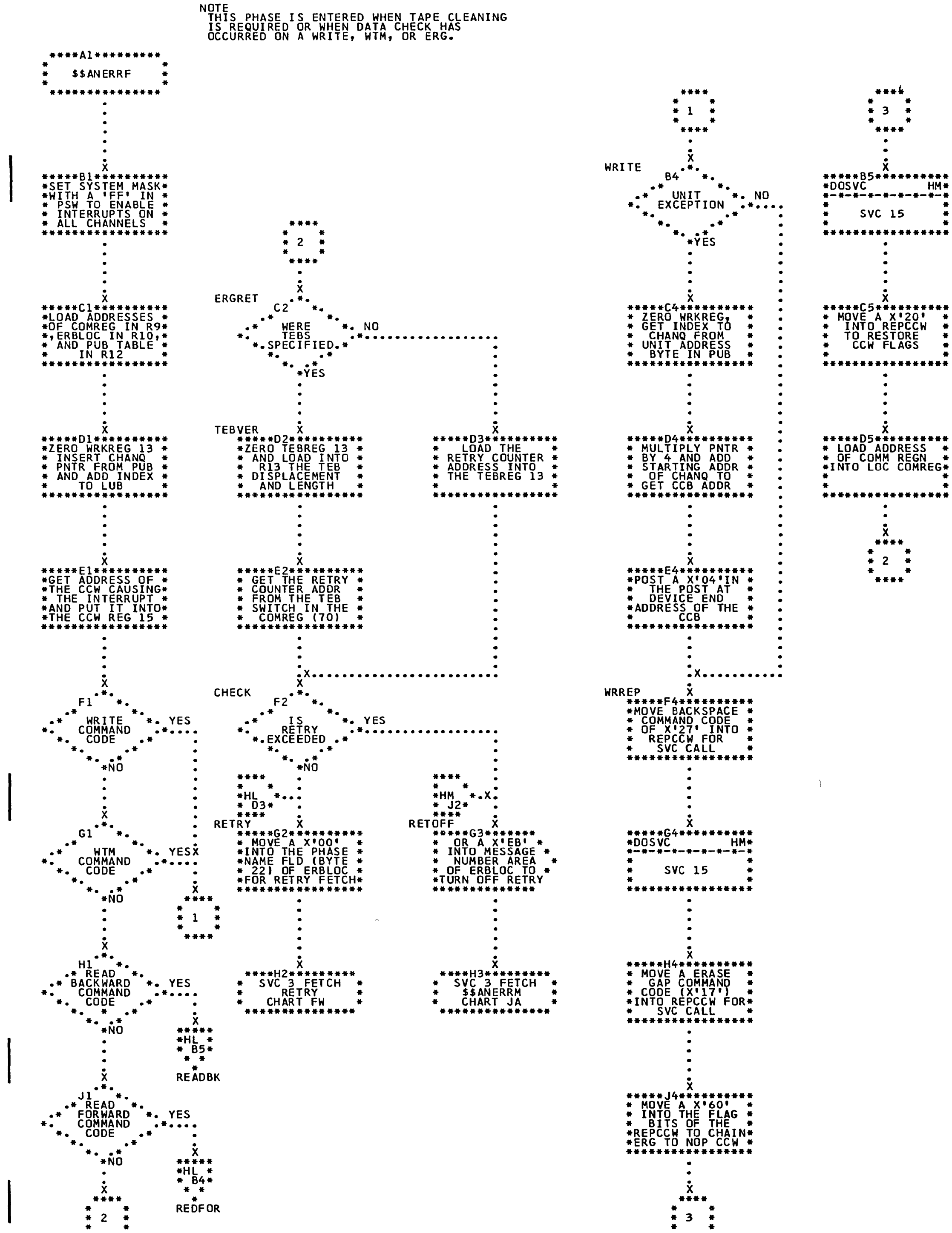
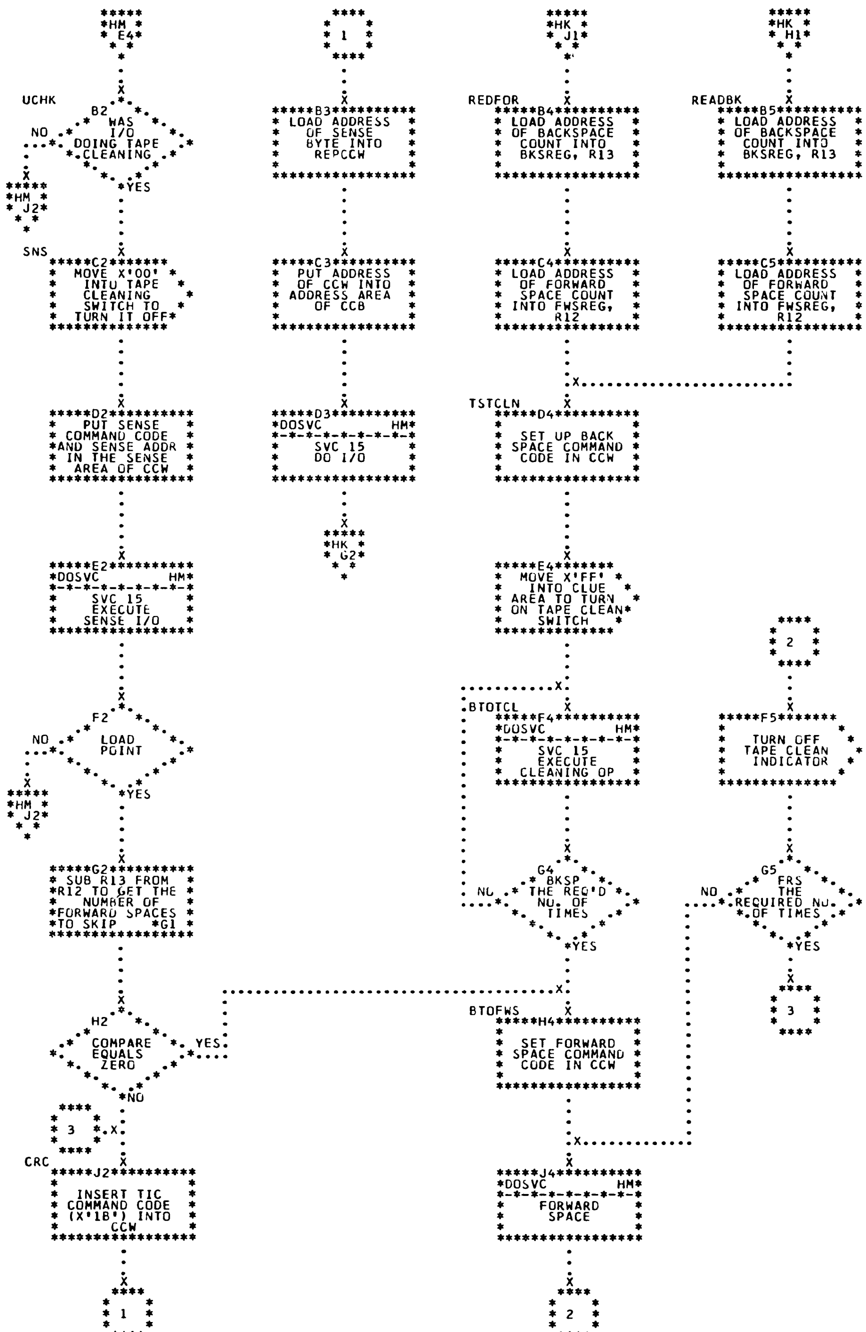


Chart HL. 2400 ERP Data Check (Part 2 of 3) $$ANERRF; Refer to Supervisor, Chart 18



*G1 THE NUMBER OF FORWARD SKIPS ARE COMPARED TO THE NUMBER OF SKIPS ISSUED MINUS THE NUMBER OF BACKWARD SKIPS. IF THE RESULT IS NOT ZERO THE BRANCH TO THE FORWARD SKIP ROUTINE IS TAKEN.

Chart HM. 2400 ERP Data Check (Part 3 of 3) $$ANERRF; Refer to Supervisor, Chart 18

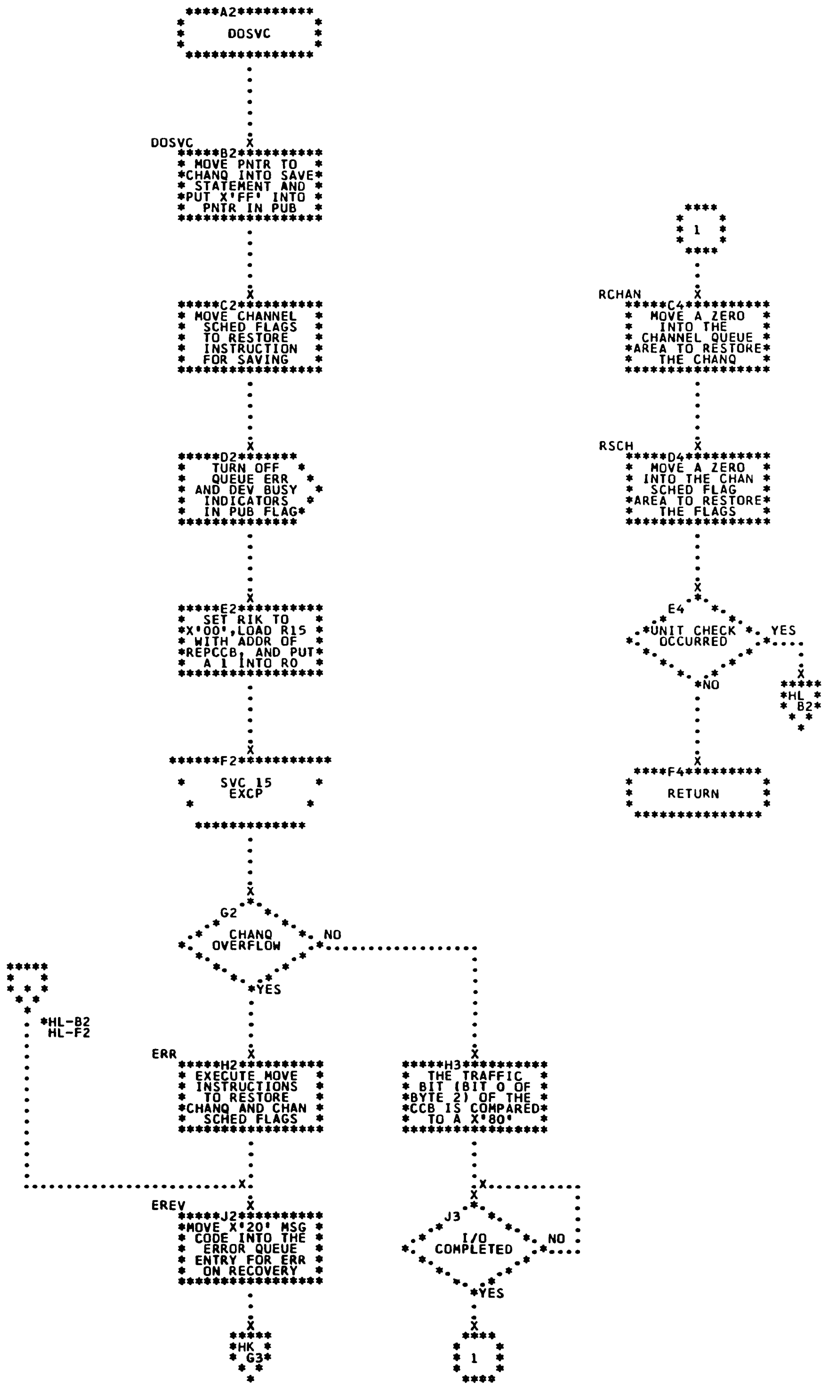


Chart HN. 2321 ERP Error Analysis (Part 1 of 3) $$ANERRG;
Refer to Supervisor Chart 18

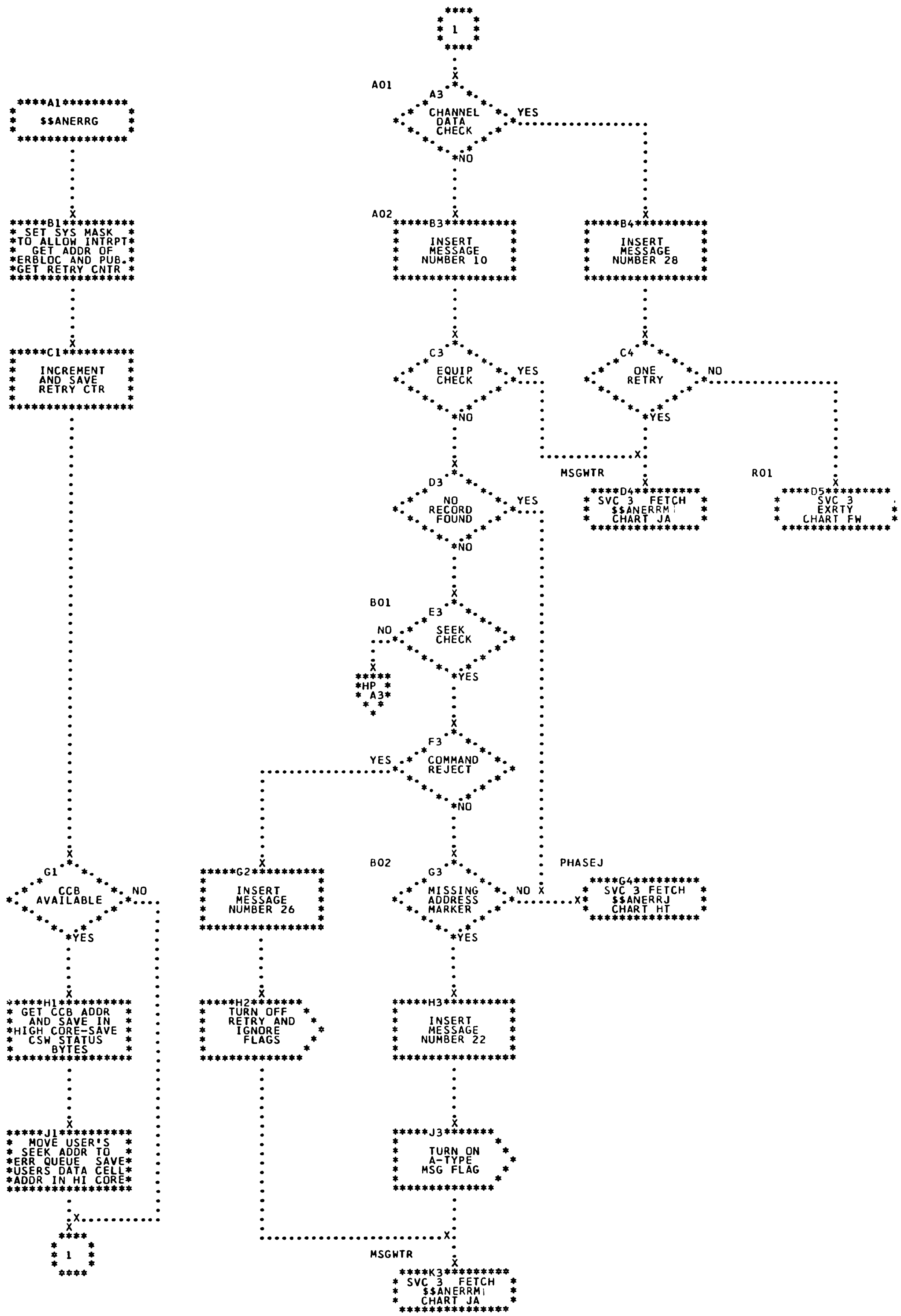


Chart HP. 2321 ERP Error Analysis (Part 2 of 3) $$ANERRG;
Refer to Supervisor, Chart 18

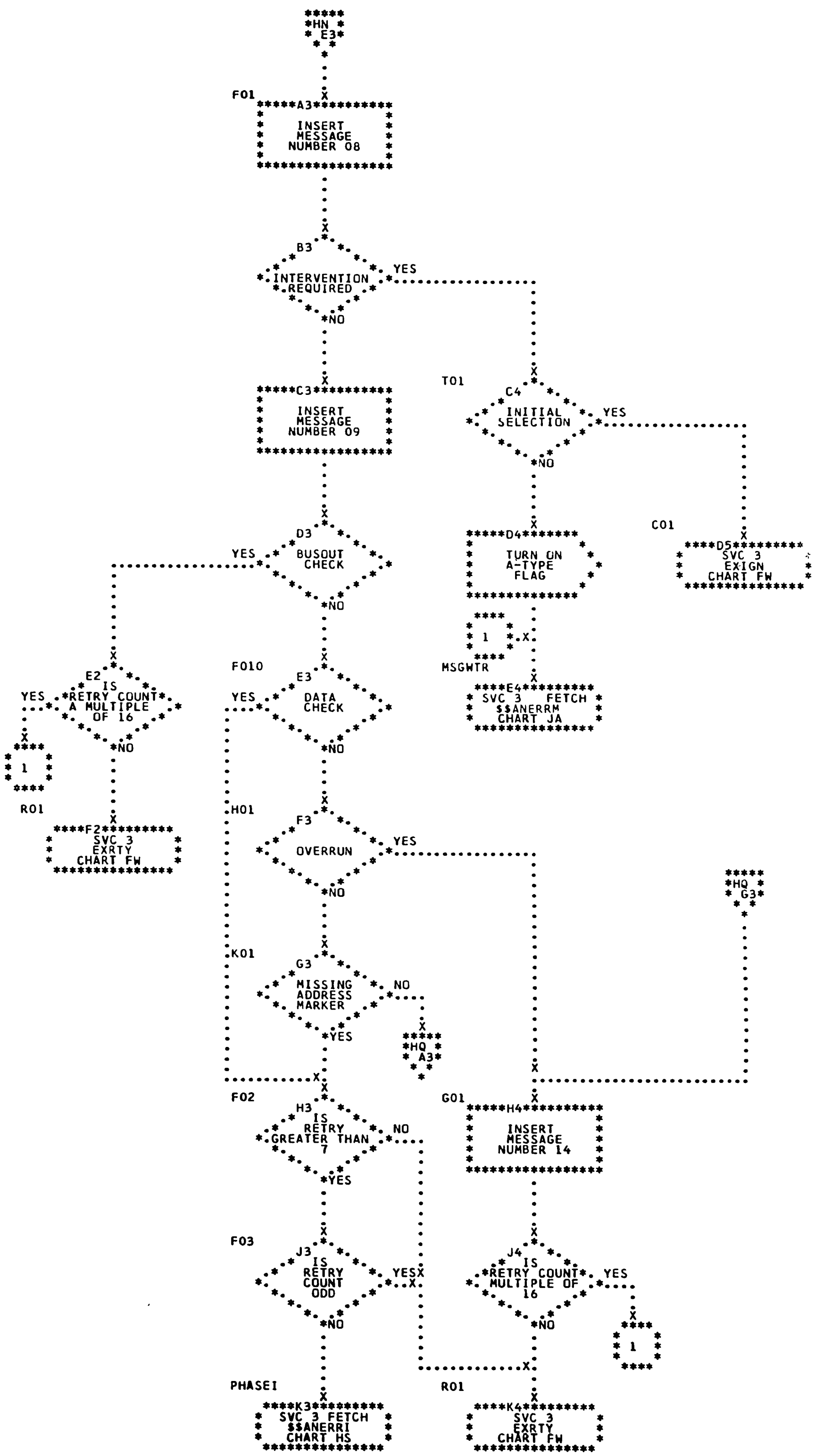


Chart HQ 2321 ERP Error Analysis (Part 3 of 3) $$ANERRG;
Refer to Supervisor, Chart 18

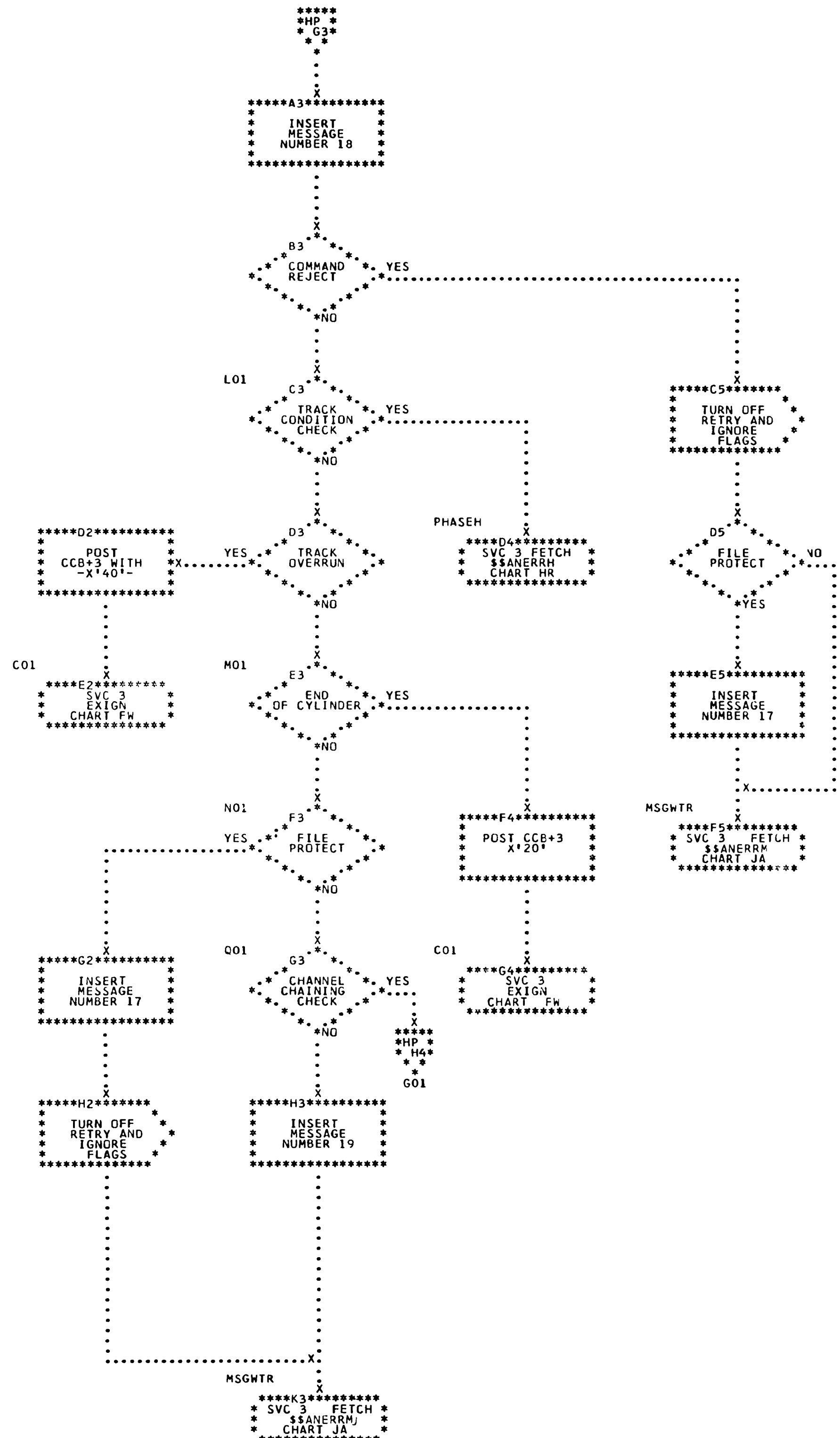


Chart HR. 2321 ERP Track Condition Check ($$ANERRH); Refer to Supervisor, Chart 18

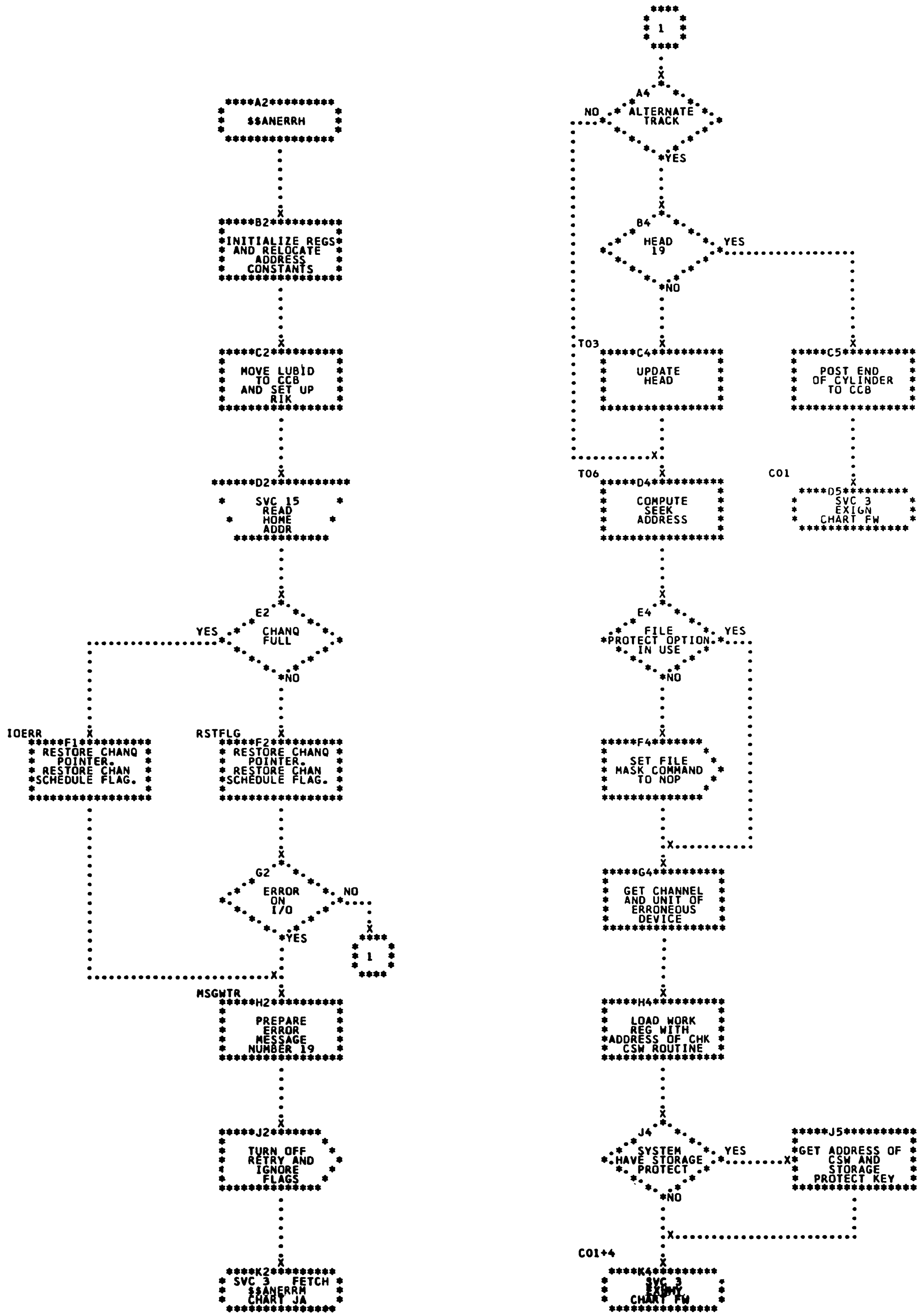


Chart HS. 2321 ERP-- Data Check/Missing Address Marker ($$ANERRI); Refer to Supervisor, Chart 18

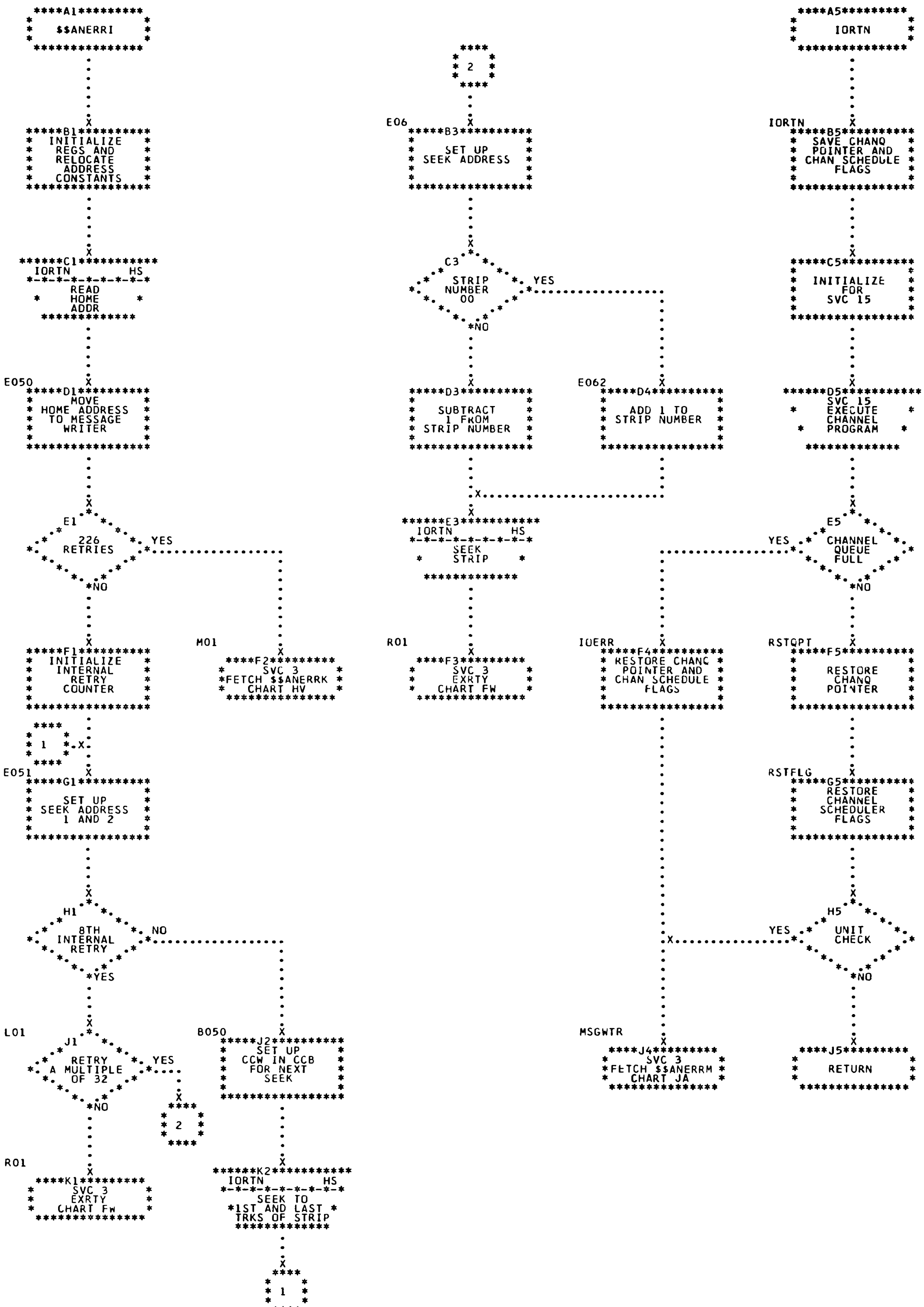


Chart HT. 2321 ERP-- NRF/Missing Address Marker, NRF/Seek Check (Part 1 of 2) $$ANERRJ; Refer to Supervisor, Chart 18

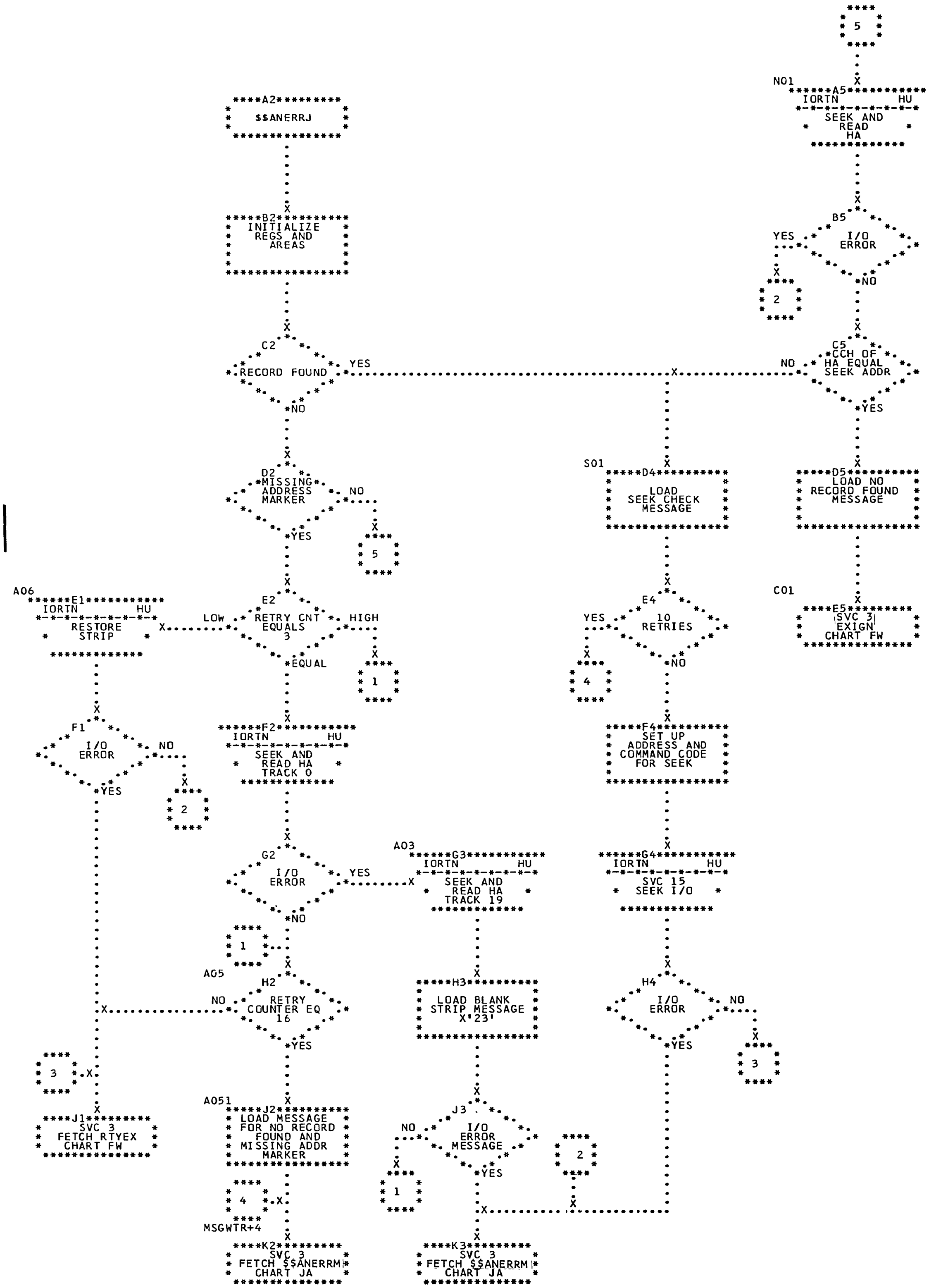


Chart HU. 2321 ERP NRF/Missing Address Marker, NRF/Seek Check (Part 2 of 2) $$ANERRJ; Refer to Supervisor, Chart 18

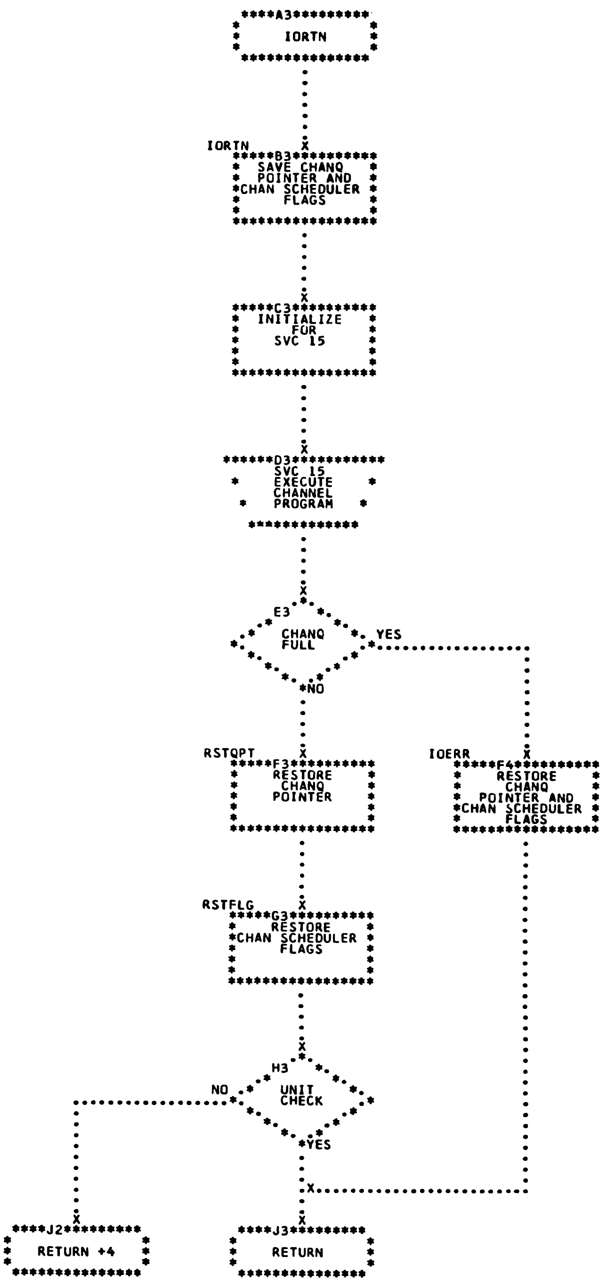


Chart HV. 2321 ERP--Continuation of $$ANERRJ ($$ANERRK); Refer to Supervisor, Chart 18

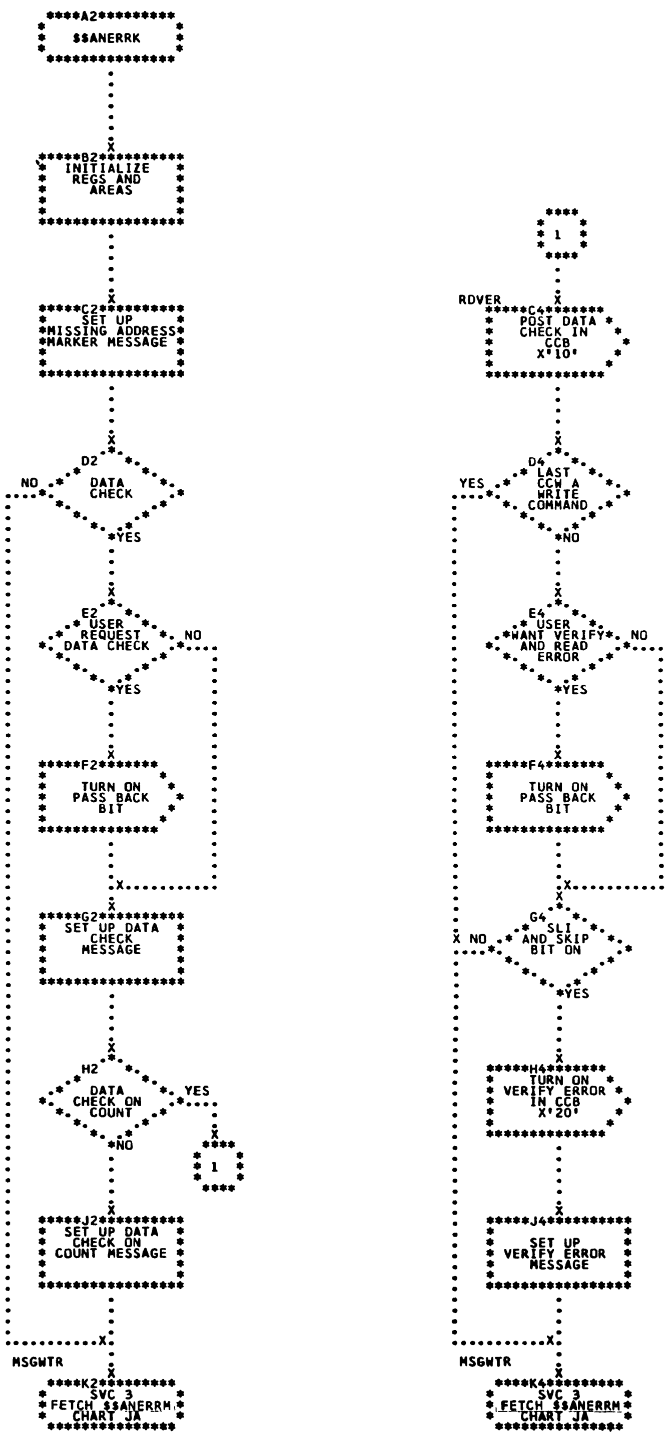


Chart JA. Message Writer-- Determine Action Type and
Targets; $$ANERRM; Refer to Supervisor, Chart 19

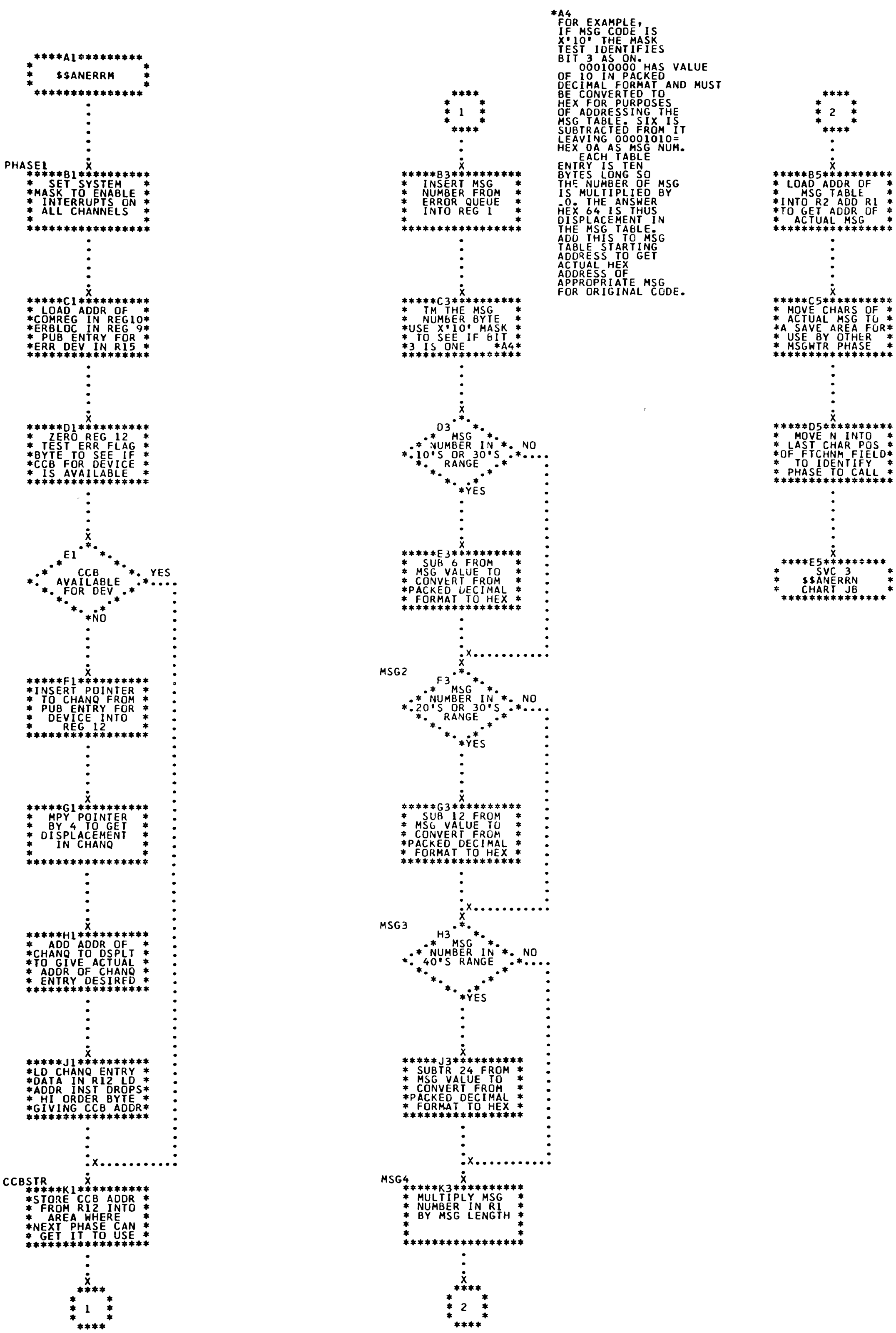


Chart JB. Message Writer-- Determine Ownership (Part 1 of 2); $$ANERRN; Refer to Supervisor, Chart 19

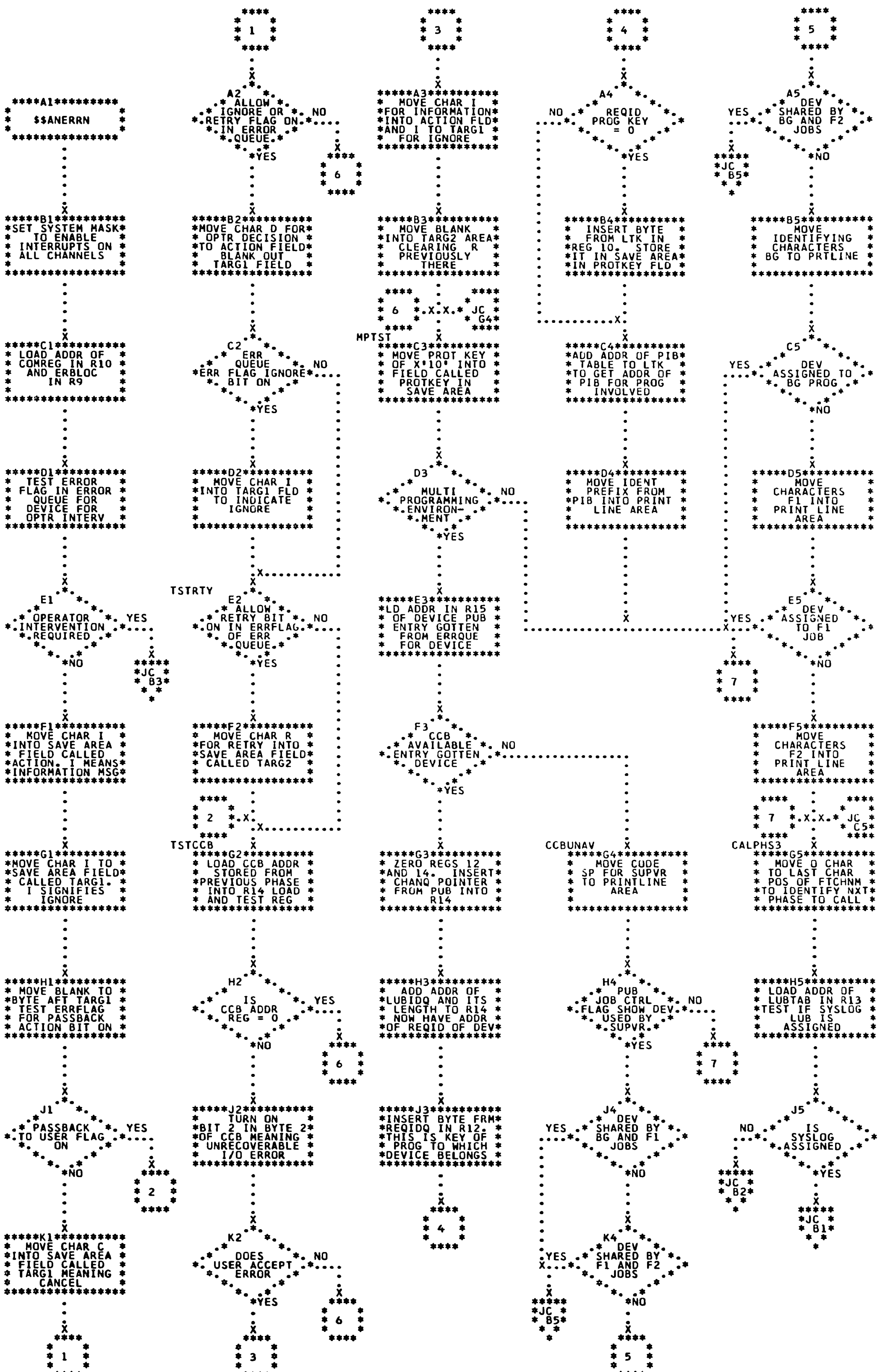


Chart JC. Message Writer-- Determine Ownership (Part 2 of 2) $$ANERRN; Refer to Supervisor, Chart 19

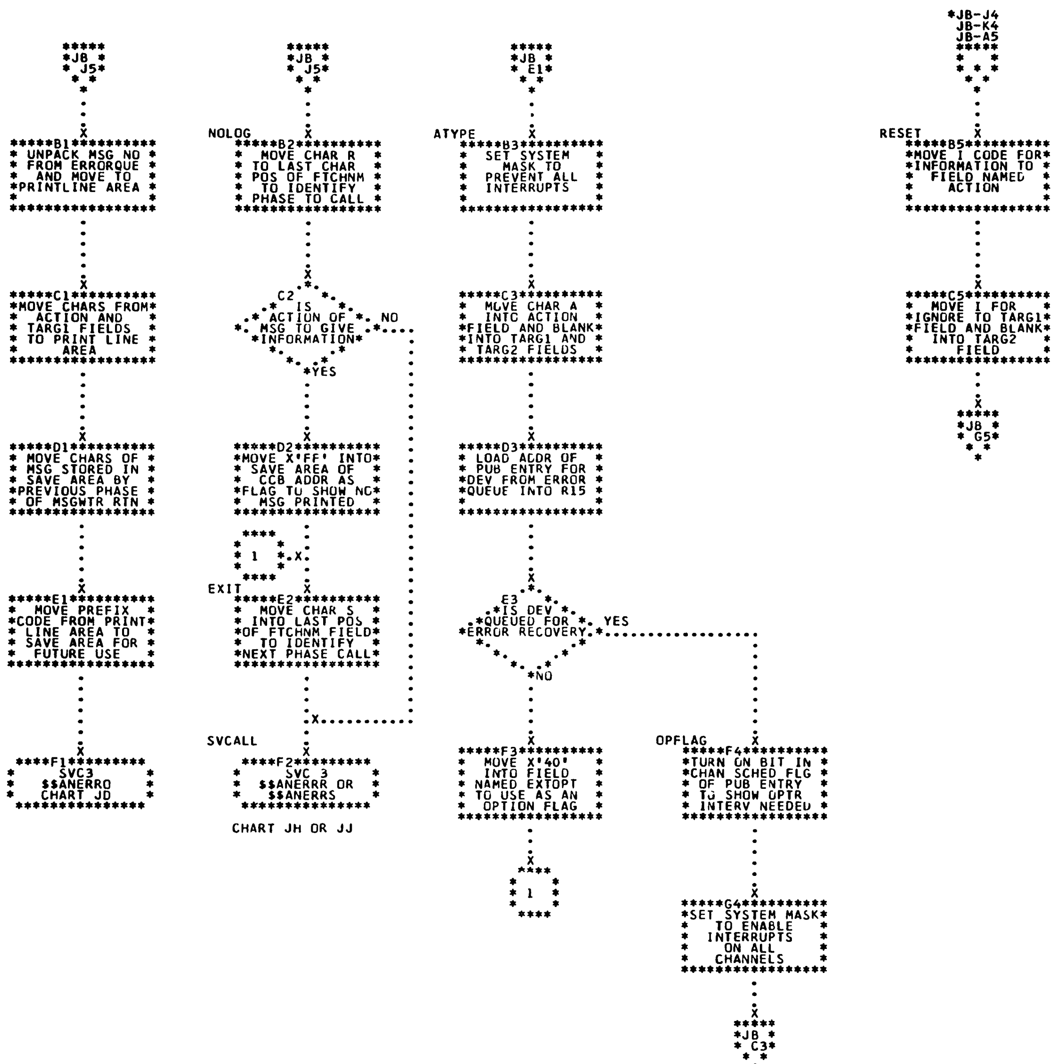


Chart JD. Message Writer-- Format Message; $$ANERRO; Refer to Supervisor, Chart 19

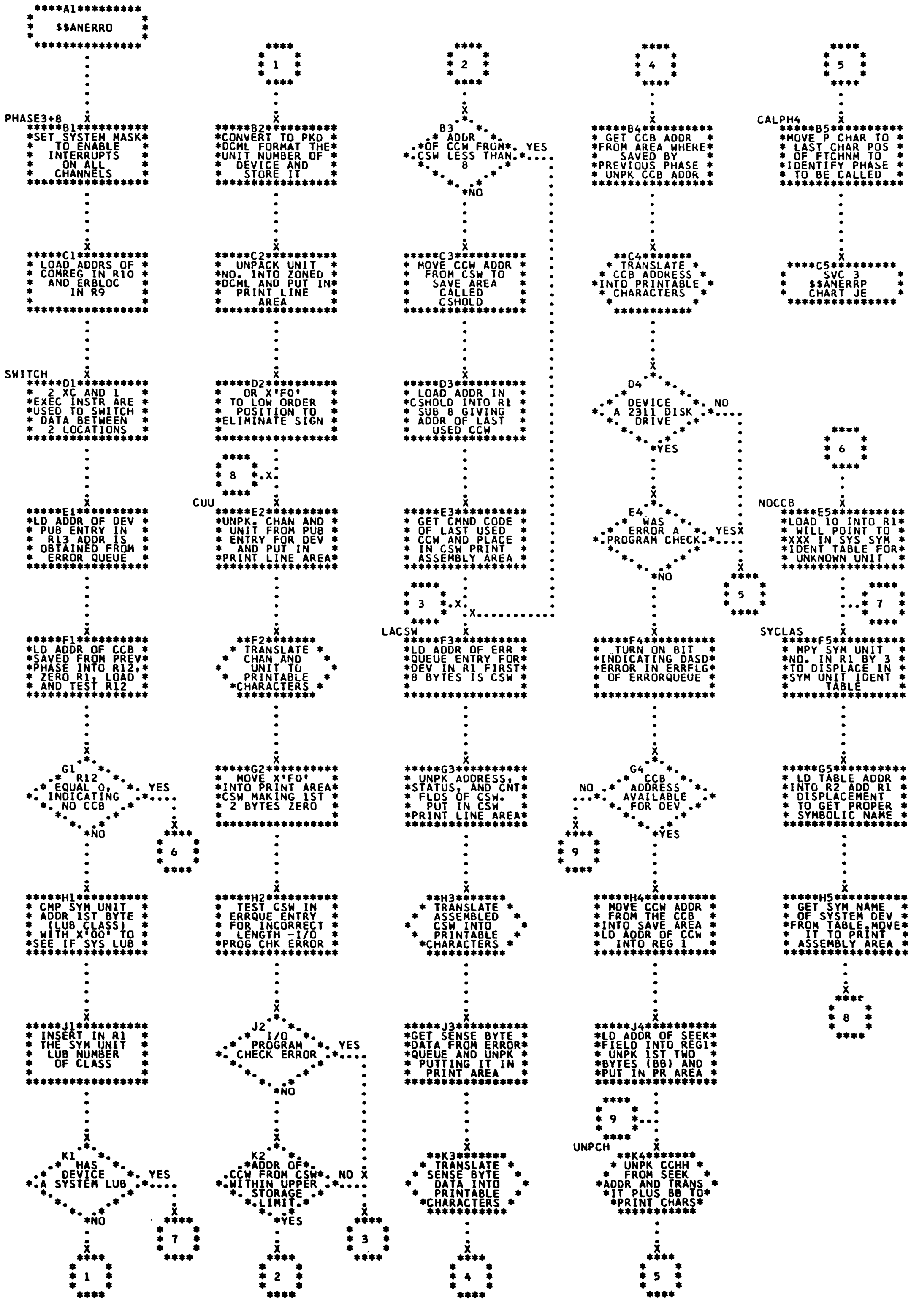


Chart JE. Message Writer -- Output Message; $$ANERRP;
Refer to Supervisor, Chart 19

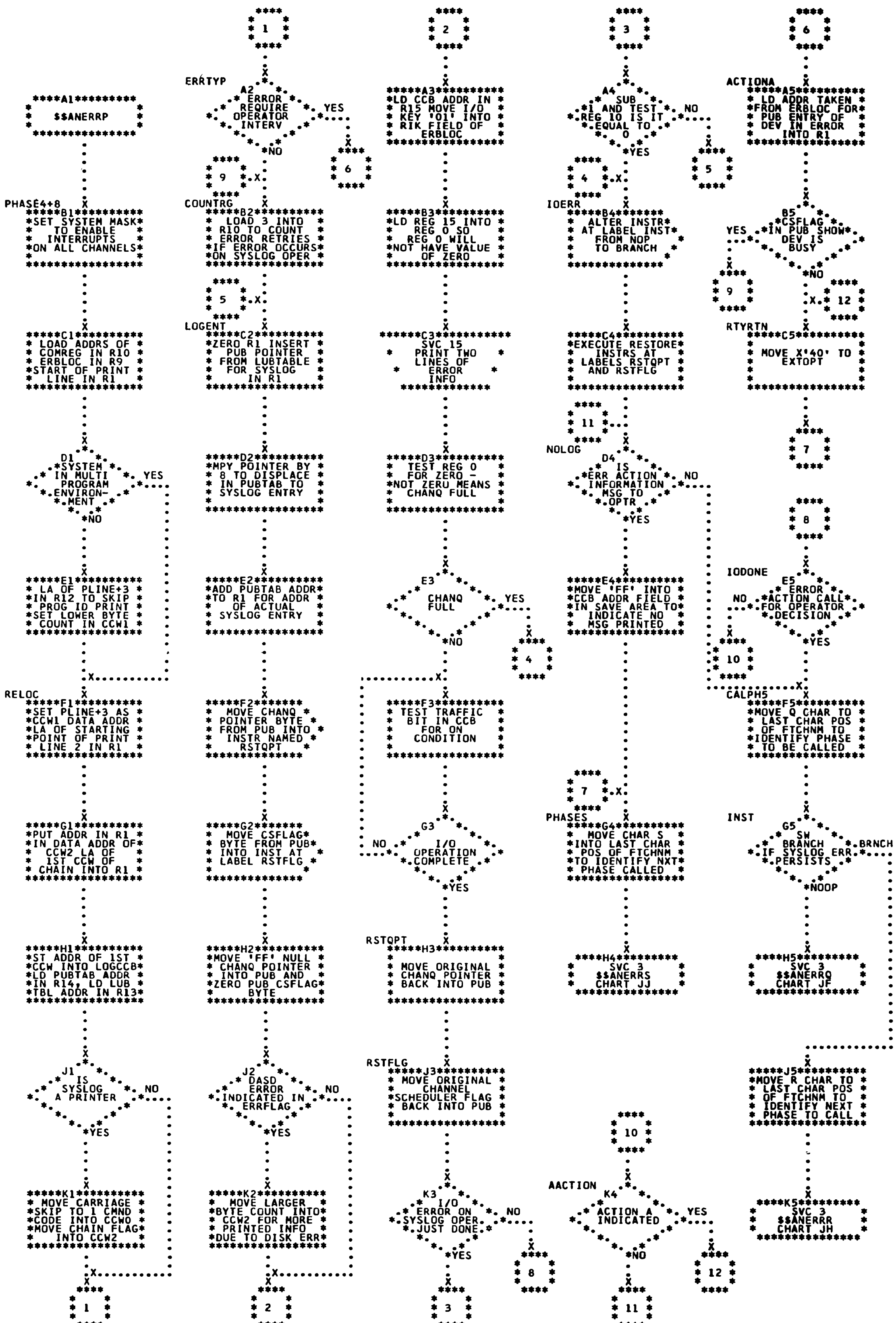


Chart JF. Message Writer-- Read Operator Reply (Part 1 of 2) $$ANERRQ; Refer to Supervisor, Chart 19

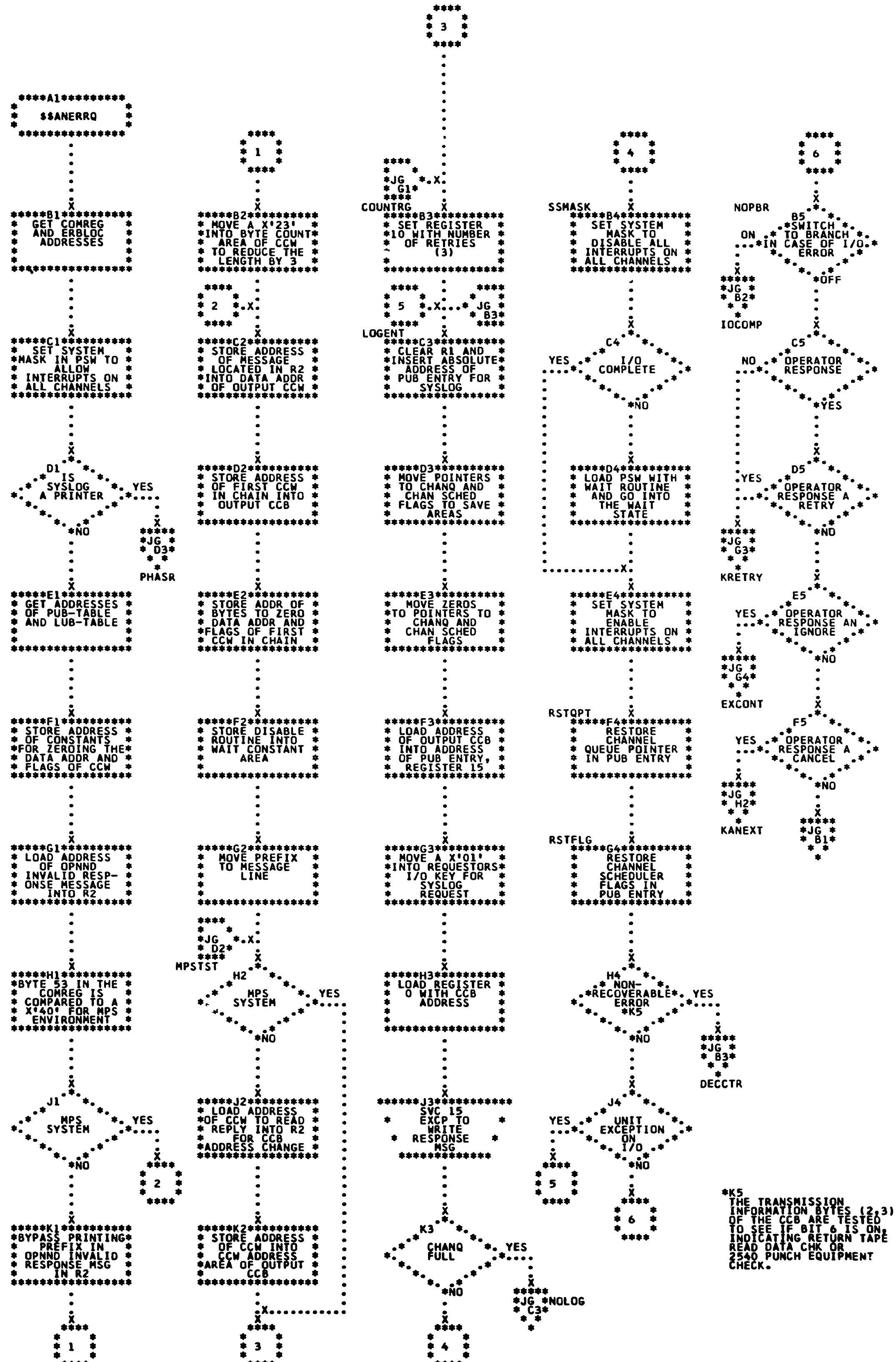


Chart JG. Message Writer-- Read Operator Reply (Part 2 of 2) $$ANERRQ; Refer to Supervisor, Chart 19

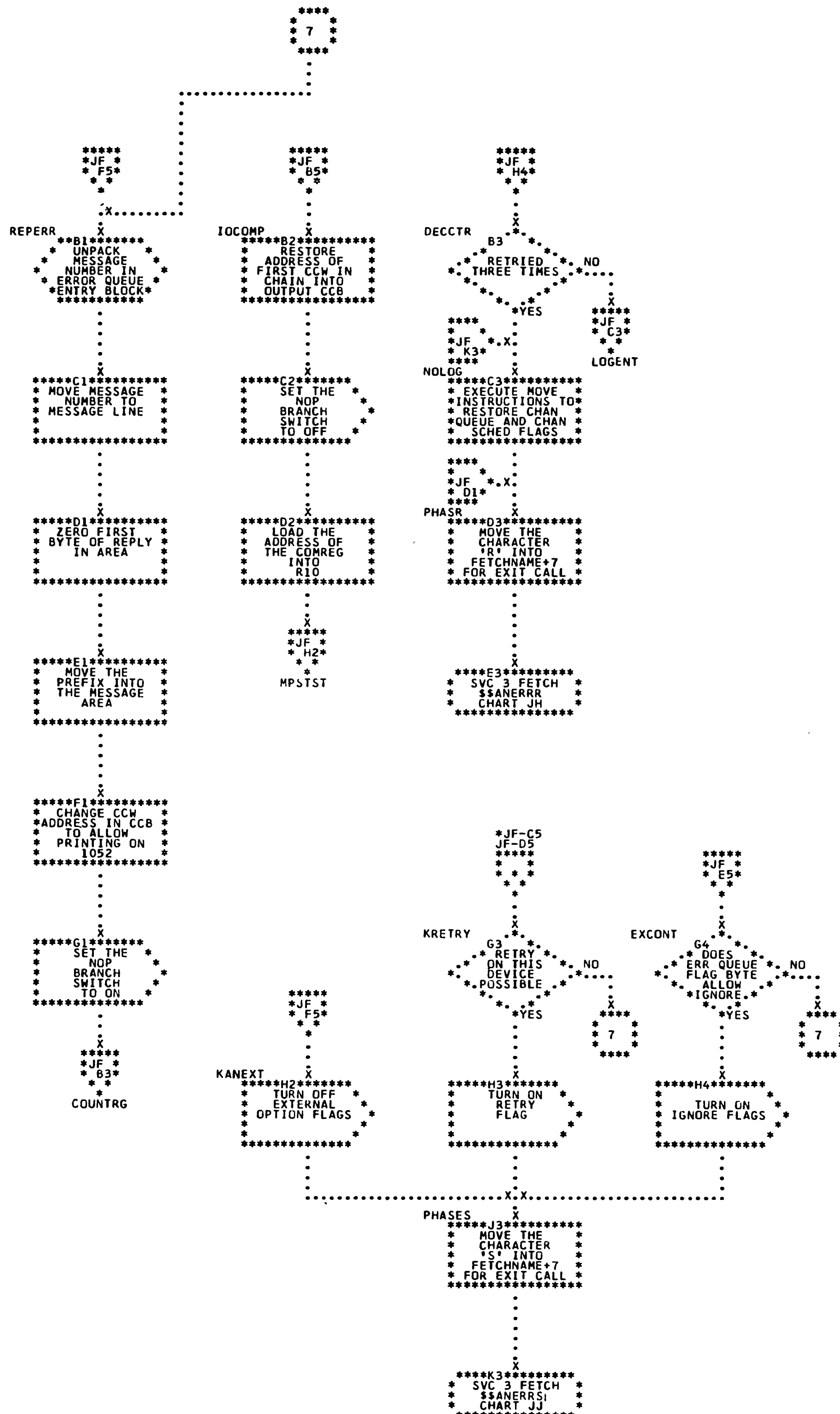


Chart JH. Message Writer-- Error Recovery; $$ANERRR; Refer to Supervisor, Chart 19

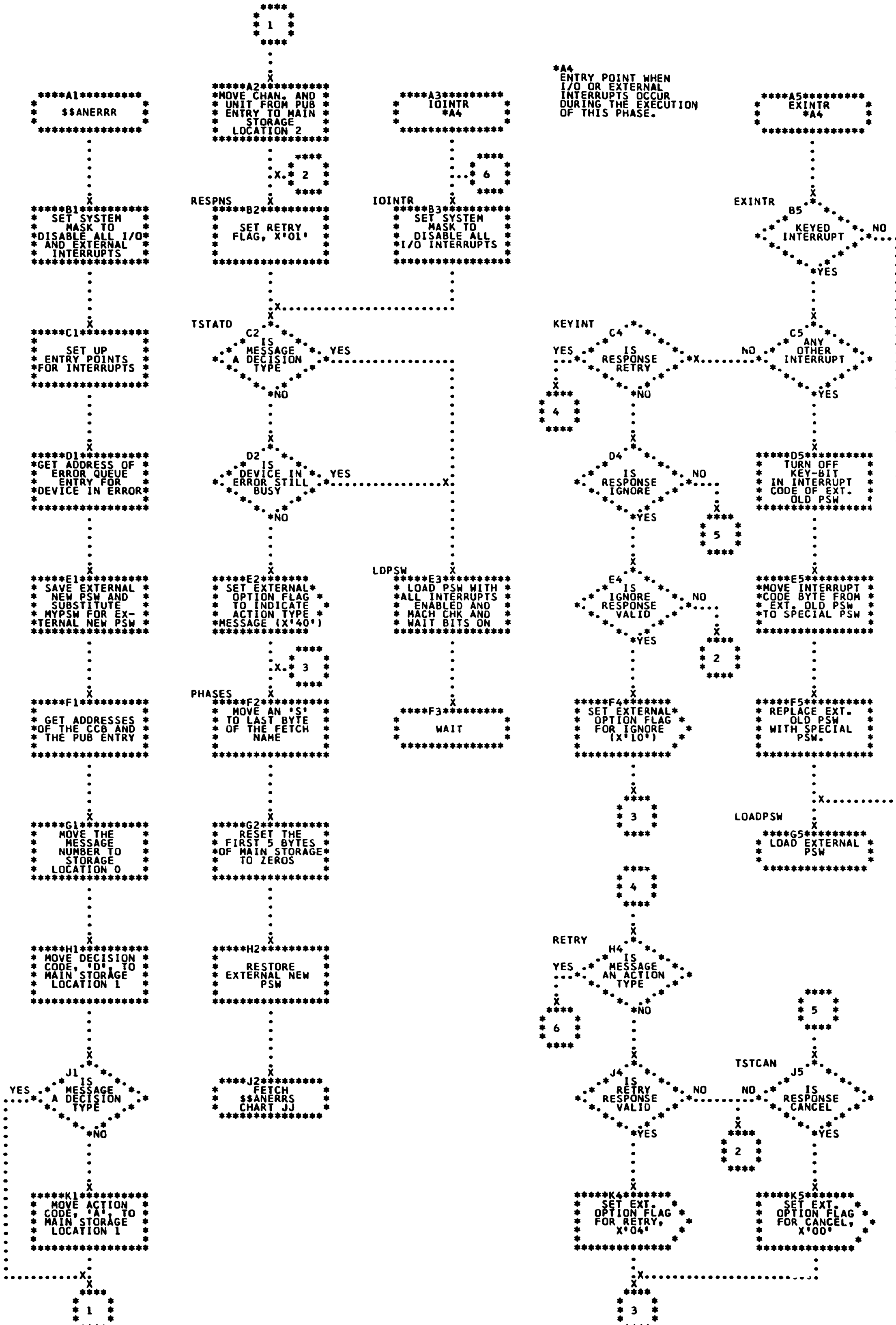


Chart JJ. Message Writer-- Cancel, Ignore or Dequeue ($$ANERRS); Refer to Supervisor, Chart 19

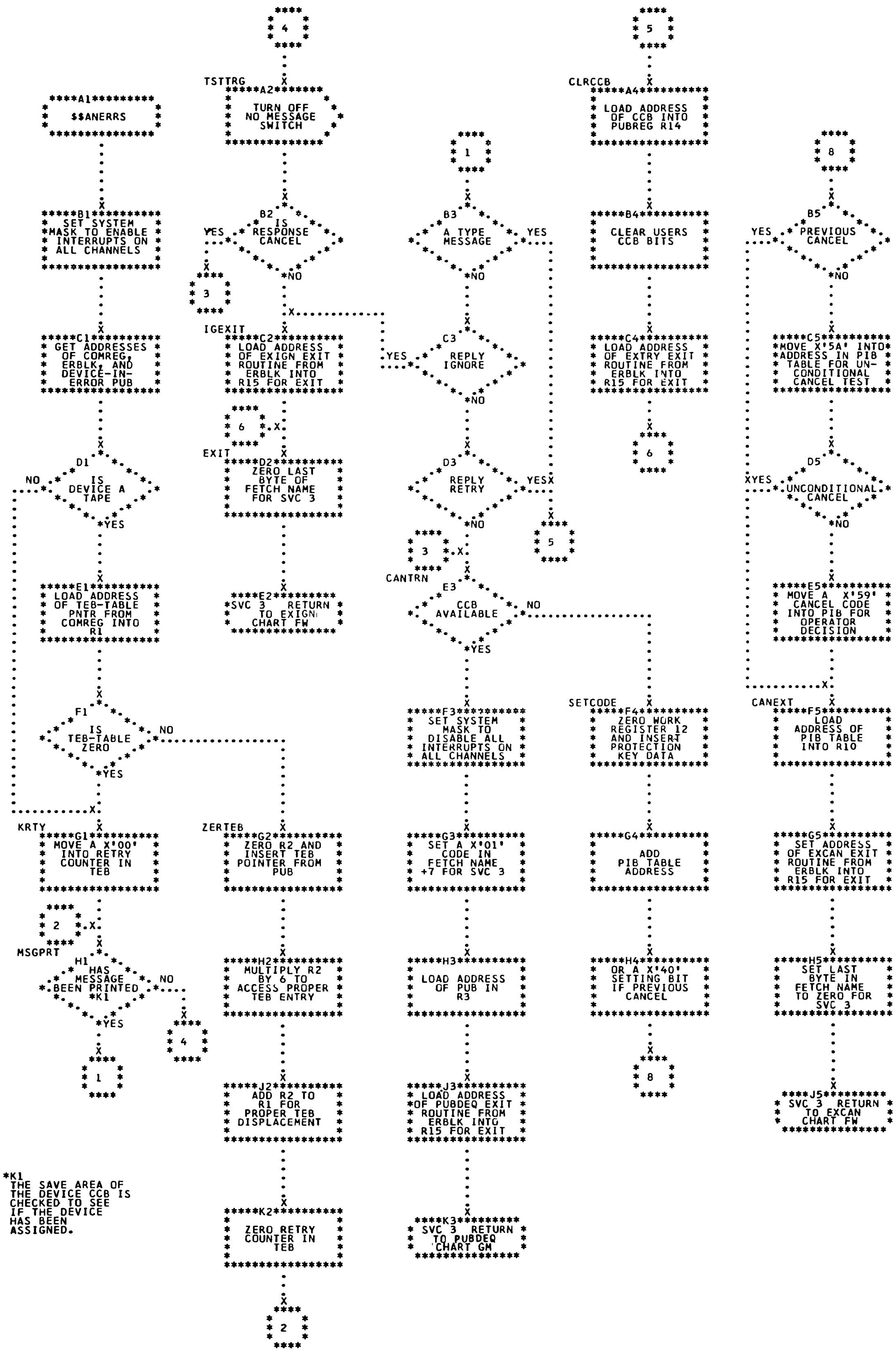


Chart JK. Unit Record ERP-- 1052 and 1056 (Part 1 of 2)
$$ANERRU; Refer to Supervisor, Chart 18

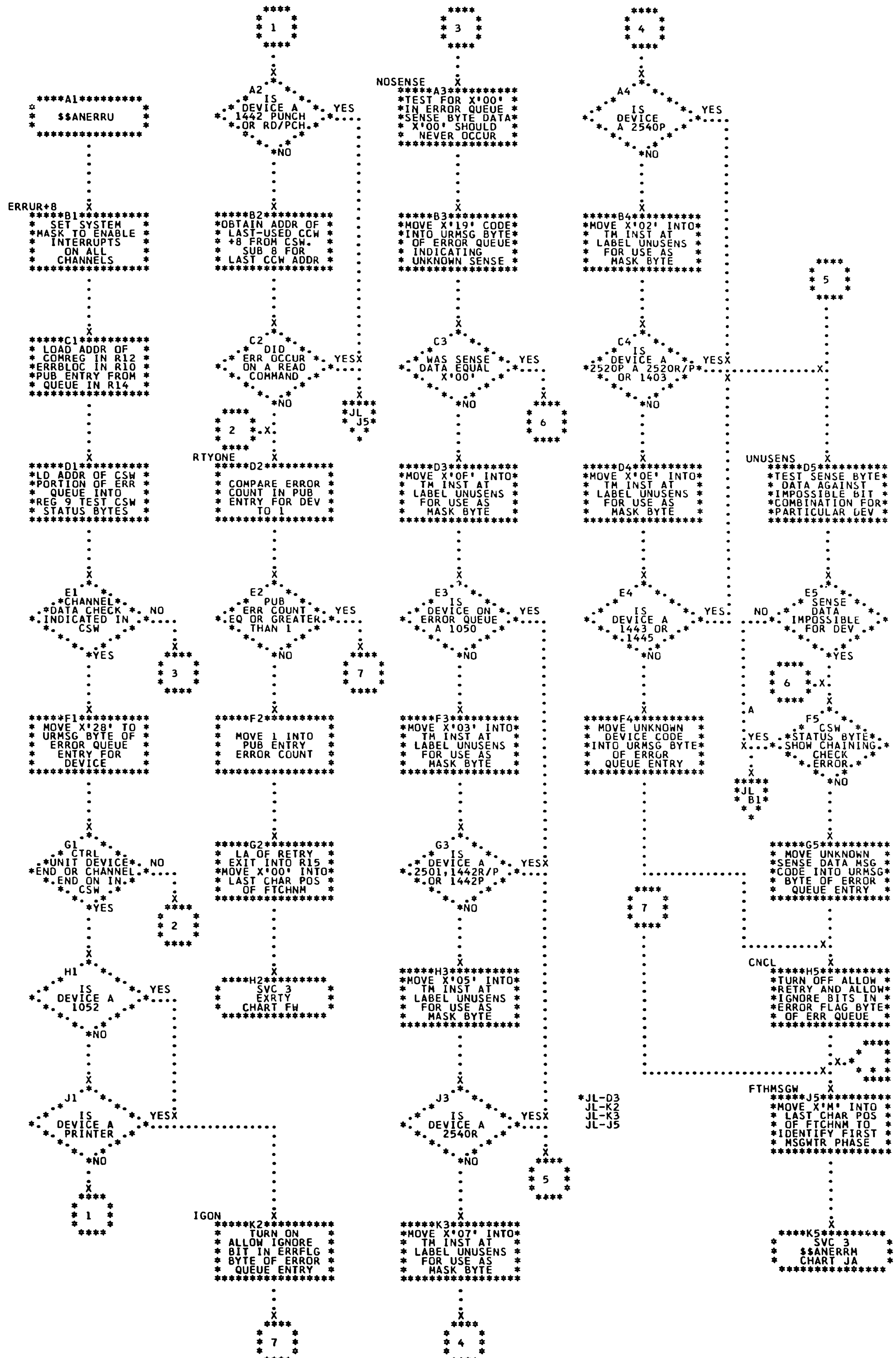


Chart JL. Unit Record ERP-- 1052 and 1056 (Part 2 of 2)
$$ANERRU; Refer to Supervisor, Chart 18

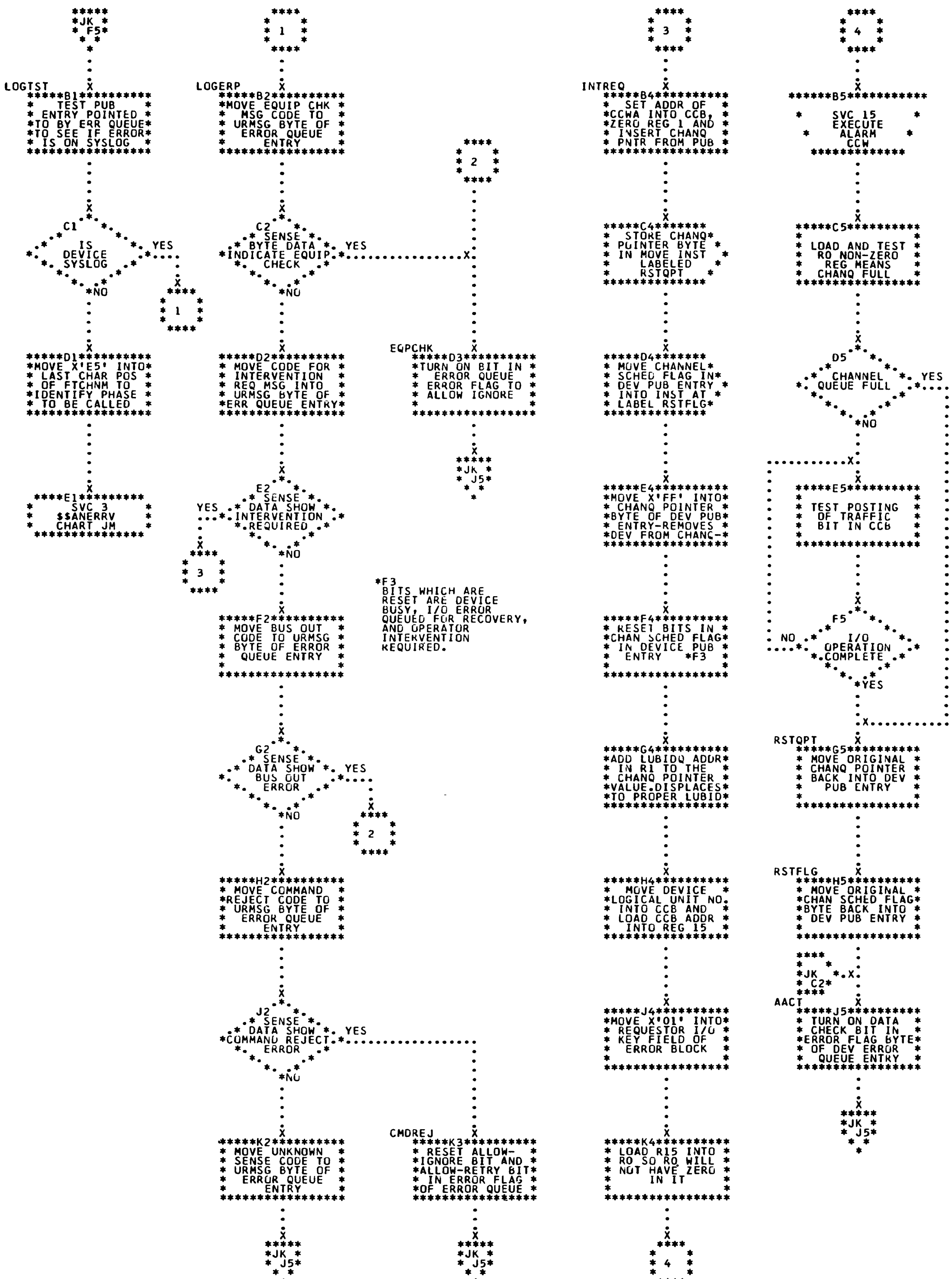


Chart JM. Unit Record ERP-- 1403, 1442, 1443, 2501, 2520, 2540, (Part 1 of 2) $$ANERRV; Refer to Supervisor, Chart 18

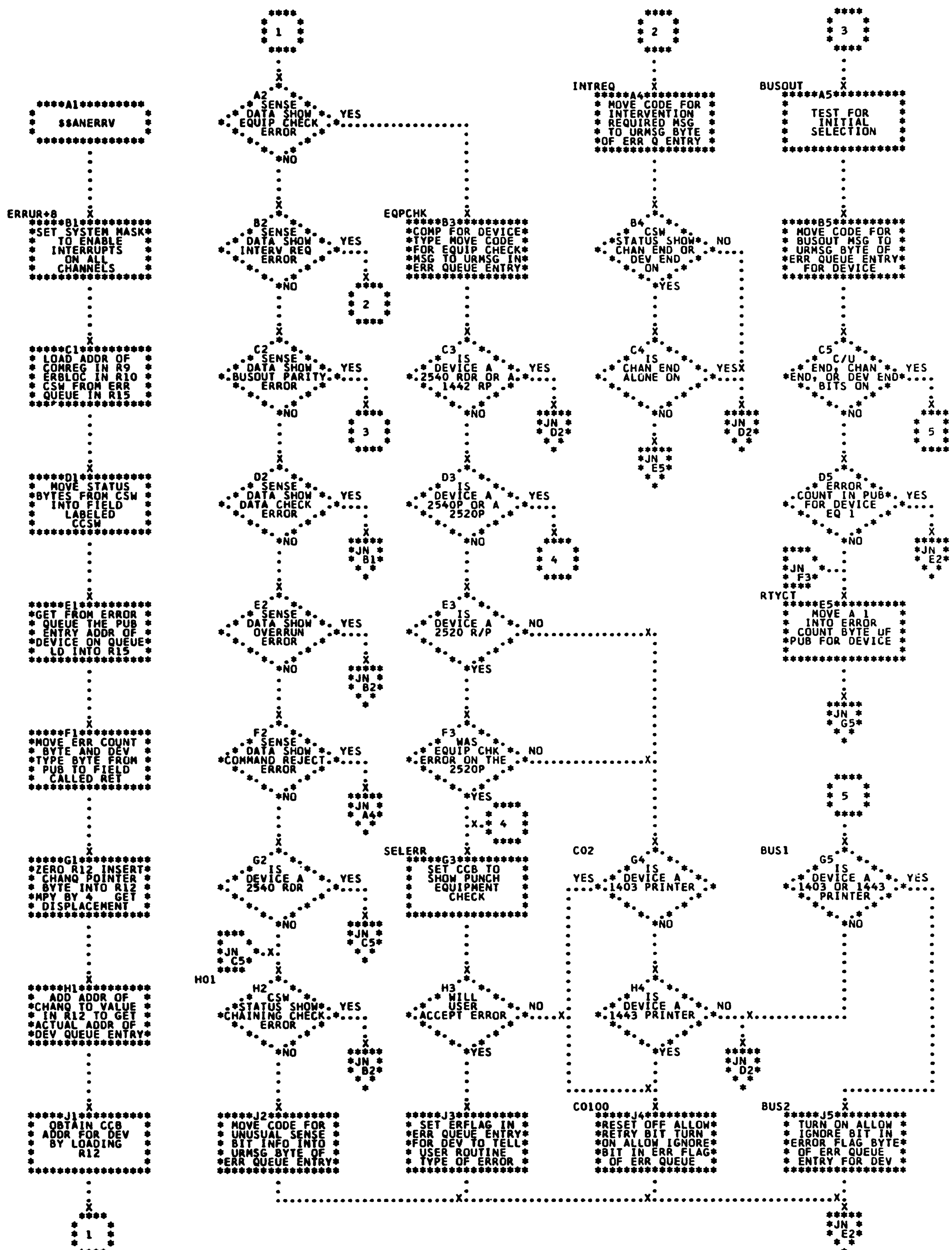


Chart JN. Unit Record ERP-- 1403, 1442, 1433, 2501, 2520, 2540, (Part 2 of 2) $$ANERRV; Refer to Supervisor, Chart 18

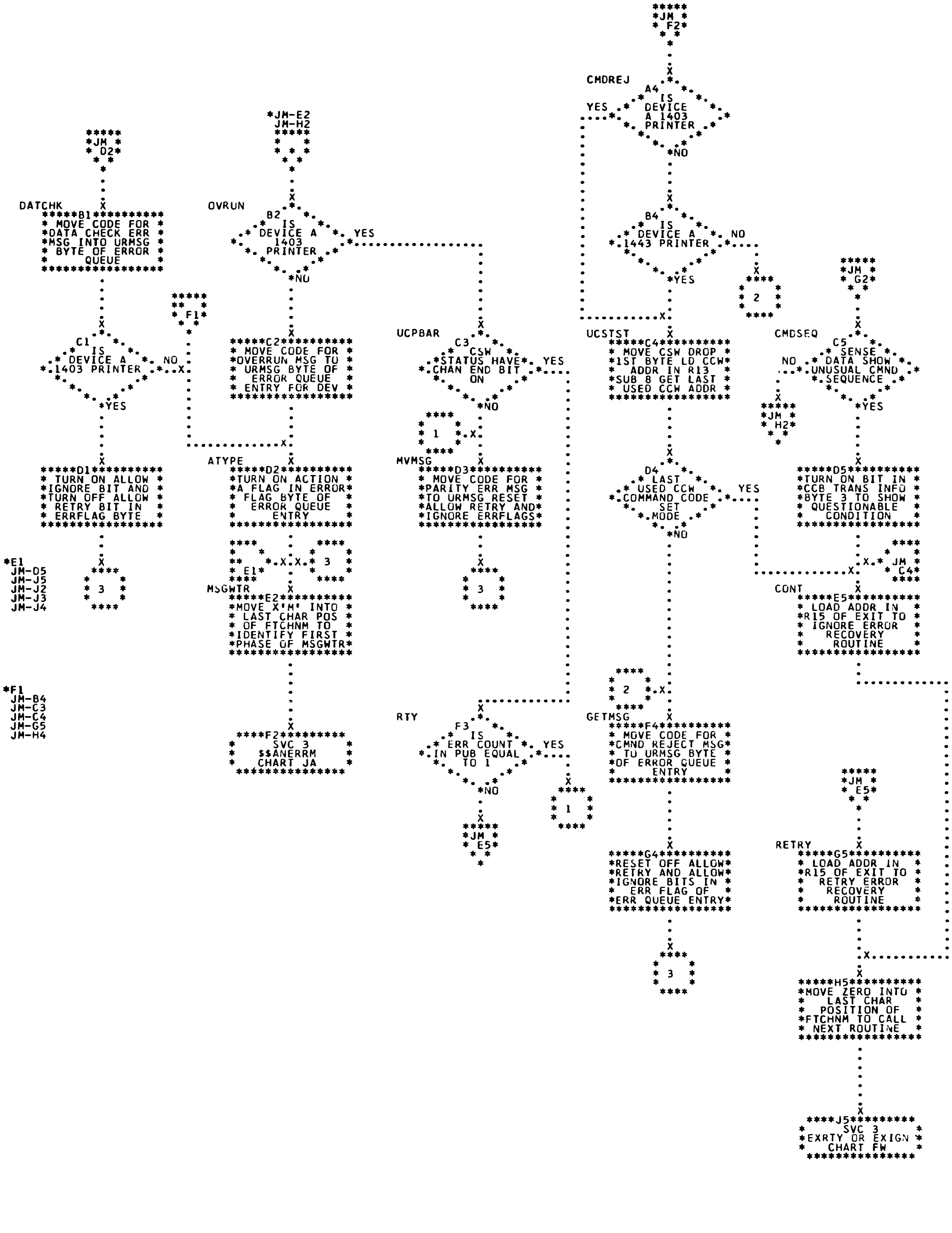
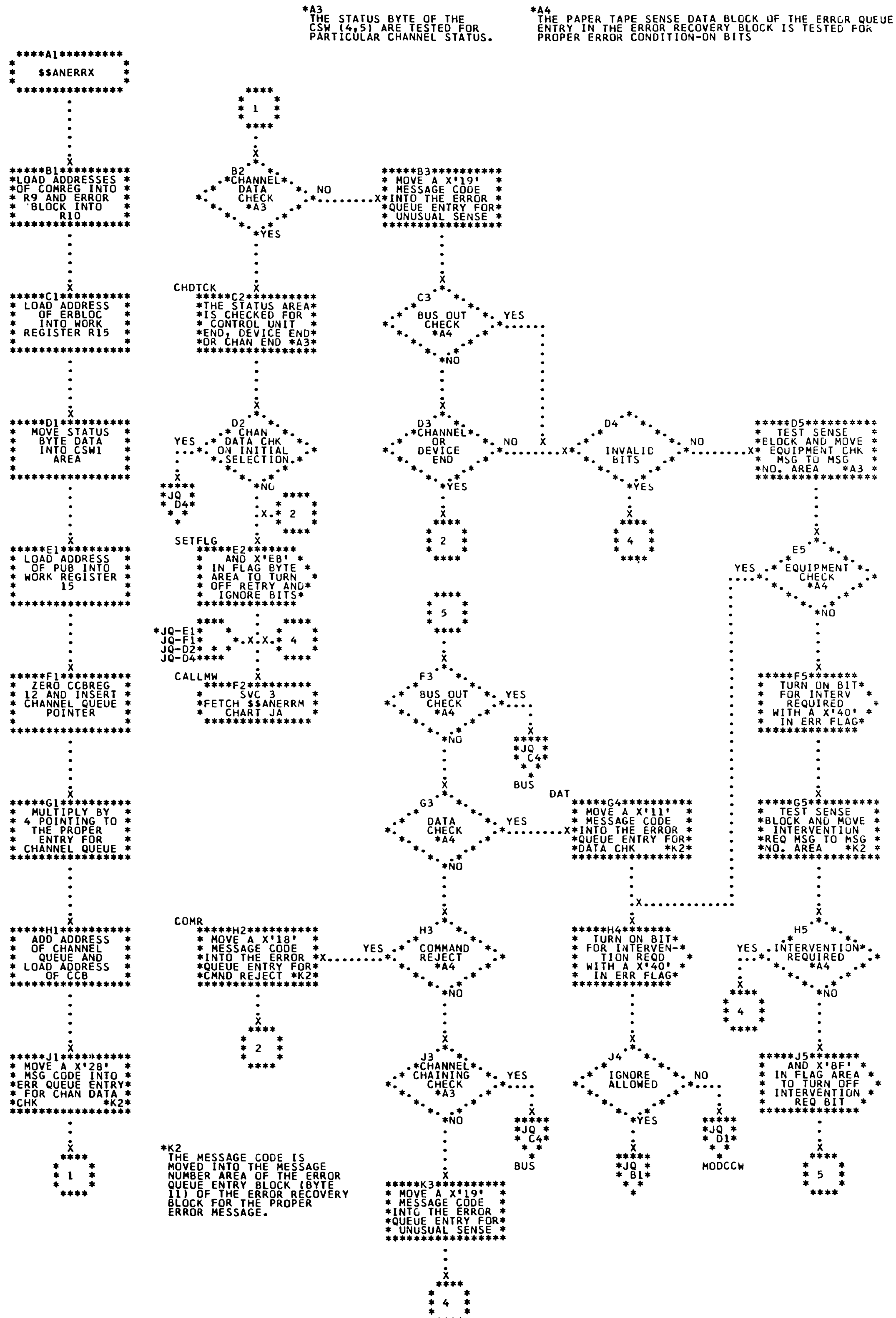


Chart JP. Paper Tape ERP--2671 (Part 1 of 2) $$ANERRX; Refer to Supervisor, Chart 18



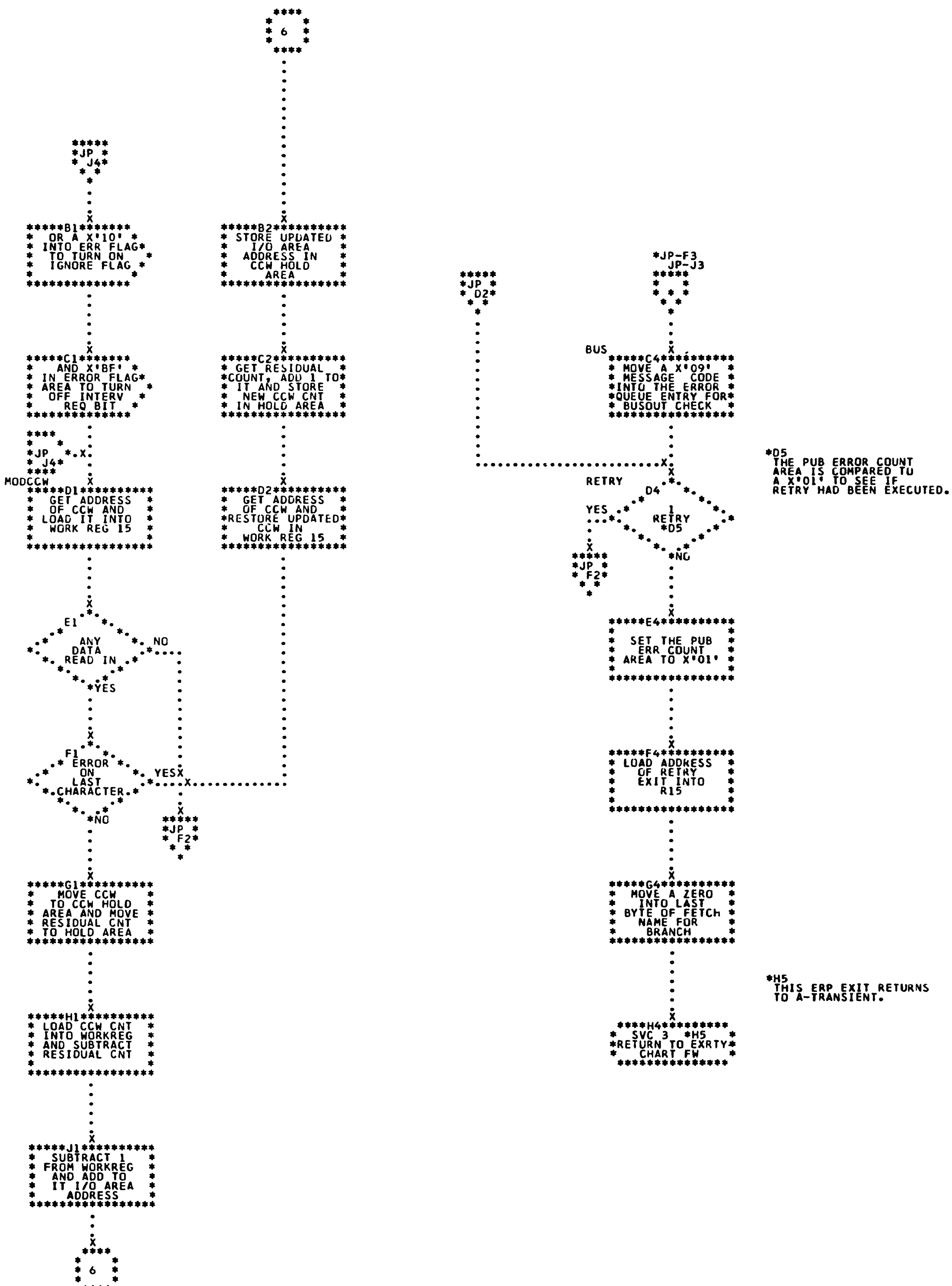

6
JP J4
B1 OR A X'10' INTO ERR FLAG TO TURN ON IGNORE FLAG
C1 AND X'BF' IN ERROR FLAG AREA TO TURN OFF INTERV REQ BIT
JP J4
MODCCW
D1 GET ADDRESS OF CCW AND LOAD IT INTO WORK REG 15
E1 ANY DATA READ IN
NO
YES
F1 ERROR ON LAST CHARACTER
YES
NO
JP F2
G1 MOVE CCW TO CCW HOLD AREA AND MOVE RESIDUAL CNT TO HOLD AREA
H1 LOAD CCW CNT INTO WORKREG AND SUBTRACT RESIDUAL CNT
J1 SUBTRACT 1 FROM WORKREG AND ADD TO IT I/O AREA ADDRESS
6
B2 STORE UPDATED I/O AREA ADDRESS IN CCW HOLD AREA
C2 GET RESIDUAL COUNT, ADD 1 TO IT AND STORE NEW CCW CNT IN HOLD AREA
D2 GET ADDRESS OF CCW AND RESTORE UPDATED CCW IN WORK REG 15
JP D2
JP-F3
JP-J3
BUS
C4 MOVE A X'09' MESSAGE CODE INTO THE ERROR QUEUE ENTRY FOR BUSOUT CHECK
RETRY
D4 1 RETRY *D5
YES
NO
JP F2
*D5 THE PUB ERROR COUNT AREA IS COMPARED TO A X'01' TO SEE IF RETRY HAD BEEN EXECUTED.
E4 SET THE PUB ERR COUNT AREA TO X'01'
F4 LOAD ADDRESS OF RETRY EXIT INTO R15
G4 MOVE A ZERO INTO LAST BYTE OF FETCH NAME FOR BRANCH
H4 SVC 3 *H5 RETURN TO EXRTY CHART FW
*H5 THIS ERP EXIT RETURNS TO A-TRANSIENT.

Chart JR. Optical Reader ERP--1285; $$ANERR9: Refer to Supervisor, Chart 18

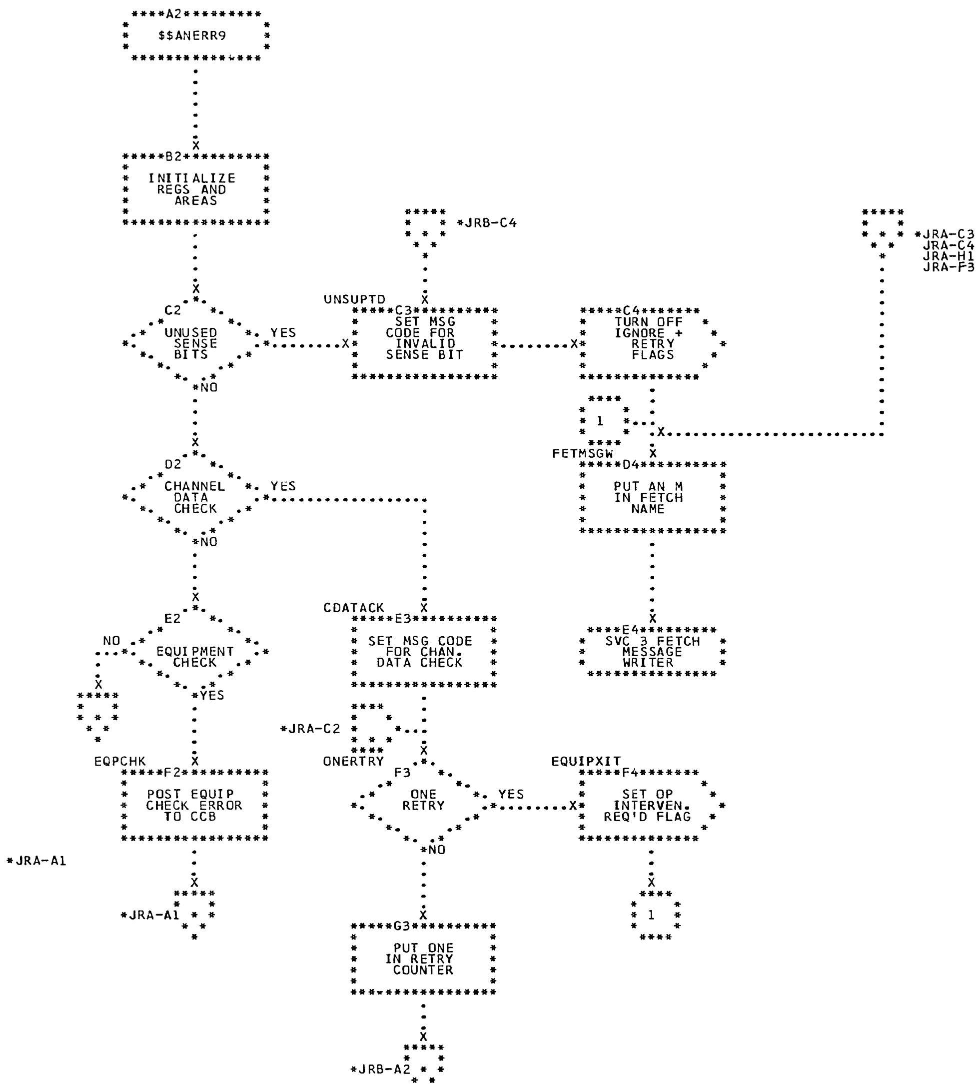


Chart JRA. Optical Reader ERP--1285; $$ANERR9: Refer to Supervisor, Chart 18

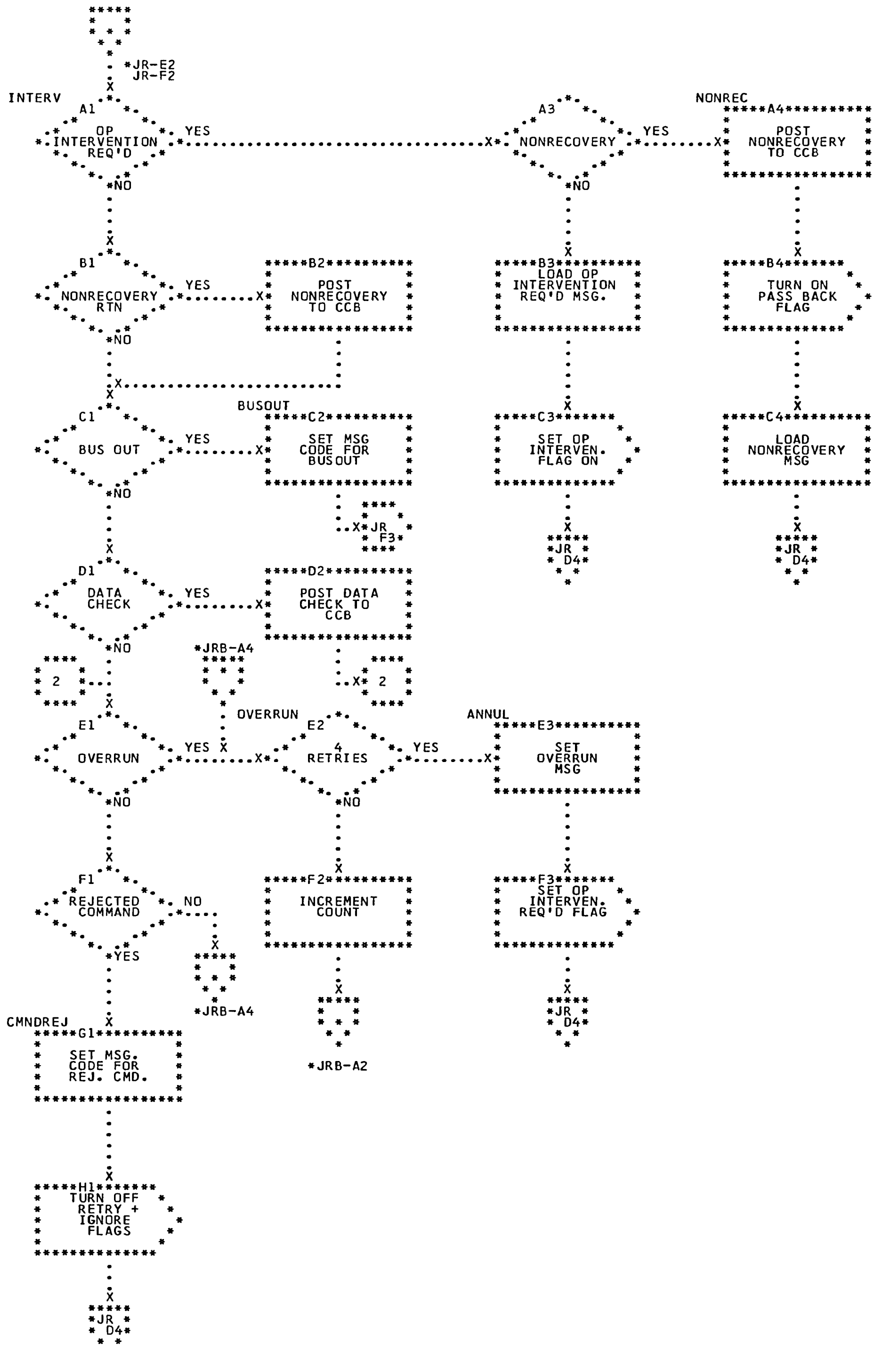


Chart JRB. Optical Reader ERP--1285; $$ANERR9: Refer to Supervisor, Chart 18

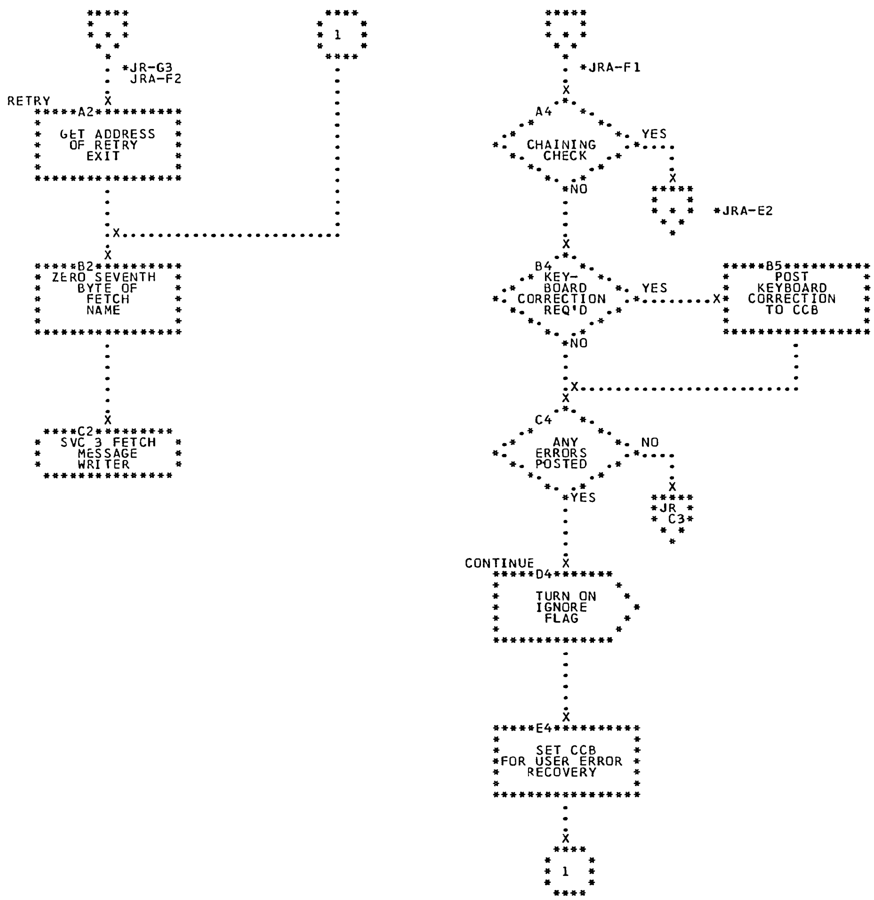


Chart JS. Physical Attention-- Send Message; $$ANERRY; Refer to Supervisor, Chart 20

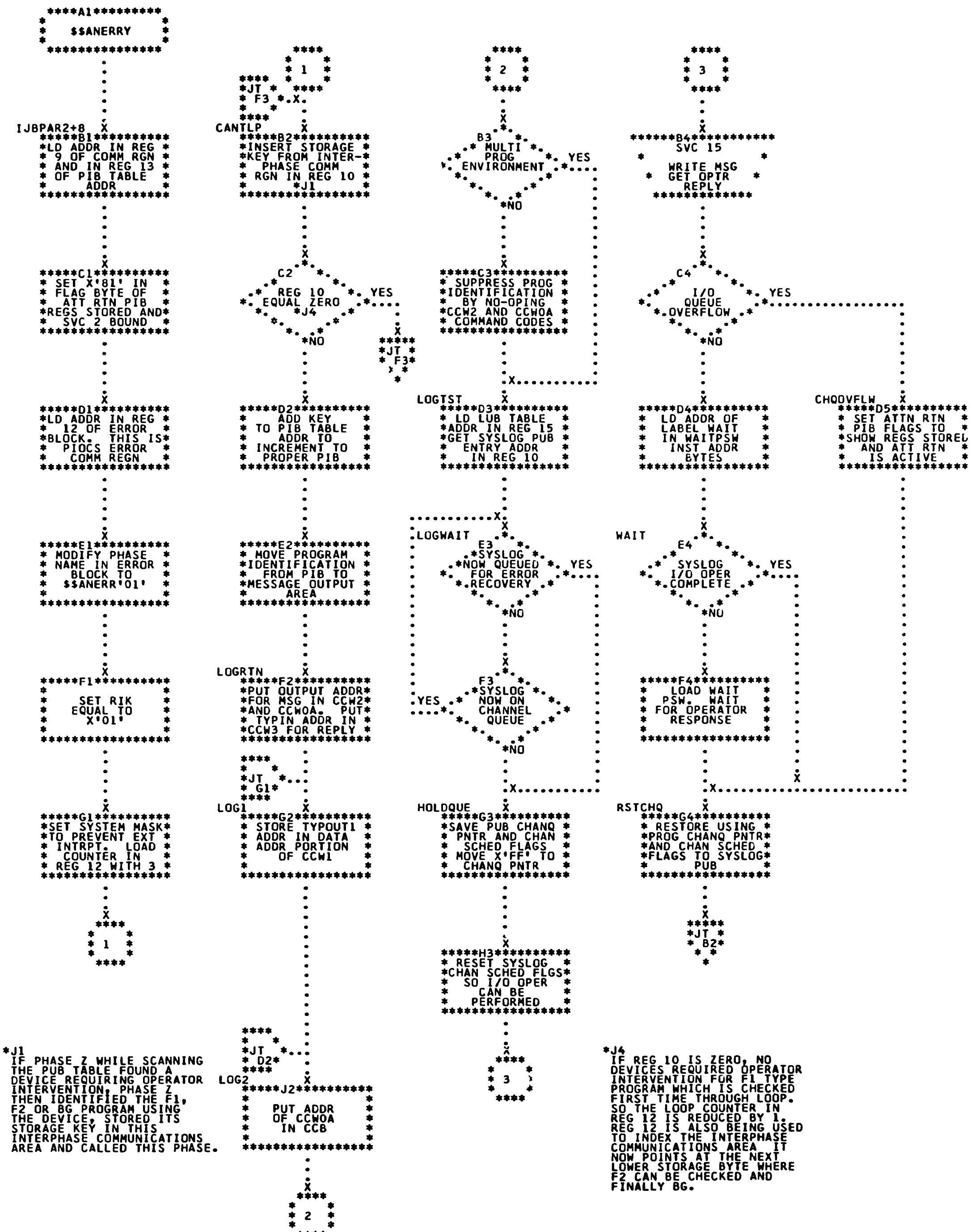
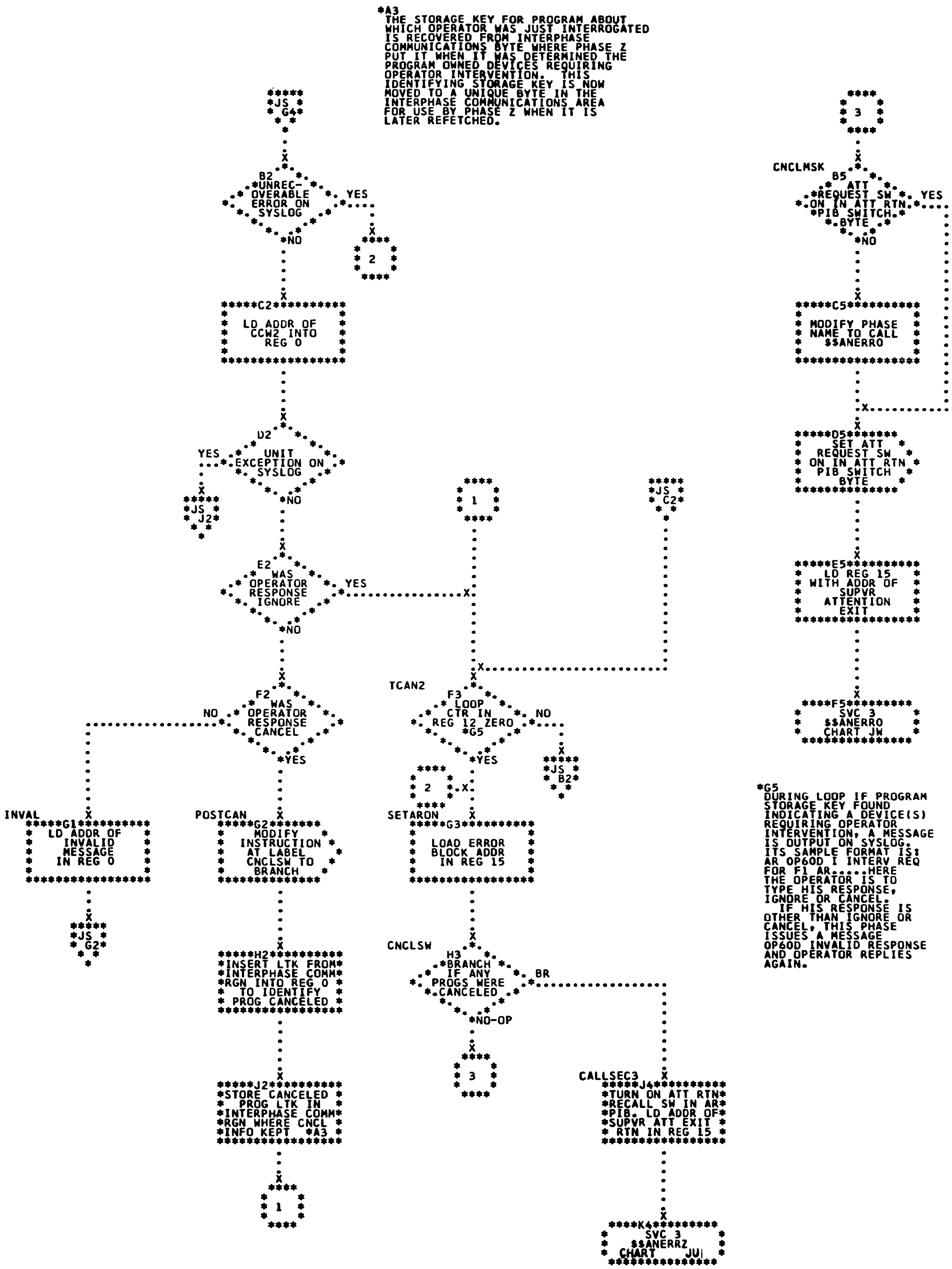


Chart JT. Physical Attention-- Read Operator Reply; $$ANERRY; Refer to Supervisor, Chart 20



## Chart JU. Physical Attention-- Initial PUB Scan; $$ANERRZ; Refer to Supervisor, Chart 20

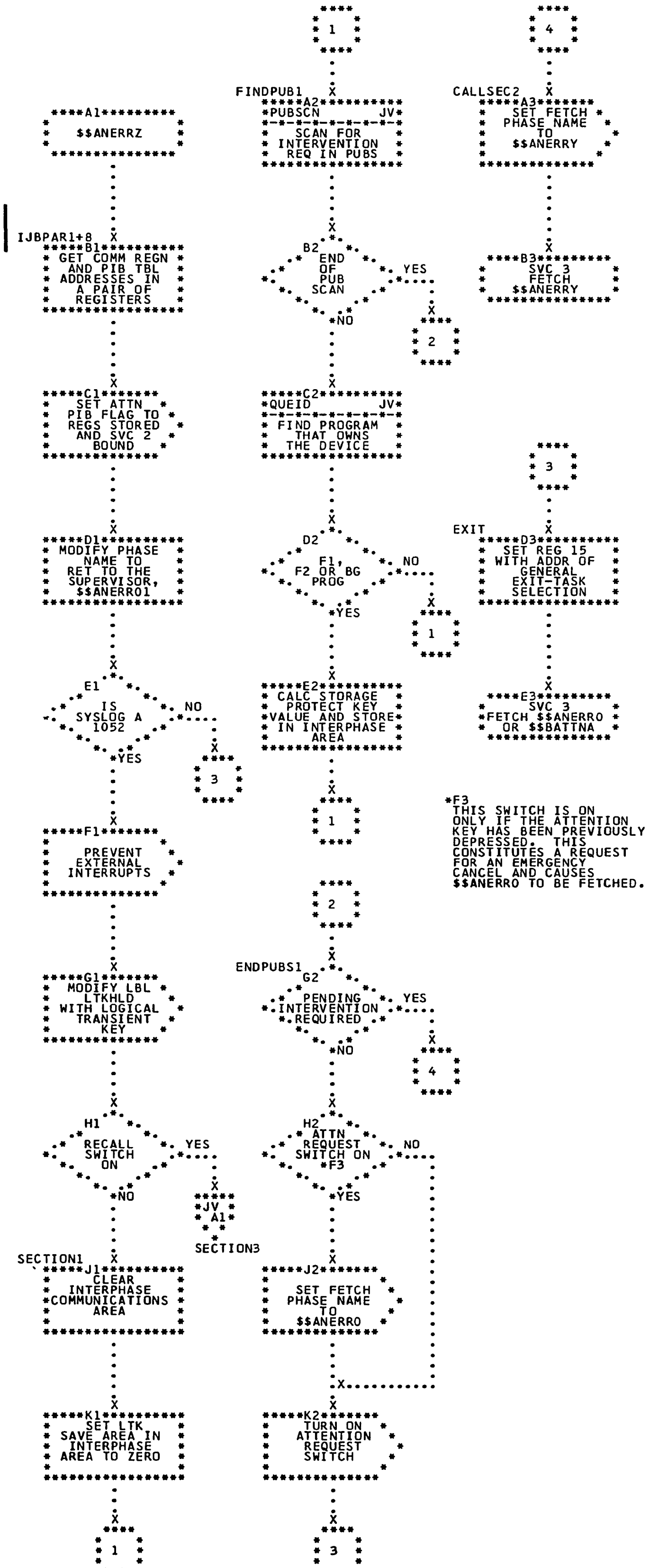


**Chart JV.** Physical Attention-- Cancel Routine and Physical Attention Subroutines ($$ANERRZ); Refer to Supervisor, Chart 20

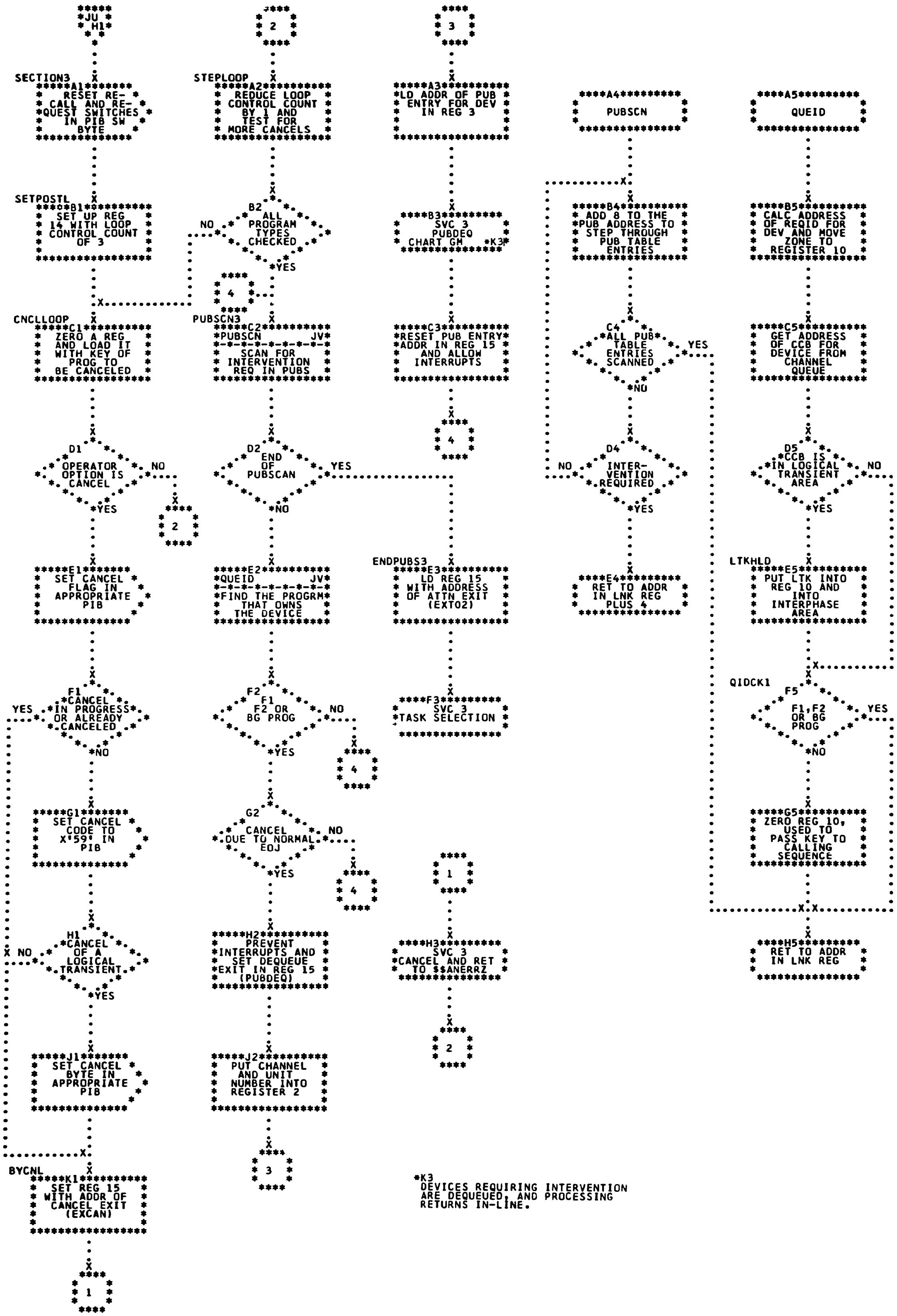


Chart JW. Physical Attention-- Emergency Cancel (Part 1 of 2) $$ANERR0; Refer to Supervisor, Chart 20

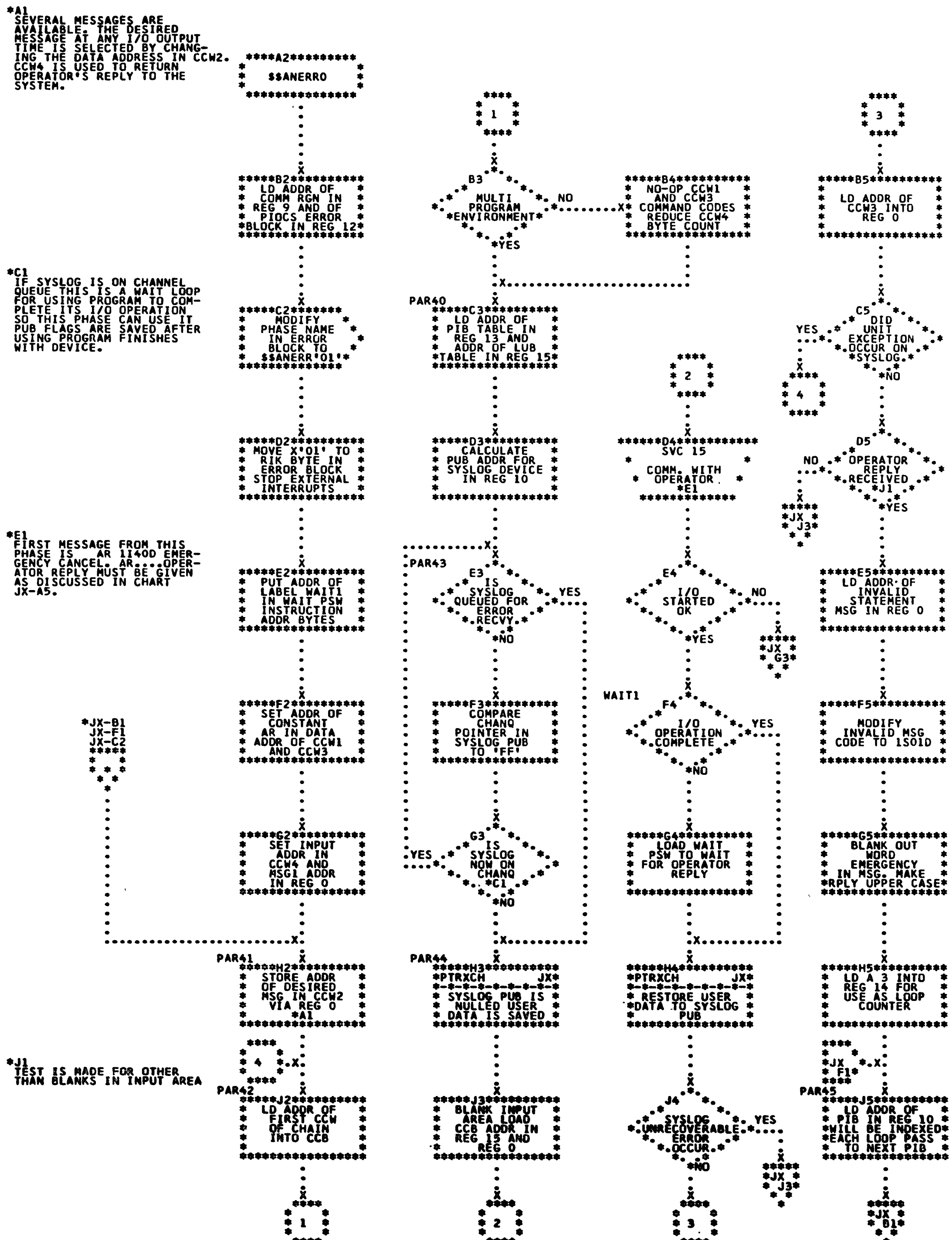
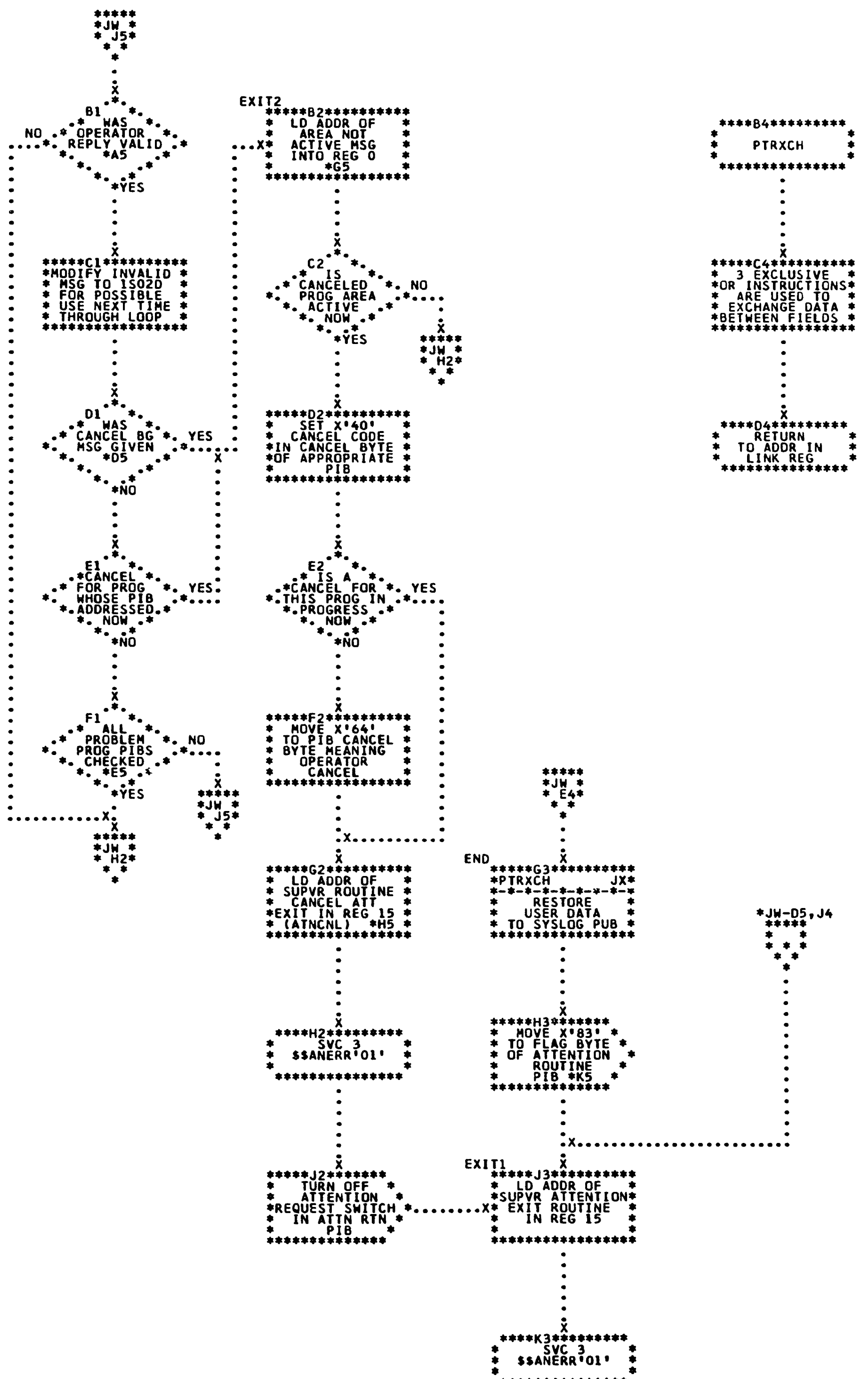


Chart JX. Physical Attention-- Emergency Cancel (Part 2 of 2) $$ANERR0; Refer to Supervisor, Chart 20



*A5
OPERATOR RESPONSES CAN BE
1. CANCEL F1
2. CANCEL F2
3. CANCEL BG
4. CANCEL
REPLY 4 MEANS CANCEL BG PROGRAMS PRIOR TO TEST FOR VALID REPLY REG 0 WAS LOADED WITH ADDRESS OF INVALID STATEMENT MESSAGE. IF REPLY PROVES INVALID, THIS ADDRESS WILL BE PUT IN CCW2 DATA ADDRESS FOR SYSLOG OUTPUT TO OPERATOR SO HE CAN TRY AGAIN TO GIVE A VALID REPLY.

*D5
BLANKS IN PROGRAM IDENTIFICATION PART OF CANCEL MESSAGE MEANS CANCEL ALL PROGRAMS.

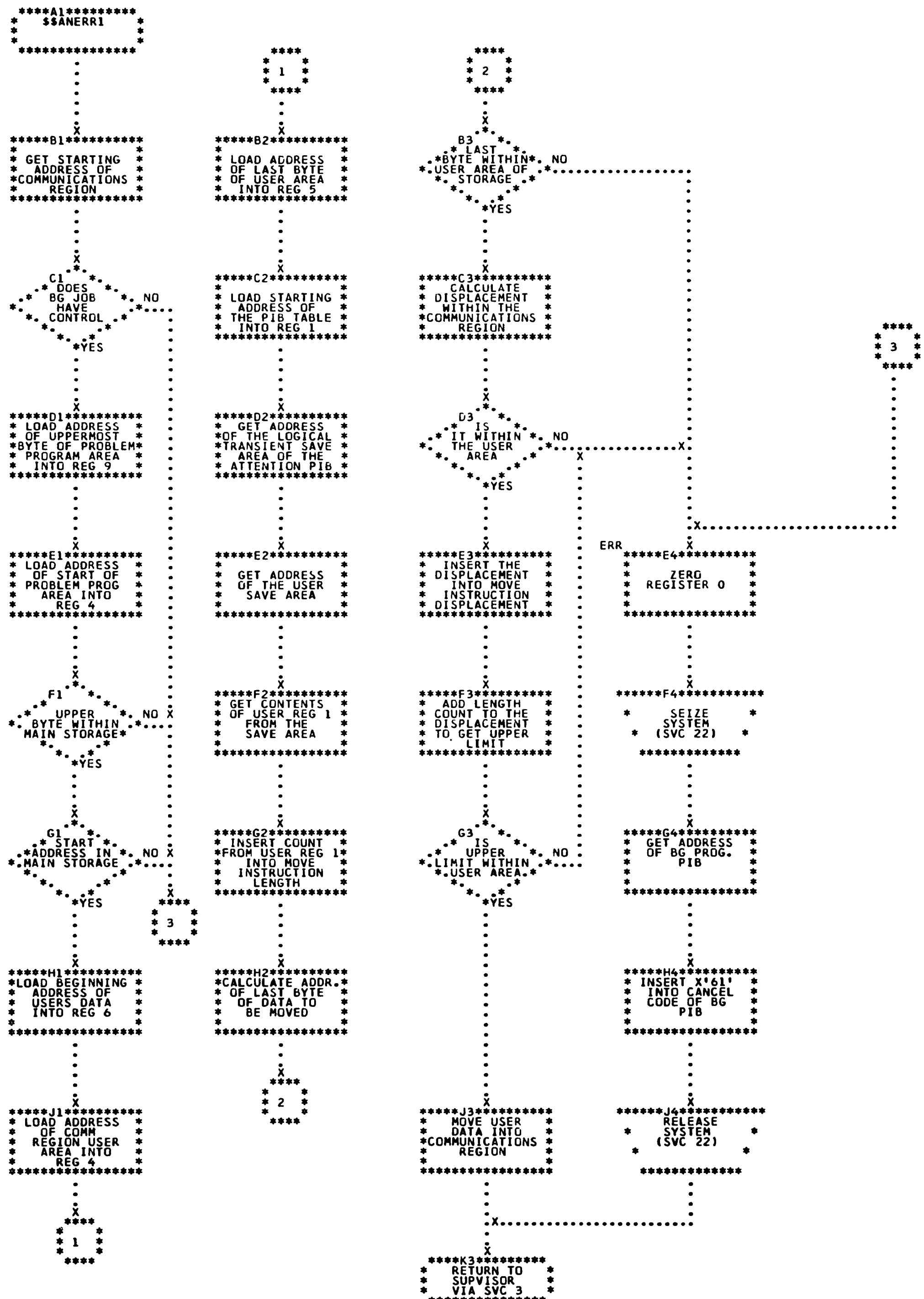
*E5
COUNTER IN REGISTER 14 IS USED TO SEE IF CANCEL MESSAGE WAS CHECKED FOR ALL 3 TYPES OF PROGRAM (F1, F2 AND BG). LOOP IS REPEATED UNTIL ALL TYPES OF PROGRAM ARE CHECKED.

*G5
IF PROGRAM AREA IS NOT NOW ACTIVE, SYSLOG IS USED TO OUTPUT 'AREA NOT ACTIVE' MESSAGE.

*H5
A RETURN TO SUPERVISOR IS TAKEN TO PROCESS THE PROGRAM CANCEL, AFTER WHICH CONTROL RETURNS TO NEXT SEQUENTIAL INSTRUCTION OF THIS A-TRANSIENT PHASE.

*K5
X'83' IN FLAG BYTE MEANS REGISTERS ARE STORED, ATTENTION ROUTINE IS ACTIVE AND NOT SVC 7 BOUND. ATTENTION ROUTINE WILL BE CALLED AGAIN.

Chart JY. Move Data to Communications Region ($$ANERR1)



## Chart KA. Nonresident Attention/Initiator Root Phase ($$BATTNA); Refer to Supervisor, Chart 21

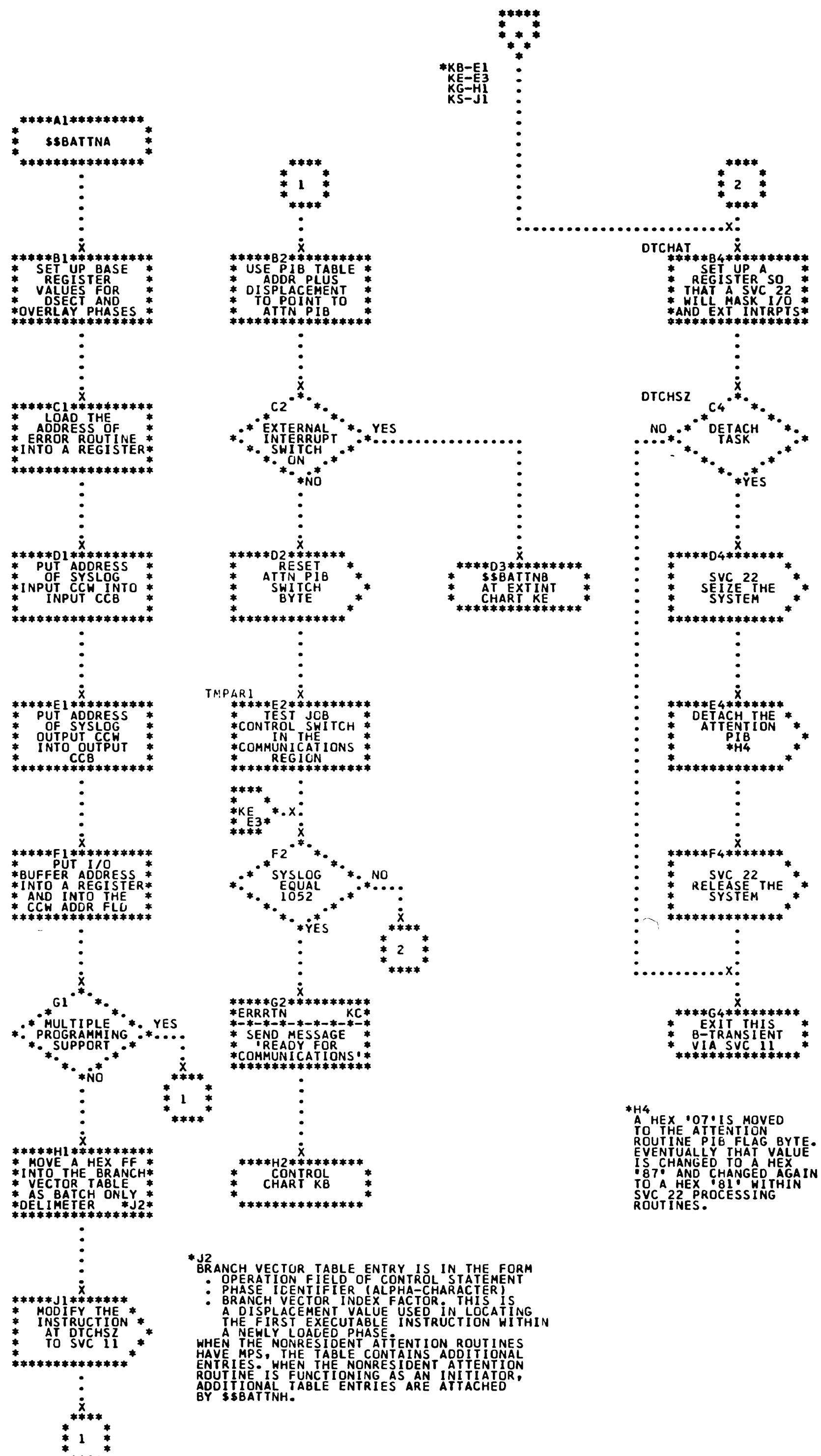


Chart KB. Control Routine ($$BATTNA); Refer to Supervisor, Chart 21

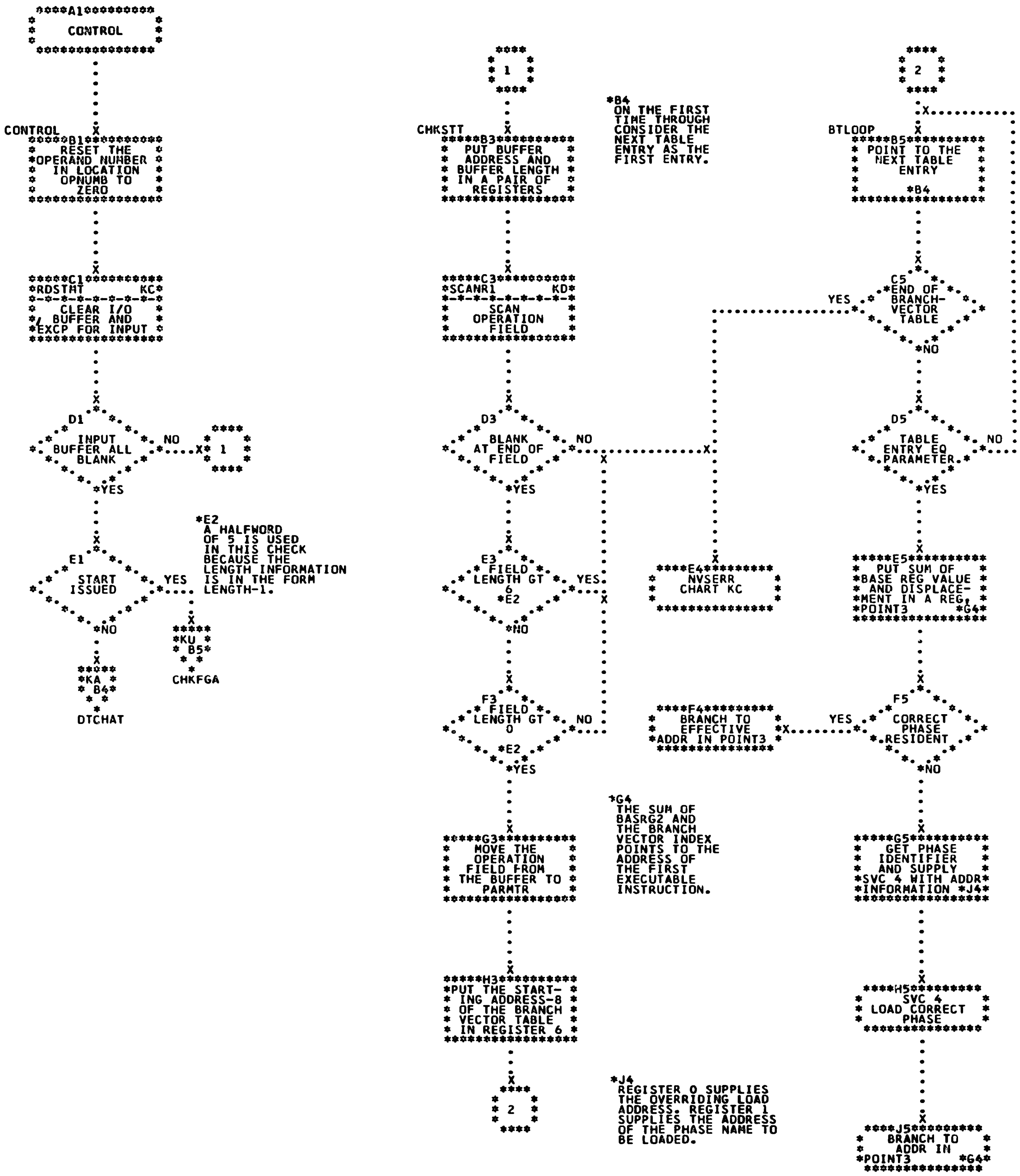
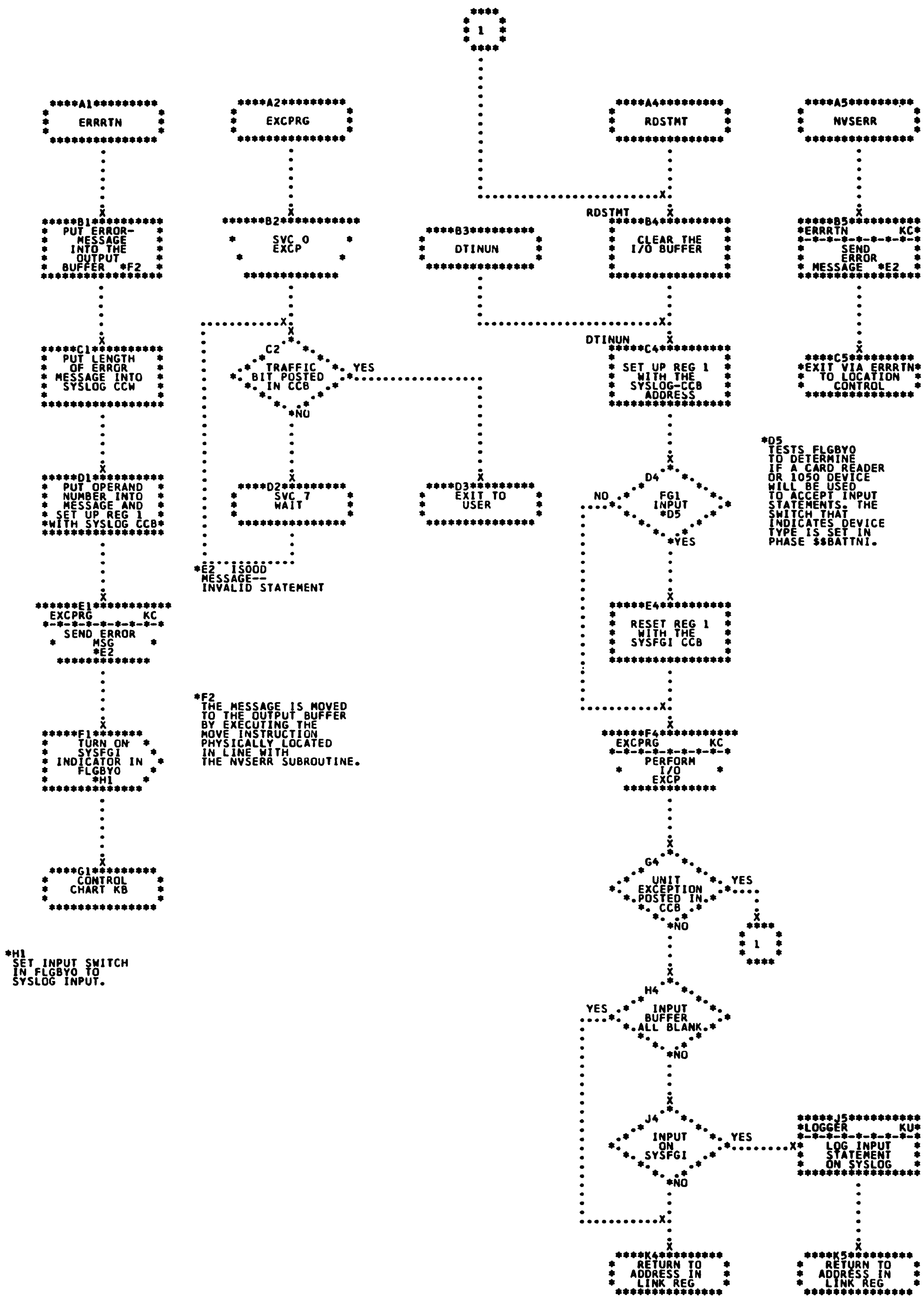


Chart KC. Root Phase Subroutines ($$BATTNA); Refer to Supervisor, Chart 21



## Chart KD. General Scan Routines ($$BATTNA); Refer to Supervisor, Chart 21

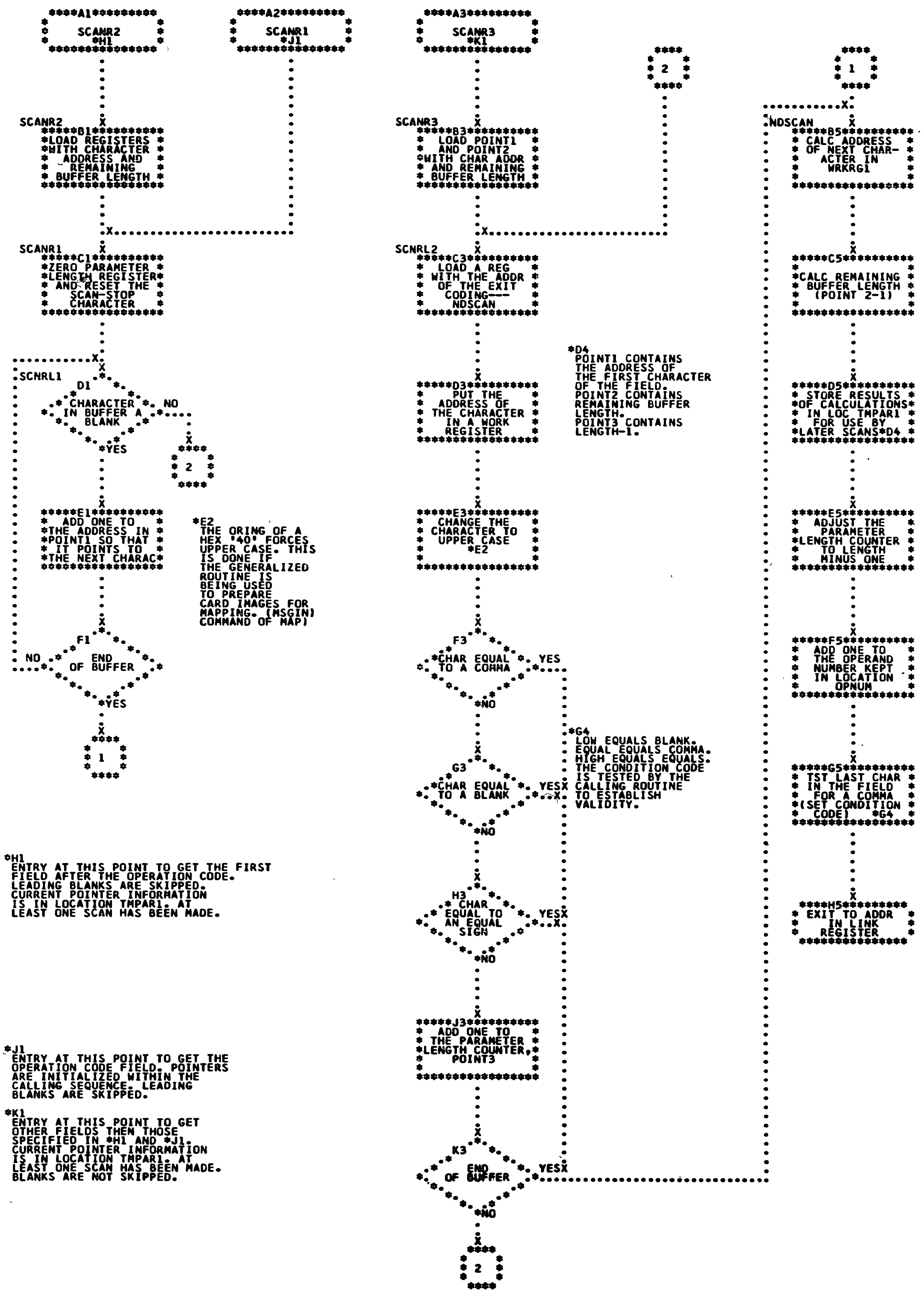


Chart KE. MSG Statement Processor ($$BATTNB); Refer to Supervisor, Chart 24

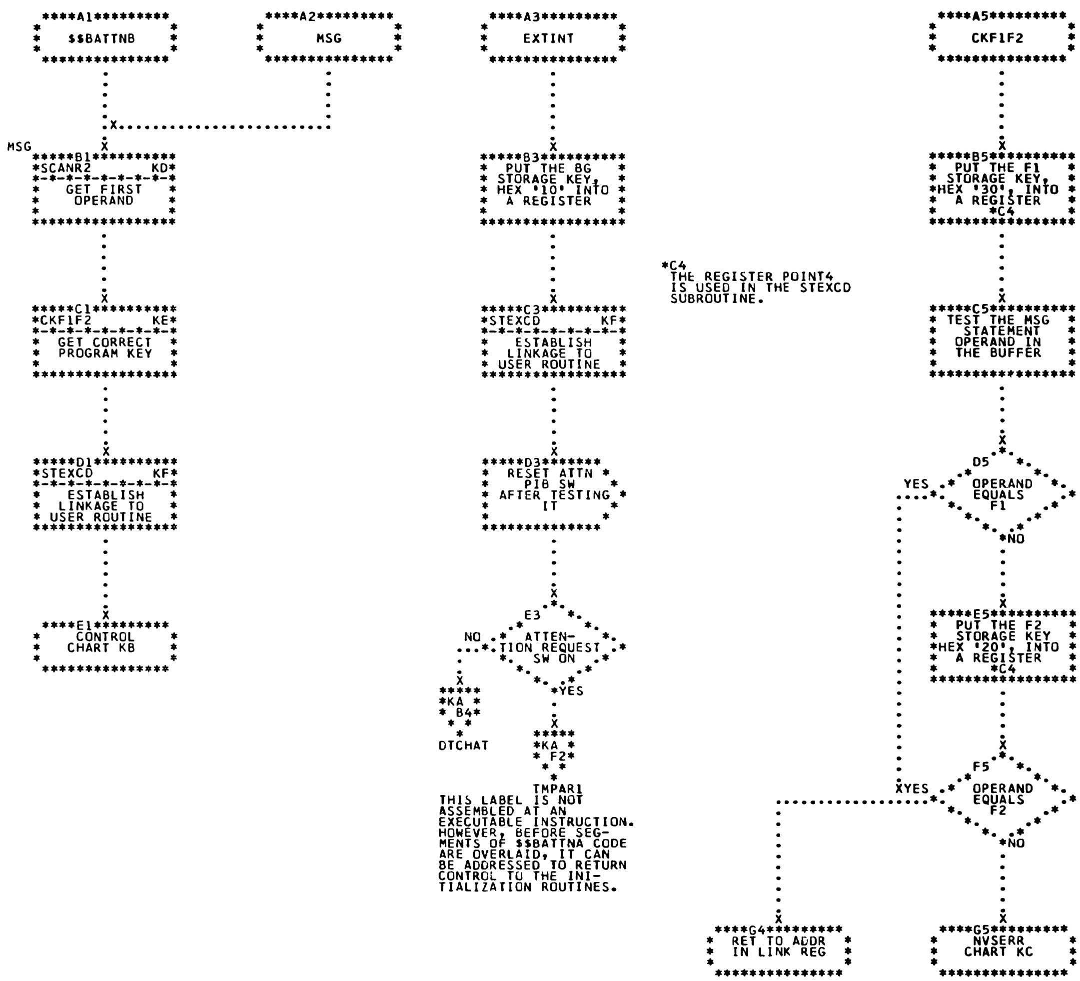


Chart KF. Set Operator Communications and Exit Table Linkage ($$BATTNB); Refer to Supervisor, Chart 24

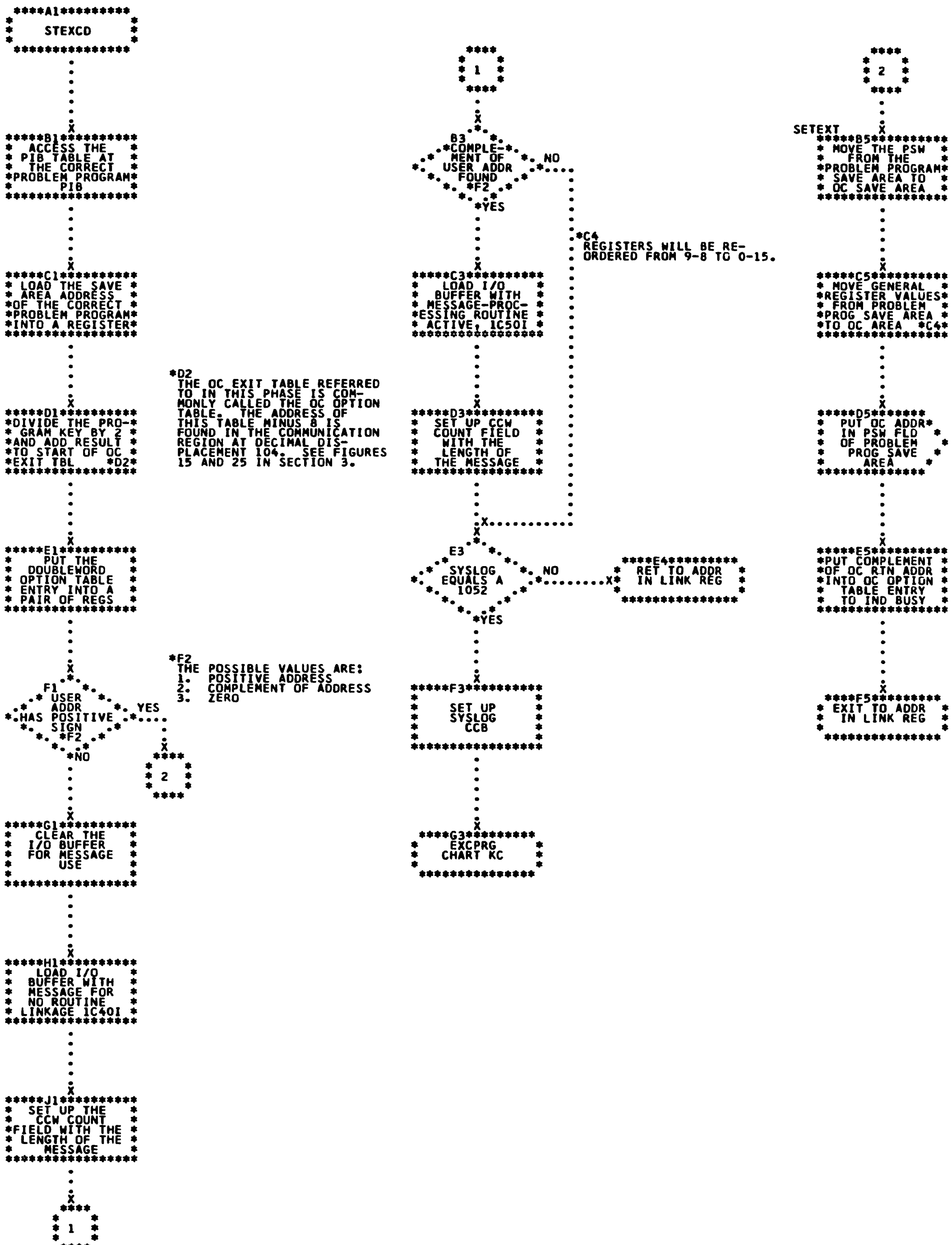


Chart KG. CANCEL Statement Processor ($$BATTNC); Refer to Supervisor, Chart 24

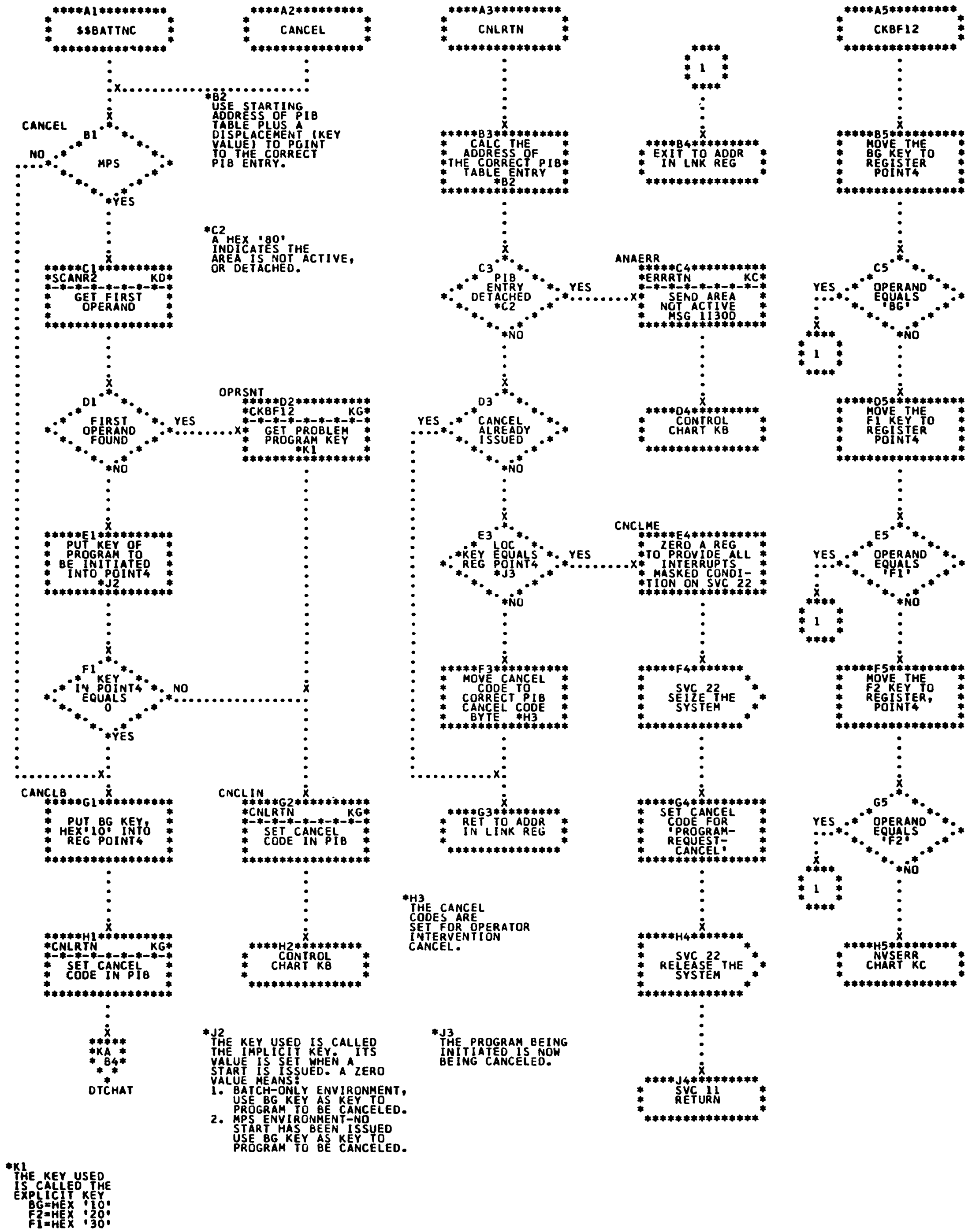


Chart KH. PAUSE, LOG, and NOLOG Statement Processors ($$BATTNC); Refer to Supervisor, Chart 24

```
 ****A1*********          ****A3*********          ****A5*********
 *             *          *             *          *             *
 *    PAUSE    *          *     LOG     *          *    NOLOG    *
 *             *          *             *          *             *
 ***************          ***************          ***************
        :                        :                        :
        :                        :                        :
        X                        X                        X
*****B1********          *****B3********          *****B5********
*  TURN ON    *          *  TURN ON    *          *  TURN OFF   *
* PAUSE SWITCH *         * LOG SWITCH   *         * LOG SWITCH   *
* IN JOB CTRL   *        * IN JOB CTRL   *        * IN JOB CTRL   *
*  SWITCH 0    *         *  SWITCH 0    *         *  SWITCH 0    *
*             *          *             *          *             *
***************          ***************          ***************
        :                        :                        :
        :                        :                        :
        X                        X                        X
 ****C1*********          ****C3*********          ****C5*********
 *   CONTROL   *          *   CONTROL   *          *   CONTROL   *
 *  CHART KB   *          *  CHART KB   *          *  CHART KB   *
 *             *          *             *          *             *
 ***************          ***************          ***************
```

## Chart KJ. MAP Statement Processor ($$BATTND); Refer to Supervisor, Chart 24

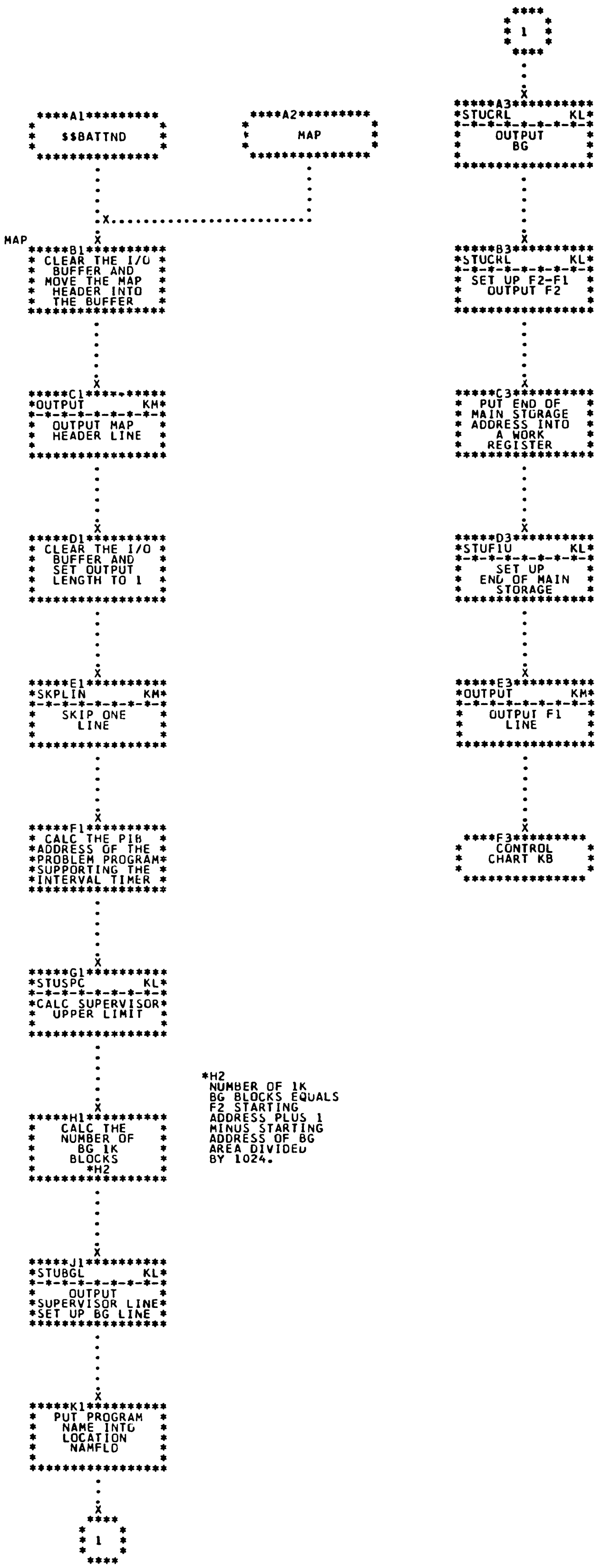


Chart KL. Output MAP Subroutines (Part 1 of 2) $$BATTND; Refer to Supervisor, Chart 24

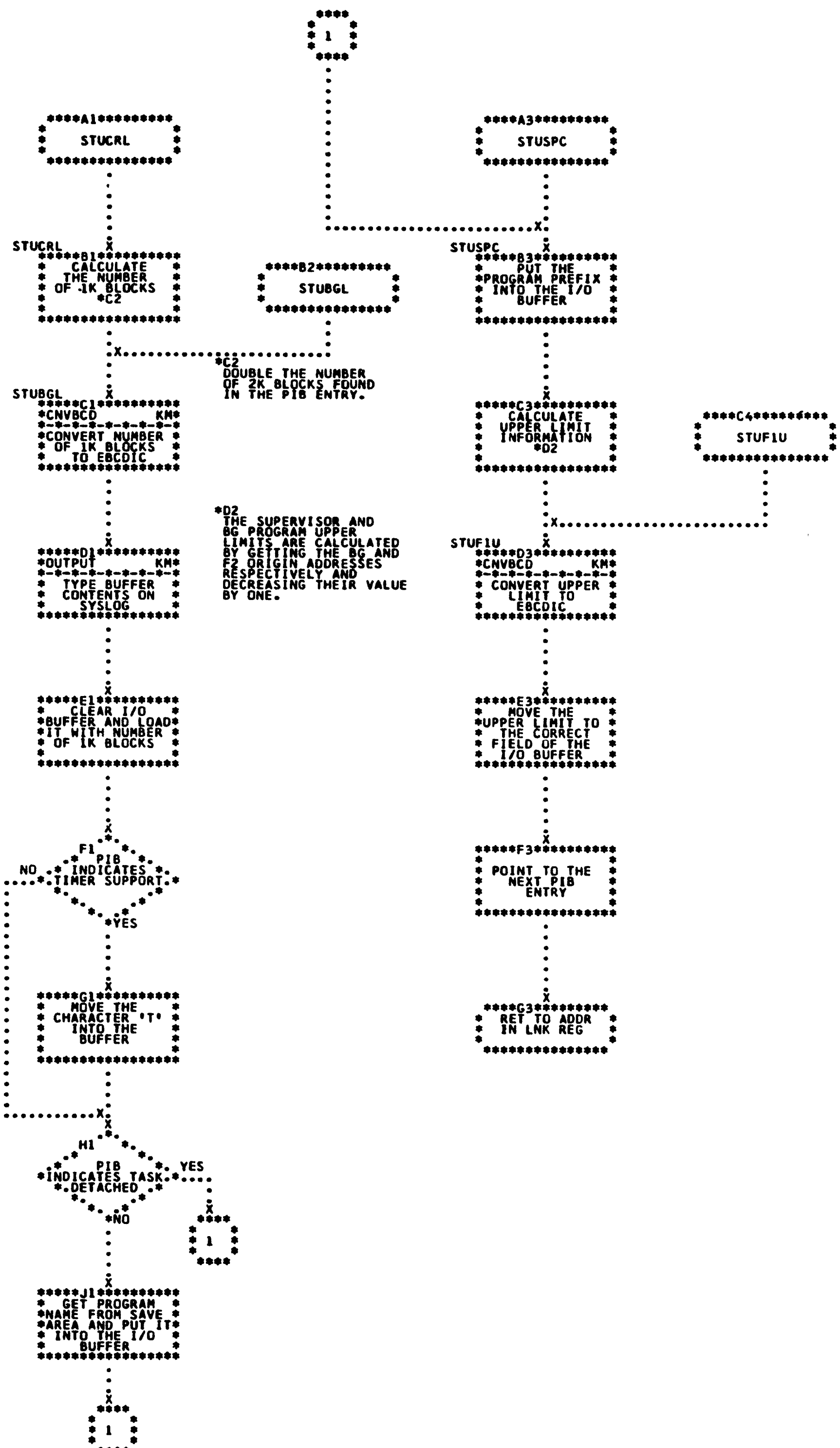


Chart KM. Output MAP Subroutines (Part 2 of 2) $$BATTND; Refer to Supervisor, Chart 24

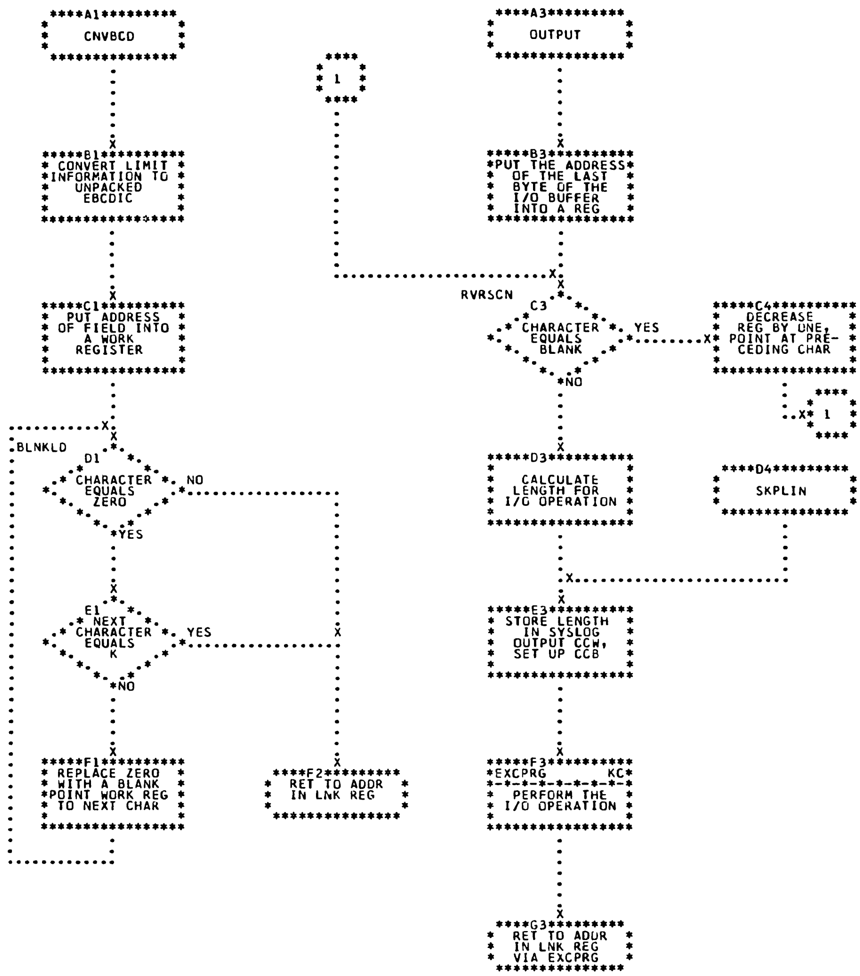


Chart KN. ALLOC Statement Processor, Part 1; ($$BATTNE);
Refer to Supervisor, Chart 25

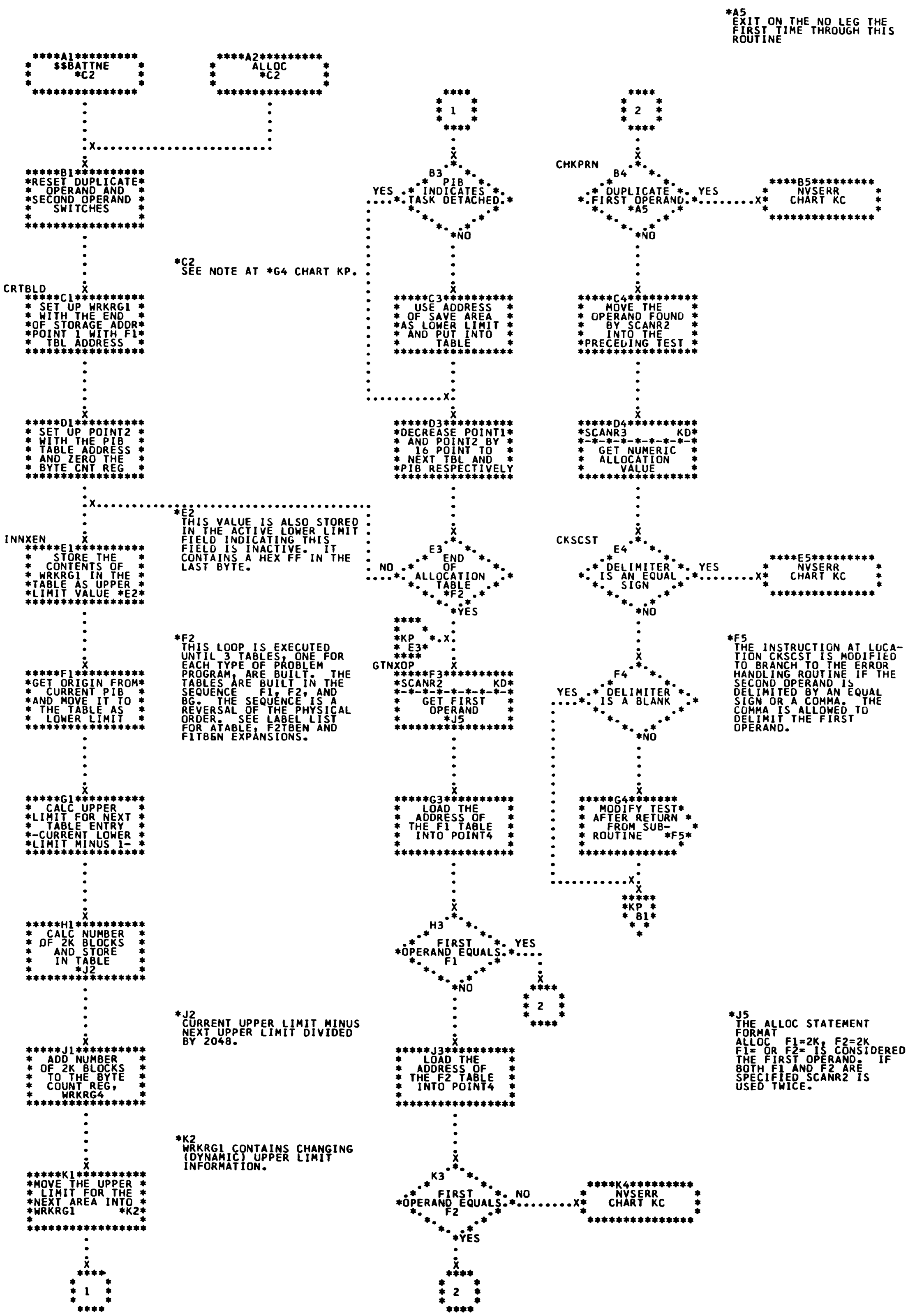
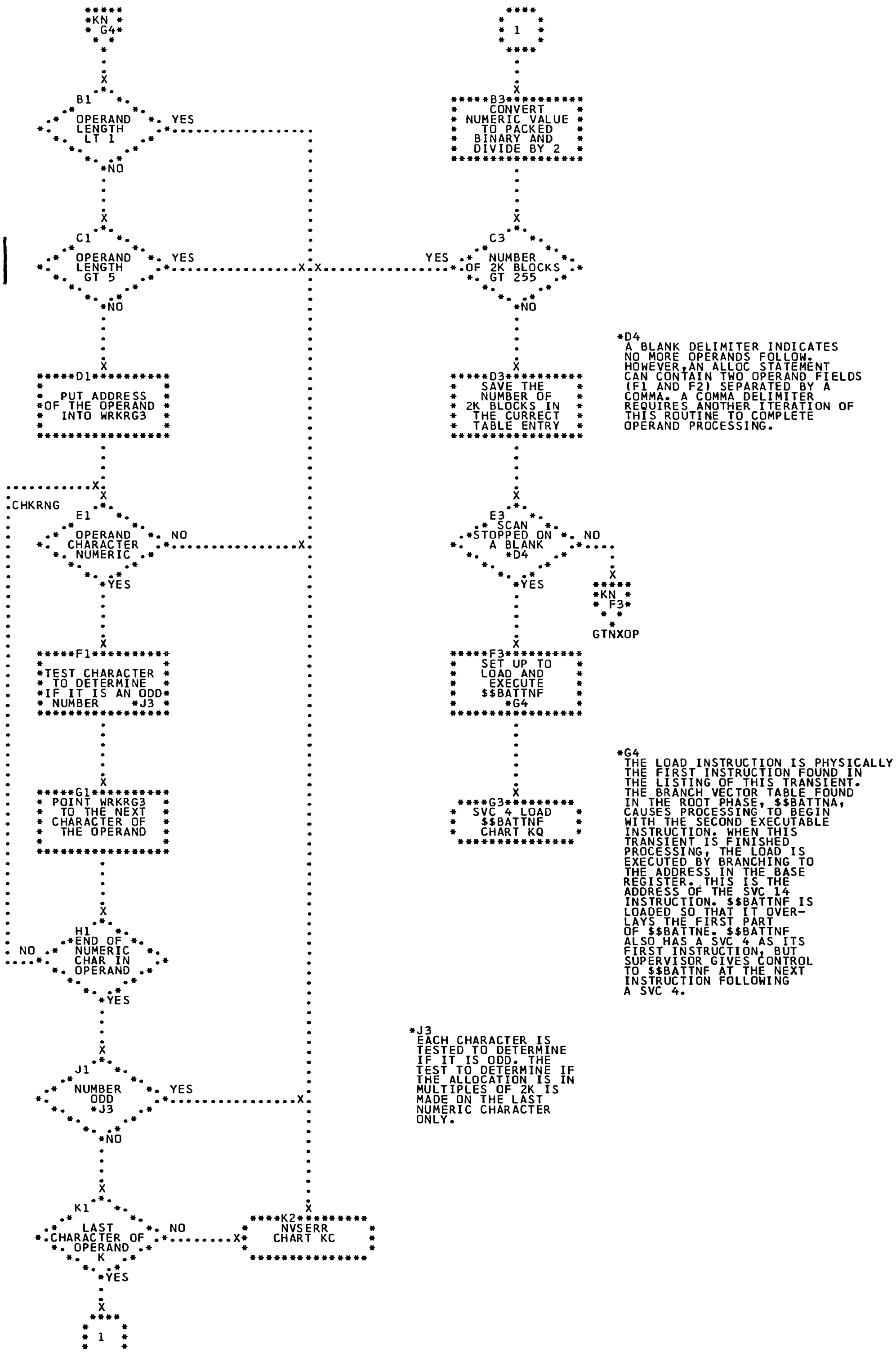


Chart KP. ALLOC Statement Operand Validity Checking;
$$BATTNE; Refer to Supervisor, Chart 25



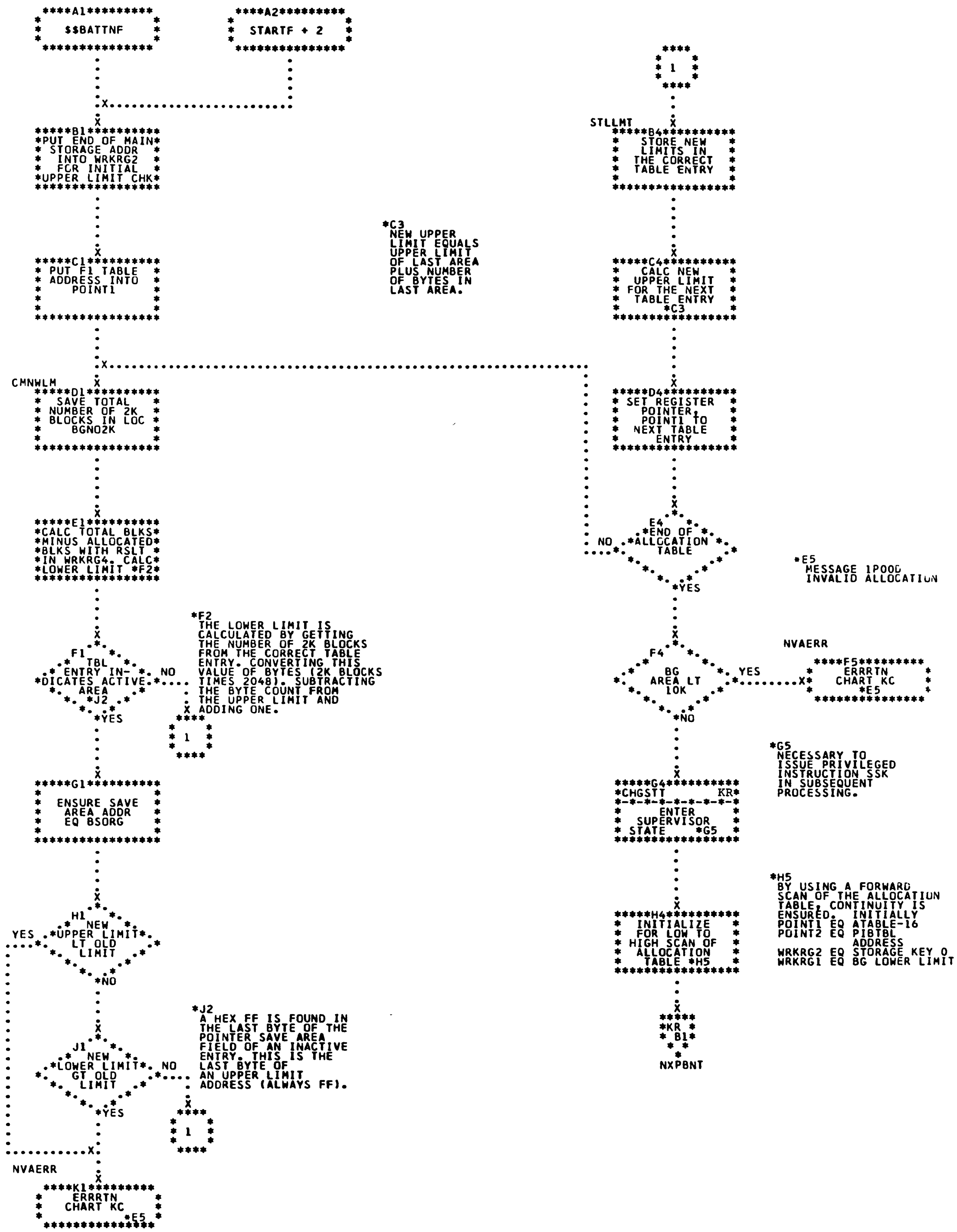


Chart KR. ALLOC Statement Processor, Part 2 (Part 2 of 2)
$$BATTNF; Refer to Supervisor, Chart 25

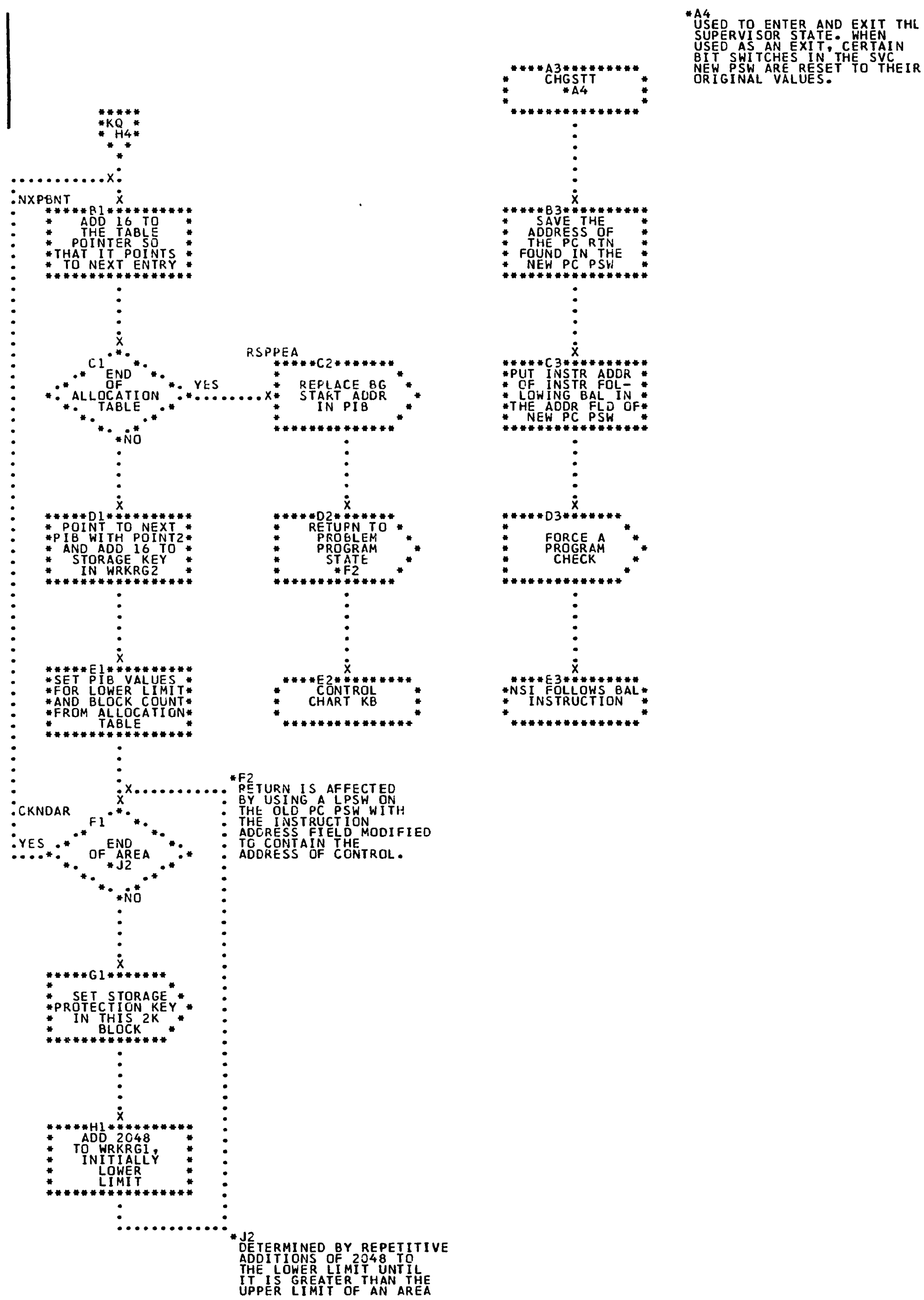


Chart KS. START Statement Processor, Part 1 ($$BATTNG); Refer to Supervisor, Chart 25

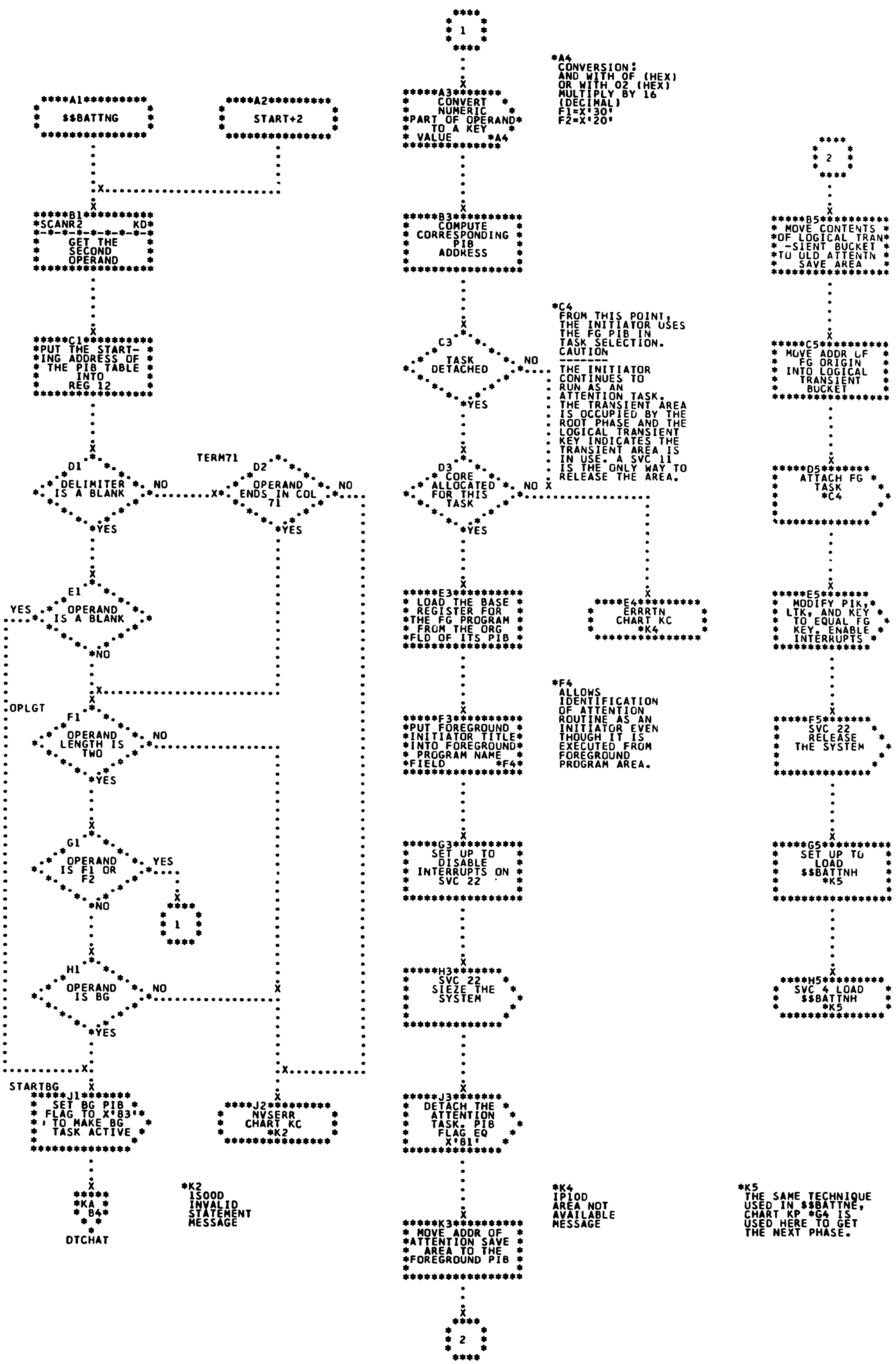


Chart KT. START Statement Processor, Part 2; $$BATTNH; Refer to Supervisor, Chart 25

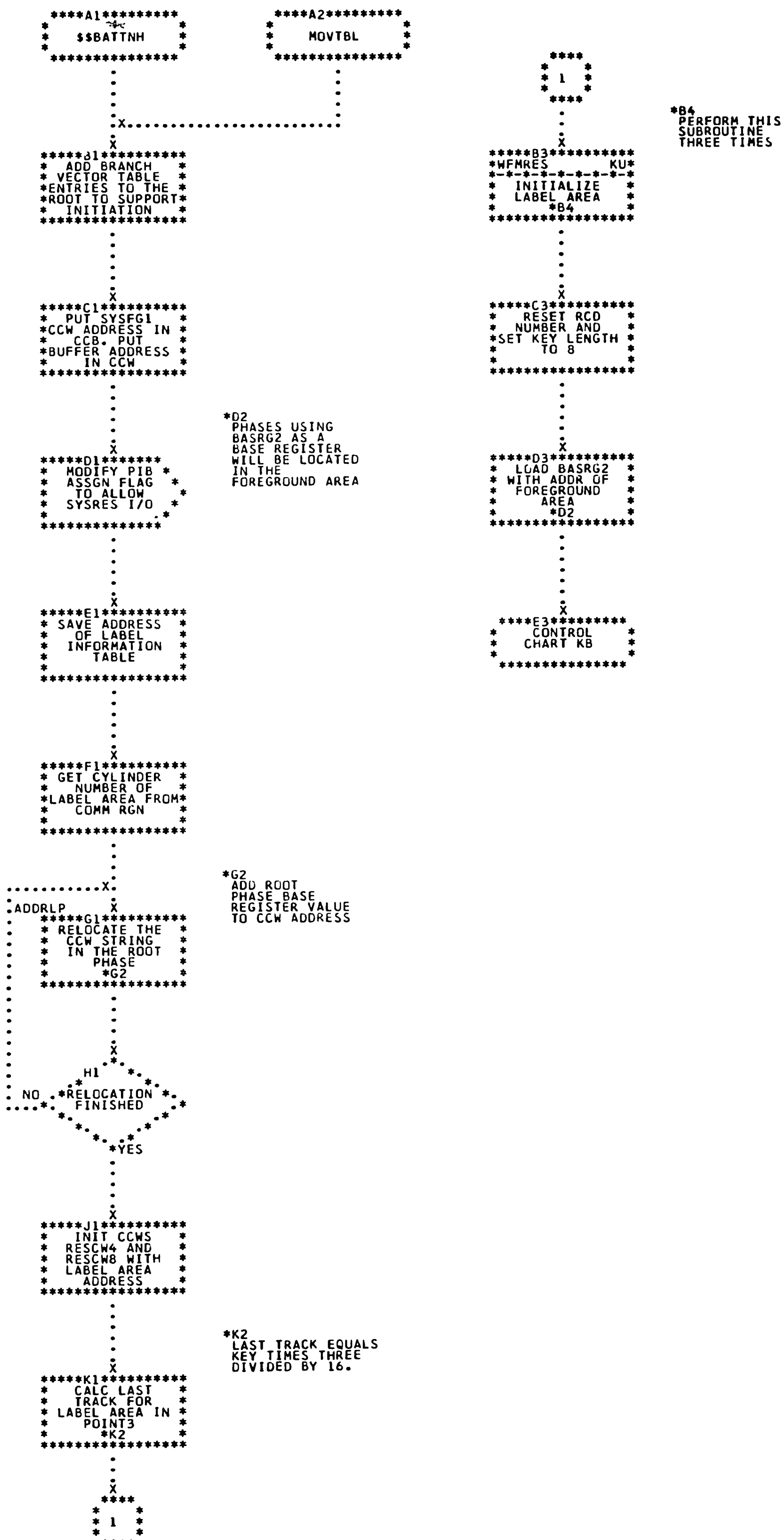


Chart KU. START Processor Subroutines $$BATTNH; Refer to Supervisor, Chart 25

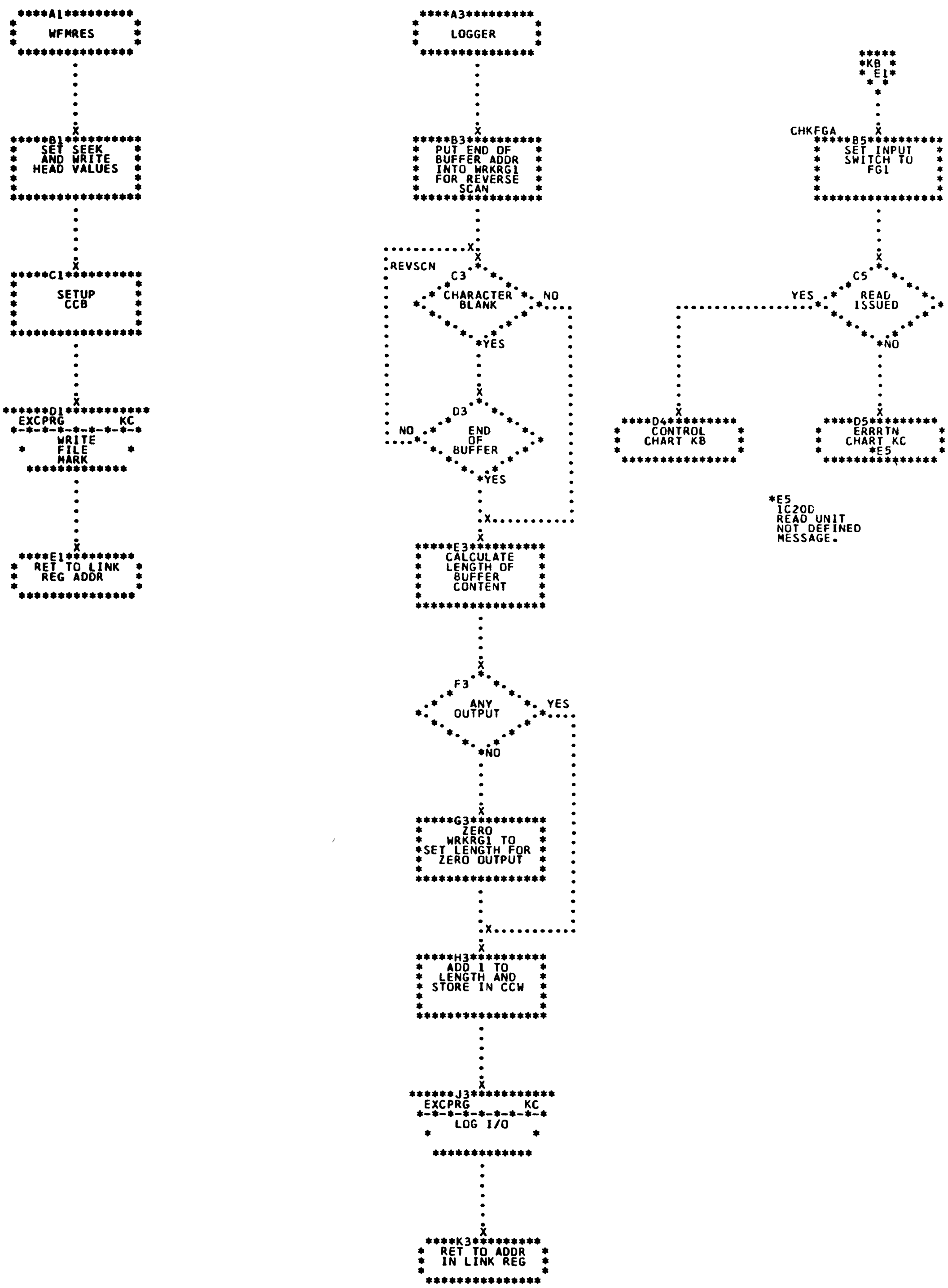


Chart KV. ASSGN Statement Processor (Part 1 of 2) $$BATTNI; Refer to Supervisor, Chart 22

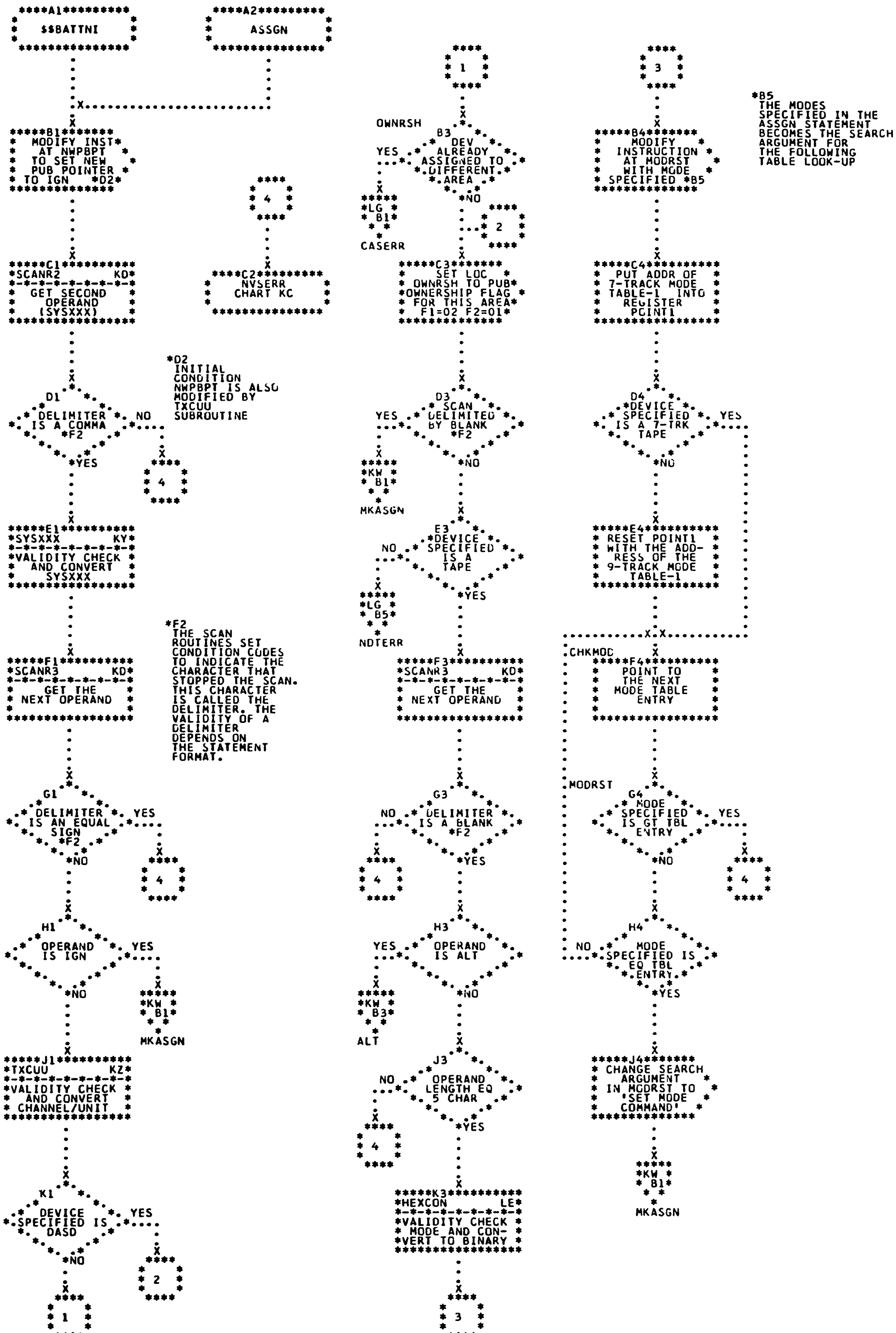
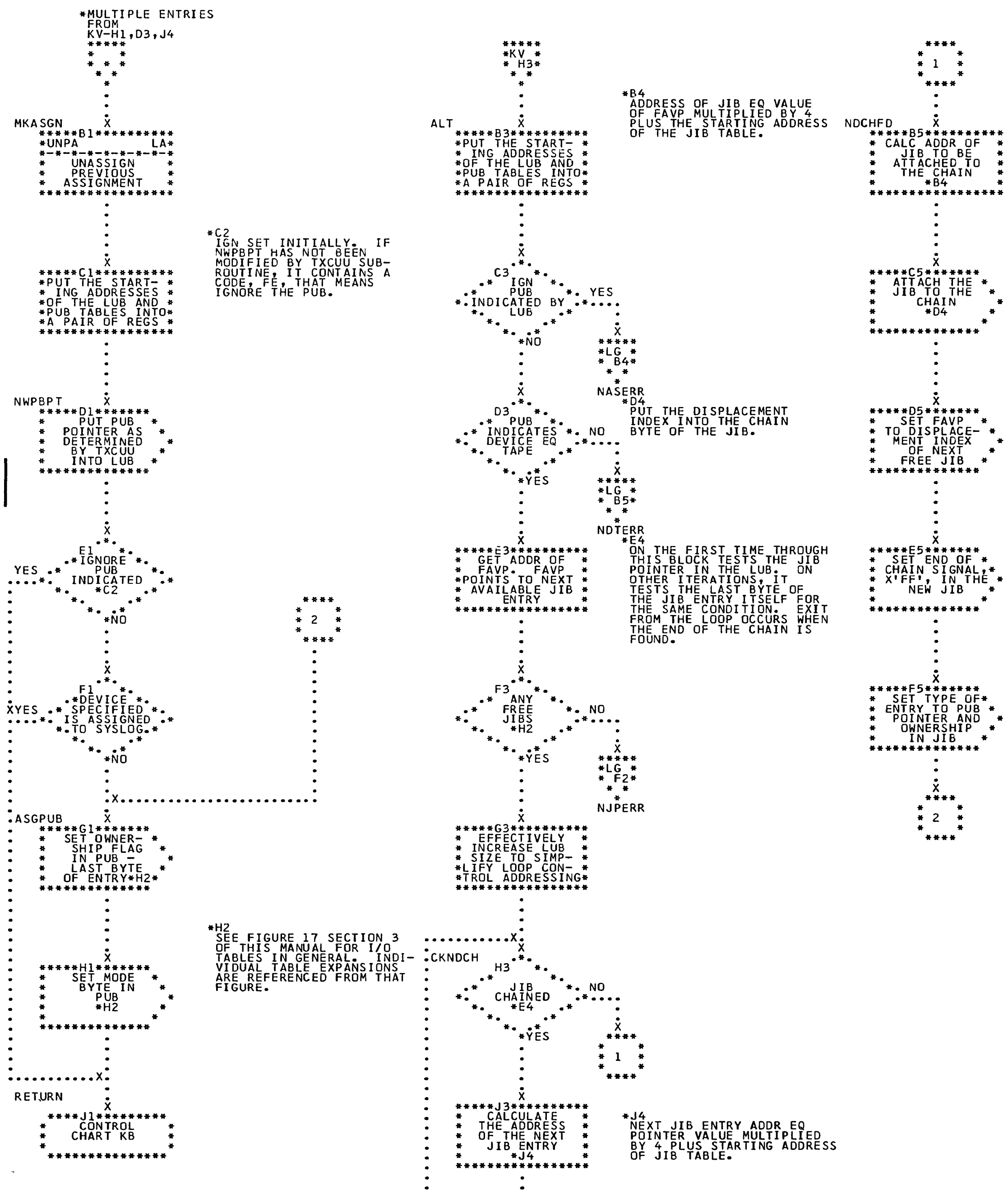
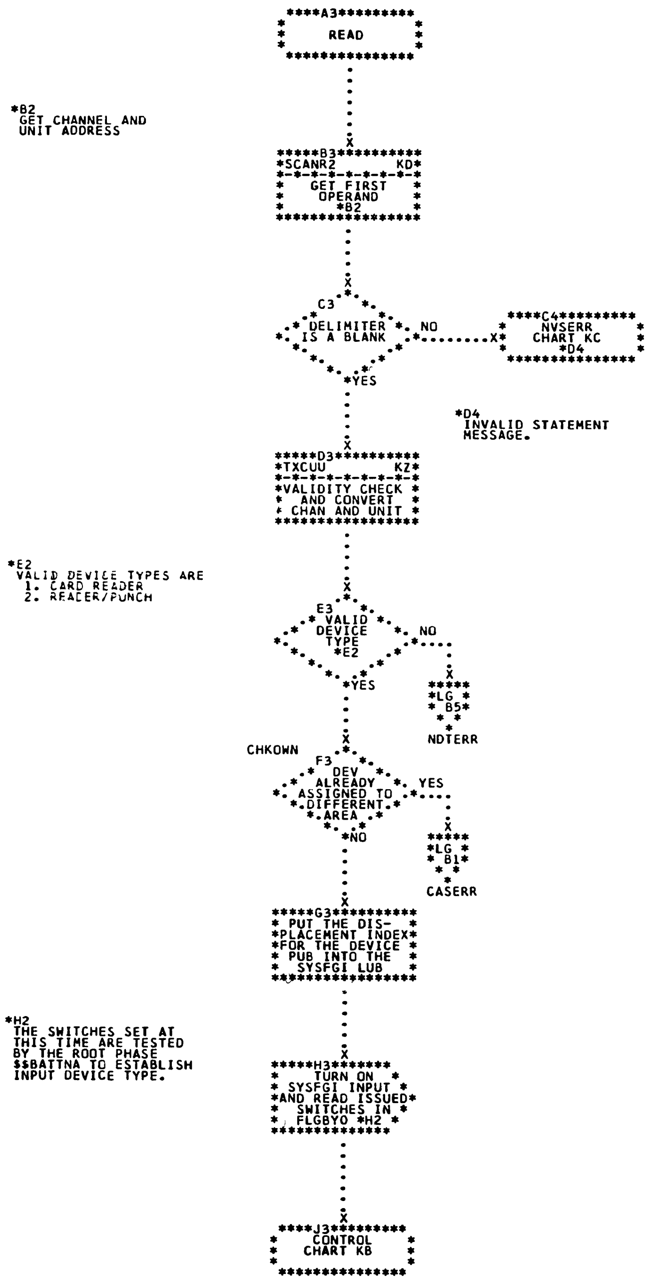


Chart KW. ASSGN Statement Processor (Part 2 of 2) $$BATTNI;
Refer to Supervisor, Chart 22



## Chart KX. READ Statement Processor $$BATTNI; Refer to Supervisor, Chart 22



## Chart KY. Validate SYSXXX Subroutine $$BATTNI; Refer to Supervisor, Chart 22

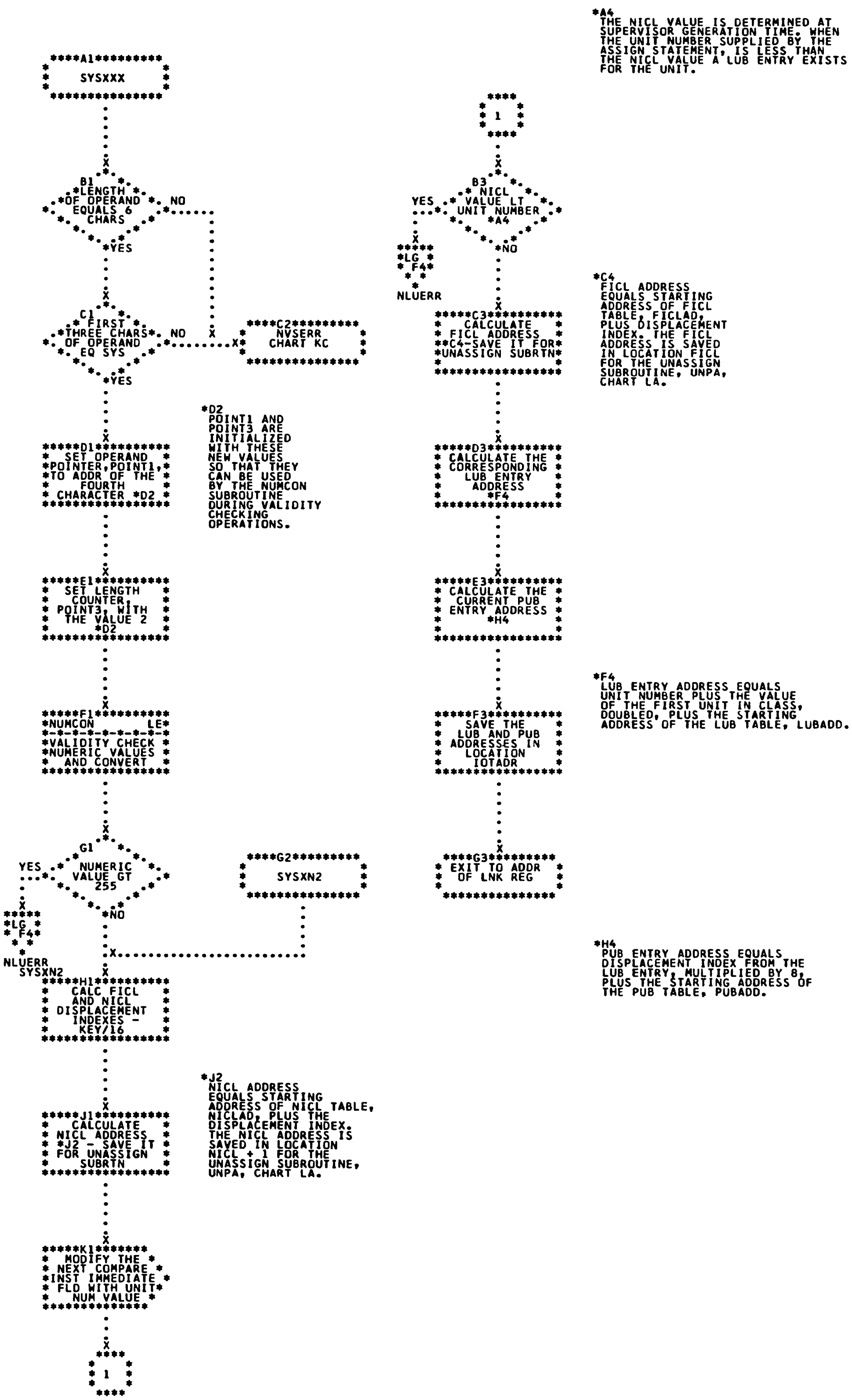
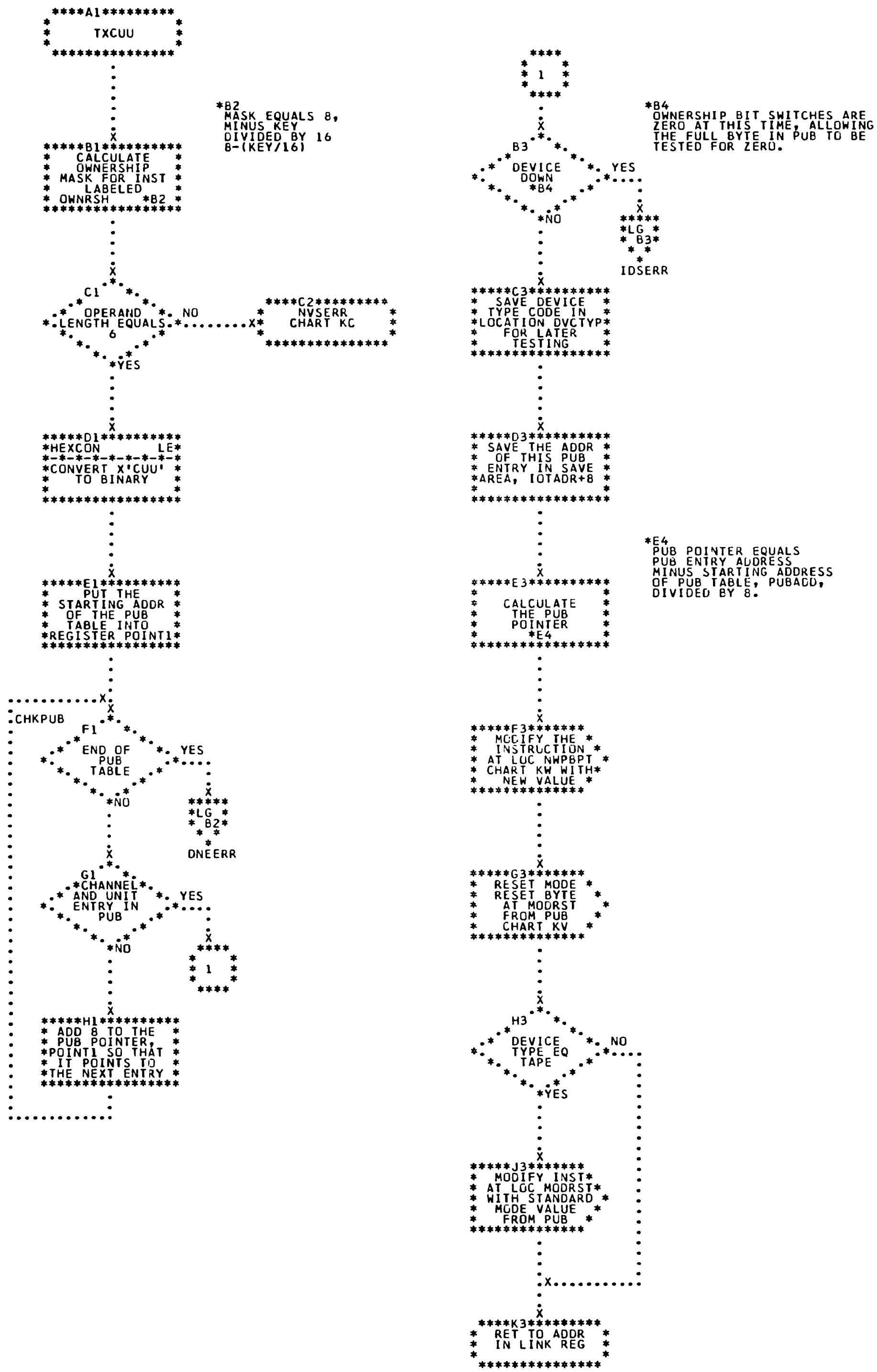


Chart KZ. Validity Check Channel and Unit and Convert to Binary; $$BATTNI; REFER TO Supervisor, Chart 22



## Chart LA. Unassign Subroutine $$BATTNI; Refer to Supervisor, Chart 22

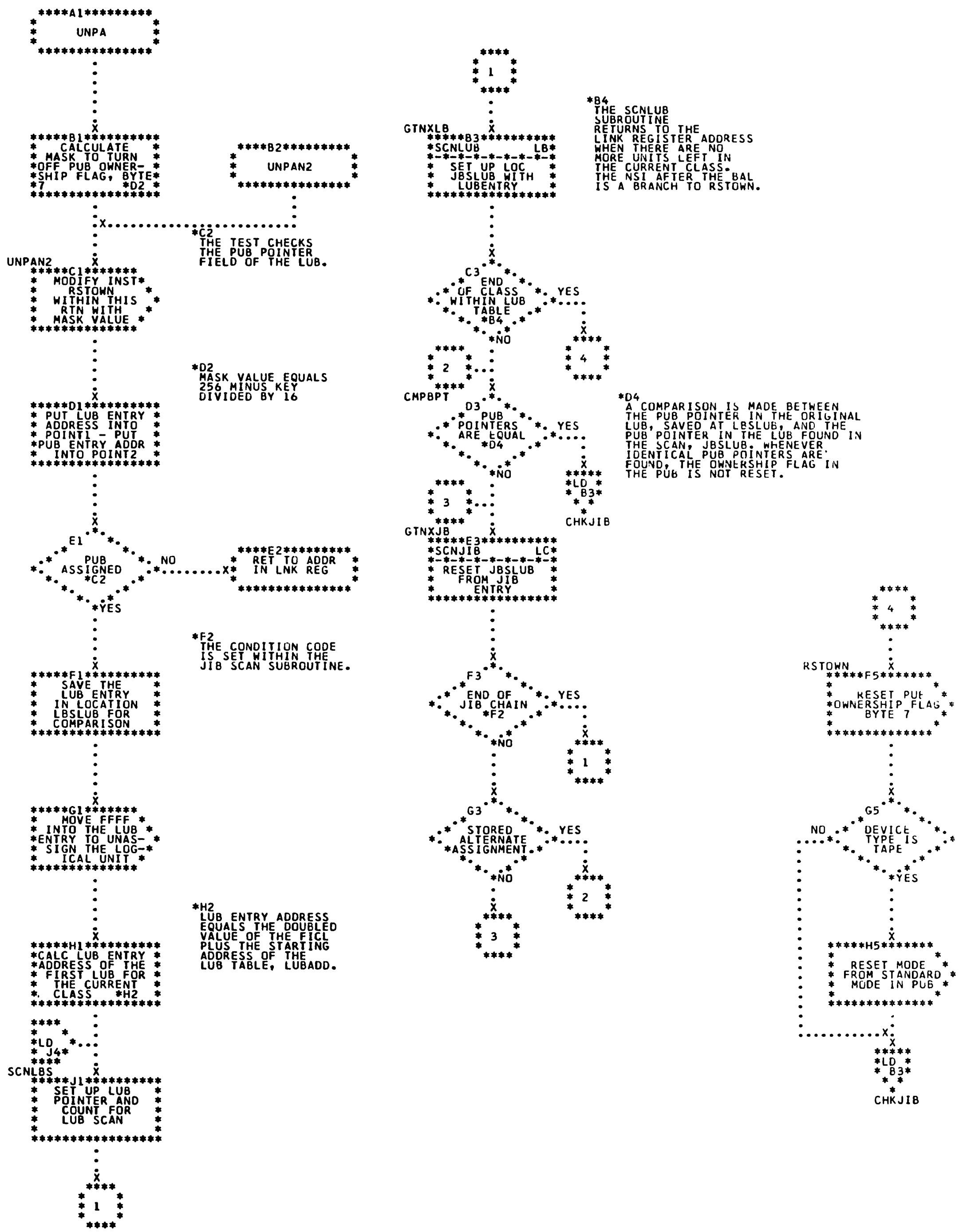


Chart LB. Scan LUBs in Class Subroutine $$BATTNI; Refer to Supervisor, Chart 22

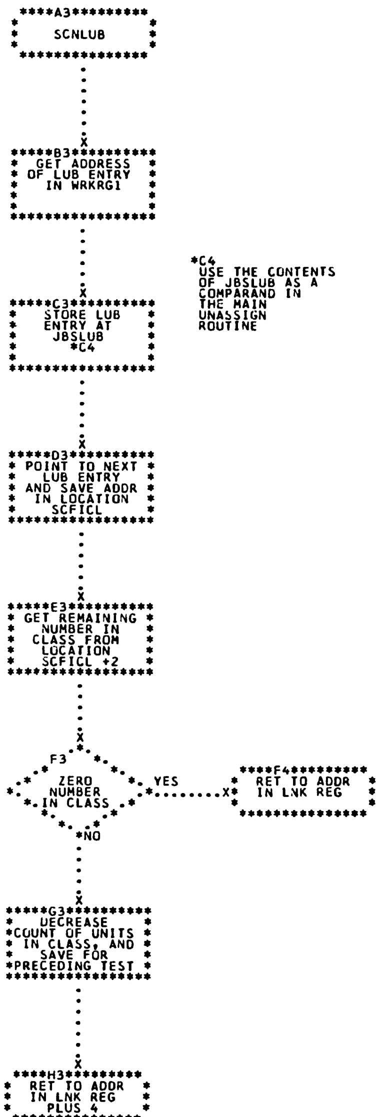


Chart LC. Scan JIB's Subroutine $$BATTNI; Refer to Supervisor, Chart 22

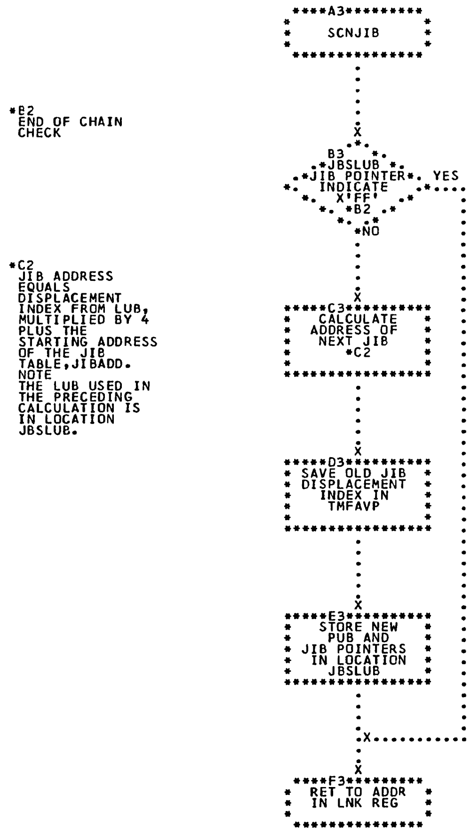


Chart LD. Reset Free List Routine $$BATTNI; Refer to Supervisor, Chart 22

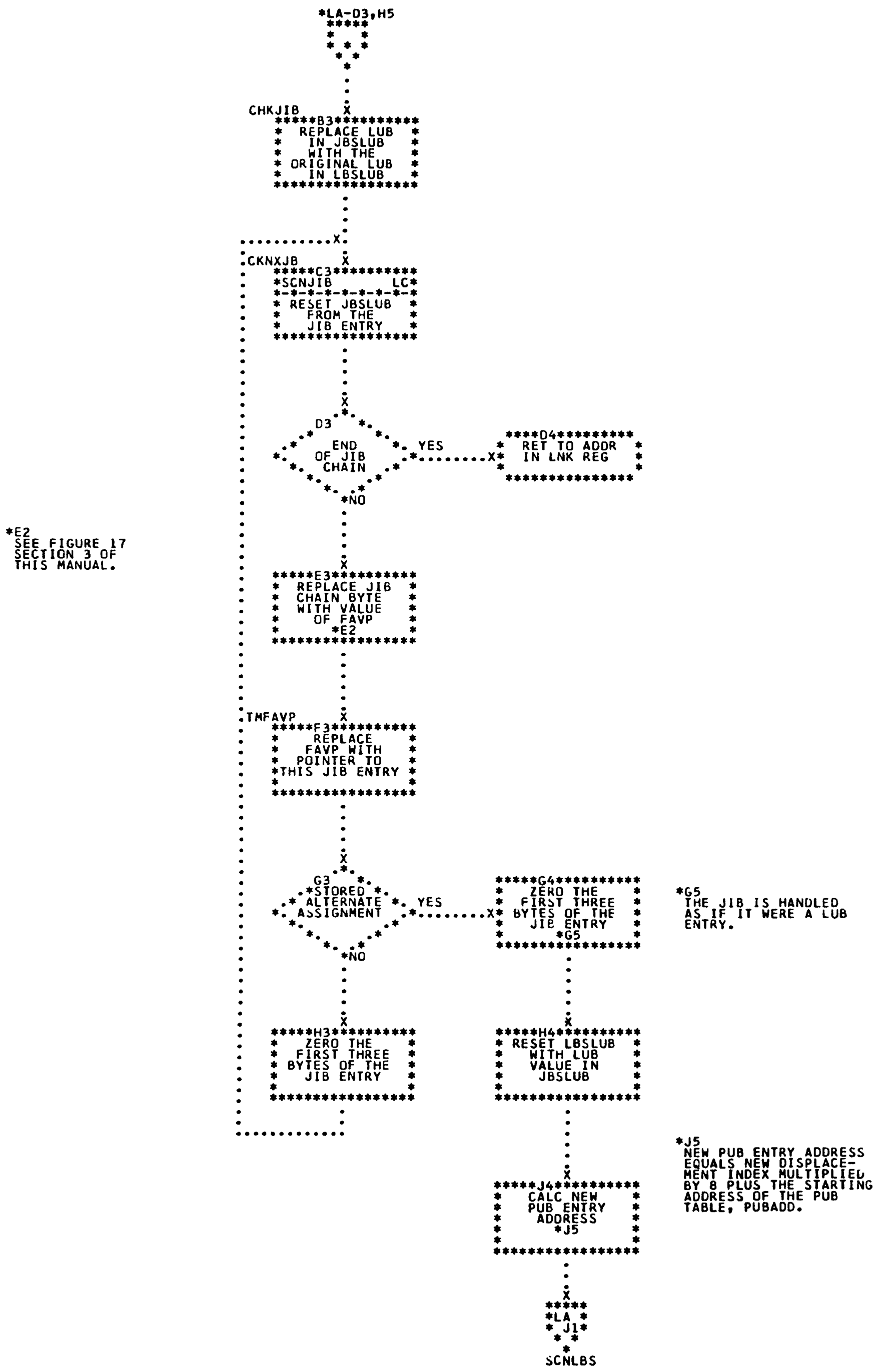


Chart LE. ASSGN Processor Subroutines (Part 1 of 2)
$$BATTNI; Refer to Supervisor, Chart 22

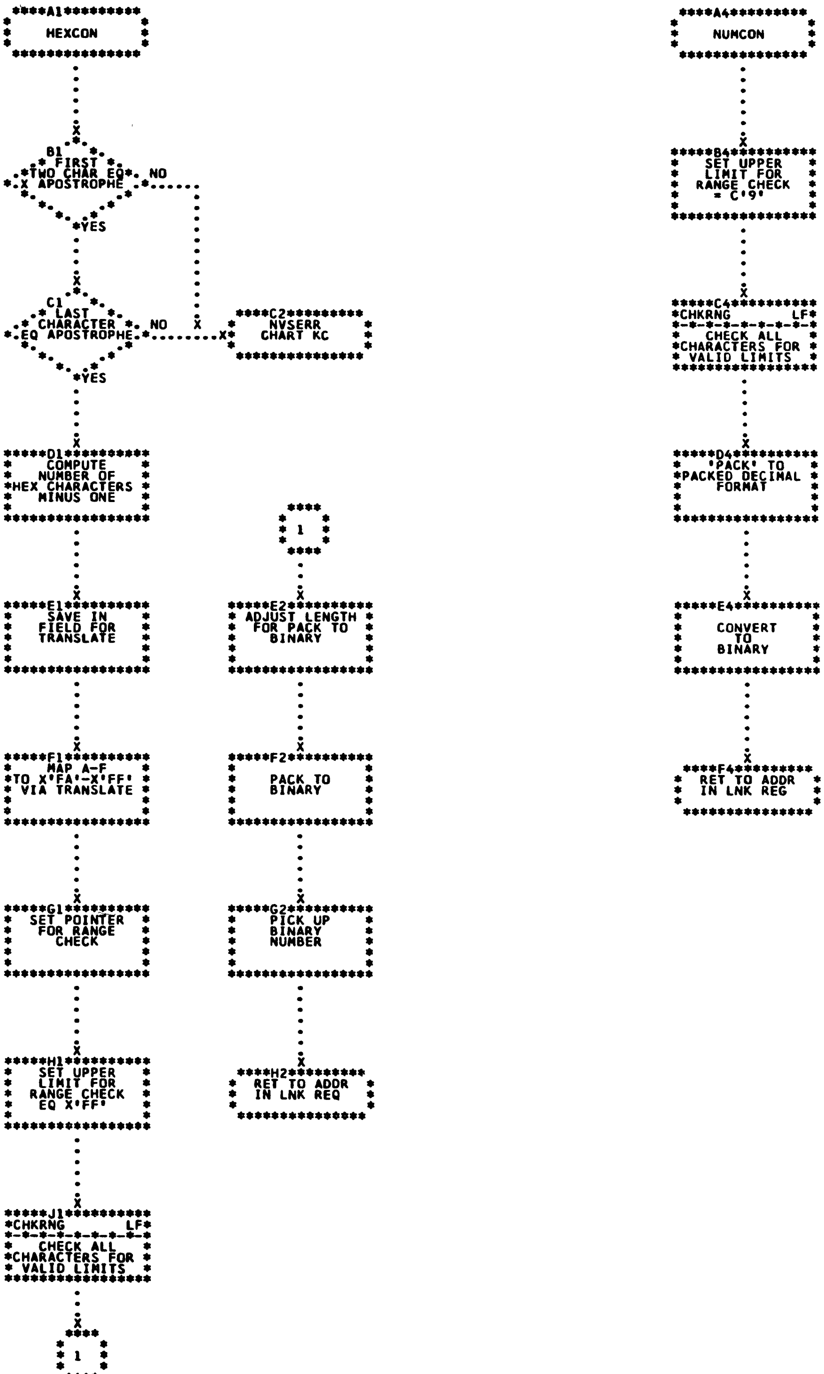


Chart LF. ASSGN Processor Subroutines (Part 2 of 2)
$$BATTNI; Refer to Supervisor Chart 22

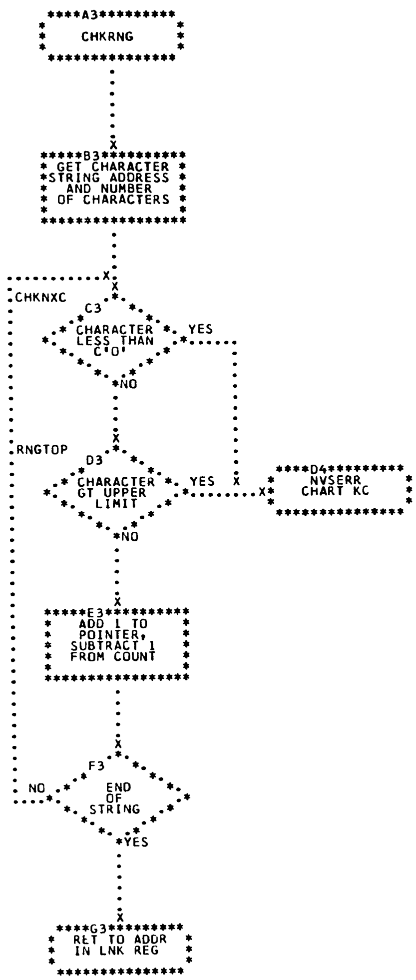


Chart LG. Common Error Exits $$BATTNI; Refer to Supervisor, Chart 22

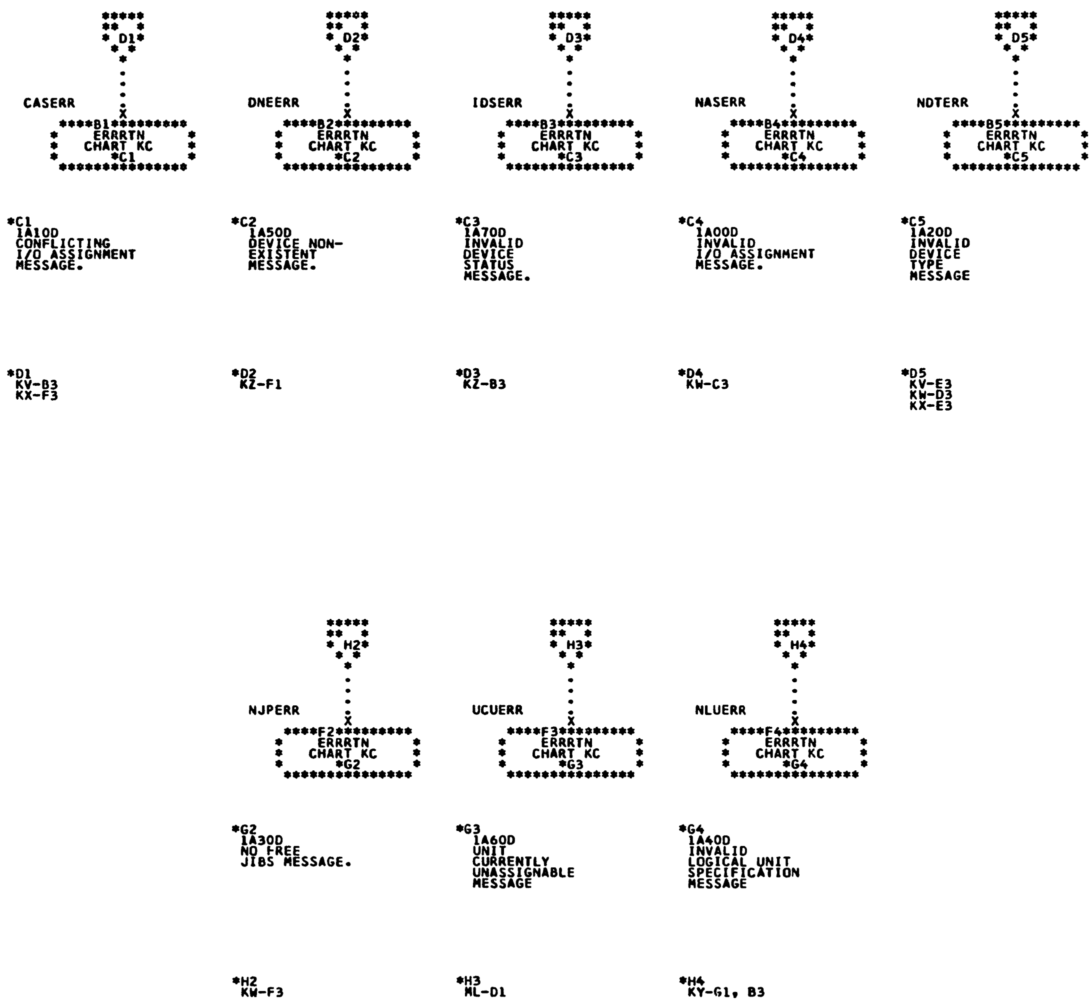
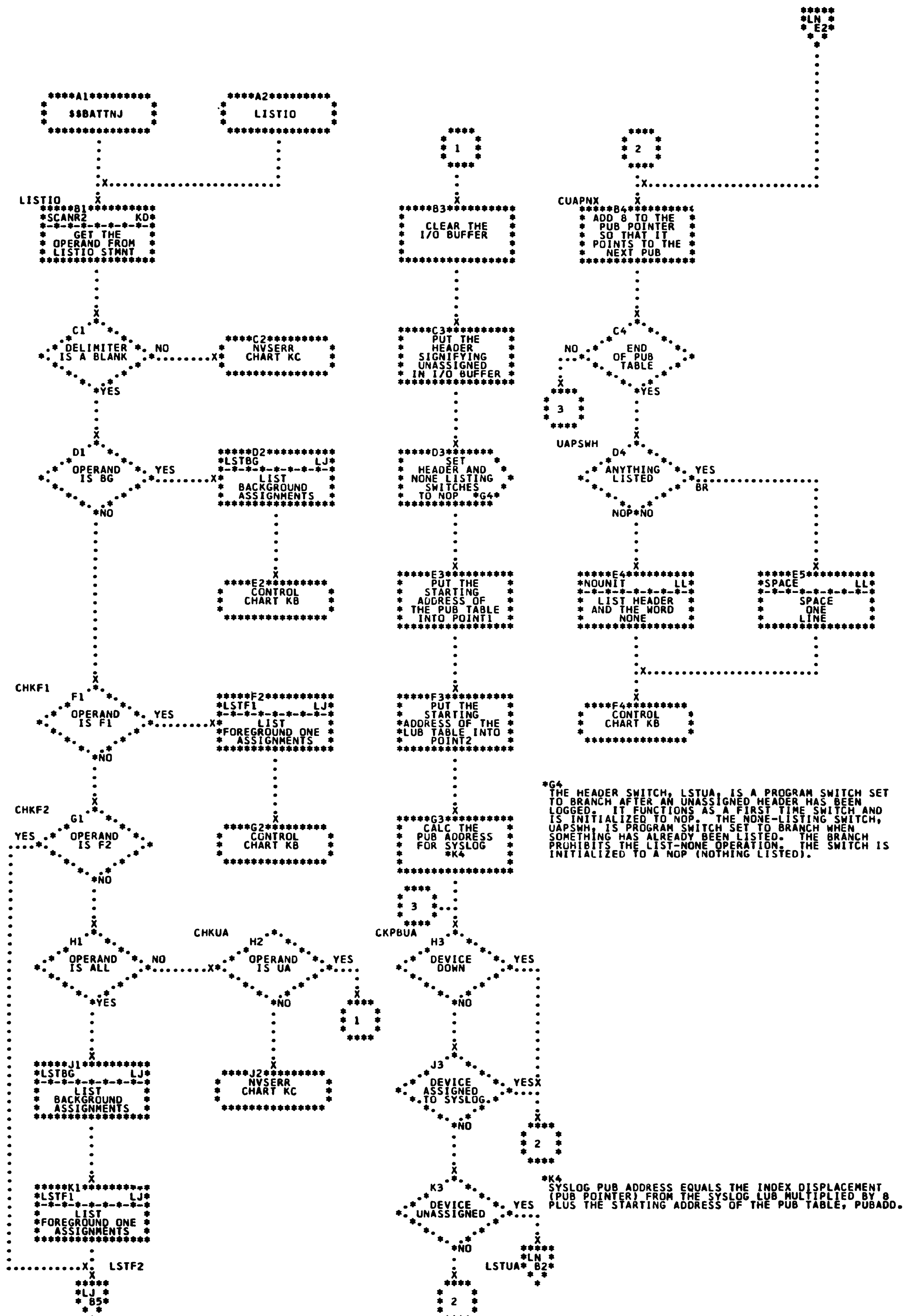


Chart LH. LISTIO Statement Processor $$BATTNJ; Refer to Supervisor, Chart 23




*G4
THE HEADER SWITCH, LSTUA, IS A PROGRAM SWITCH SET TO BRANCH AFTER AN UNASSIGNED HEADER HAS BEEN LOGGED. IT FUNCTIONS AS A FIRST TIME SWITCH AND IS INITIALIZED TO NOP. THE NONE-LISTING SWITCH, UAPSWH, IS PROGRAM SWITCH SET TO BRANCH WHEN SOMETHING HAS ALREADY BEEN LISTED. THE BRANCH PROHIBITS THE LIST-NONE OPERATION. THE SWITCH IS INITIALIZED TO A NOP (NOTHING LISTED).

*K4
SYSLOG PUB ADDRESS EQUALS THE INDEX DISPLACEMENT (PUB POINTER) FROM THE SYSLOG LUB MULTIPLIED BY 8 PLUS THE STARTING ADDRESS OF THE PUB TABLE, PUBADD.

Chart LJ. List Subroutines $$BATTNJ; Refer to Supervisor, Chart 23

```
A1  LSTBG
B1  PUT THE SYSTEM FICL AND NICL VALUES INTO WRKRG1
C1  CLEAR THE OUTPUT BUFFER
D1  PUT THE SYSTEM HEADER IN THE OUTPUT BUFFER
E1  SET SYSTEM UNIT LIST SW, SYSSWH TO BRANCH
F1  PUT ADDR OF SYSTEM CLASS TABLE INTO A REG, WRKRG2   *F2
    LK B2  LSTBUN

*F2 THE REGISTER CONTAINS THE LOGICAL UNIT SPECIFICATION.

A3  LSTF1
B3  PUT THE F1 FICL AND NICL VALUES INTO WRKRG1
C3  CLEAR THE OUTPUT BUFFER
D3  PUT THE FOREGROUND ONE HEADER IN THE OUTPUT BUFFER
    LK B1  LSTAUN

LH K1
LSTF2
B5  PUT THE F2 FICL AND NICL VALUES INTO WRKRG1
C5  CLEAR THE OUTPUT BUFFER
D5  PUT THE FOREGROUND TWO HEADER IN THE OUTPUT BUFFER
E5  LSTAUN  LK
    LIST FOREGROUND TWO UNIT ASSIGNMENTS
F5  CONTROL CHART KB
```

Chart LK. Locate Assignment Routine $$BATTNJ; Refer to Supervisor, Chart 23

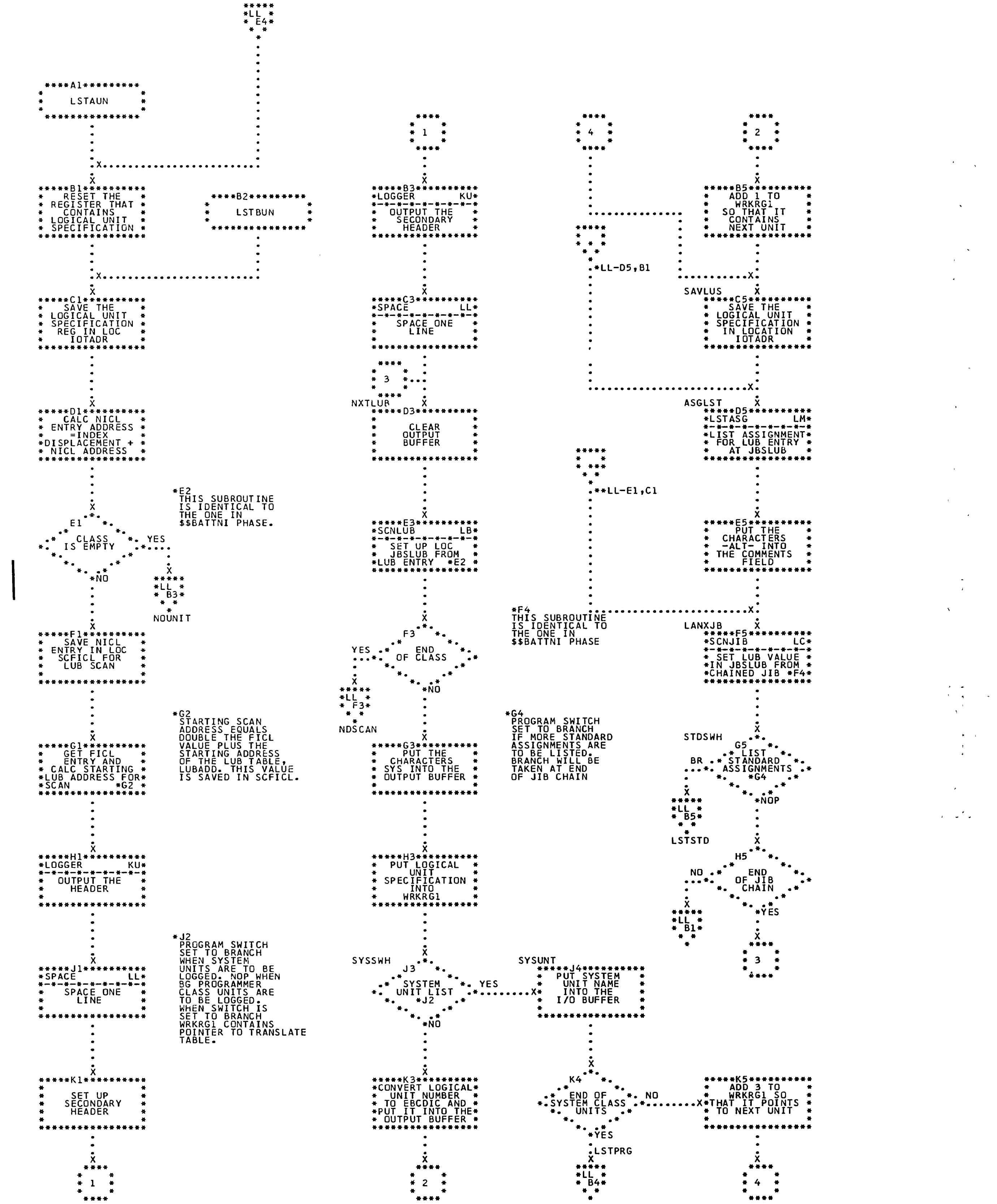


Chart LL. Output List (Part 1 of 3) $$BATTNJ; Refer to Supervisor, Chart 23

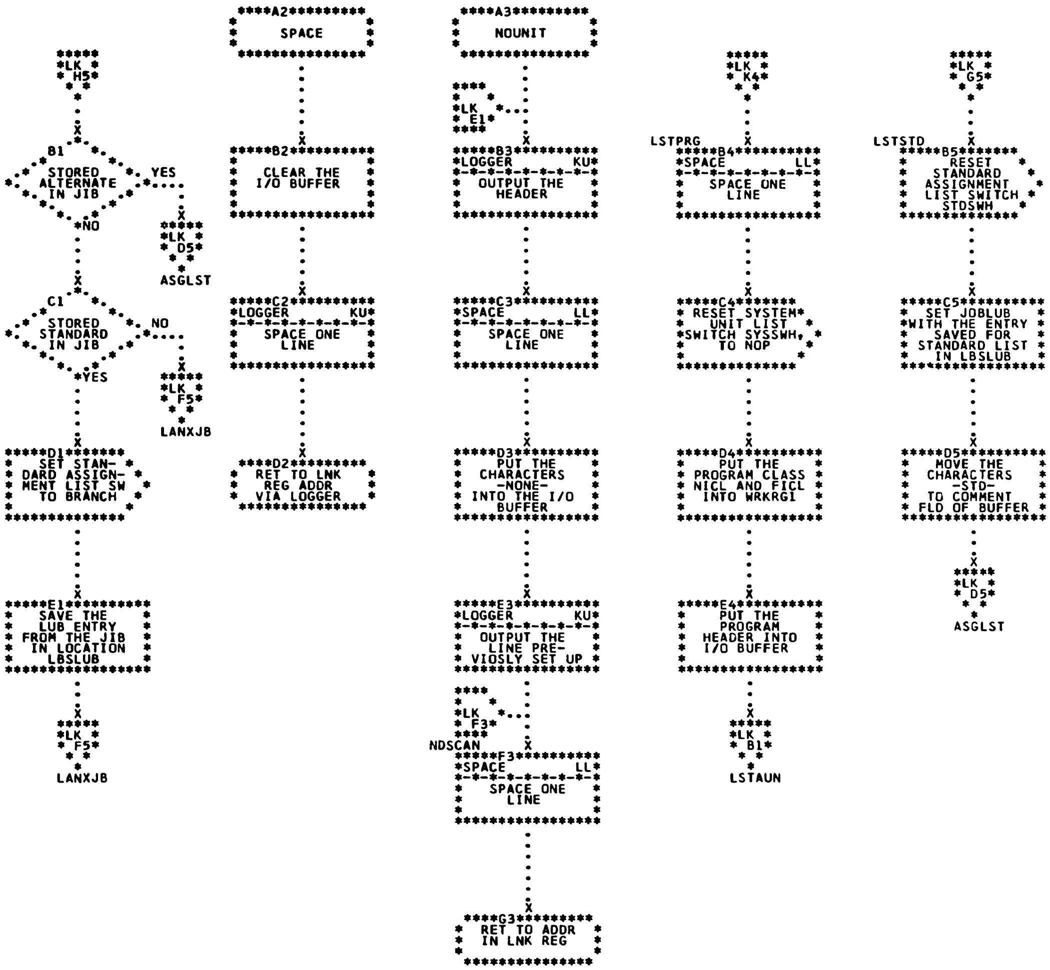


Chart LM. Output List (Part 2 of 3) $$BATTNJ; Refer to Supervisor, Chart 23

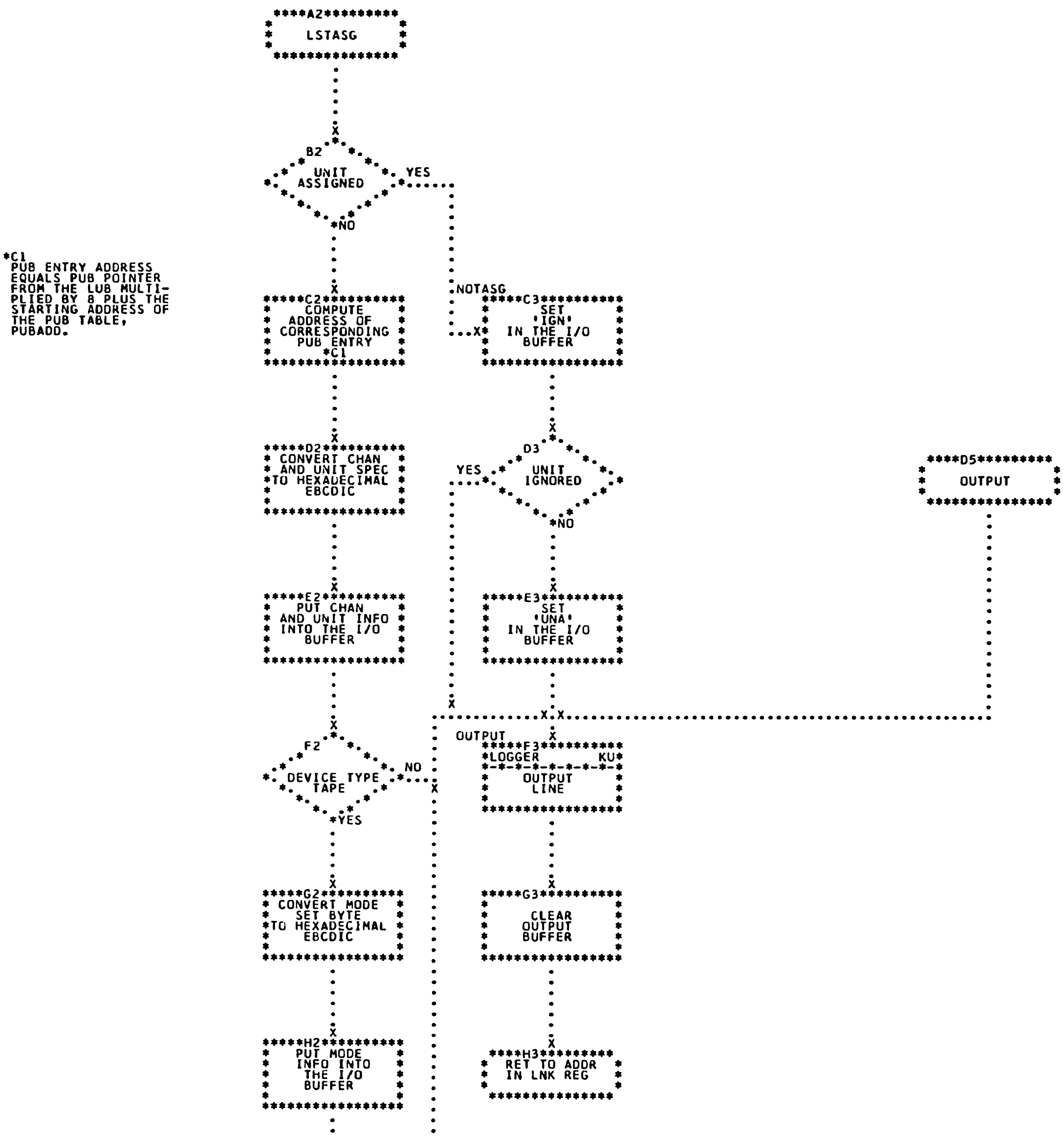


Chart LN. Output List (Part 3 of 3) $$BATTNJ; Refer to Supervisor, Chart 23

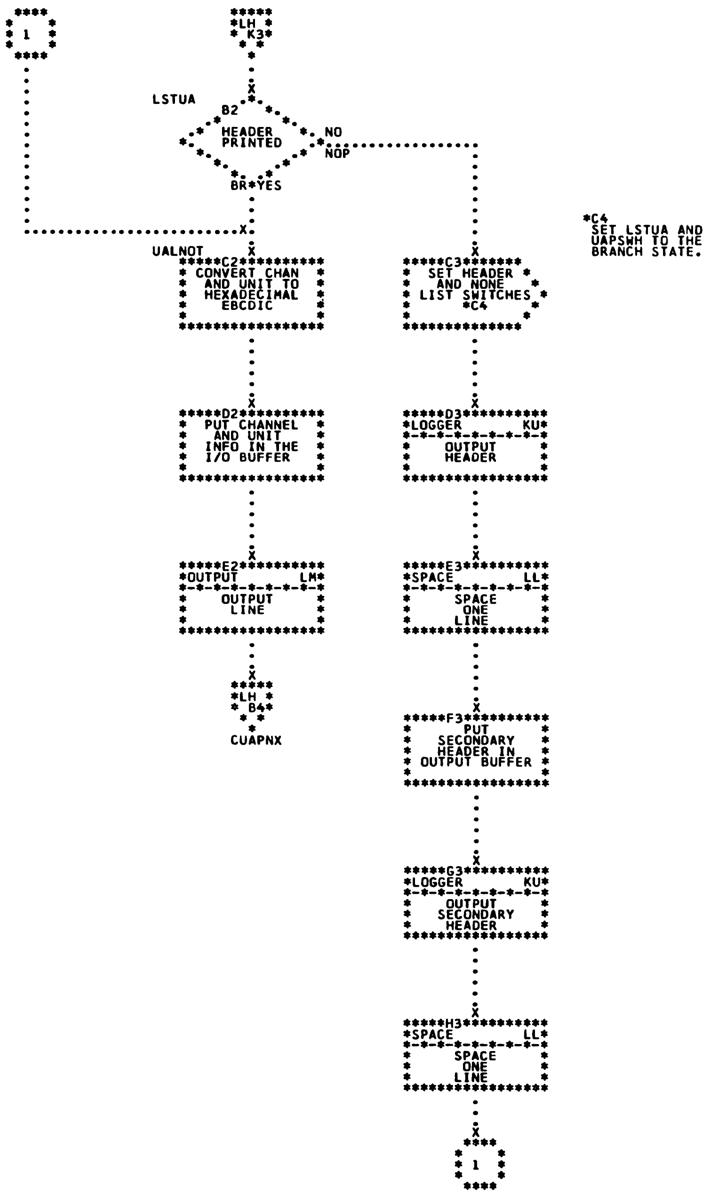


Chart LP. VOL Statement Processor $$BATTNK; Refer to Supervisor, Chart 23

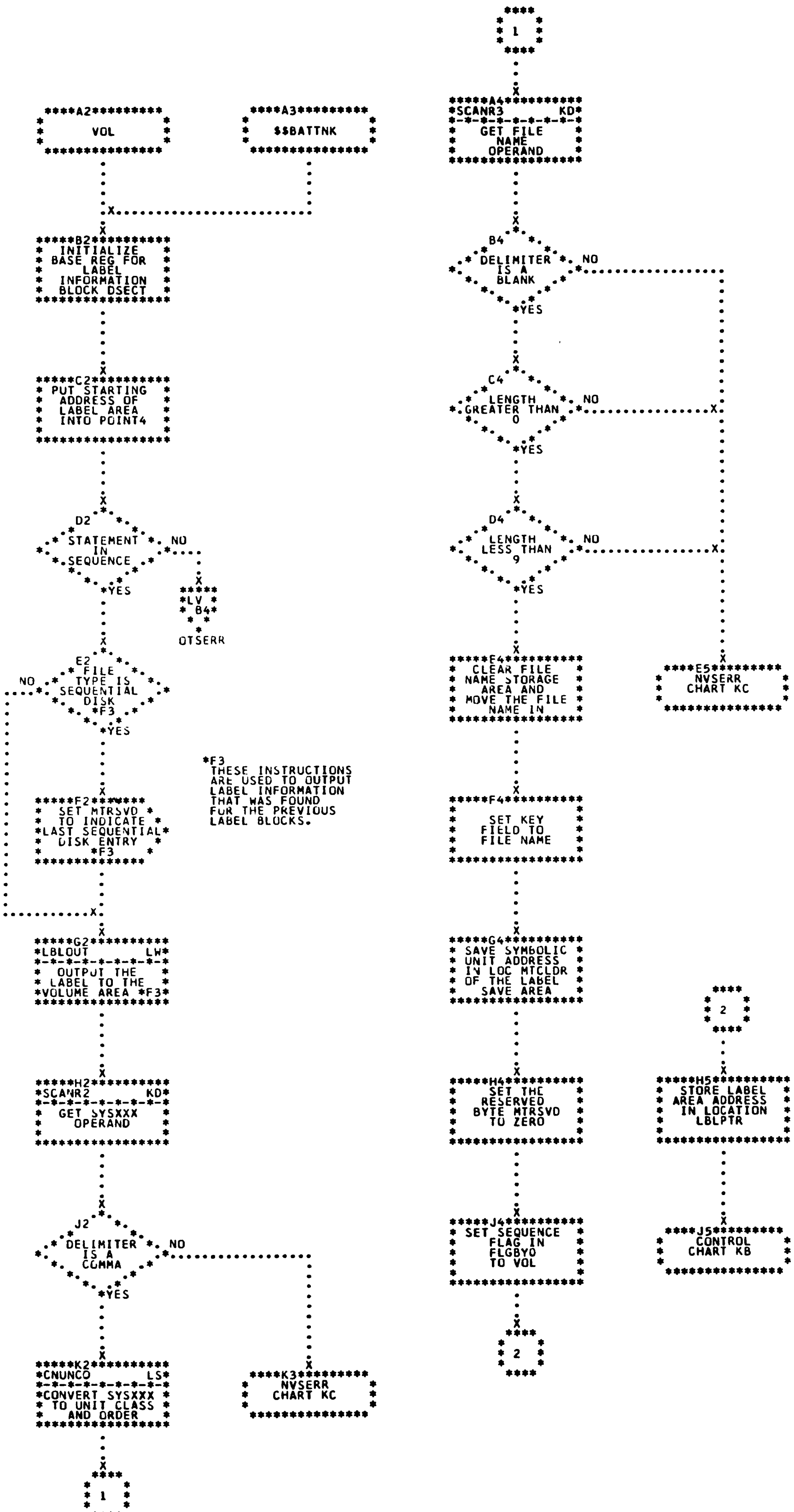


Chart LQ. TPLAB Statement Processor $$BATTNK; Refer to Supervisor, Chart 23

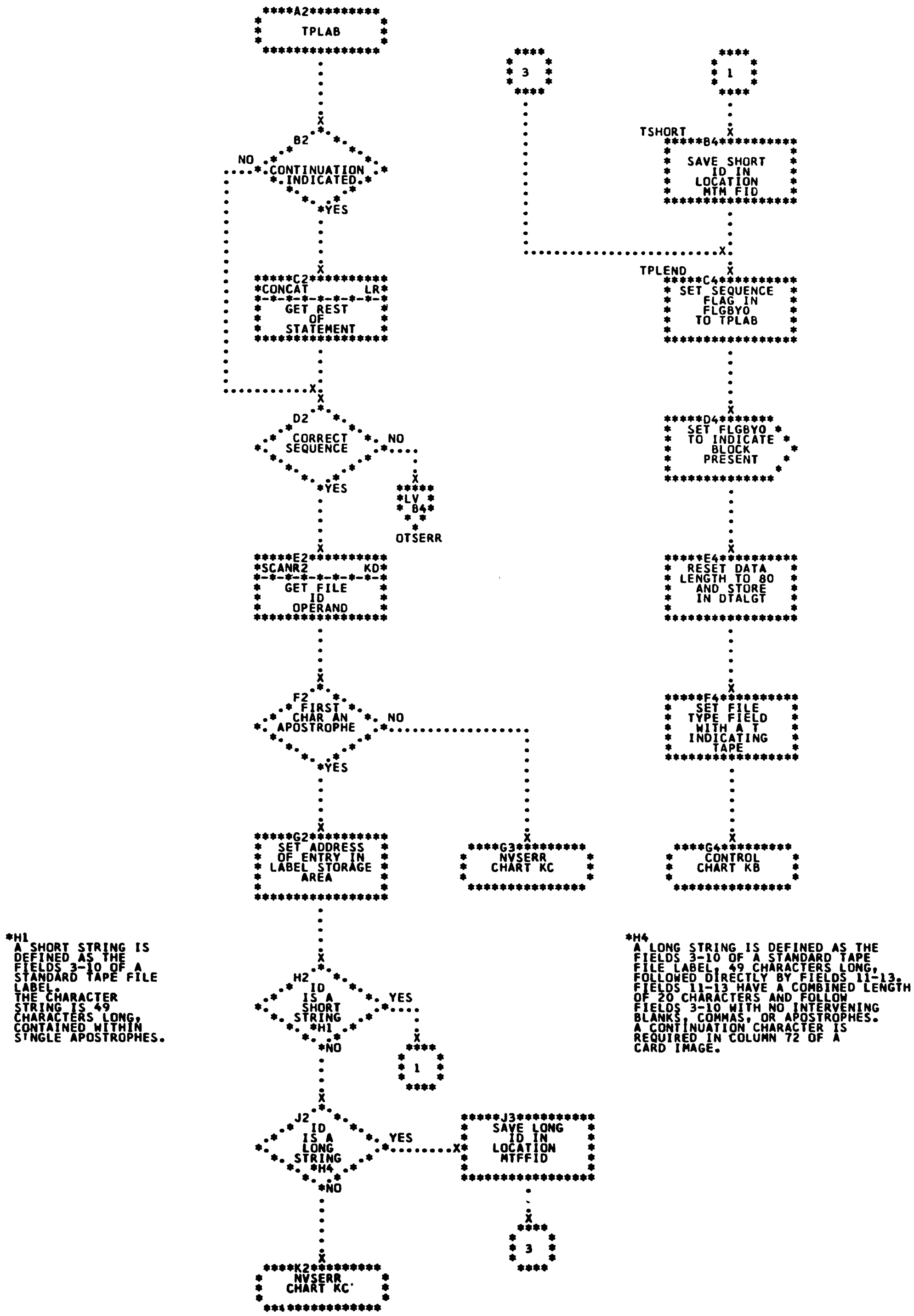


**Chart LR.** Concatenate Subroutine $$BATTNK; Refer to Supervisor, Chart 23

```
 ****A3*********
*               *
*    CONCAT     *
*               *
 ***************
        *
        *
        *
        *
        *
        *
        *
        X
*****B3**********
*     CLEAR     *
* CONCATENATION *
* BUFFERS WITH  *
*    BLANKS     *
*  (127 BYTES)  *
*****************
        *
        *
        *
        *                 *C4
        *                  SAVE THE FIRST 70 BYTES
        *                  OF THE I/O BUFFER,
        X                  IN LOCATION BUFFER1 BY
*****C3**********          ORING THE TWO FIELDS.
*  SAVE FIRST   *          THIS WILL MAP ALL
*    PART OF    *          LOWERCASE CHARACTERS TO
*   STATEMENT   *          UPPERCASE.
*      *C4      *
*               *
*****************
        *
        *
        *
        *                 *D4
        *                  THE SECOND PART OF THE
        *                  STATEMENT IS NOW IN
        X                  I/O BUFFER, AT LOCATION
*****D3**********          BUFFER.
*RDSTMT       KC*
*-*-*-*-*-*-*-*-*
* INPUT SECOND  *
*    PART OF    *
*STATEMENT   *D4*
*****************
        *
        *
        *
        *                 *E4
        *                  COMBINE THE TWO PARTS BY
        *                  ORING THE CONTENTS OF THE
        X                  I/O BUFFER (STARTING AT
*****E3**********          BUFFER +15) WITH BUFFR2.
*  COMBINE THE  *          BUFFR2 IS CONTIGUOUS
*TWO PARTS INTO *          WITH BUFFR1.
*   A SINGLE    *
*   STATEMENT   *
*      *E4      *
*****************
        *
        *
        *
        *                 *F4
        *                  GET VALUES SAVED BY THE
        *                  SCAN SUBROUTINE:
        X                  1. REMAINING BUFFER
*****F3**********          LENGTH.
* RESET POINTER *          2. ADDRESS OF OPERAND
* AND LIMIT FOR *          SUBTRACT THE ADDRESS
* NEXT ENTRY TO *          FIELD OF THE CCW FROM
*     SCANR     *          THE STARTING ADDRESS OF
*      *F4      *          THE OPERAND. ADD BUFFR1
*****************          ADDRESS TO THE DIFFERENCE.
        *                  THE RESULTANT NEW OPERAND
        *                  ADDRESS IS IN POINT1. ADD
        *                  56 TO THE REMAINING BUFFER
        *                  FIELD. THE RESULTANT NEW
        *                  BUFFER LENGTH IS IN POINT2.
        *                  RESTORE POINT1 AND POINT2
        *                  IN LOCATION TMPAR1 WHERE
        X                  IT IS REFERENCED BY THE
 ****G3*********           SCAN SUBROUTINE.
*  RET TO ADDR  *
*  IN LNK REG   *
*               *
 ***************
```

Chart LS. Validity Check Subroutine $$BATTNK; Refer to Supervisor, Chart 23

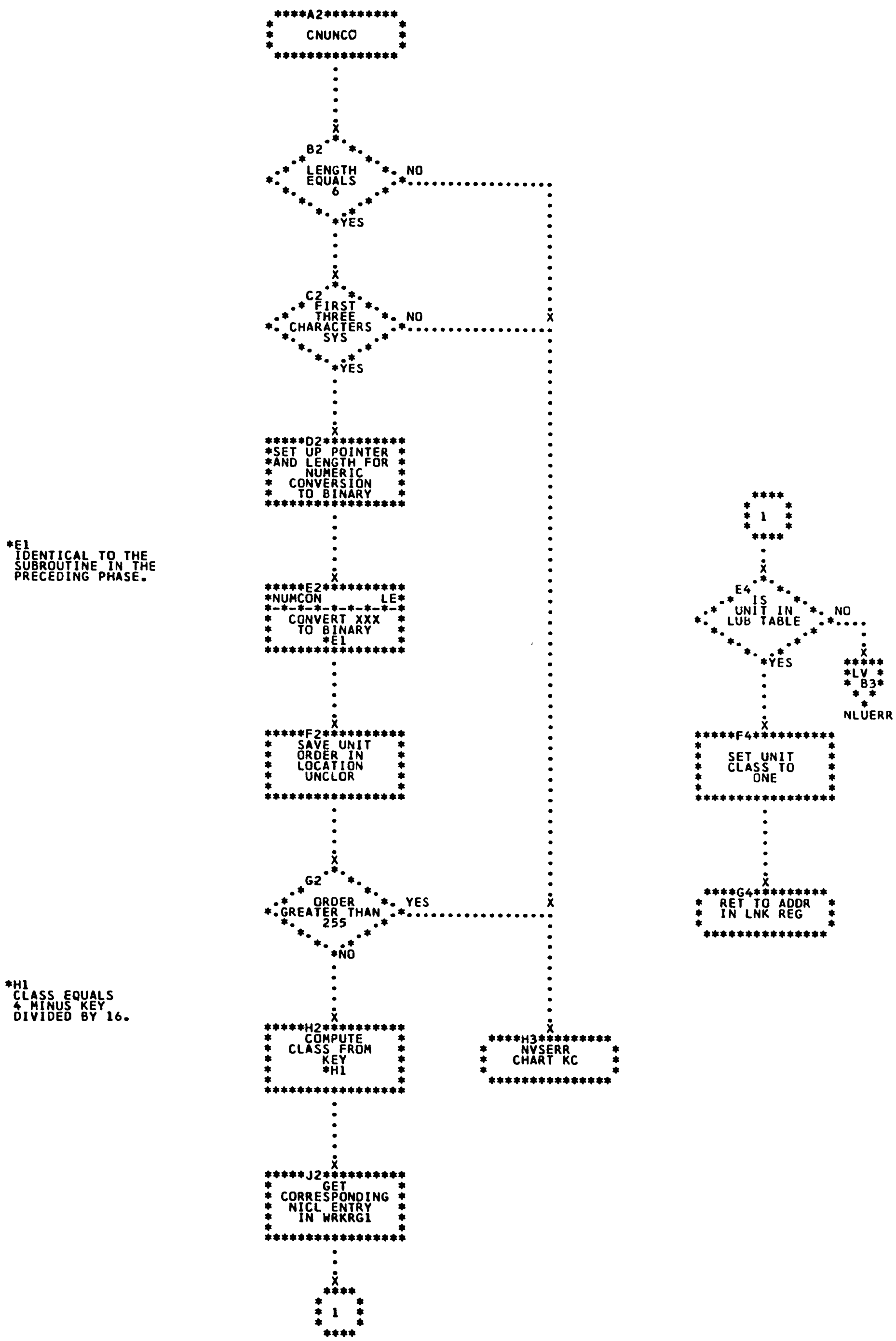


Chart LT. DLAB Statement Processor $$BATTNK; Refer to Supervisor, Chart 23

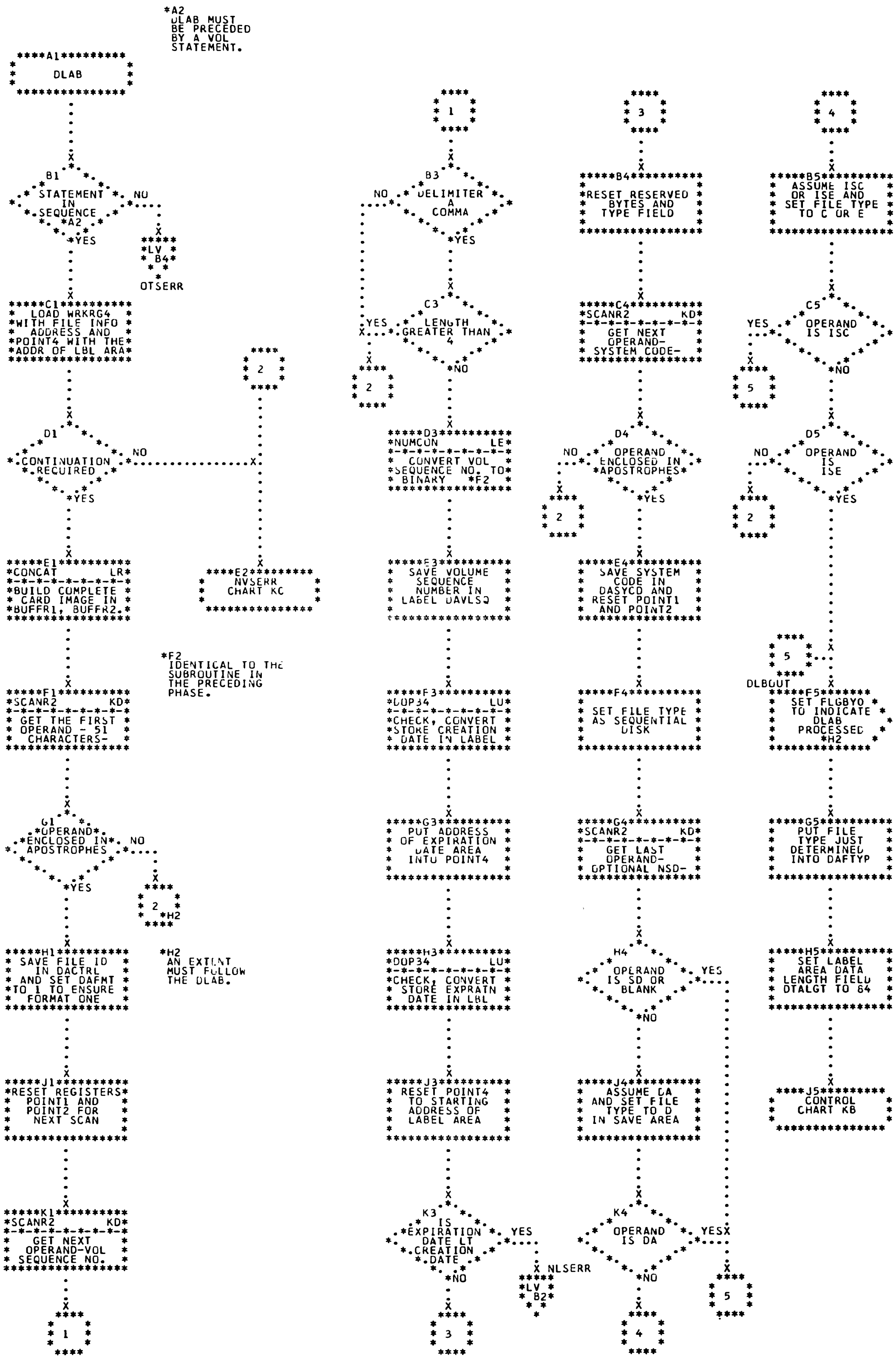
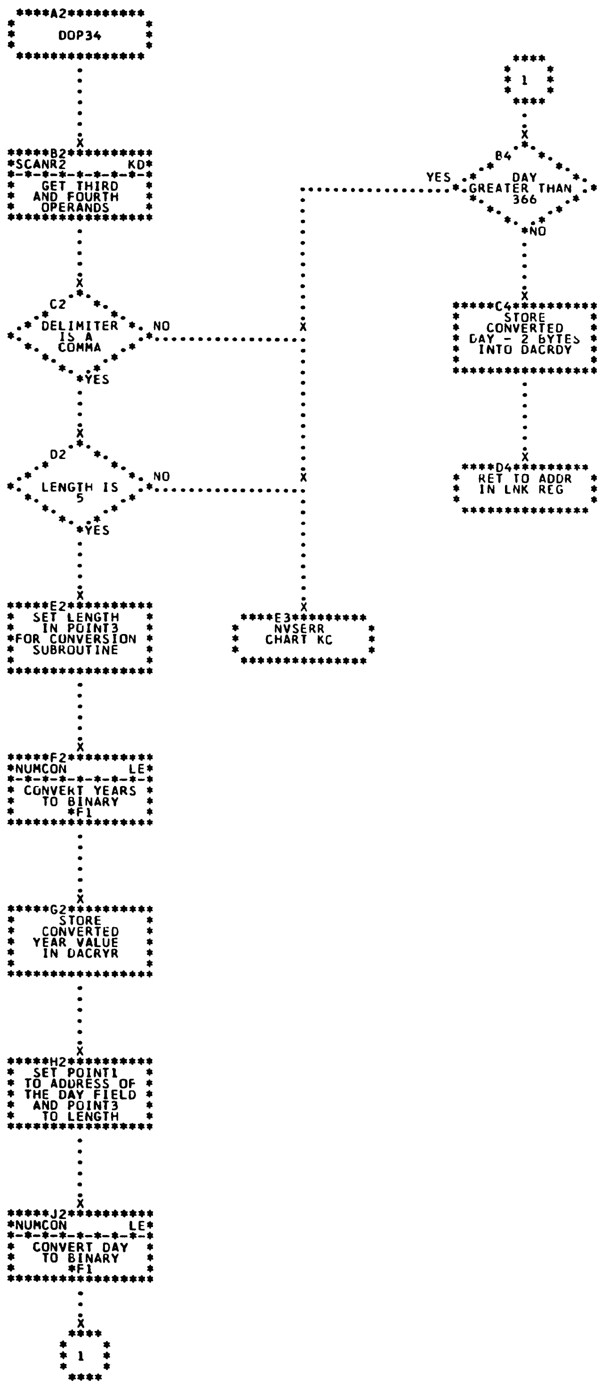
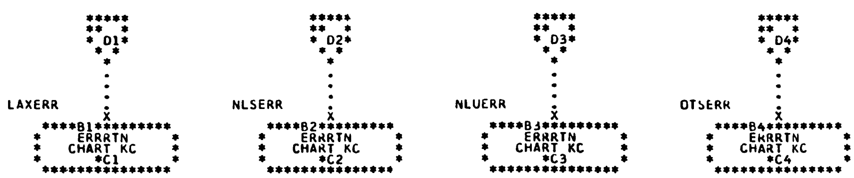


**Chart LU. Extract Operand from Statement Subroutine $$BATTNK; Refer to Supervisor, Chart 23**

*F1
IDENTICAL TO THE
SUBROUTINE IN THE
PRECEDING PHASE.



## Chart LV. Common Error Exits $$BATTNK; Refer to Supervisor, Chart 23




```
*C1
 1L10D
 LABEL AREA
 EXHAUSTED
 MESSAGE.

*C2
 1L000
 INVALID
 LABEL SYNTAX
 MESSAGE.

*C3
 1A40D
 INVALID
 LOGICAL UNIT
 SPECIFICATION
 MESSAGE.

*C4
 1S10D
 STATEMENT
 OUT OF SEQUENCE
 MESSAGE.

*D1
 LW-F3
 LX-H1
 MB-B5
 MD-D3

*D2
 LT-K3
 LX-F3
 LY-E1
 LZ-K1, B4
 MA-C4, F3, G1

*D3
 LS-E4

*D4
 LP-D2
 LQ-D2
 LT-H1
 LX-H1, J3
 LY-H1
 MC-C3
```

Chart LW. Output Label Data Subroutine $$BATTNK; Refer to Supervisor, Chart 23

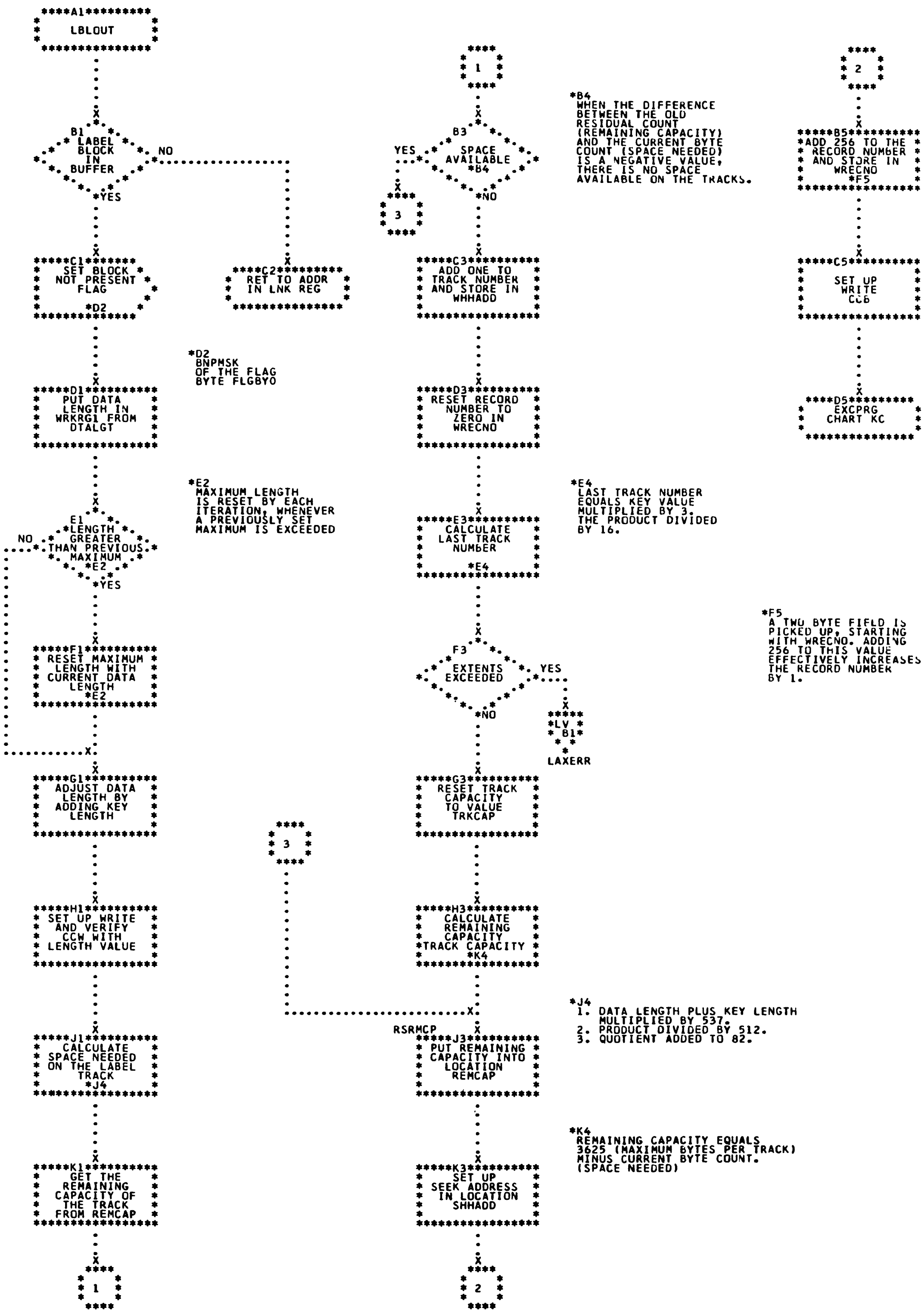


Chart LX. XTENT Statement Processor, Type and Sequence (Part 1 of 2) $$BATTNL; Refer to Supervisor, Chart 23

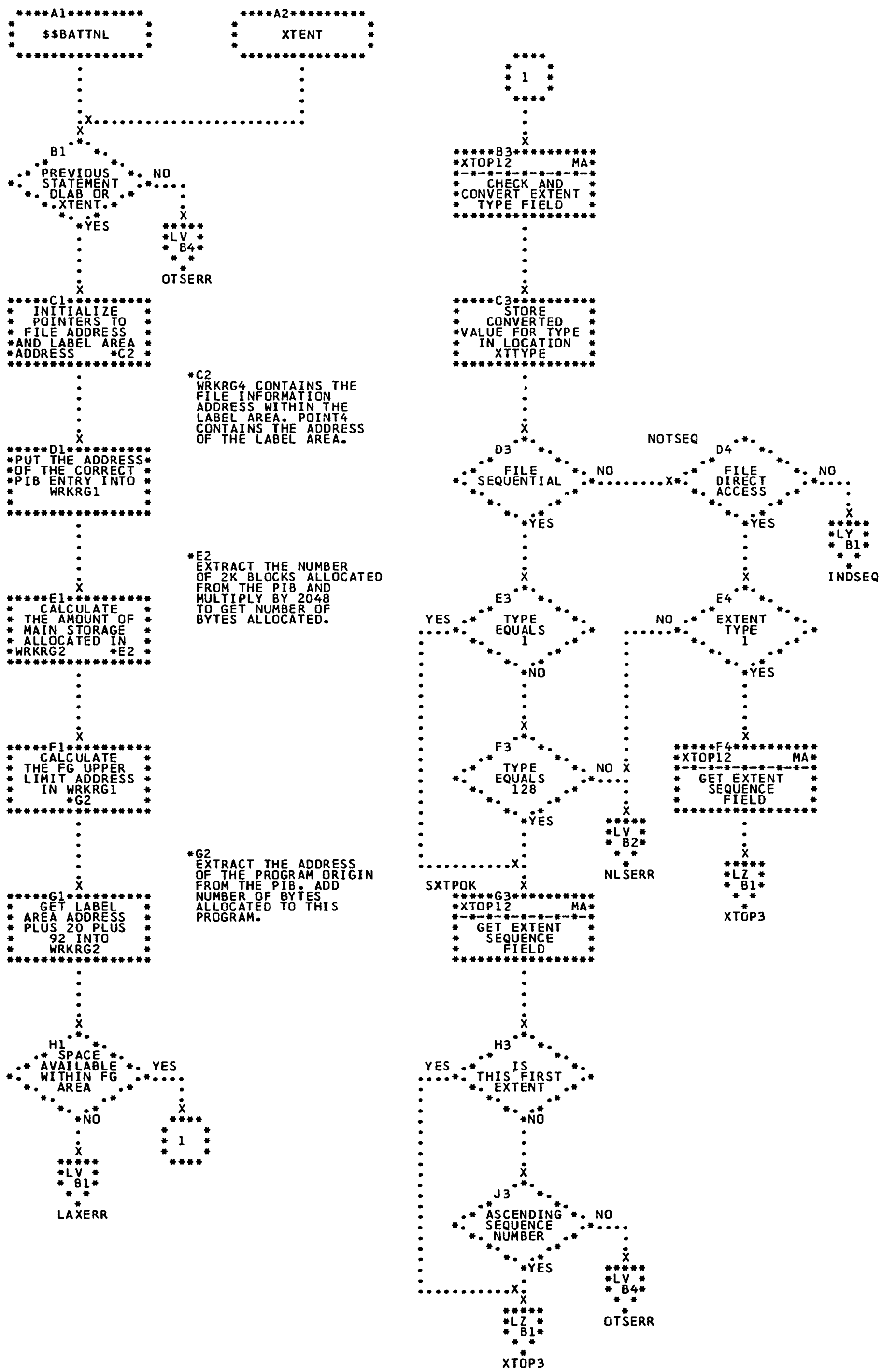


Chart LY. XTENT Statement Processor, Type and Sequence (Part 2 of 2) $$BATTNL; Refer to Supervisor, Chart 23

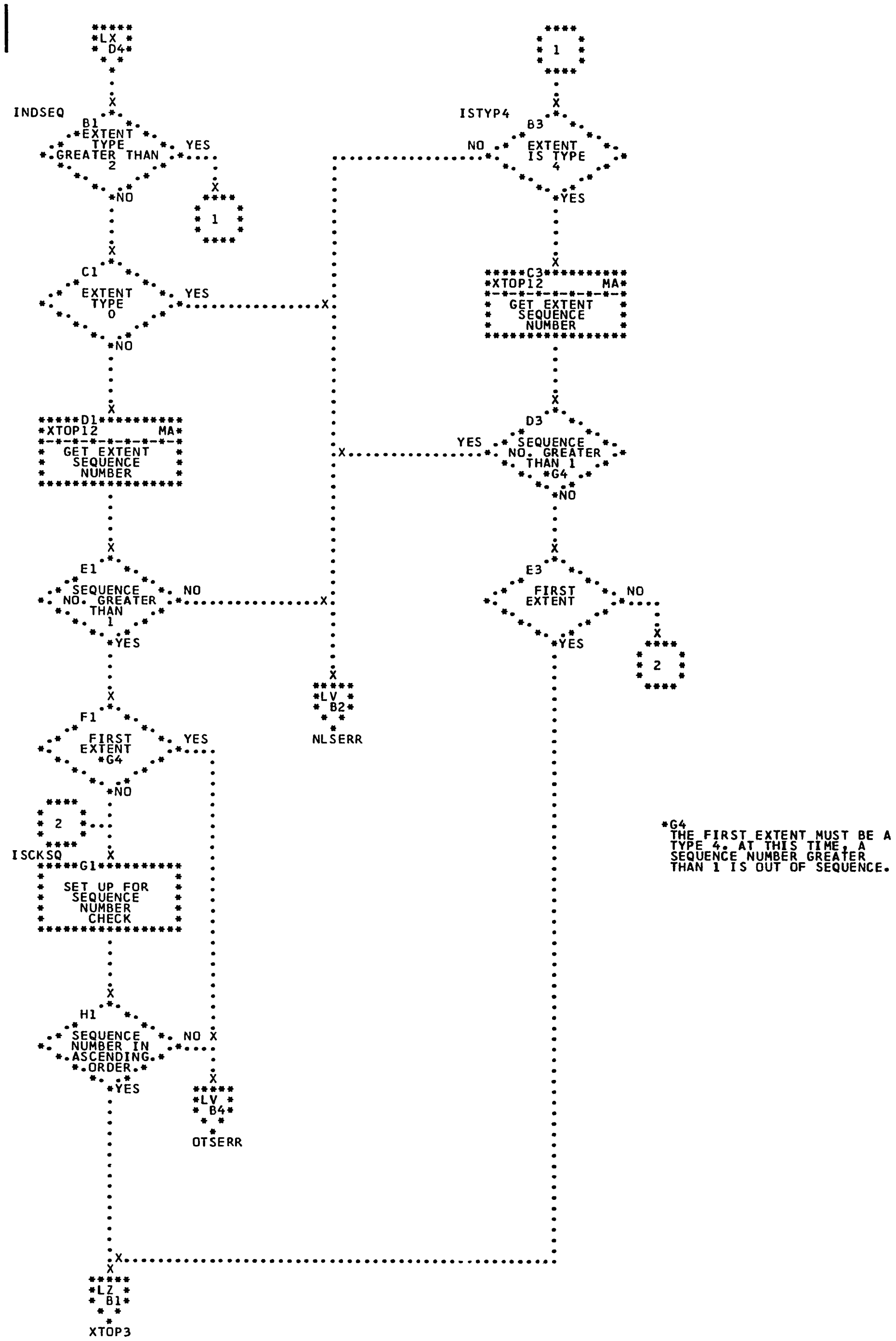


Chart LZ. XTENT Limit Processing $$BATTNL; Refer to Supervisor, Chart 23

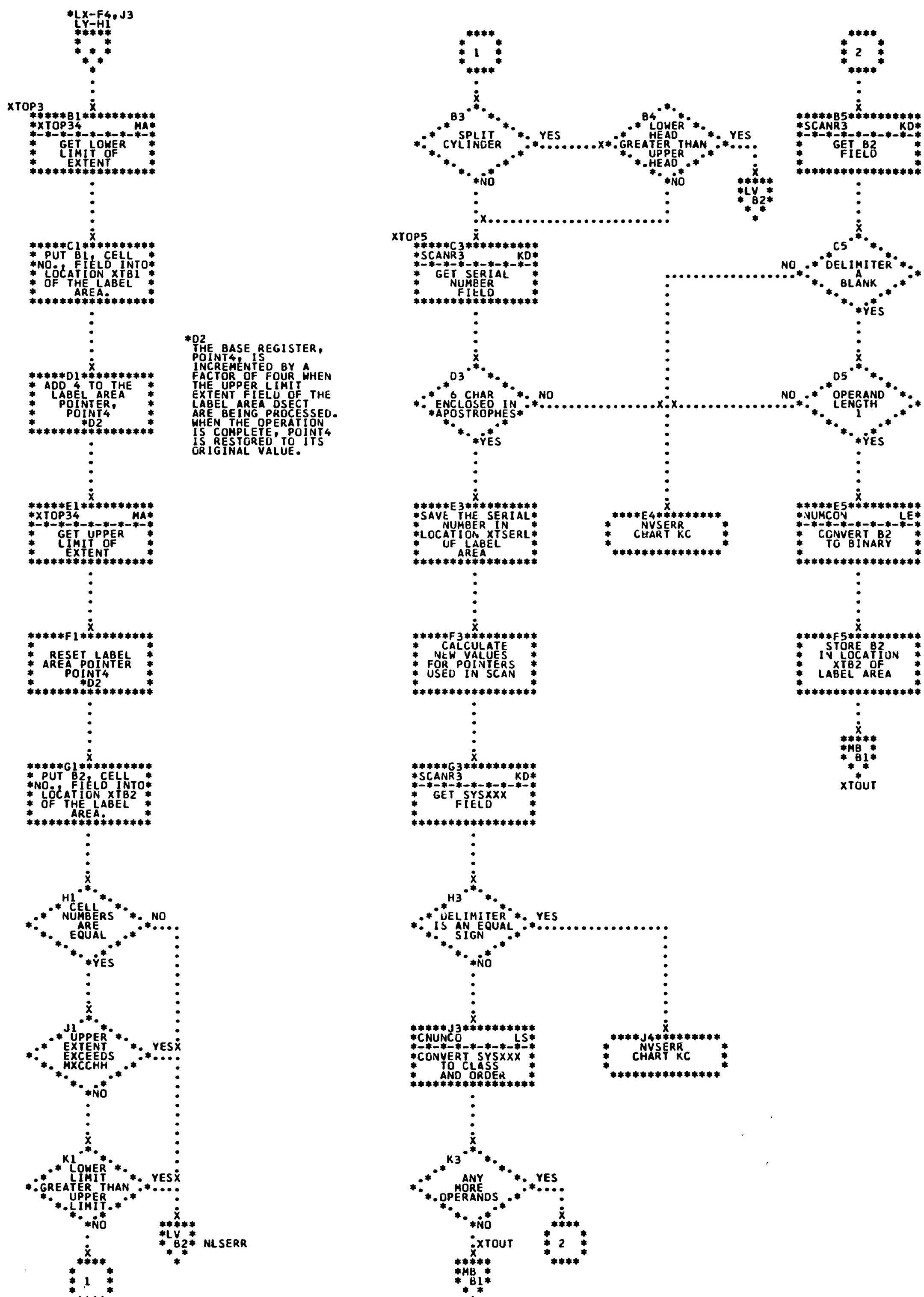


Chart MA. XTENT Processor Subroutines $$BATTNL; Refer to Supervisor, Chart 23

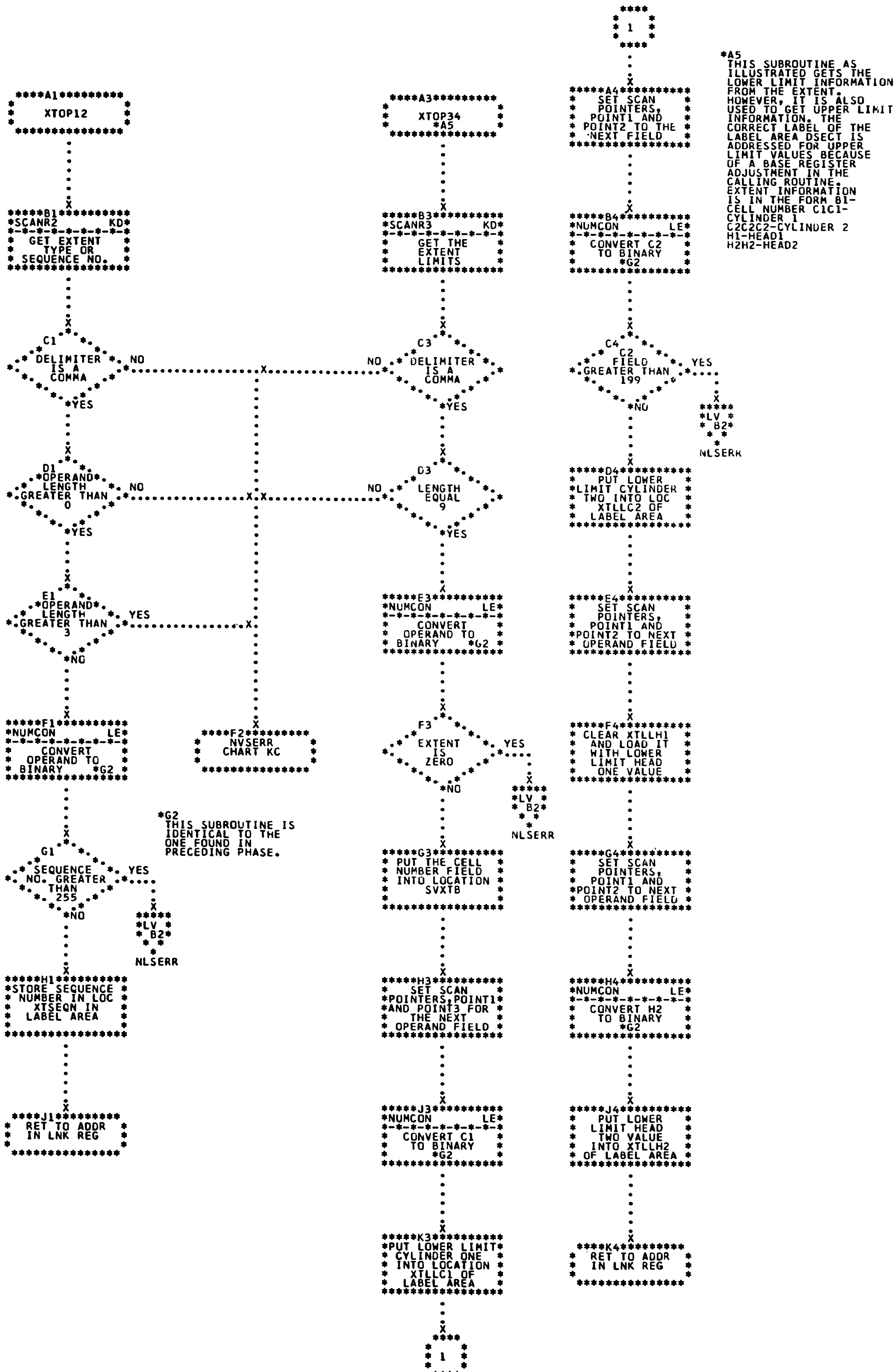


Chart MB. Terminal XTENT Statement Processing $$BATTNL;
Refer to Supervisor, Chart 23

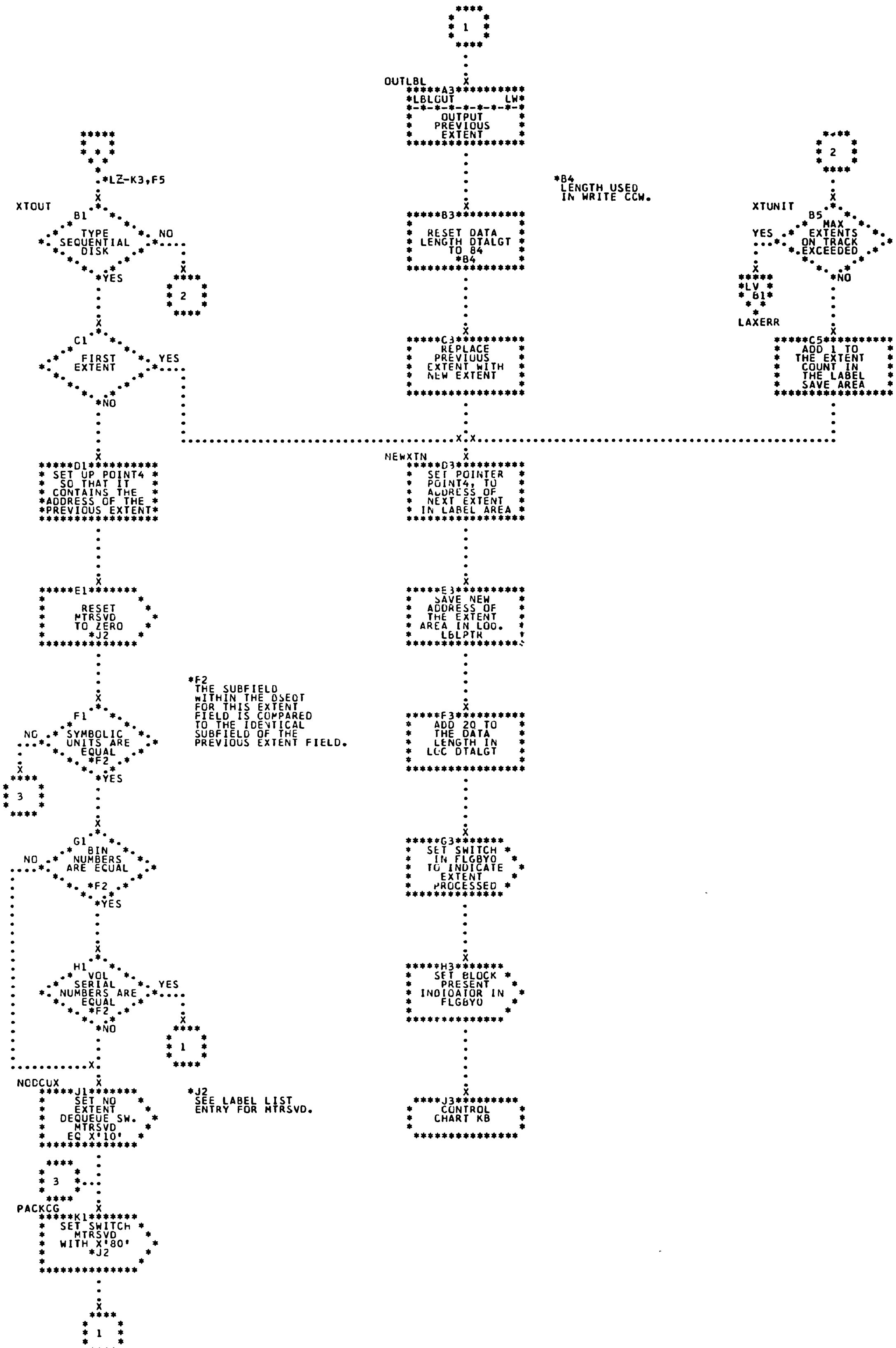
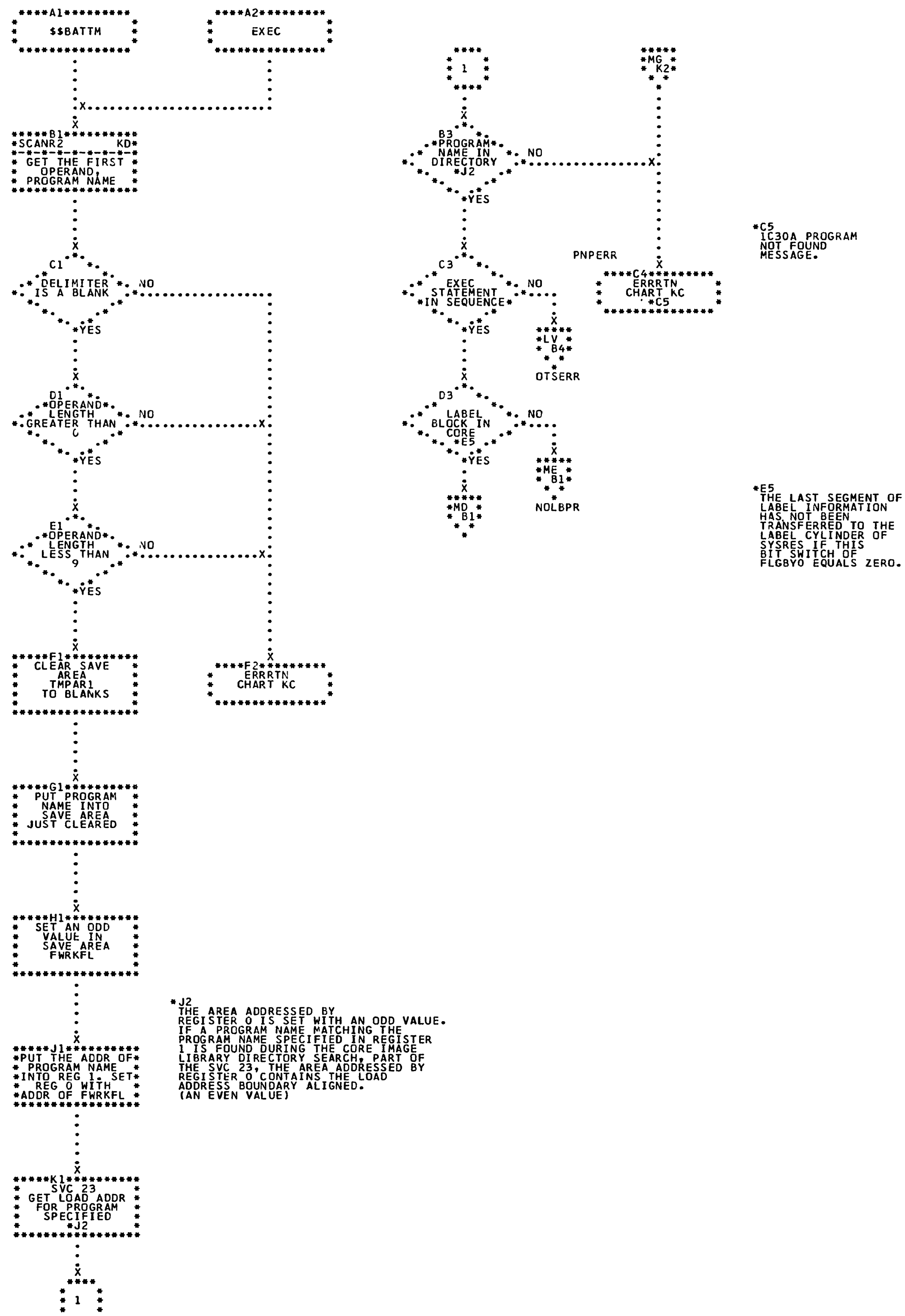


Chart MC. EXEC Statement Processor $$BATTNM; Refer to Supervisor, Chart 23



## Chart MD. Output Last Block of Label Information $$BATTNM; Refer to Supervisor, Chart 23

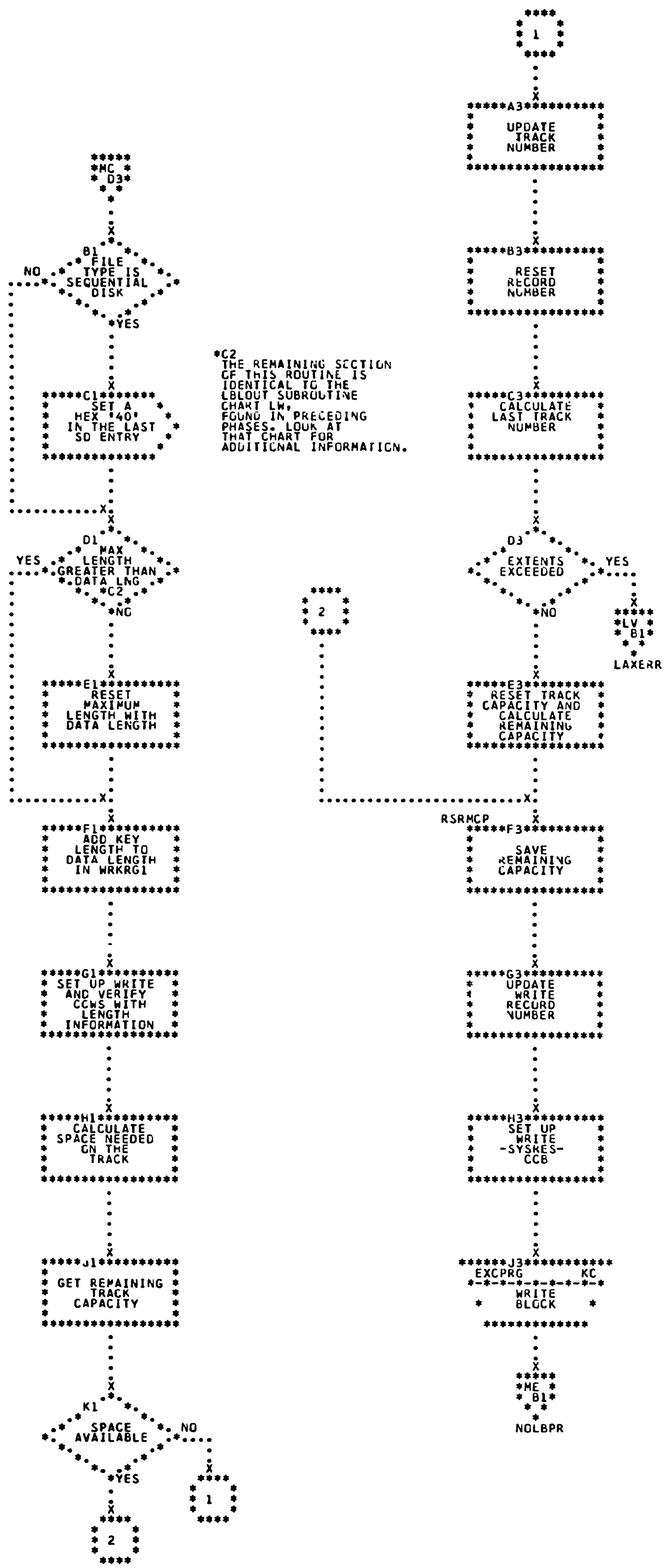


Chart ME. Move Last Block Routine $$BATTNM; Refer to Supervisor, Chart 23

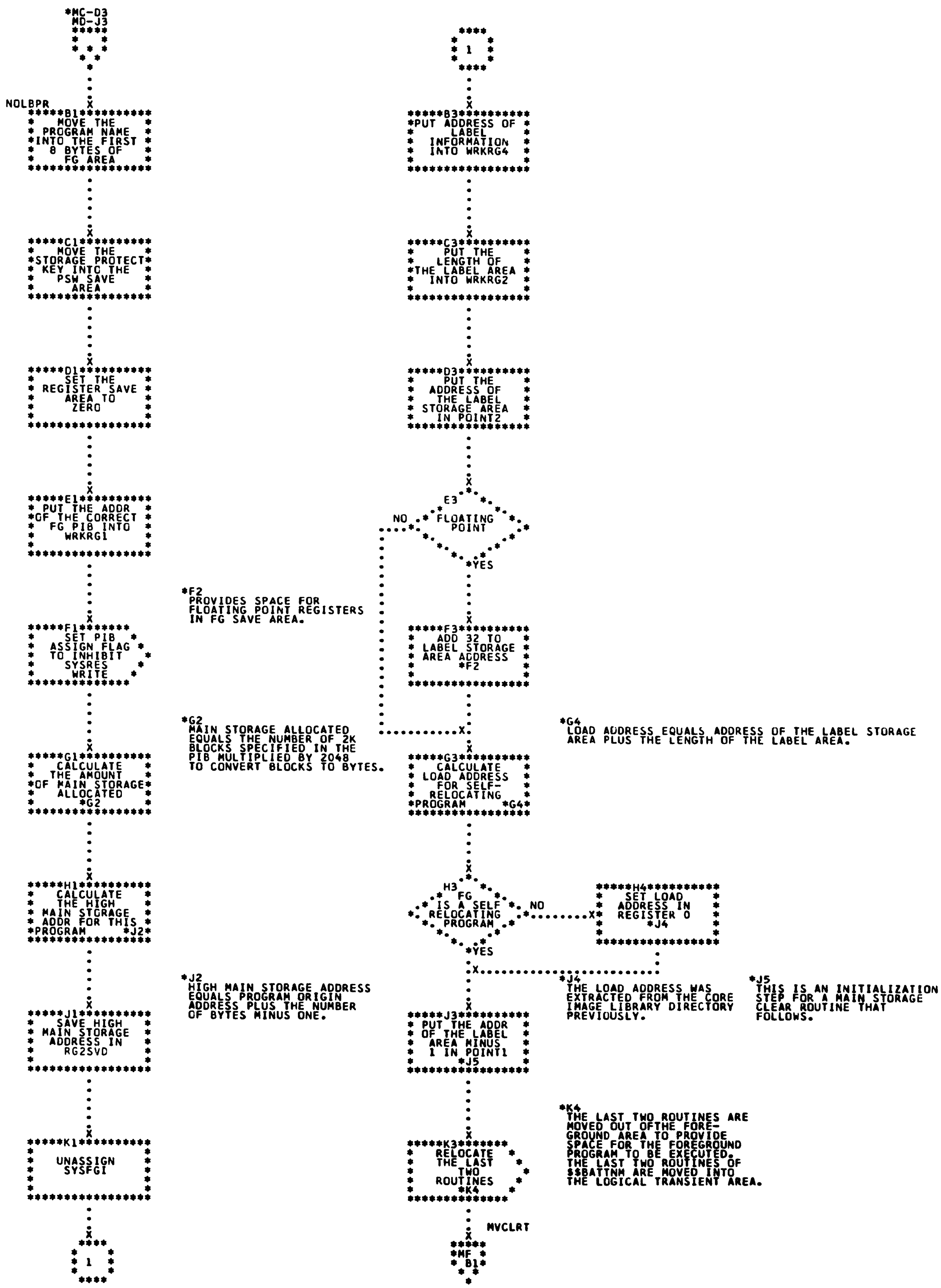


Chart MF. Move Subroutine and INITIALIZE FOR FG Program Load Routine $$BATTNM; Refer to Supervisor, Chart 23

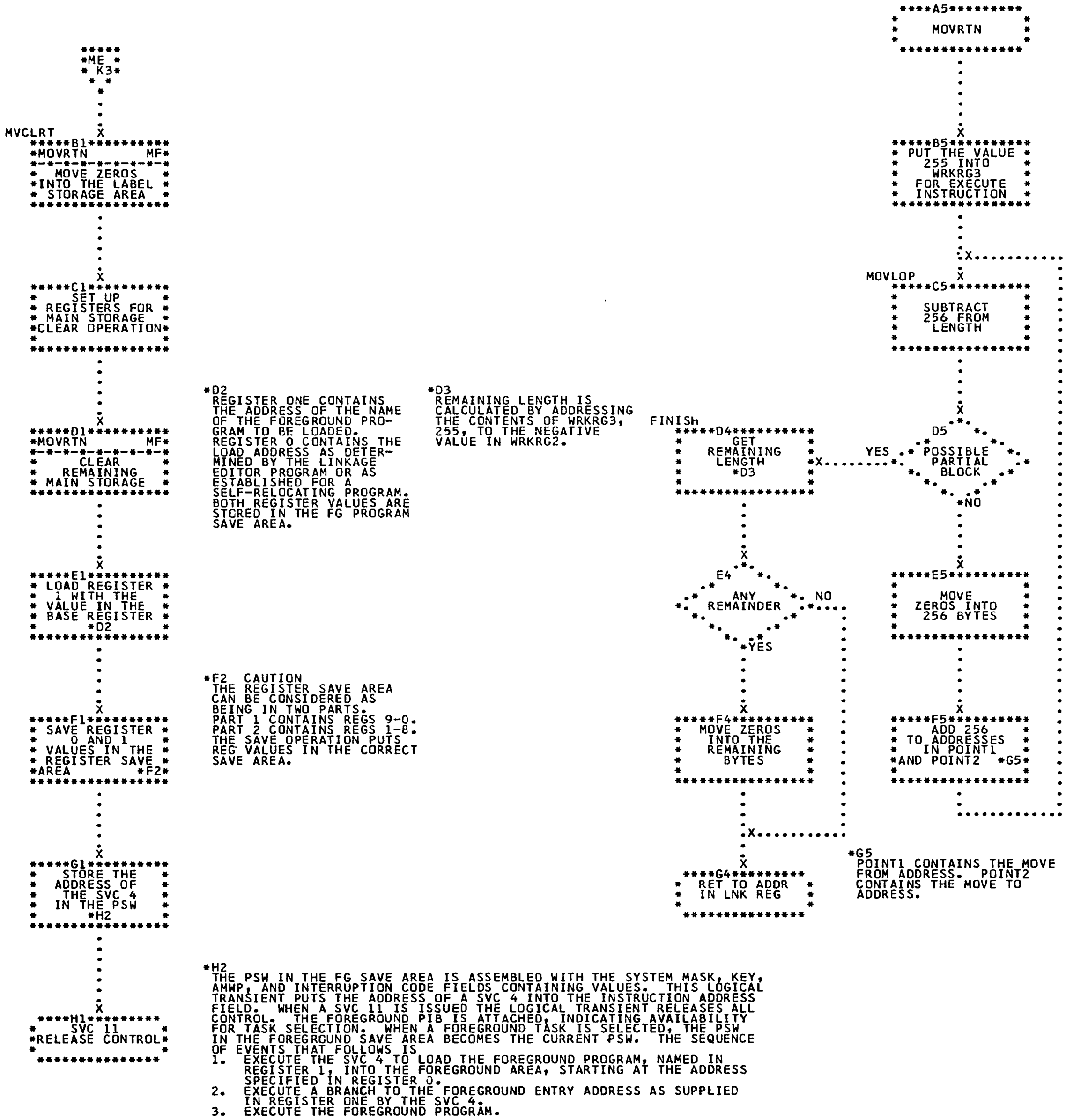


Chart MG. UCS Statement Processor $$BATTNM; Refer to Supervisor, Chart 23

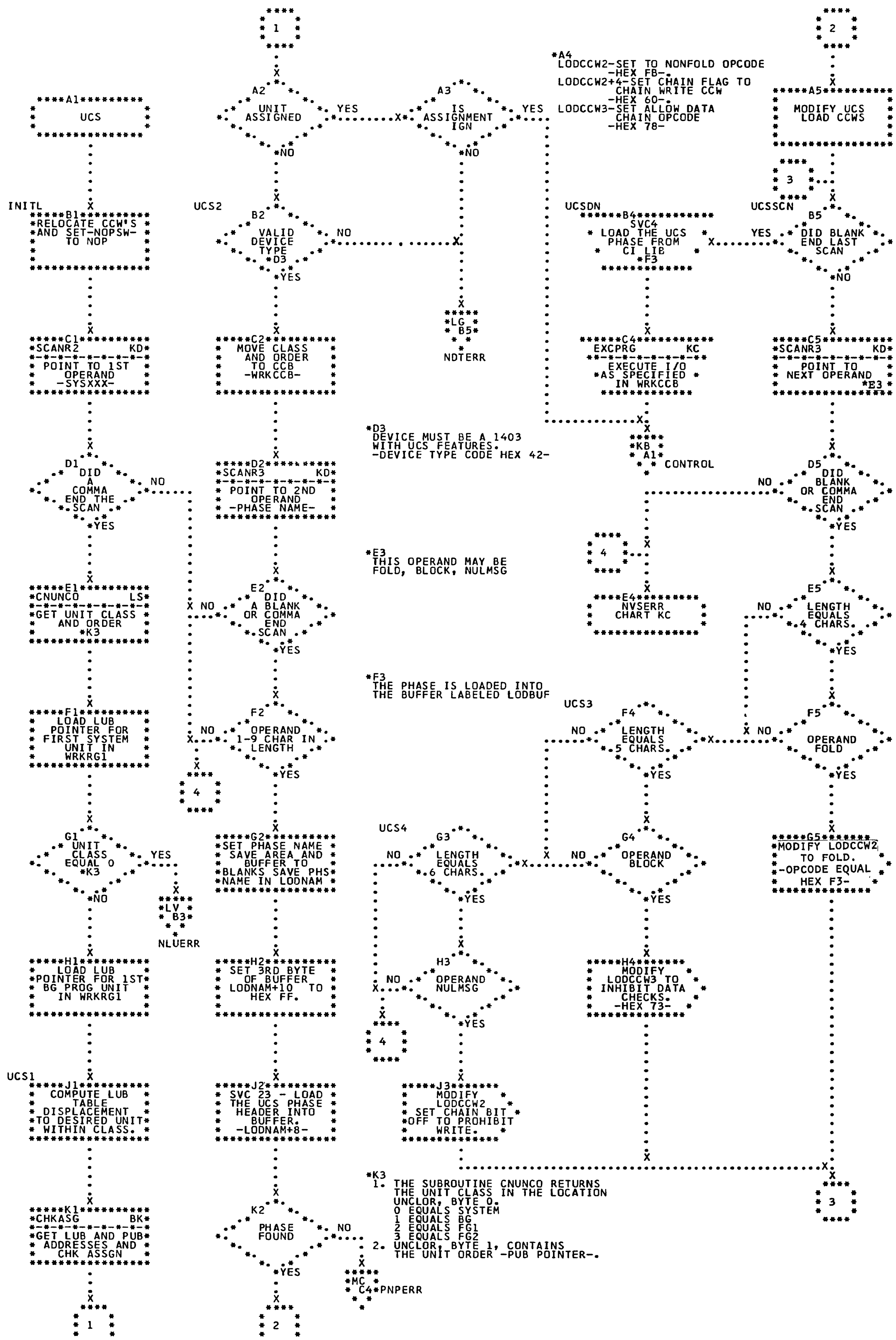


Chart MH. TIMER Statement Processor $$BATTNN; Refer to Supervisor, Chart 25

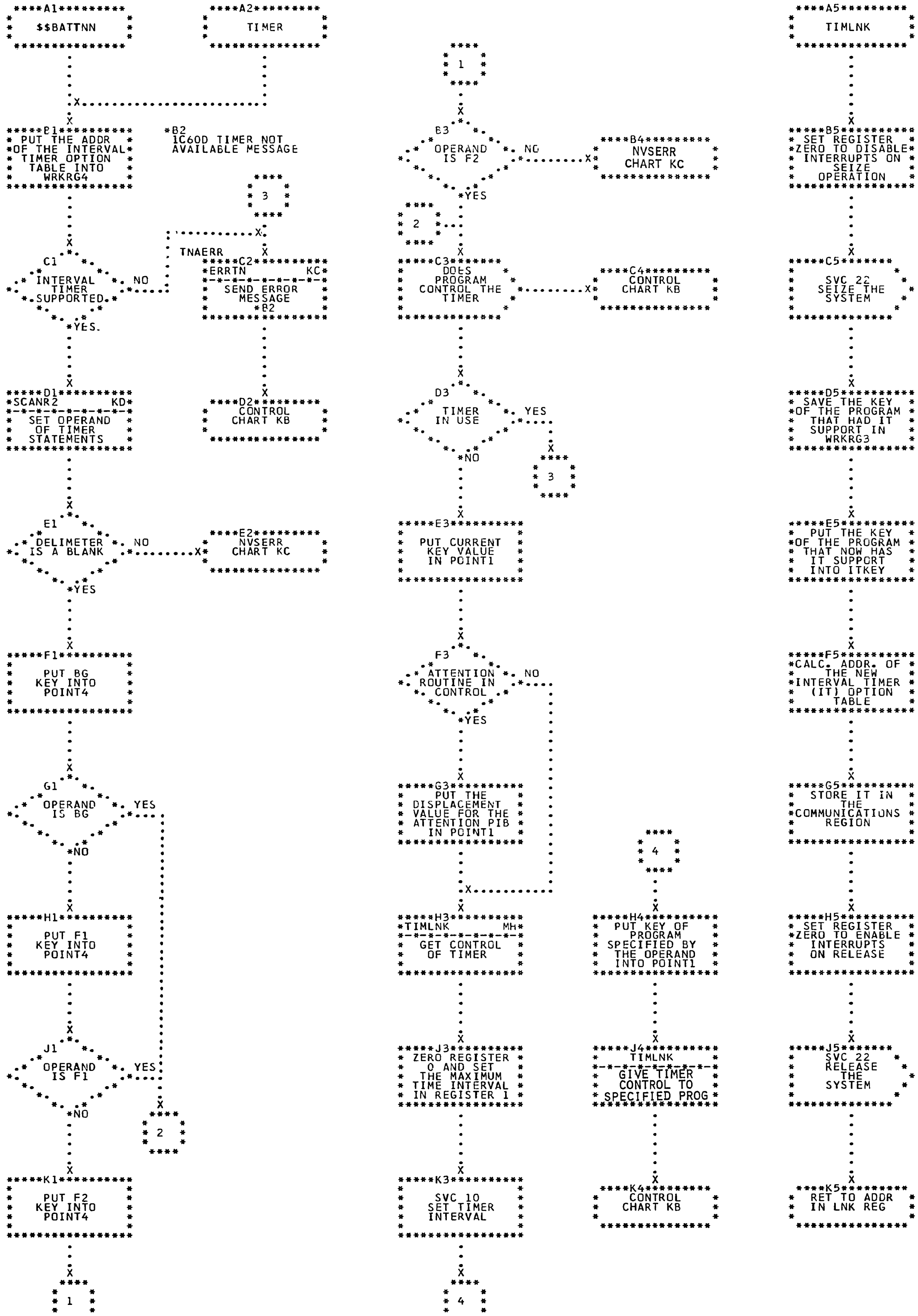


Chart MJ. UNA Statement Processor $$BATTNI; Refer to Supervisor, Chart 22

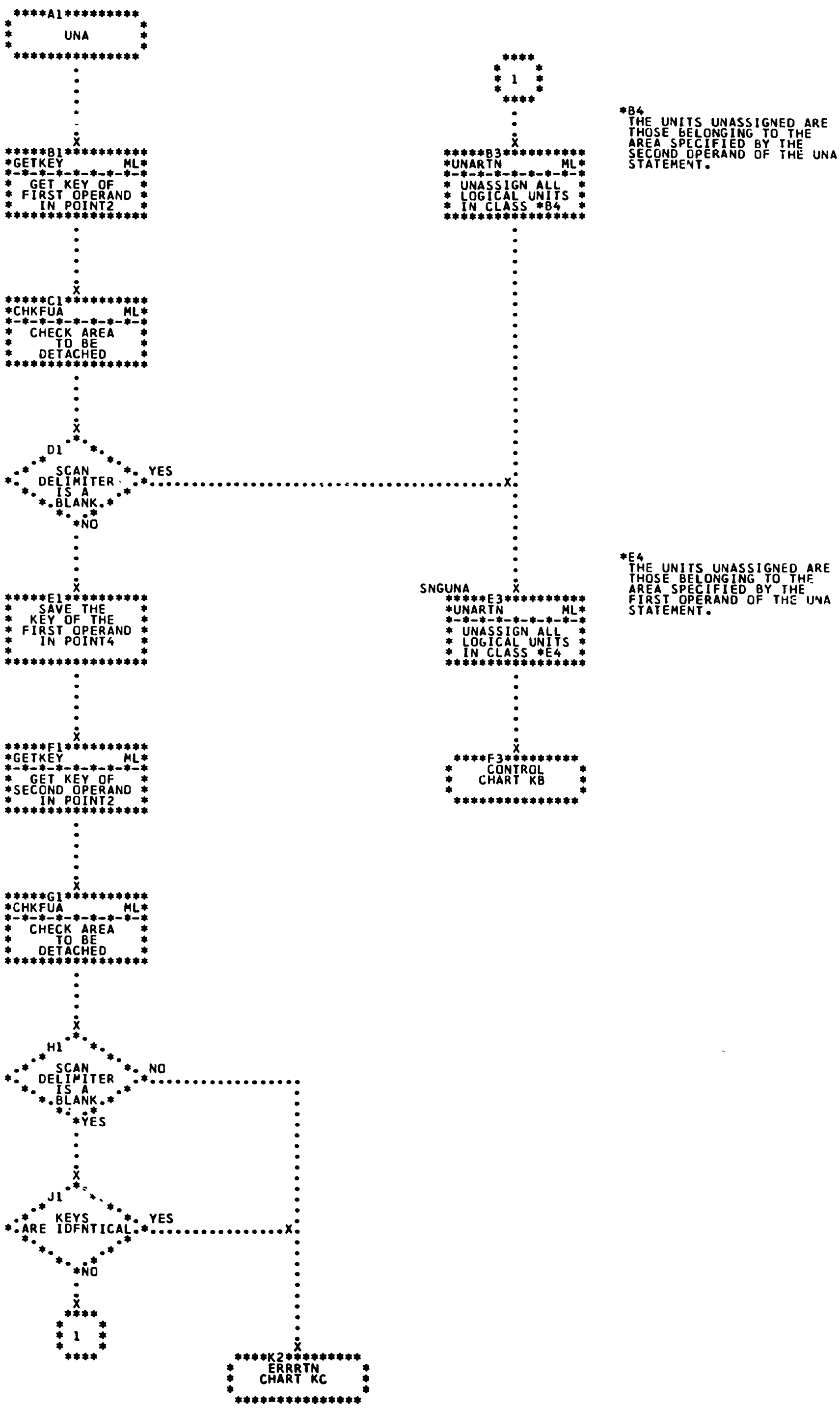


Chart MK. HOLD or RELSE Statement Processor $$BATTNI; Refer to Supervisor, Chart 22

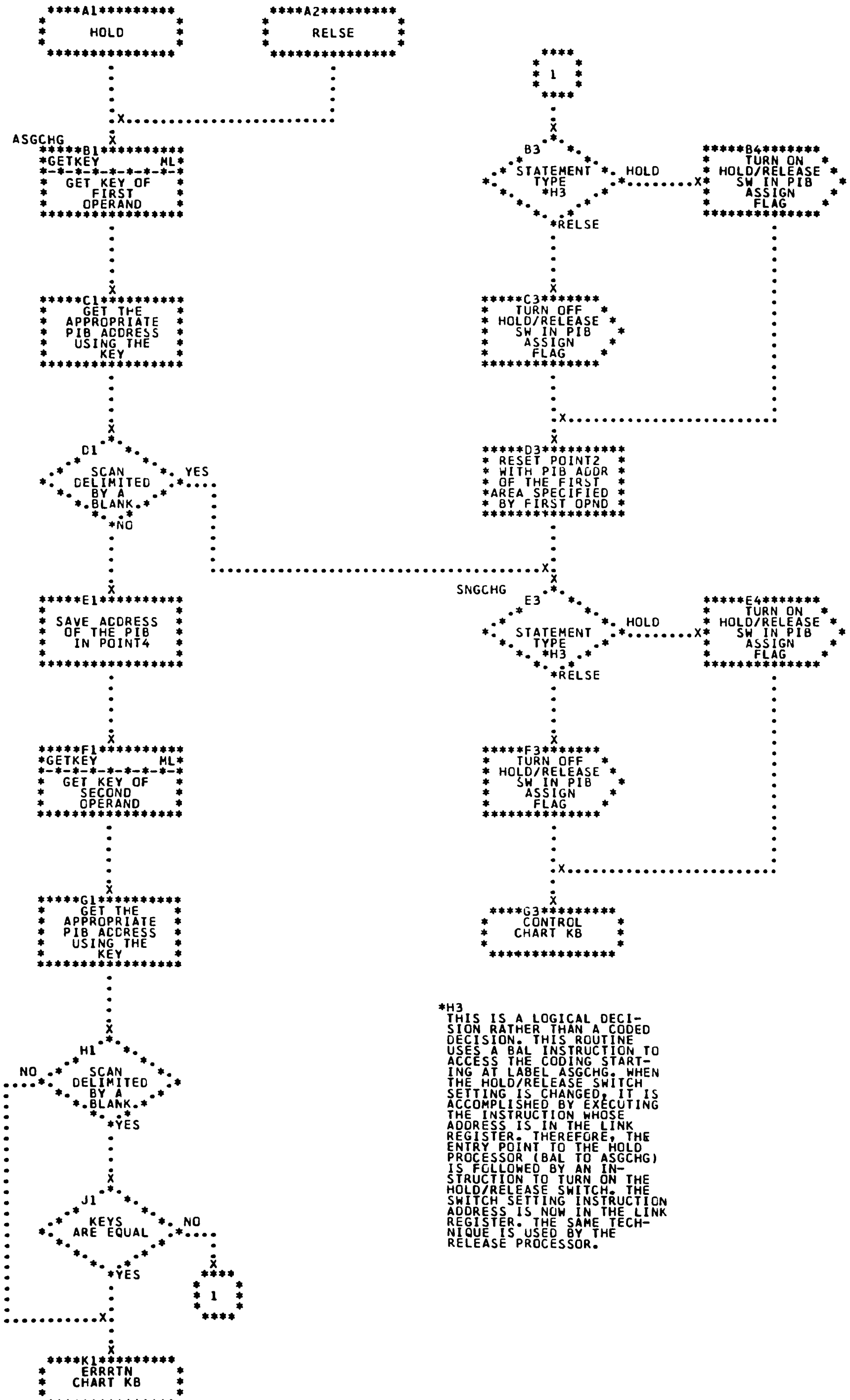


Chart ML. UNA, HOLD, RELSE Processor Subroutines $$BATTNI; Refer to Supervisor, Chart 22

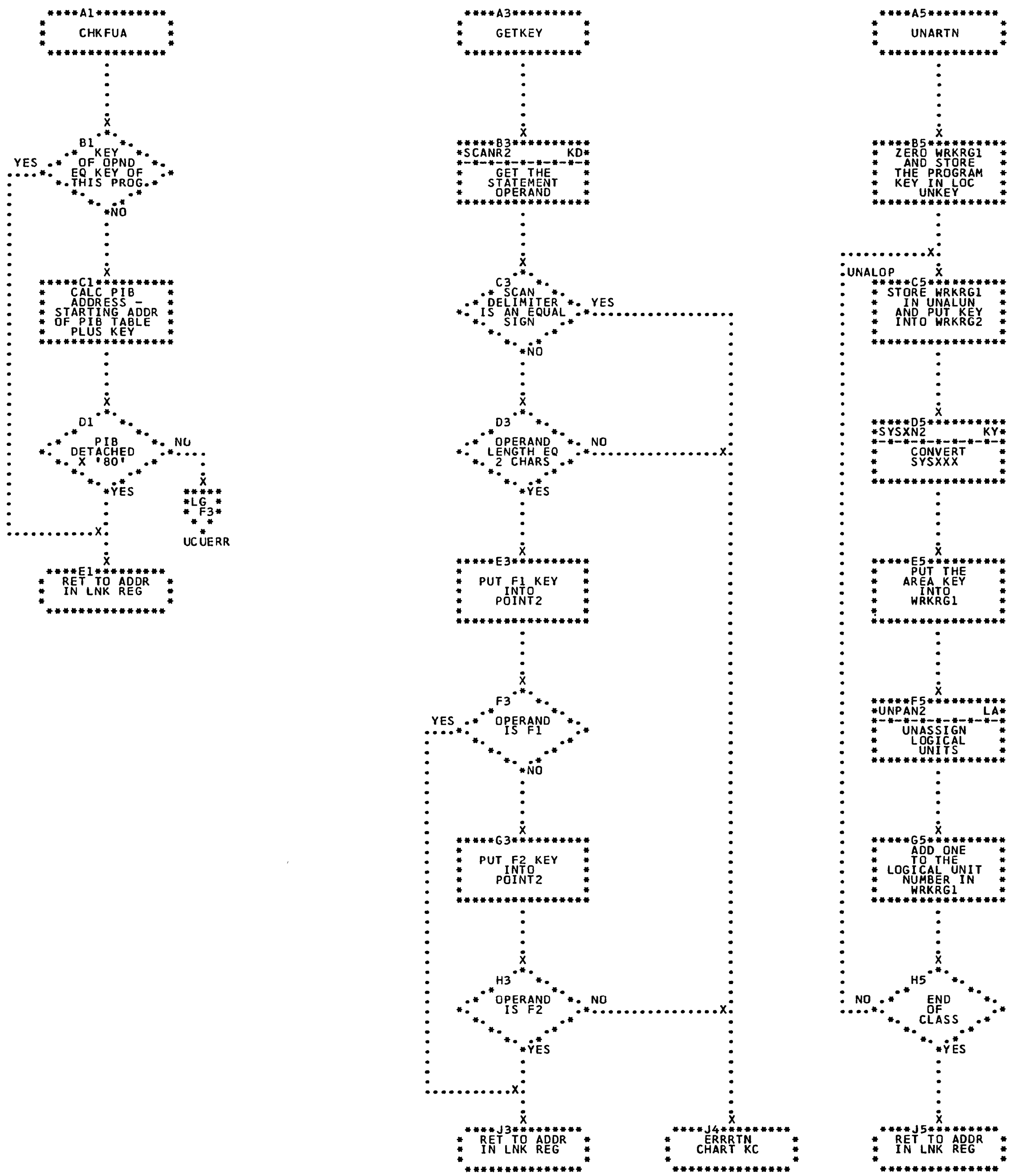


Chart NA. Terminated Program I/O Handling $$BEOJ; Refer to Supervisor, Chart 26

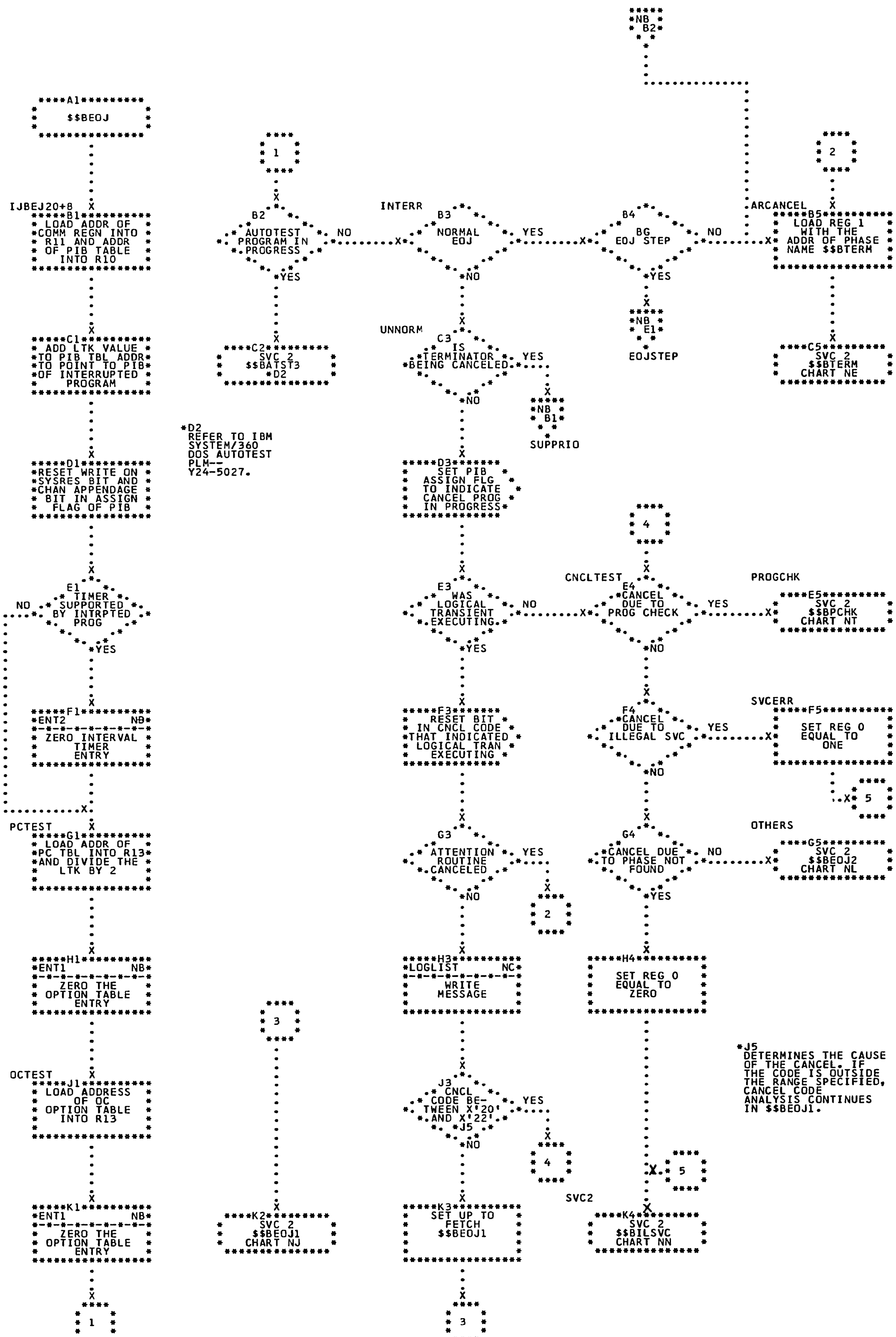


Chart NB. EOJ Processing Routine and $$BEOJ Subroutines
$$BEOJ; Refer to Supervisor, Chart 26

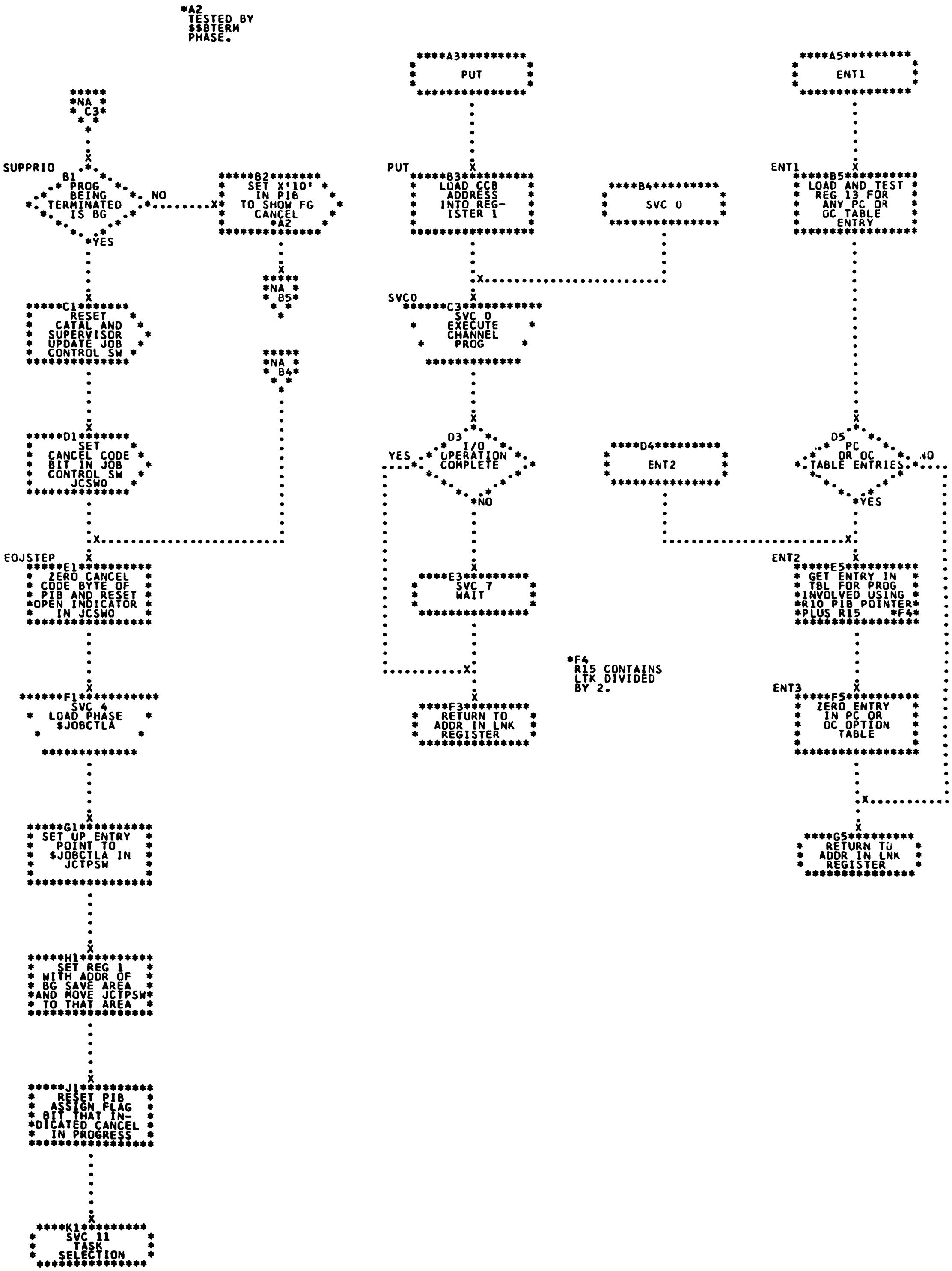


Chart NC. Message Output Subroutine $$BEOJ; Refer to Supervisor, Chart 26

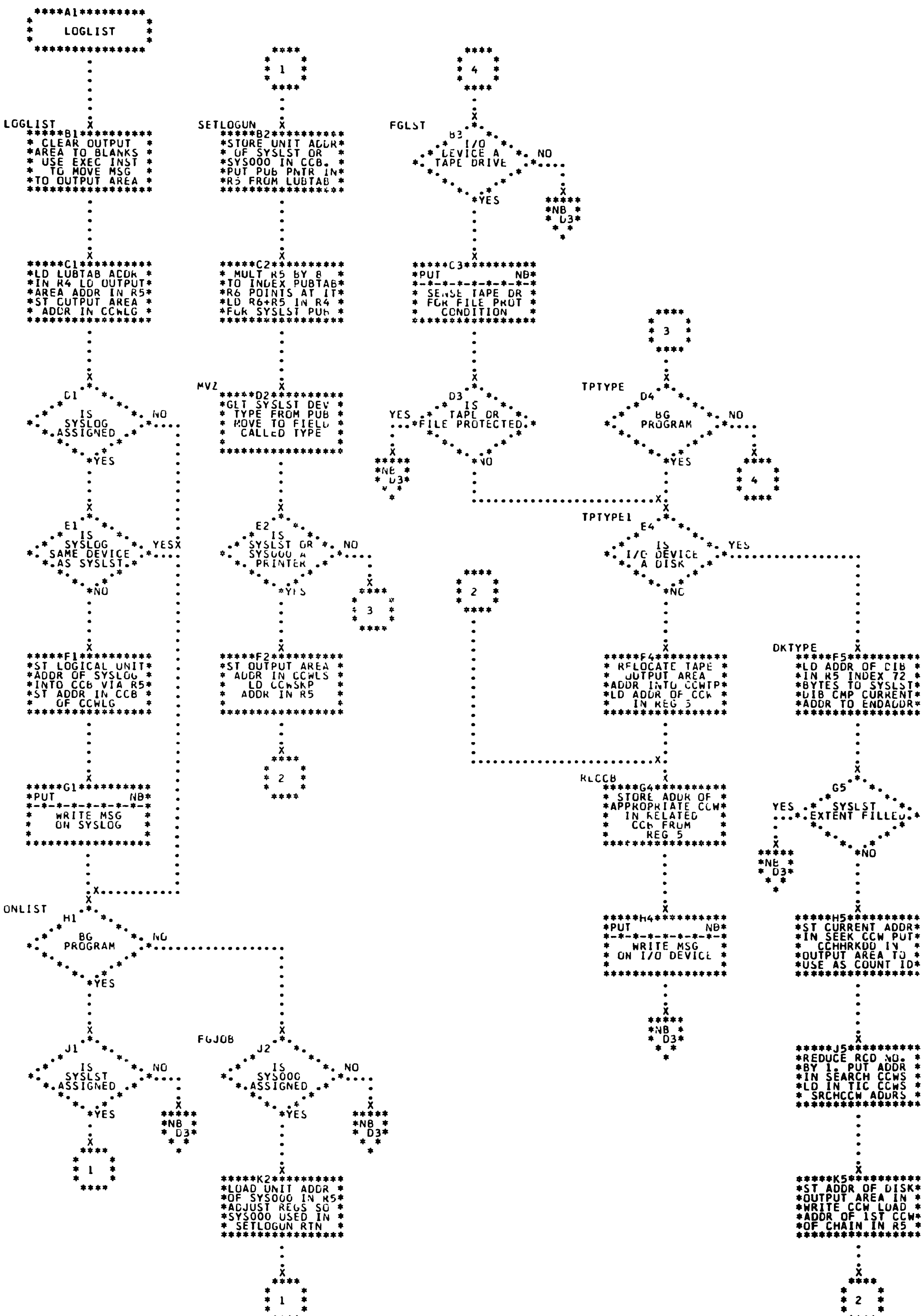


Chart ND. Quiesce I/O Phase $$BEOJ3; Refer to Supervisor, Chart 26

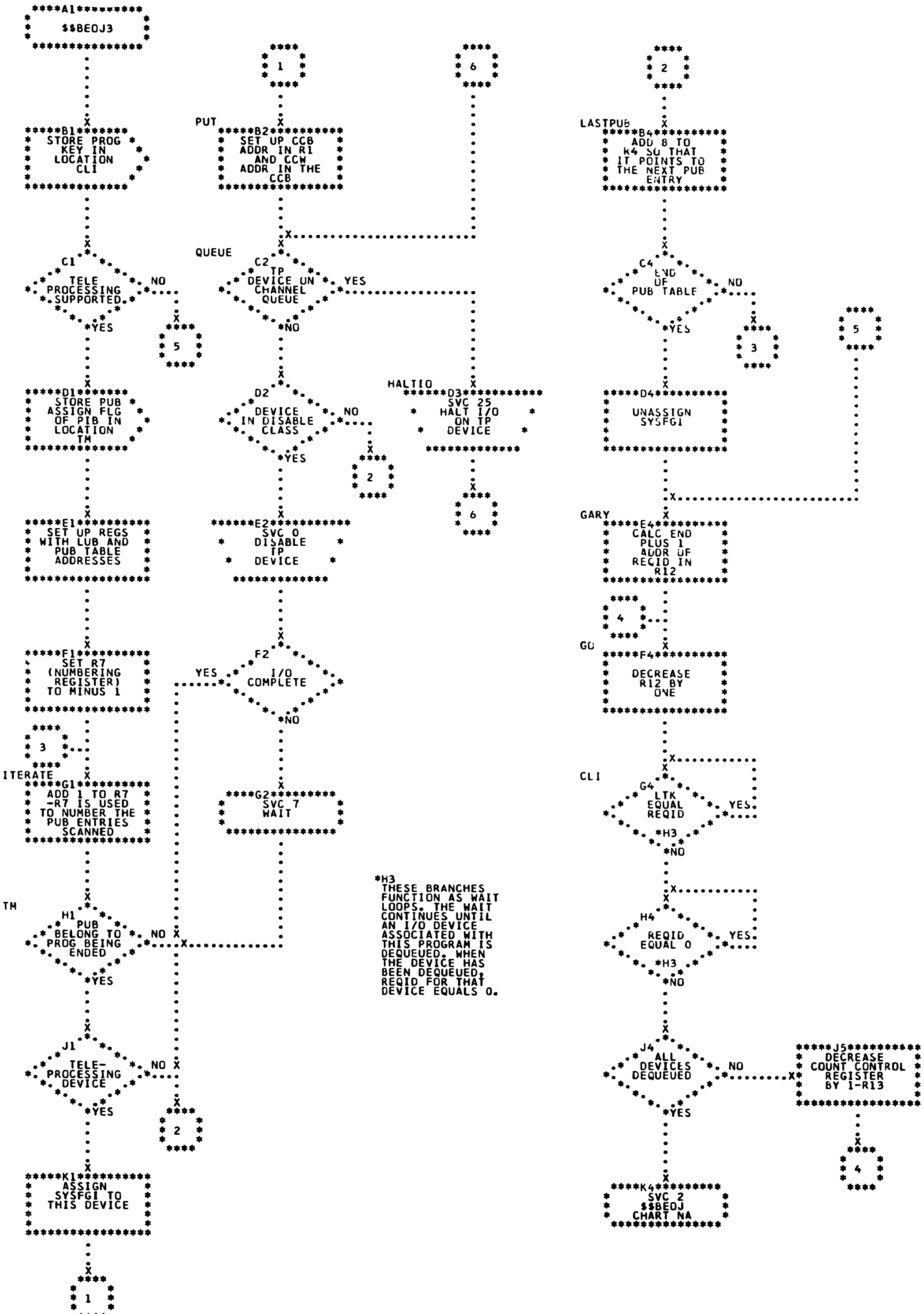


Chart NE. Reset Foreground PUB Ownership and Detach Attention Routine $$BTERM; Refer to Supervisor, Chart 26

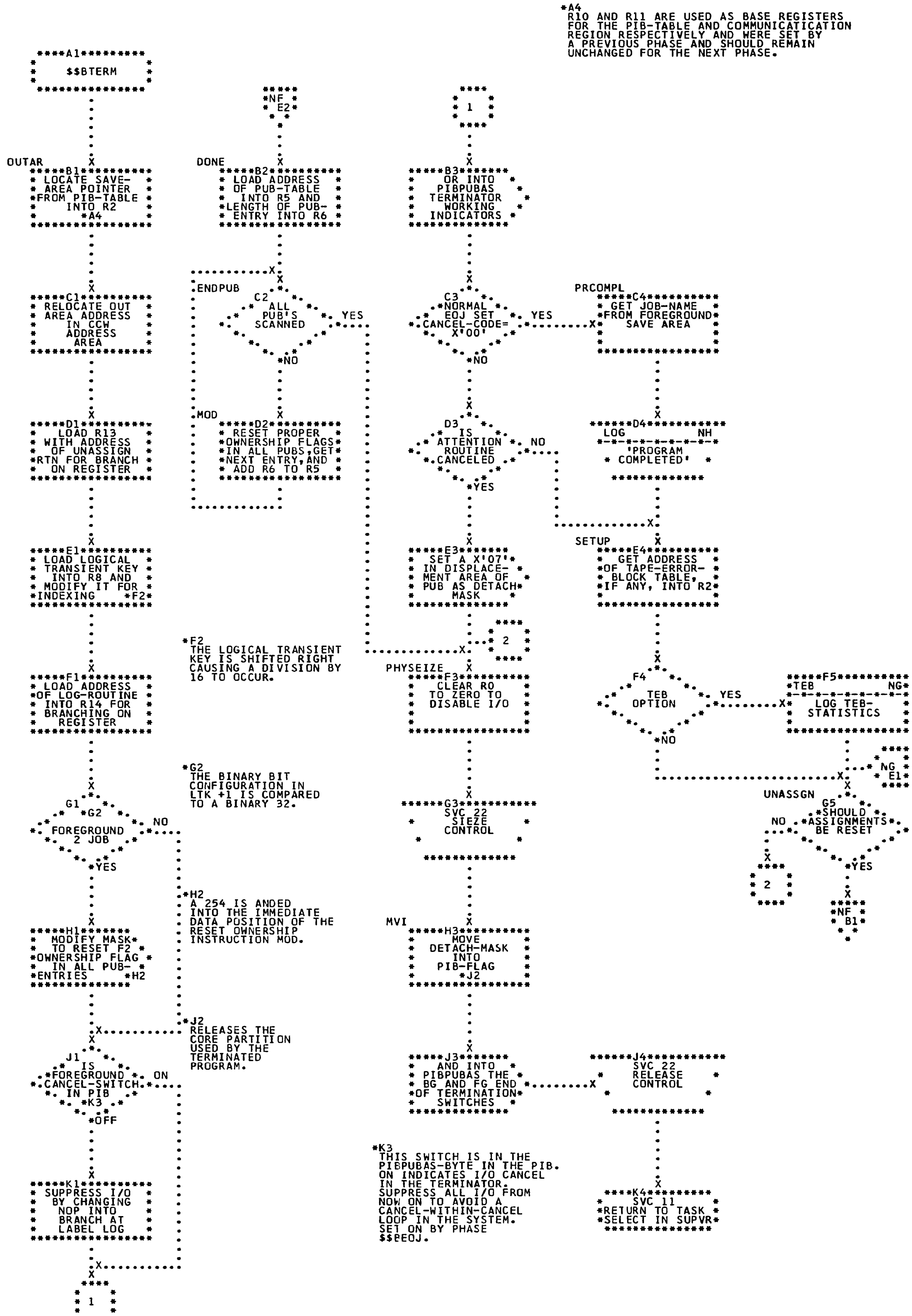


Chart NF. Reset JIB's for I/O Devices of Terminated Program
$$BTERM; Refer to Supervisor, Chart 26

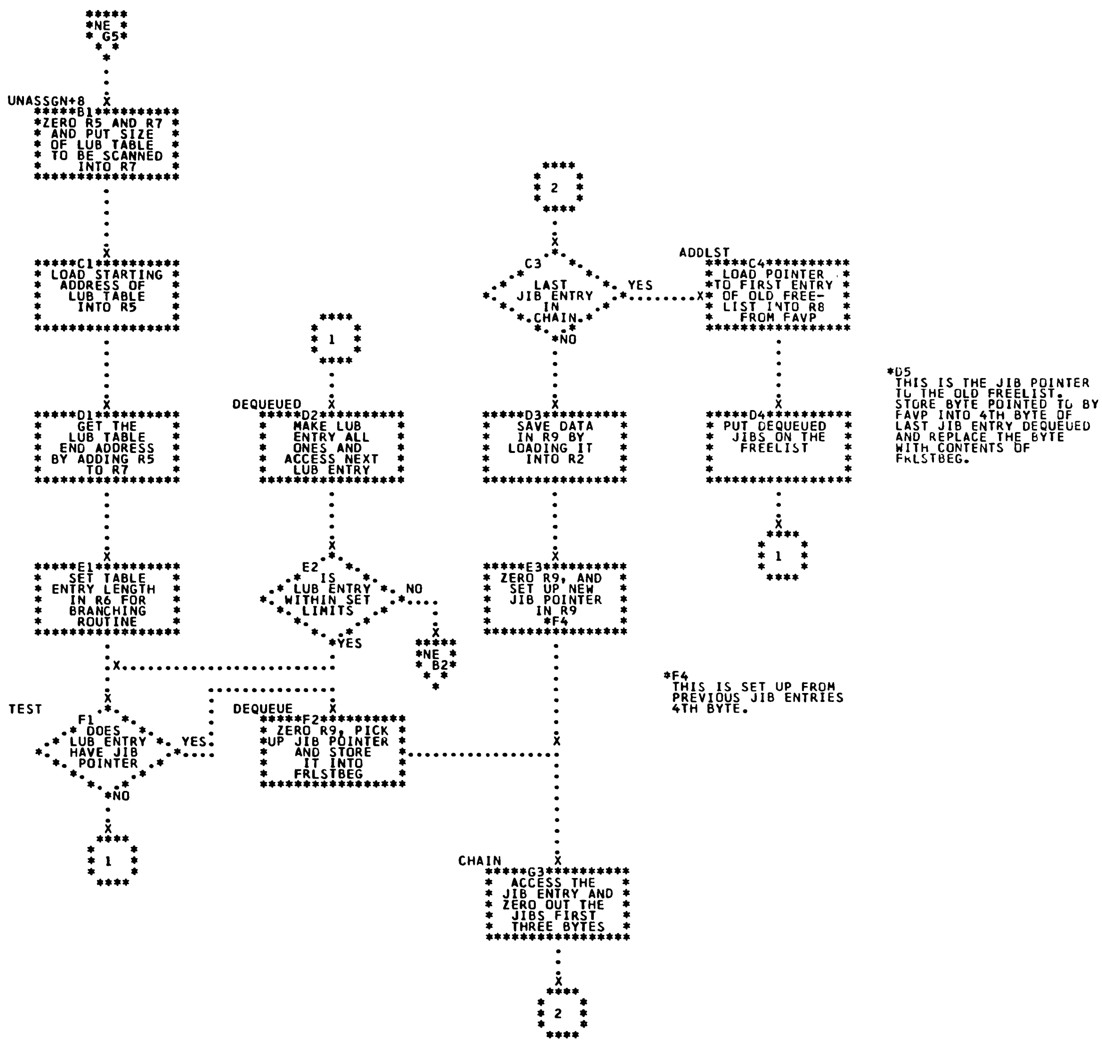


Chart NG. Get TEB Statistics and Reset TEB's $$BTERM; Refer to Supervisor, Chart 26

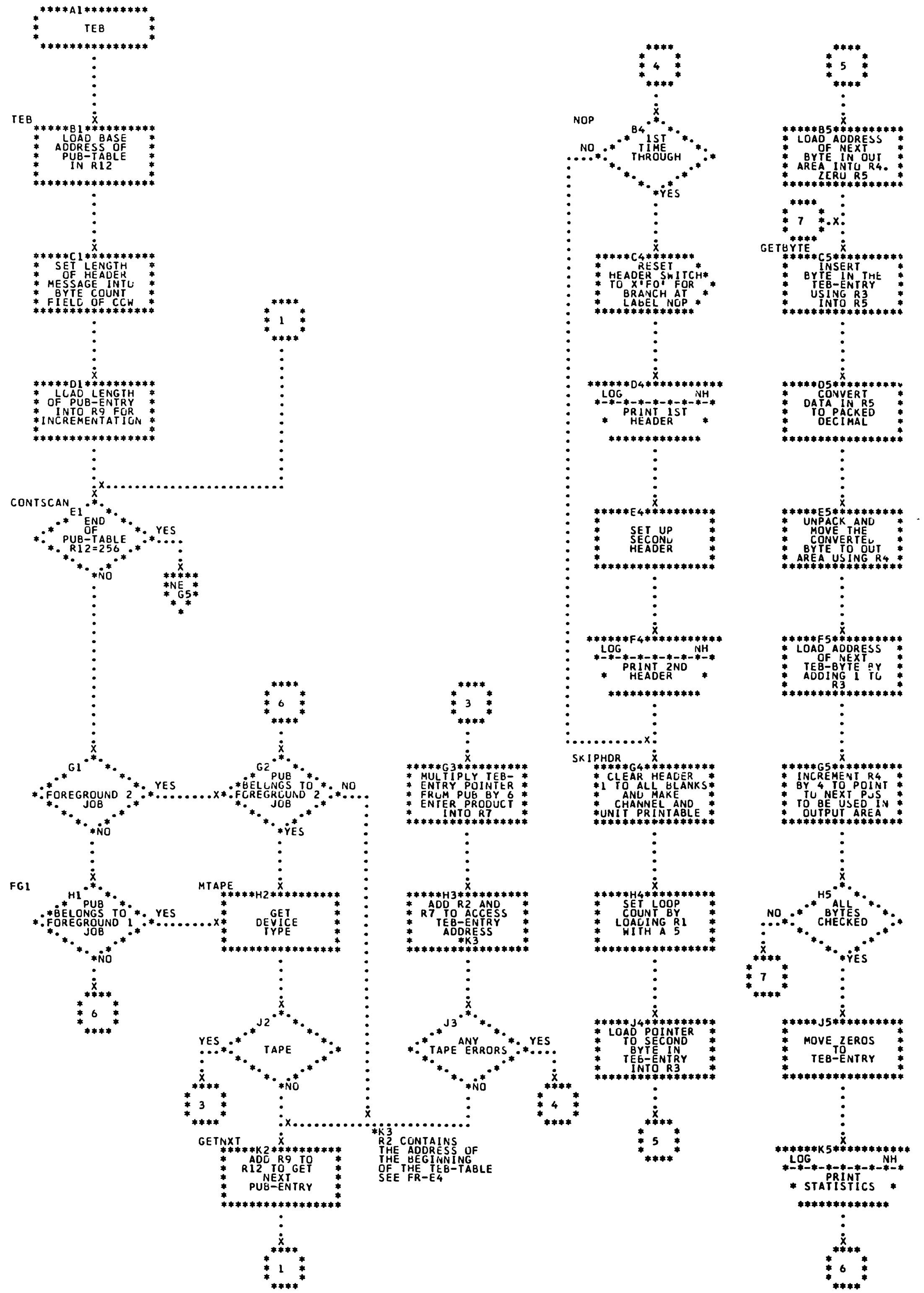


Chart NH. Print Message and TEB Statistics Subroutine
$$BTERM; Refer to Supervisor, Chart 26

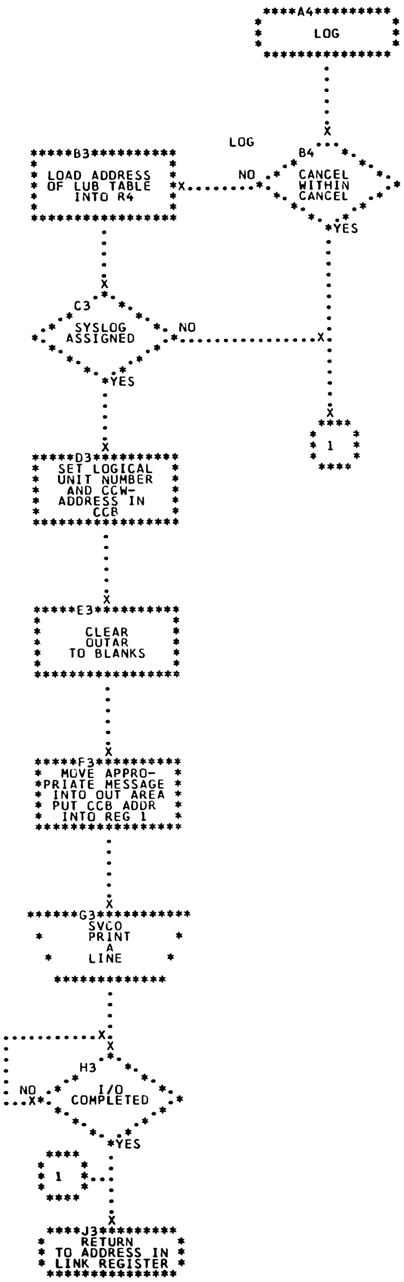


Chart NJ. Prepare Cancel Cause Message $$BEOJ1; Refer to Supervisor, Chart 27

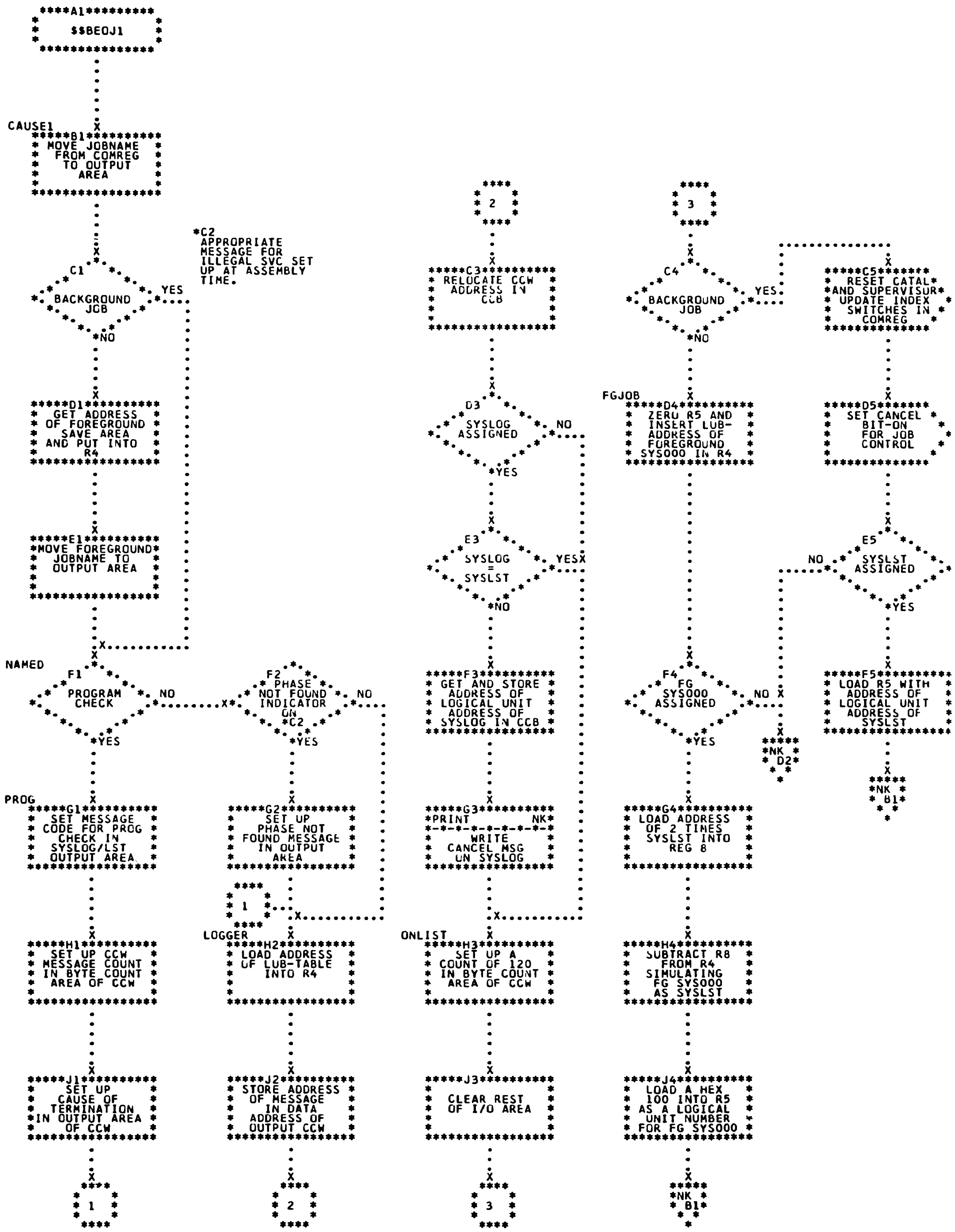


Chart NK. Output Cancel Message on SYSLST; $$BEOJ1; Refer to Supervisor, Chart 27

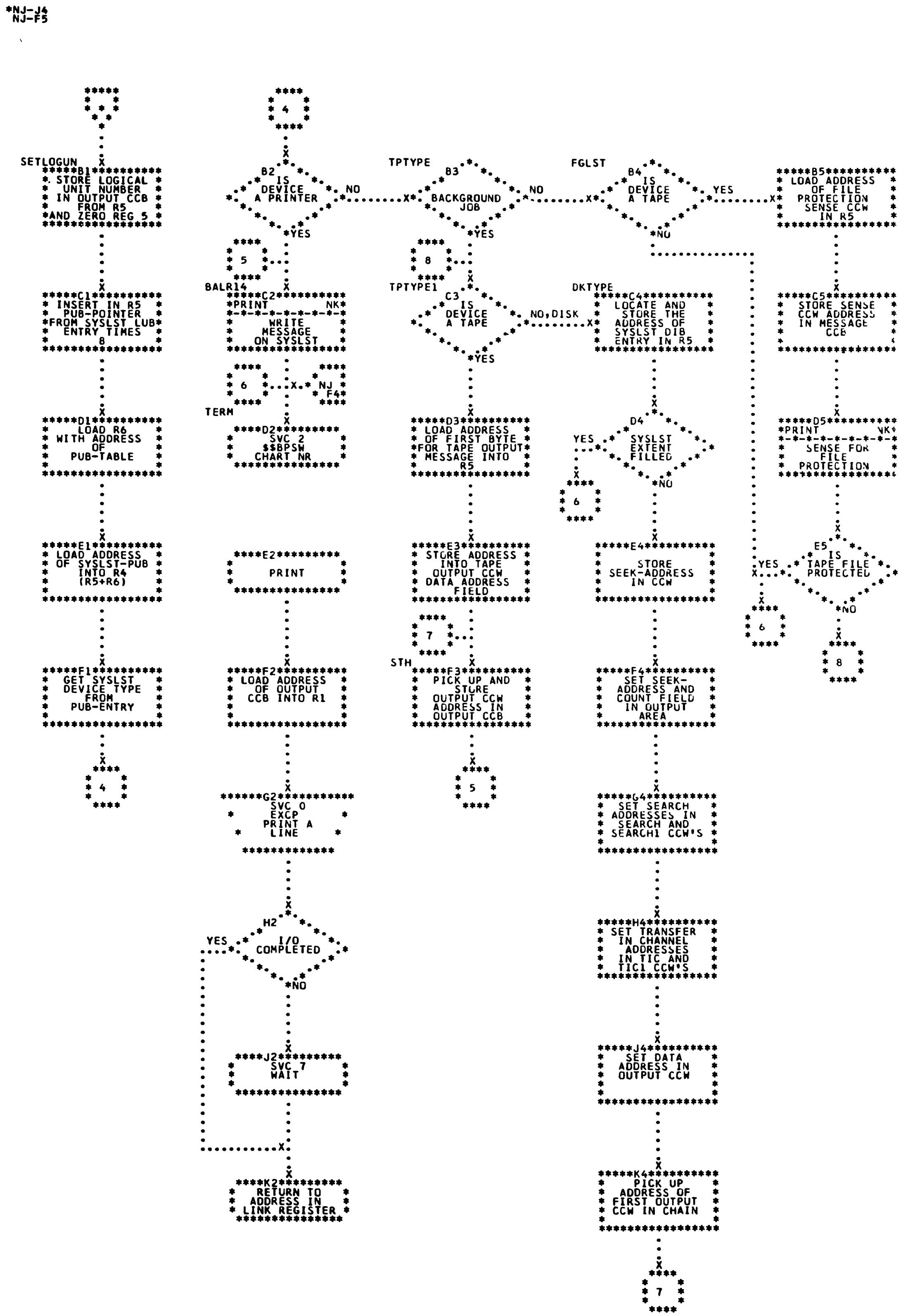


Chart NL. Select Cancel Message and Program Identification $$BEOJ2; Refer to Supervisor, Chart 28

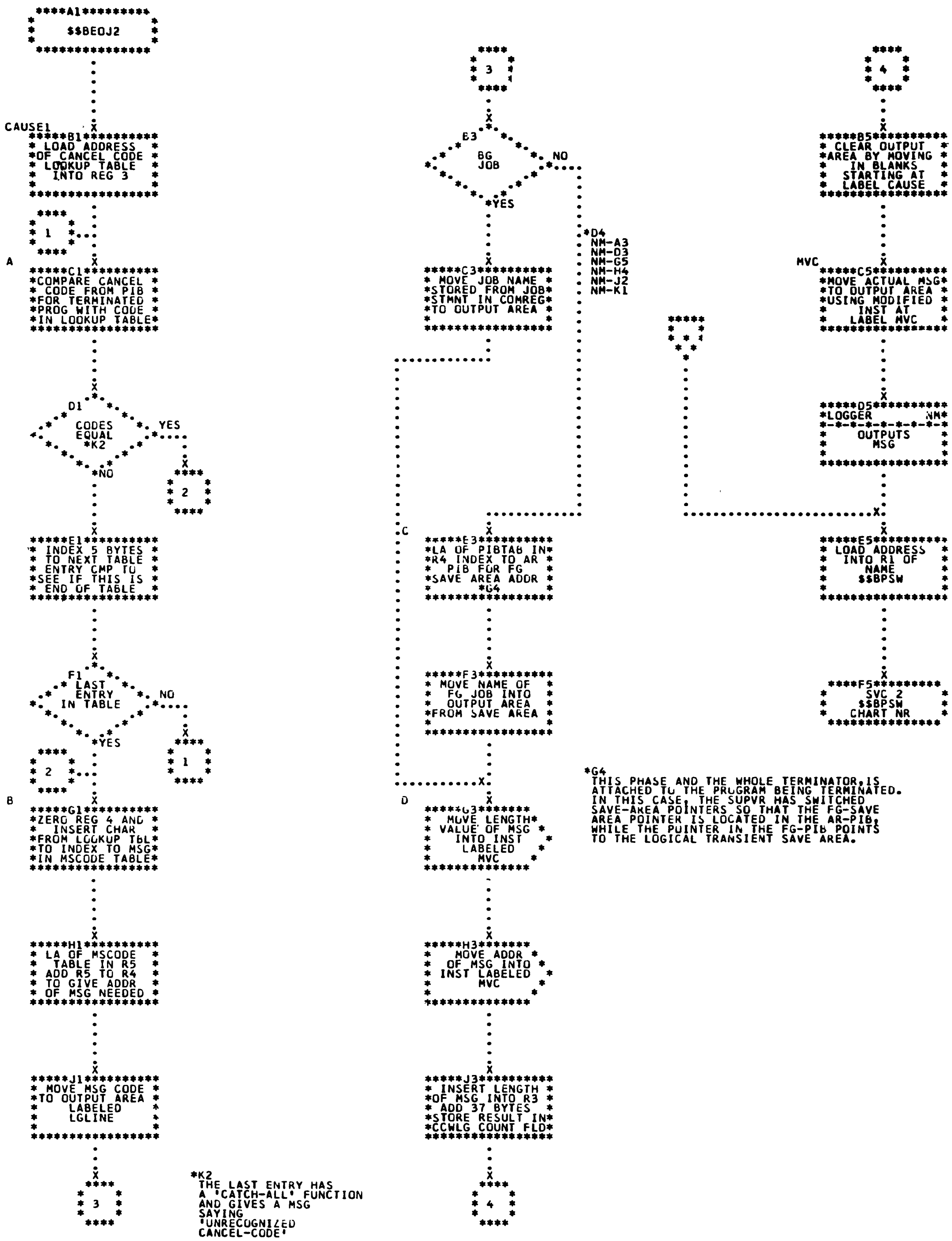


Chart NM. Select I/O Device and Output the Cancel Message $$BEOJ2; Refer to Supervisor, Chart 28

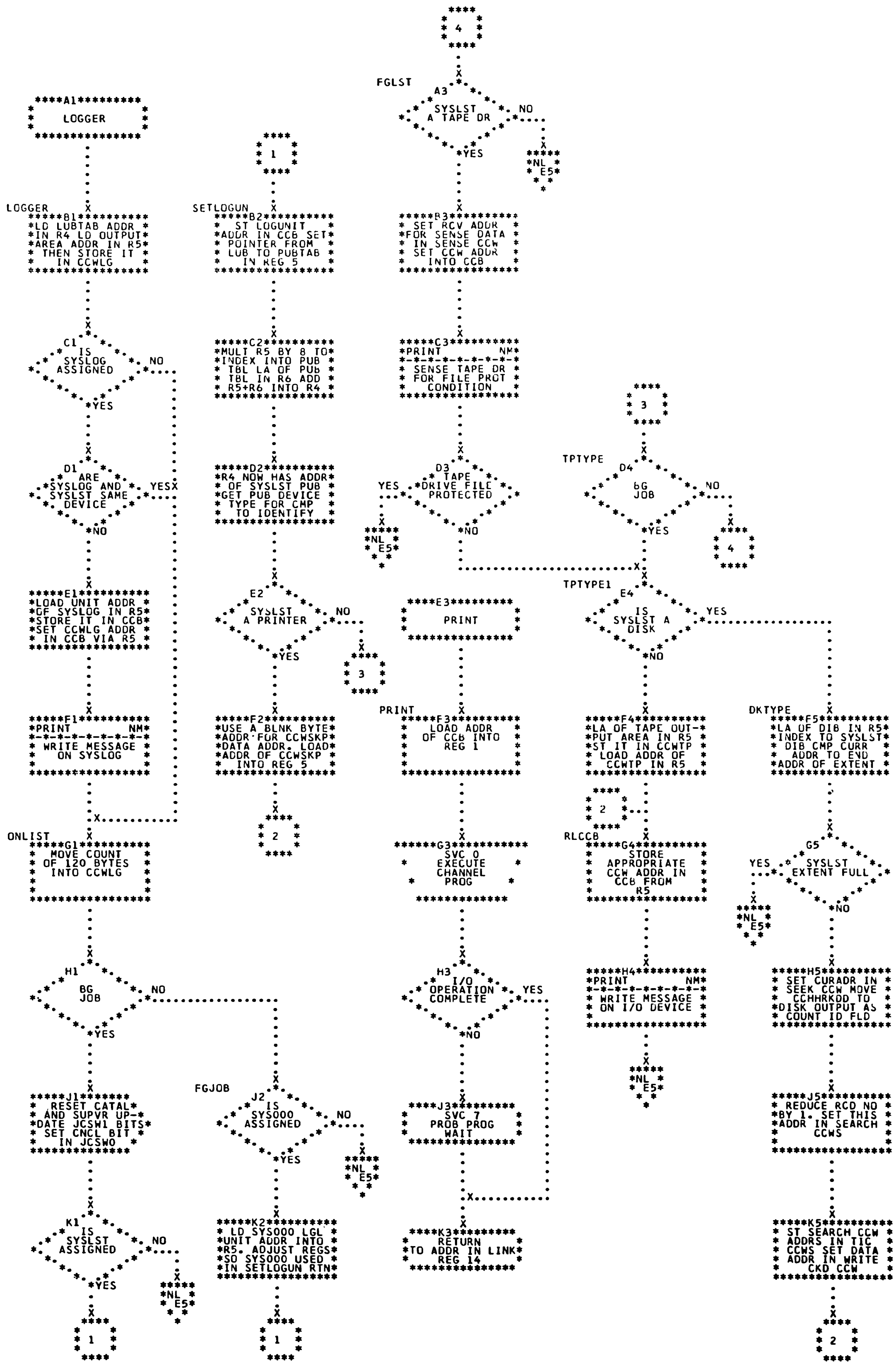


**Chart NN. Prepare Information About Cancel Cause $$BILSVC; Refer to Supervisor, Chart 28**

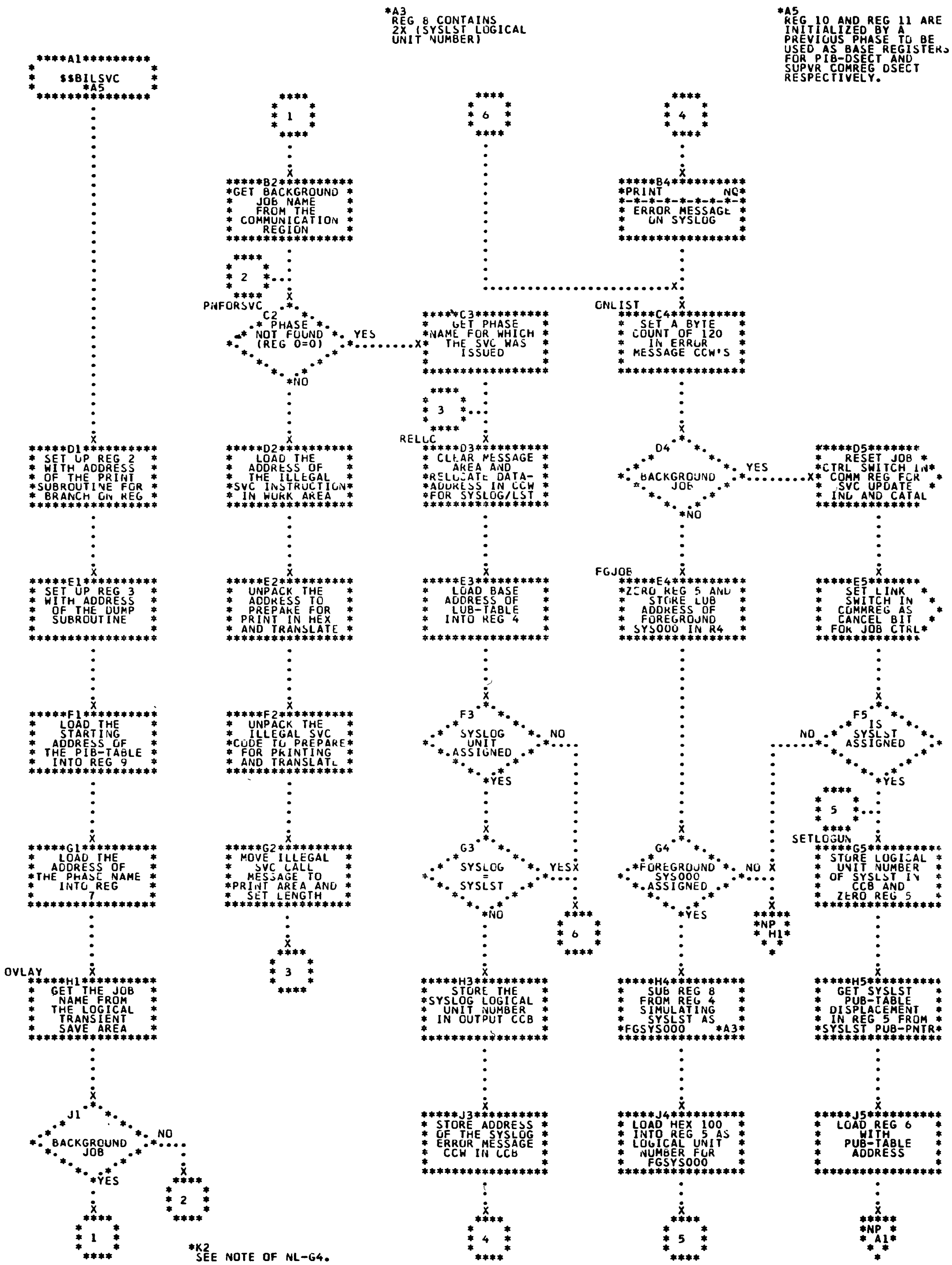


Chart NP. Select I/O Device and Prepare to Output a Message
$$BILSVC; Refer to Supervisor, Chart 28

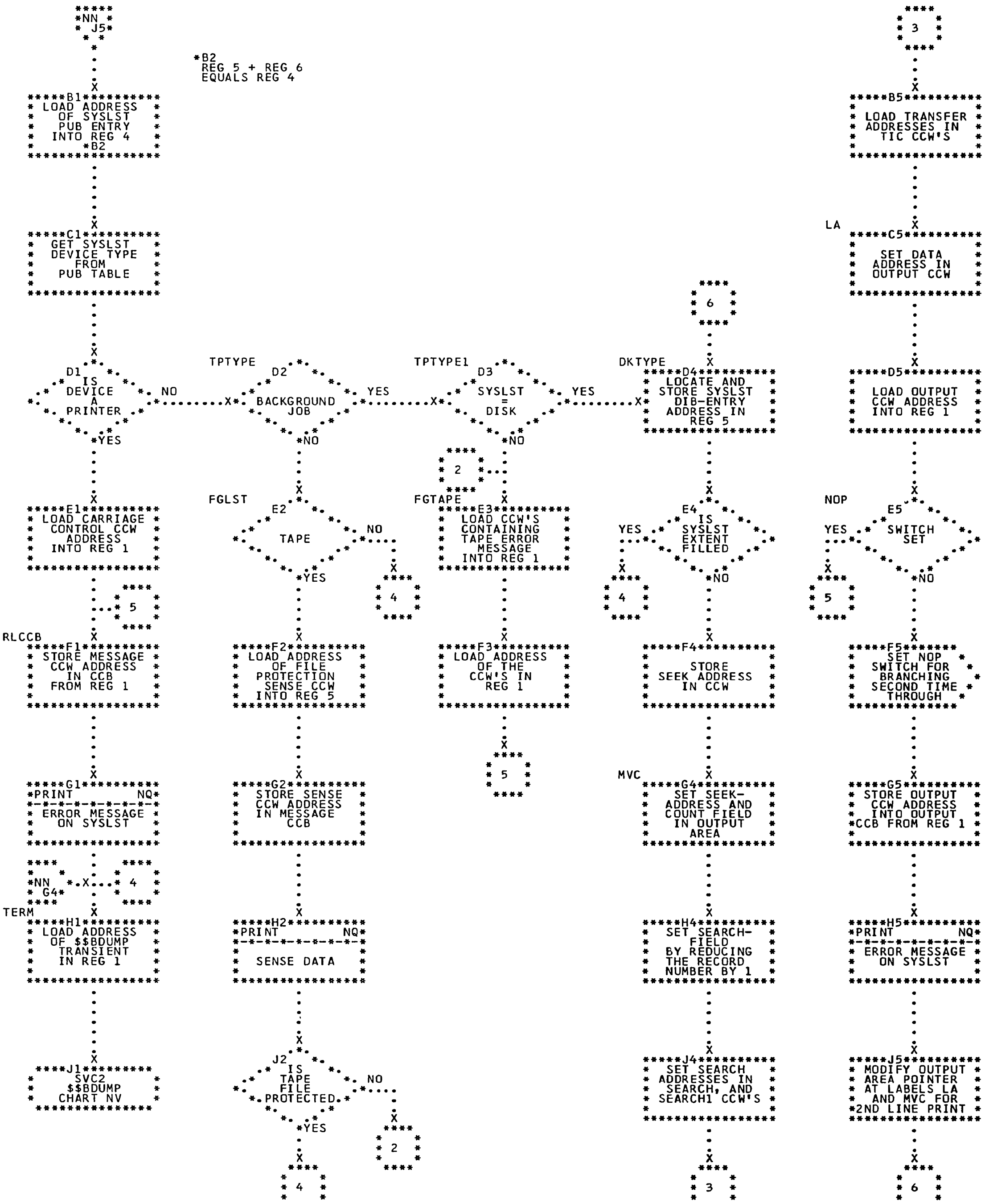


Chart NQ. Output Message on Selected I/O Device $$BILSVC; Refer to Supervisor, Chart 28

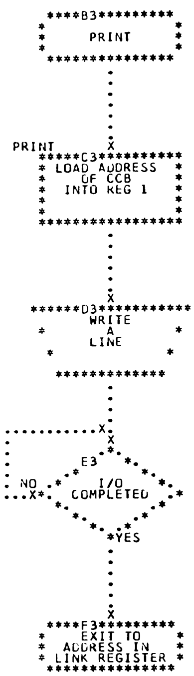


Chart NR. Prepare Canceled Program's PSW for Output Message and PIOCS Subroutine $$BPSW; Refer to Supervisor, Chart 27

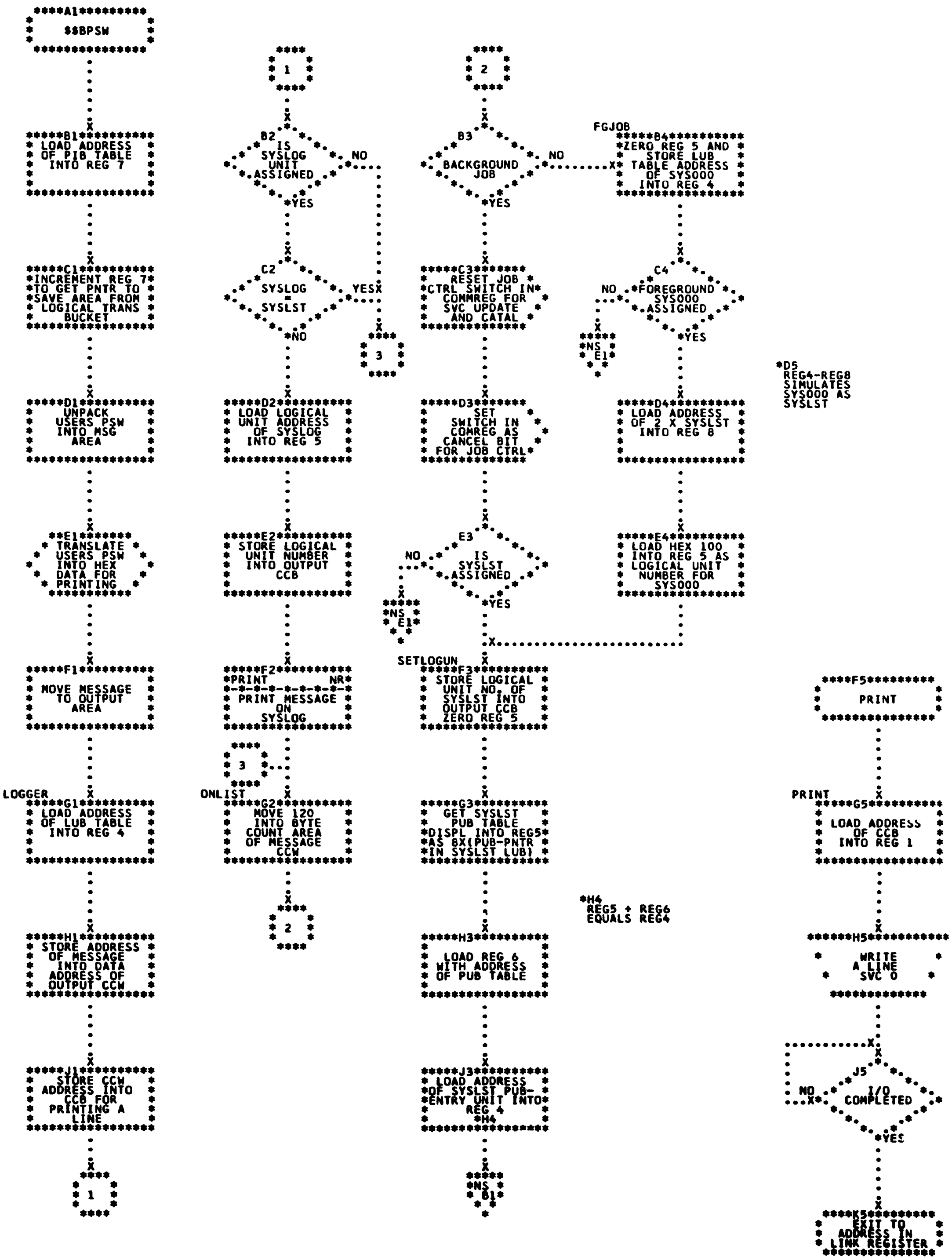


Chart NS. Select I/O Device and Prepare to Output a Message
$$BPSW; Refer to Supervisor, Chart 27

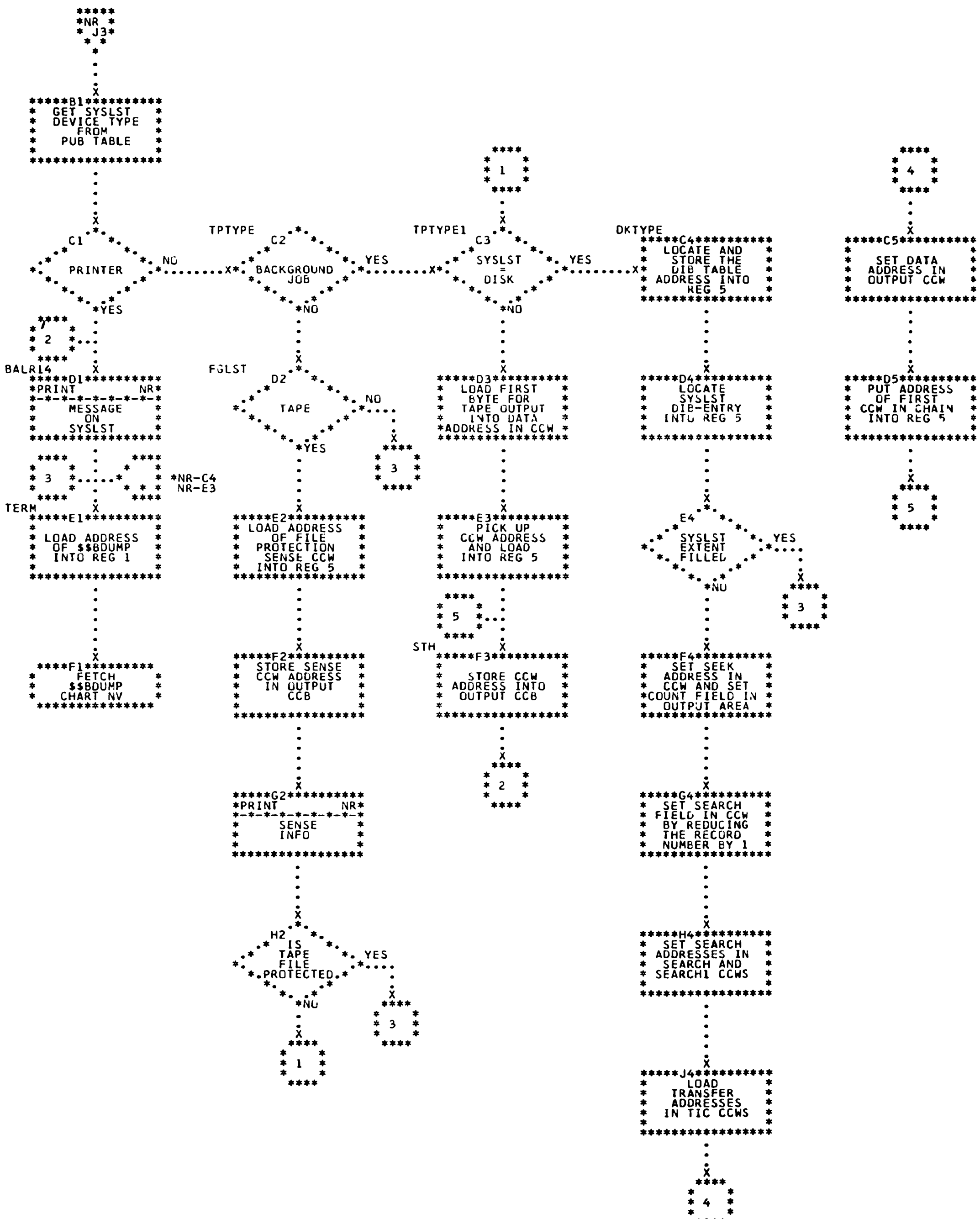


Chart NT. Prepare Information for Message about PC Cancel and Select I/O Device $$BPCHK; Refer to Supervisor, Chart 28

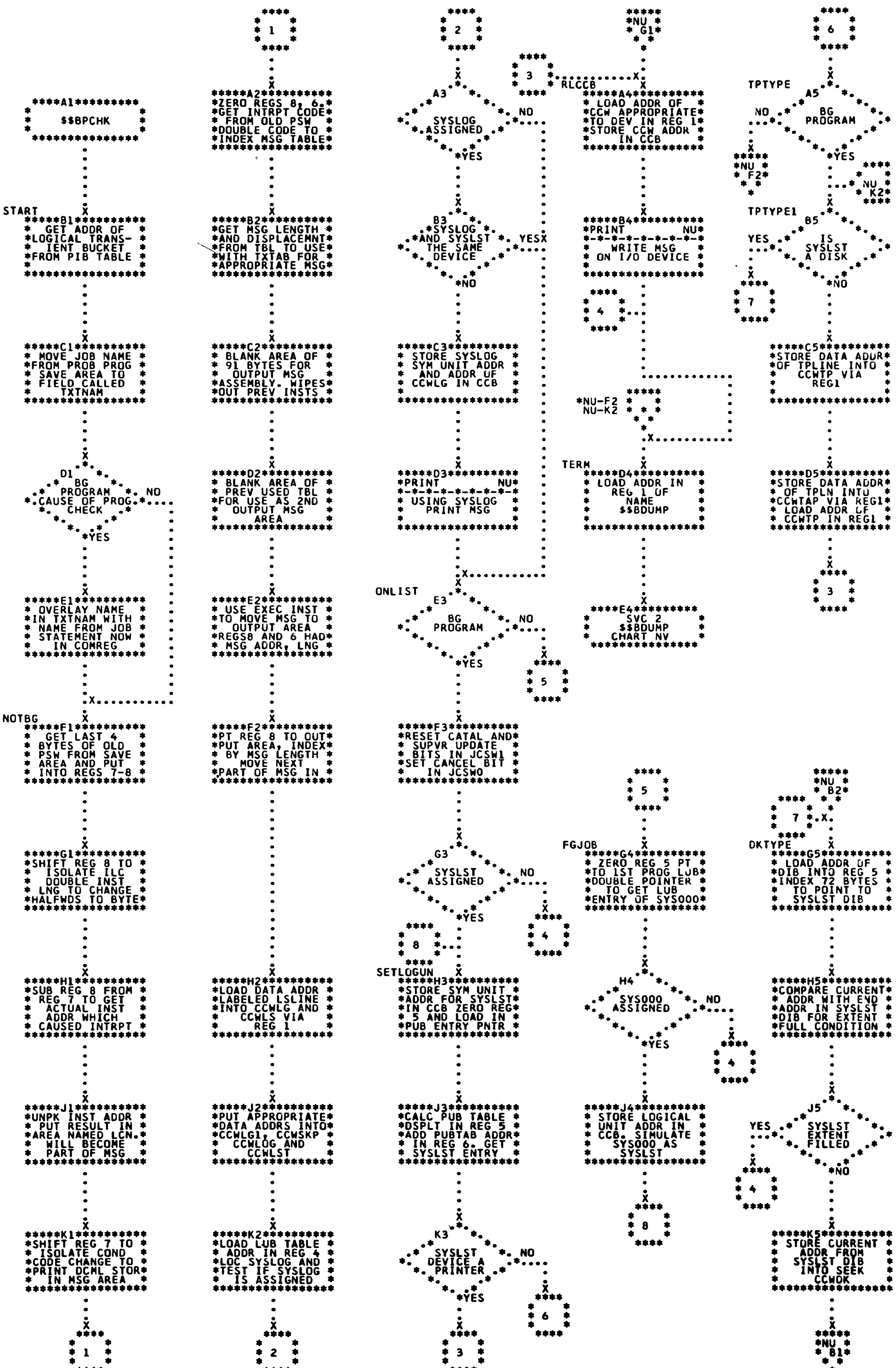


Chart NU. Set Up for I/O and Output the Message $$BPCHK; Refer to Supervisor, Chart 28

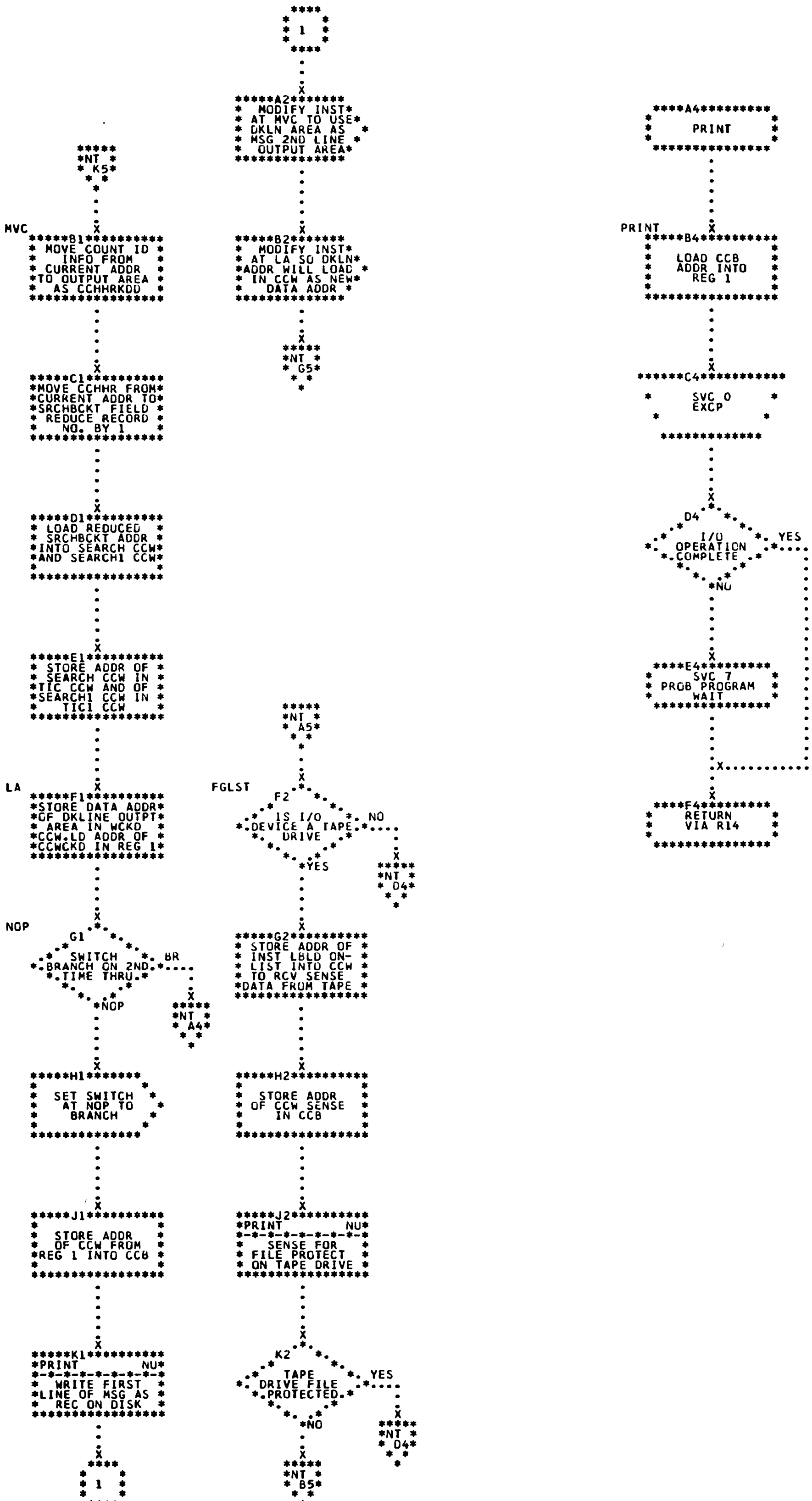


Chart NV. Monitor Background Program Dump $$BDUMP; Refer to Supervisor, Chart 27

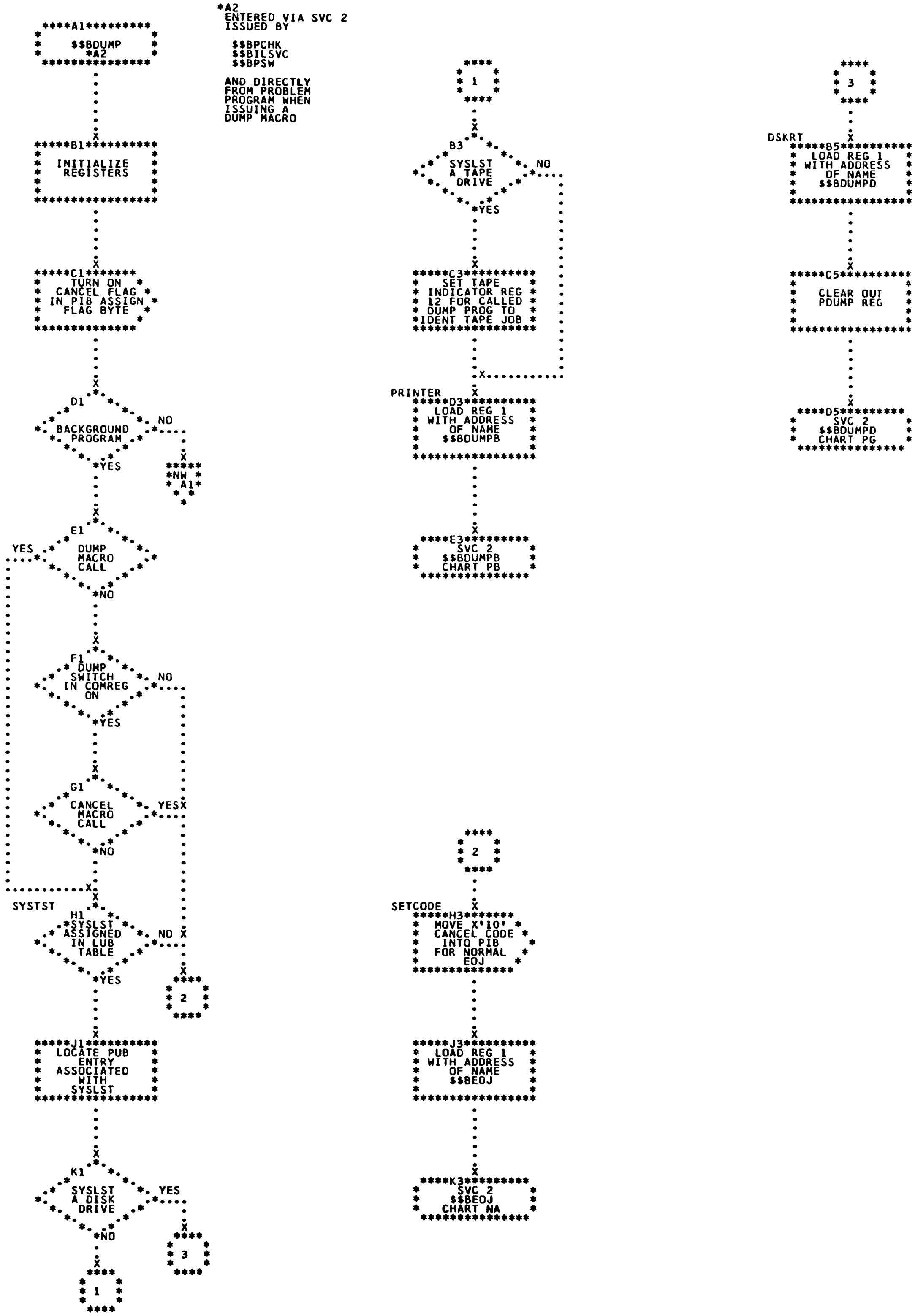


Chart NW. Monitor Foreground Program Dump $$BDUMP; Refer to Supervisor, Chart 27

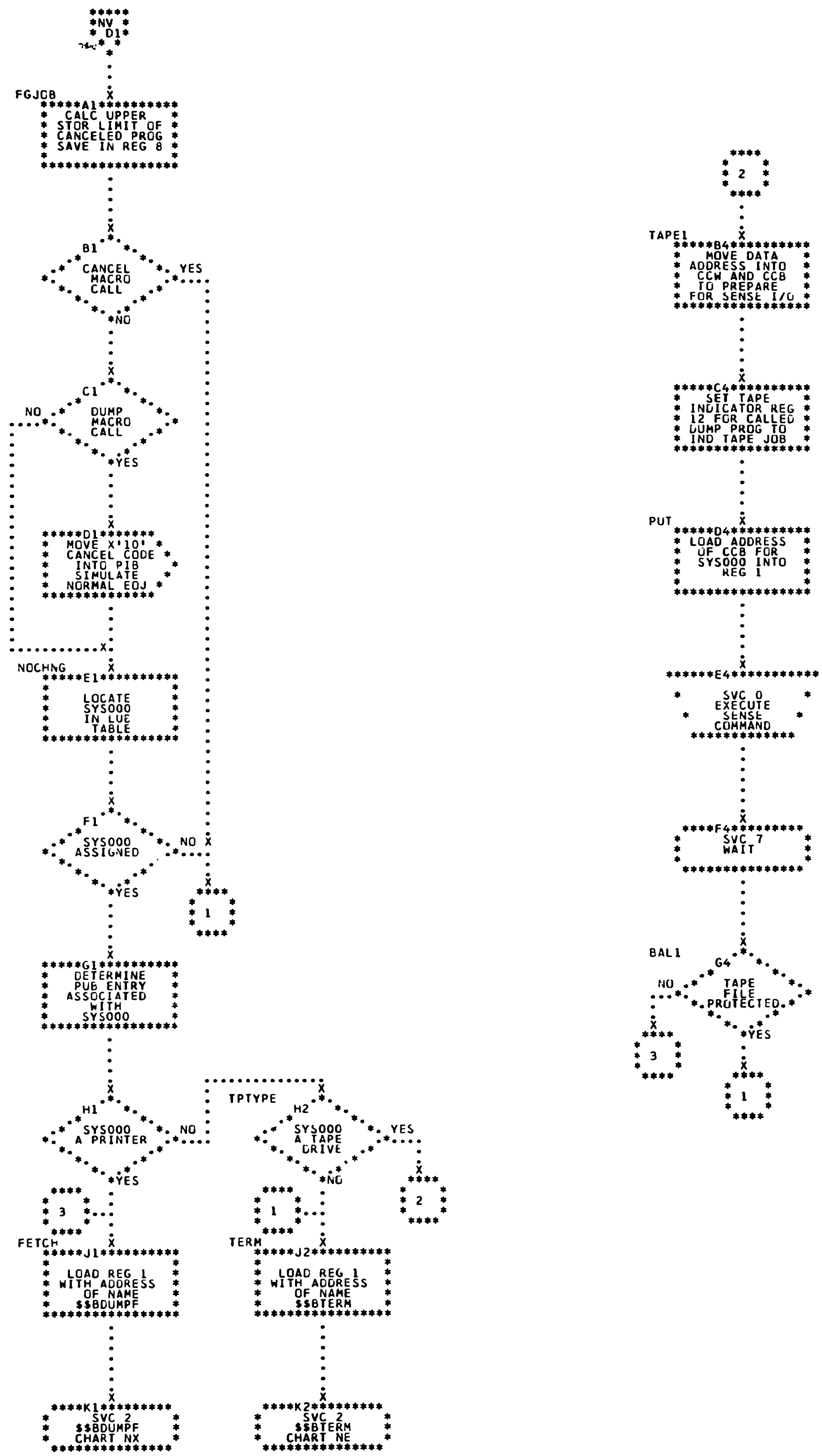


Chart NX. Foreground Program Dump $$BDUMPF; Refer to Supervisor, Chart 29

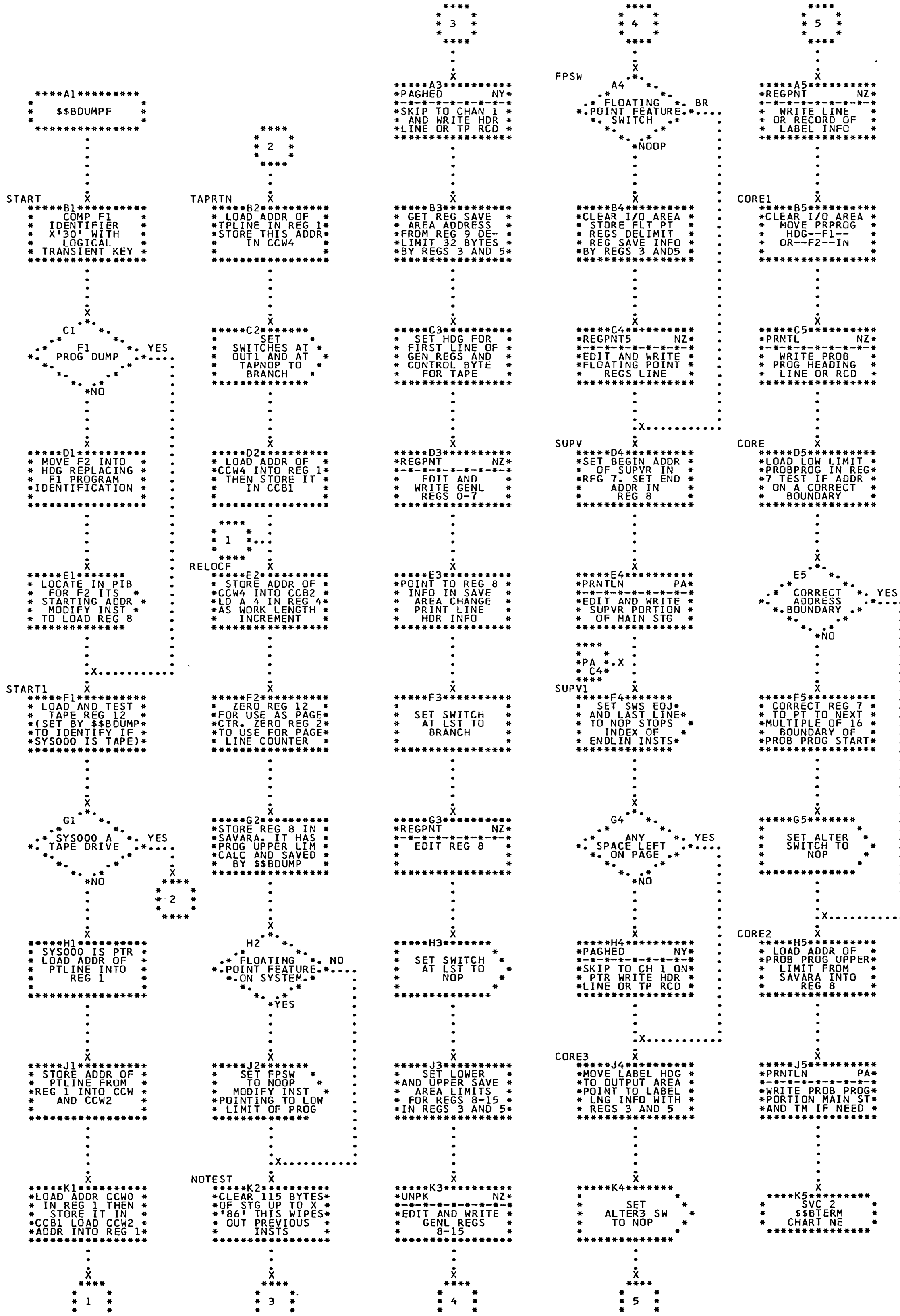


Chart NY. Prepare Page Headings and PIOCS Subroutines
$$BDUMPF; Refer to Supervisor, Chart 29

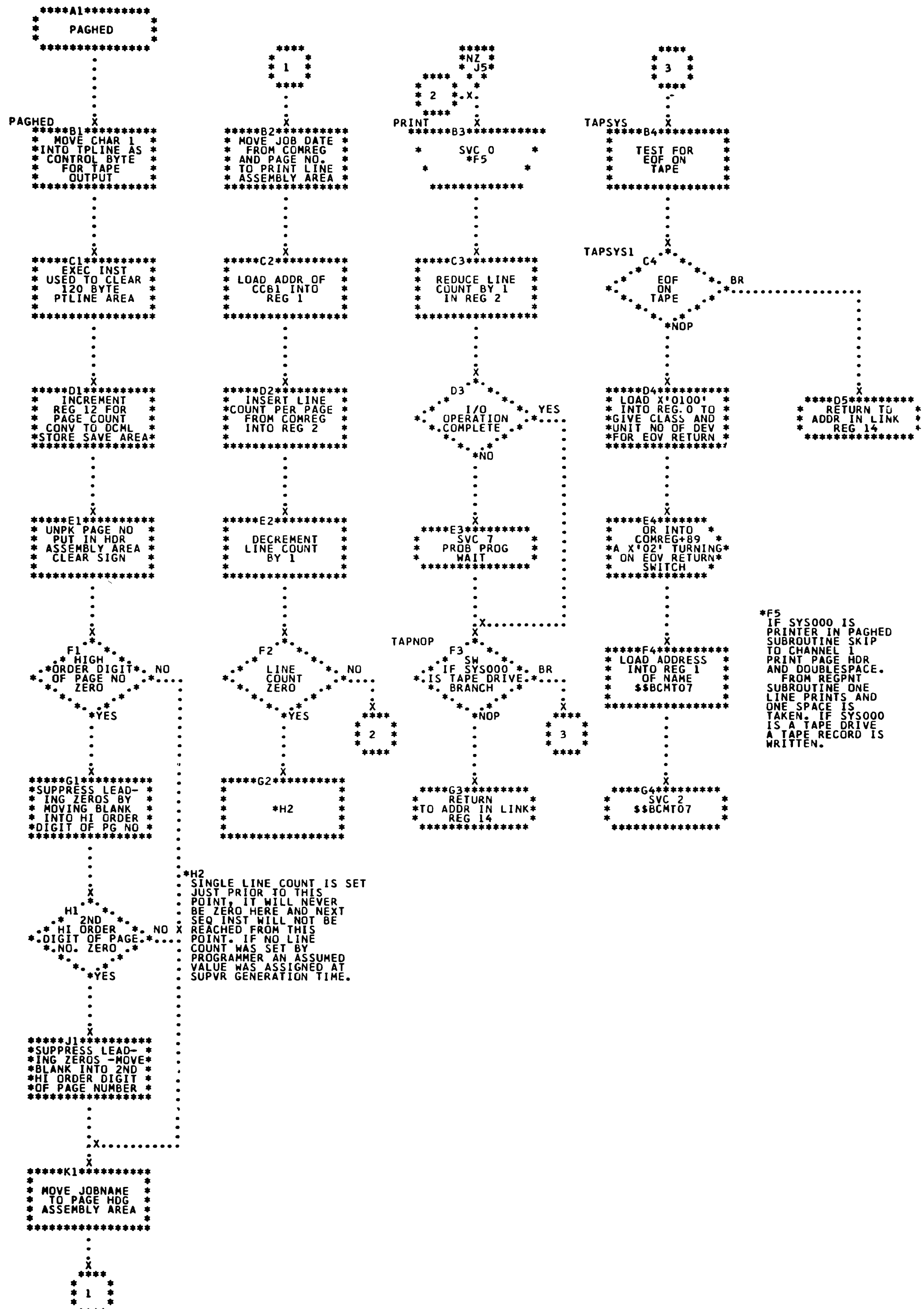


Chart NZ. Prepare and Edit a Line Subroutine $$BDUMPF;
Refer to Supervisor, Chart 29

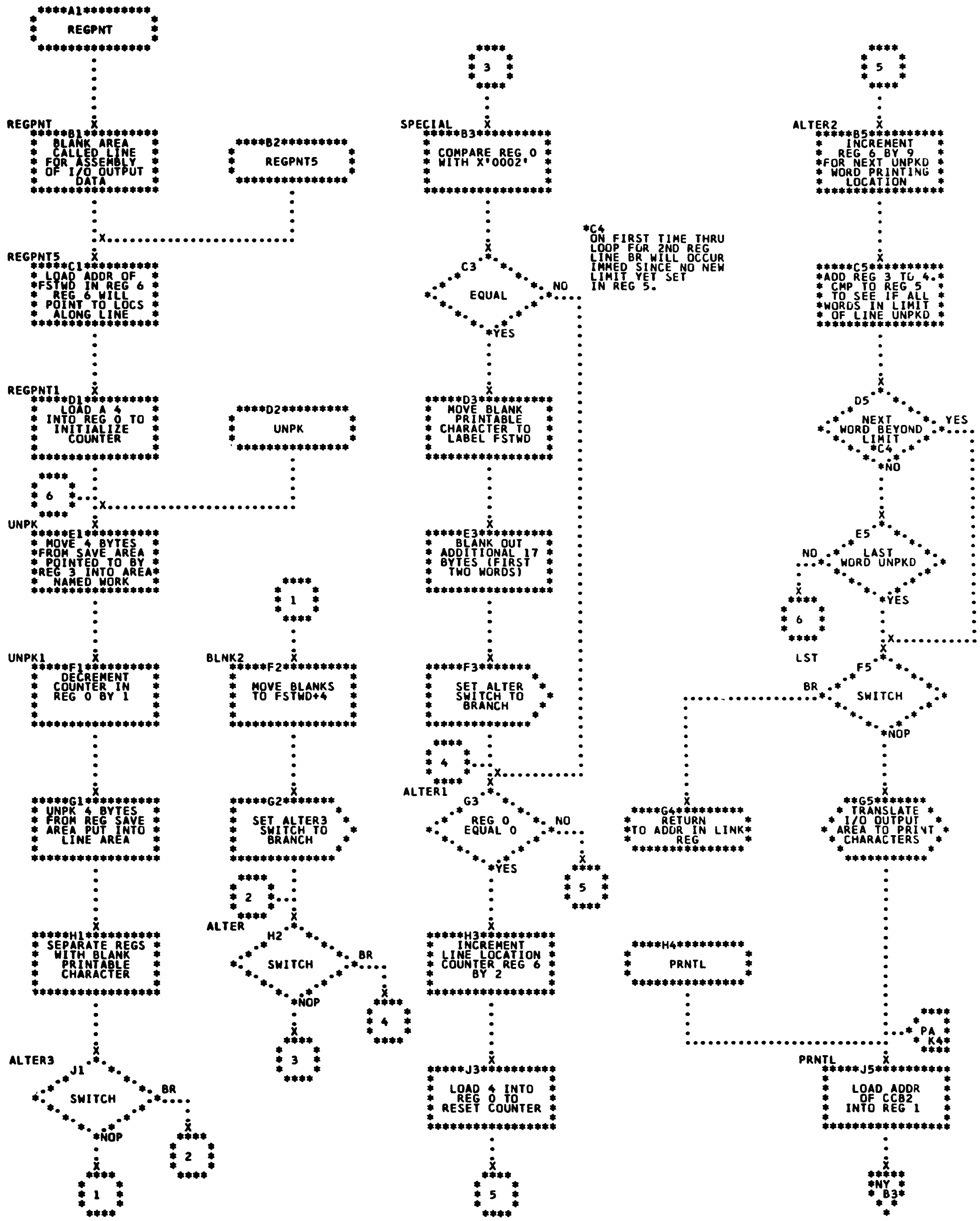


Chart PA. Line Test Subroutines $$BDUMPF; Refer to Supervisor, Chart 29

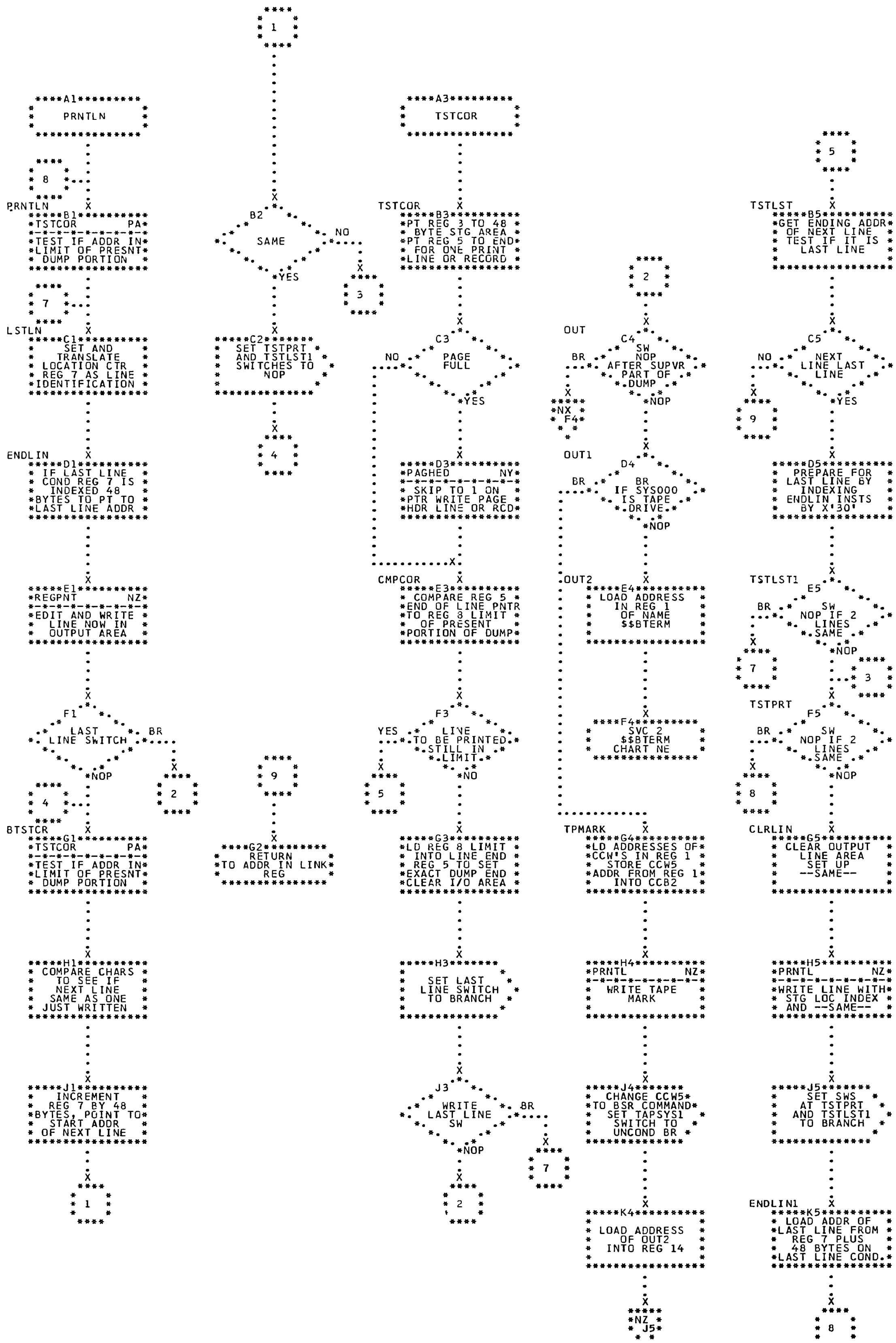


Chart PB. Initialize for BG Storage Dump on Printer or Tape
$$BDUMPB; Refer to Supervisor, Chart 29

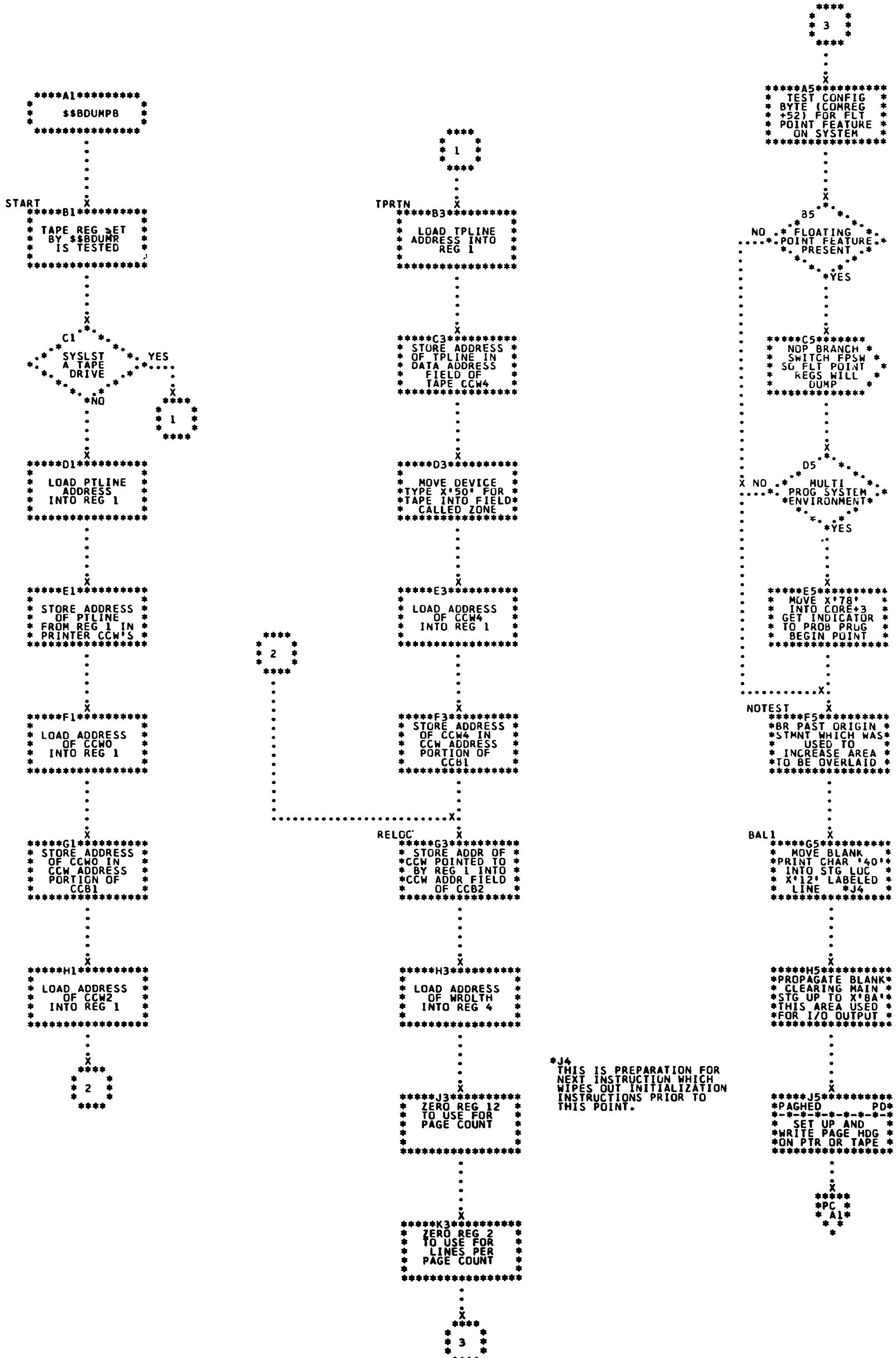
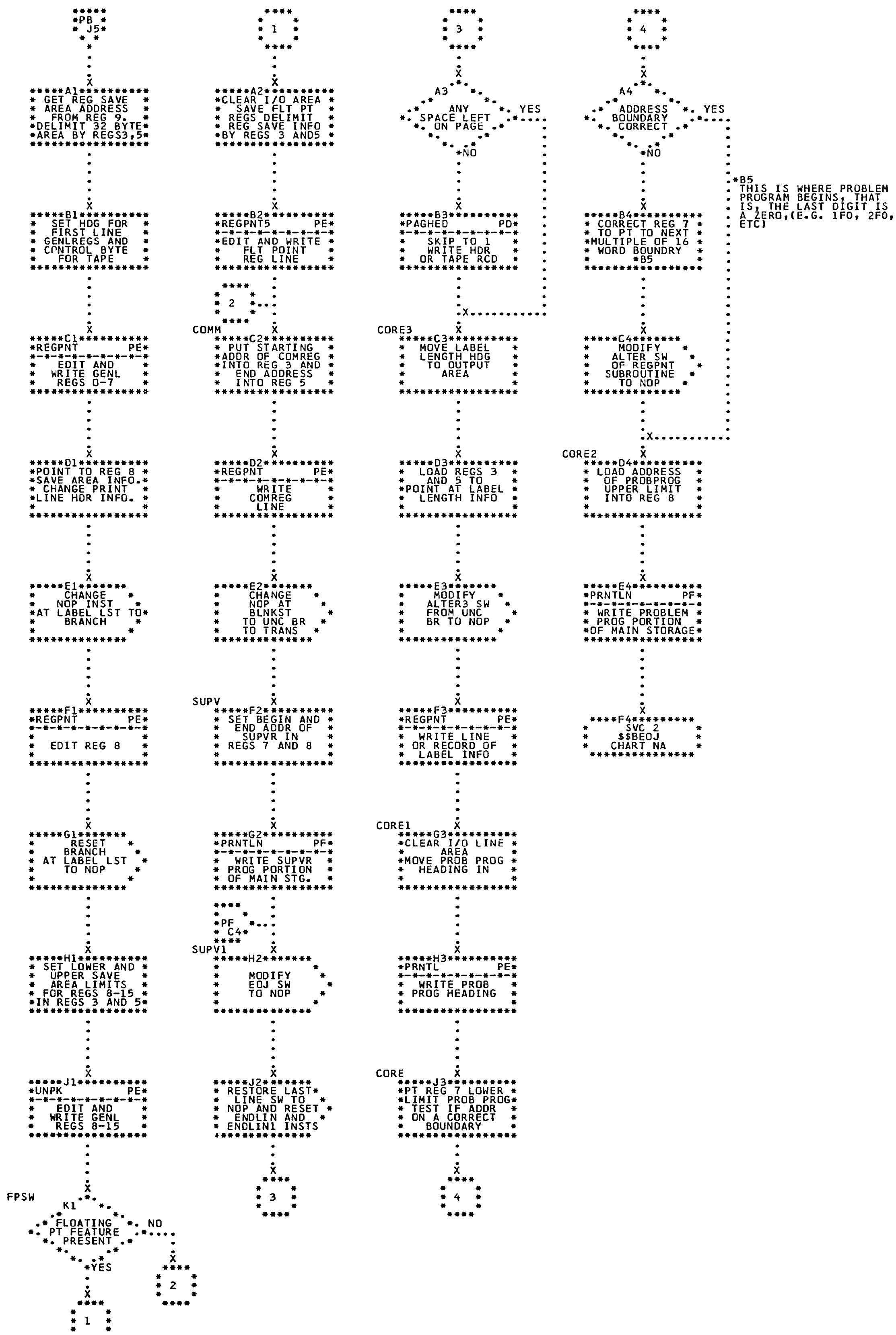


Chart PC. BG Dump on Printer or Tape $$BDUMPB; Refer to Supervisor, Chart 29



## Chart PD. Prepare Page Headings and PIOCS Subroutines $$BDUMPB; Refer to Supervisor, Chart 29

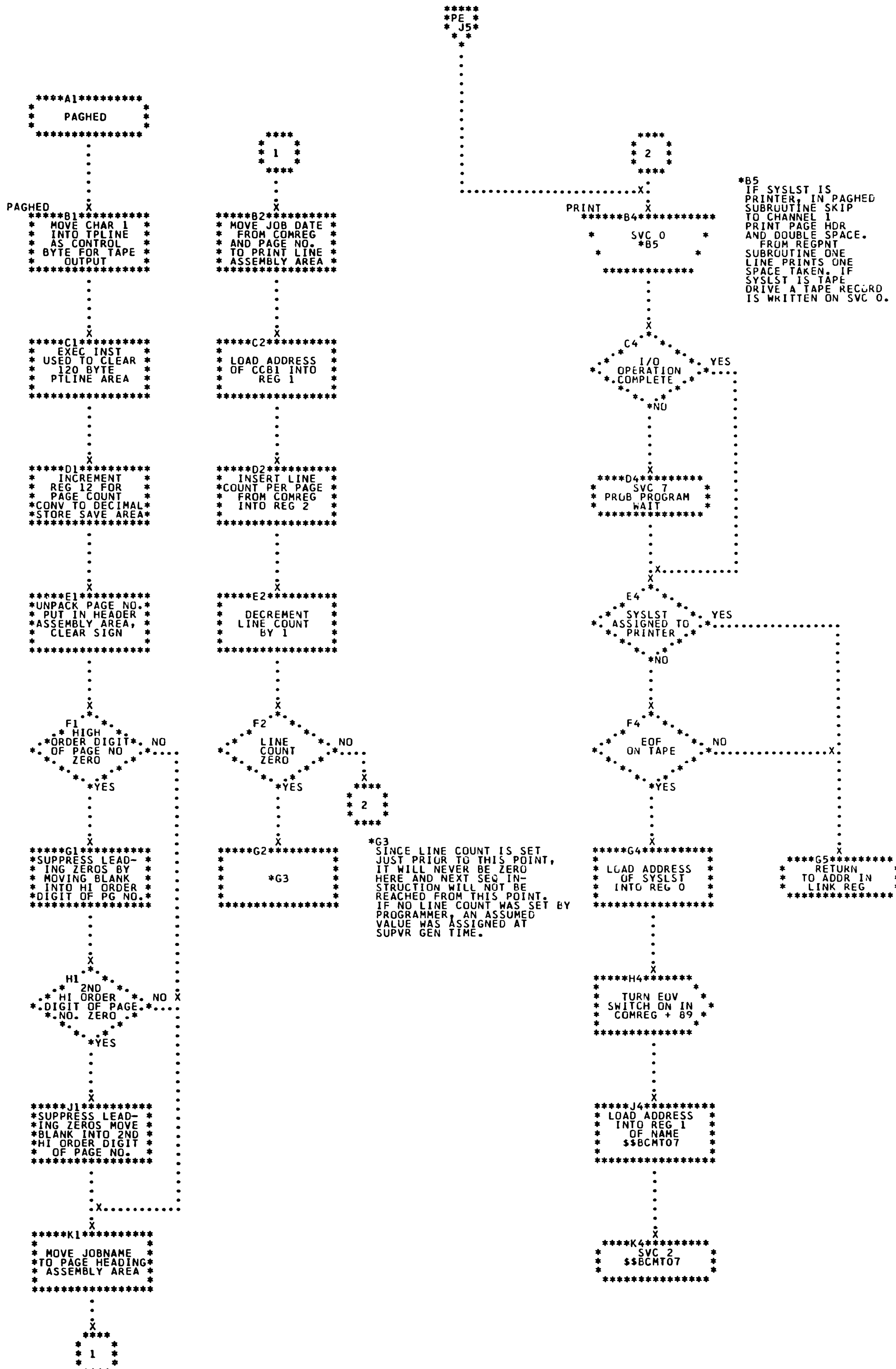


Chart PE. Prepare and Edit a Line Subroutine $$BDUMPB; Refer to Supervisor, Chart 29

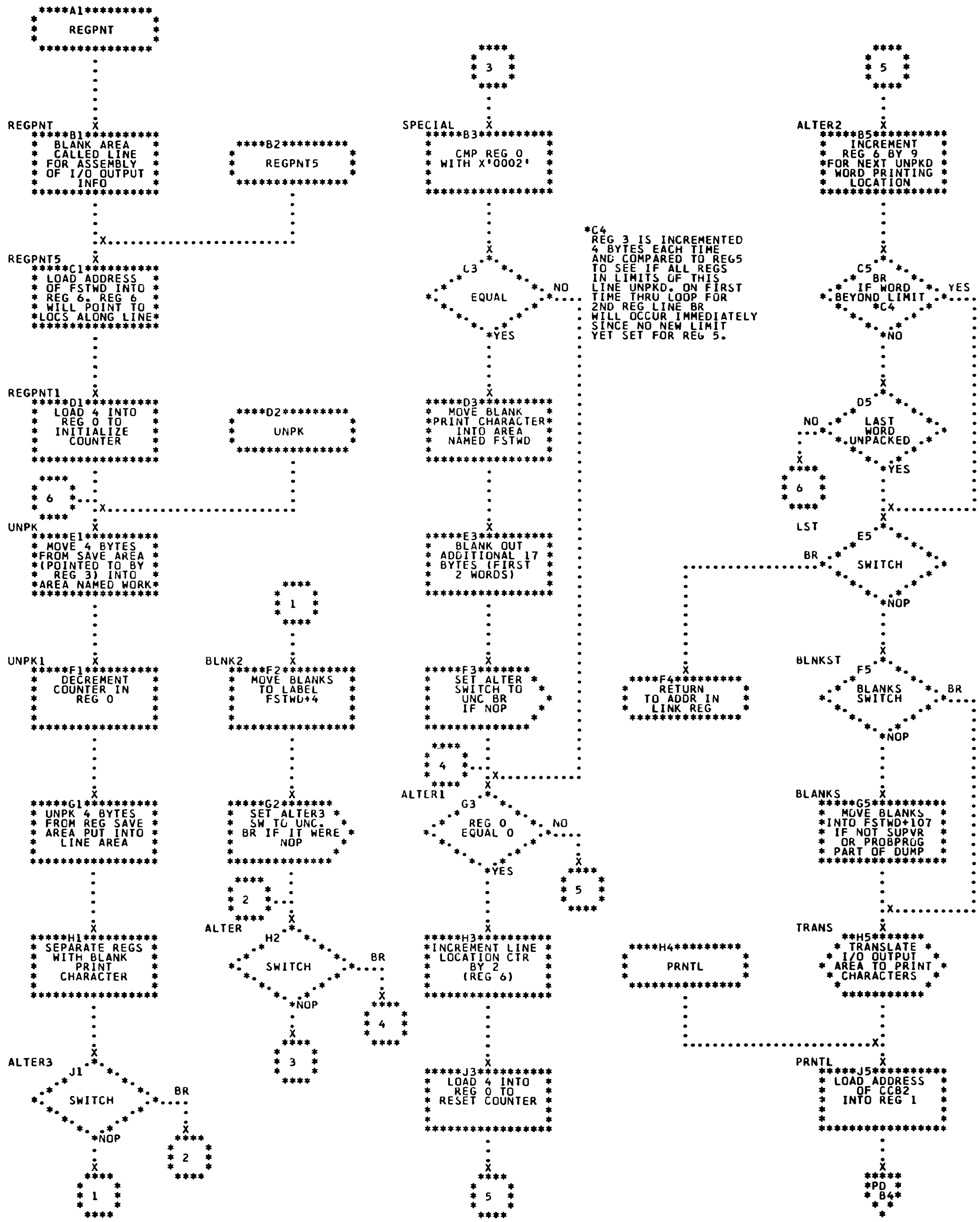


Chart PF. Line Test Subroutines $$BDUMPB; Refer to Supervisor, Chart 29

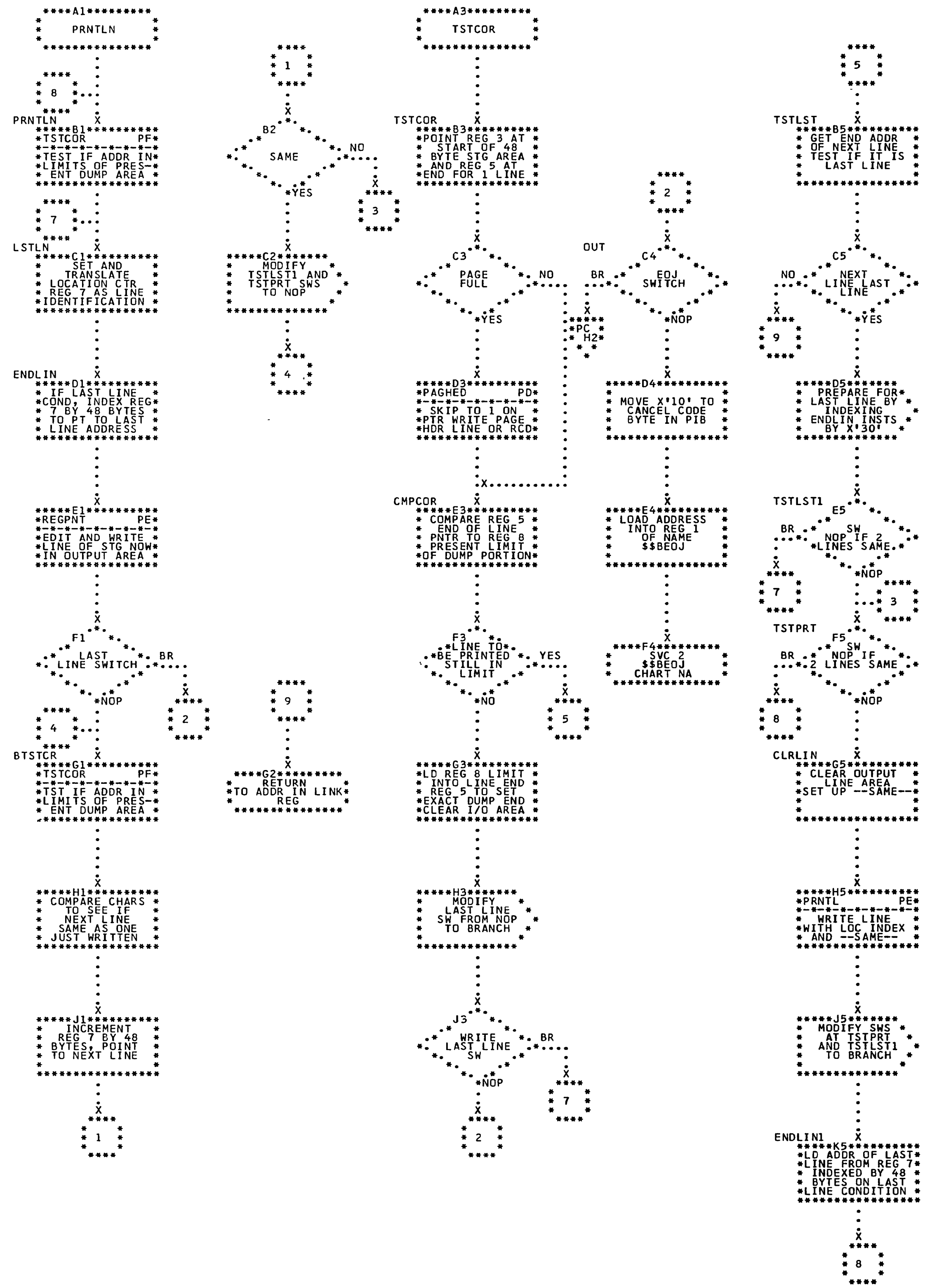


Chart PG. BG Dump on Disk Device $$BDUMPD; Refer to Supervisor, Chart 29

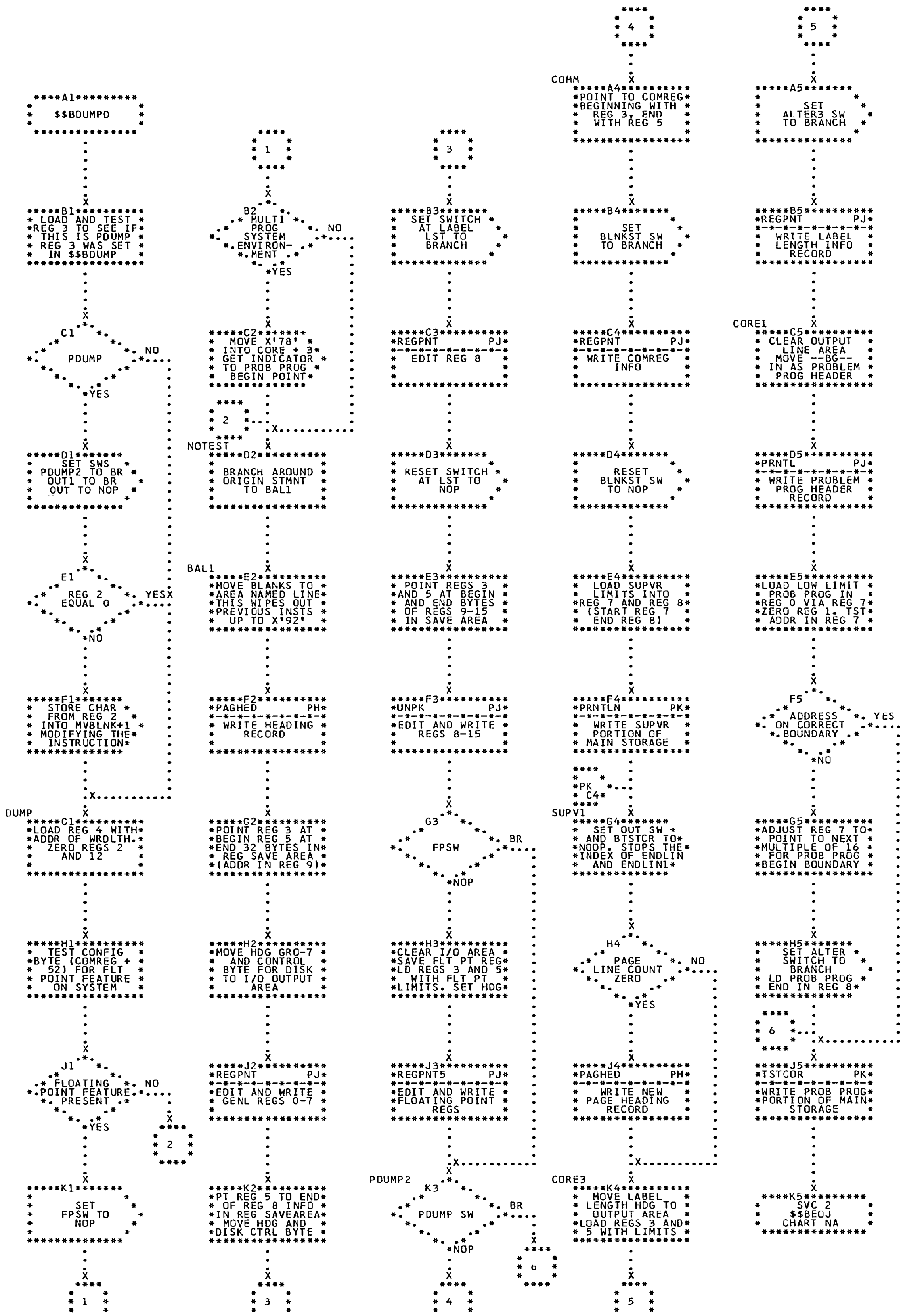


Chart PH. Prepare Page Headings and PIOCS Subroutines
$$BDUMPD; Refer to Supervisor, Chart 29

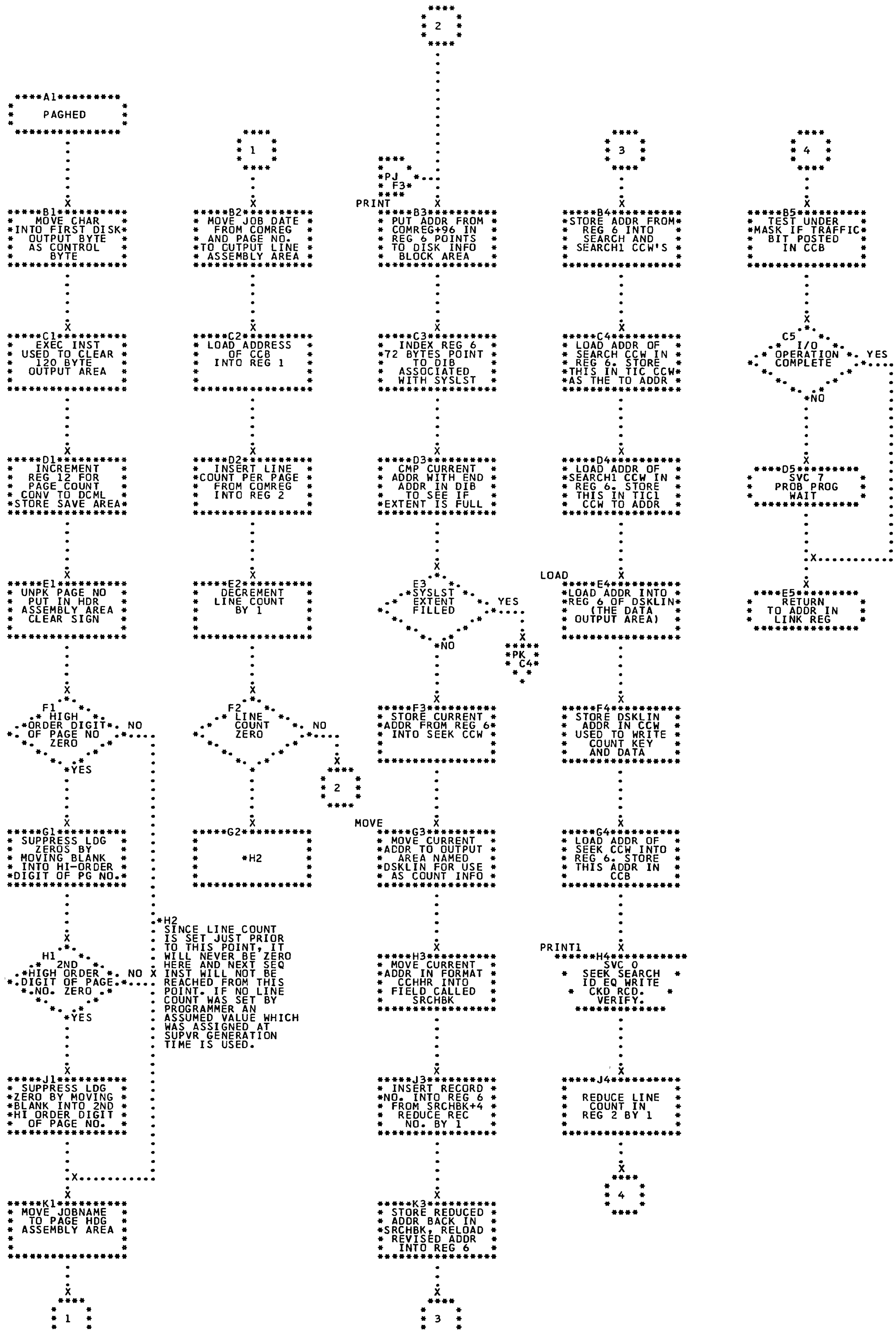


Chart PJ. Prepare and Edit a Line Subroutine $$BDUMPD; Refer to Supervisor, Chart 29

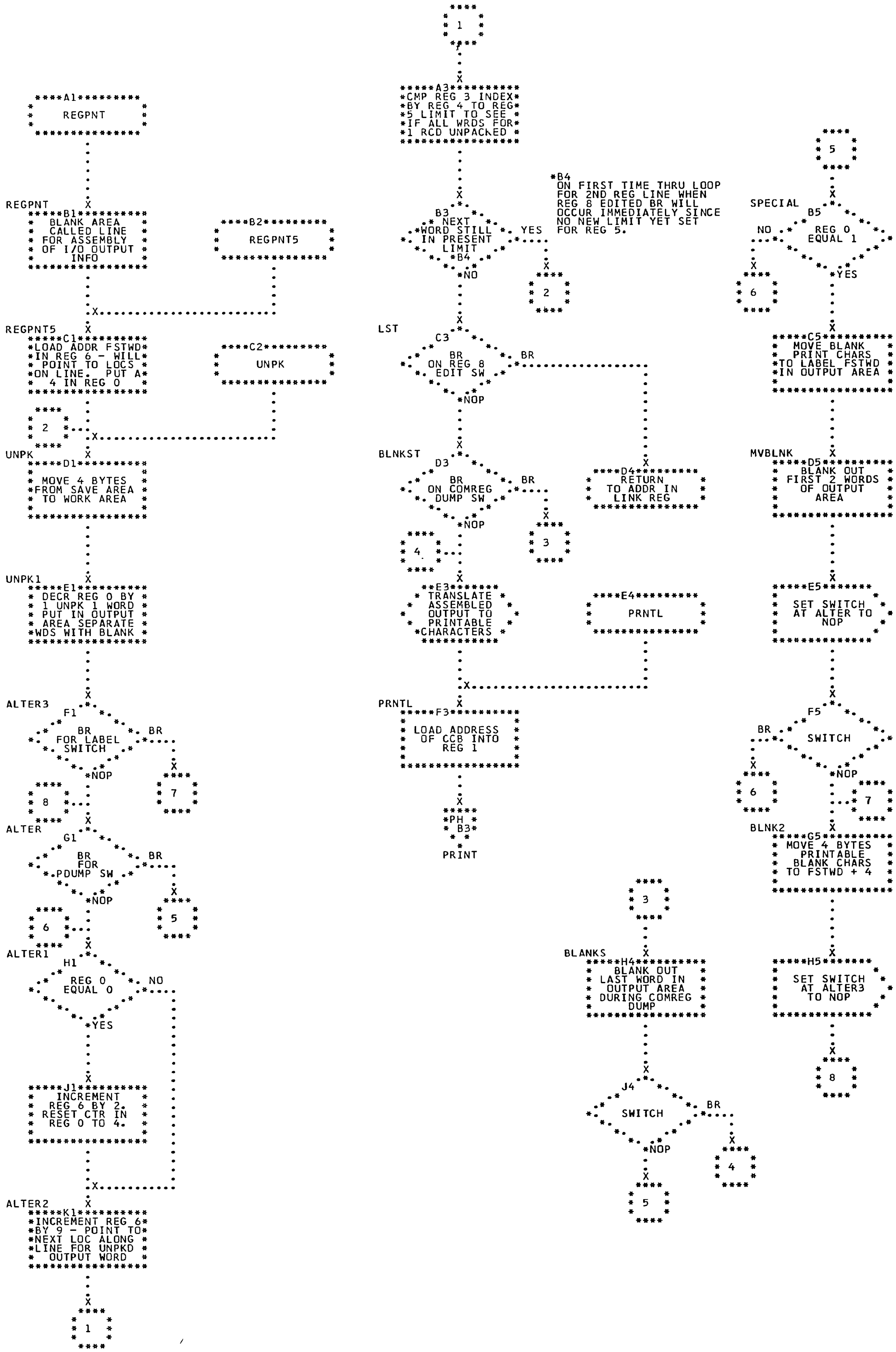


Chart PK. Line Test Subroutines $$BDUMPD; Refer to Supervisor, Chart 29

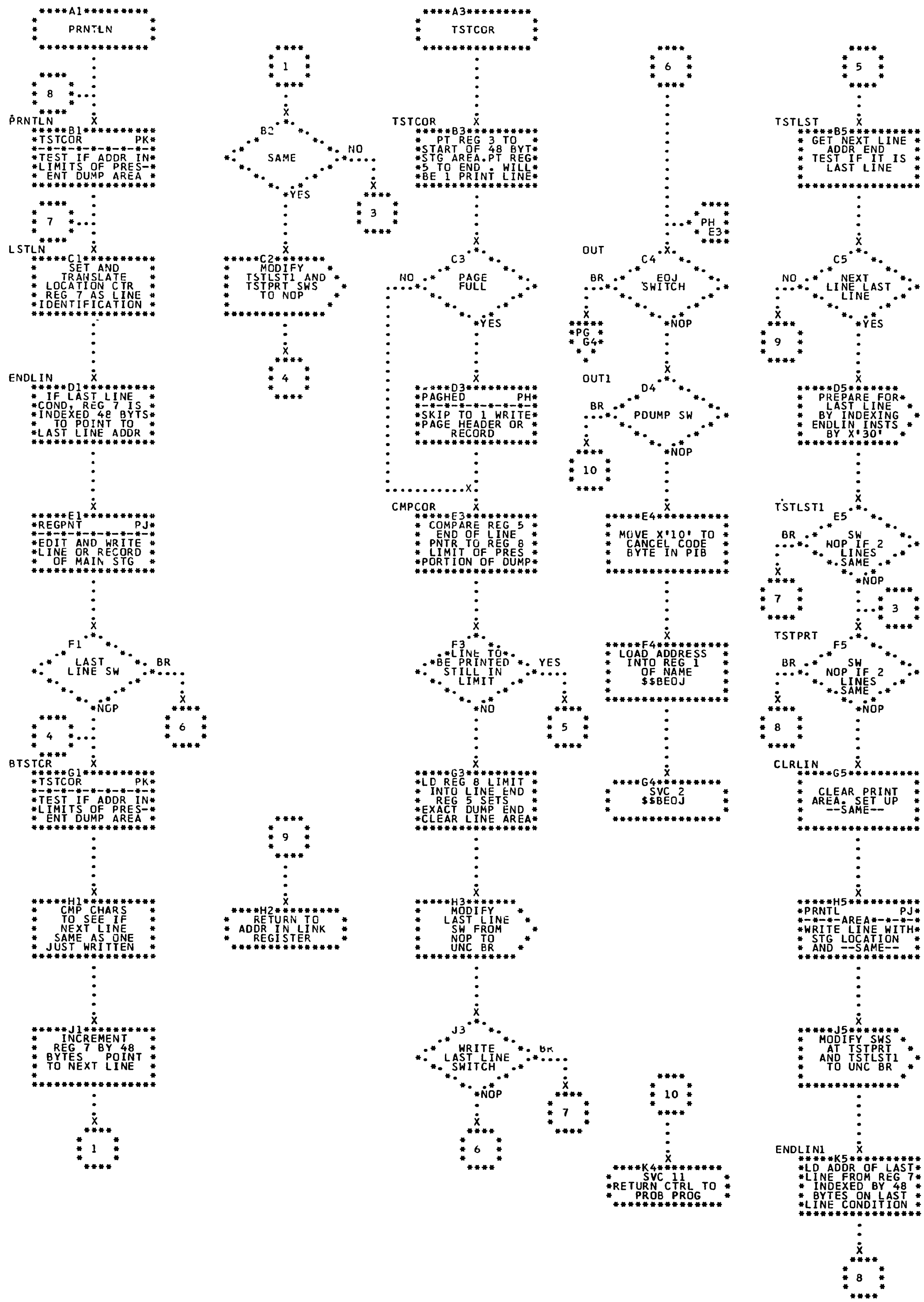


Chart PL. Parameter Storage Dump Monitor $$BPDUMP; Refer to Supervisor, Chart 30

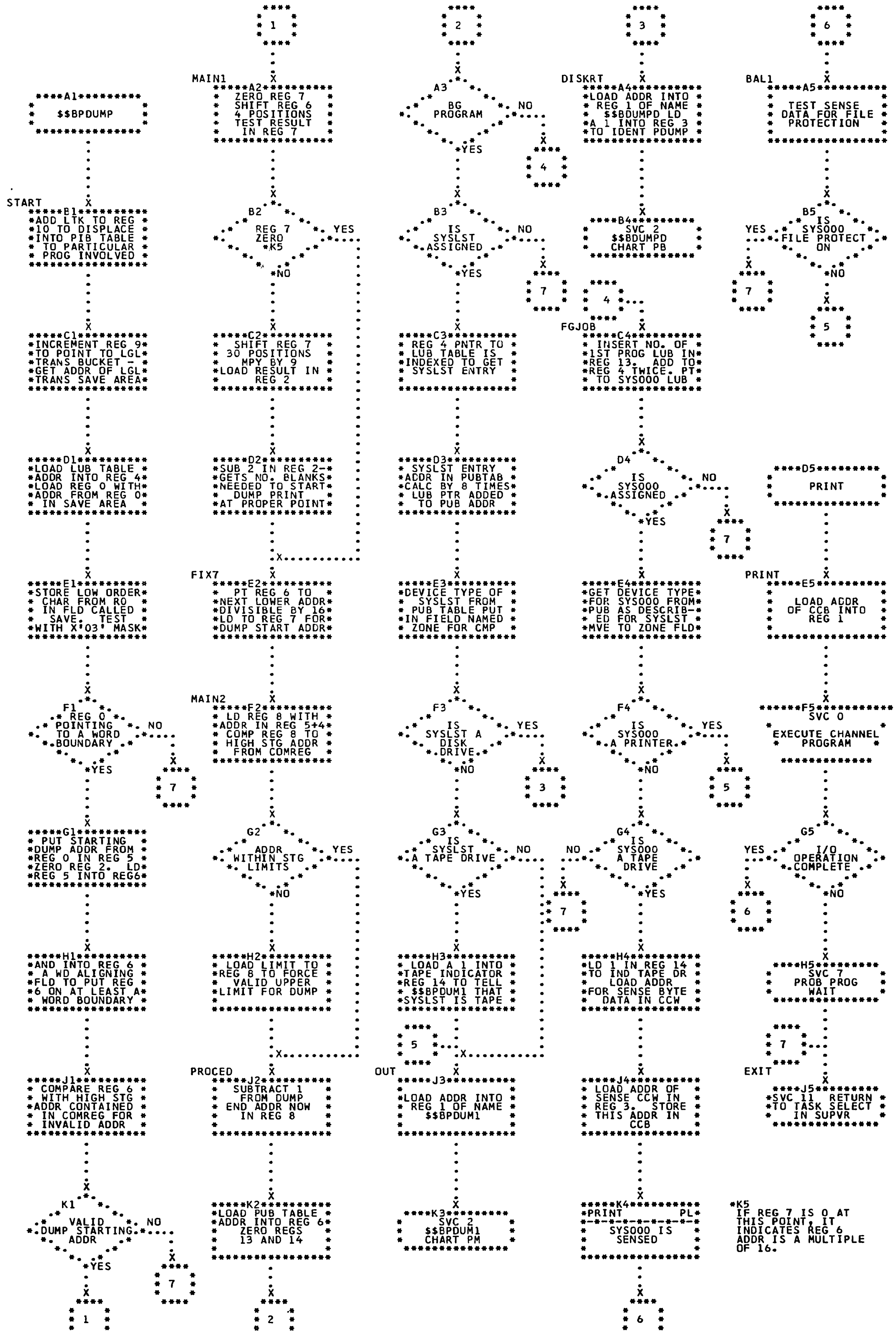


Chart PM. Initialize Parameter Dump or Printer or Tape
$$BPDUM1; Refer to Supervisor, Chart 30

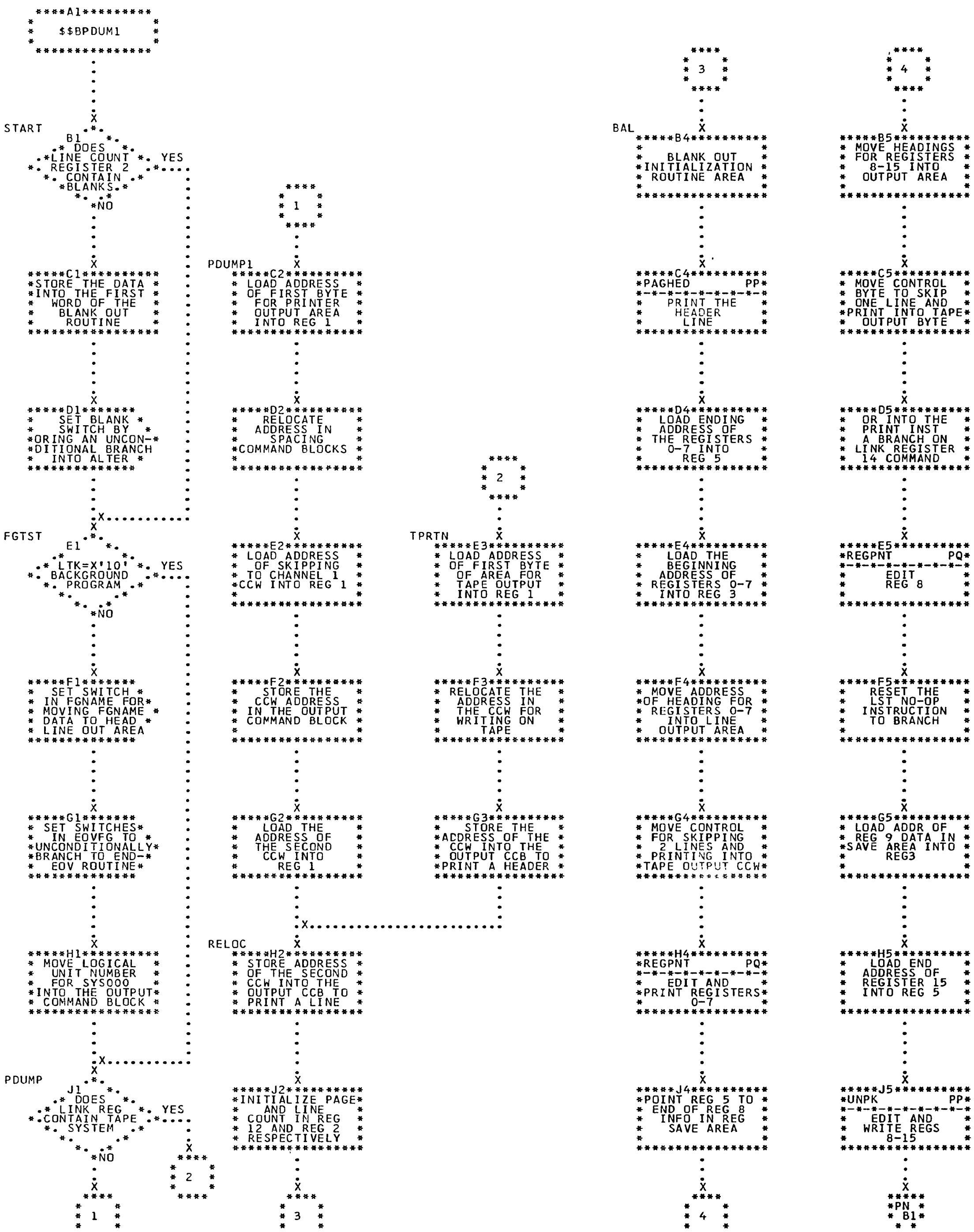


Chart PN. Parameter Storage Dump on Printer or Tape
$$BPDUM1; Refer to Supervisor, Chart 30

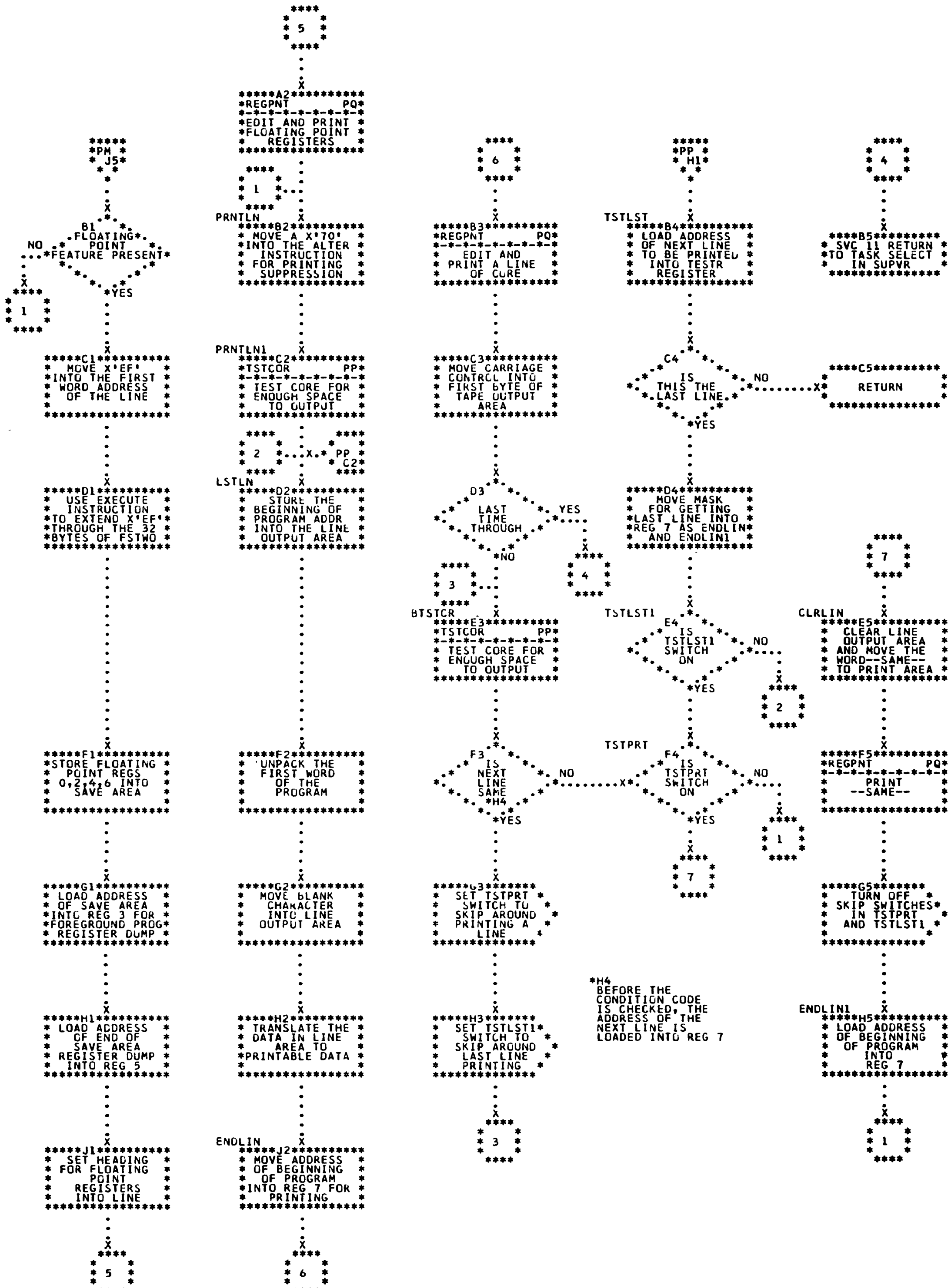


Chart PP. Line Test Subroutines $$BPDUM1; Refer to Supervisor, Chart 30

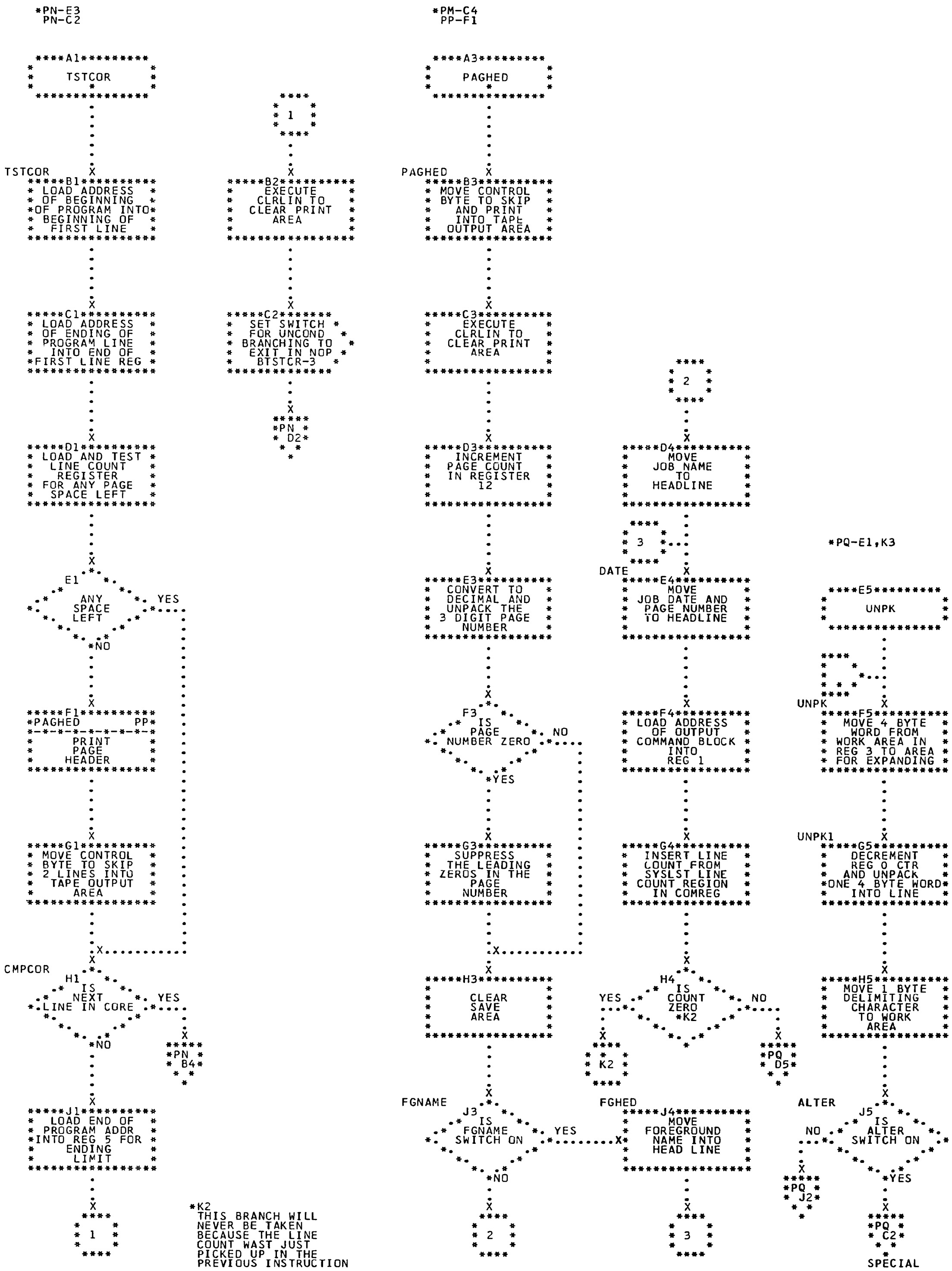


Chart PQ. Prepare and Edit a Line Subroutine $$BPDUM1;
Refer to Supervisor, Chart 30

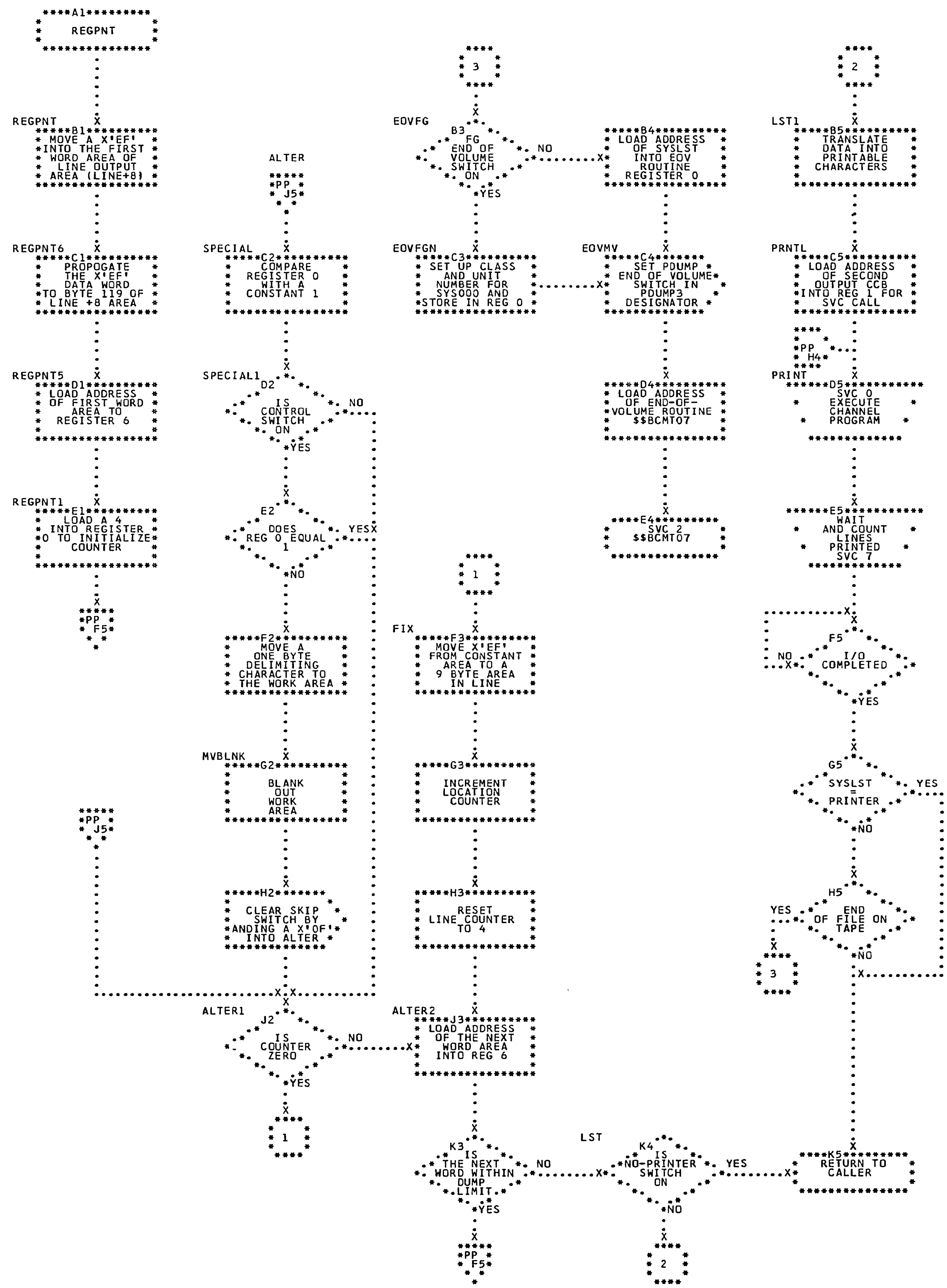


Chart PS. Set Up a Write on SYSRES Operation $$BYSYSWR

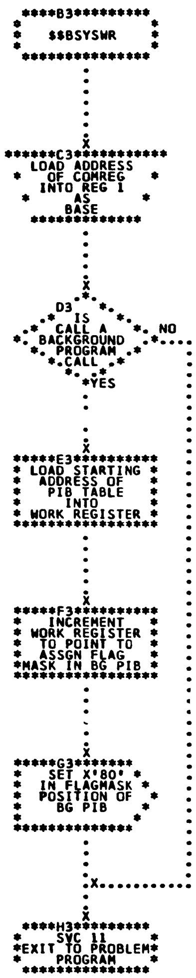


Chart QA. Initialization, Part 2 $LNKEDT; Refer to Linkage Editor, Chart 31

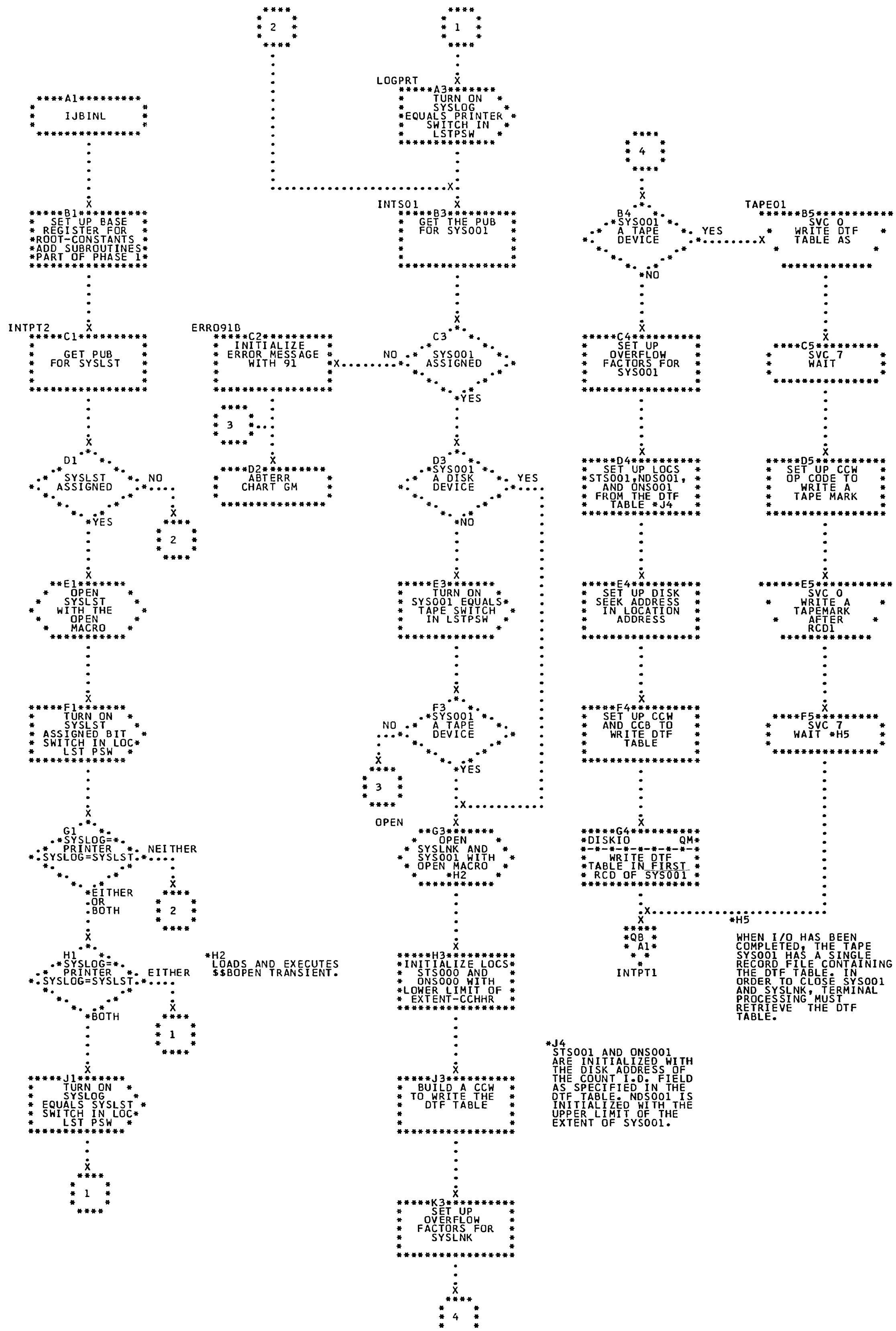


Chart QB. Initialization, Part 1 (Part 1 of 2) $LNKEDT;
Refer to Linkage Editor, Chart 31

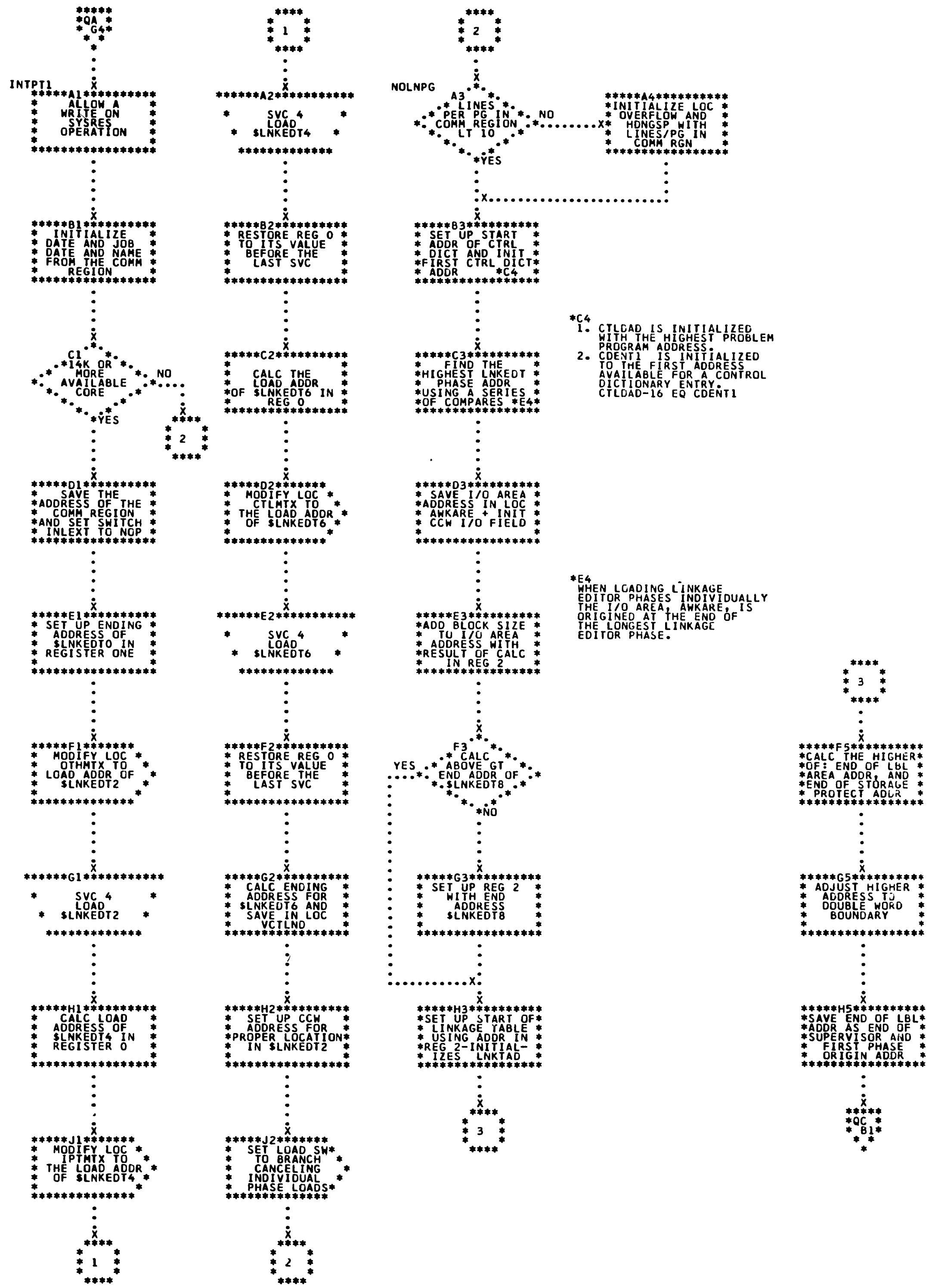


Chart QC. Initialization, Part 1 (Part 2 of 2) $LNKEDT;
Refer to Linkage Editor, Chart 31

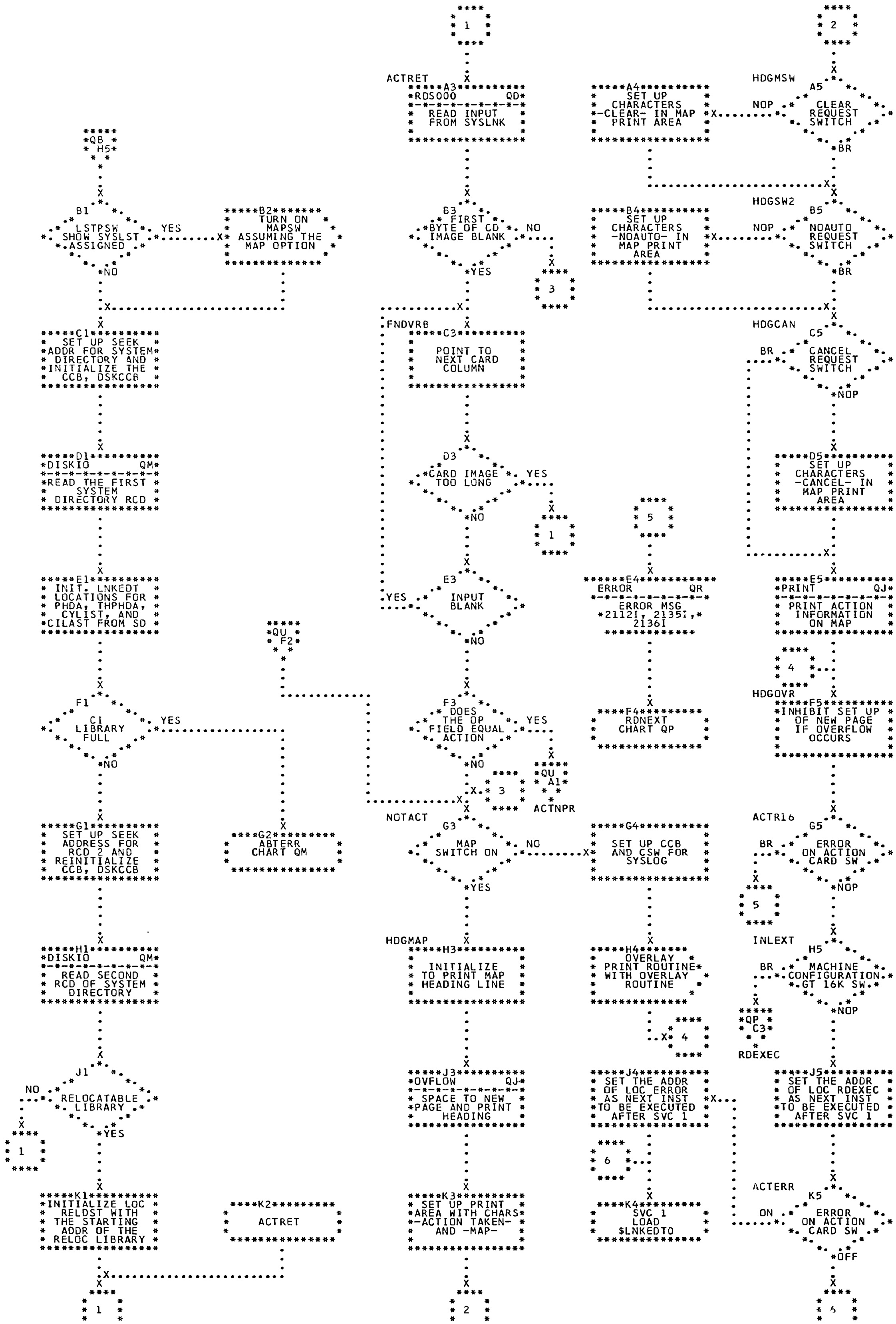


Chart QD. Read SYSLNK Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

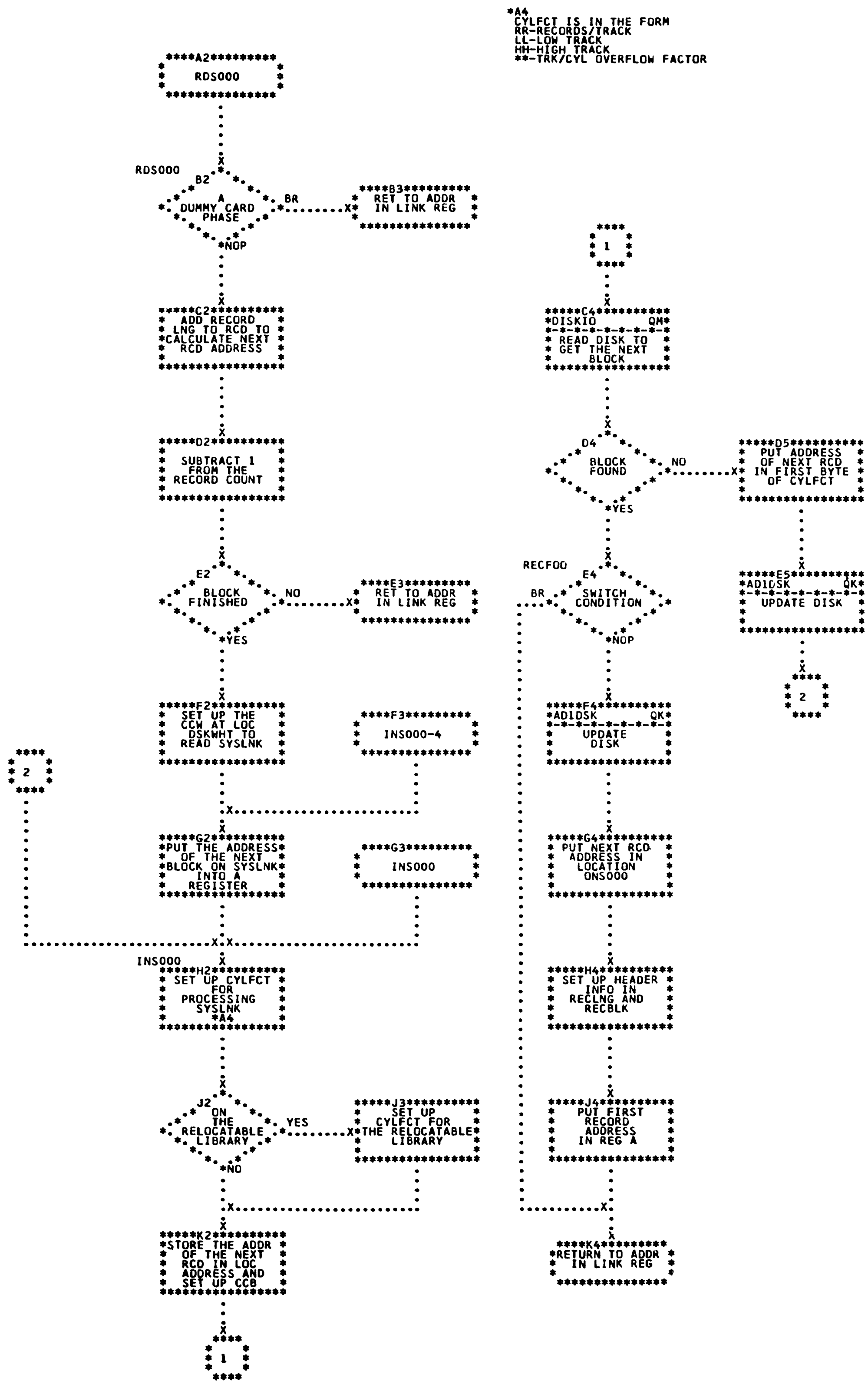


Chart QE. Control Dictionary Search Subroutine $LNKEDT;
Refer to Linkage Editor, Chart 31

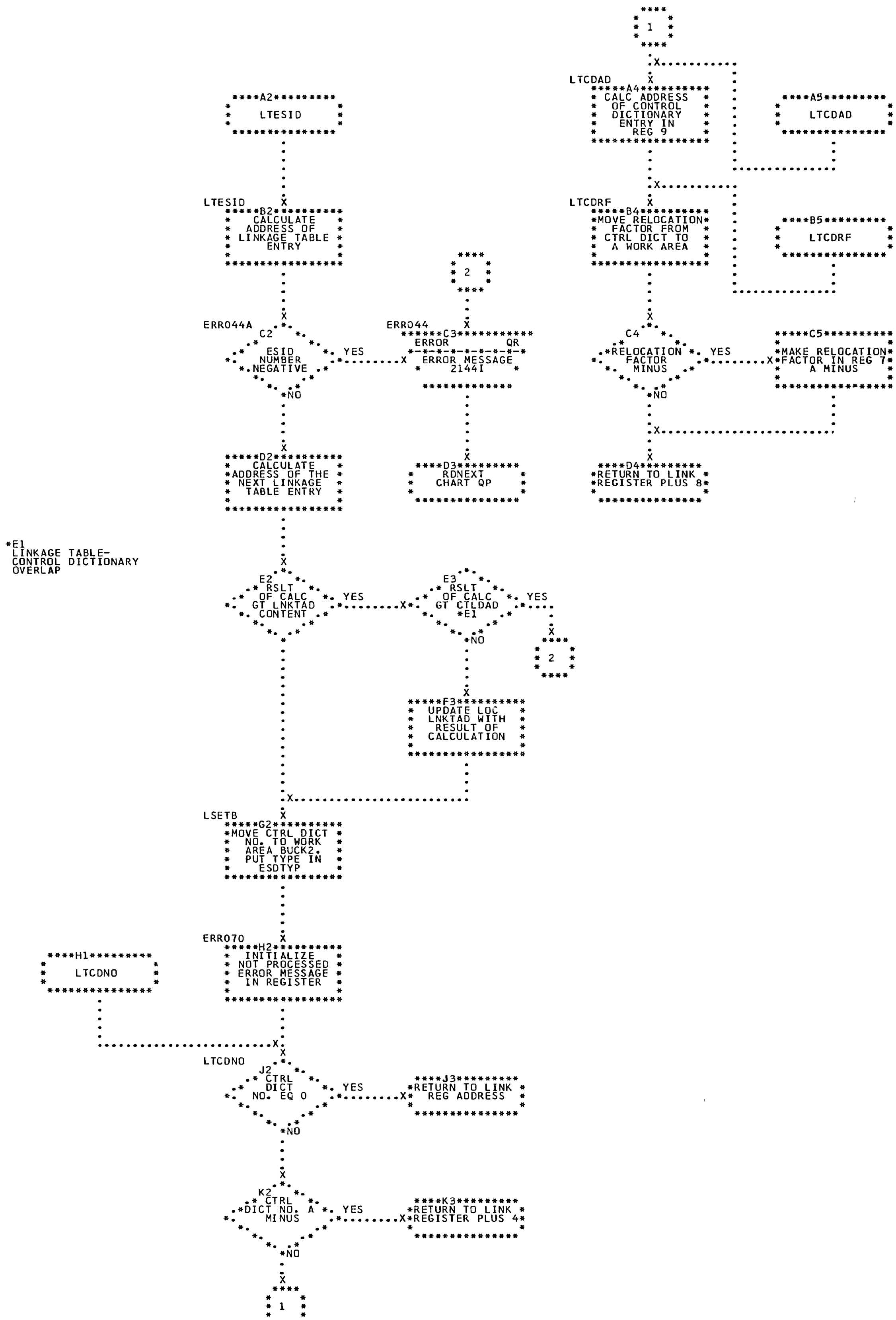


Chart QF. Label Search Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

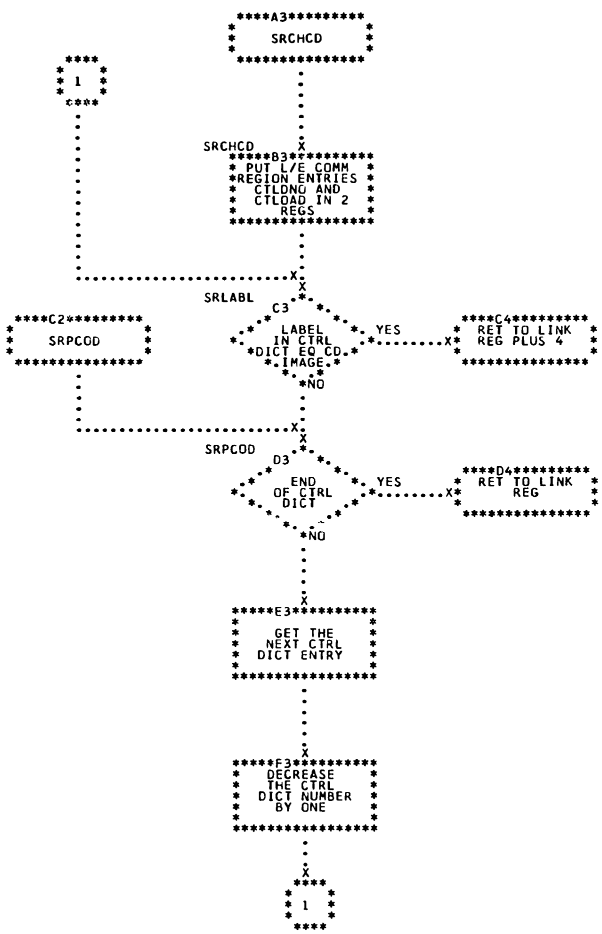


Chart QH. Convert to Binary Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

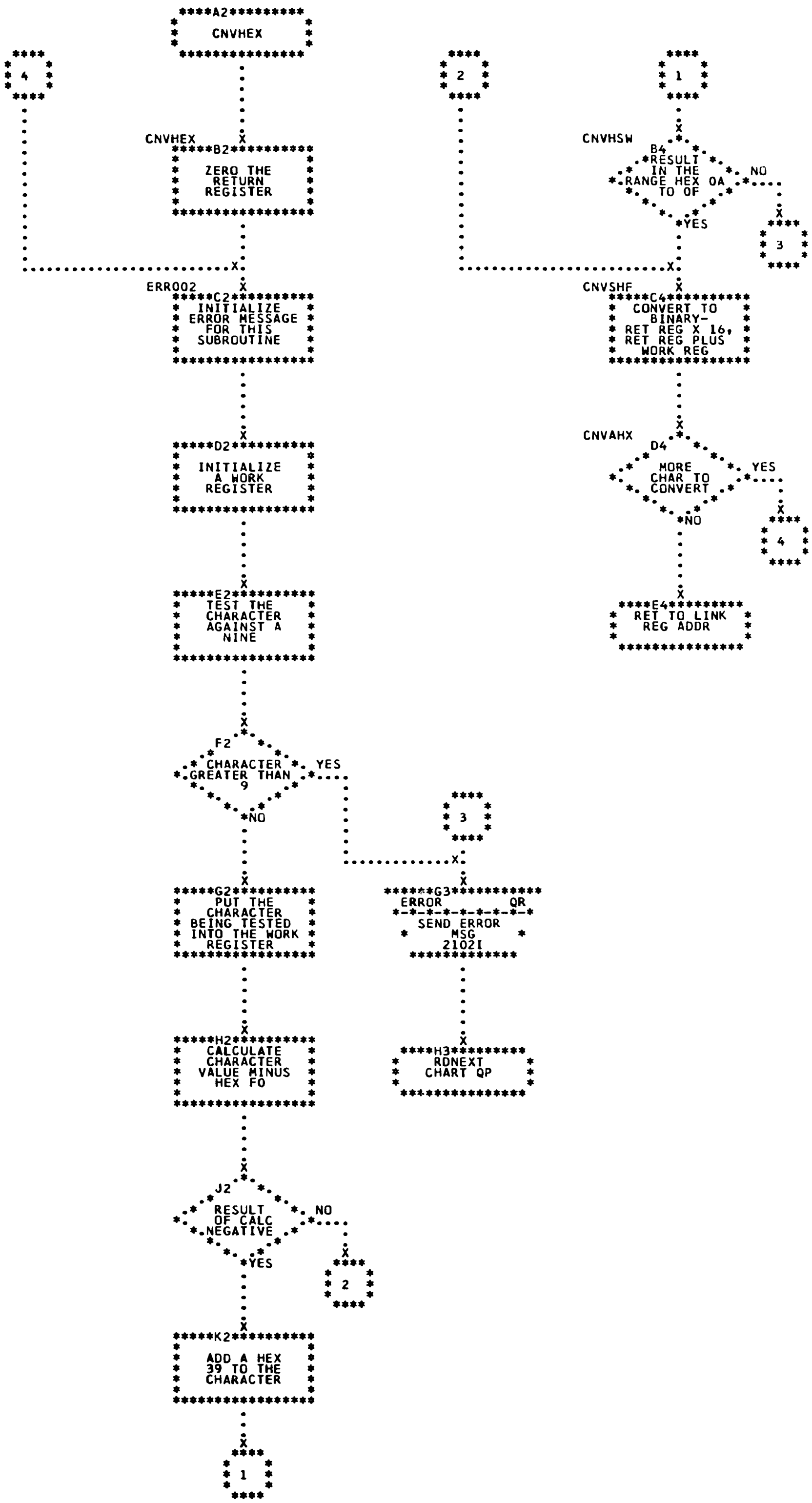


Chart QJ. Print/Carriage Control Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

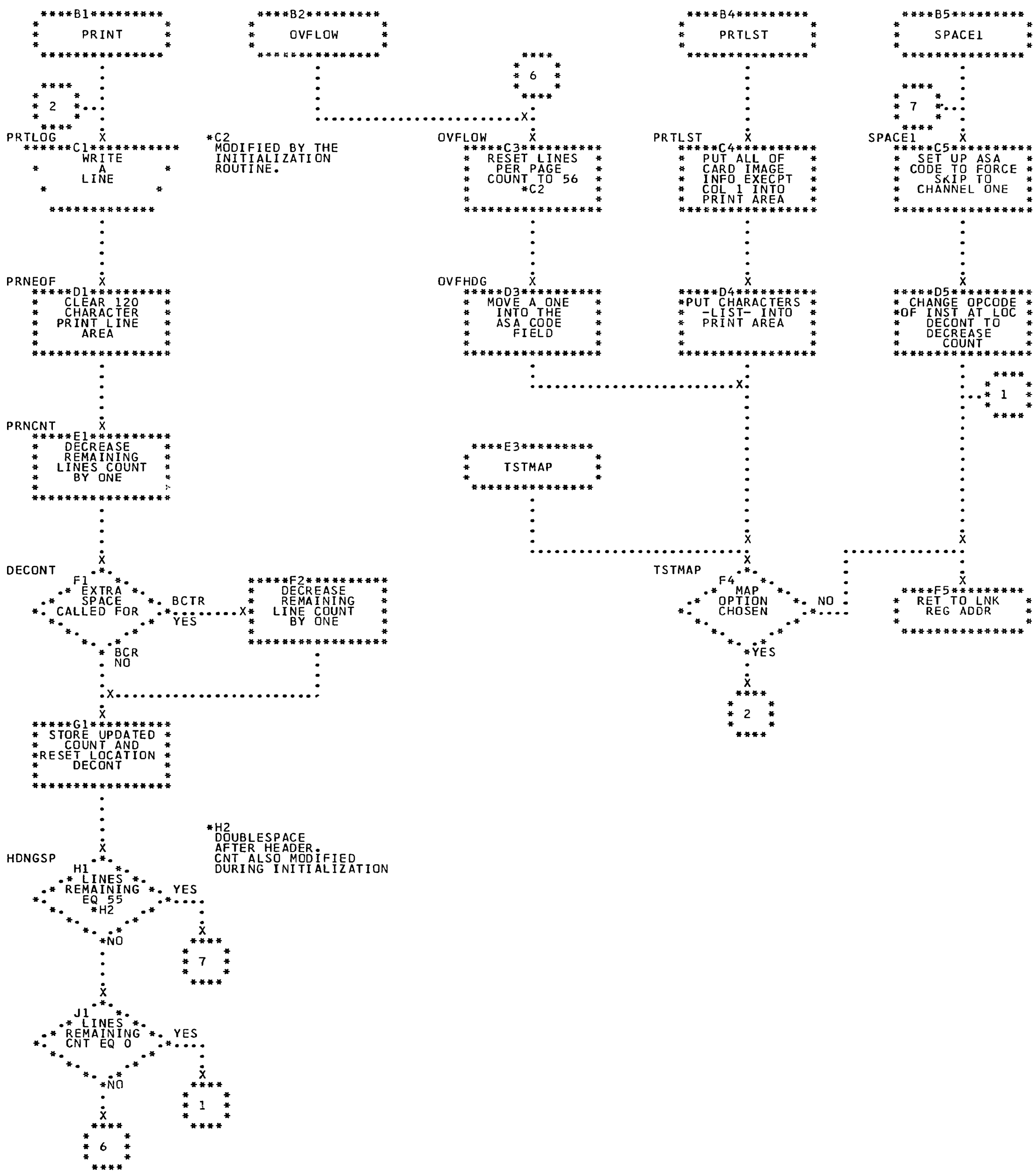


Chart QK. Update Disk Address Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

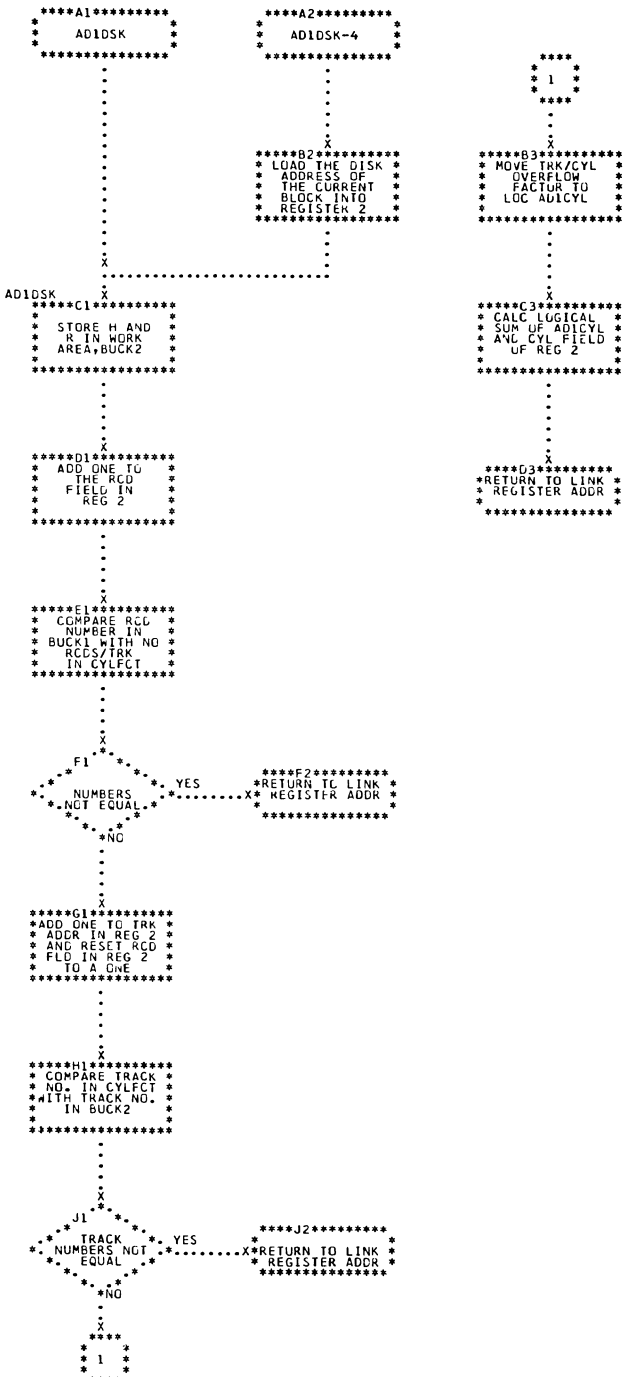


Chart QL. Extract Phase Number Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

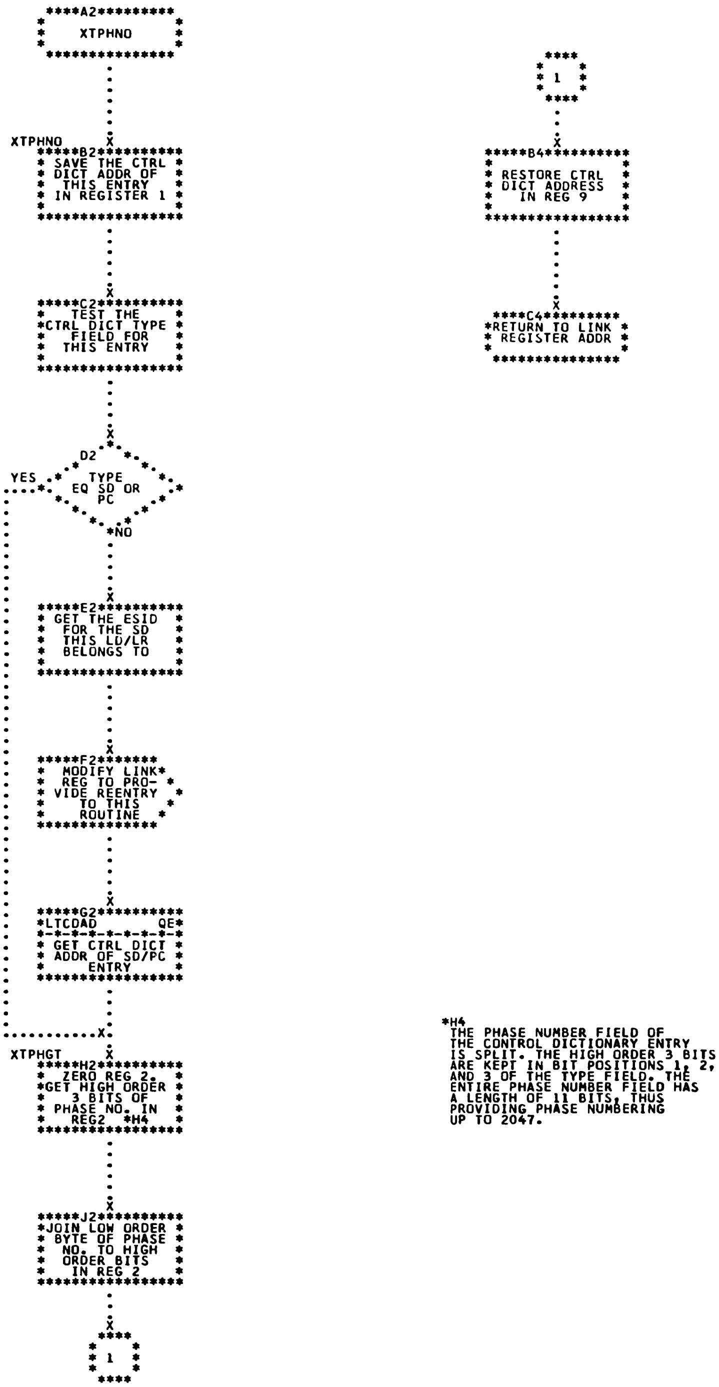


Chart QM. Read/Write Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

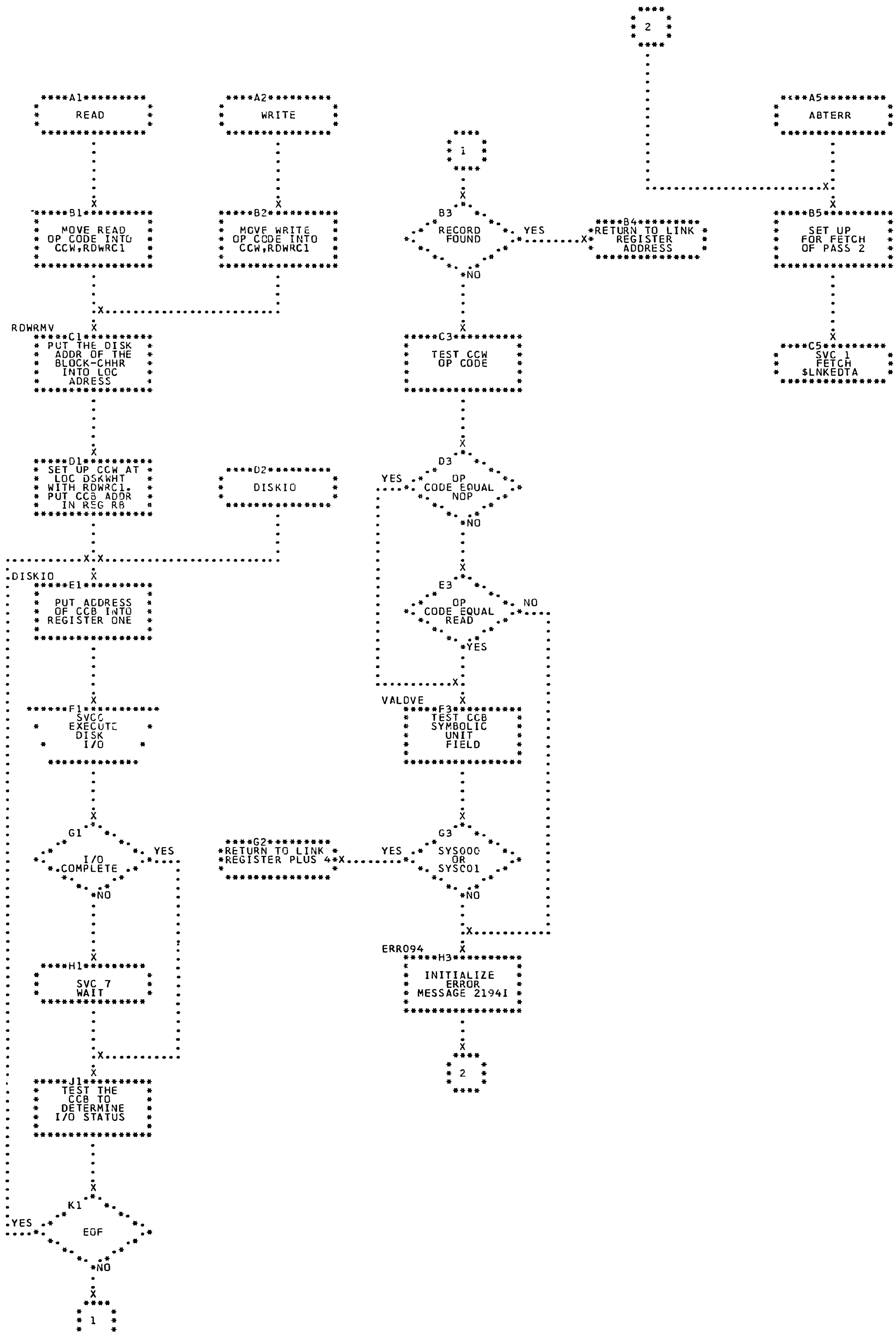


Chart QN. Overflow Test Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

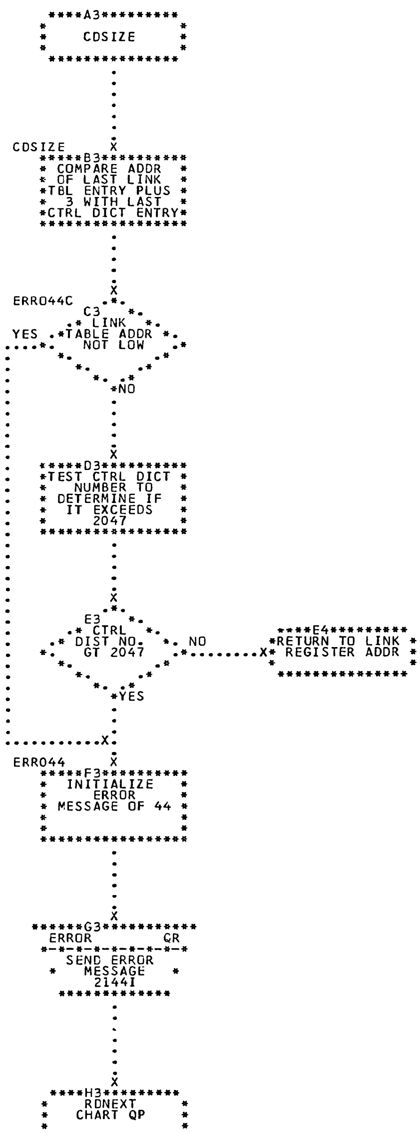


Chart QP. Read Input Stream $LNKEDT; Refer to Linkage Editor, Chart 31

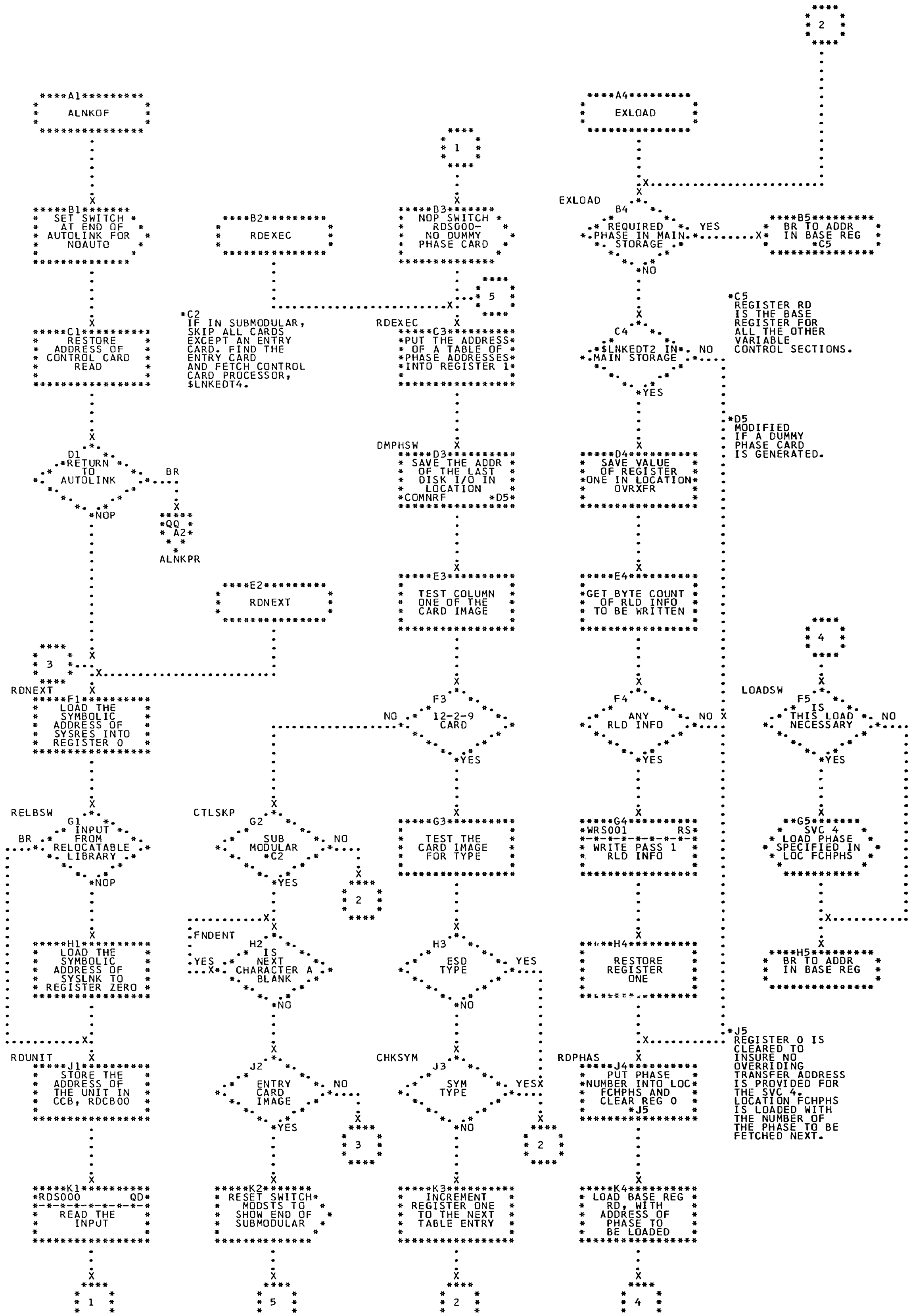


Chart QQ. Autolink Processing Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

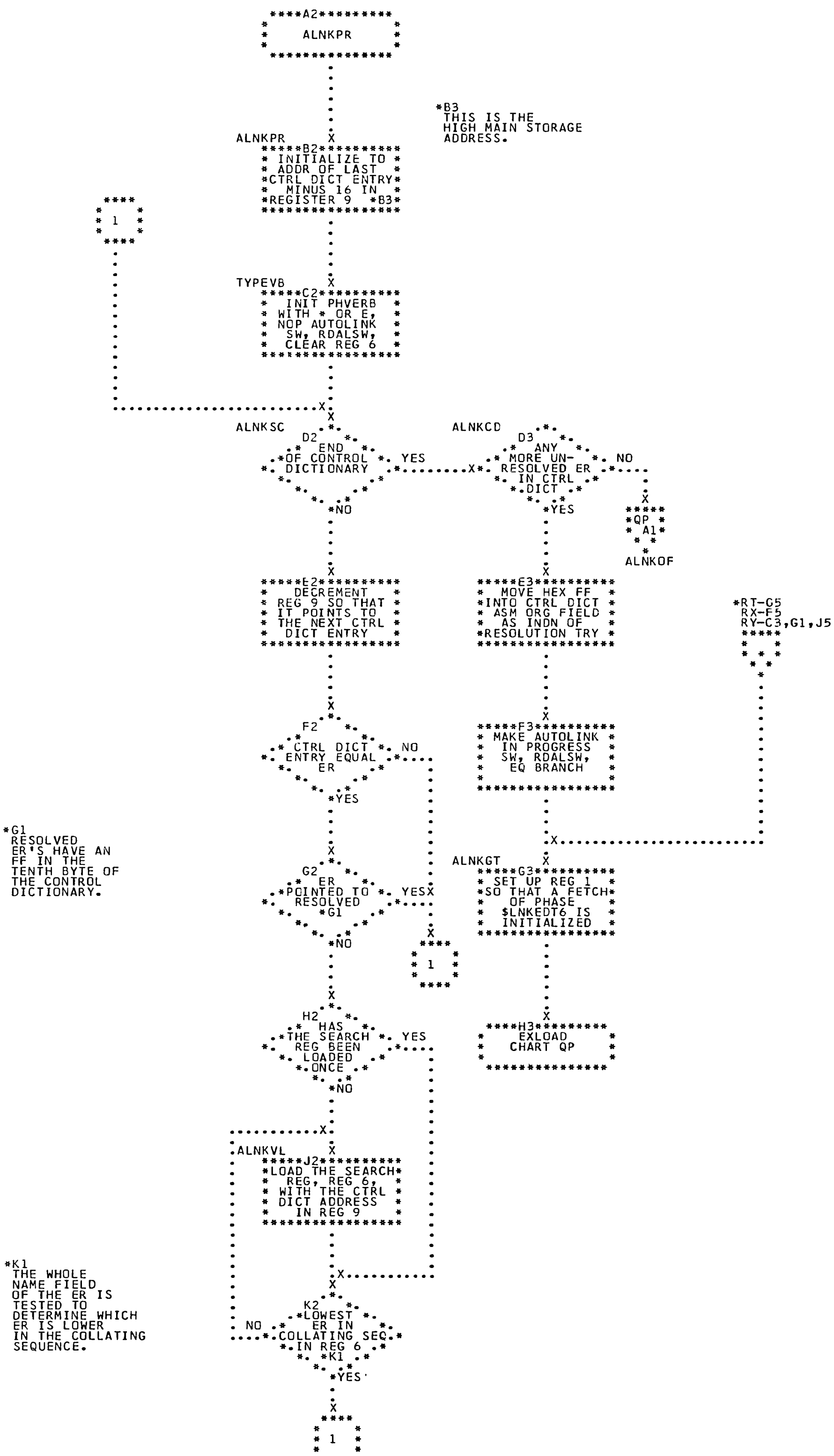


Chart QR. Non-Abort Error Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

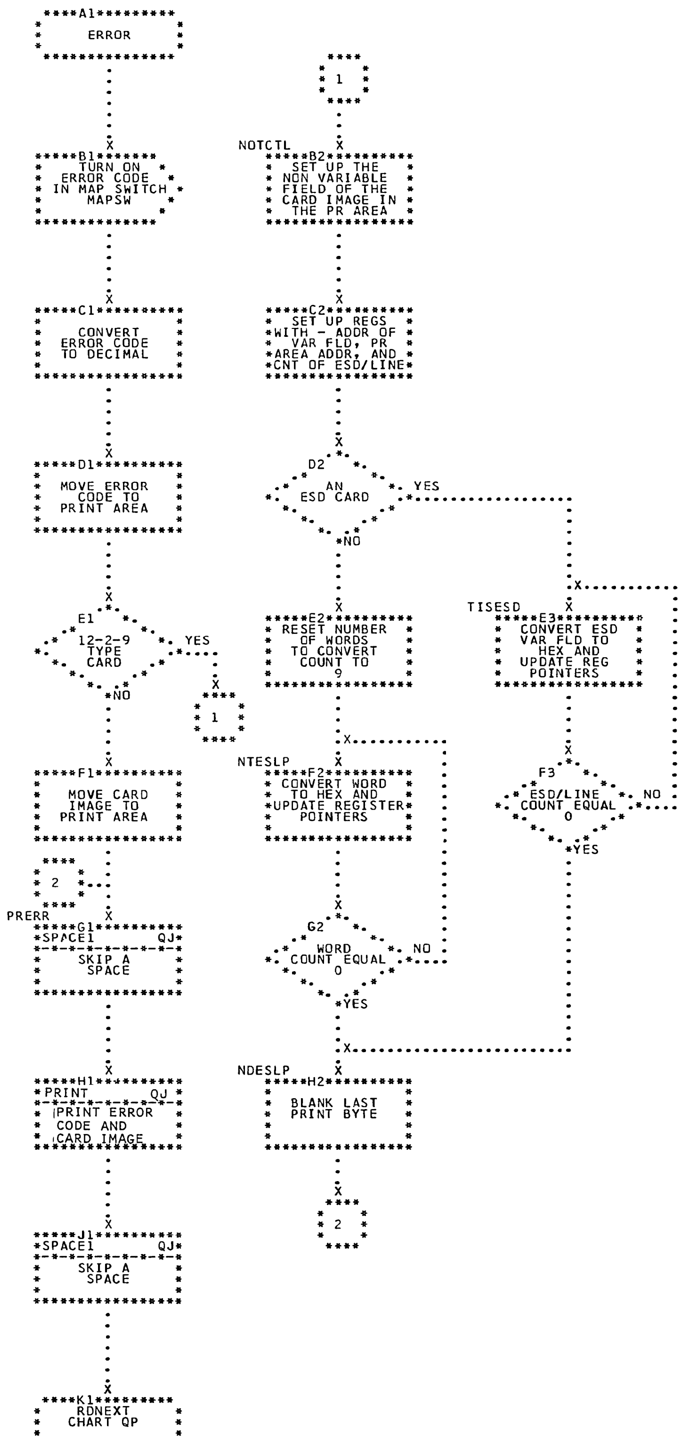
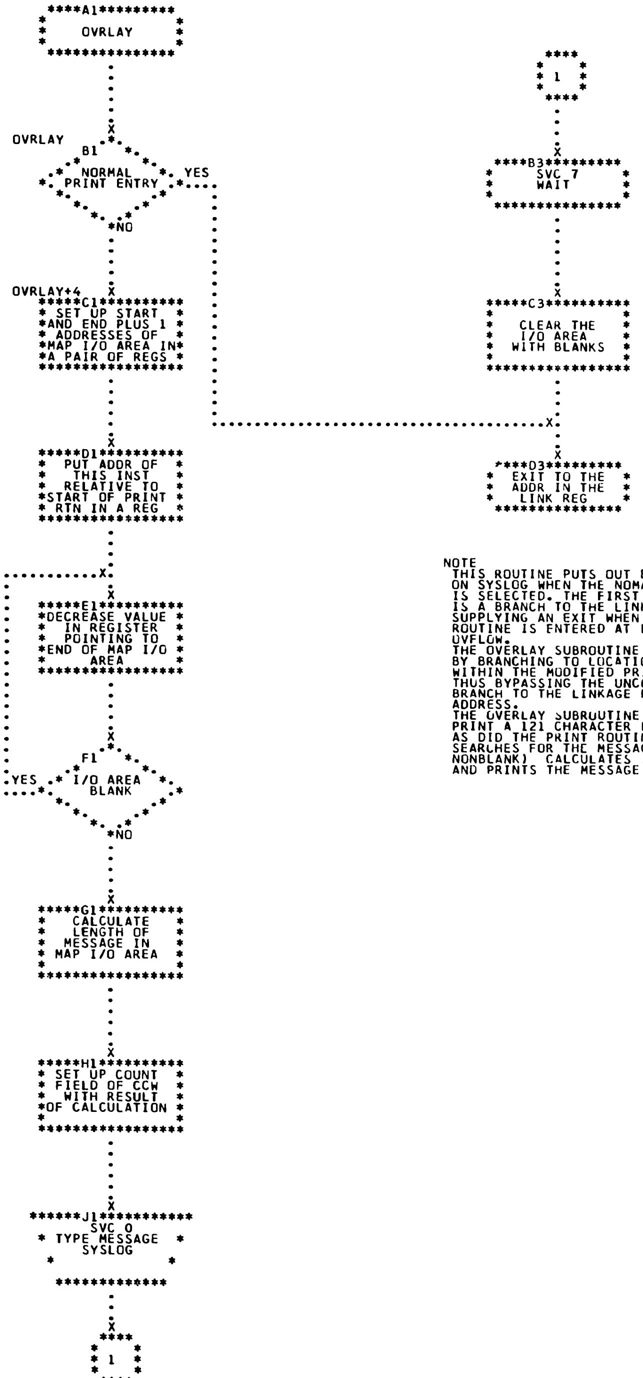


Chart QS. Overlay Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31




NOTE
THIS ROUTINE PUTS OUT ERROR MESSAGES ON SYSLOG WHEN THE NOMAP OPTION IS SELECTED. THE FIRST INSTRUCTION IS A BRANCH TO THE LINK REGISTER ADDRESS, SUPPLYING AN EXIT WHEN THE MODIFIED PRINT ROUTINE IS ENTERED AT LOCATION OVFLOW.
THE OVERLAY SUBROUTINE IS ENTERED BY BRANCHING TO LOCATION PRTLOG WITHIN THE MODIFIED PRINT ROUTINE, THUS BYPASSING THE UNCONDITIONAL BRANCH TO THE LINKAGE REGISTER ADDRESS.
THE OVERLAY SUBROUTINE DOES NOT PRINT A 121 CHARACTER LENGTH MESSAGE AS DID THE PRINT ROUTINE. IT SEARCHES FOR THE MESSAGE (A NONBLANK) CALCULATES ITS LENGTH, AND PRINTS THE MESSAGE ON SYSLOG.

Chart QT. Core Image Block Building Subroutine $LNKEDT; Refer to Linkage Editor, Chart 31

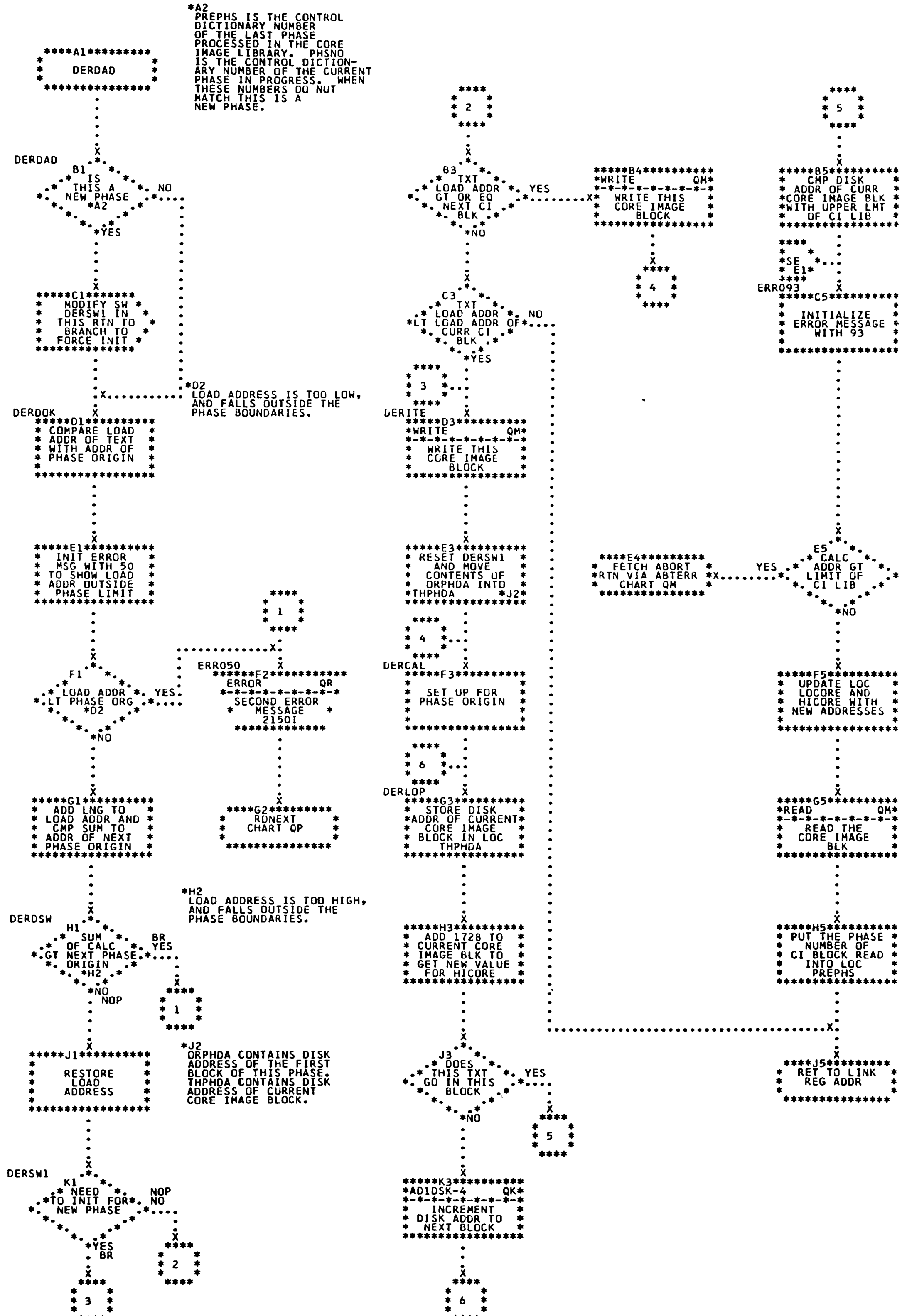


Chart QU. Action Processor $LNKEDT; Refer to Linkage Editor, Chart 31

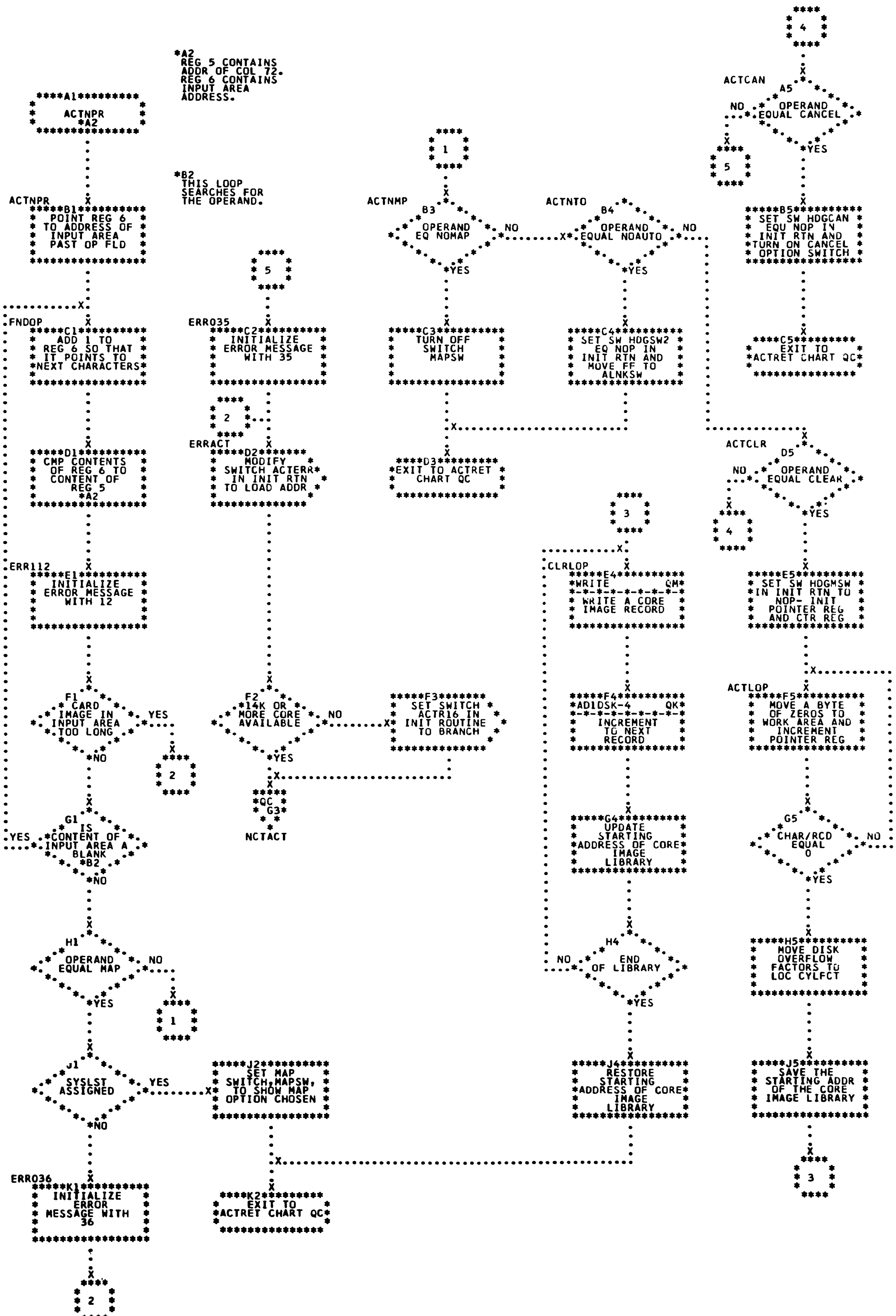


Chart RA. Initialize ESD Processor $LNKEDT0; Refer to Linkage Editor, Chart 32

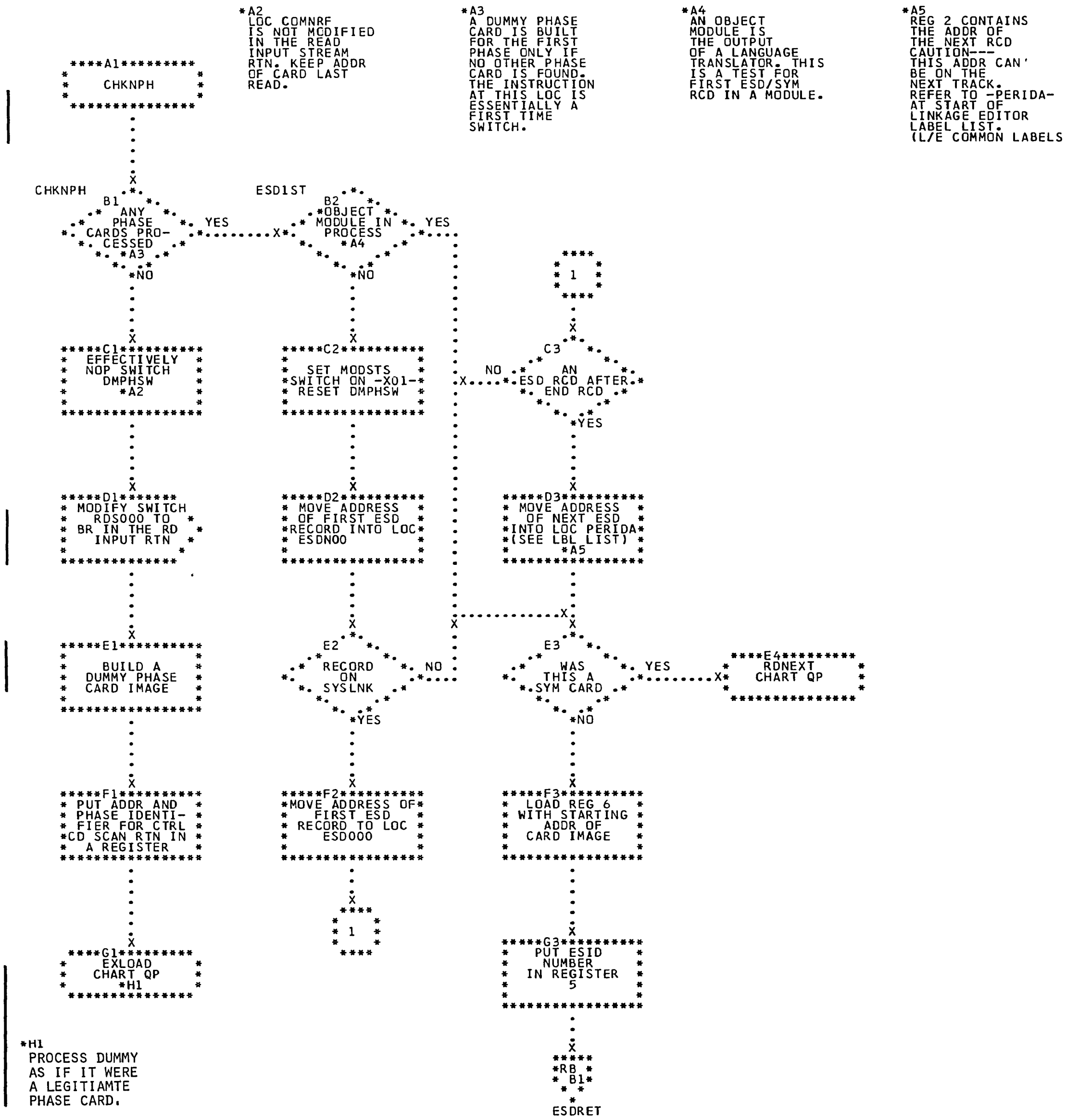


Chart RB. ESD Processor, Card Image Check, (Part 1 of 2)
$LNKEDT0; Refer to Linkage Editor, Chart 32

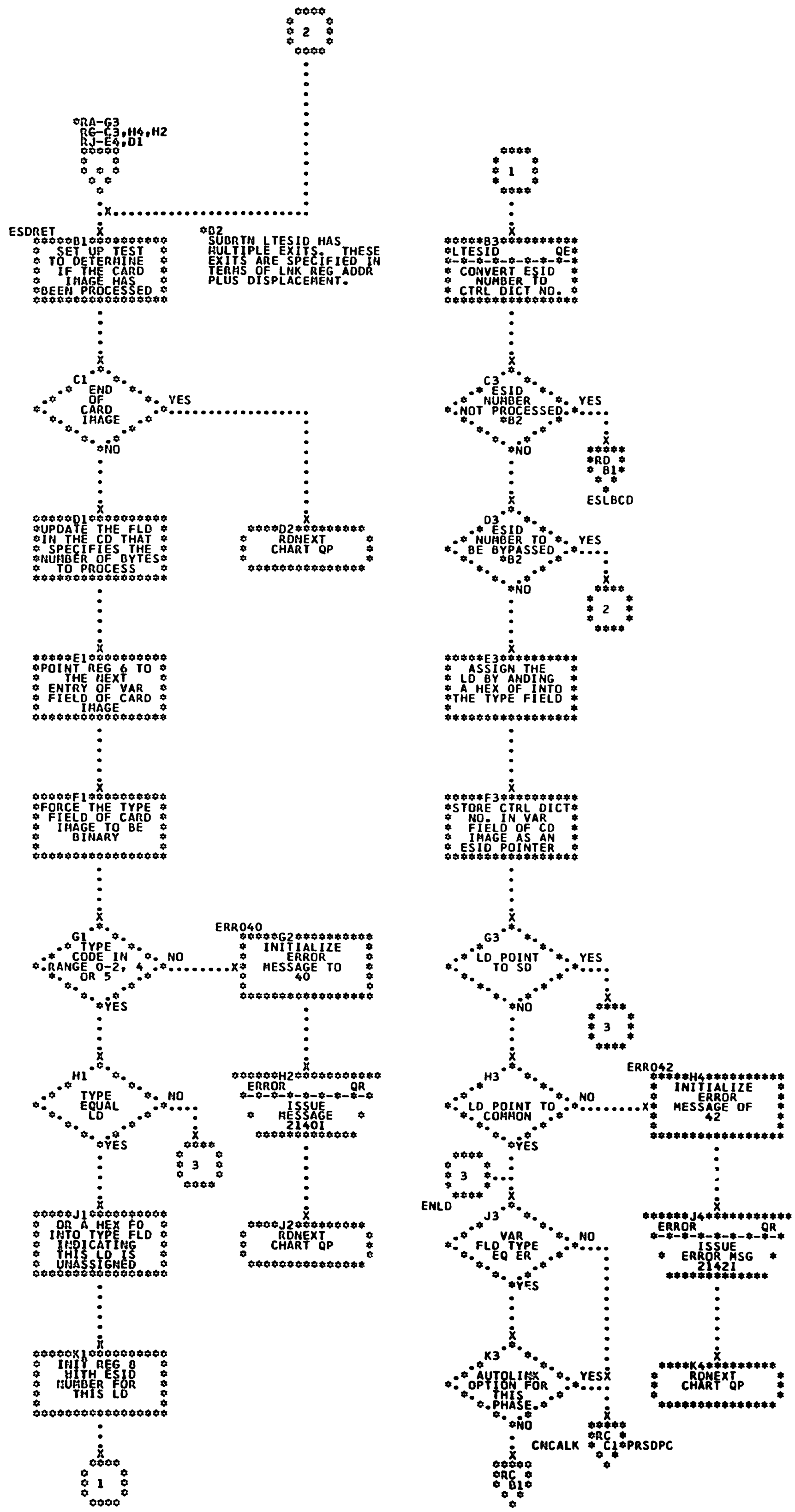


Chart RC    ESD Processor, Card Image Check, (Part 2 of 2)
$LNKEDT0; Refer to Linkage Editor, Chart 32

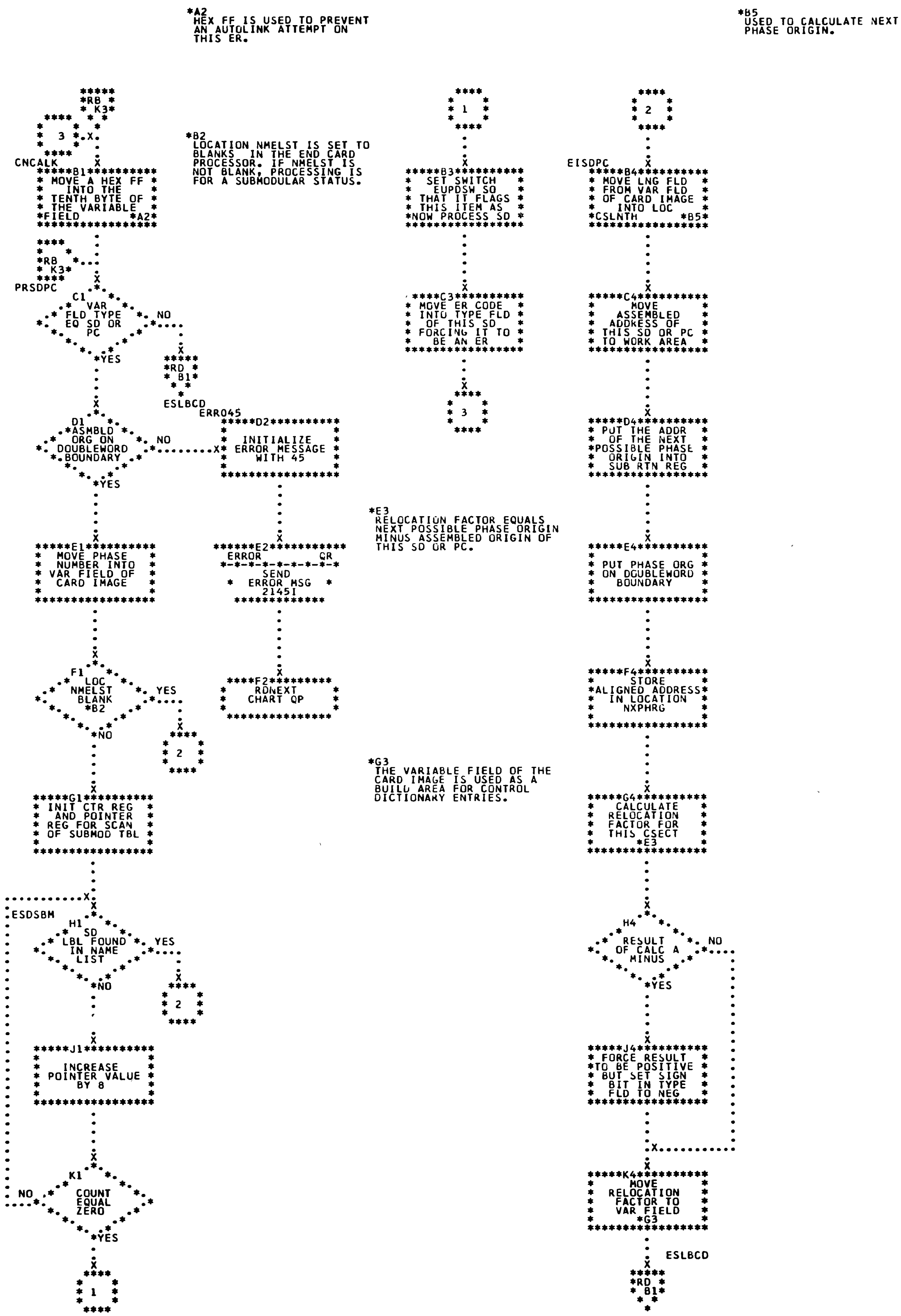


Chart RD. ESD Processor, Process ESD Items Against Control Dictionary $LNKEDT0; Refer to Linkage Editor, Chart 32

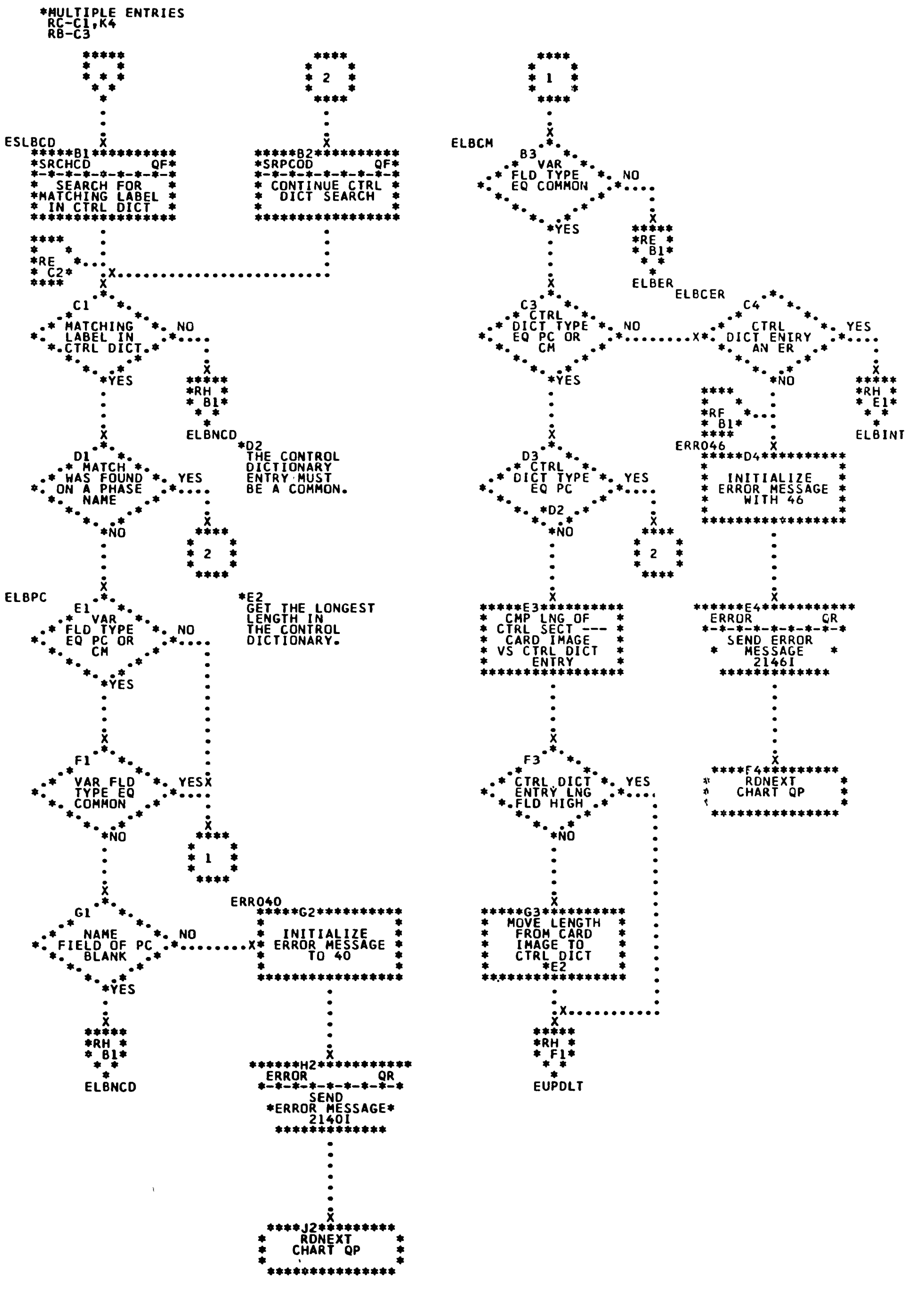


Chart RE. ESD Processor, Process ER $LNKEDT0; Refer to Linkage Editor, Chart 32

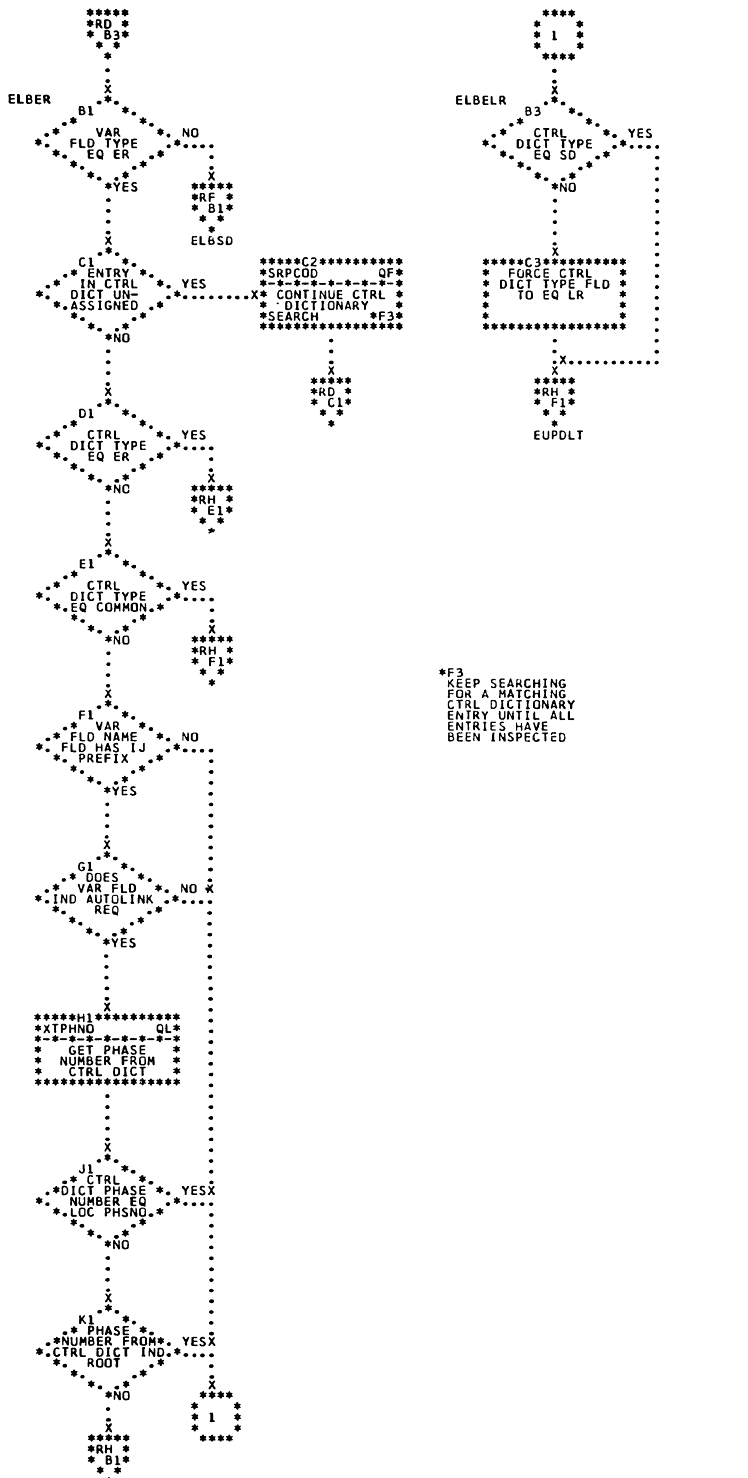


Chart RF. ESD Processor, Process SD $LNKEDT0; Refer to Linkage Editor, Chart 32

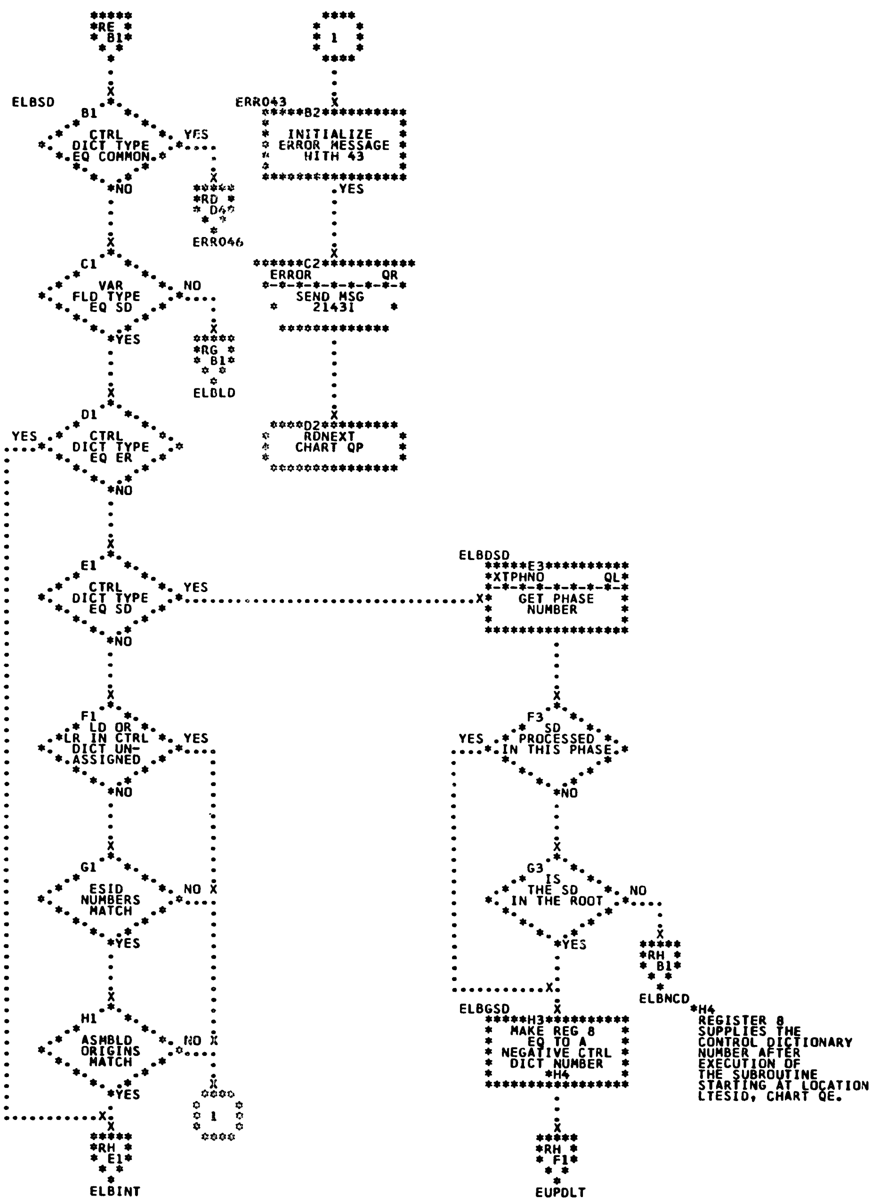


Chart RG. ESD Processor, Process LD/LR $LNKEDT0; Refer to Linkage Editor, Chart 32

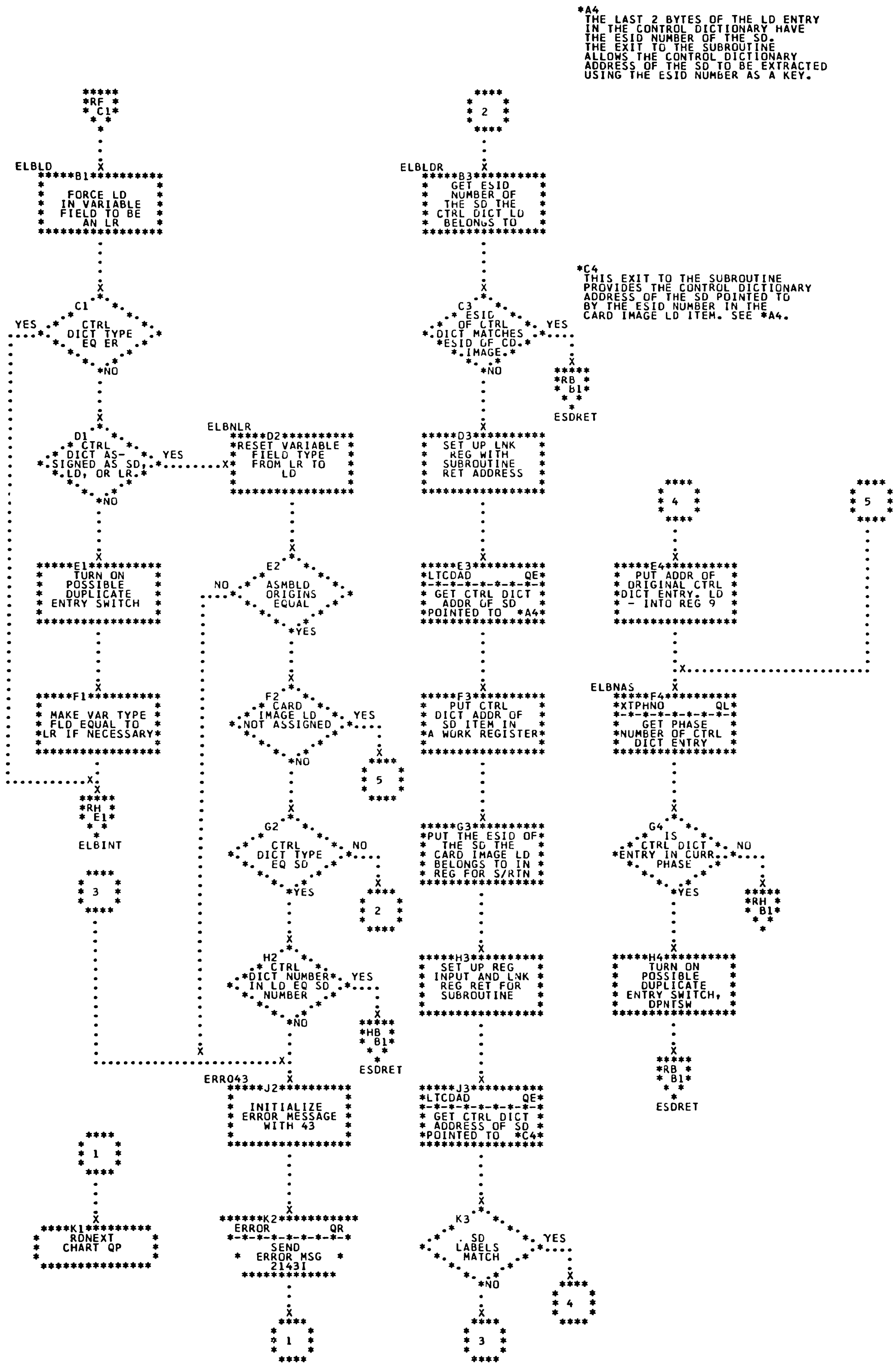


Chart RH. ESD Processor, Update Linkage Table and Control Dictionary (Part 1 of 2) $LNKEDT0; Refer to Linkage Editor, Chart 32

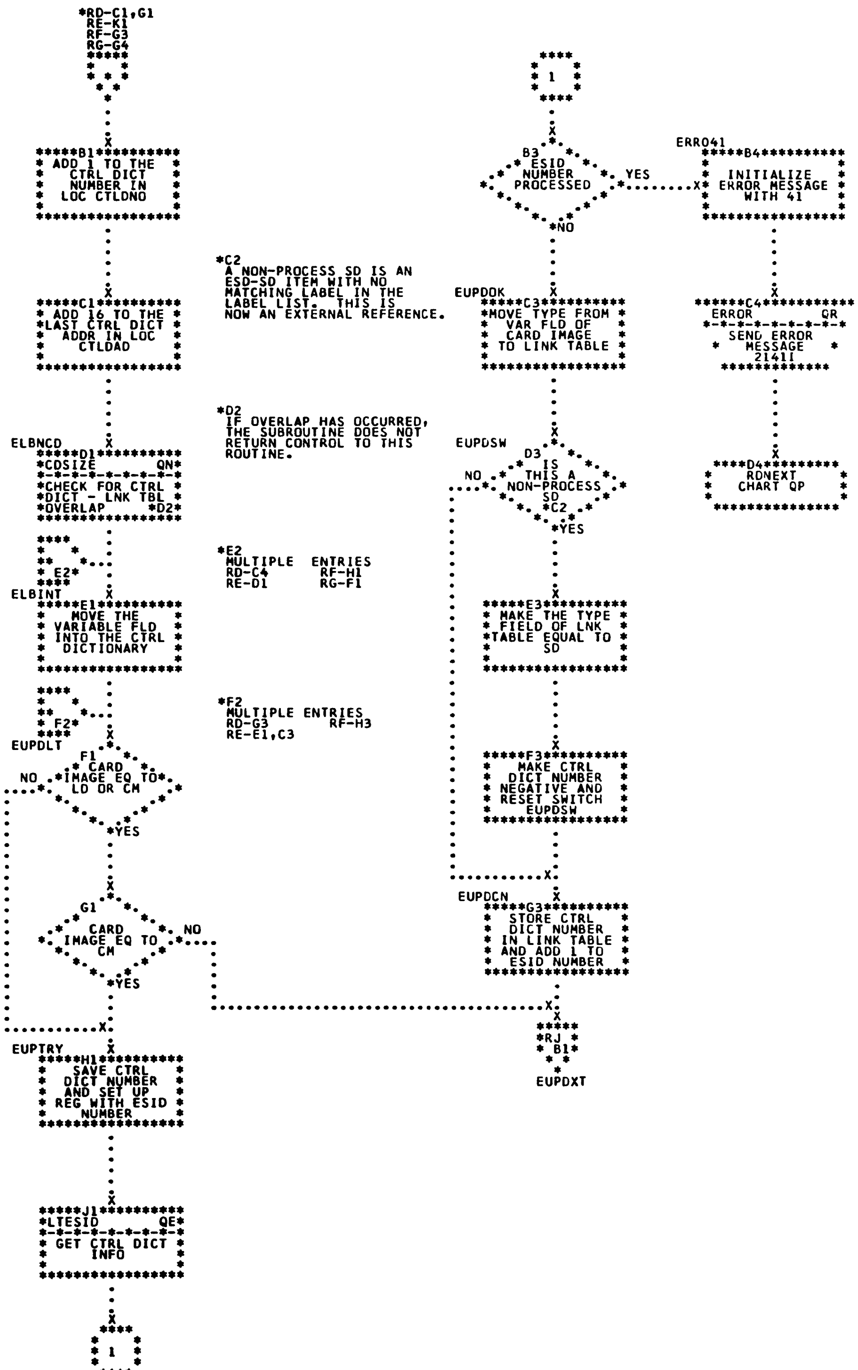
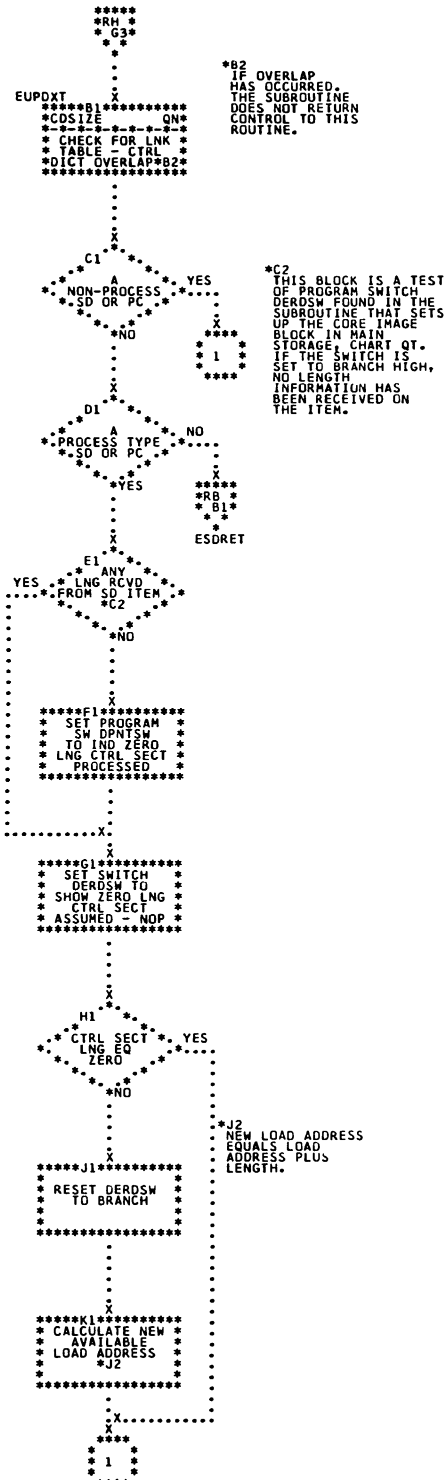


Chart RJ. ESD Processor, Update Linkage Table and Control Dictionary (Part 2 of 2) $LNKEDT0; Refer to Linkage Editor, Chart 32



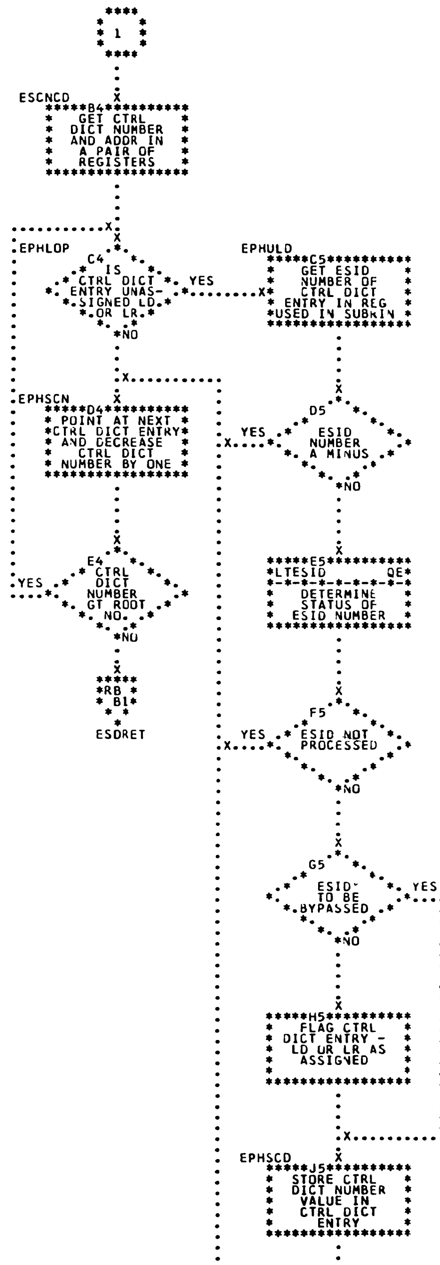


Chart RK. Initialize for $LNKEDT2; Refer to Linkage Editor, Chart 33

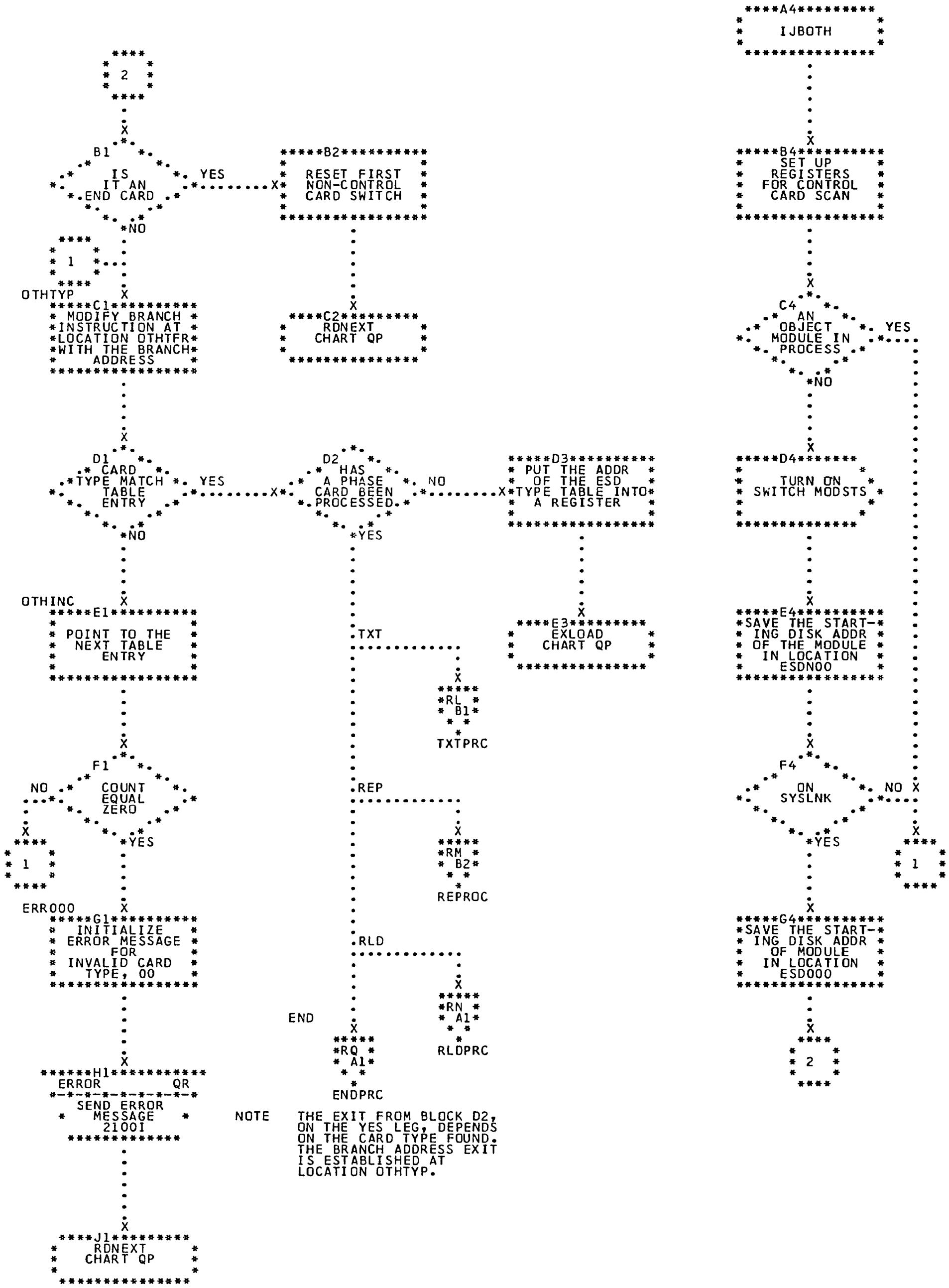


Chart RL. TXT Processor $LNKEDT2; Refer to Linkage Editor, Chart 33

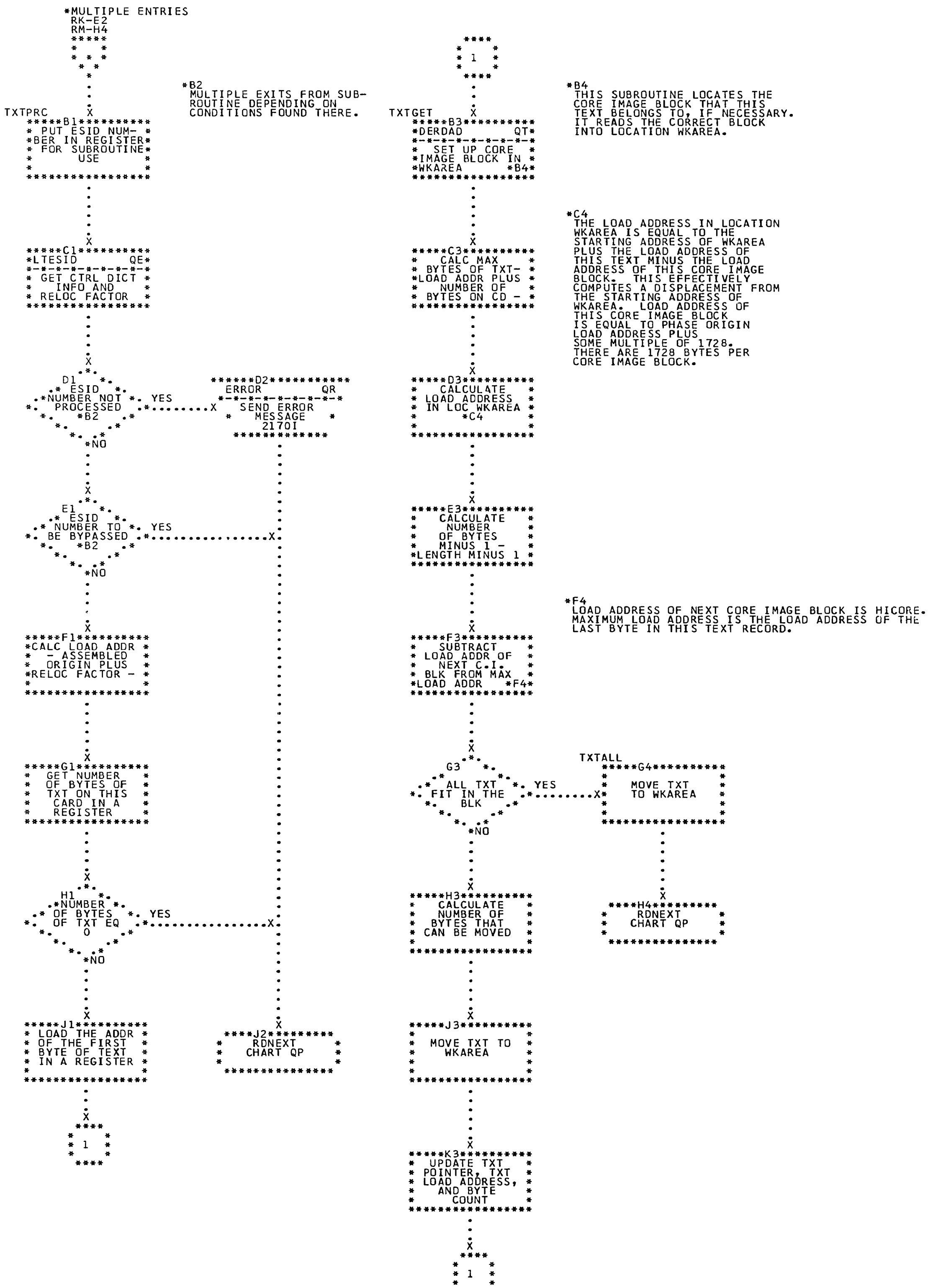


Chart RM. REP Processor $LNKEDT2; Refer to Linkage Editor, Chart 33

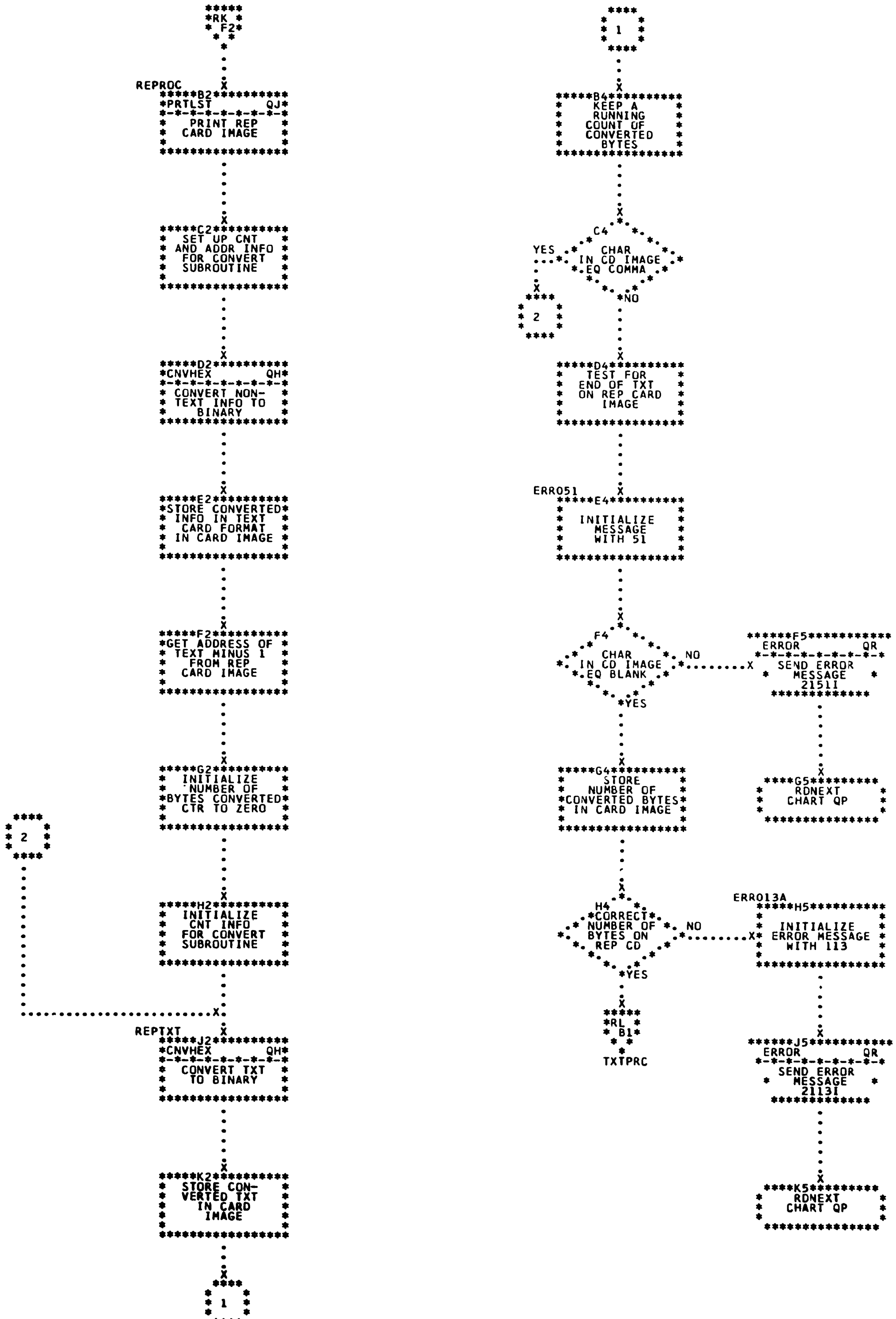


Chart RN. RLD Pass 1 Processing (Part 1 of 2) $LNKEDT2; Refer to Linkage Editor, Chart 33

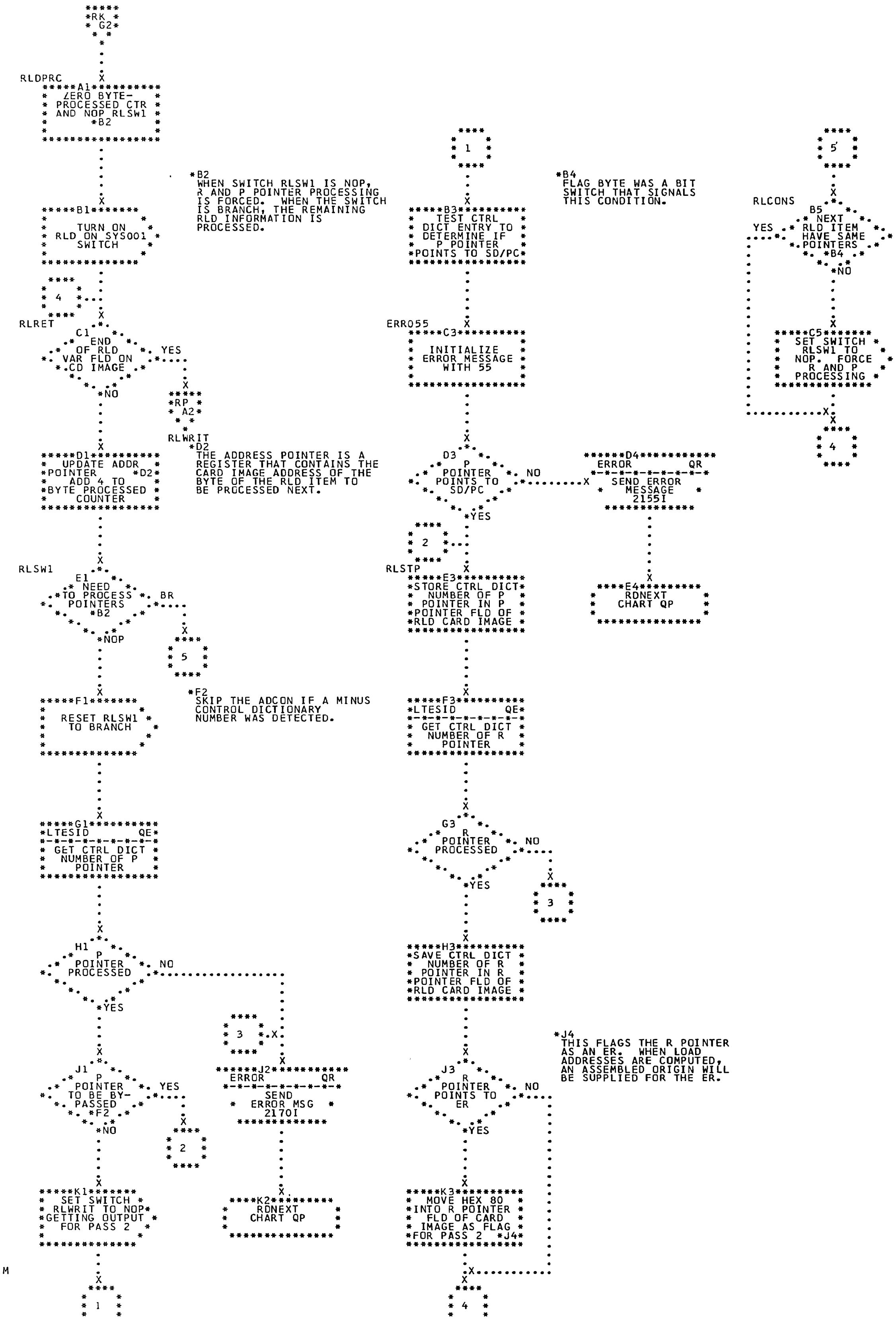


Chart RP. RLD Pass 1 Processing (Part 2 of 2) $LNKEDT2;
Refer to Linkage Editor, Chart 33

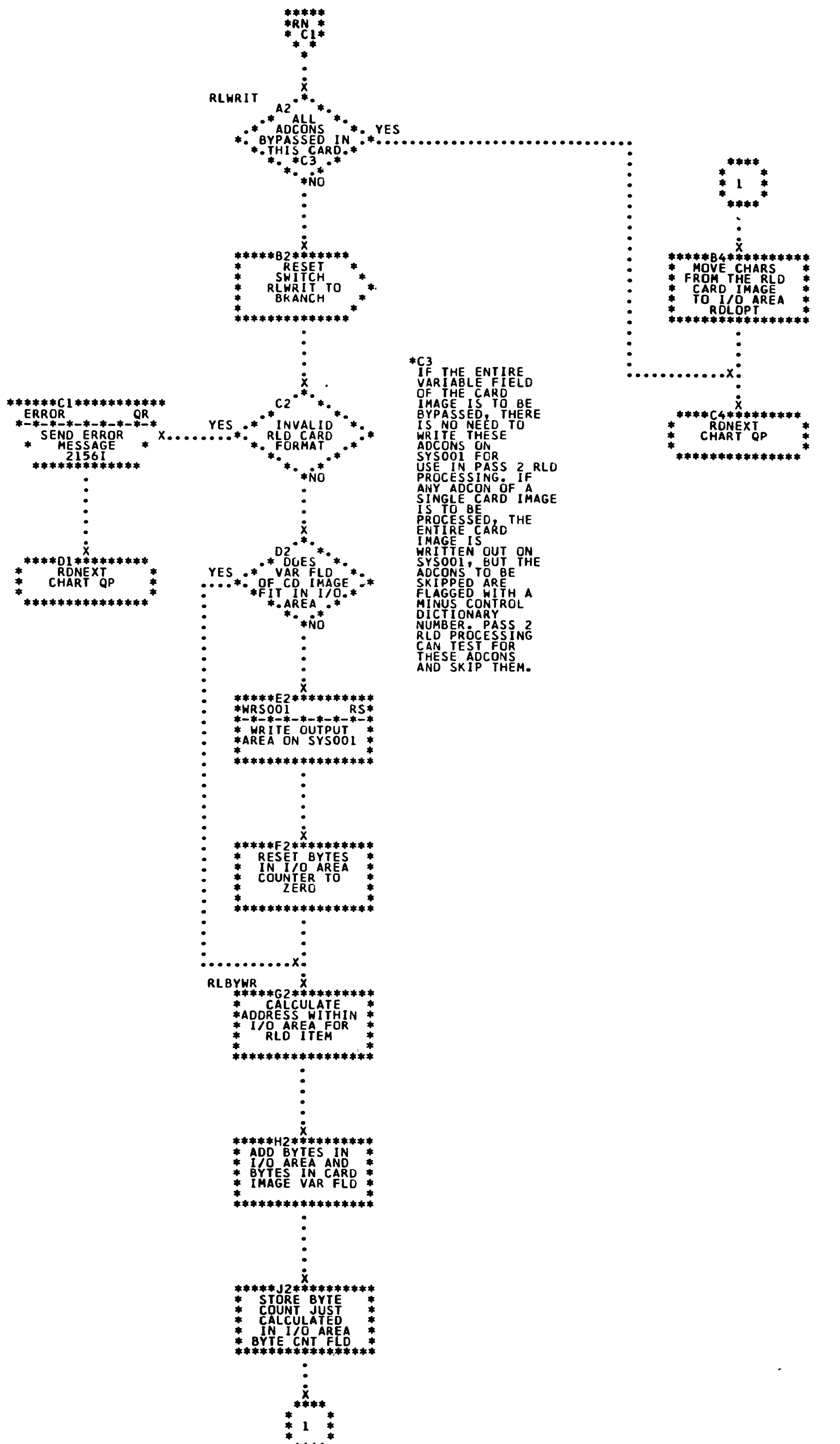


Chart RQ. END Processor (Part 1 of 2) $LNKEDT2; Refer to Linkage Editor, Chart 33

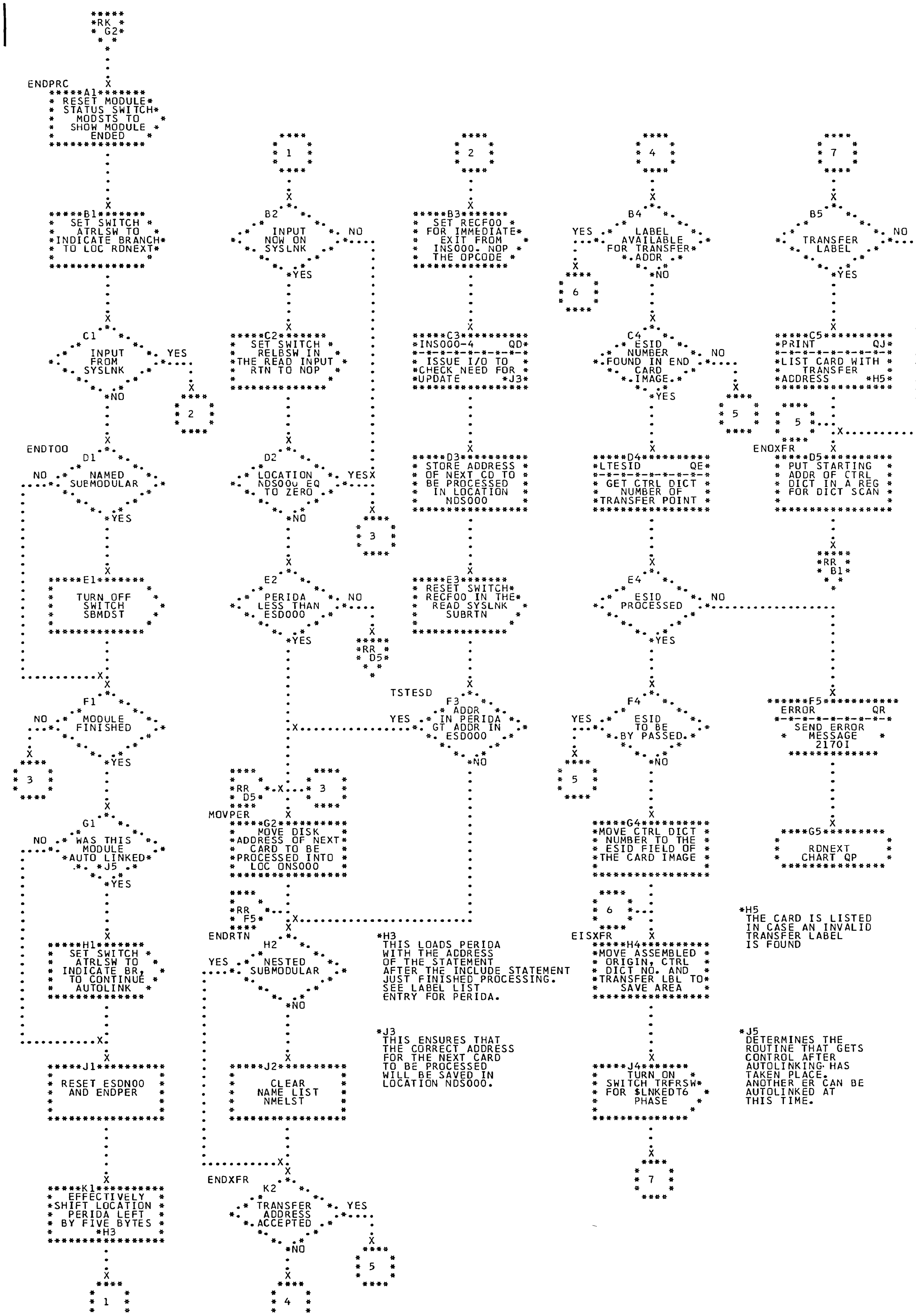


Chart RR. END Processor (Part 2 of 2) $LNKEDT2; Refer to Linkage Editor, Chart 33

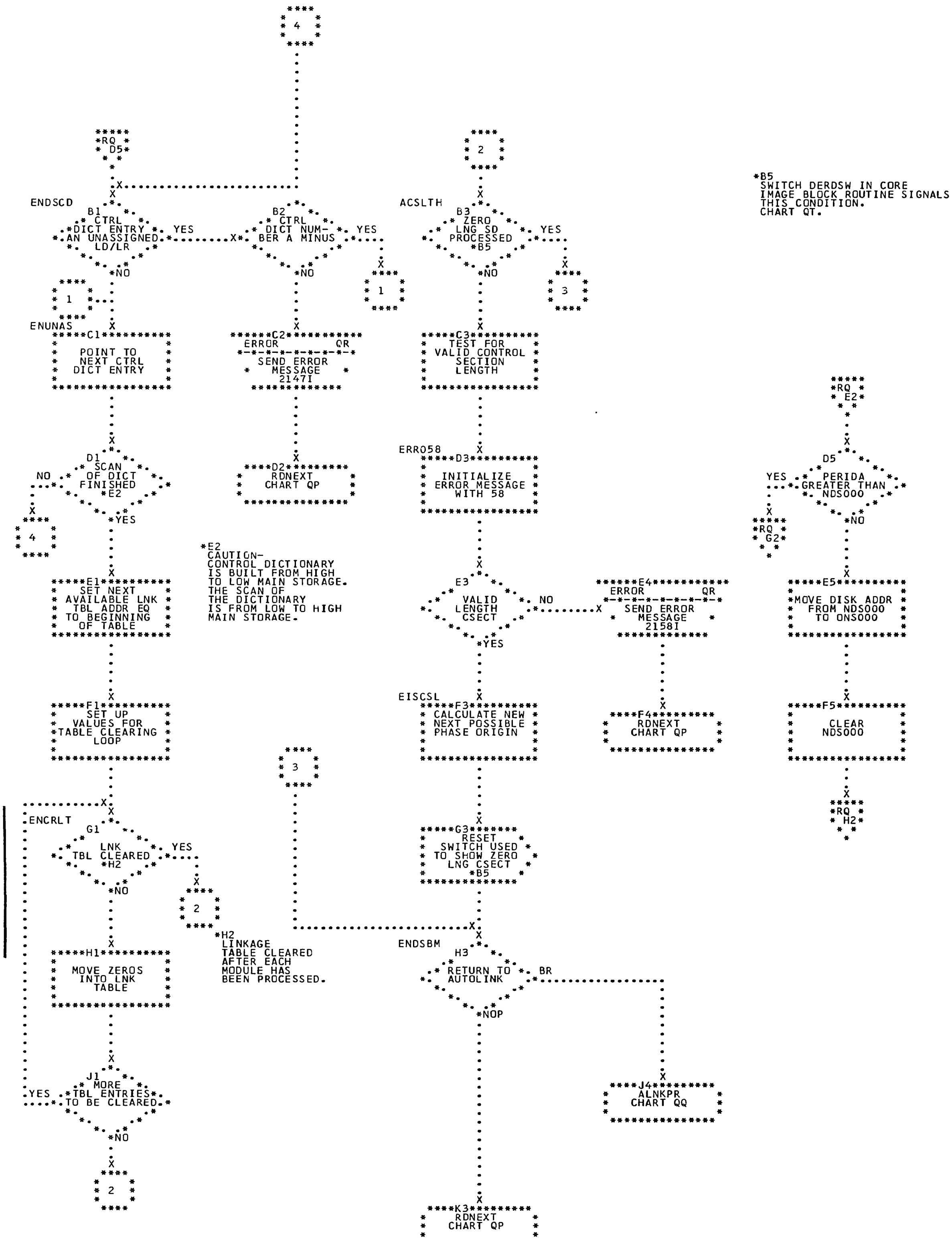


Chart RS. Write SYS001 Subroutine $LNKEDT2; Refer to Linkage Editor, Chart 33

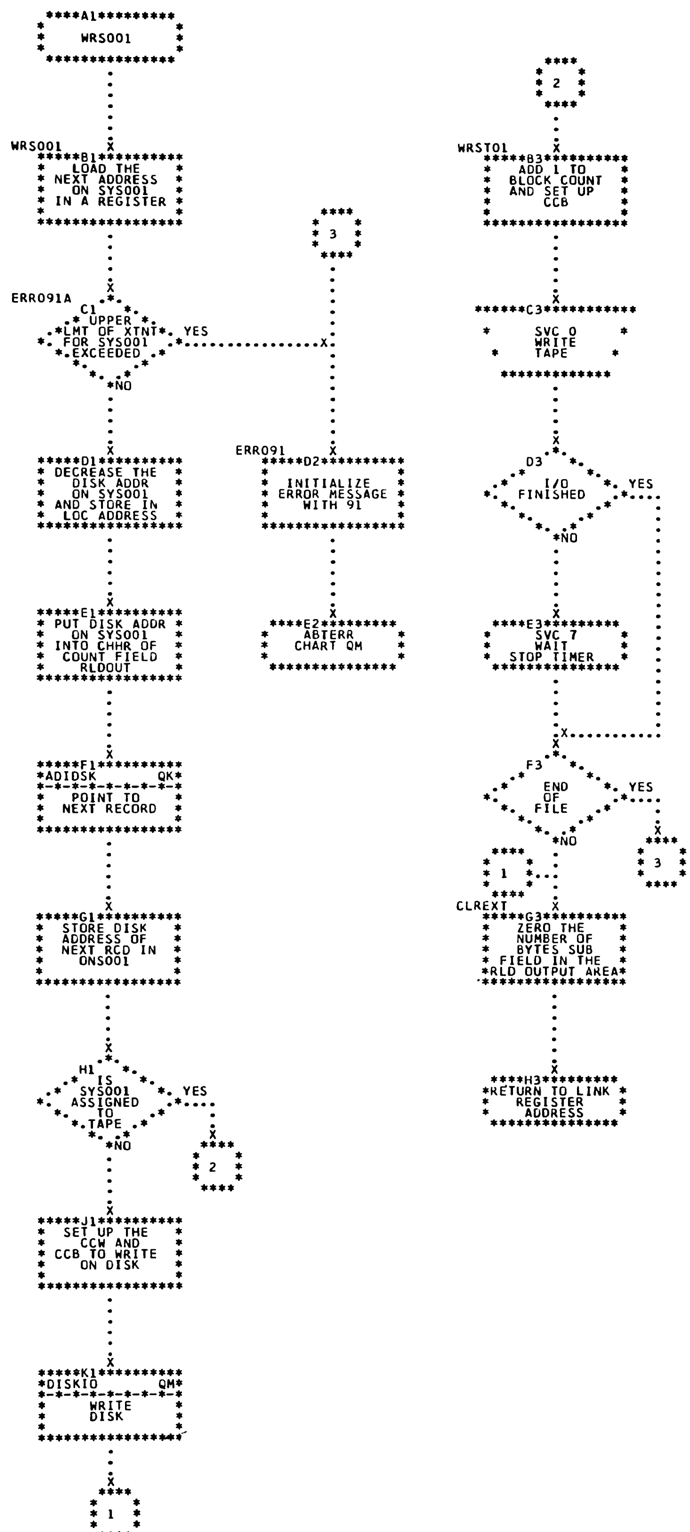


Chart RT. Initialize Control Card Processor $LNKEDT4; Refer to Linkage Editor, Chart 34

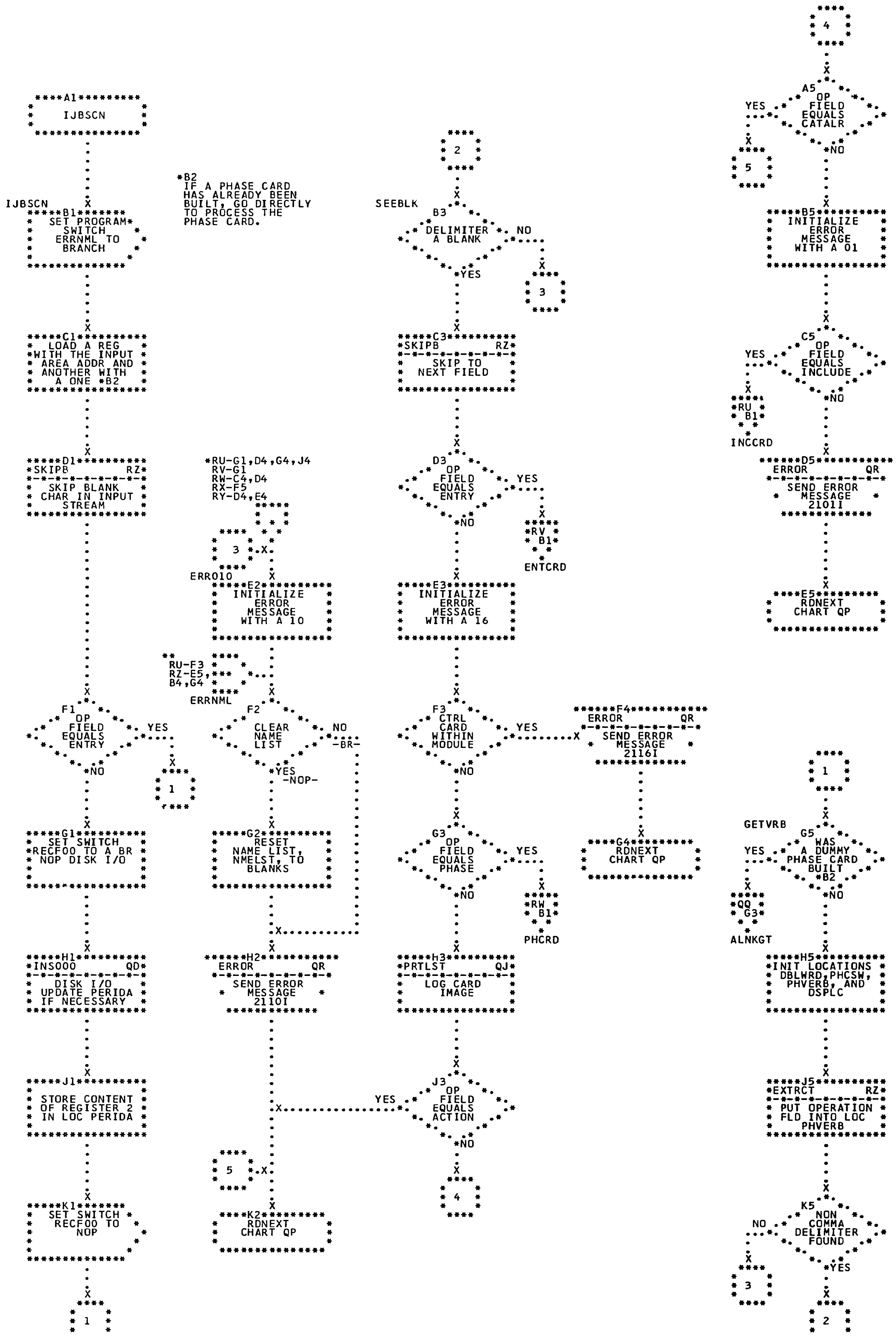


Chart RU. Include Card Processor $LNKEDT4; Refer to Linkage Editor, Chart 34

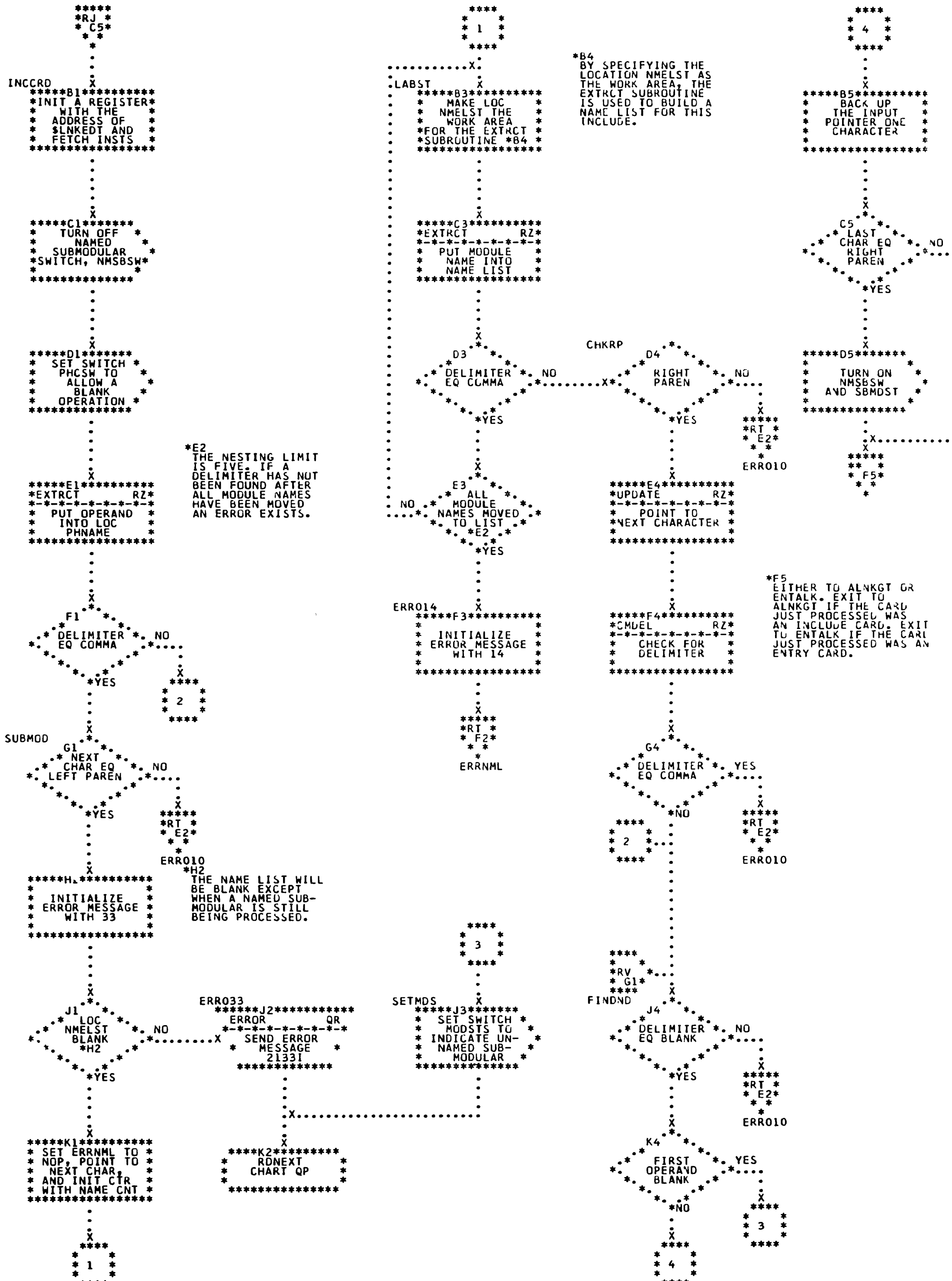


Chart RV. Entry Card Processor $LNKEDT4; Refer to Linkage Editor Chart 34

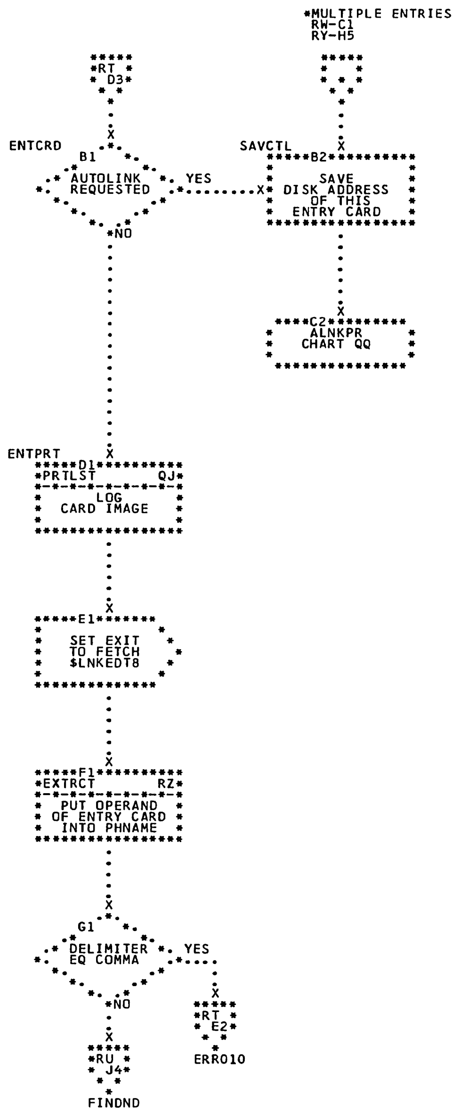


Chart RW. Phase Card Processor (Part 1 of 3) $LNKEDT4;
Refer to Linkage Editor, Chart 34

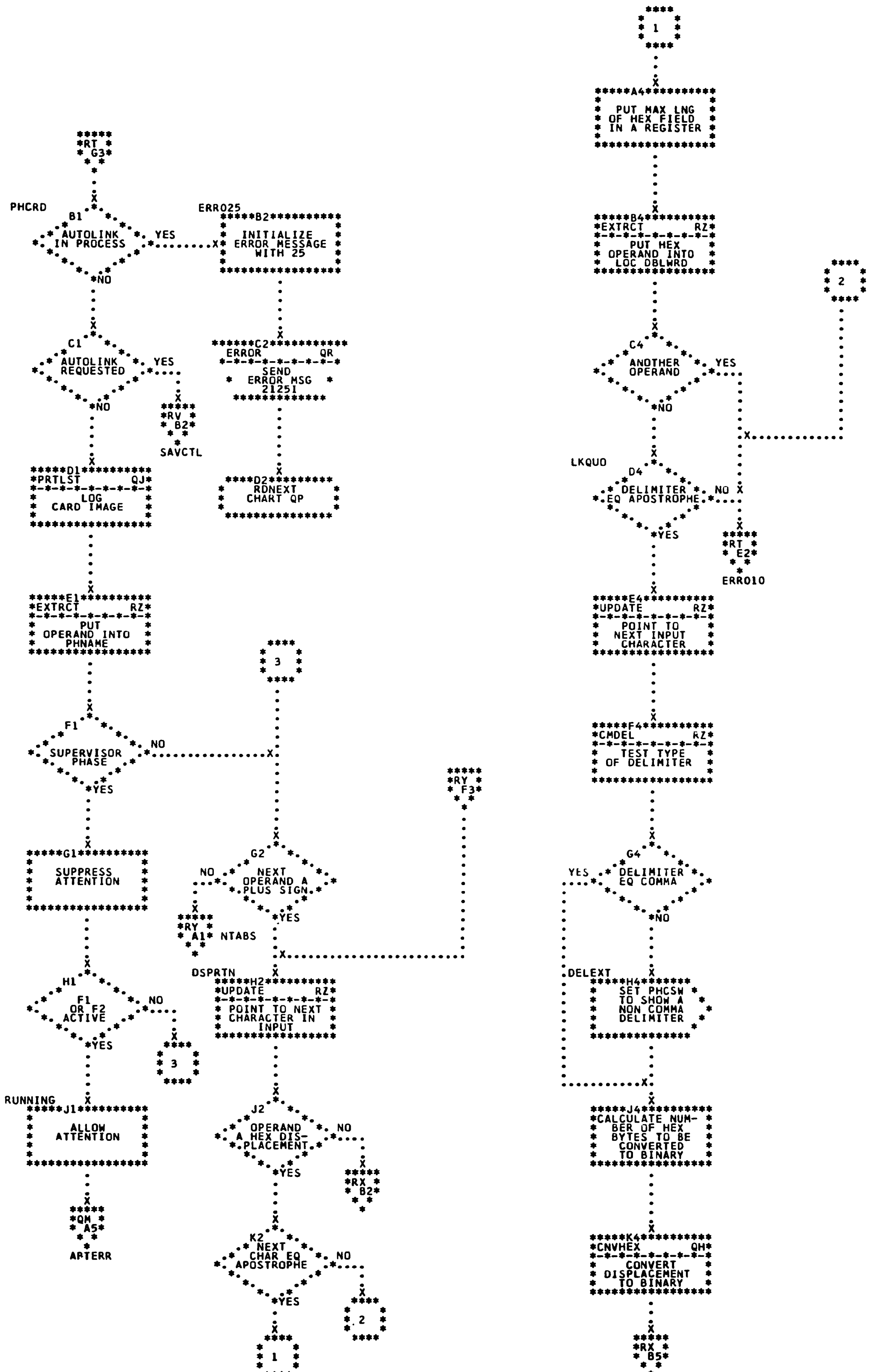


Chart RX. Phase Card Processor (Part 2 of 3) $LNKEDT4; Refer to Linkage Editor, Chart 34

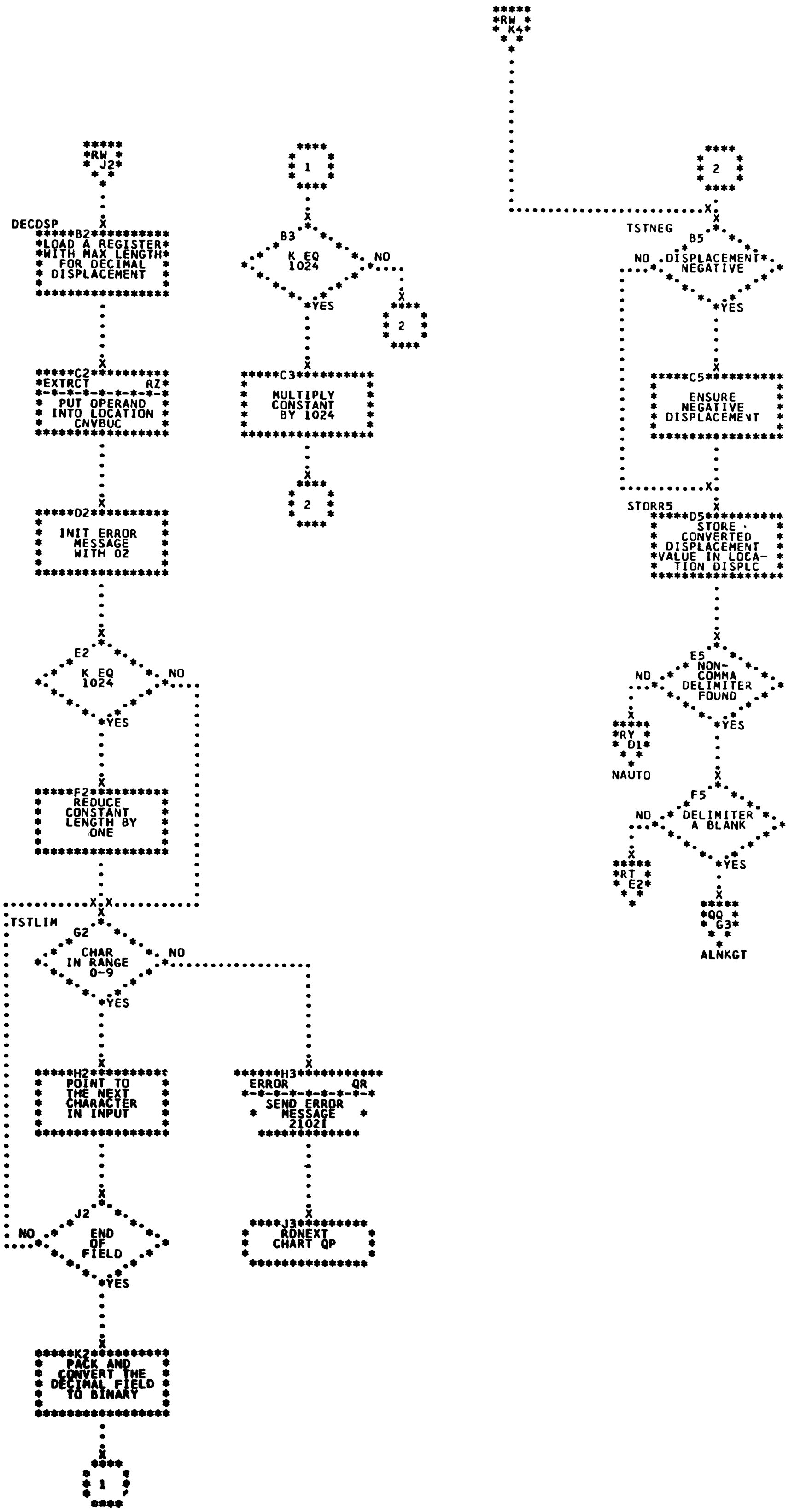


Chart RY. Phase Card Processor (Part 3 of 3) $LNKEDT4; Refer to Linkage Editor, Chart 34

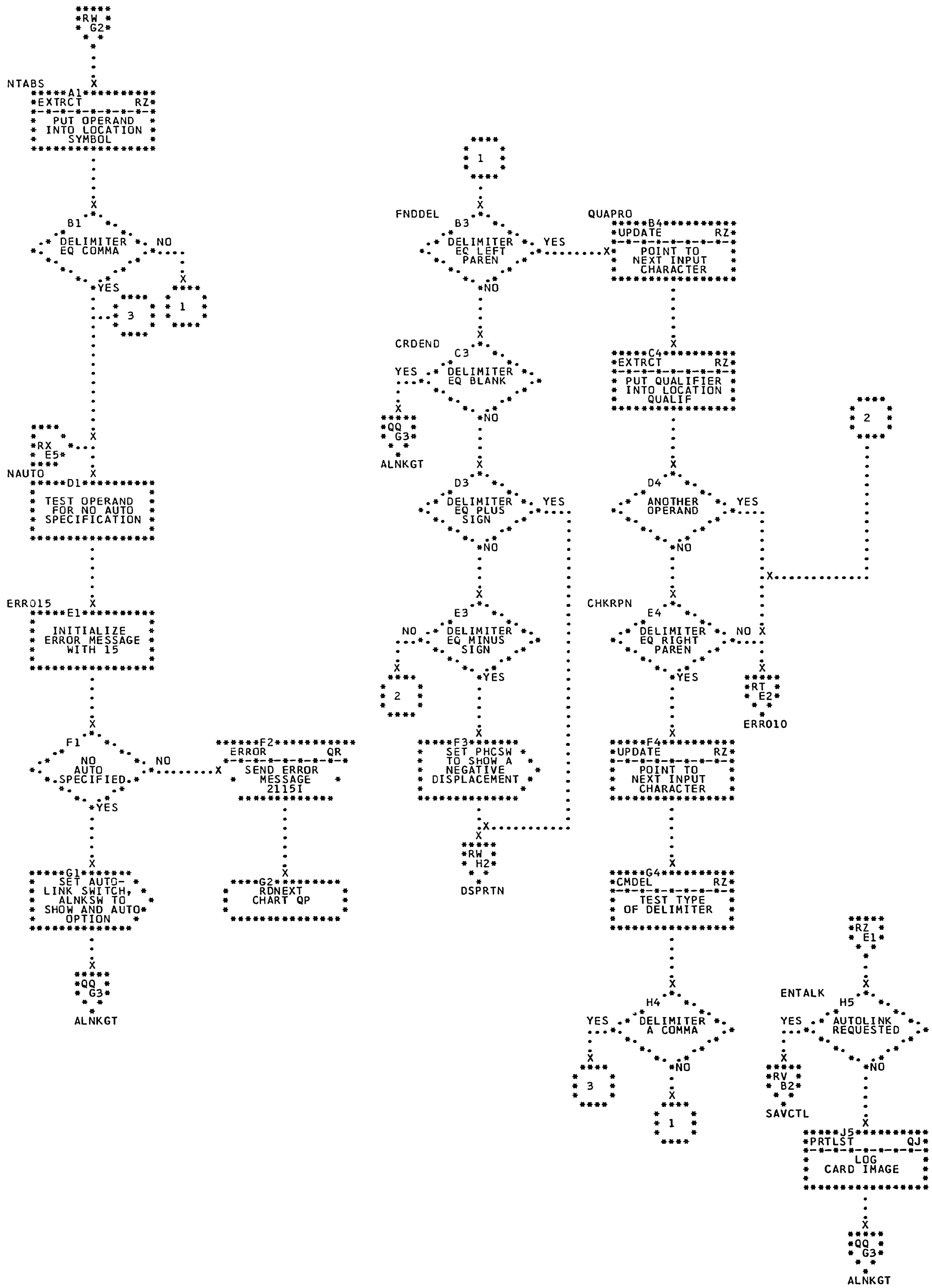


Chart RZ. Skip Blanks and Extract Field Subroutine
$LNKEDT4; Refer to Linkage Editor, Chart 34

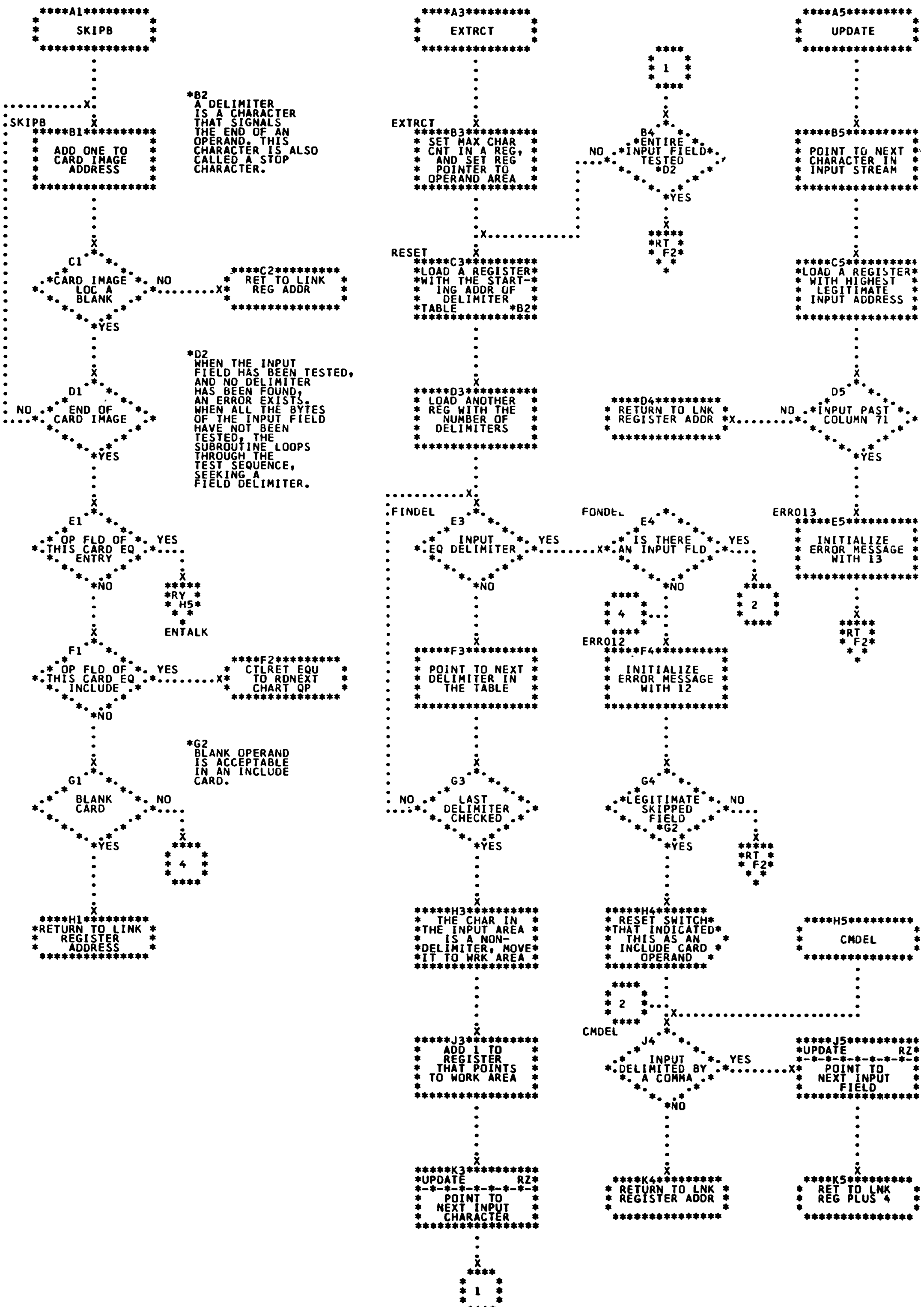


Chart SA. Phase Post Processing $LNKEDT6 (Part 1 of 6); Refer to Linkage Editor, Chart 35

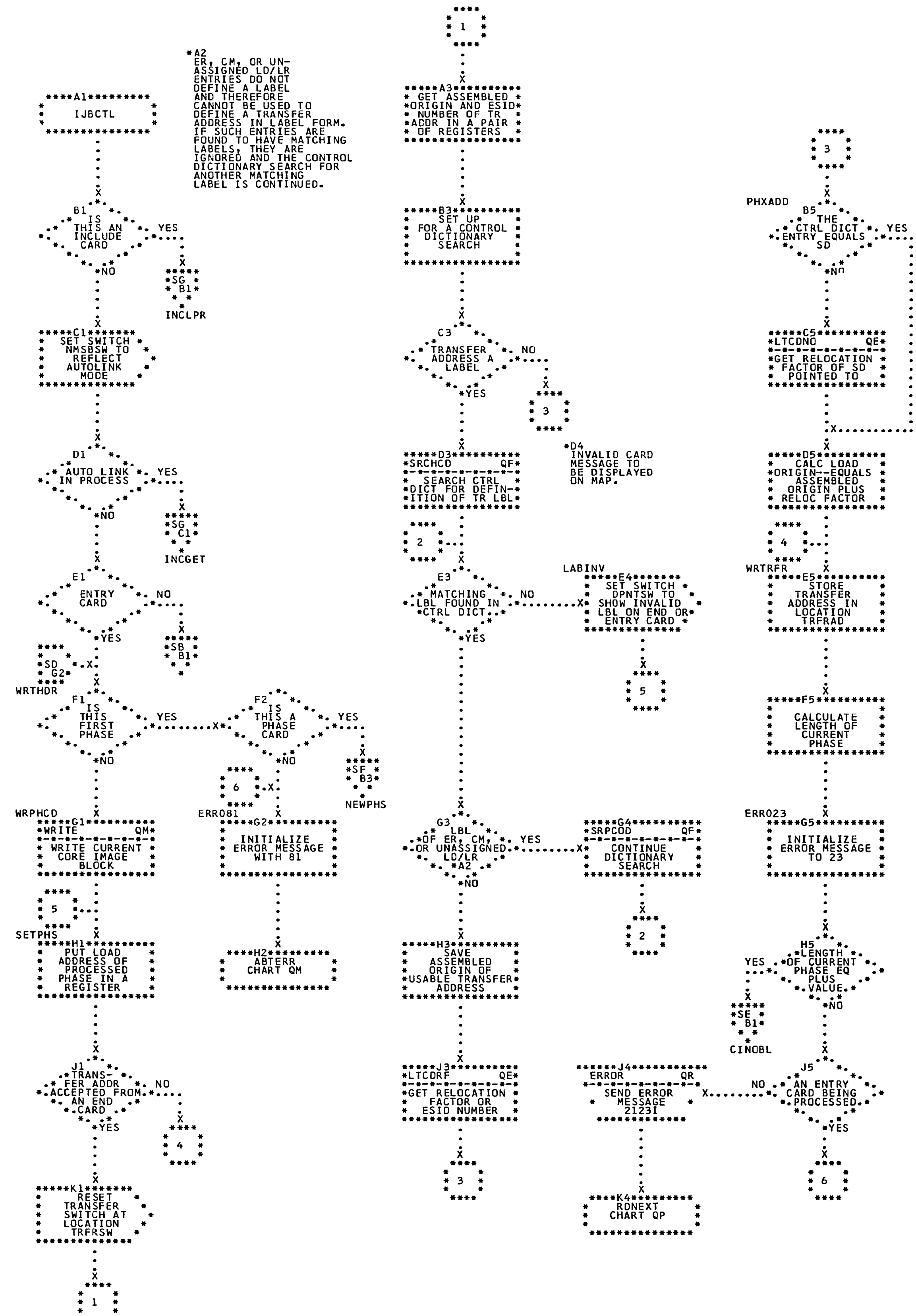
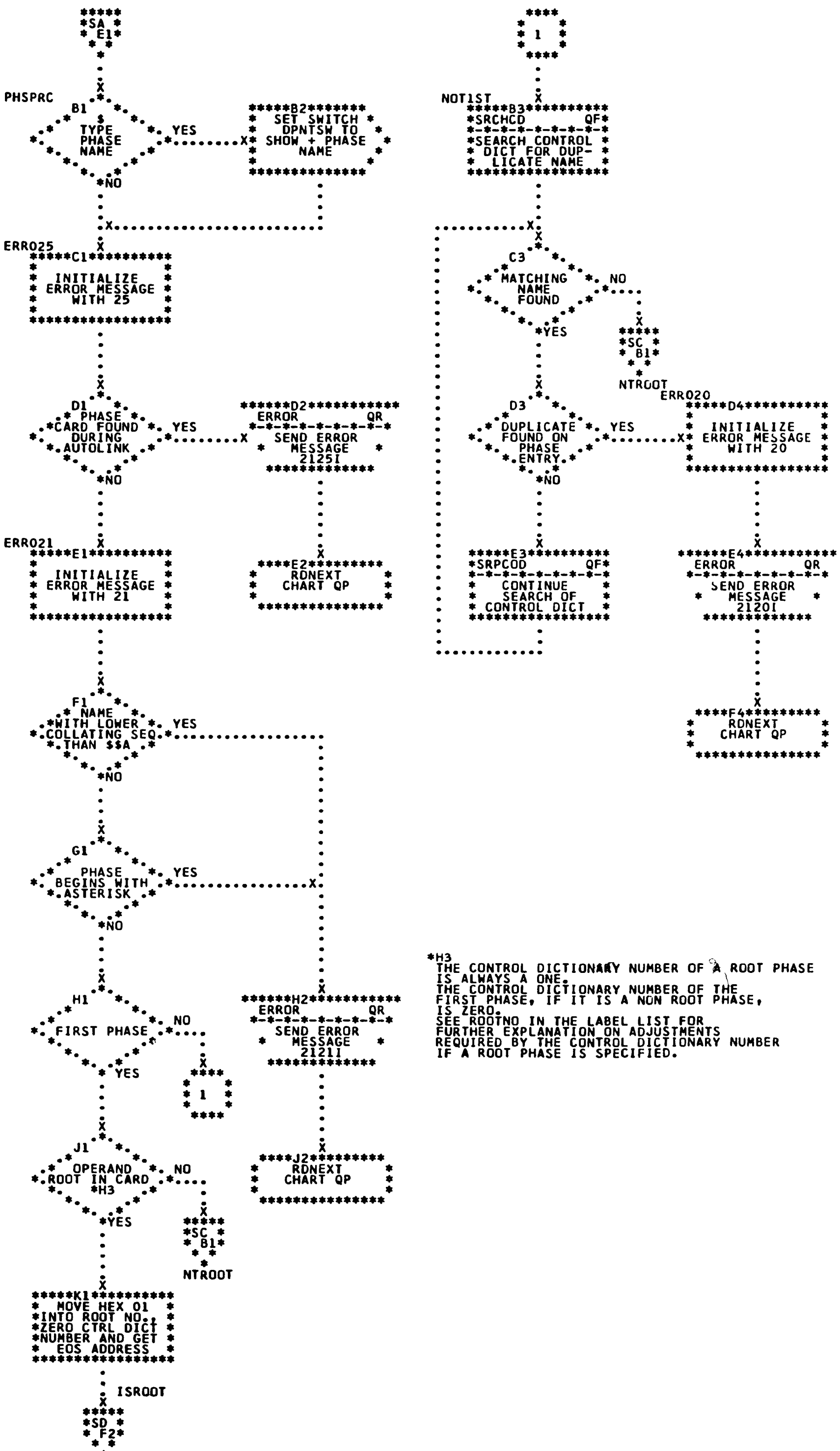


Chart SB. Phase Post Processing $LNKEDT6 (Part 2 of 6); Refer to Linkage Editor, Chart 35




*H3
THE CONTROL DICTIONARY NUMBER OF A ROOT PHASE IS ALWAYS A ONE.
THE CONTROL DICTIONARY NUMBER OF THE FIRST PHASE, IF IT IS A NON ROOT PHASE, IS ZERO.
SEE ROOTNO IN THE LABEL LIST FOR FURTHER EXPLANATION ON ADJUSTMENTS REQUIRED BY THE CONTROL DICTIONARY NUMBER IF A ROOT PHASE IS SPECIFIED.

Chart SC. Phase Post Processing $LNKEDT6 (Part 3 of 6); Refer to Linkage Editor, Chart 35

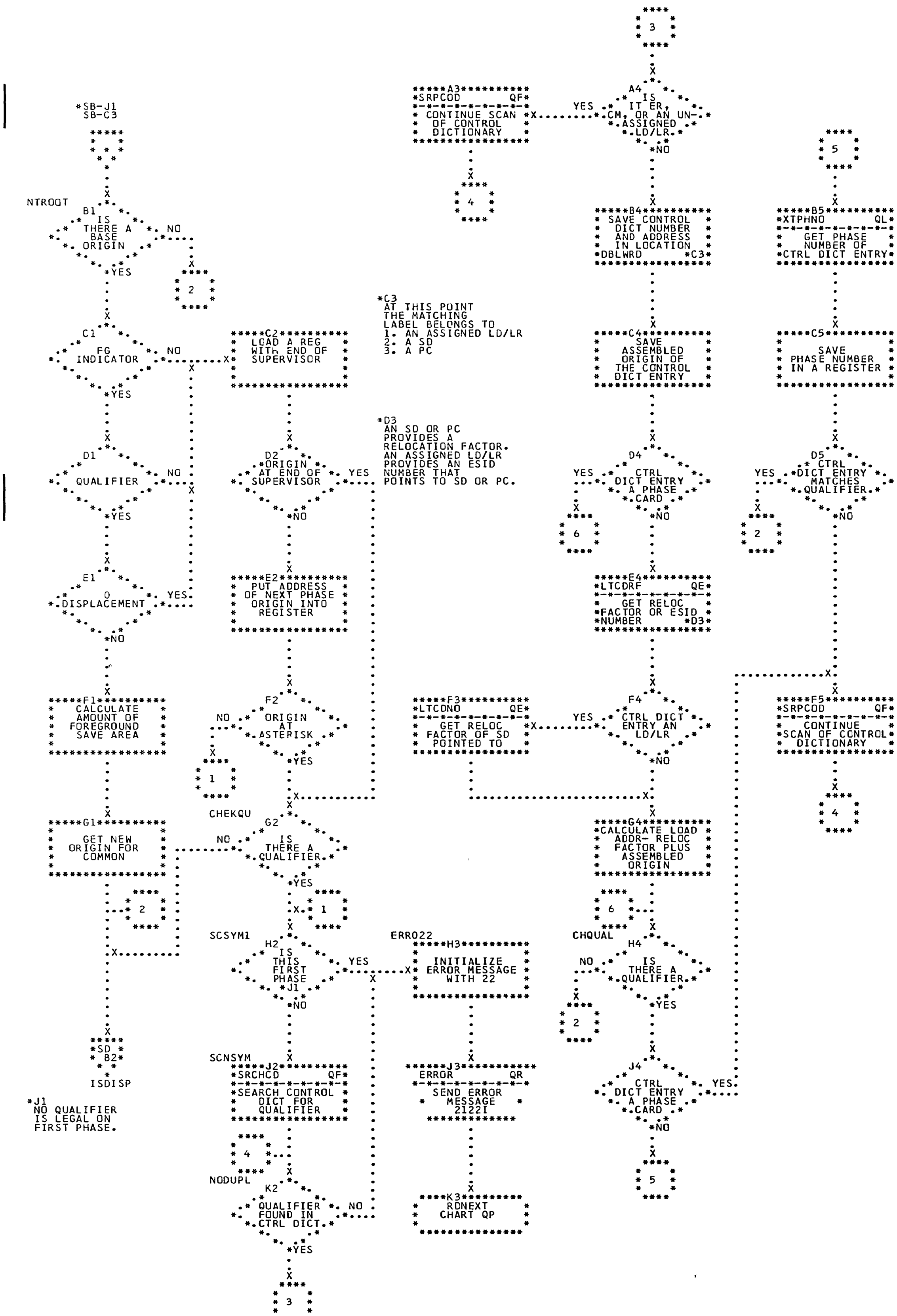


Chart SD. Phase Post Processing $LNKEDT6 (Part 4 of 6);
Refer to Linkage Editor, Chart 35

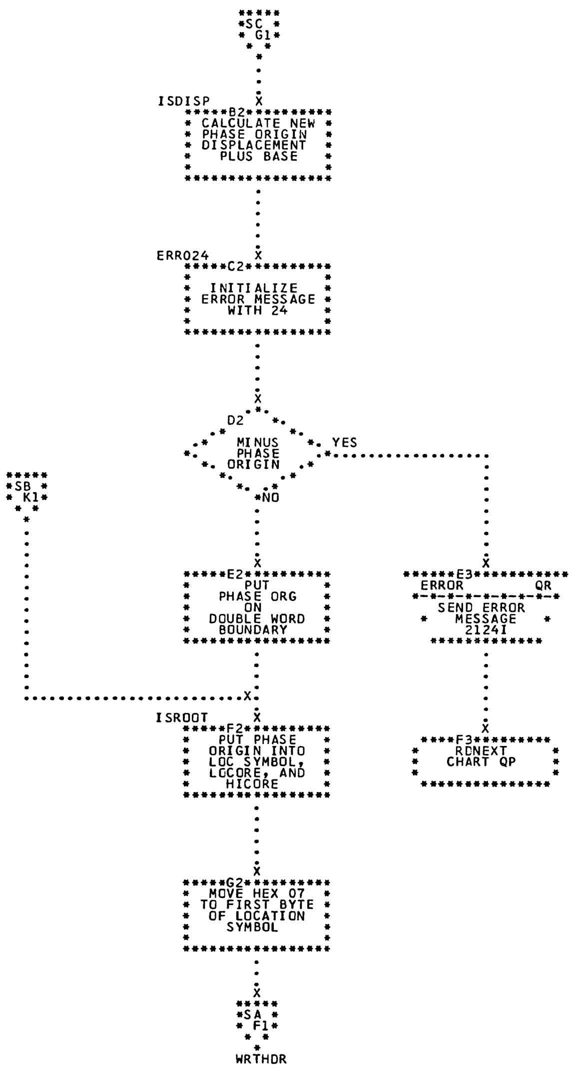


Chart SE. Phase Post Processing $LNKEDT6 (Part 5 of 6);
Refer to Linkage Editor, Chart 35

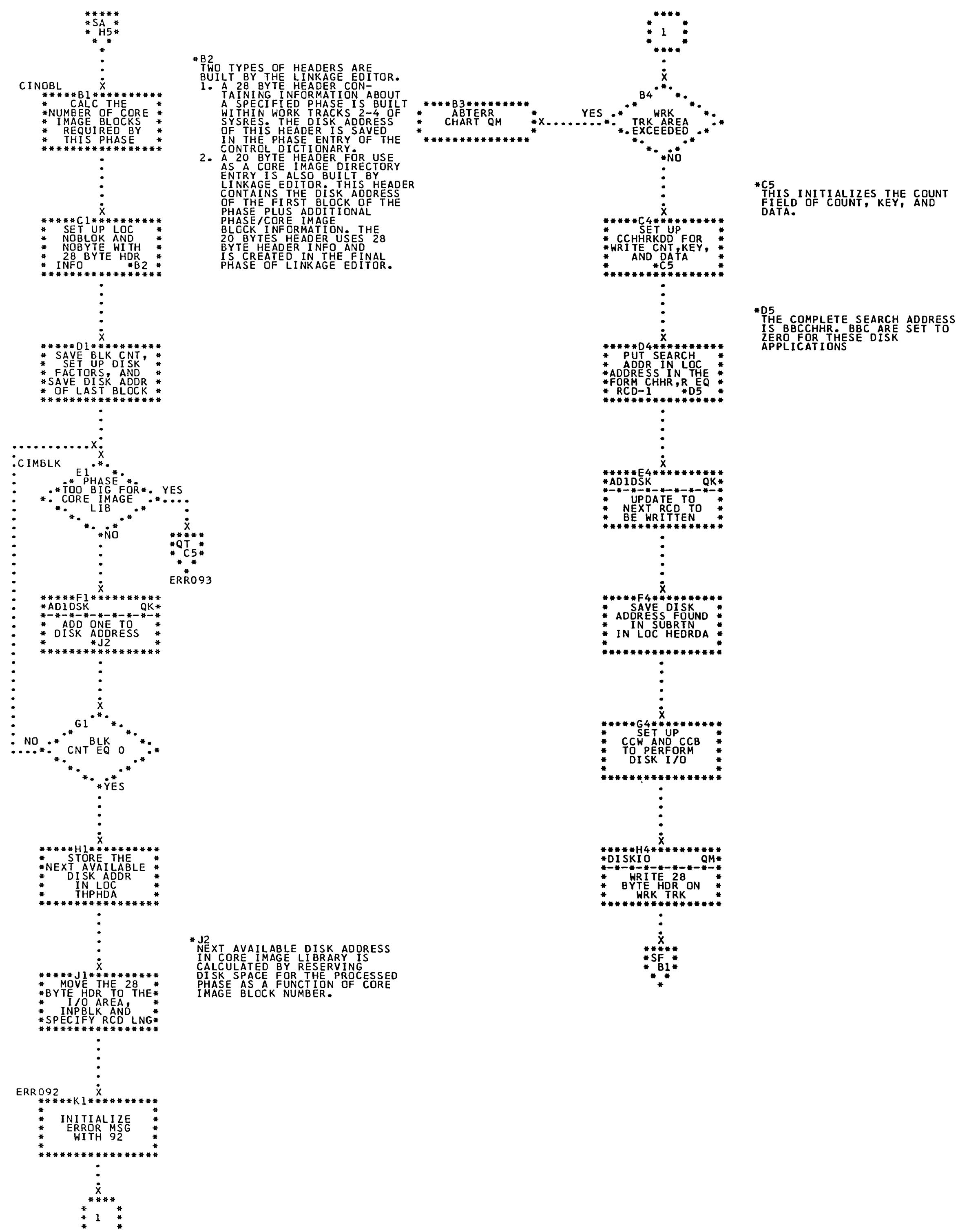
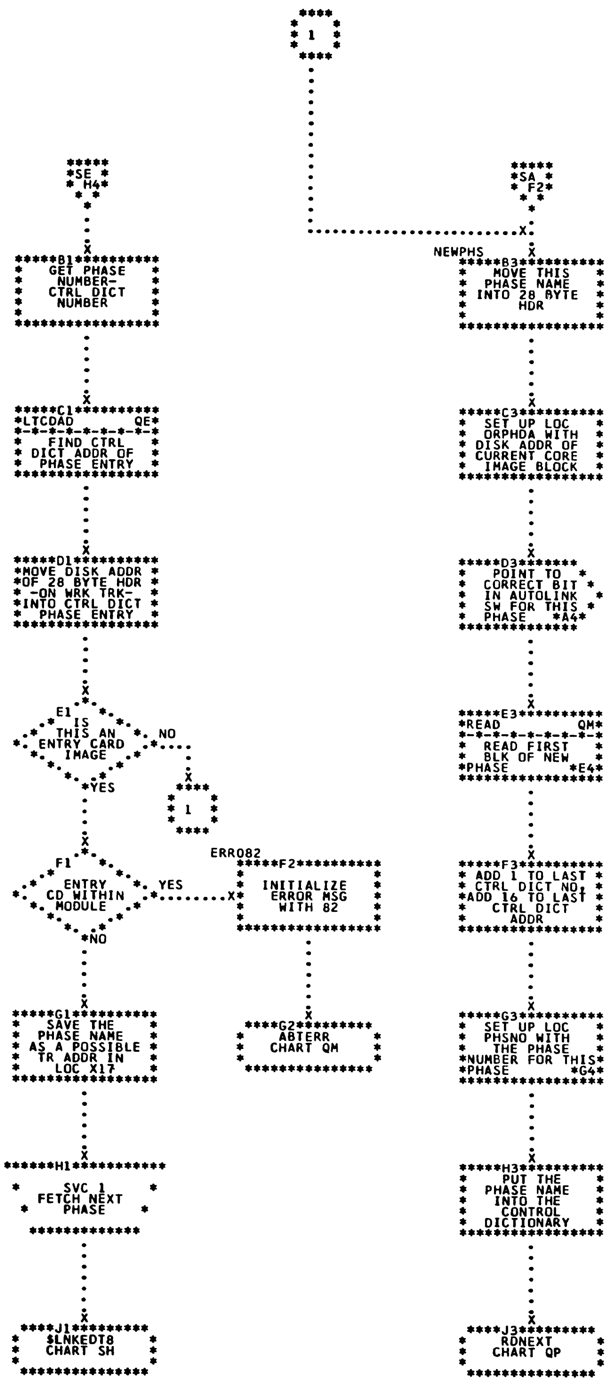


Chart SF. Phase Post Processing $LNKEDT6 (Part 6 of 6); Refer to Linkage Editor, Chart 35




*A4
THE AUTOLINK SWITCH, ALNKSW, IS INITIALIZED WITH ZEROS IN ALL BIT POSITIONS SIGNIFYING AN AUTOMATIC AUTOLINK OPTION FOR ALL PHASES. IF AN ACTION CARD WITH A NOAUTO OPERAND IS FOUND, THE BIT POSITIONS OF ALNKSW ARE SET TO ONES. THIS INDICATES THE AUTOLINK FEATURE IS SURPRESSED FOR ALL PHASES. IF AUTOLINK IS SUPPRESSED BY PHASE, ONLY THE BIT POSITION OF ALNKSW CORRESPONDING TO THAT PHASE IS SET TO ONE. THEREFORE, THE PROPER BIT POSITION FOR THIS PHASE IS SHIFTED INTO THE LOW ORDER BIT POSITION OF ALNKSW FOR TESTING BY OTHER PARTS OF THE LINKAGE EDITOR PROGRAM.

*E4
THIS READ INITIALIZES FOR THE DERDAD SUBROUTINE USED DURING TEXT CARD PROCESSING.

*G4
THE SAME PHASE NUMBER IS THE SAME VALUE AS THE CONTROL DICTIONARY NUMBER.

Chart SG. Include Post Processing $LNKEDT6; Refer to Linkage Editor, Chart 35

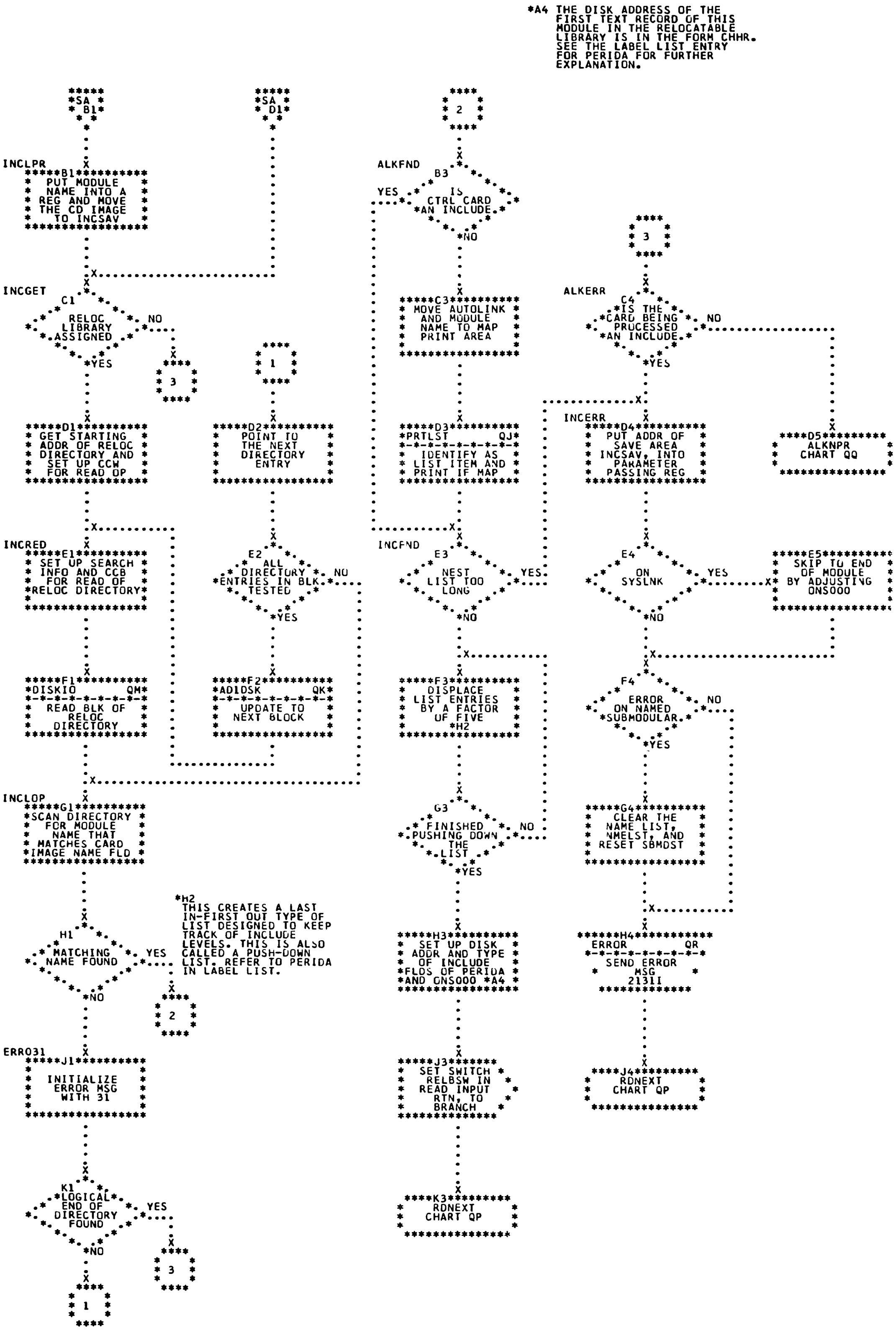
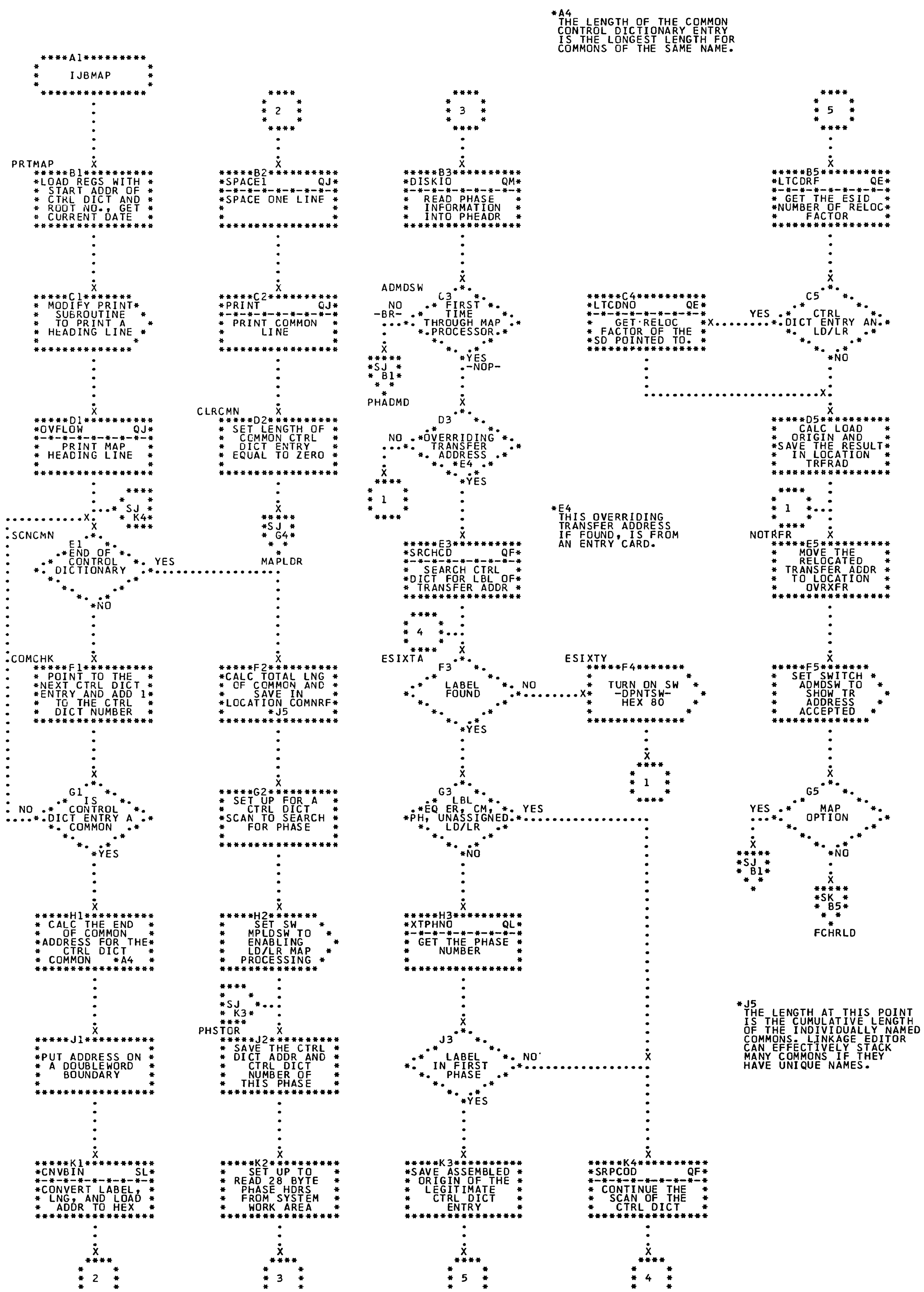


Chart SH. Print Map $LNKEDT8 (Part 1 of 4); Refer to Linkage Editor, Chart 36



*A3
THE FOLLOWING FIELDS ARE CONVERTED TO HEXADECIMAL
1. TRANSFER ADDRESS-UPDATED BY THE LNG OF COMMONS.
2. PHASE ORIGIN LOAD ADDRESS-MAPPED AS LOCORE.
3. END OF PHASE LOAD ADDRESS-MAPPED AS HICORE.
4. CORE IMAGE LIBRARY DISK ADDRESS IN THE FORM CHR. C IS CONVERTED BY ENTERING THE CNVBIN SUBROUTINE AT LOCATION CNVLOP. H AND R ARE CONVERTED BY ADDING A CONVERSION FACTOR TO THE BINARY VALUE.

*D3
THE CONTROL DICTIONARY NUMBERS OF THE CSECT BEING PROCESSED AND THE CSECT POINTED TO BY THE LD/LR ARE COMPARED. IF AN EQUAL CONDITION IS FOUND, THE LD/LR BELONGS TO THE CSECT BEING PROCESSED.

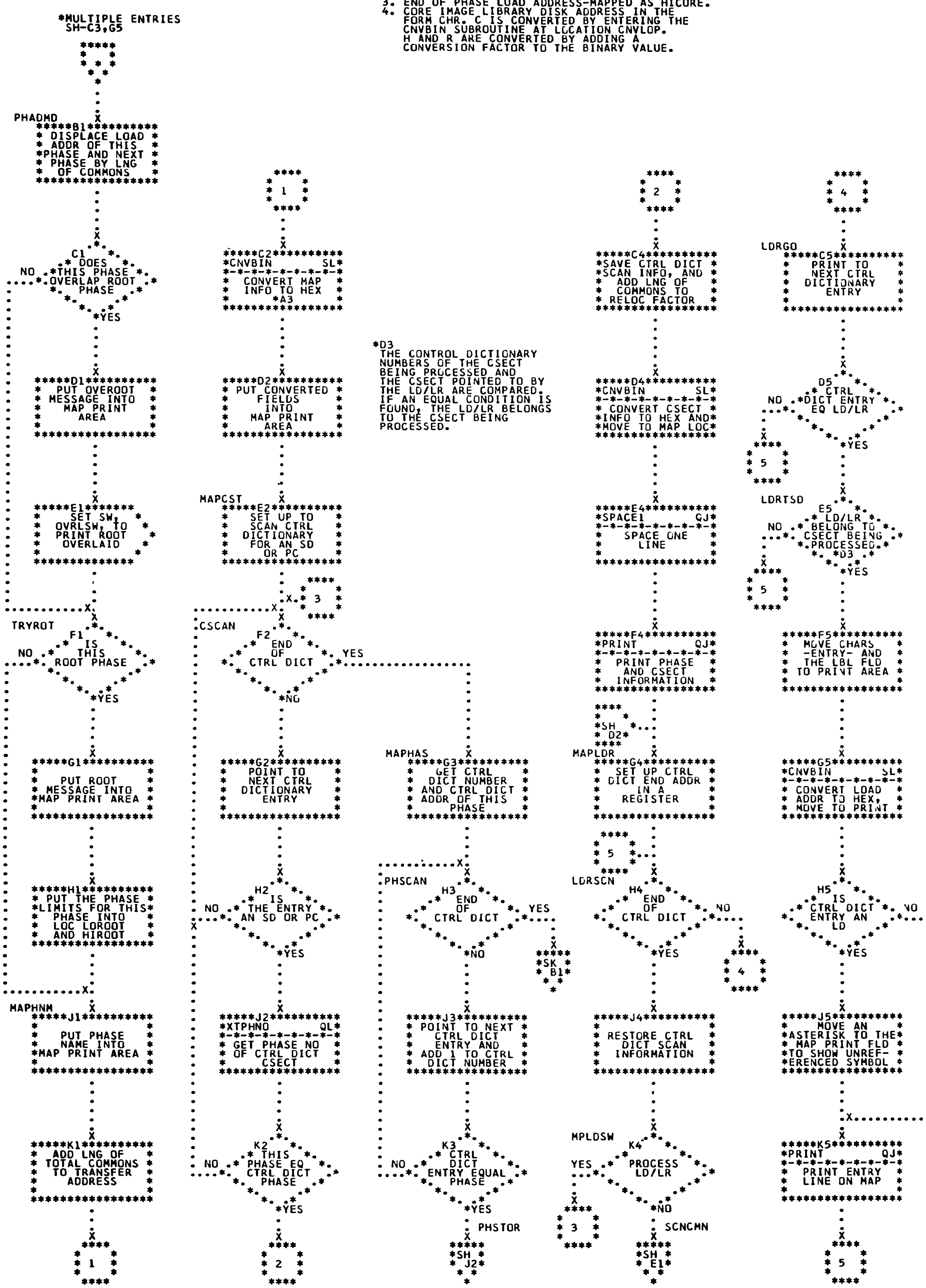


Chart SK. Print Map $LNKEDT8 (Part 3 of 4); Refer to Linkage Editor, Chart 36

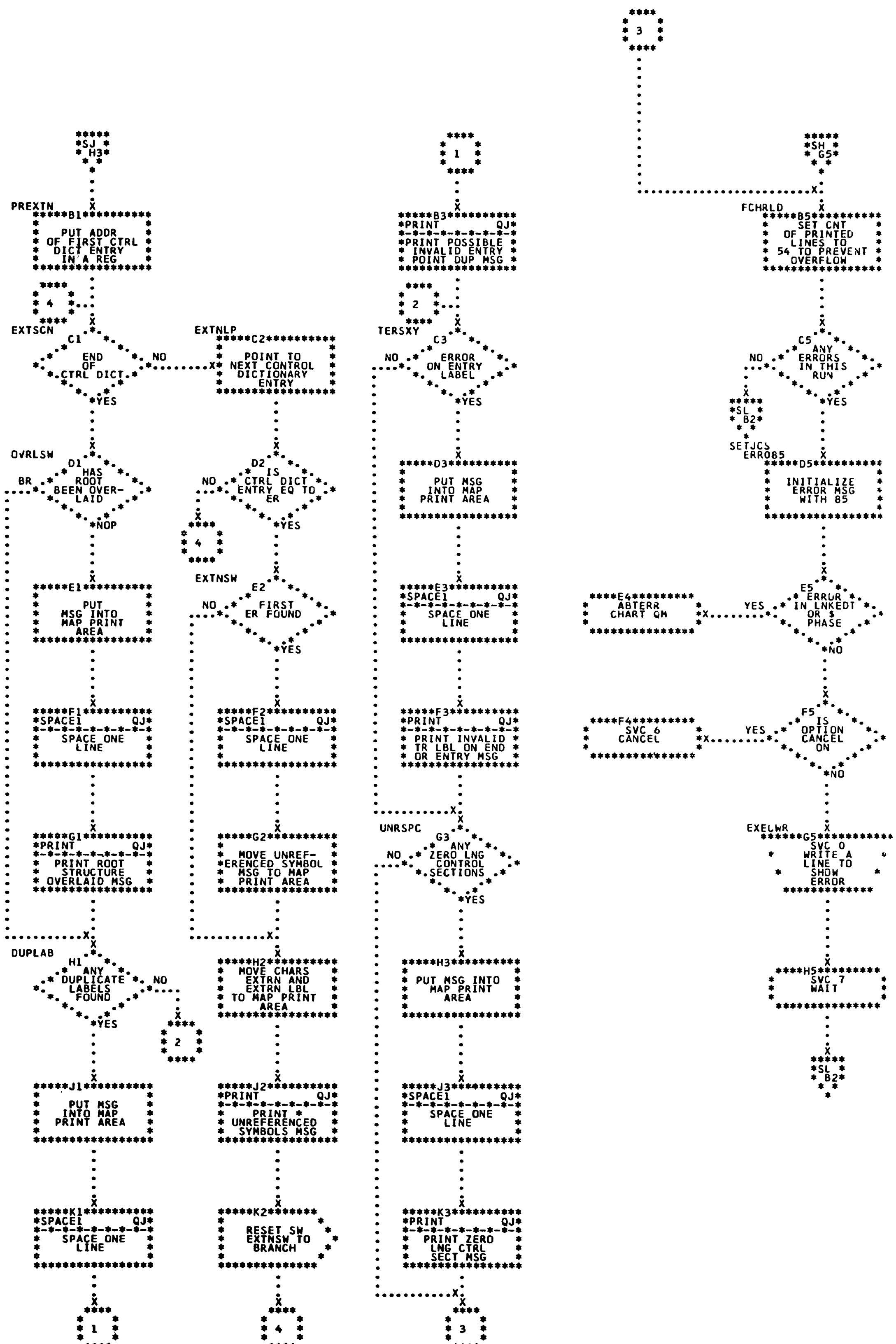


Chart SL. Print Map $LNKEDT8 (Part 4 of 4); Refer to Linkage Editor, Chart 36

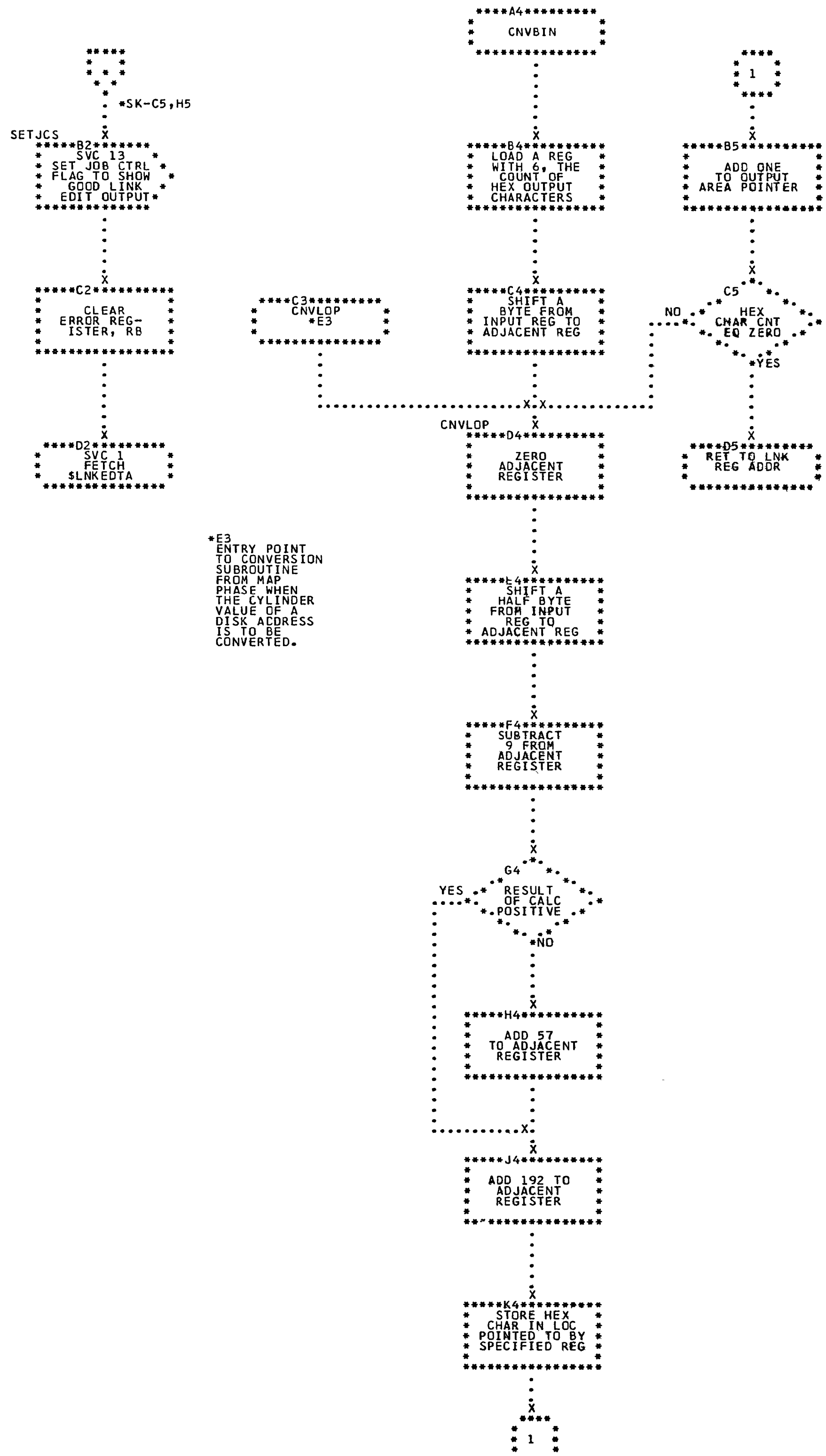


Chart SM. Pass 2 P-Pointer Processor $LNKEDTA; Refer to Linkage Editor, Chart 37

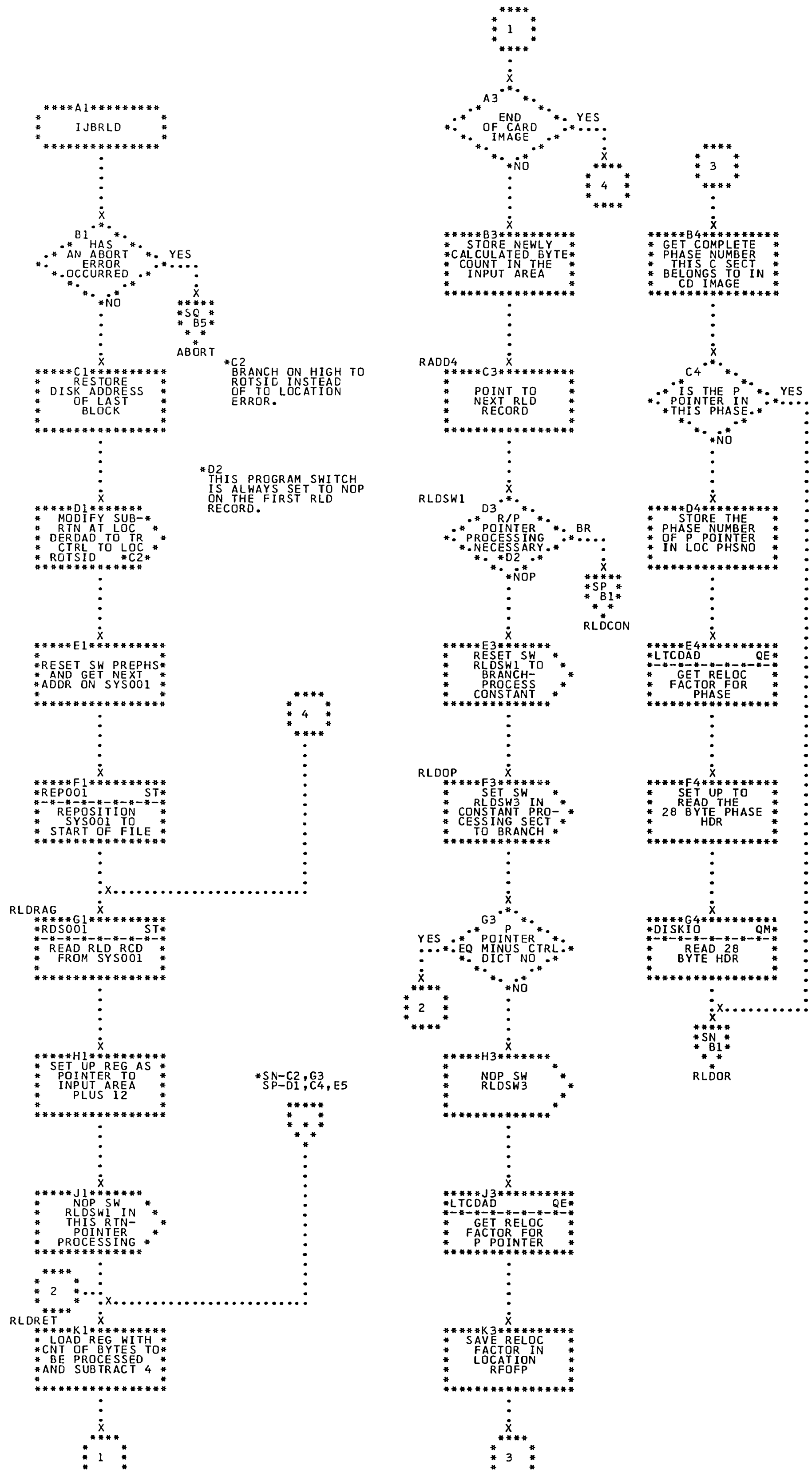
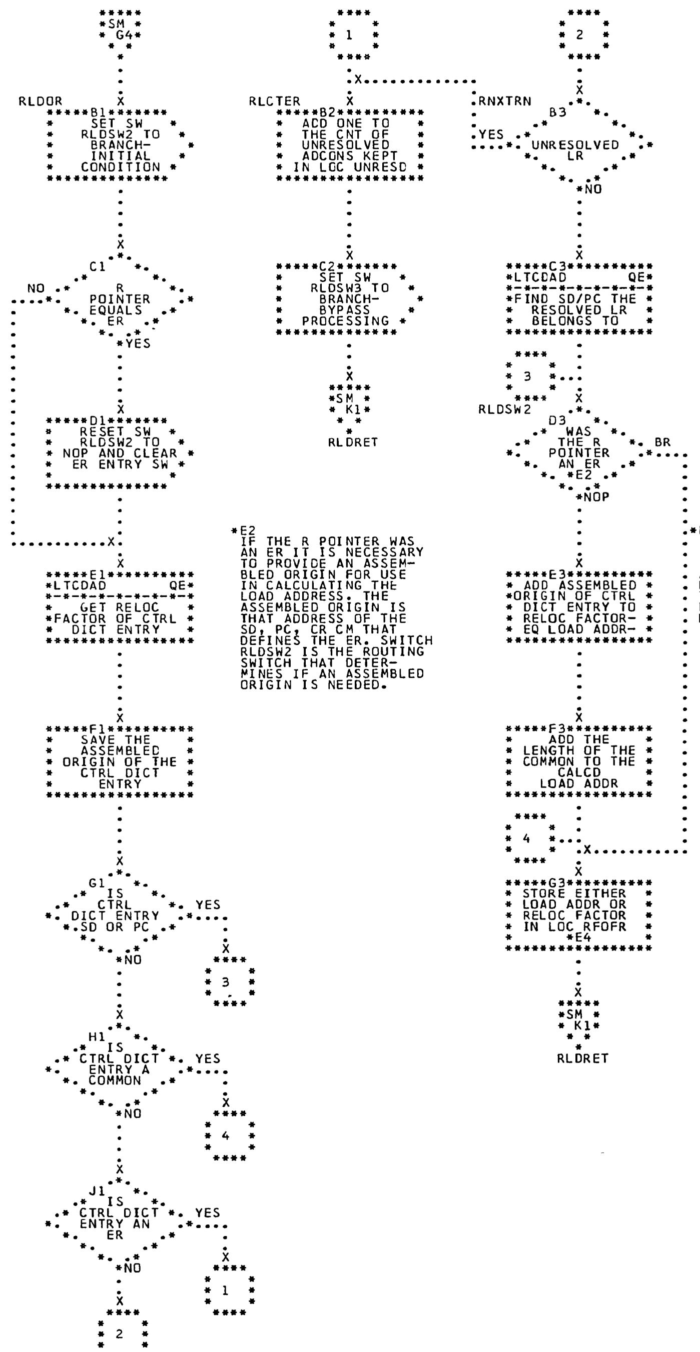


Chart SN. Pass 2 R-Pointer Processor $LNKEDTA; Refer to Linkage Editor, Chart 37




*E2 IF THE R POINTER WAS AN ER IT IS NECESSARY TO PROVIDE AN ASSEMBLED ORIGIN FOR USE IN CALCULATING THE LOAD ADDRESS. THE ASSEMBLED ORIGIN IS THAT ADDRESS OF THE SD, PC, CR CM THAT DEFINES THE ER. SWITCH RLDSW2 IS THE ROUTING SWITCH THAT DETERMINES IF AN ASSEMBLED ORIGIN IS NEEDED.

*E4 IF THE R POINTER WAS AN ER, THE CONTENT OF RFOFR IS THE SUM OF THE ASSEMBLED ORIGIN AND THE RELOCATION FACTOR. IF THE R POINTER WAS A NON ER, ONLY THE VALUE OF THE RELOCATION FACTOR IS STORED IN LOCATION RFOFR.

Chart SP. Pass 2 RLD Constant Processor $LNKEDTA; Refer to Linkage Editor, Chart 37

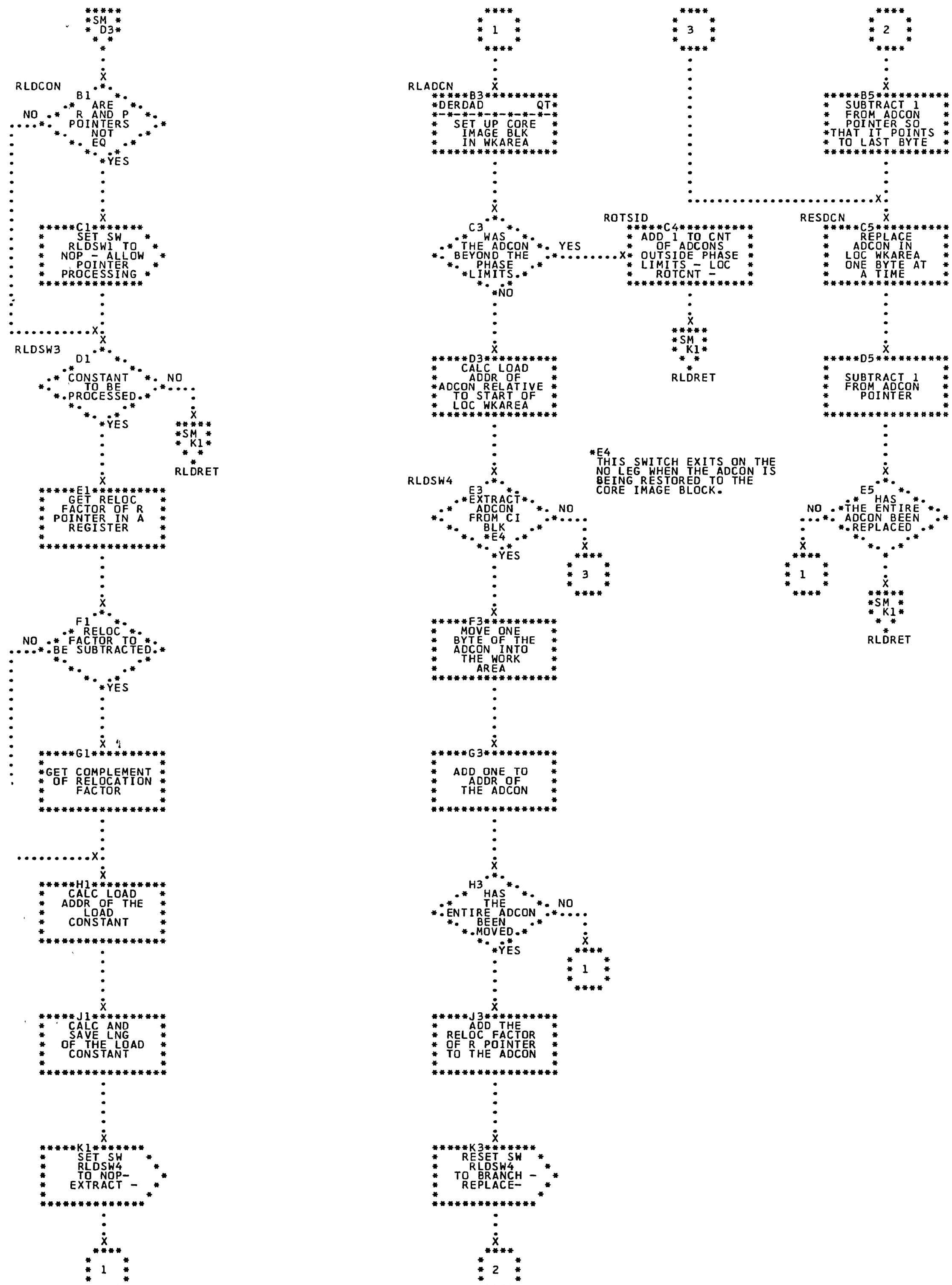


Chart SQ. Pass 2 ABORT and MAP Routines $LNDEDTA; Refer to Linkage Editor, Chart 37

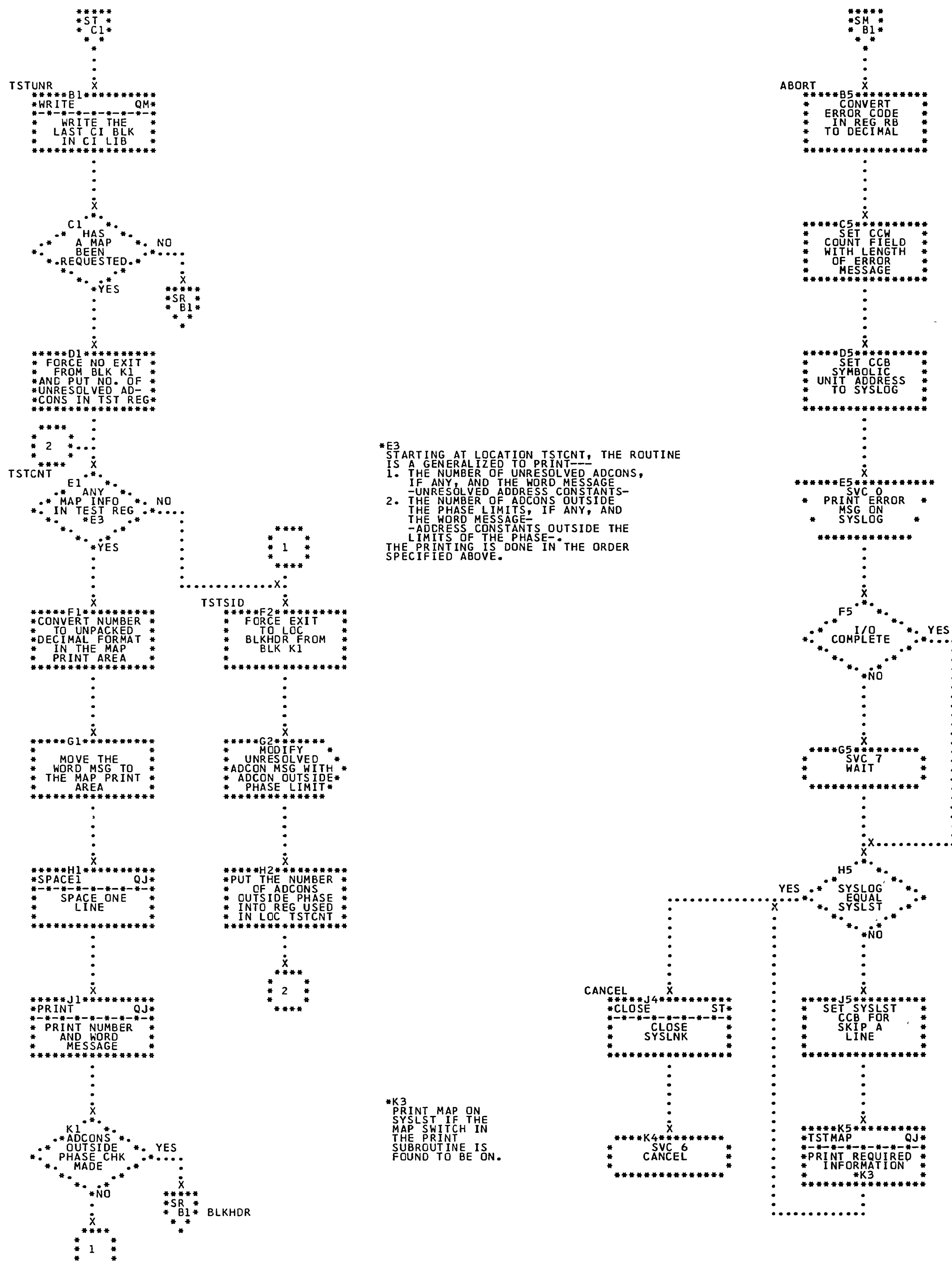


Chart SR. Pass 2, Block Phase Header $LNKEDTA; Refer to Linkage Editor, Chart 37

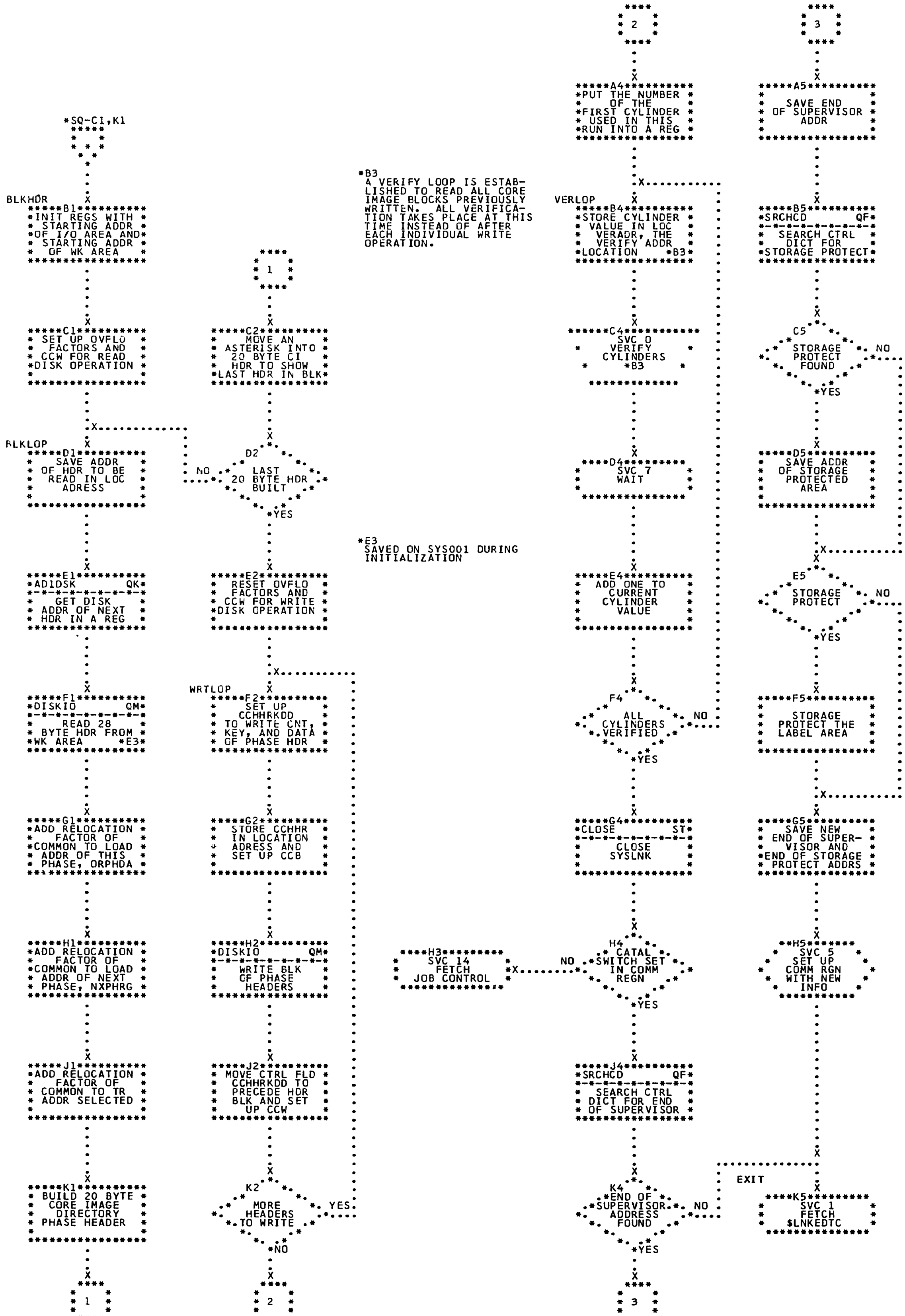


Chart ST. Pass 2 Subroutines $LNKEDTA; Refer to Linkage Editor Chart 37

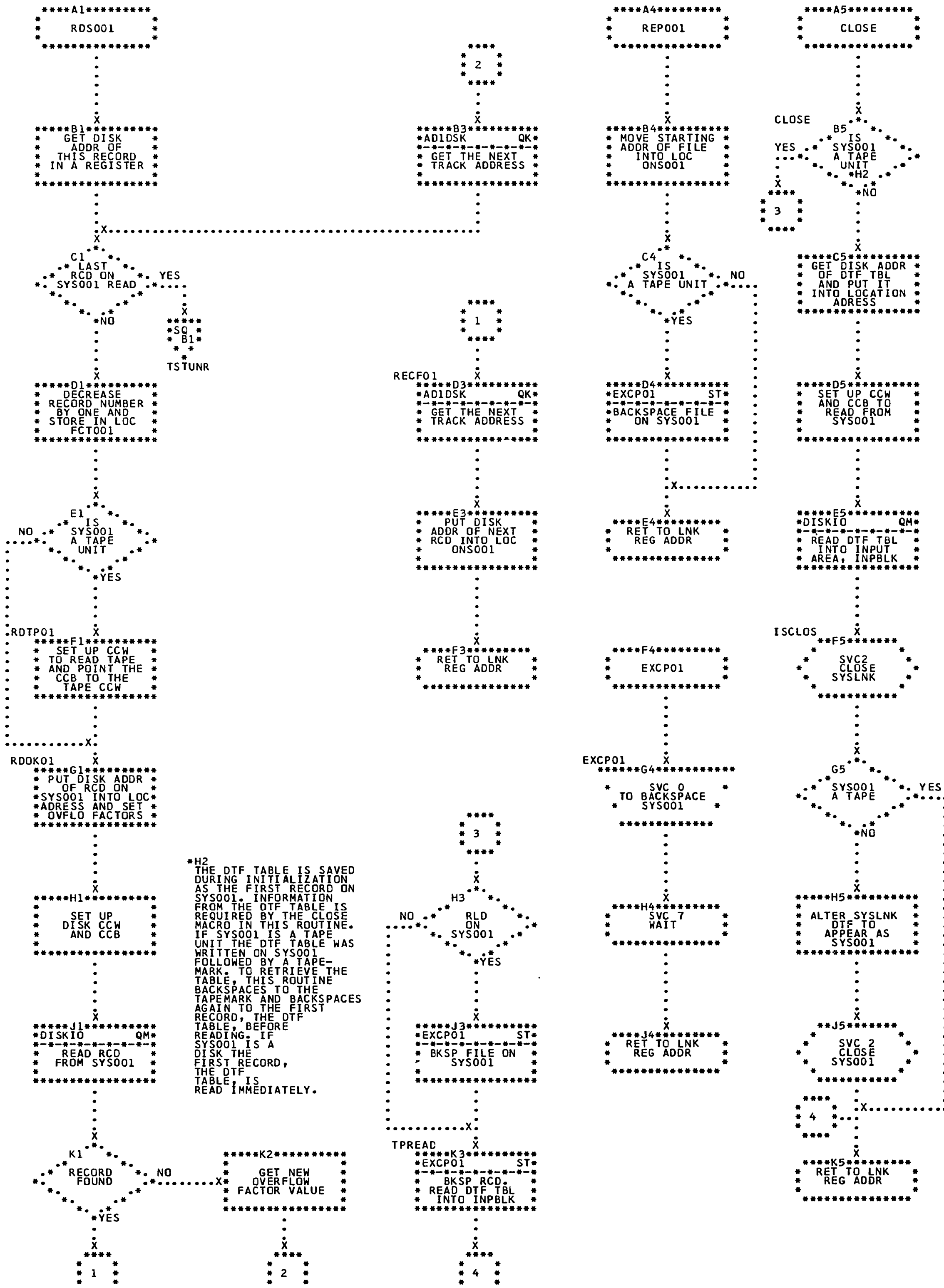


Chart SU. Determine If Phases to be Cataloged Fit in Core Image Directory $LNKEDTC; Refer to Linkage Editor, Chart 38

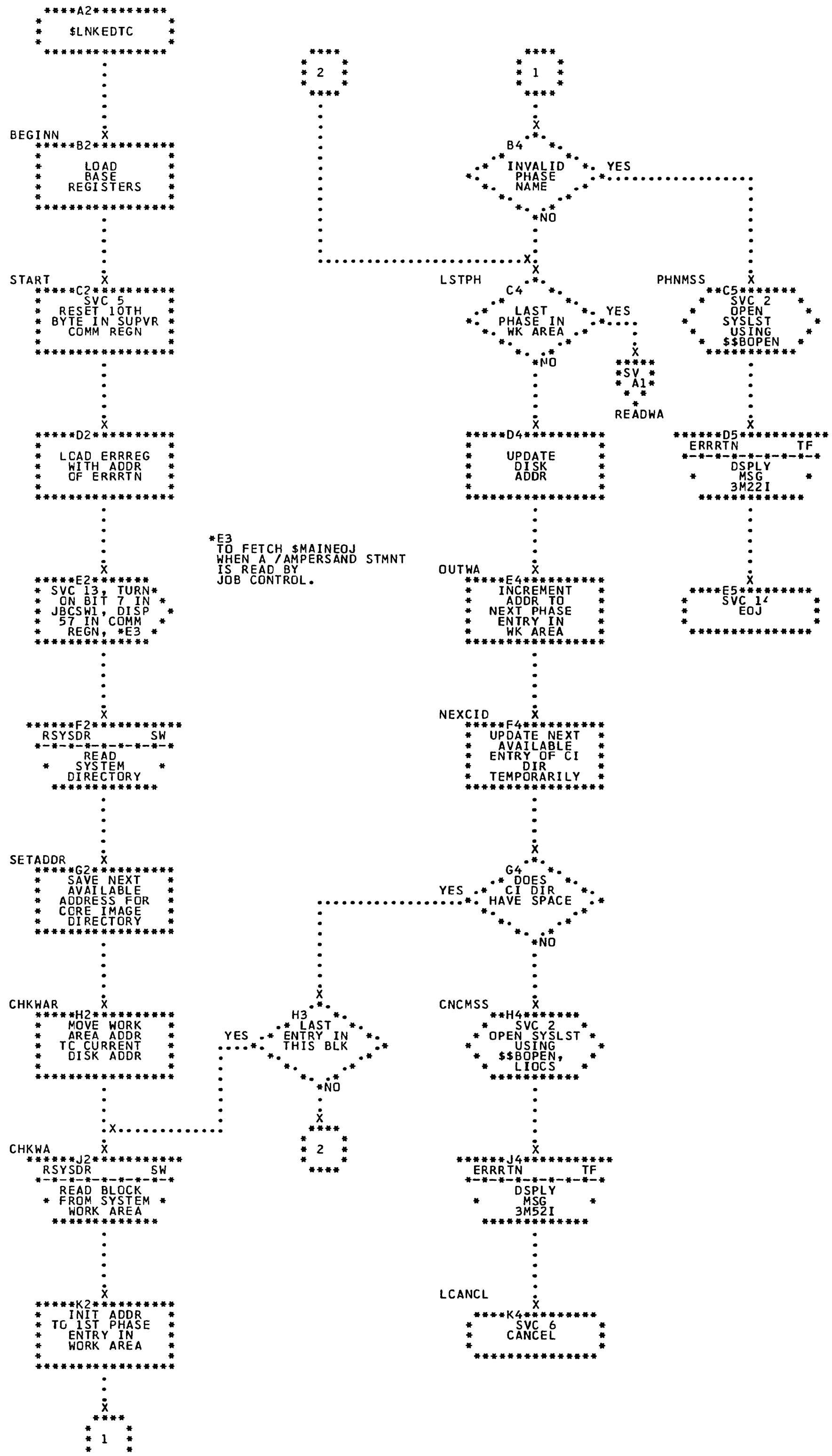


Chart SV. Check Core Image Directory for Entries Being Replaced $LNKEDTC; Refer to Linkage Editor, Chart 38

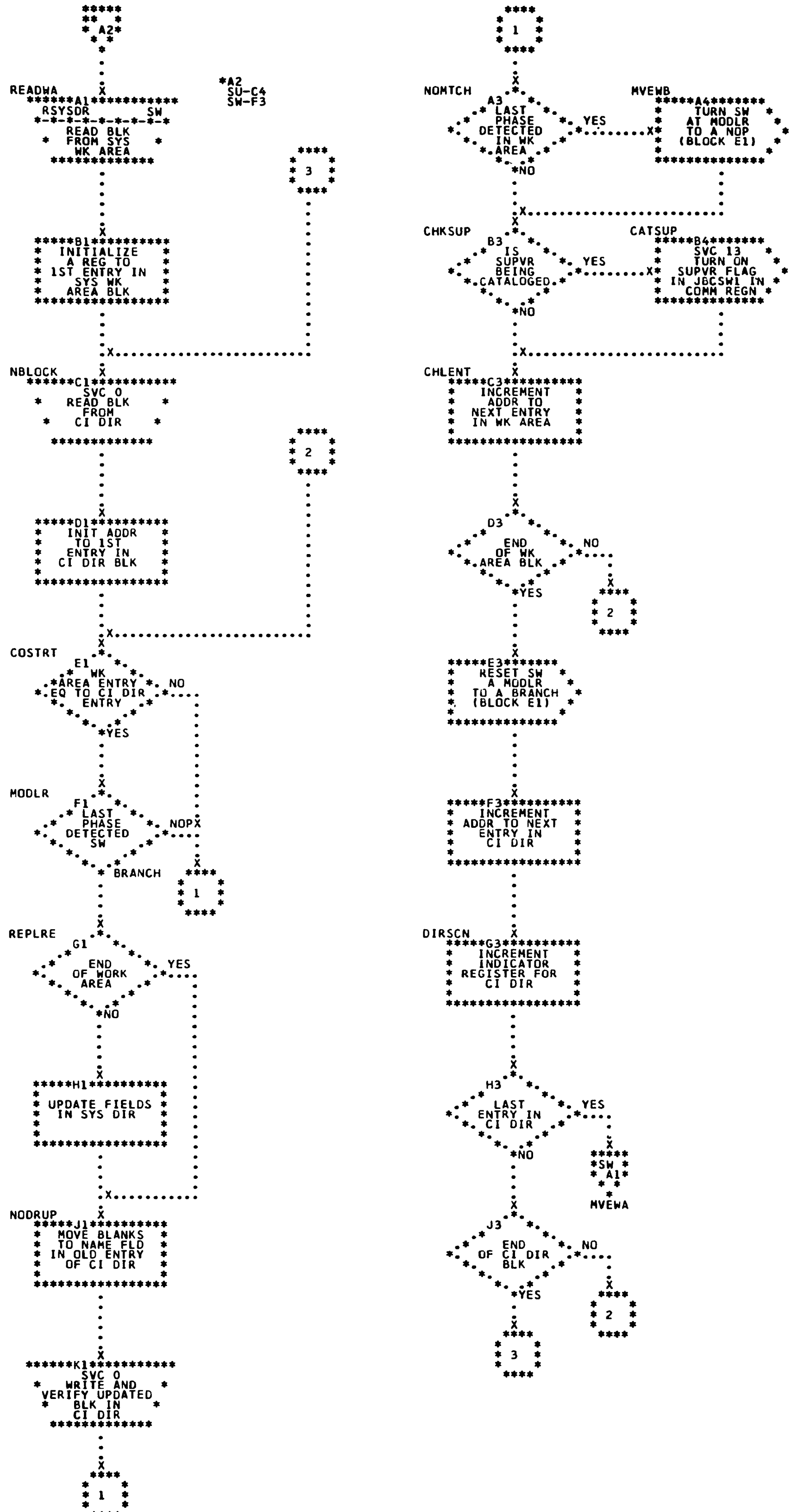


Chart SW. Catalog Phase Entries to Core Image Directory
$LNKEDTC; Refer to Linkage Editor, Chart 38

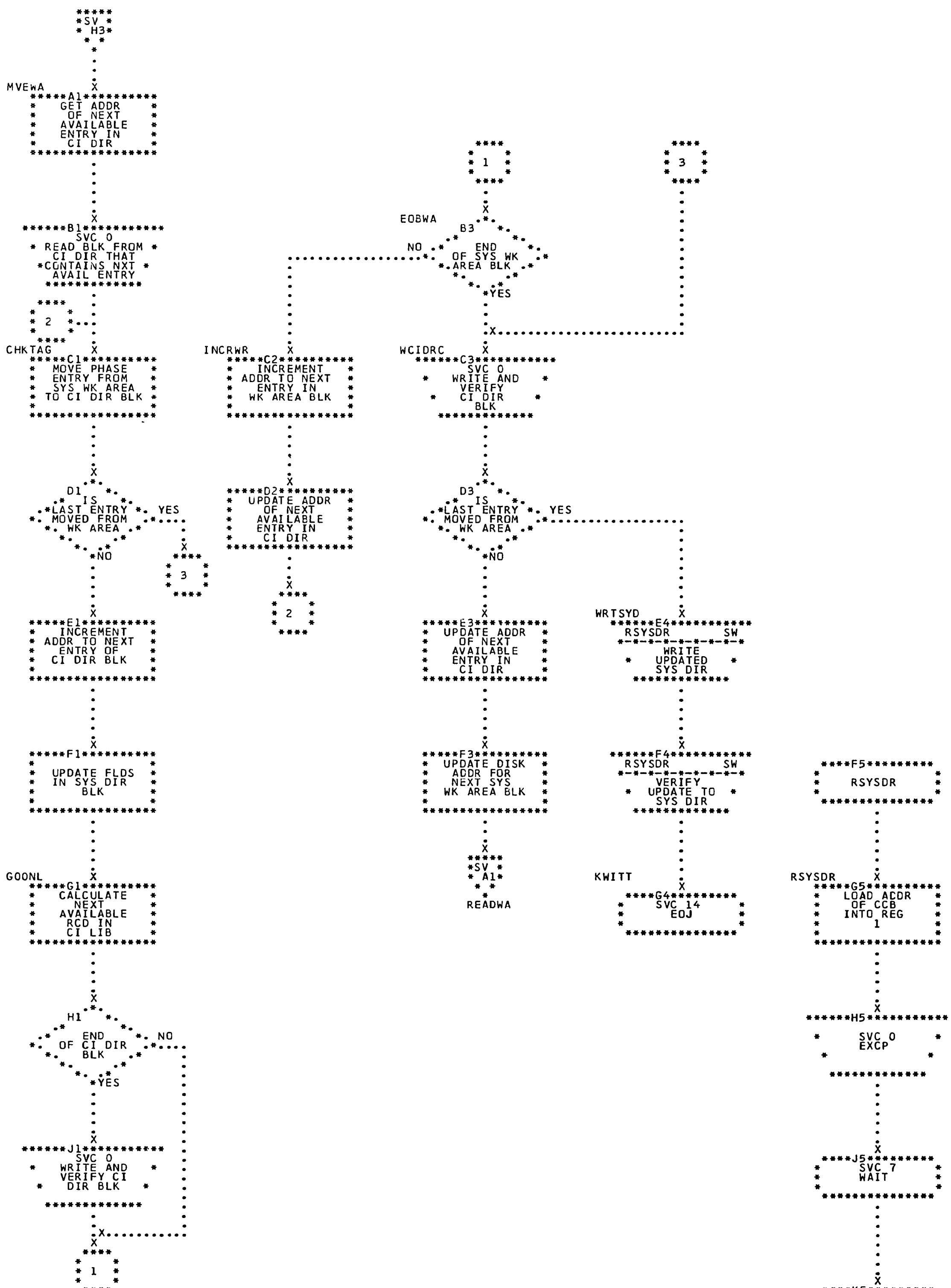


Chart TA. Read Control Statements MAINT; Refer to Maintenance, Chart 39

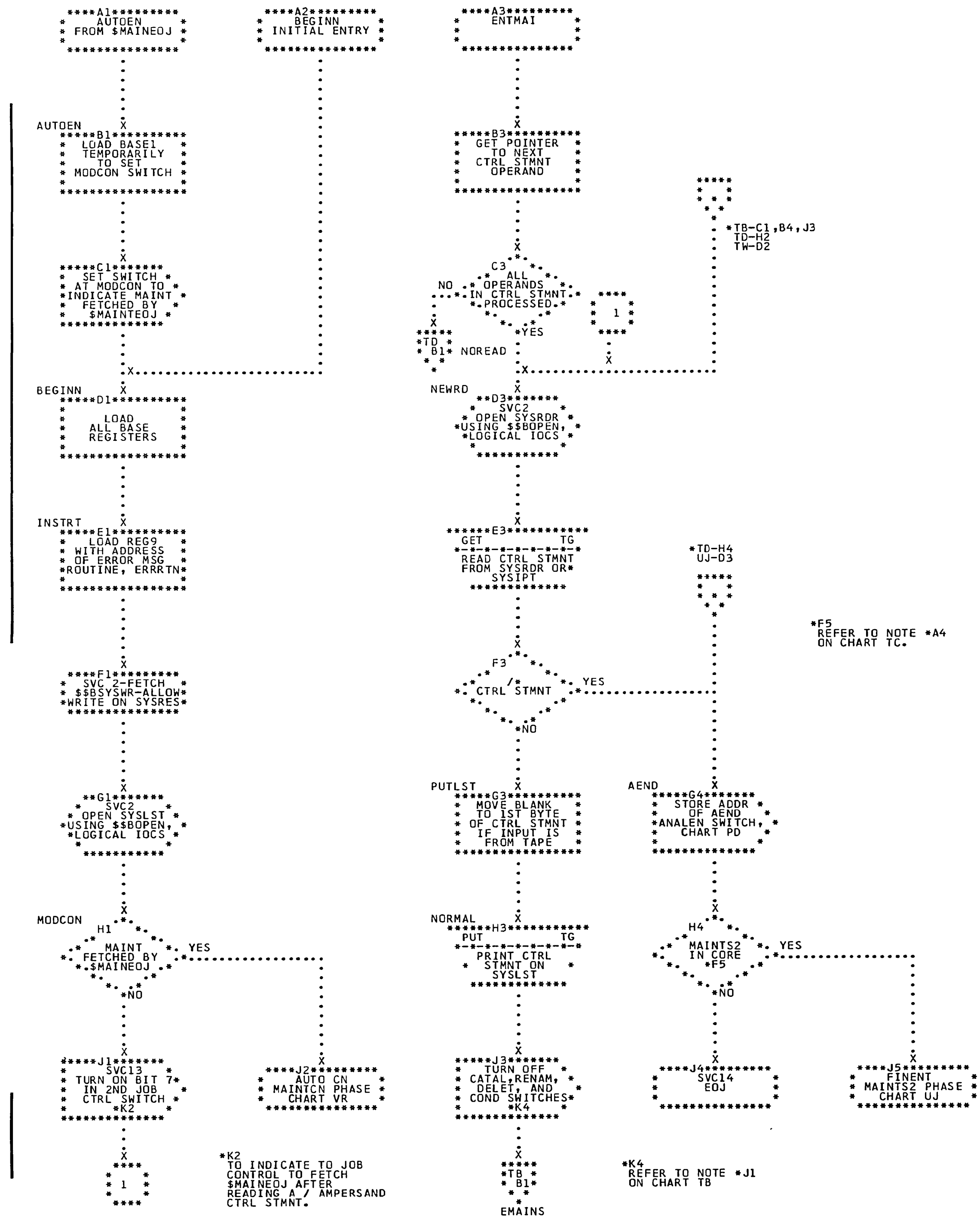


Chart TB. Analyze Control Statements MAINT; Refer to Maintenance, Chart 39

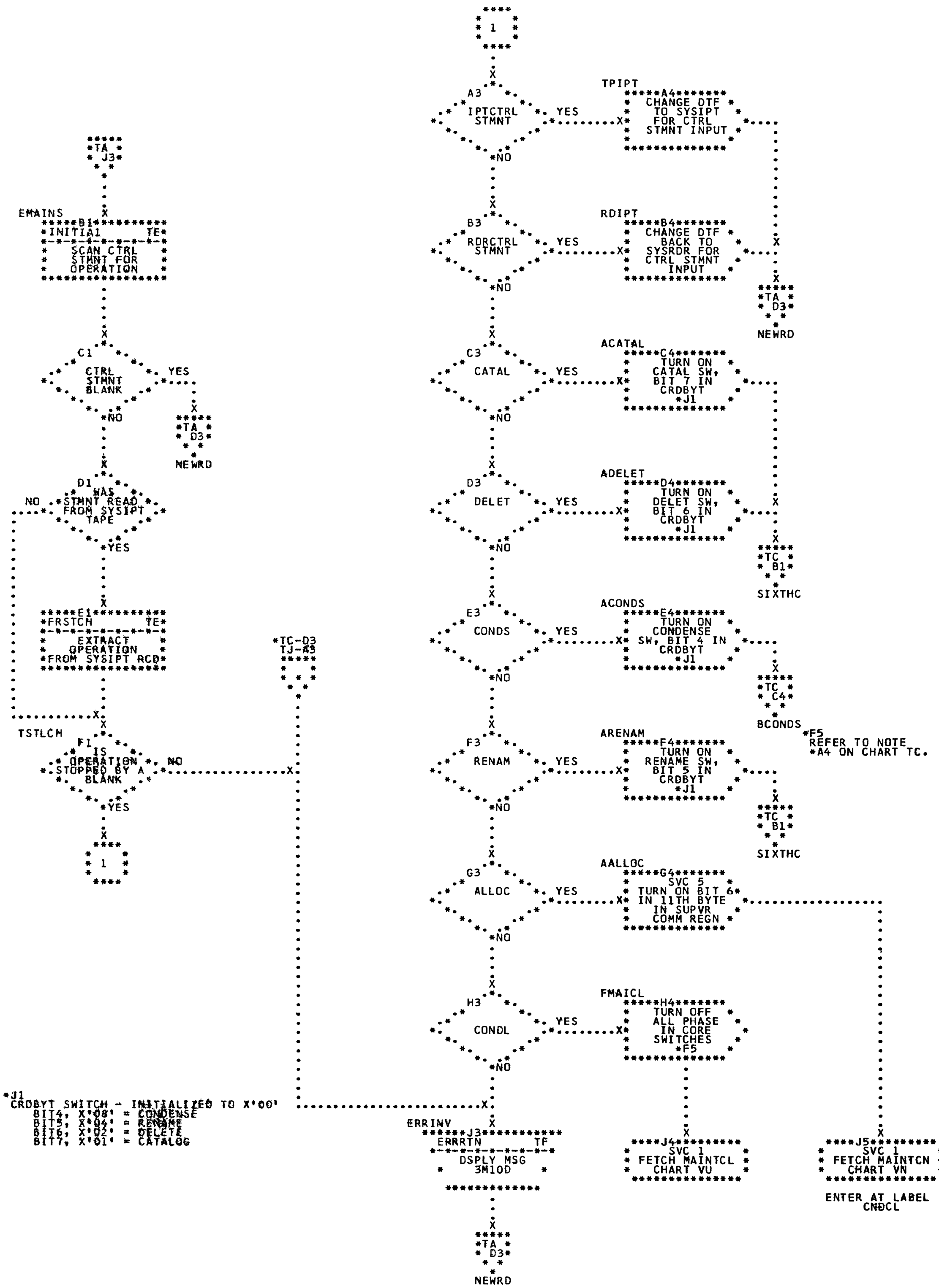


Chart TC. Load Phases MAINT; Refer to Maintenance, Chart 39

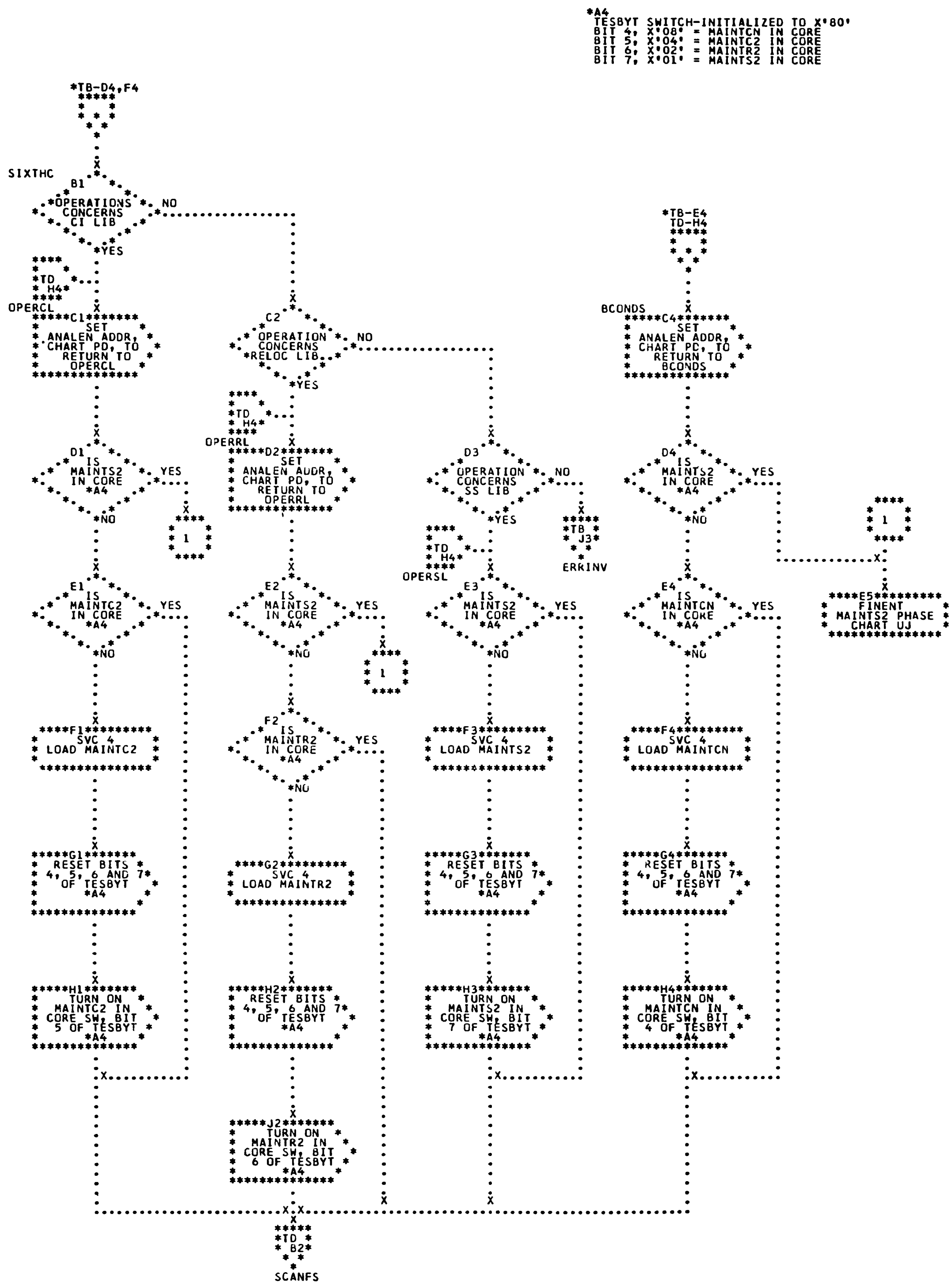


Chart TD. Branch to Phases MAINT; Refer to Maintenance, Chart 39

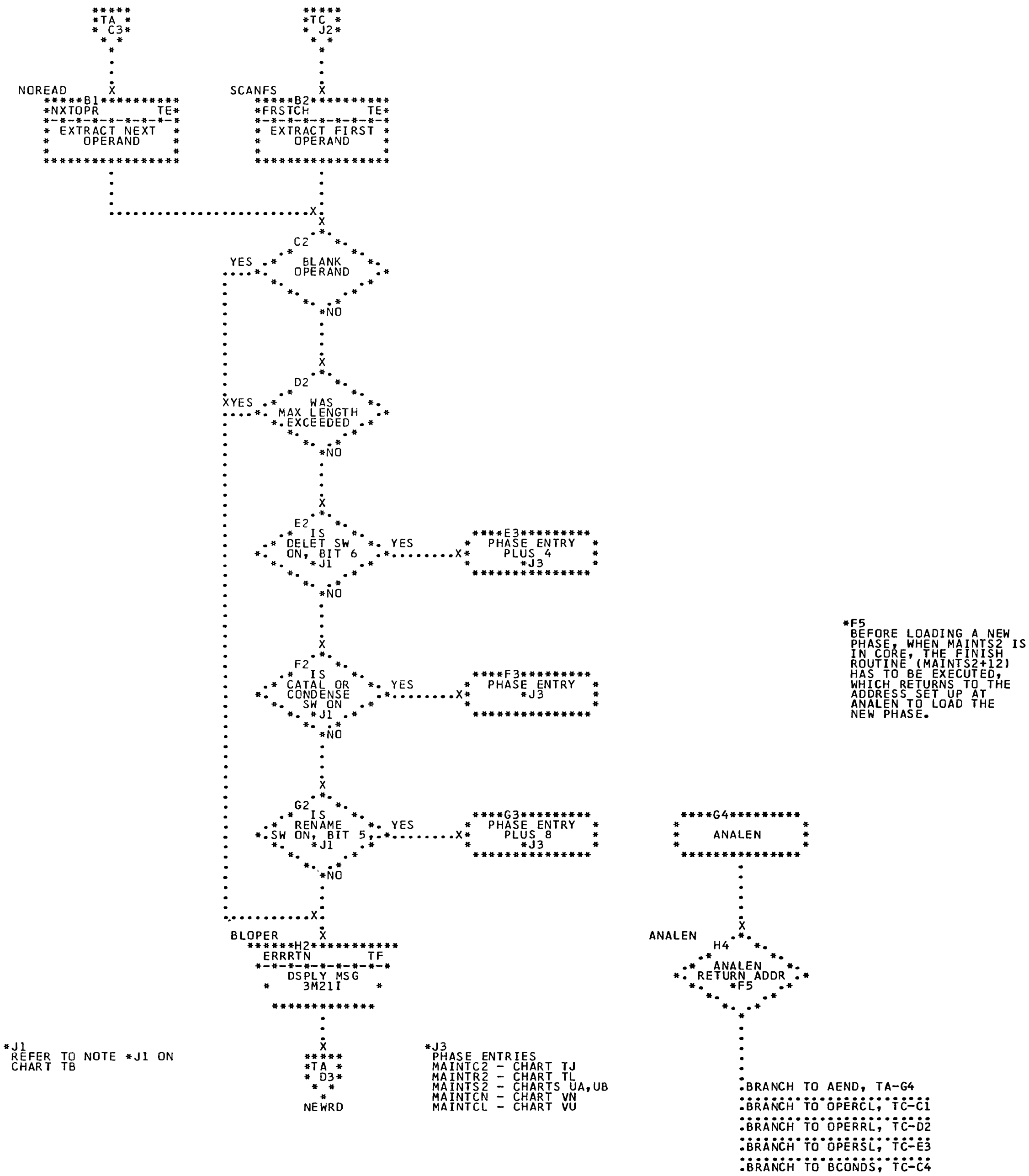


Chart TE. Scan Control Statements MAINT; Refer to Maintenance, Chart 39

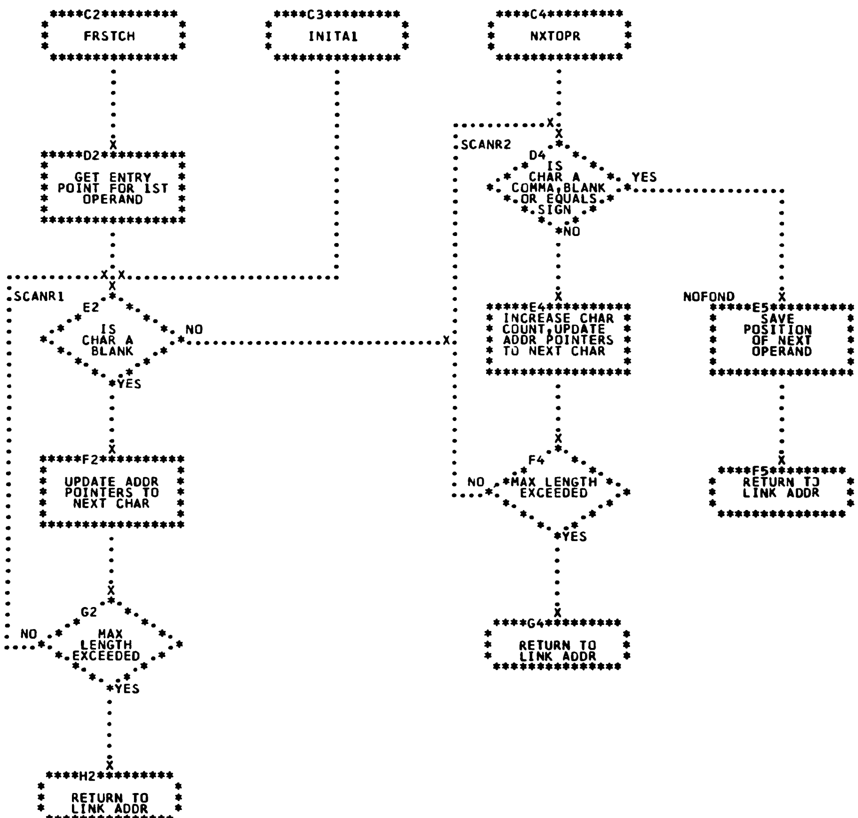


Chart TF. Common Error Message Routine MAINT; Refer to Maintenance, Chart 39

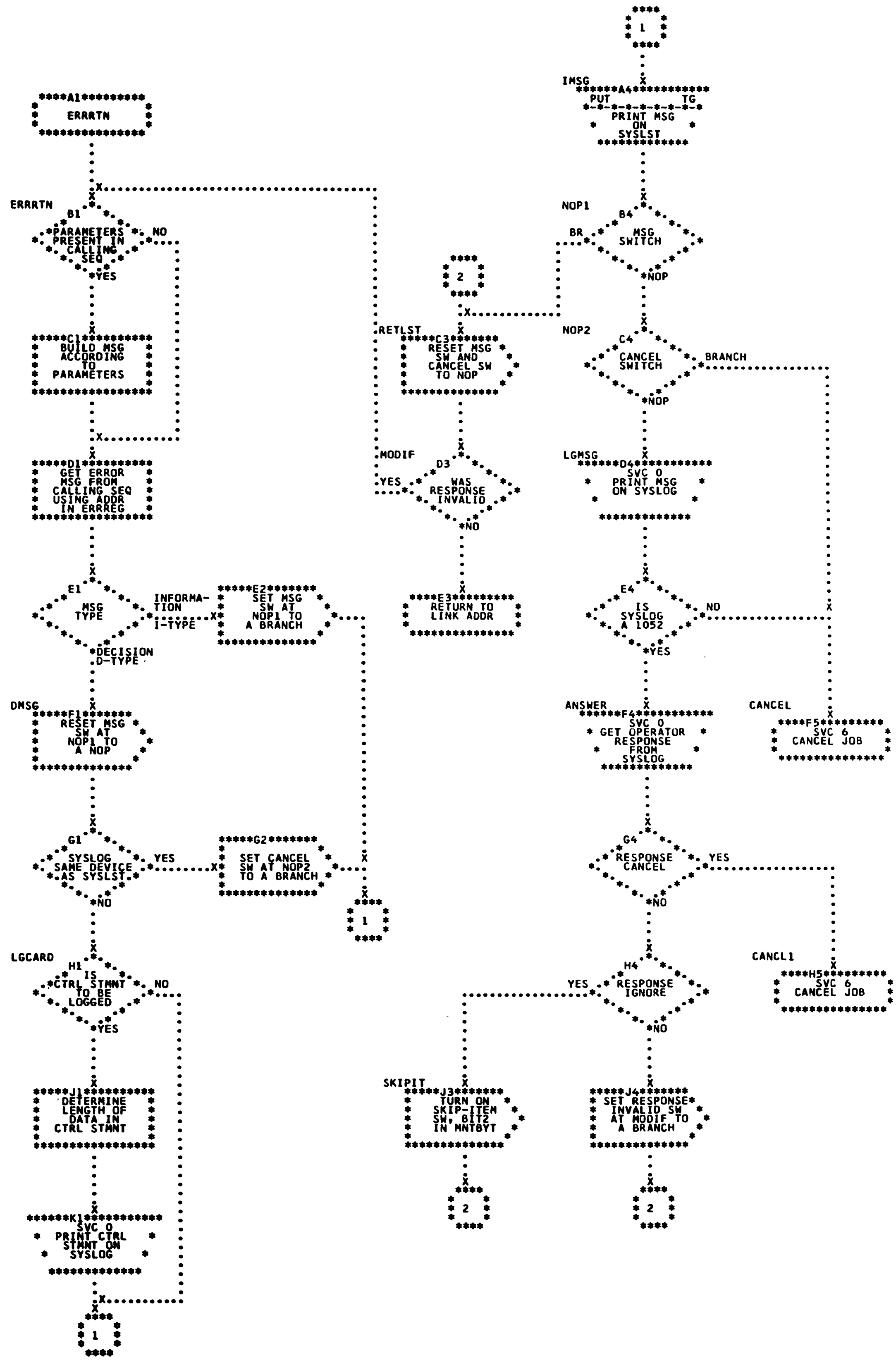


Chart TG. Common IOCS I/O Routine MAINT (Part 1 of 2); Refer to Maintenance, Chart 39

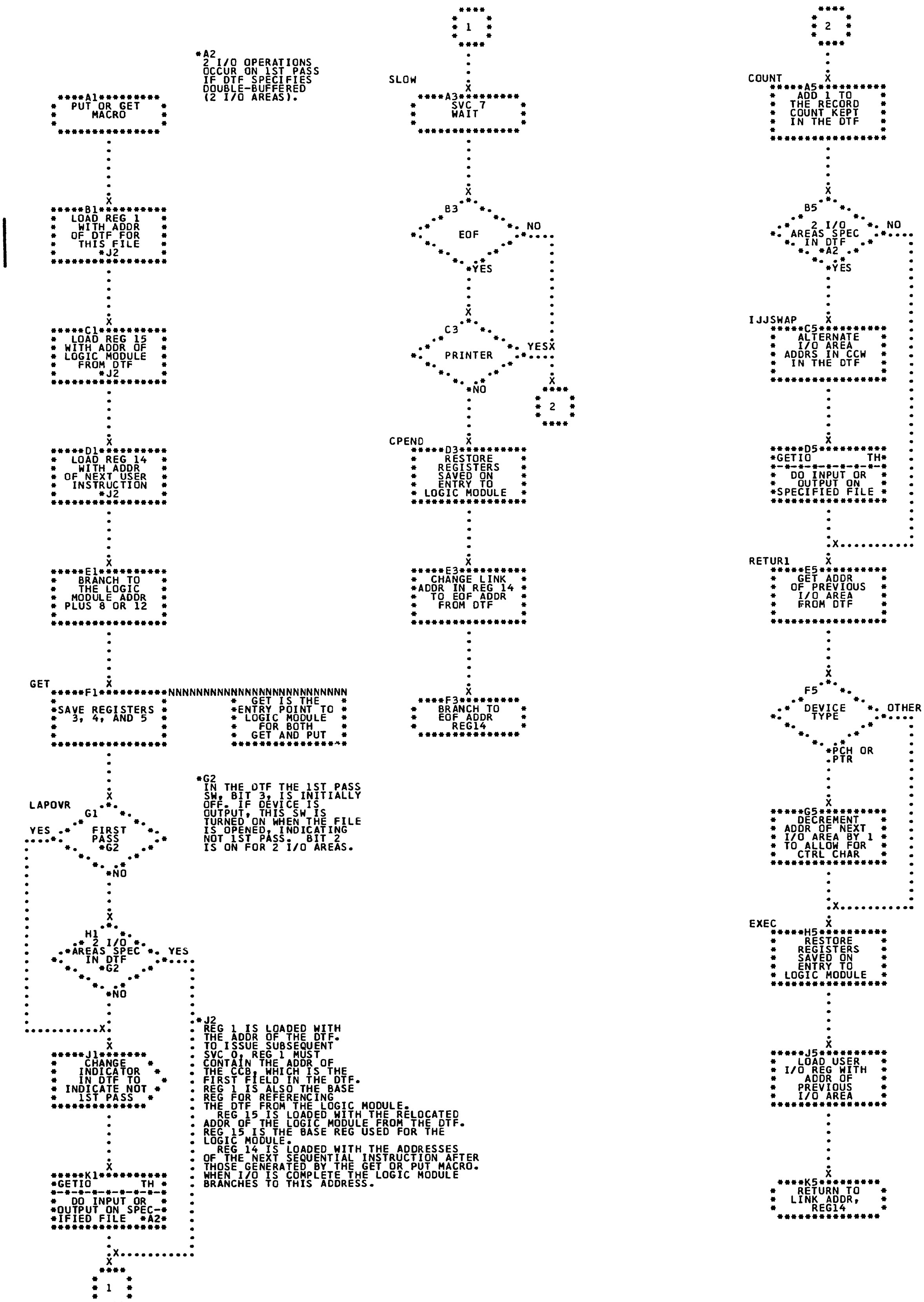


Chart TH. Common IOCS I/O Routine MAINT (Part 2 of 2); Refer to Maintenance, Chart 39

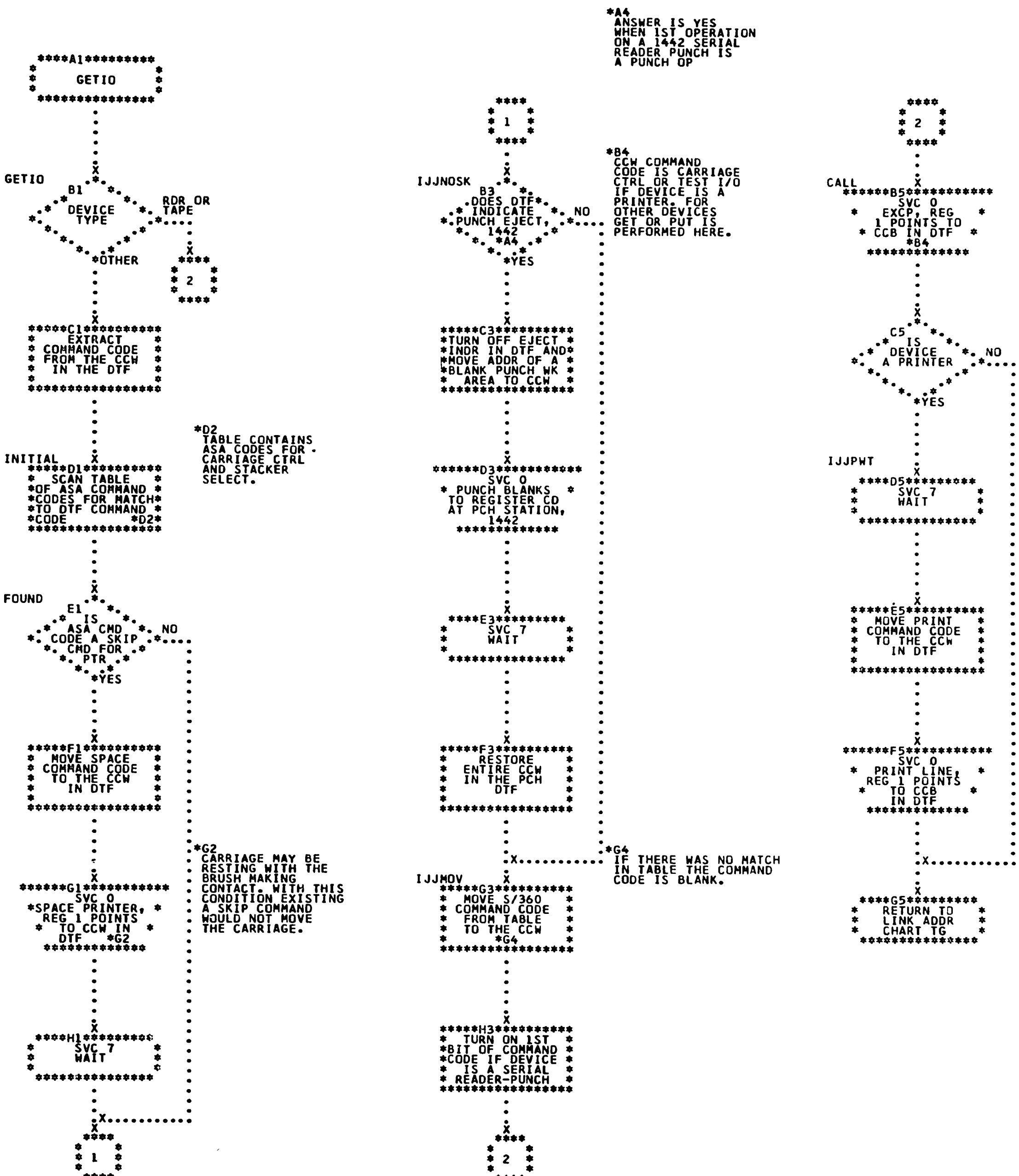


Chart TJ. Core Image Library Maintenance MAINTC2; Refer to Maintenance, Chart 40

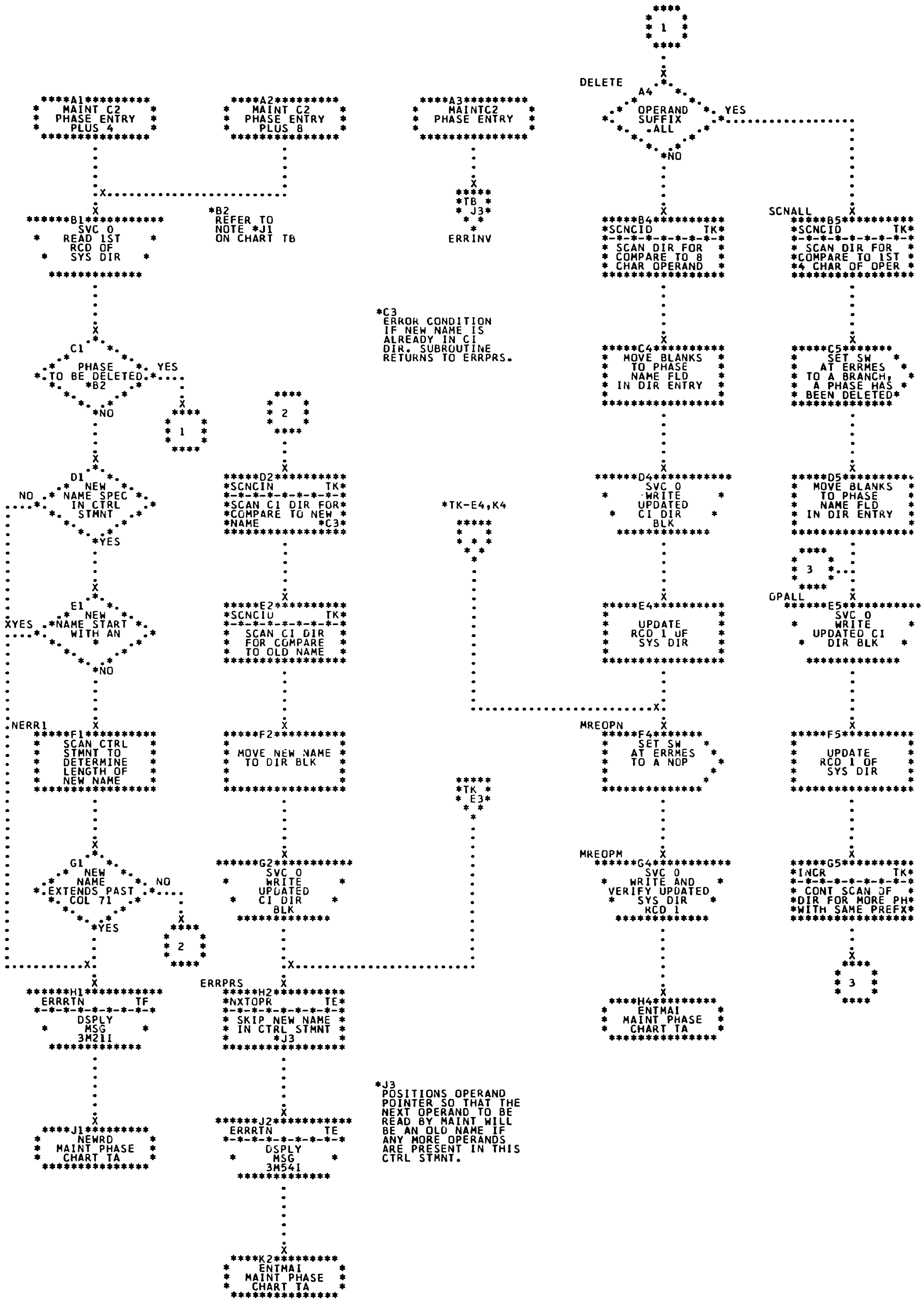


Chart TK. Scan Core Image Directory MAINTC2; Refer to Maintenance, Chart 40

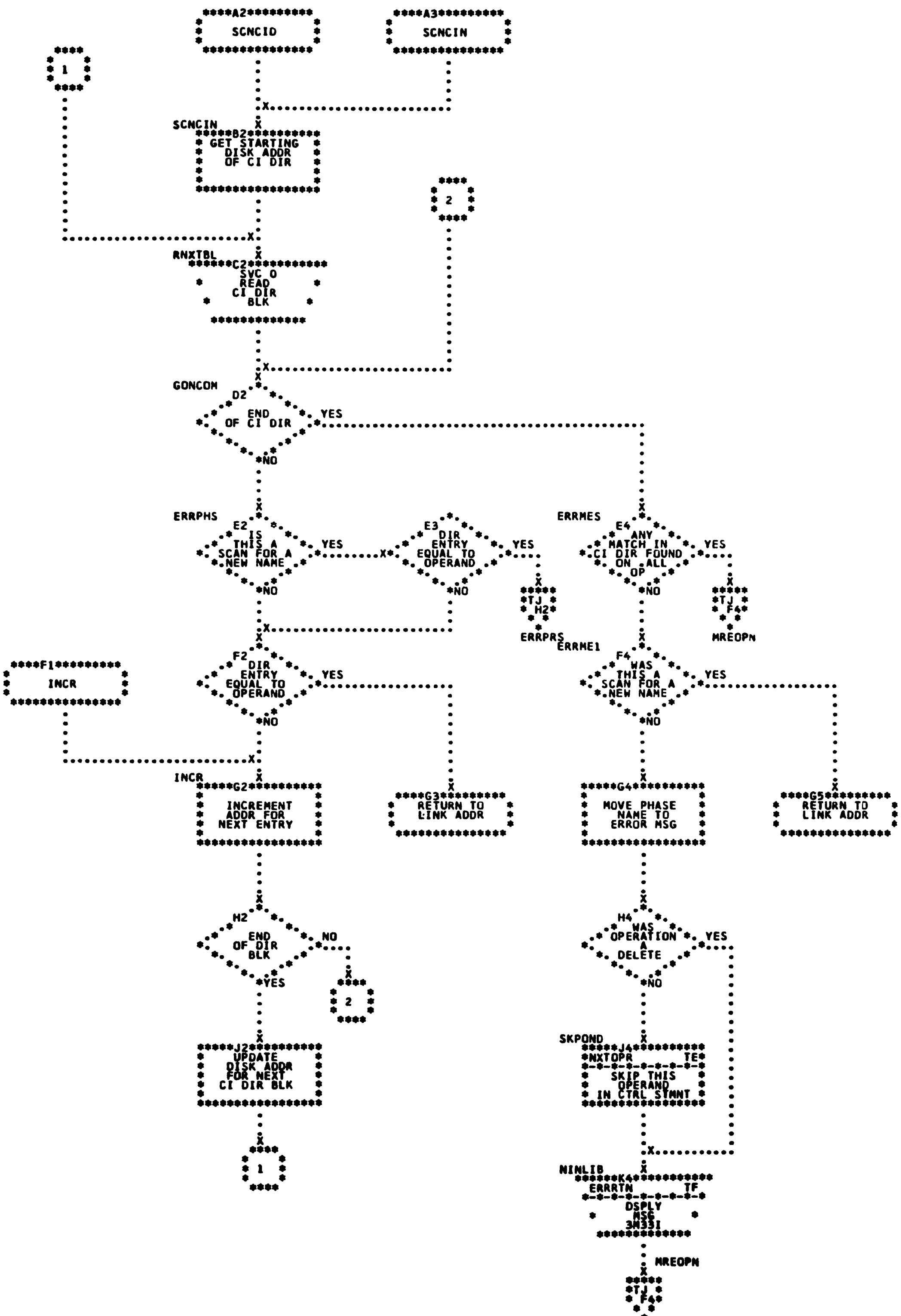


Chart TL. Initialize for Relocatable Library Maintenance
MAINTR2; Refer to Maintenance, Chart 41

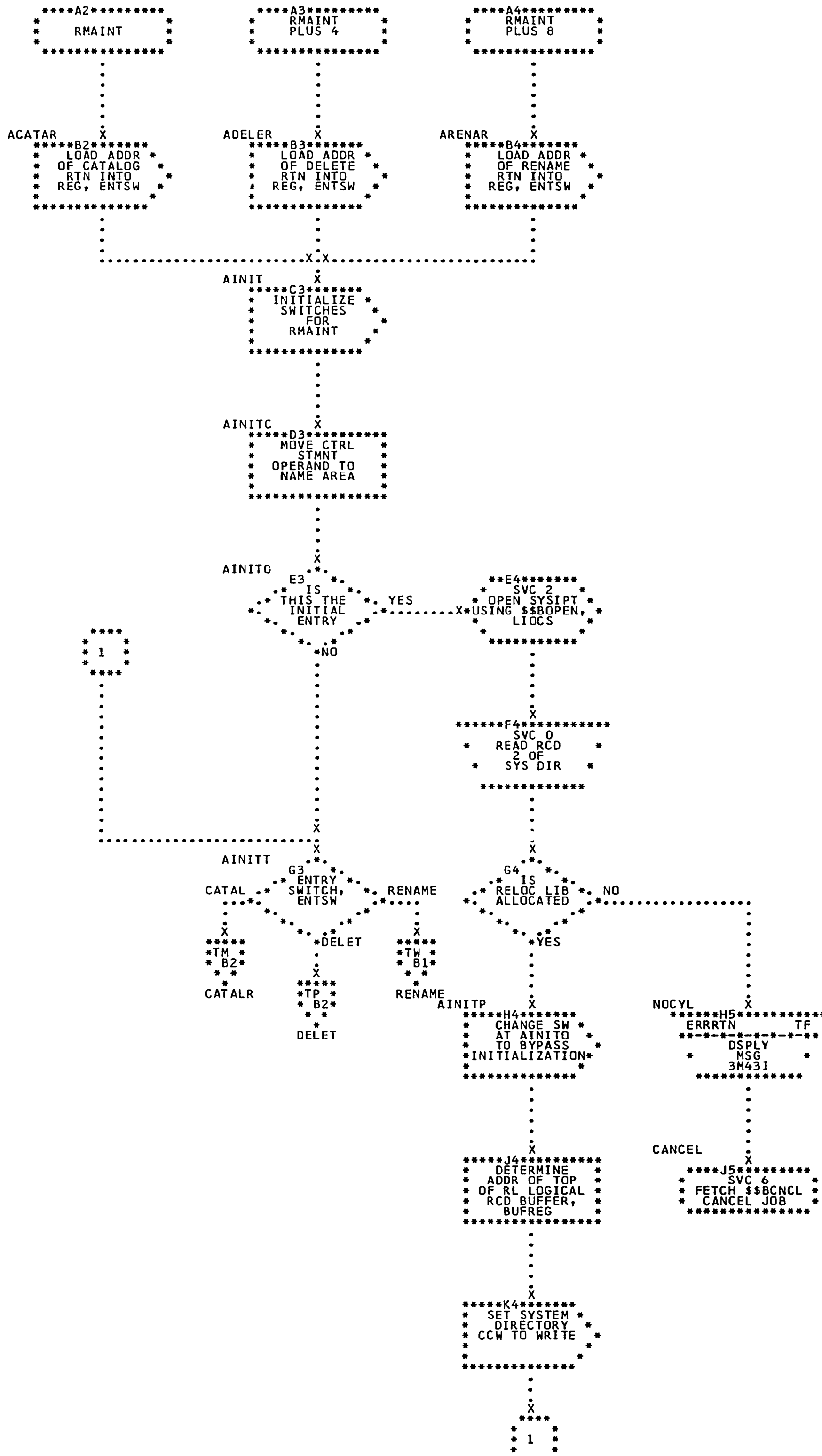


Chart TM. Catalog Relocatable Library MAINTR2 (Part 1 of 2); Refer to Maintenance, Chart 41

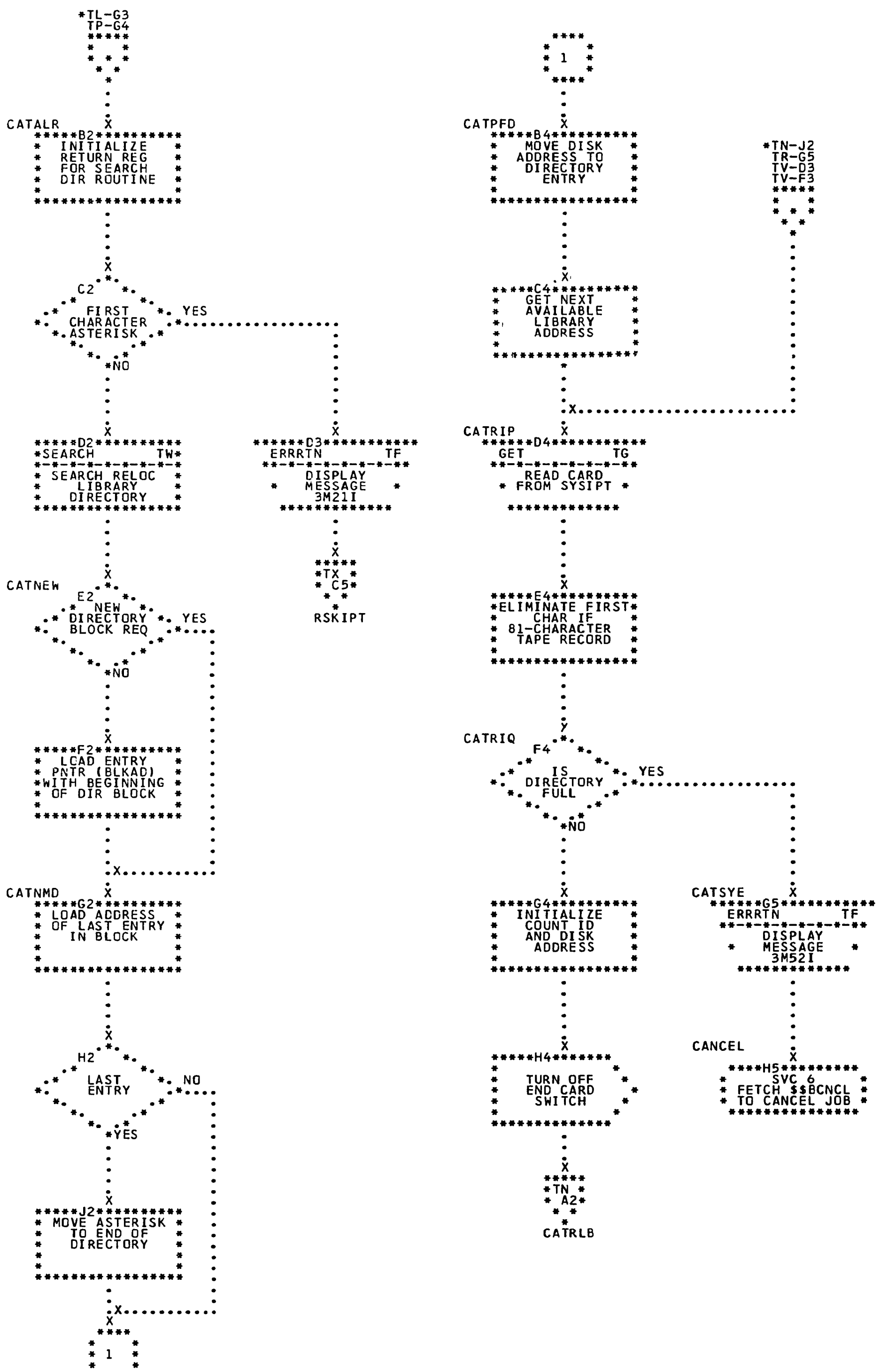


Chart TN. Catalog Relocatable Library MAINTR2 (Part 2 of 2); Refer to Maintenance, Chart 41

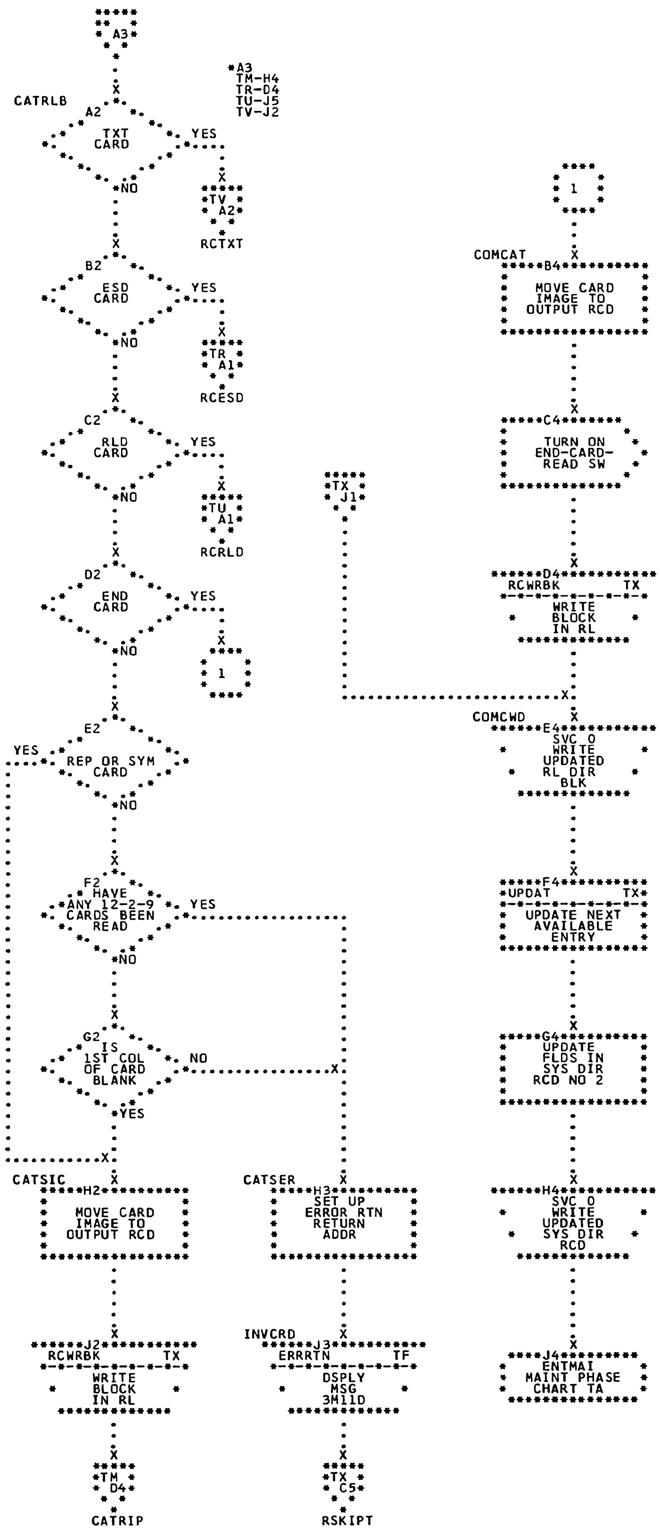


Chart TP. Delete from Relocatable Library MAINTR2 (Part 1 of 2); Refer to Maintenance, Chart 41

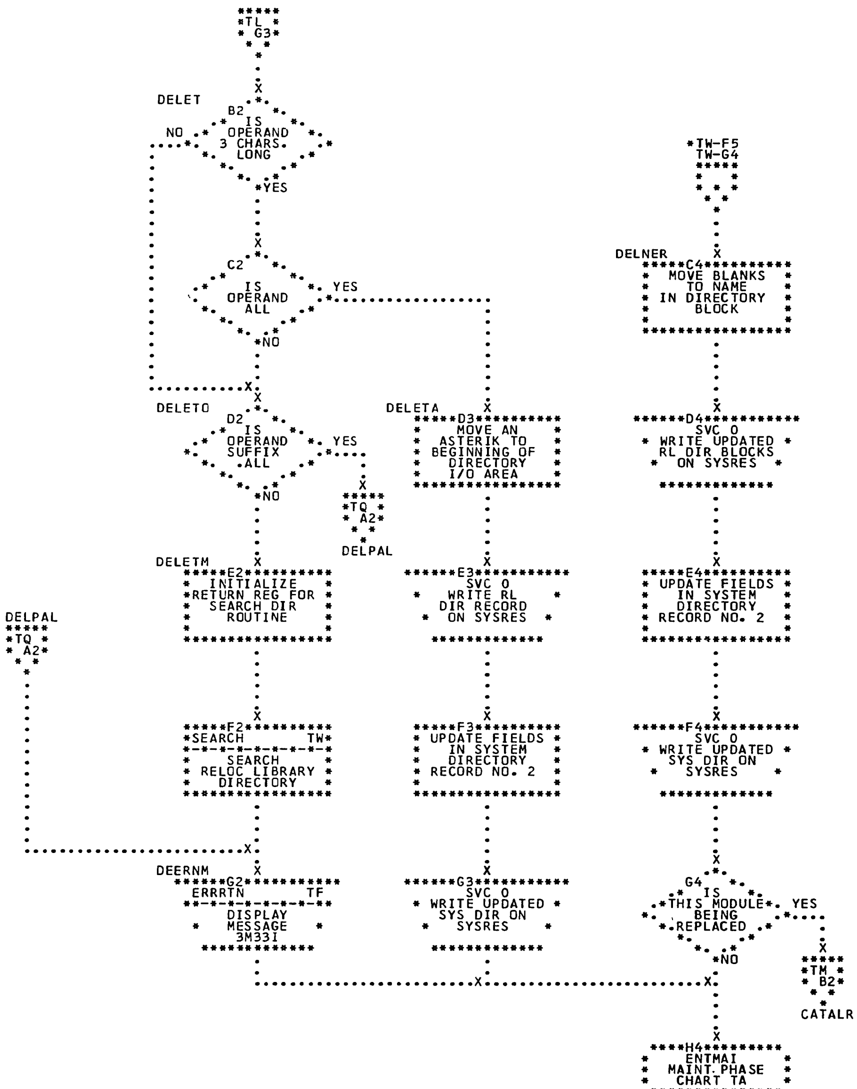


Chart TQ. Delete from Relocatable Library MAINTR2 (Part 2 of 2); Refer to Maintenance, Chart 41

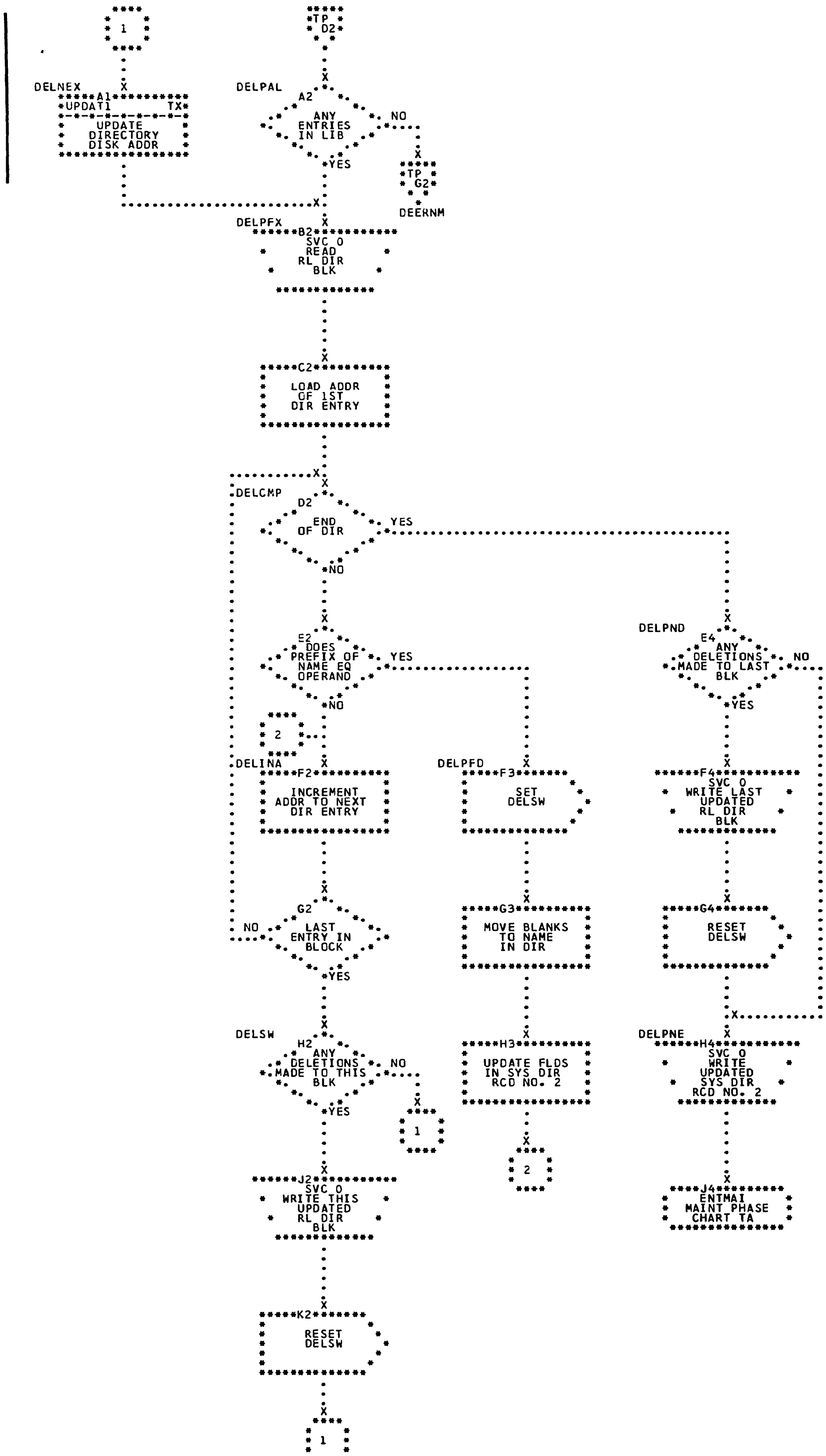


Chart TR. Build ESD Record for Relocatable Library MAINTR2 (Part 1 of 3); Refer to Maintenance, Chart 41

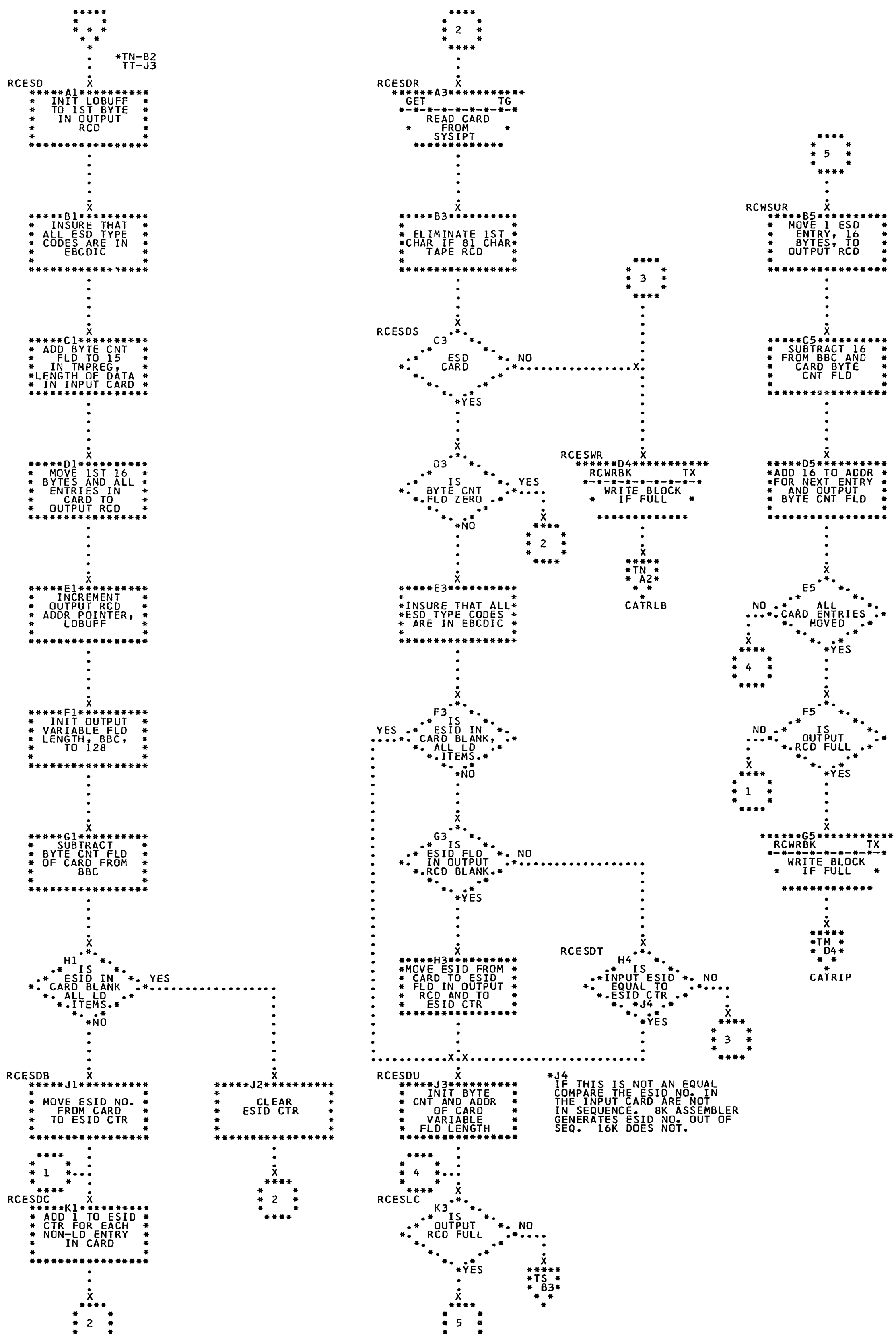


Chart TS. Build ESD Record for Relocatable Library MAINTR2
(Part 2 of 3); Refer to Maintenance, Chart 41

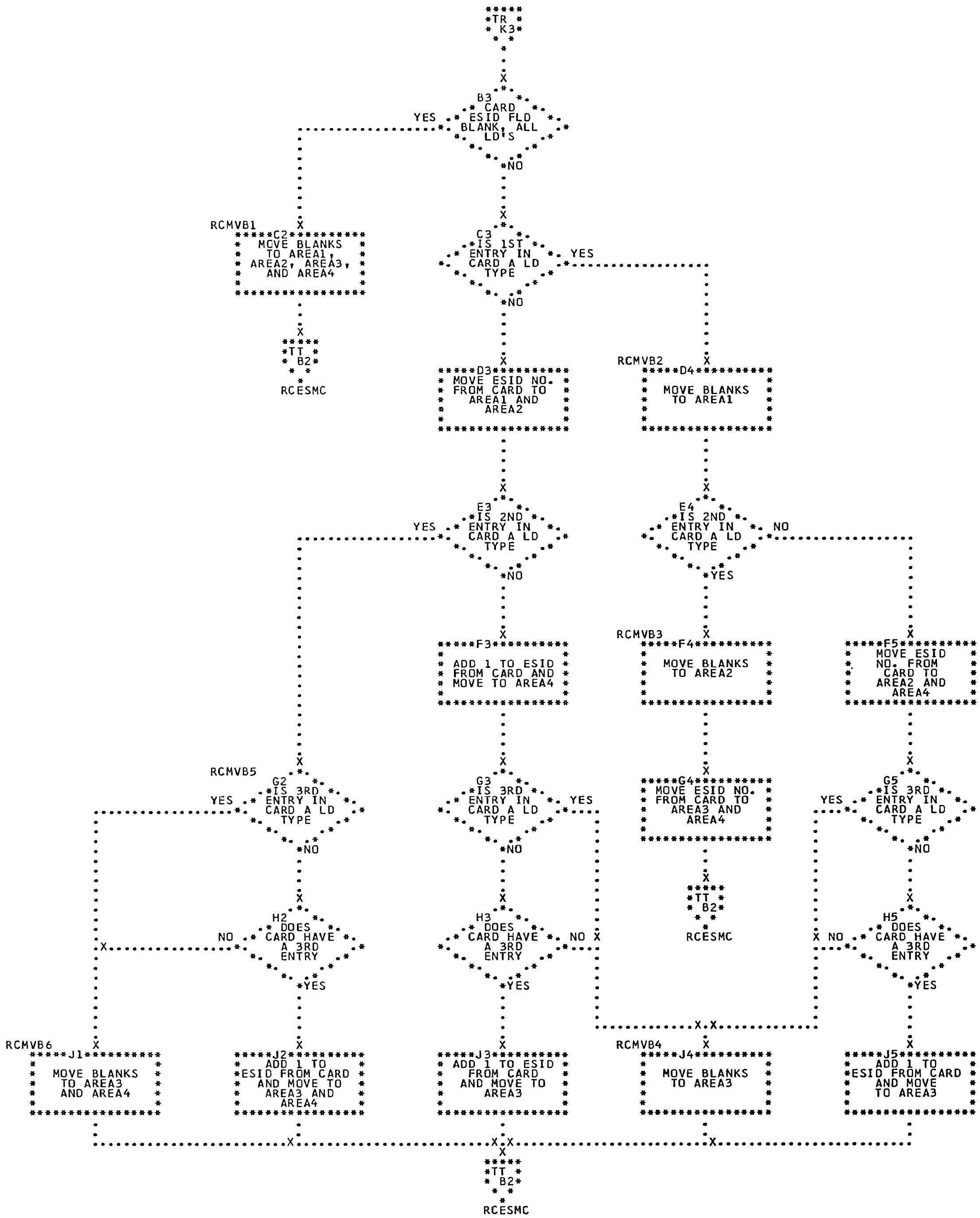
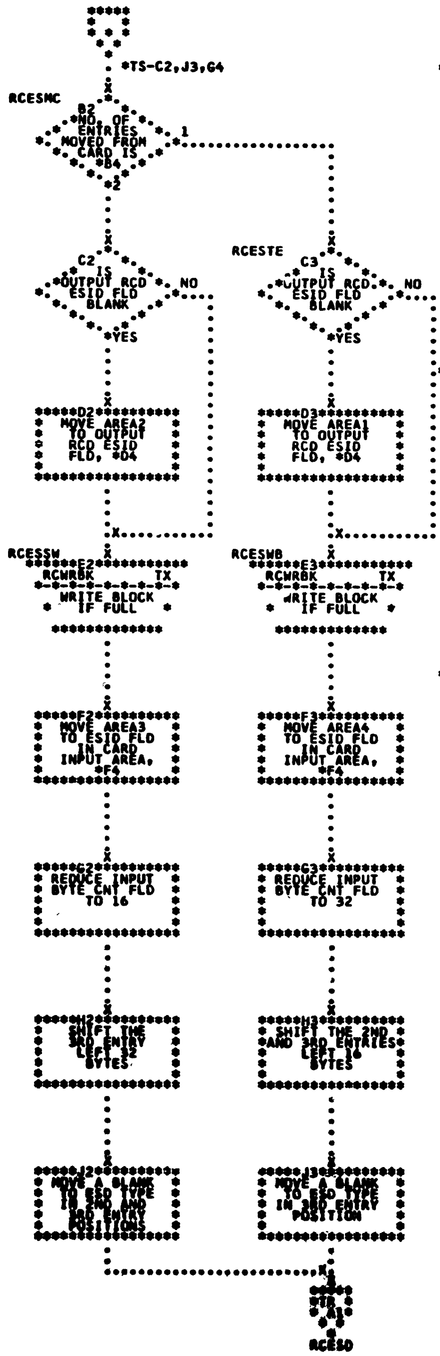


Chart TT. Build ESD Record for Relocatable Library MAINTR2 (Part 3 of 3); Refer to Maintenance, Chart 41




*B4
NO. OF ENTRIES MOVED MUST BE EITHER 1 OR 2 AT THIS POINT IN THE PROGRAM.

*D4
IF THE ESID NO. IN OUTPUT RCD IS BLANK AT THIS POINT IN THE PROGRAM ALL ENTRIES, EXCLUDING THOSE FROM THE LAST CARD, ARE LD ITEMS. IF THE ENTRIES MOVED TO THE OUTPUT RCD FROM THE LAST CARD ARE LD ITEMS THE ESID NO. OF THE OUTPUT RCD WILL BE BLANK. THE ESID NO. OF THE FIRST NON-LD ITEM IS THE ONE USED IF A NON-LD ITEM IS PRESENT.

*F4
THIS AREA CONTAINS THE ESID NO. OF THE FIRST NON-LD ITEM THAT WILL BE MOVED TO THE NEXT OUTPUT RCD. IF ENTRIES THAT ARE LEFT IN THE CARD INPUT AREA ARE ALL LD ITEMS THIS ESID WILL BE BLANK UNTIL MORE INPUT CARDS CAN BE ANALYZED.

Chart TU. Build RLD Record for Relocatable Library MAINTR2; Refer to Maintenance, Chart 41

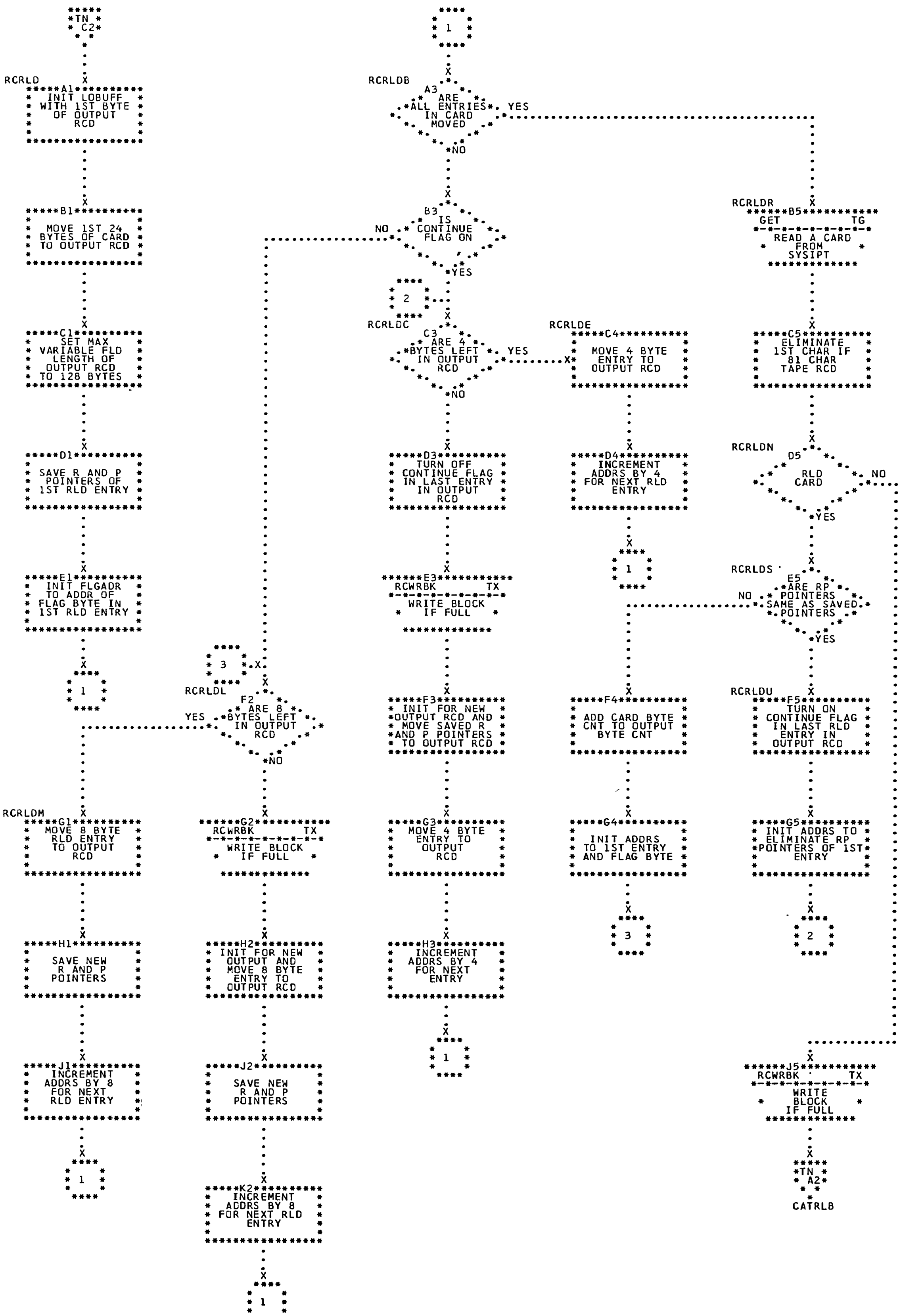


Chart TV. Build TXT Record for Relocatable Library MAINTR2; Refer to Maintenance, Chart 41

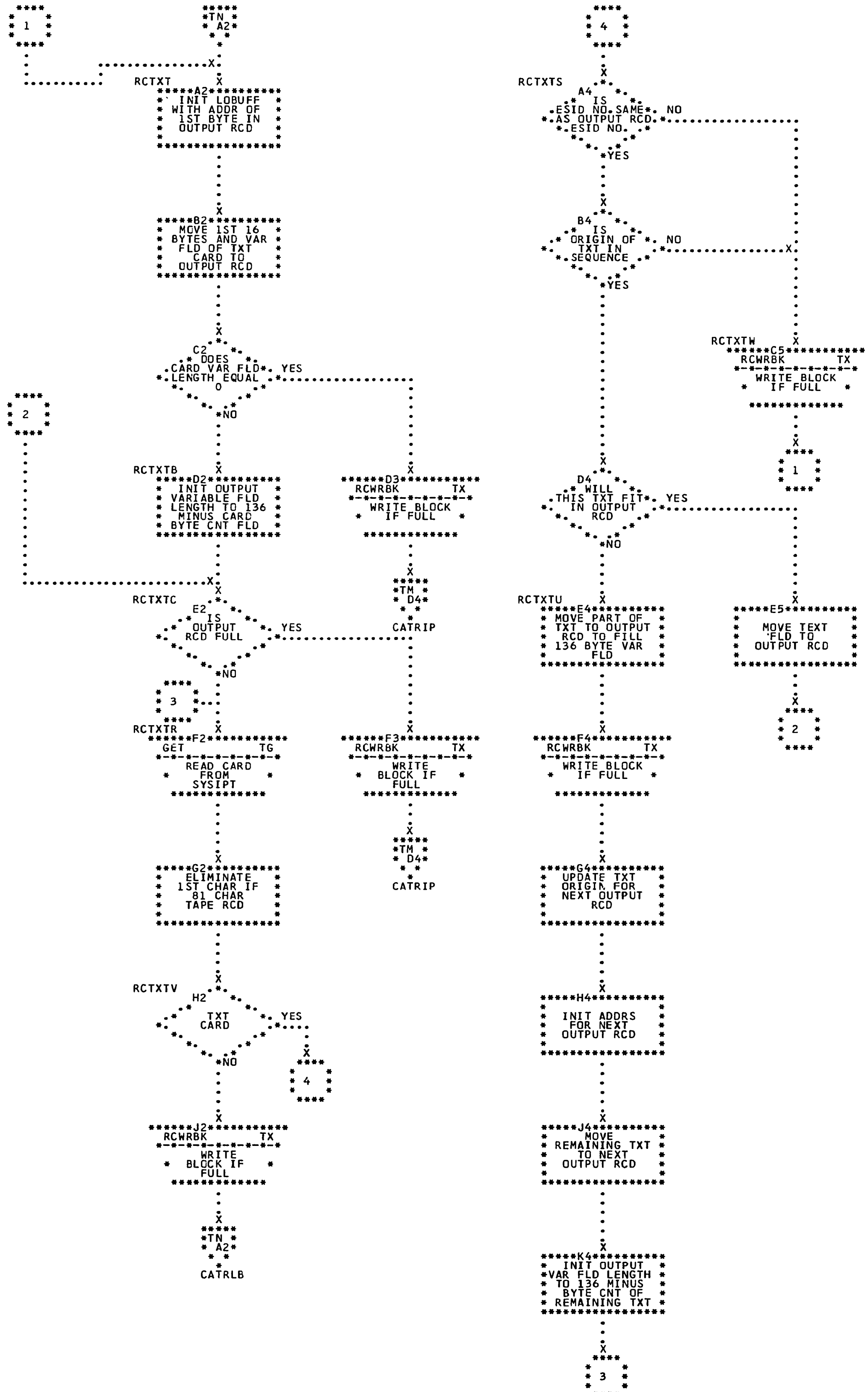


Chart TW. Rename a Module in Relocatable Library MAINTR2;
Refer to Maintenance, Chart 41

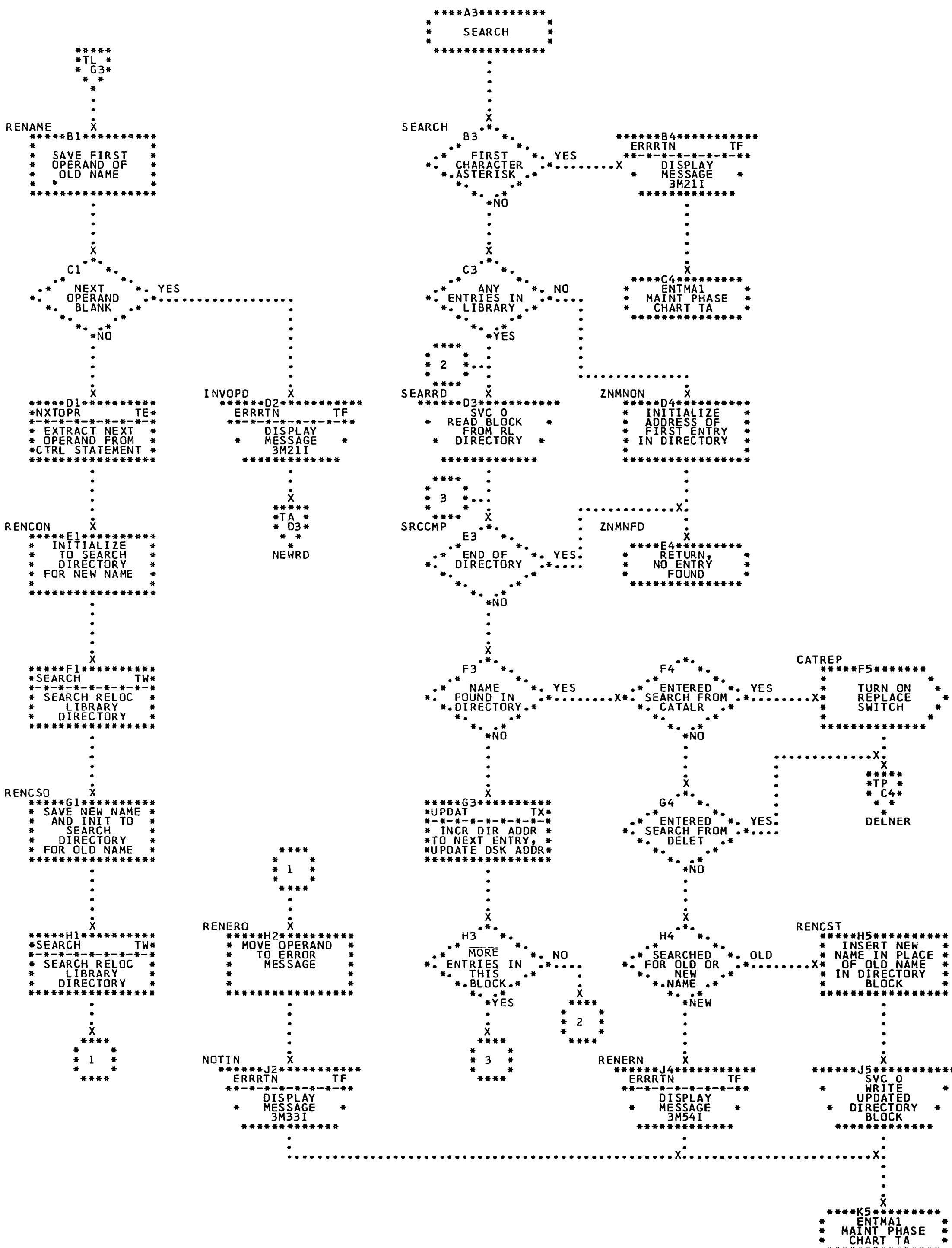


Chart TX. Write Block in Relocatable Library MAINTR2; Refer to Maintenance, Chart 41

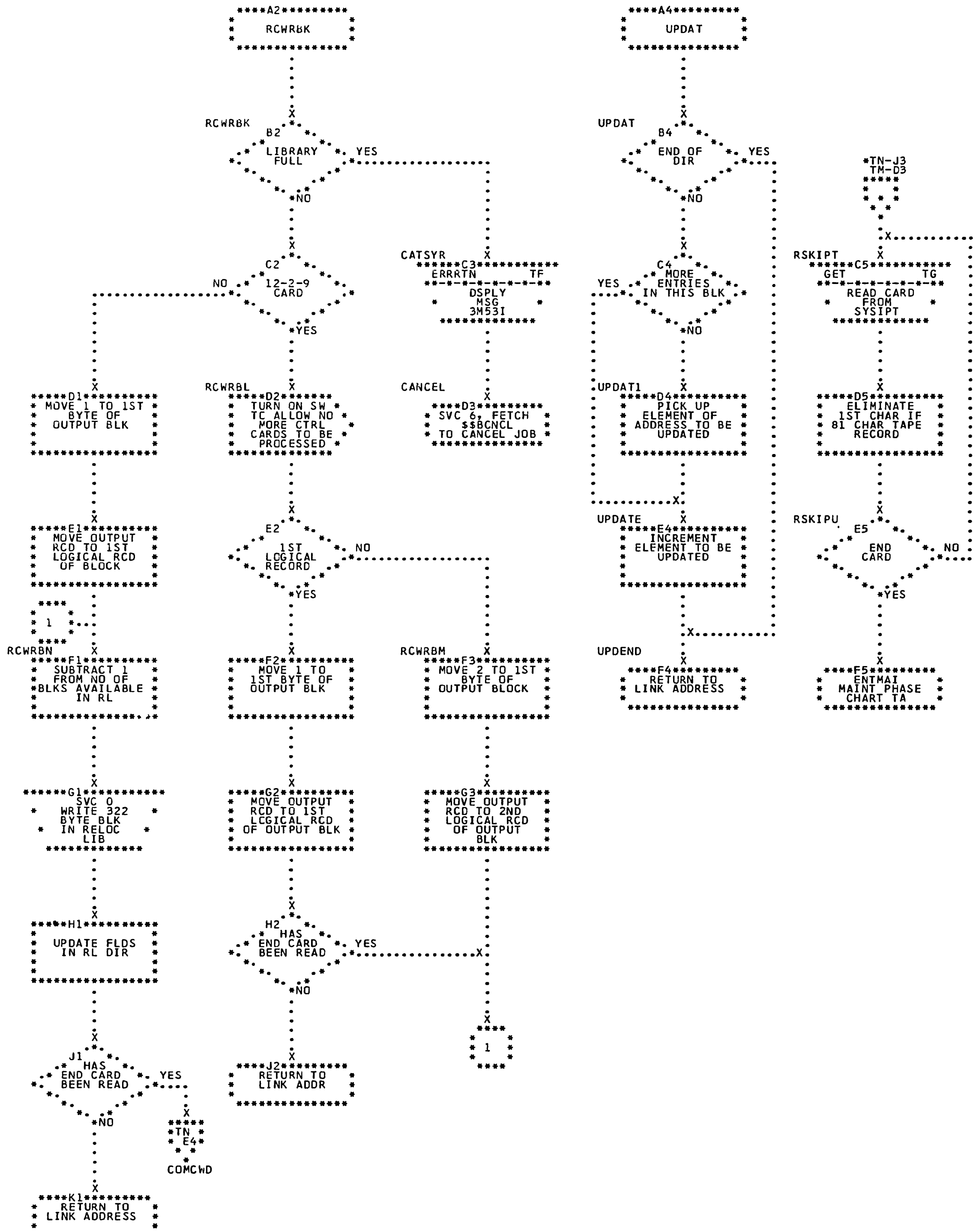
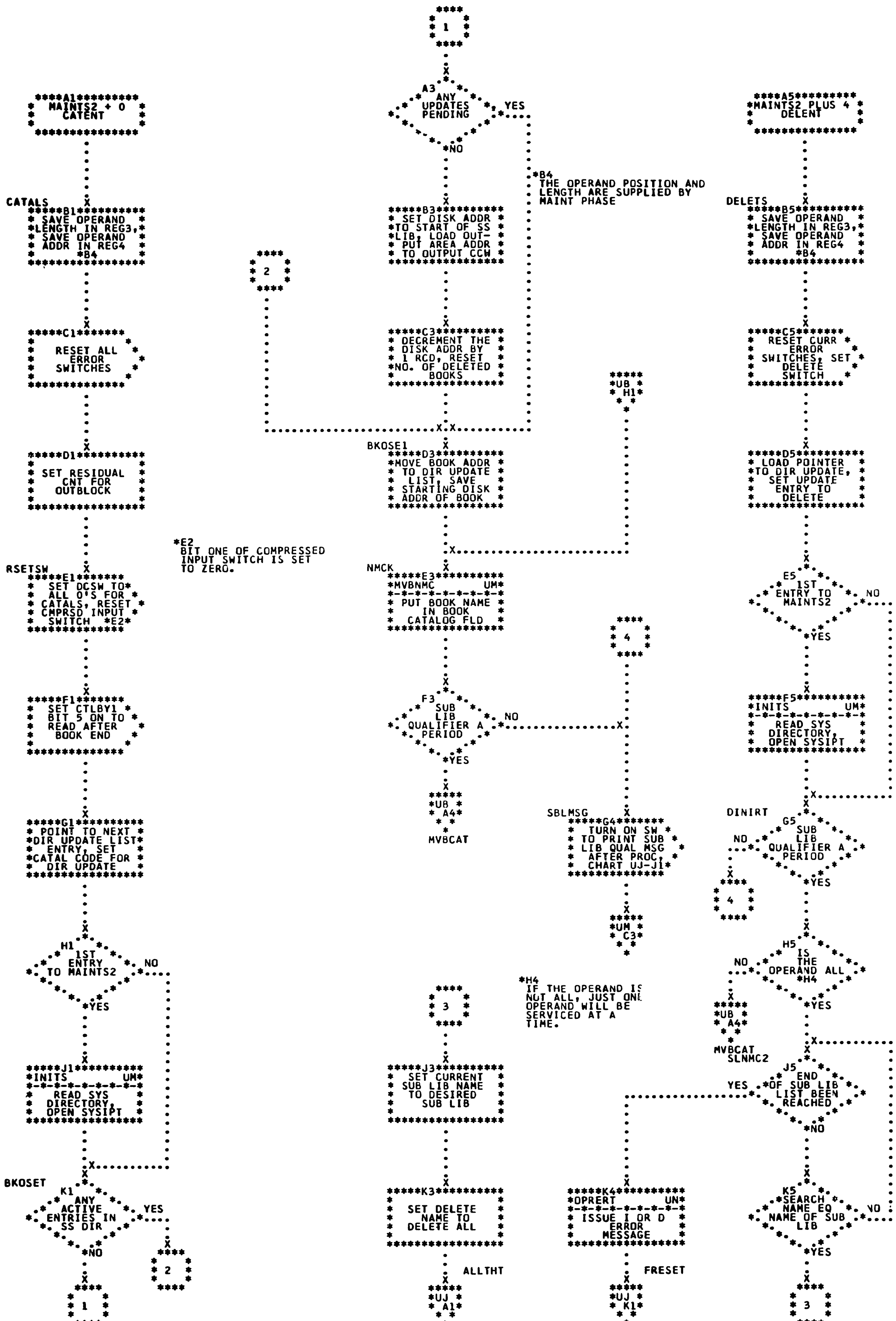


Chart UA. Catalog and Delete Entries MAINTS2; Refer to Maintenance, Chart 42



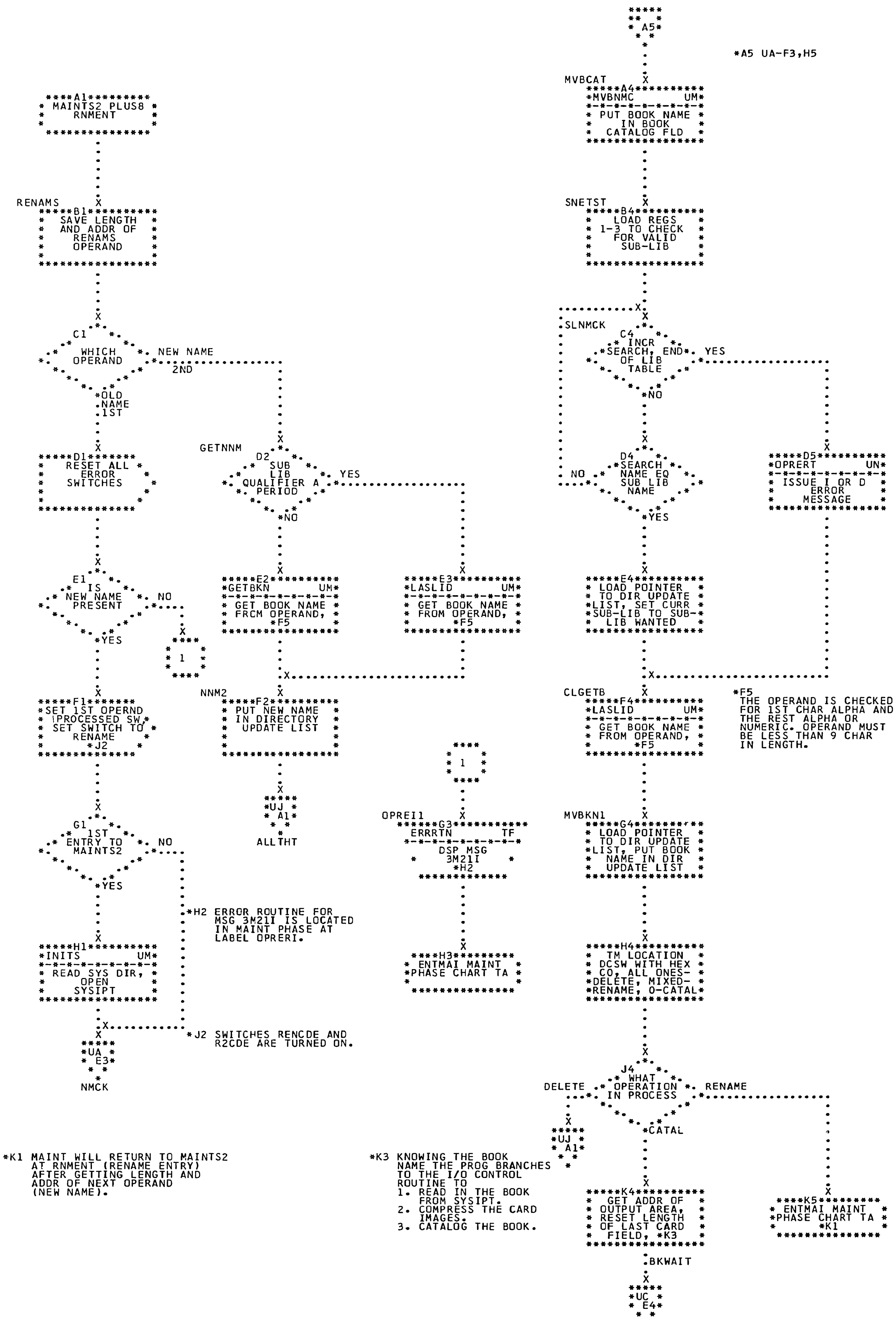

A5
*A5 UA-F3,H5
MVBCAT
A4 MVBNMC UM
PUT BOOK NAME IN BOOK CATALOG FLD
A1
MAINTS2 PLUS8 RNMENT
RENAMS
B1
SAVE LENGTH AND ADDR OF RENAMS OPERAND
SNETST
B4
LOAD REGS 1-3 TO CHECK FOR VALID SUB-LIB
SLNMCK
C4
INCR SEARCH, END OF LIB TABLE
YES
NO
C1
WHICH OPERAND
NEW NAME 2ND
OLD NAME 1ST
GETNNM
D2
SUB LIB QUALIFIER A PERIOD
YES
NO
D1
RESET ALL ERROR SWITCHES
D4
SEARCH NAME EQ SUB LIB NAME
NO
YES
D5 OPRERT UN
ISSUE I OR D ERROR MESSAGE
E1
IS NEW NAME PRESENT
NO
YES
1
E2 GETBKN UM
GET BOOK NAME FRCM OPERAND, *F5
E3 LASLID UM
GET BOOK NAME FROM OPERAND, *F5
E4
LOAD POINTER TO DIR UPDATE LIST, SET CURR SUB-LIB TO SUB-LIB WANTED
F1
SET 1ST OPERND PROCESSED SW, SET SWITCH TO RENAME *J2
NNM2
F2
PUT NEW NAME IN DIRECTORY UPDATE LIST
CLGETB
F4 LASLID UM
GET BOOK NAME FROM OPERAND, *F5
*F5 THE OPERAND IS CHECKED FOR 1ST CHAR ALPHA AND THE REST ALPHA OR NUMERIC. OPERAND MUST BE LESS THAN 9 CHAR IN LENGTH.
UJ A1
ALLTHT
1
OPREI1
G3 ERRRTN TF
DSP MSG 3M21I *H2
MVBKN1
G4
LOAD POINTER TO DIR UPDATE LIST, PUT BOOK NAME IN DIR UPDATE LIST
G1
1ST ENTRY TO MAINTS2
NO
YES
*H2 ERROR ROUTINE FOR MSG 3M21I IS LOCATED IN MAINT PHASE AT LABEL OPRERI.
H1 INITS UM
READ SYS DIR, OPEN SYSIPT
H3
ENTMAI MAINT PHASE CHART TA
H4
TM LOCATION DCSW WITH HEX CO, ALL ONES-DELETE, MIXED-RENAME, O-CATAL
UA E3
NMCK
*J2 SWITCHES RENCDE AND R2CDE ARE TURNED ON.
J4
WHAT OPERATION IN PROCESS
DELETE
RENAME
CATAL
UJ A1
*K1 MAINT WILL RETURN TO MAINTS2 AT RNMENT (RENAME ENTRY) AFTER GETTING LENGTH AND ADDR OF NEXT OPERAND (NEW NAME).
*K3 KNOWING THE BOOK NAME THE PROG BRANCHES TO THE I/O CONTROL ROUTINE TO
1. READ IN THE BOOK FROM SYSIPT.
2. COMPRESS THE CARD IMAGES.
3. CATALOG THE BOOK.
K4
GET ADDR OF OUTPUT AREA, RESET LENGTH OF LAST CARD FIELD, *K3
K5
ENTMAI MAINT PHASE CHART TA *K1
BKWAIT
UC E4

Chart UC. I/O Control MAINTS2 (Part 1 of 2); Refer to Maintenance, Chart 42

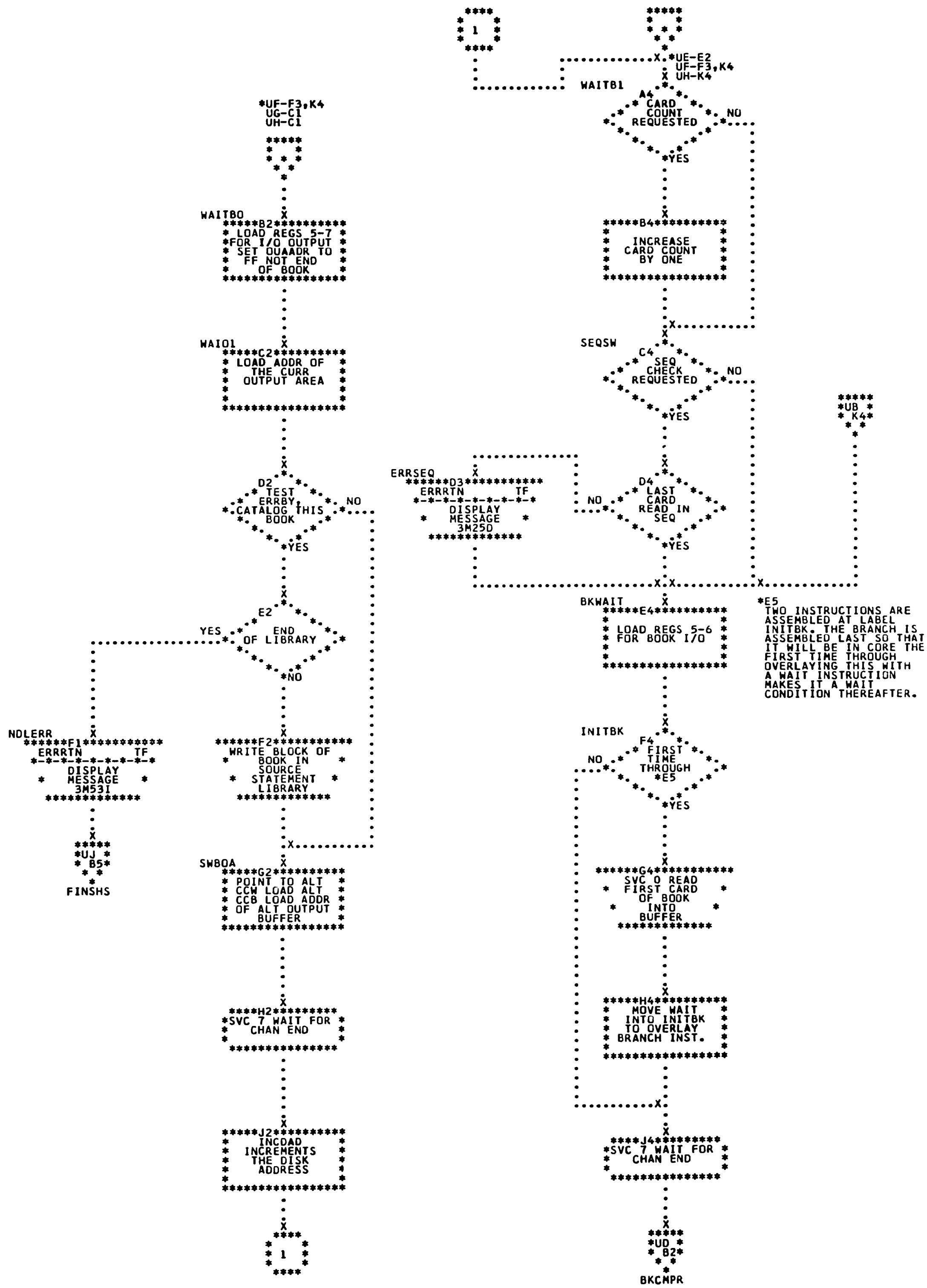


Chart UD. I/O Control MAINTS2 (Part 2 of 2); Refer to Maintenance, Chart 42

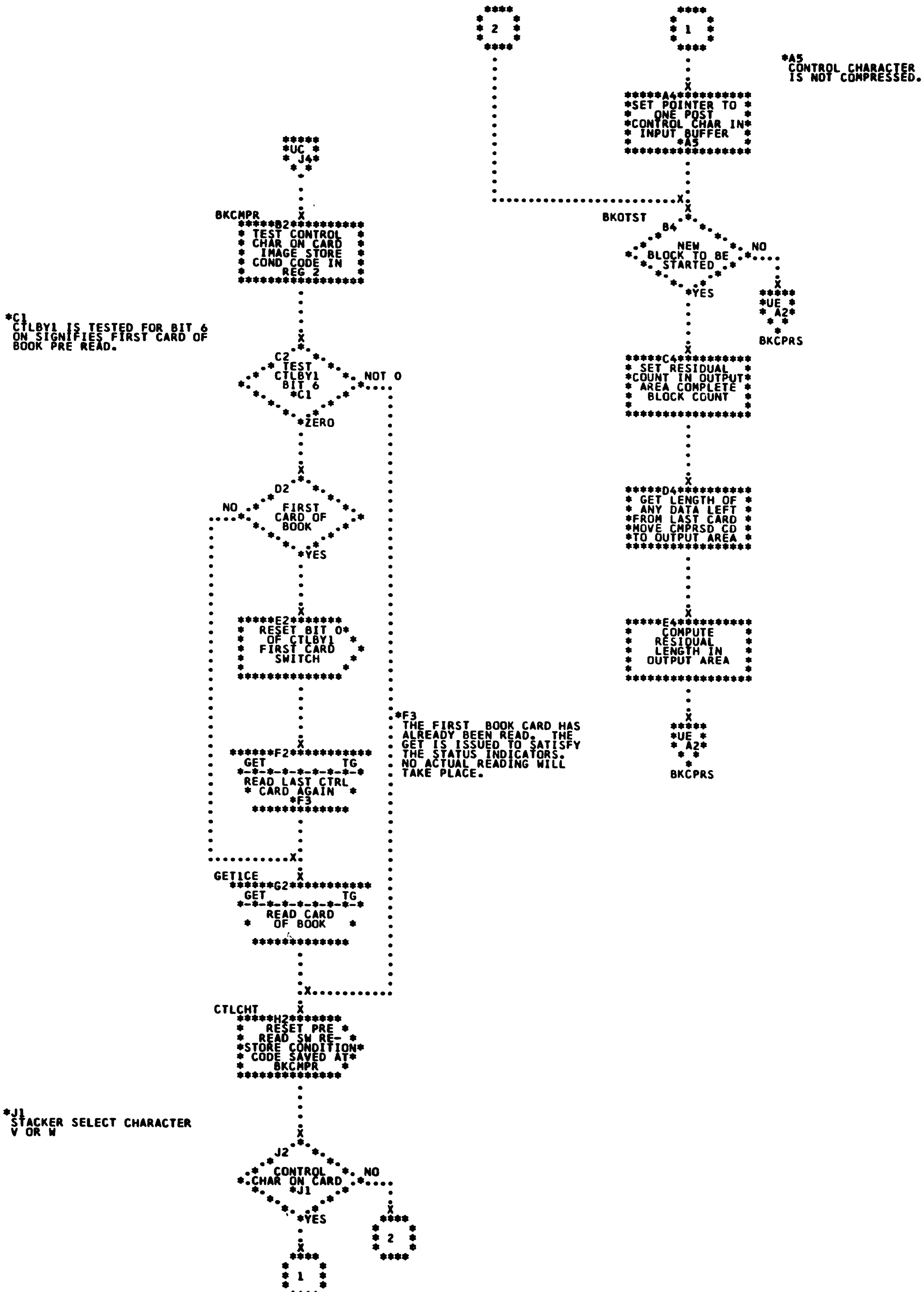


Chart UE. Format Book MAINTS2; Refer to Maintenance, Chart 42

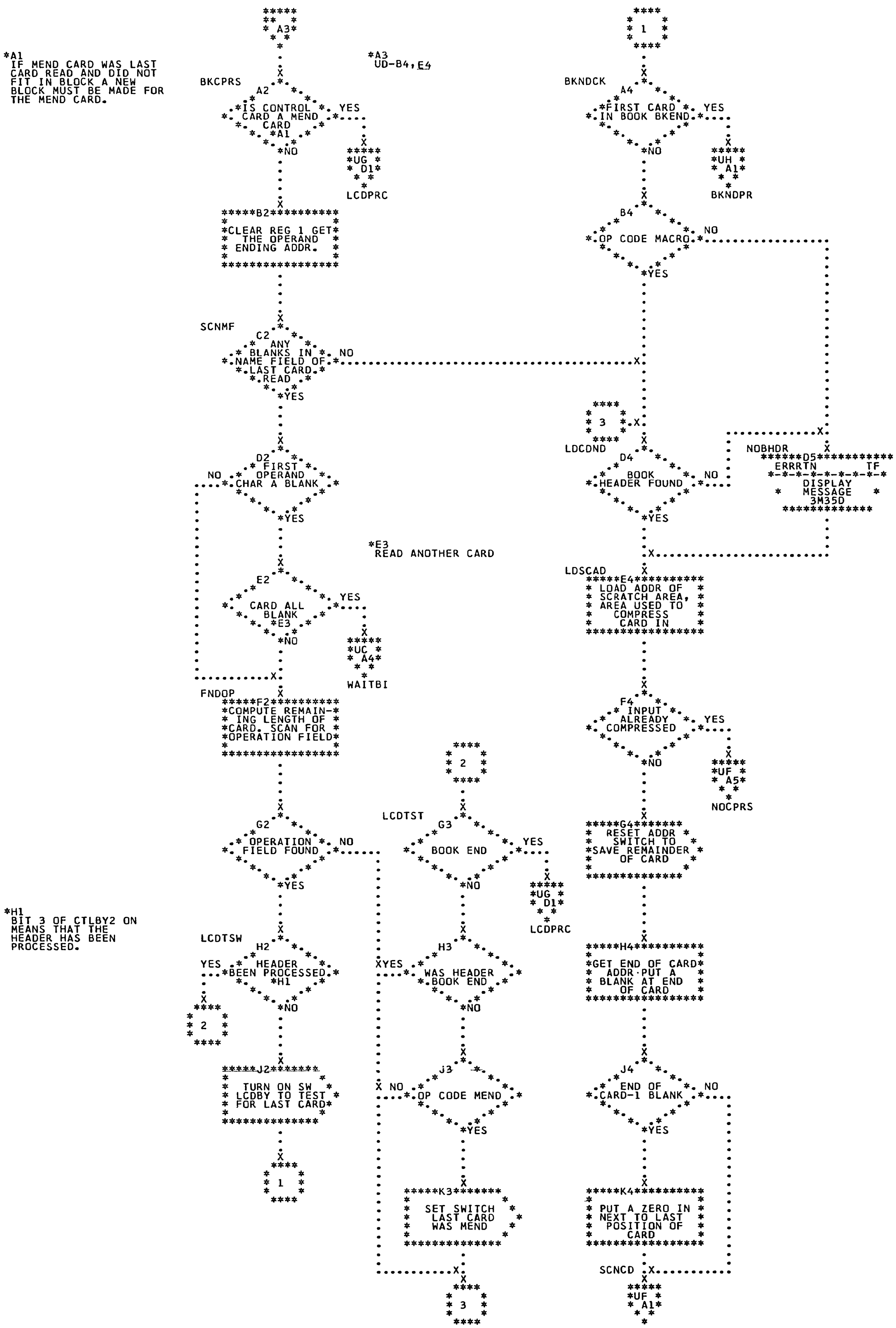


Chart UF. Compress Book and Format Book Already Compressed
MAINTS2: Refer to Maintenance, Chart 42

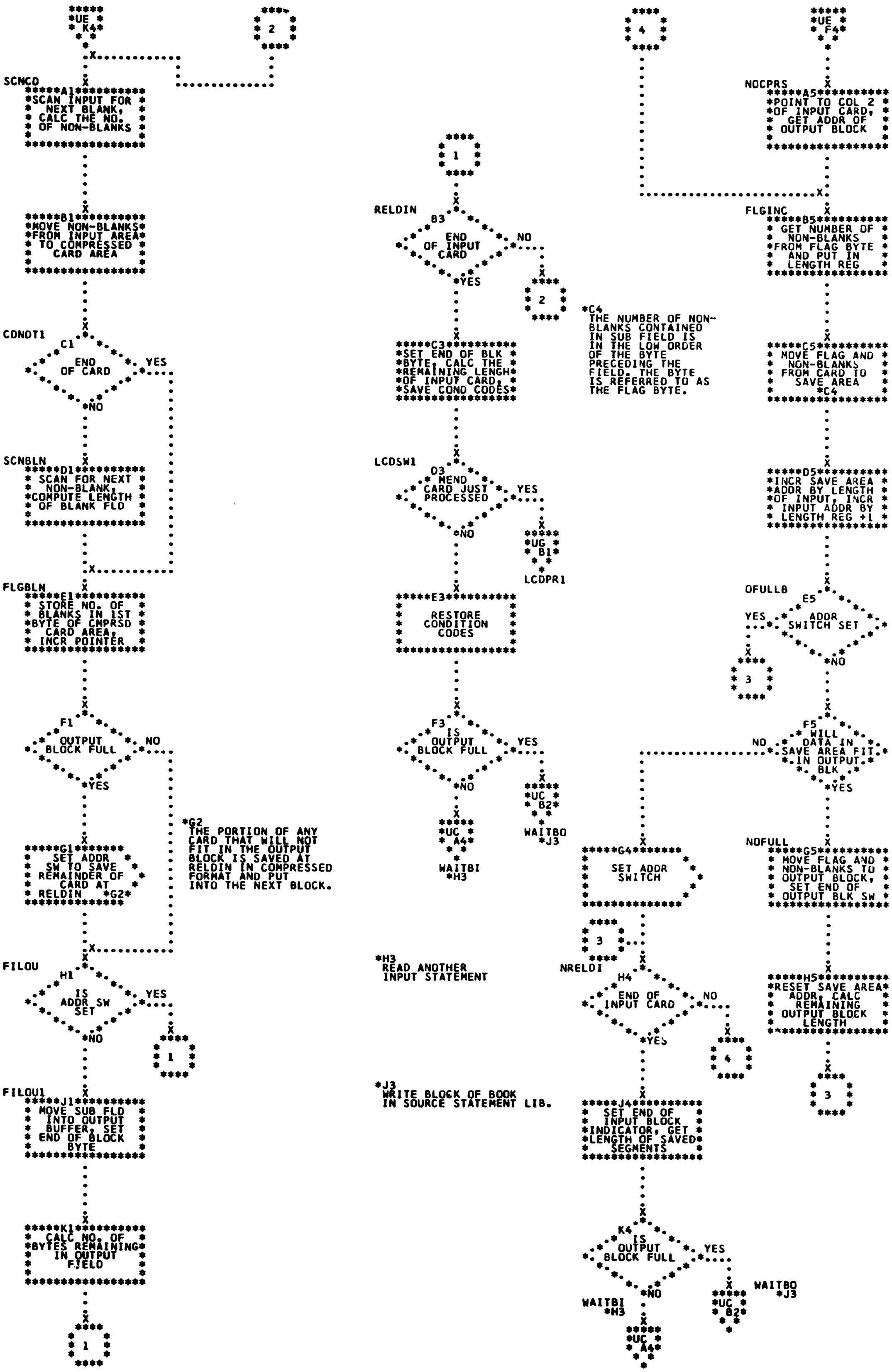


Chart UG. Last Card in Book Processing MAINTS2; Refer to Maintenance, Chart 42

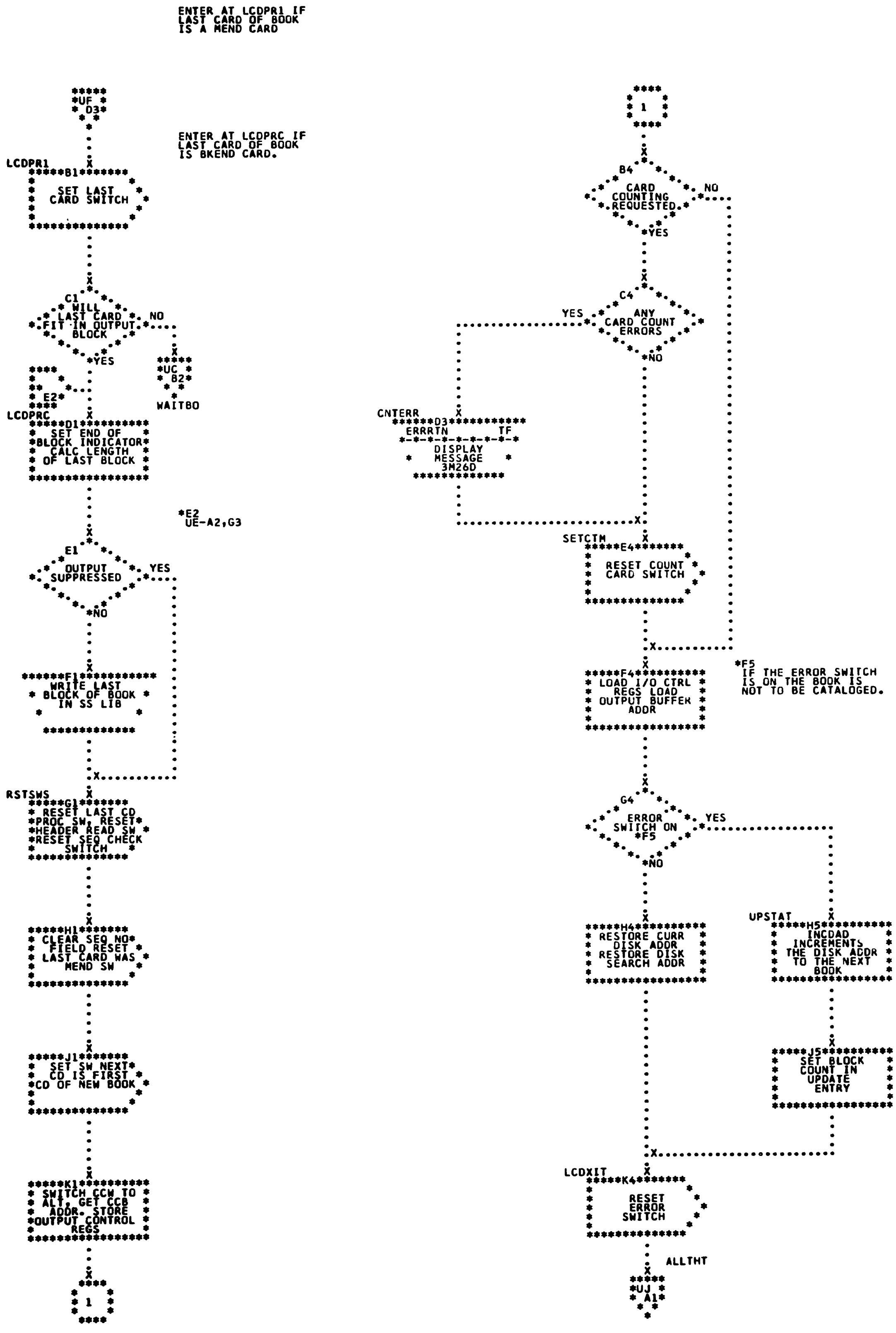


Chart UH. Book End Statement Processor MAINTS2; Refer to Maintenance, Chart 42

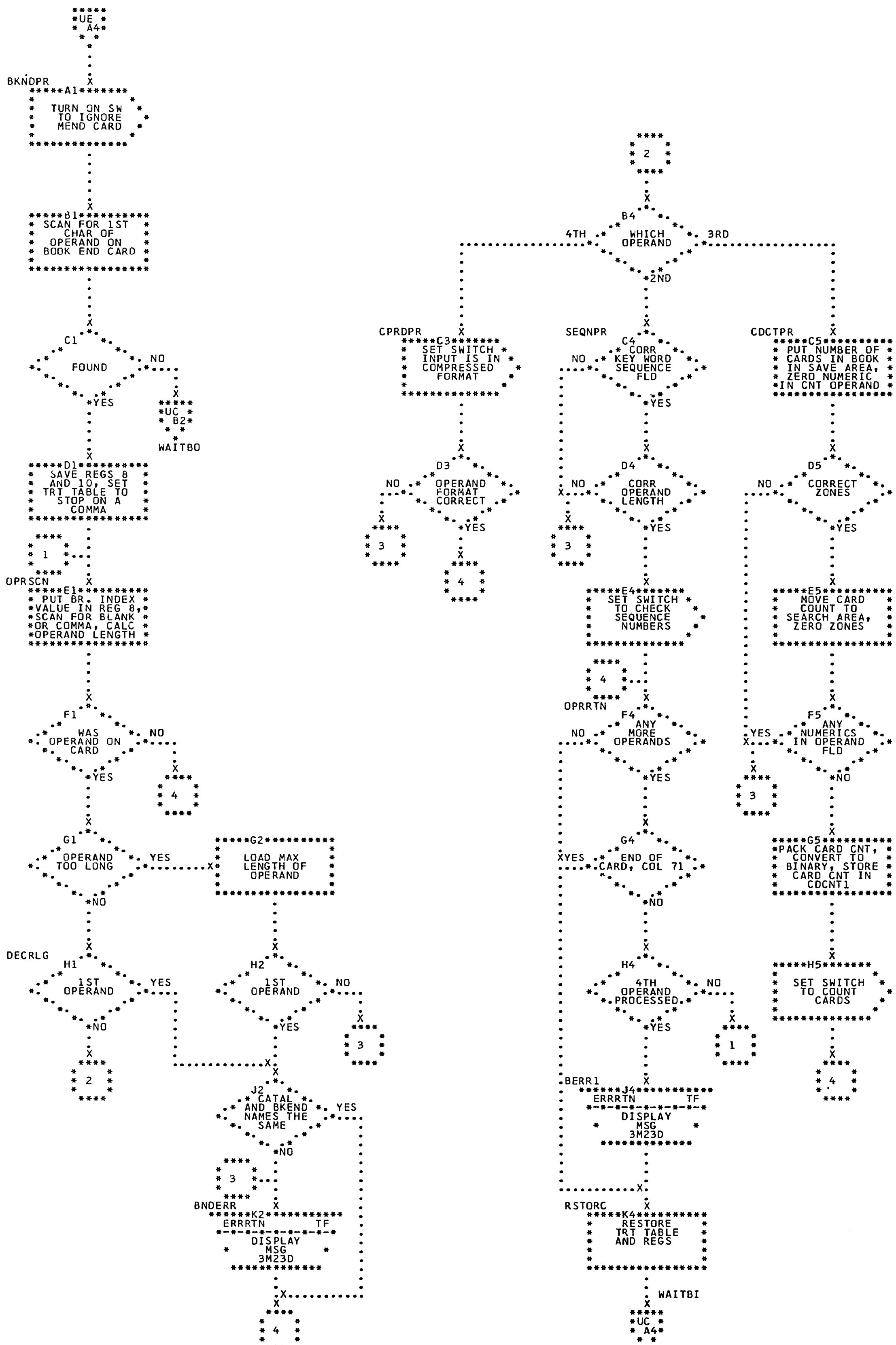


Chart UJ. Finish MAINTS2 Entry and All Through Processing Routine MAINTS2; Refer to Maintenance, Chart 42

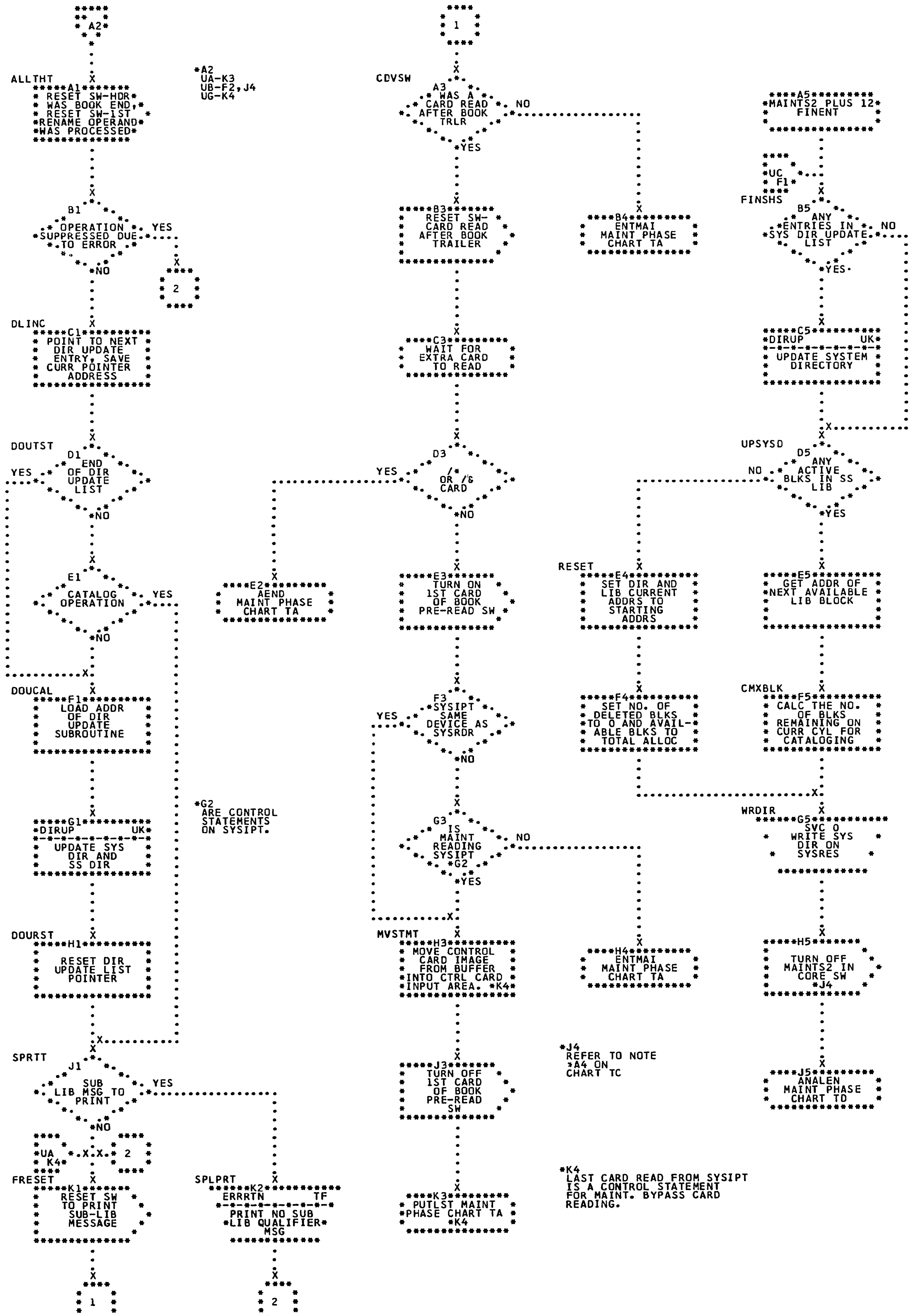


Chart UK. Update Source Statement Directory Subroutine MAINTS2 (Part 1 of 2); Refer to Maintenance, Chart 42

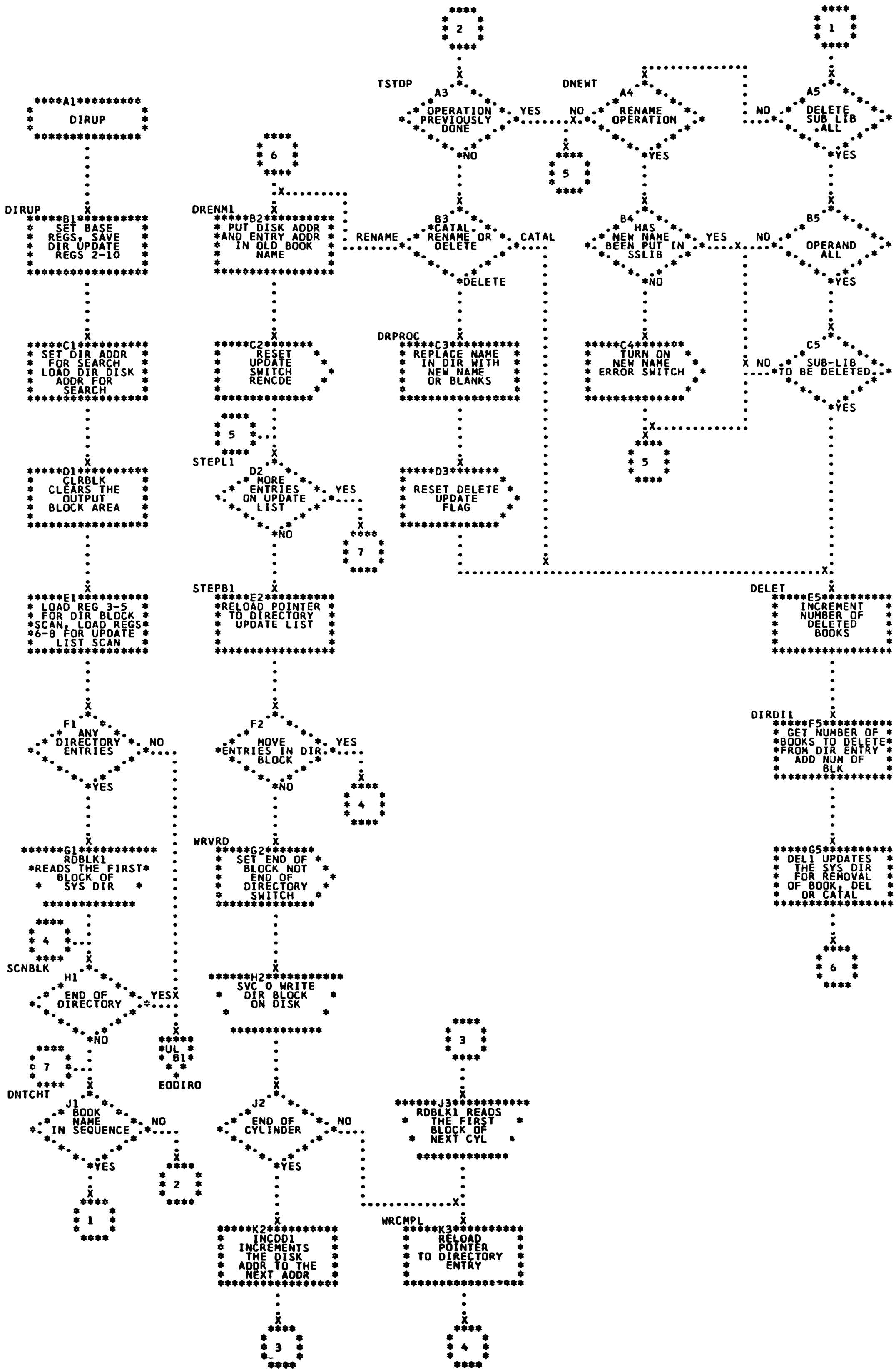


Chart UL. Update Source Statement Directory Subroutine MAINTS2 (Part 2 of 2); Refer to Maintenance, Chart 42

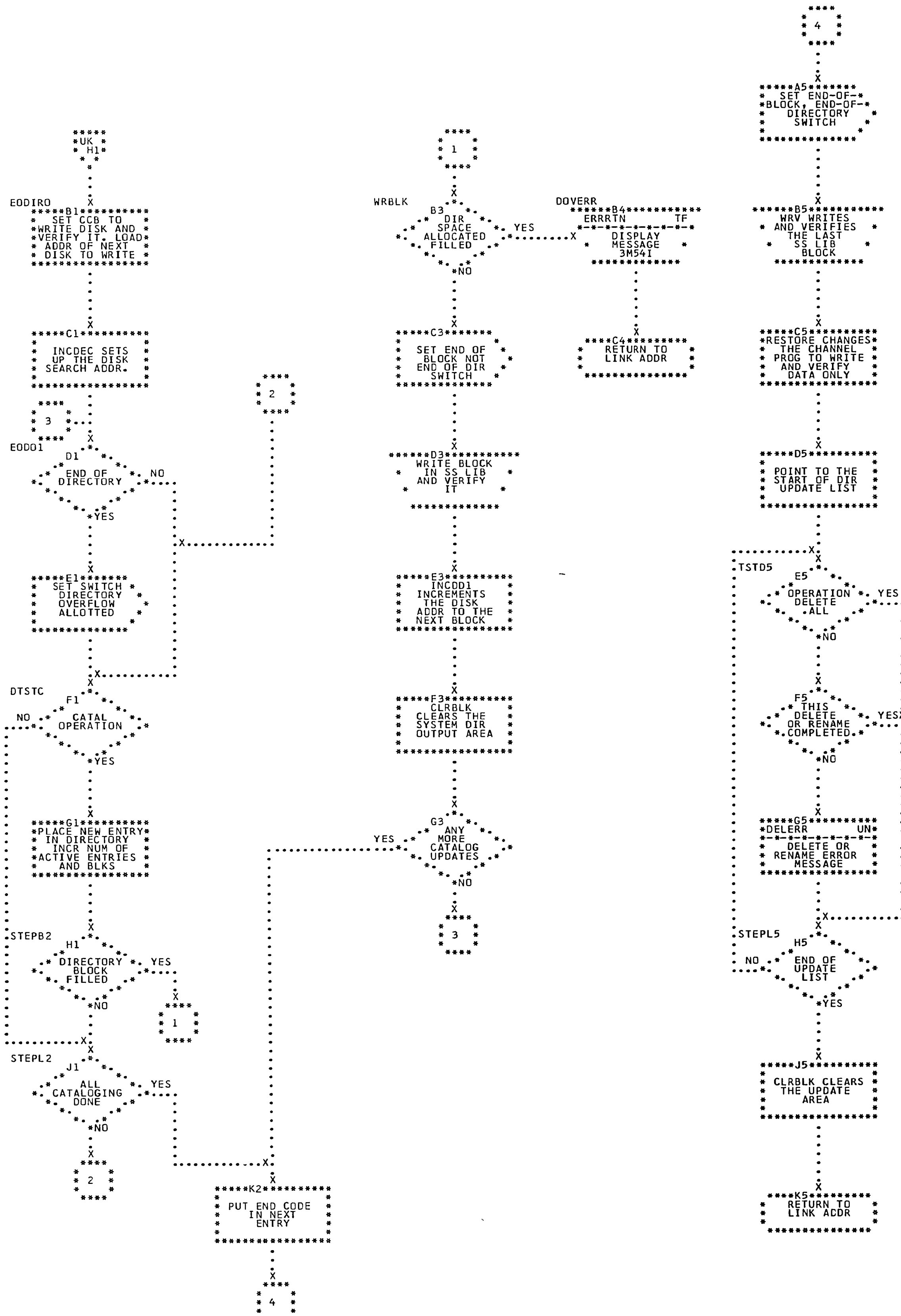


Chart UM. INITS, GETBKN, LASLID, and MVBNMC Subroutines
MAINTS2; Refer to Maintenance, Chart 42

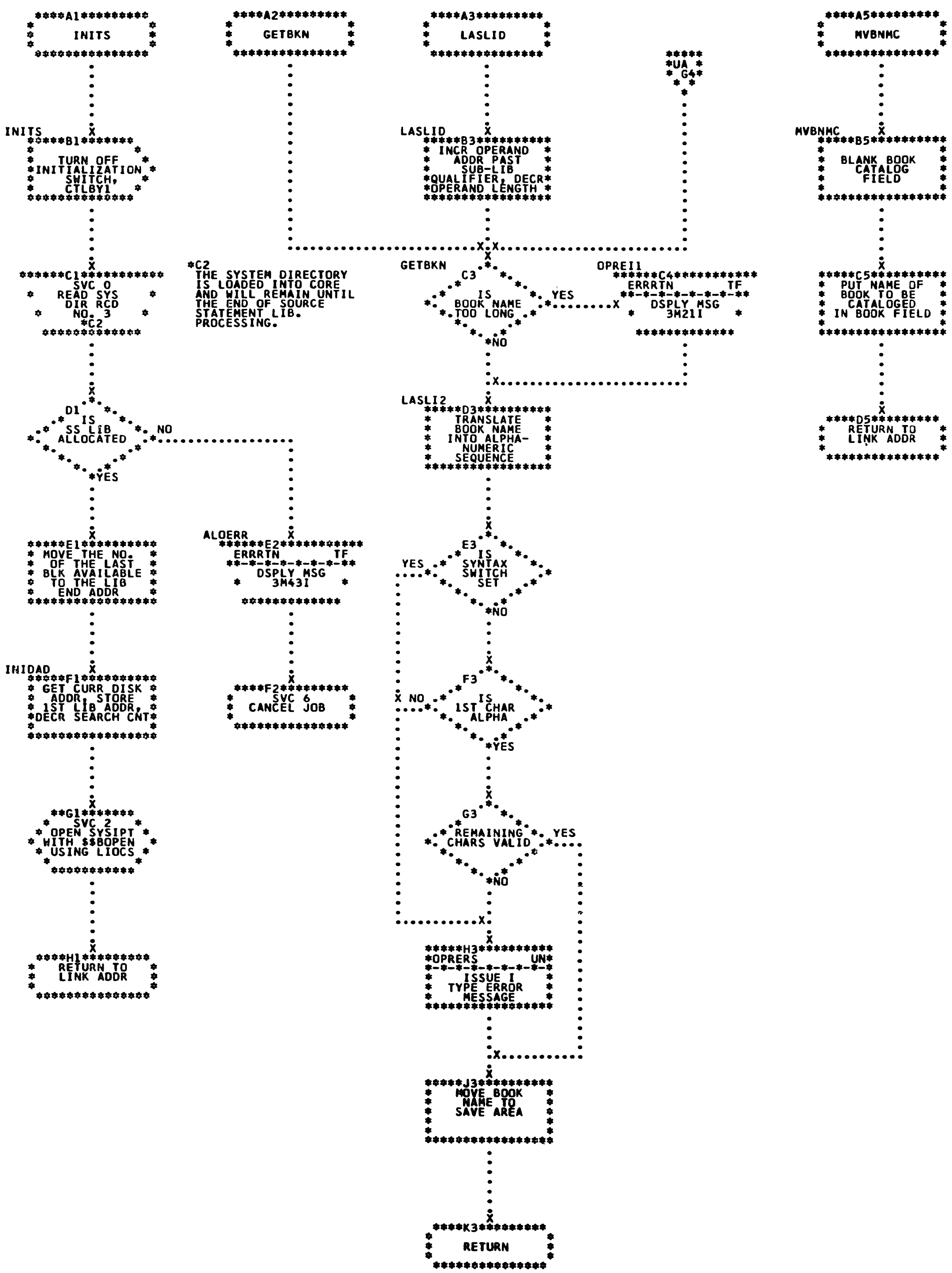


Chart UN. OPRERS, OPRERT, and DELERR Error Subroutines
MAINTS2; Refer to Maintenance, Chart 42

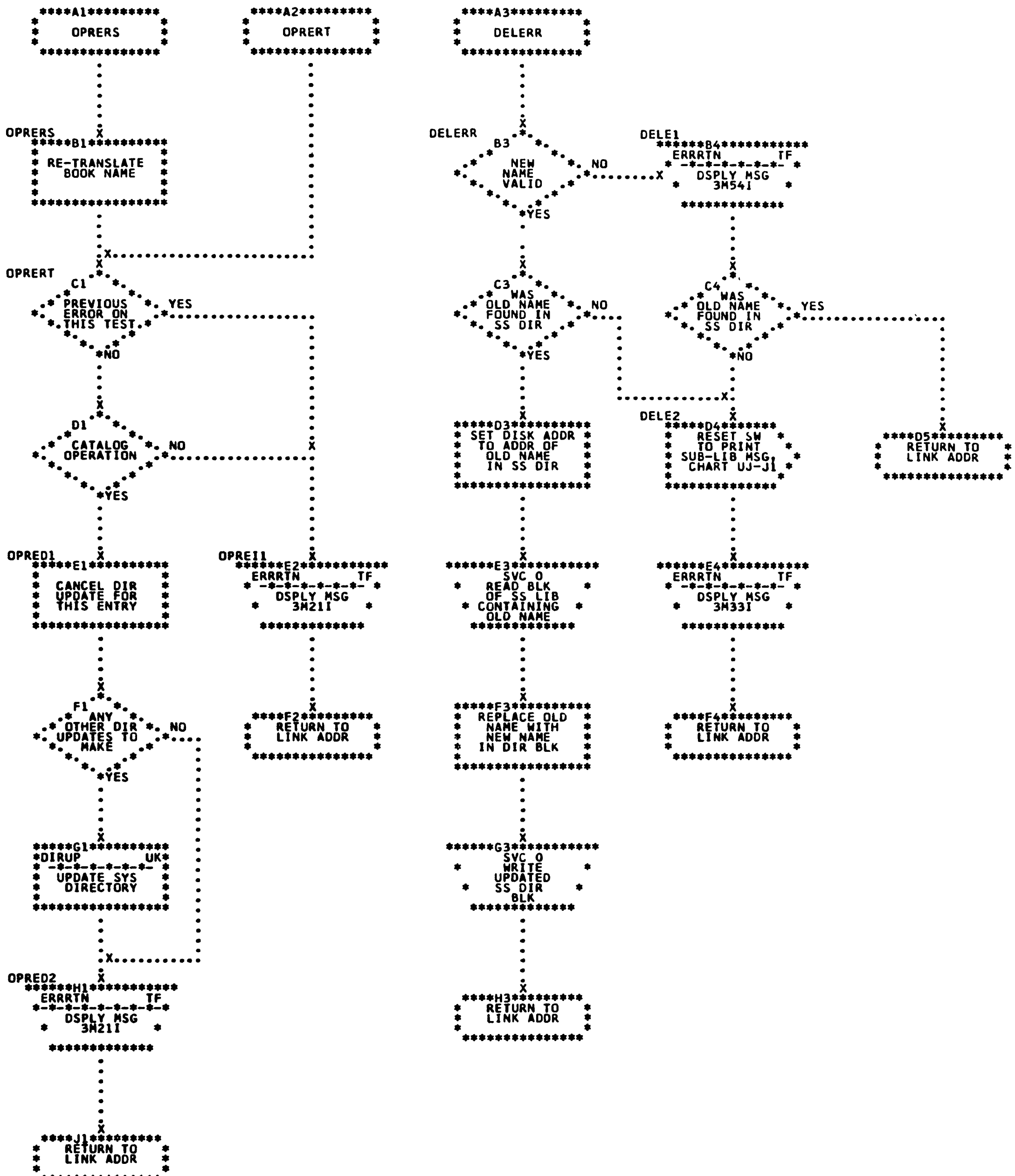


Chart VA. Process Allocate Control Statement MAINTA (Part 1 of 2); Refer to Maintenance, Chart 43

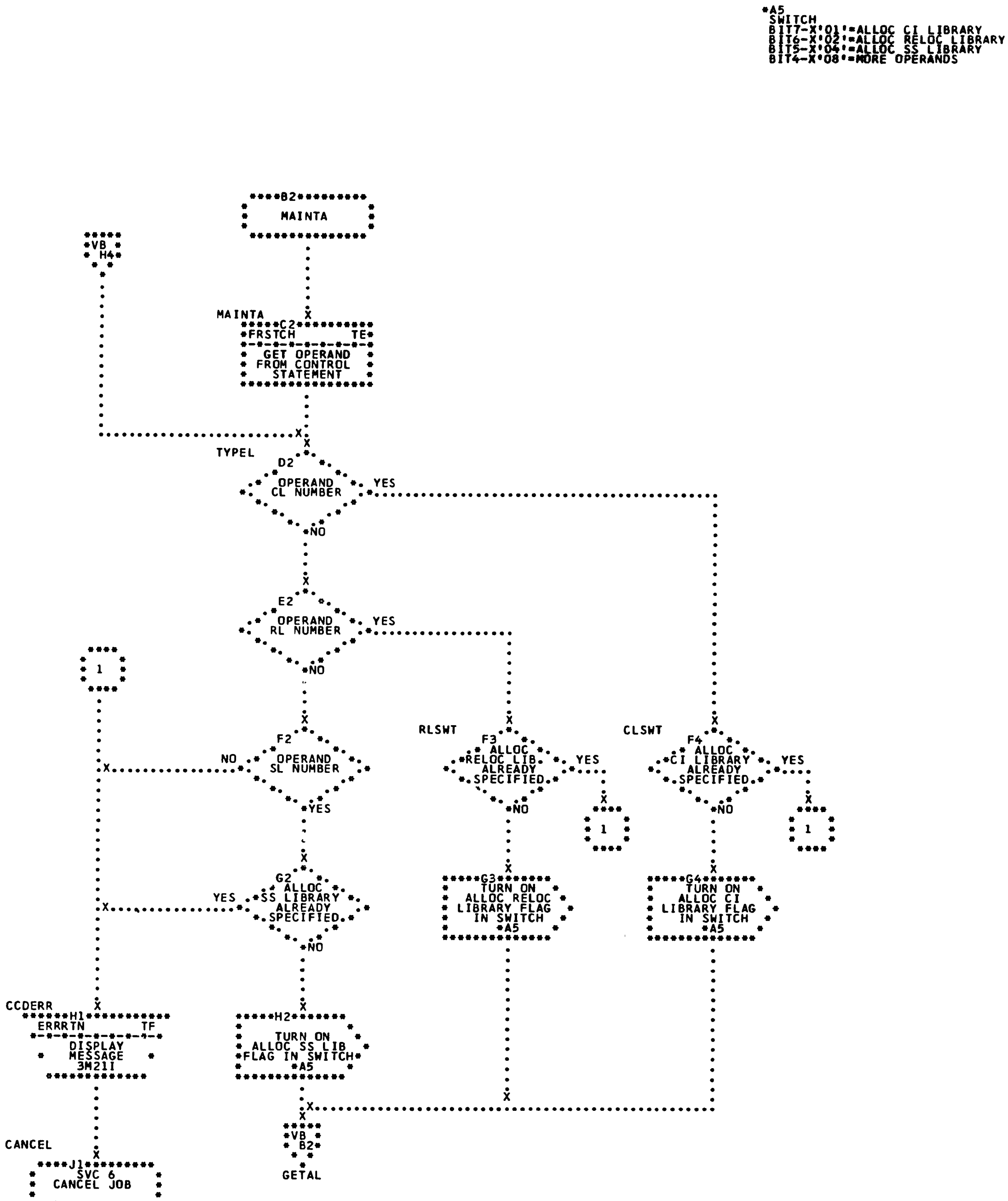


Chart VB. Process Allocate Control Statement MAINTA (Part 2 of 2); Refer to Maintenance, Chart 43

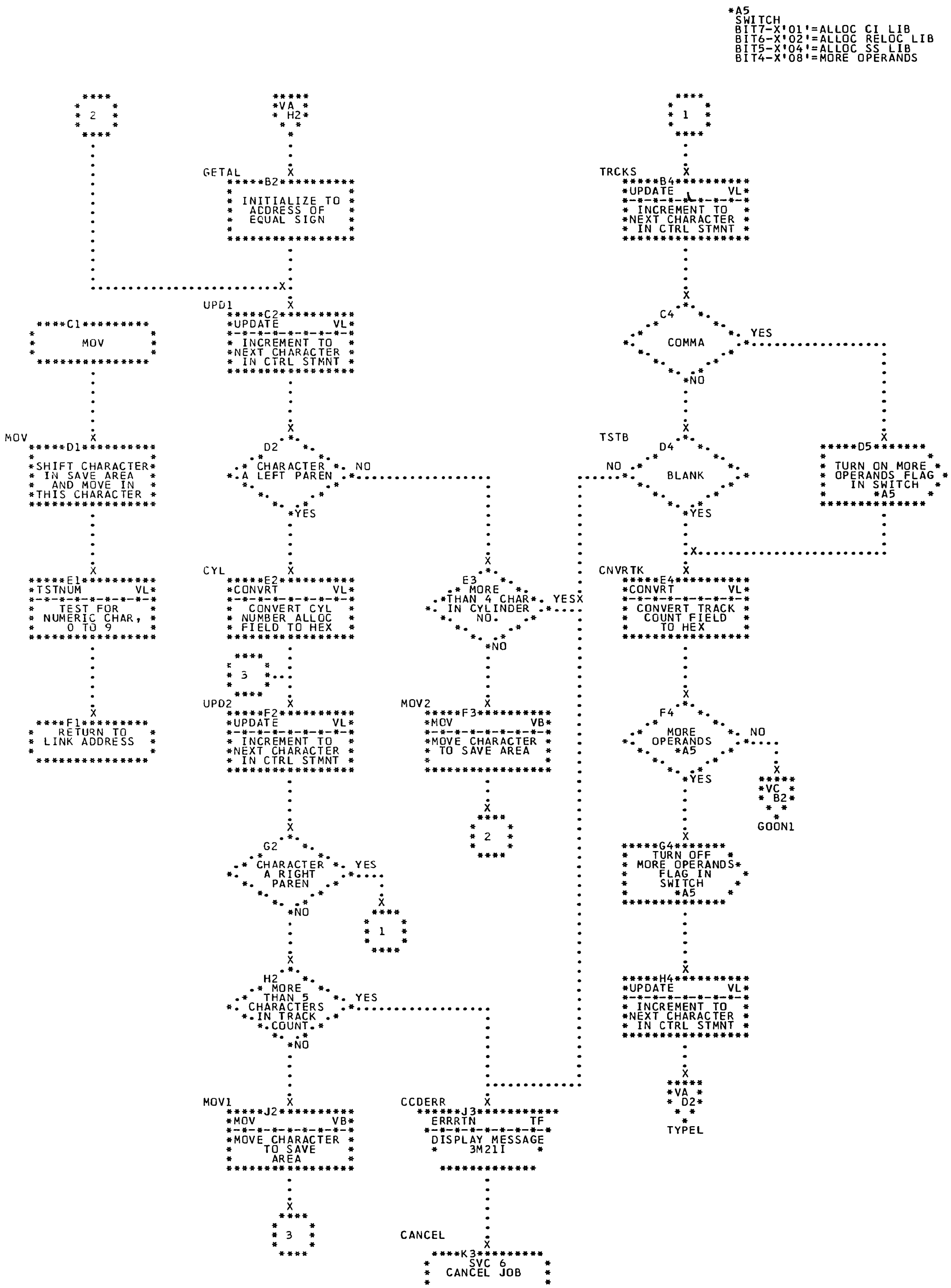


Chart VC. Update Record 4 of System directory MAINTA; Refer to Maintenance, Chart 43

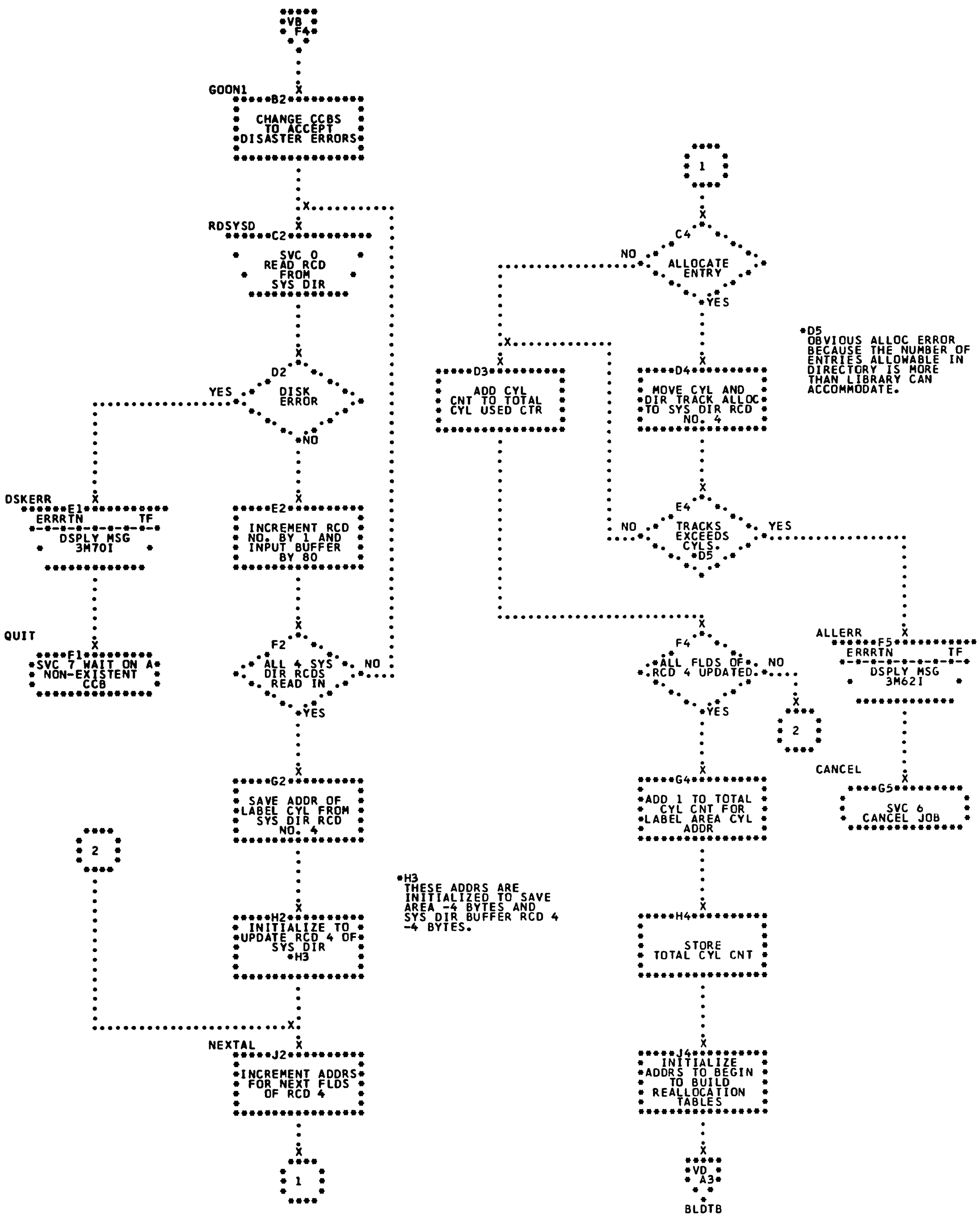


Chart VD. Build Directory and Library Reallocation Tables
MAINTA; Refer to Maintenance, Chart 43

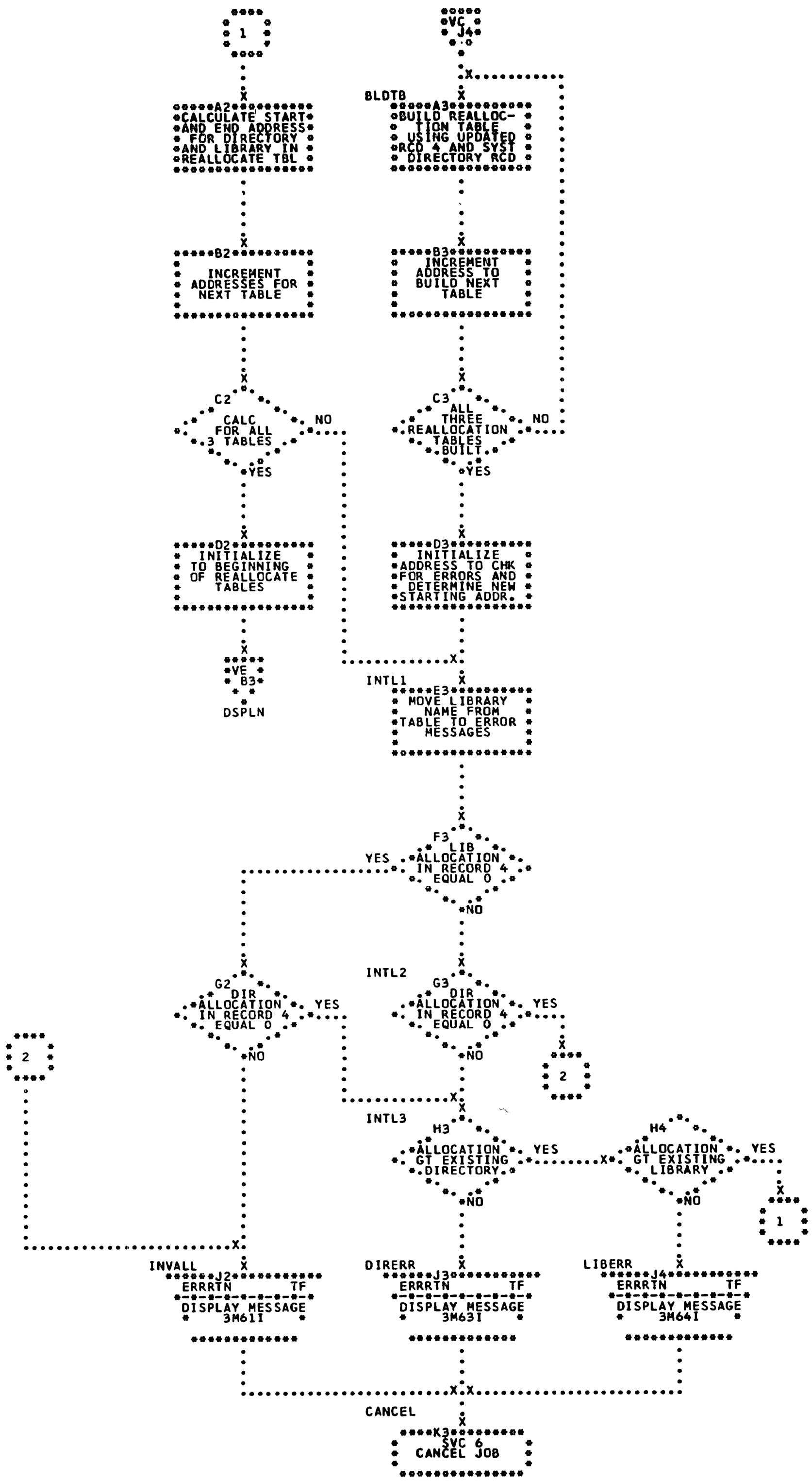
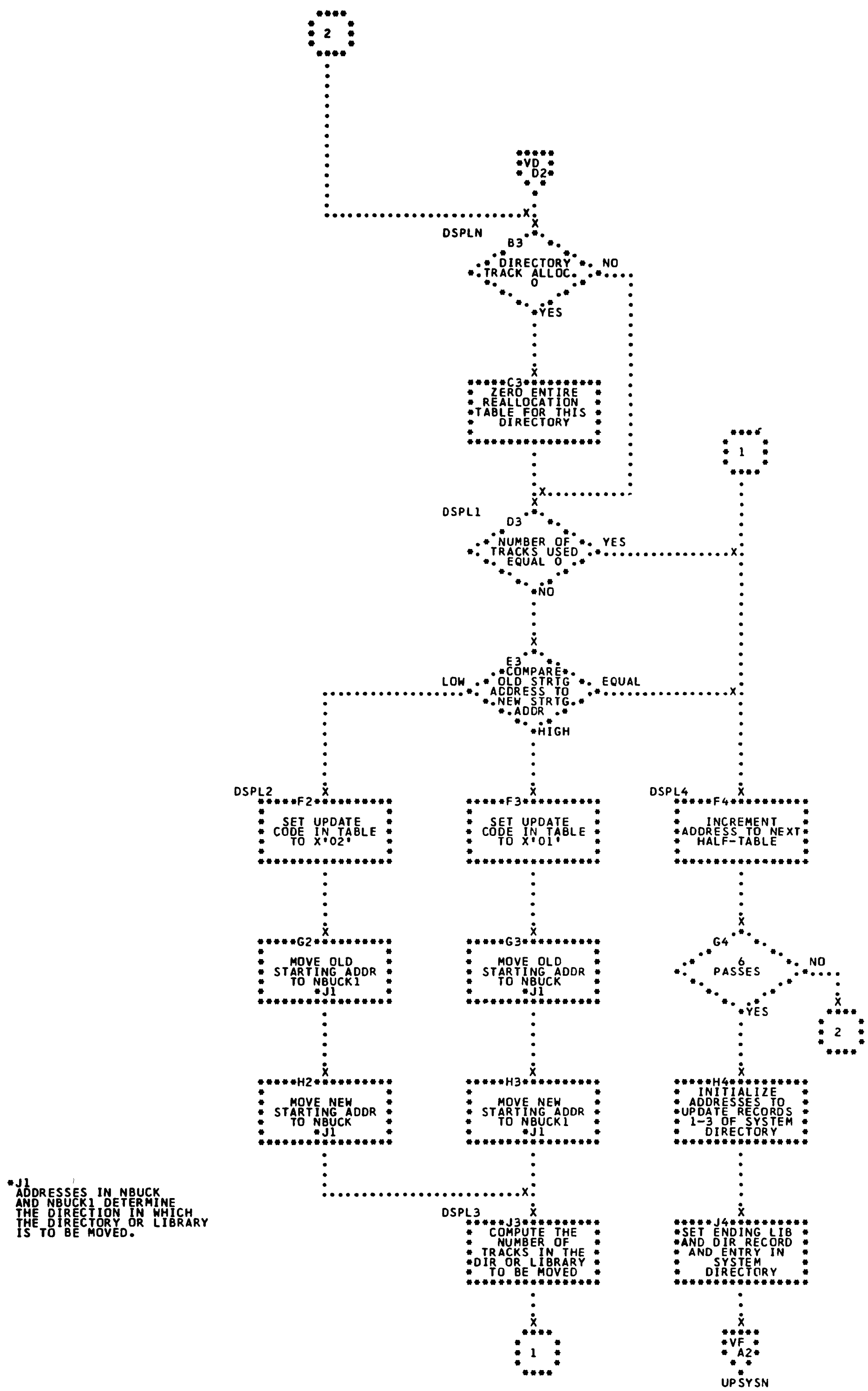


Chart VE. Compute Displacement and Direction for Directory and Library Movement MAINTA; Refer to Maintenance, Chart 43



## Chart VF. Update System Directory Records 1, 2, and 3
MAINTA; Refer to Maintenance, Chart 43

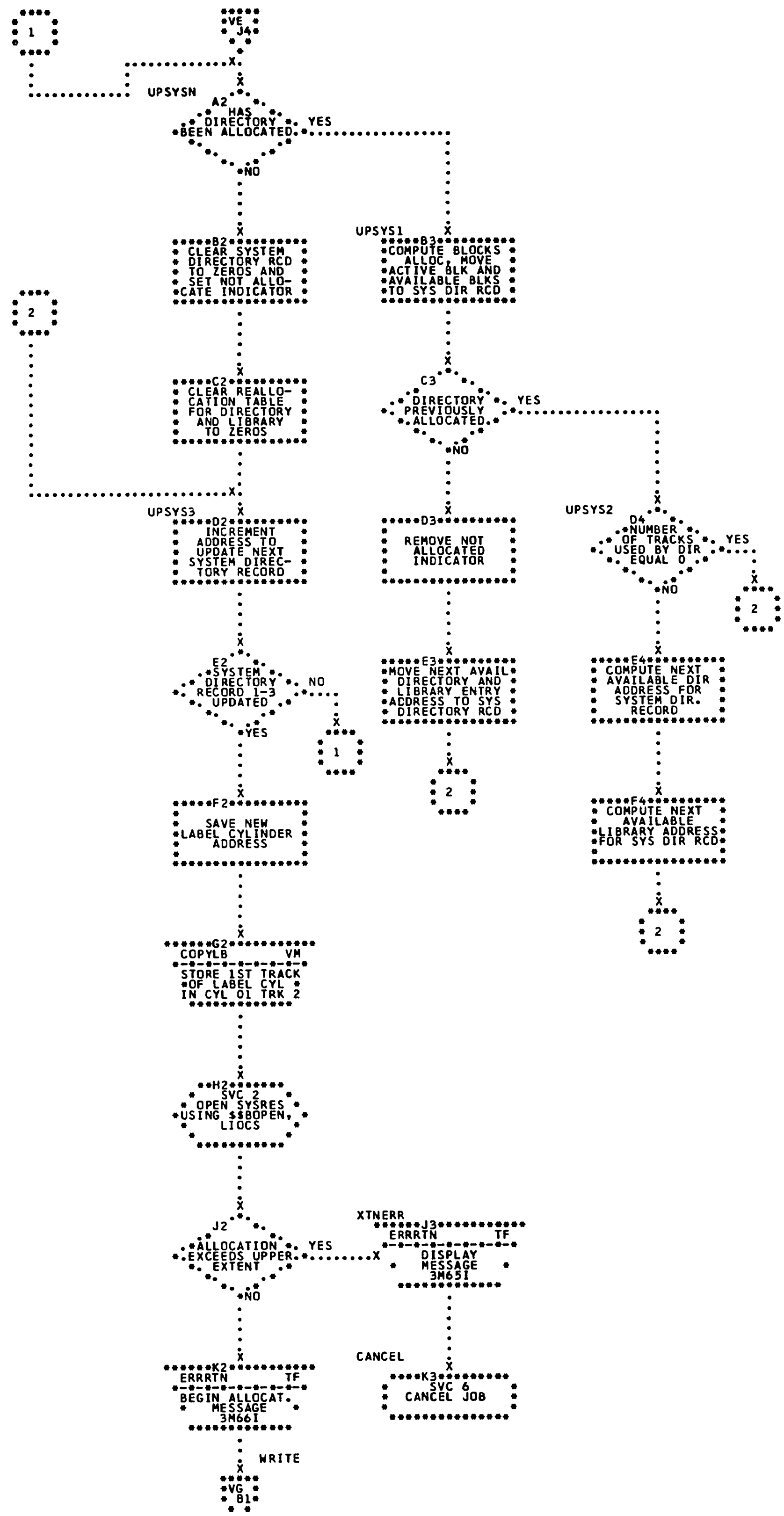


Chart VG. Write Updated System Directory MAINTA; Refer to Maintenance, Chart 43

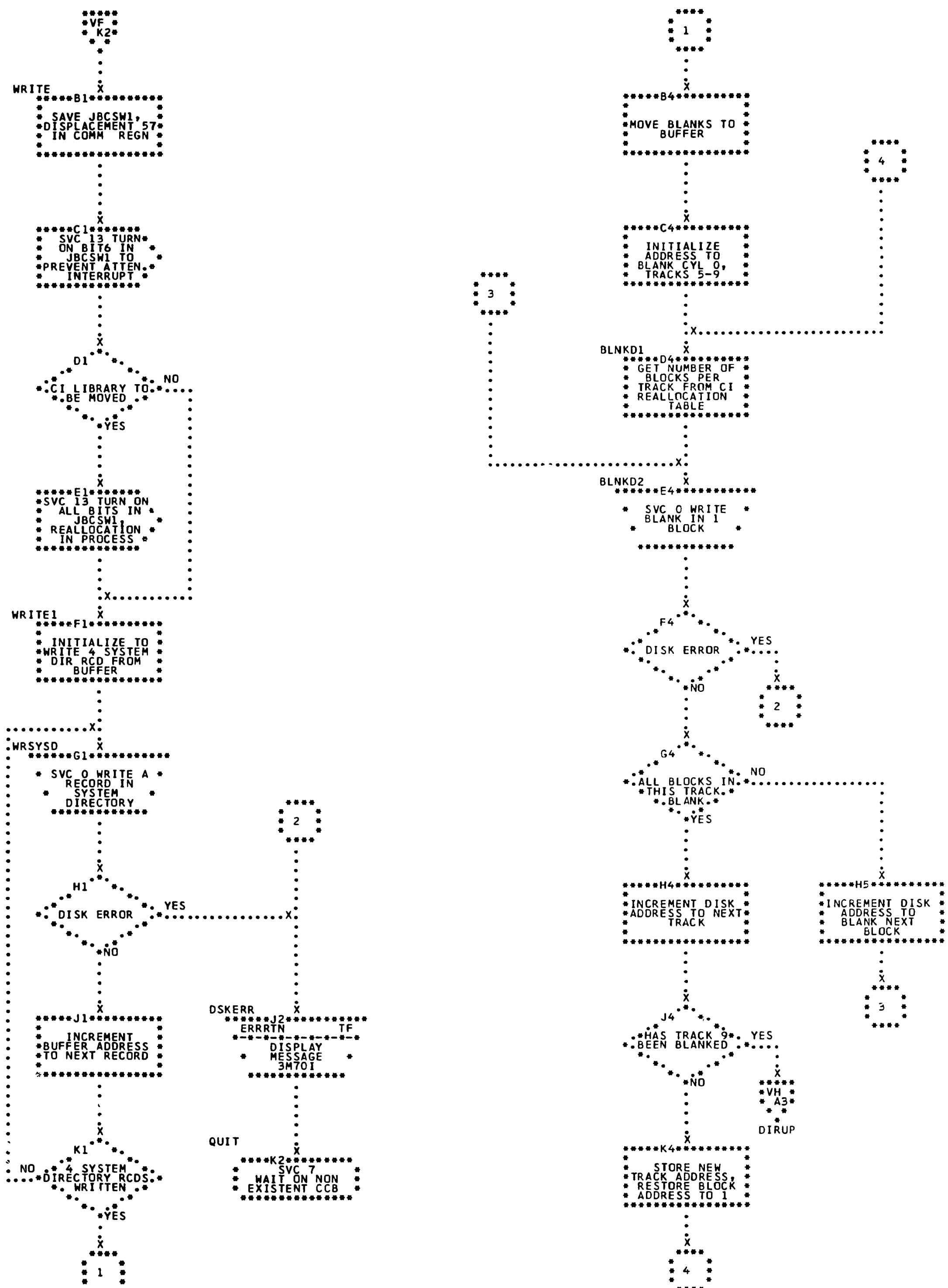


Chart VH. Update Library Directories MAINTA; Refer to Maintenance, Chart 43

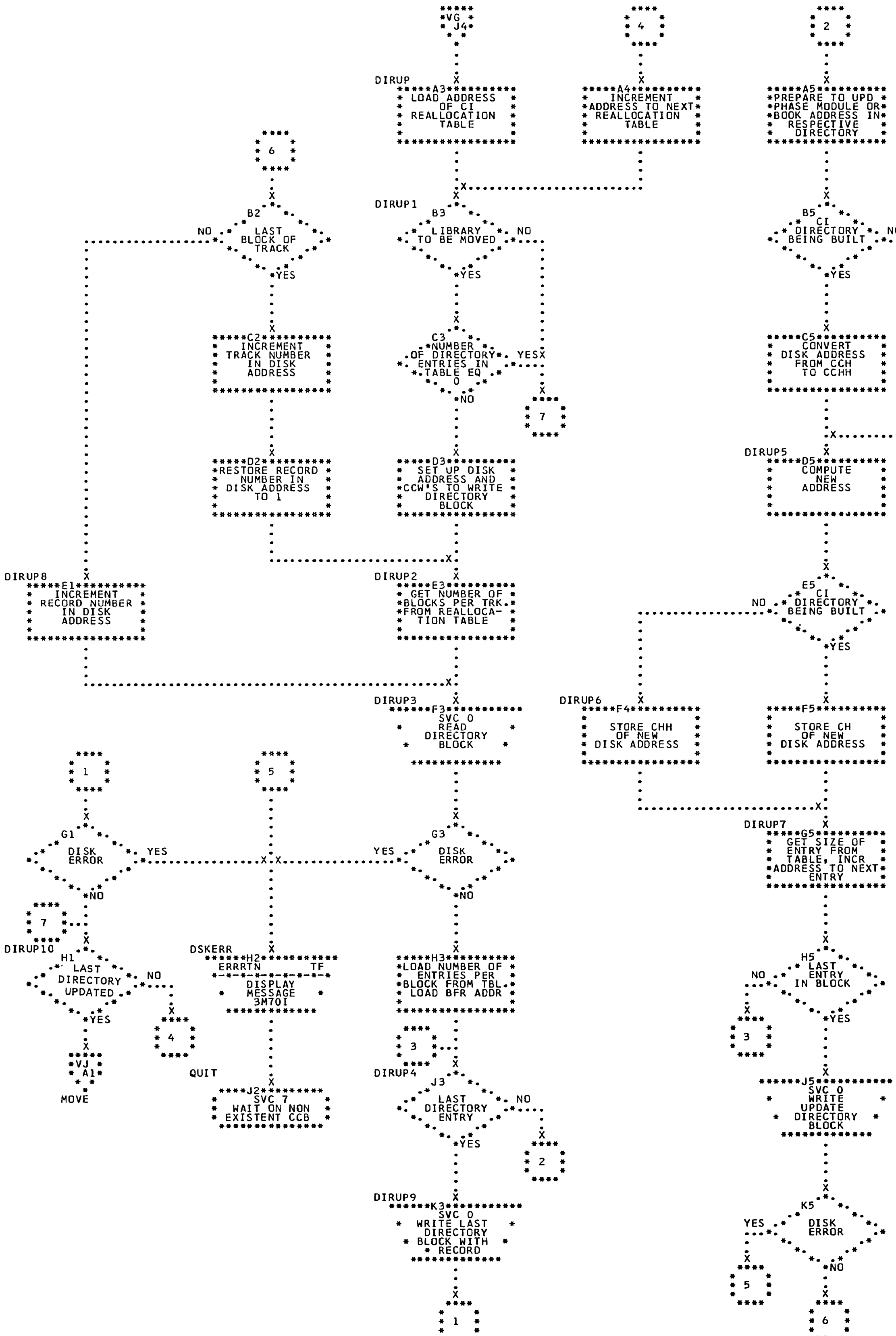
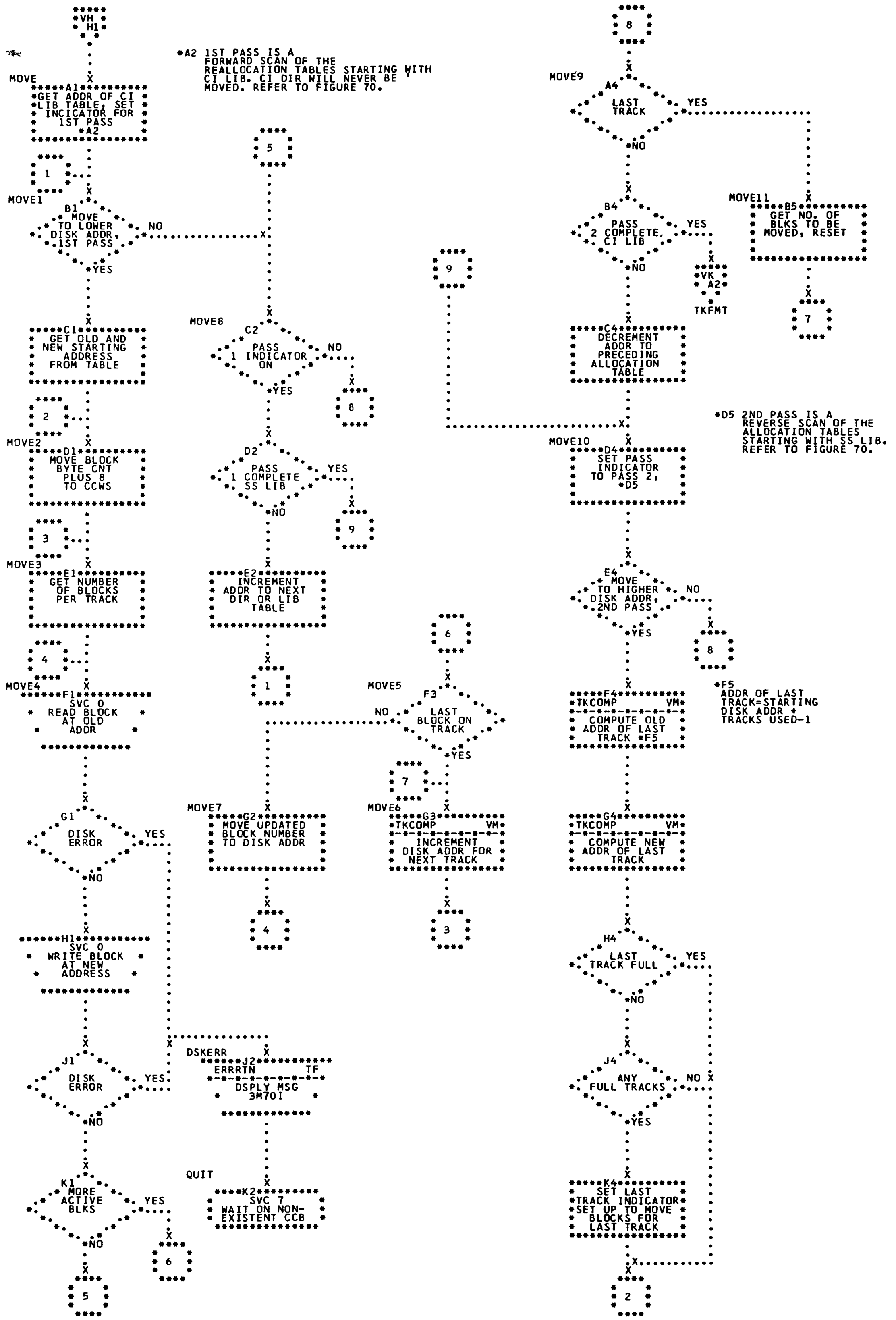


Chart VJ. Relocate Directories and Libraries MAINTA; Refer to Maintenance, Chart 43



## Chart VK. Format Unused Tracks MAINTA; Refer to Maintenance, Chart 43

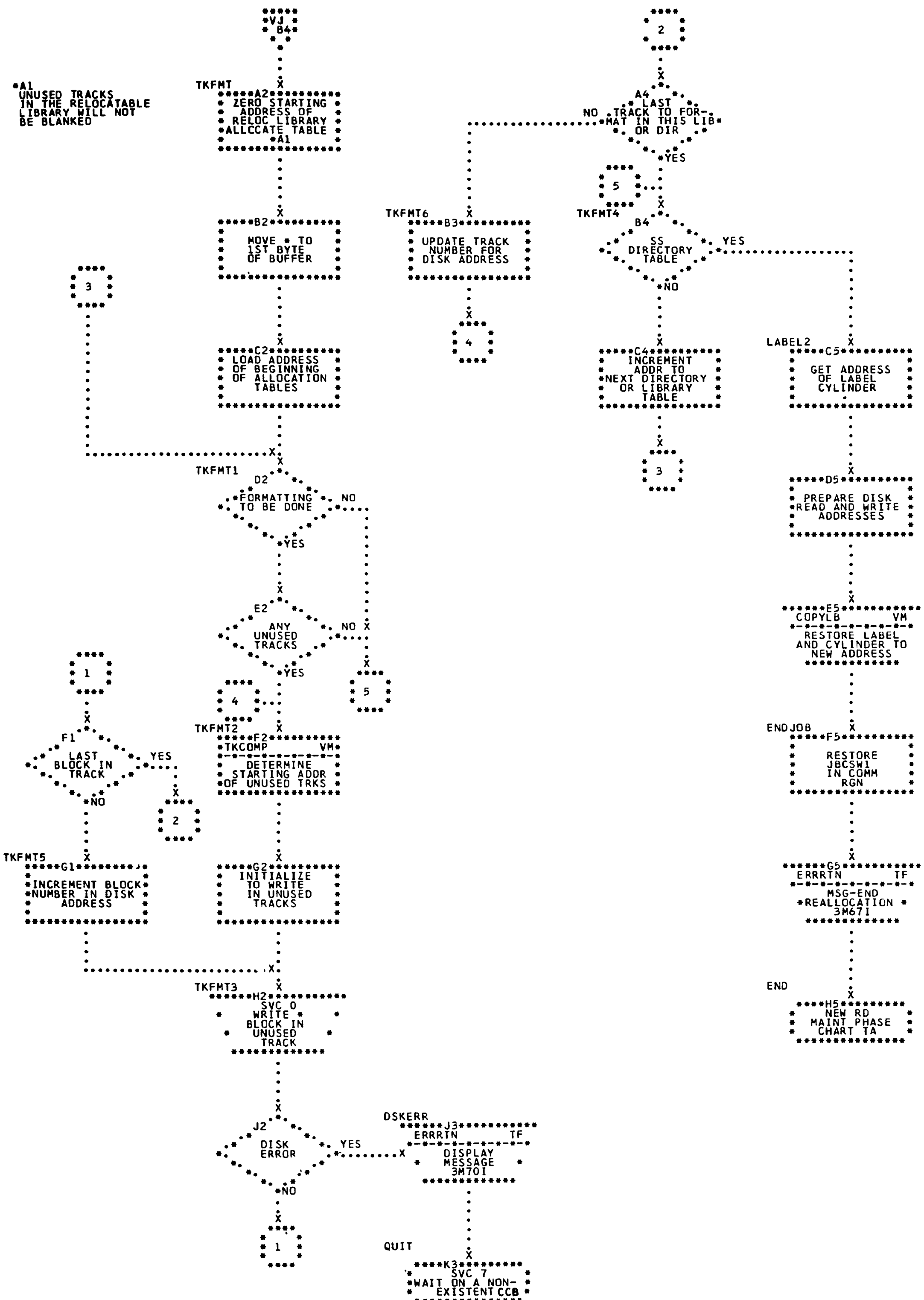


Chart VL. TSTNUM, CONVRT, and UPDATE Subroutines MAINTA; Refer to Maintenance, Chart 43

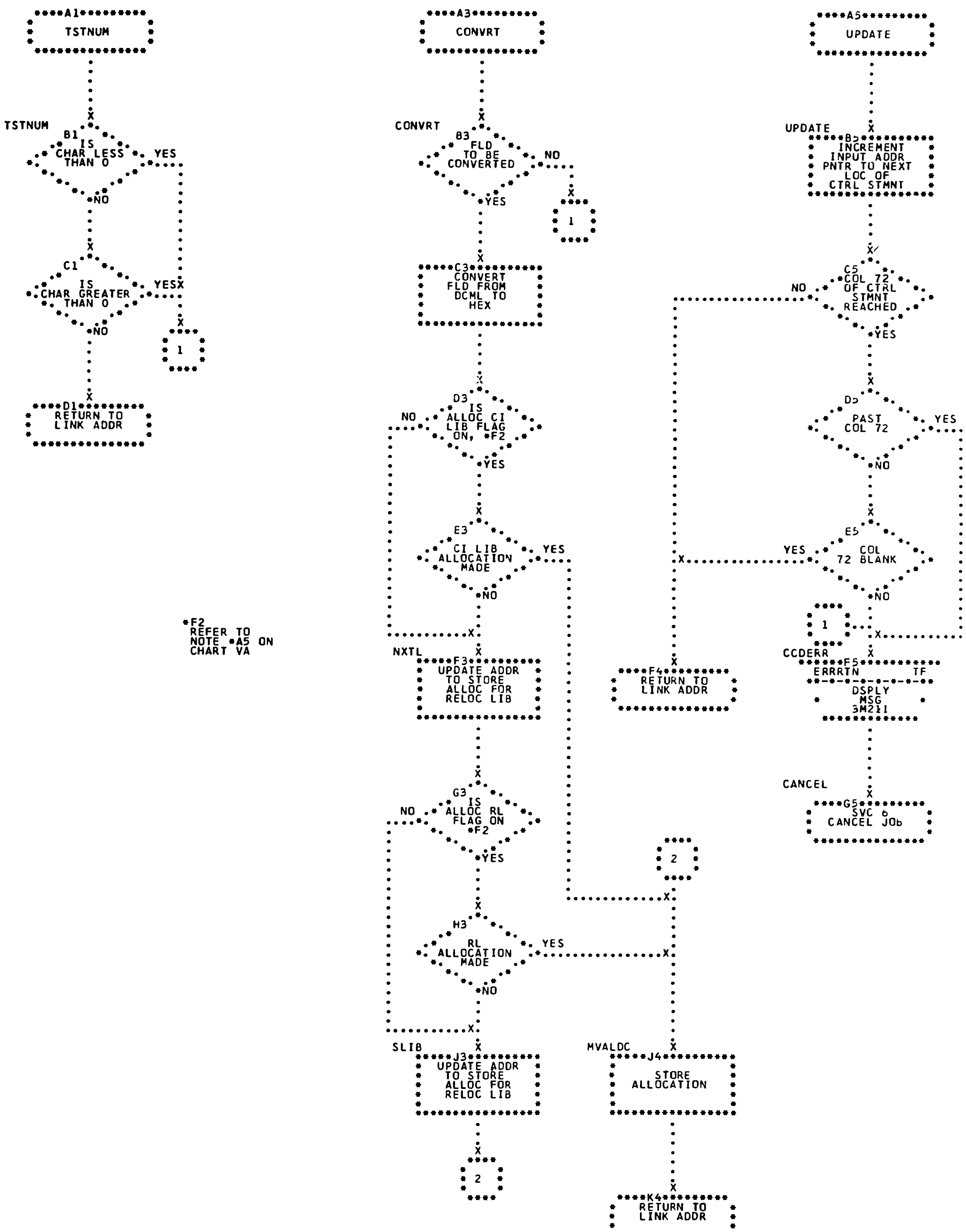


Chart VM. Update Disk Address and Copy Label Track Subroutines MAINTA; Refer to Maintenance, Chart 43

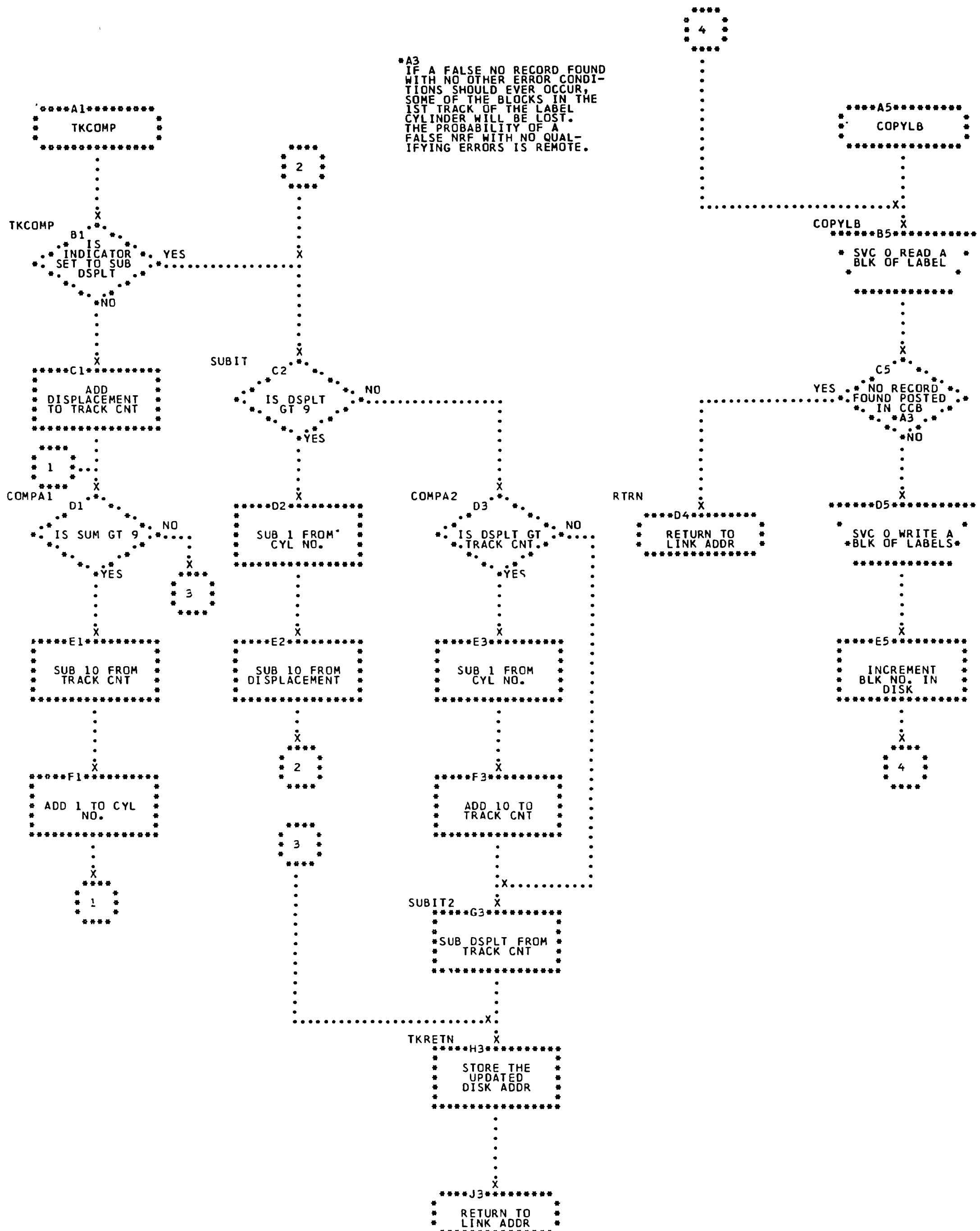


Chart VN. Initialize to Condense a Library MAINTCN; Refer to Maintenance, Chart 44

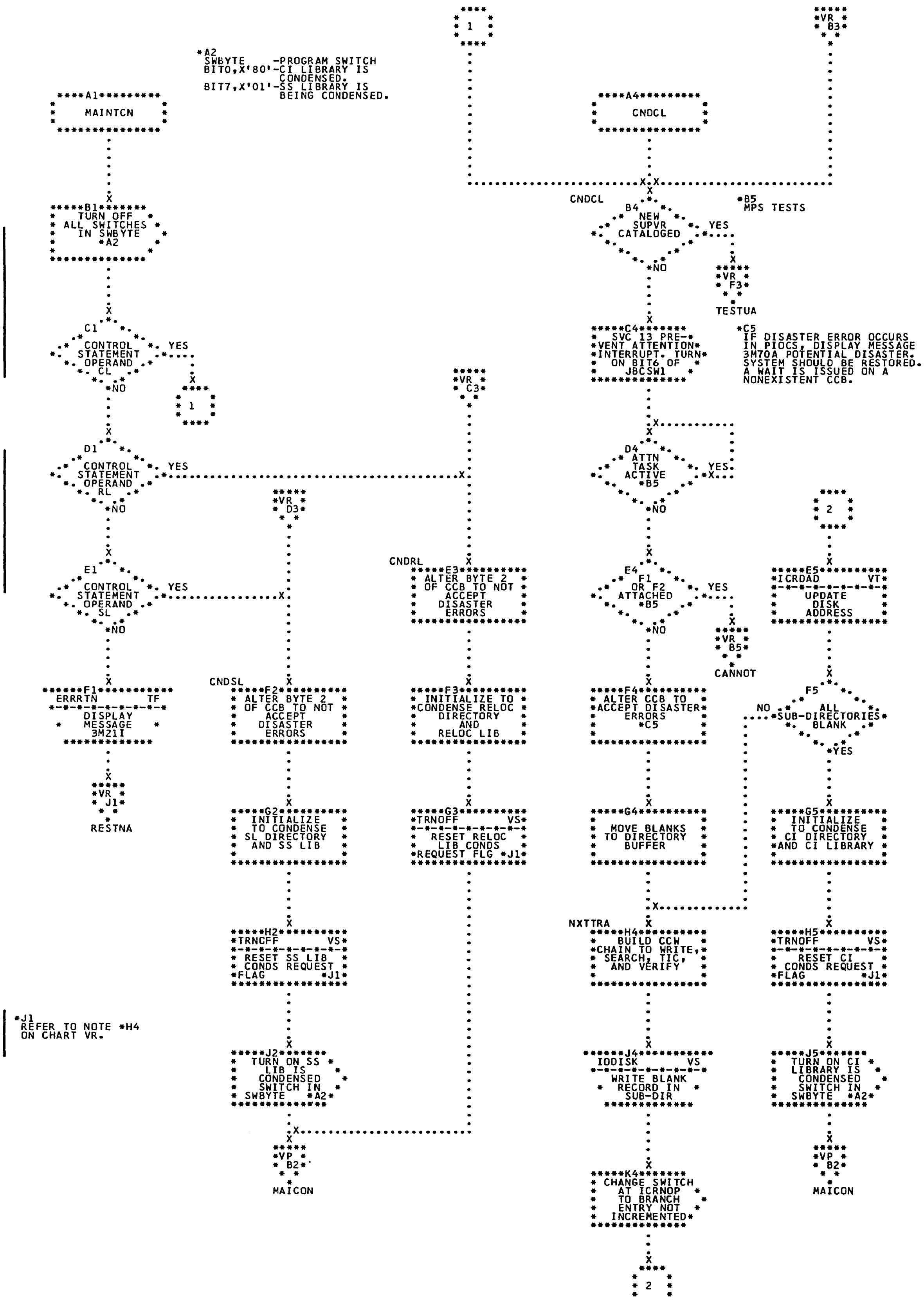


Chart VP. Condense a Directory MAINTCN; Refer to Maintenance, Chart 44

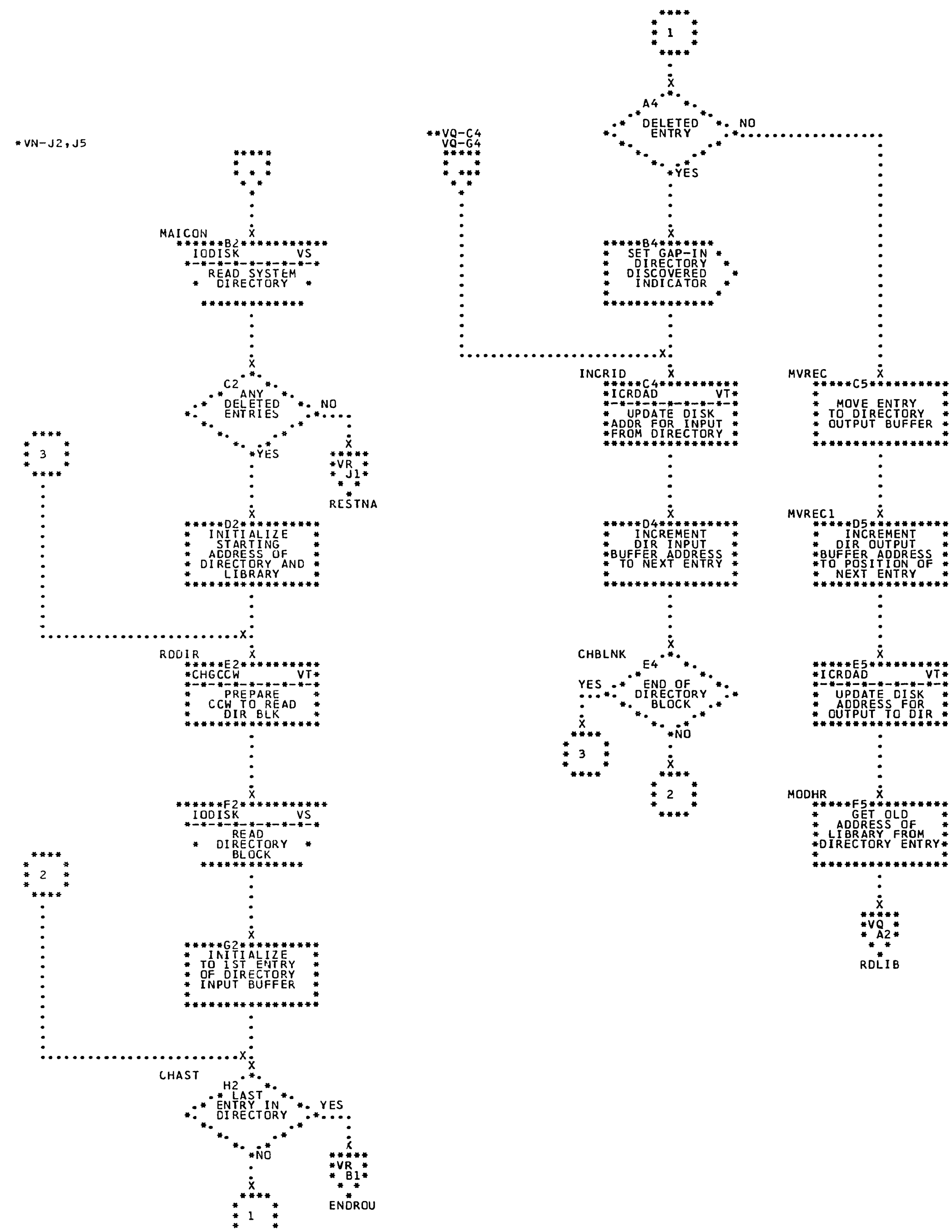


Chart VQ. Condense a Library MAINTCN; Refer to Maintenance, Chart 44

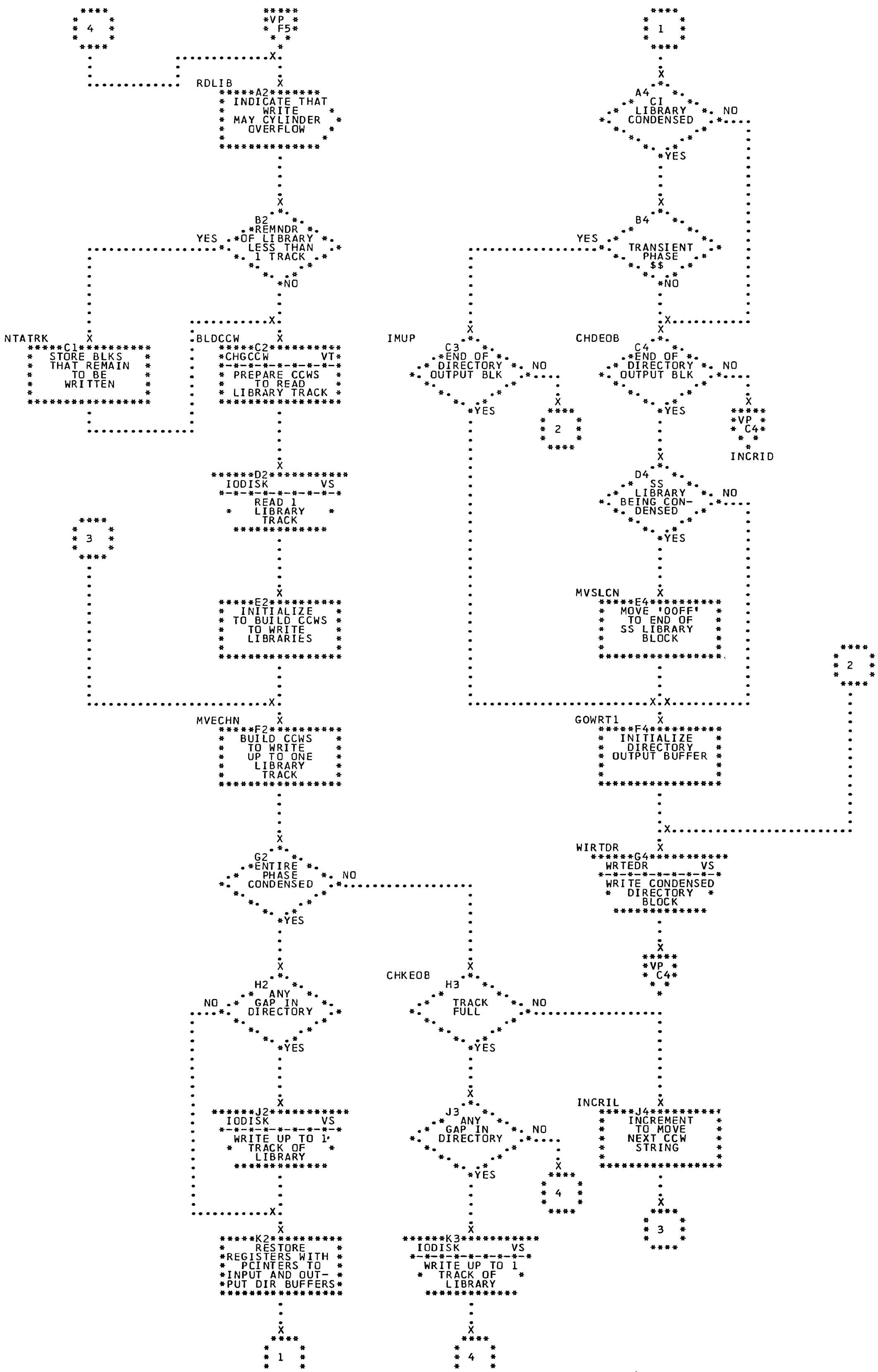


Chart VR. Automatic Condense MAINTCN; Refer to Maintenance, Chart 44

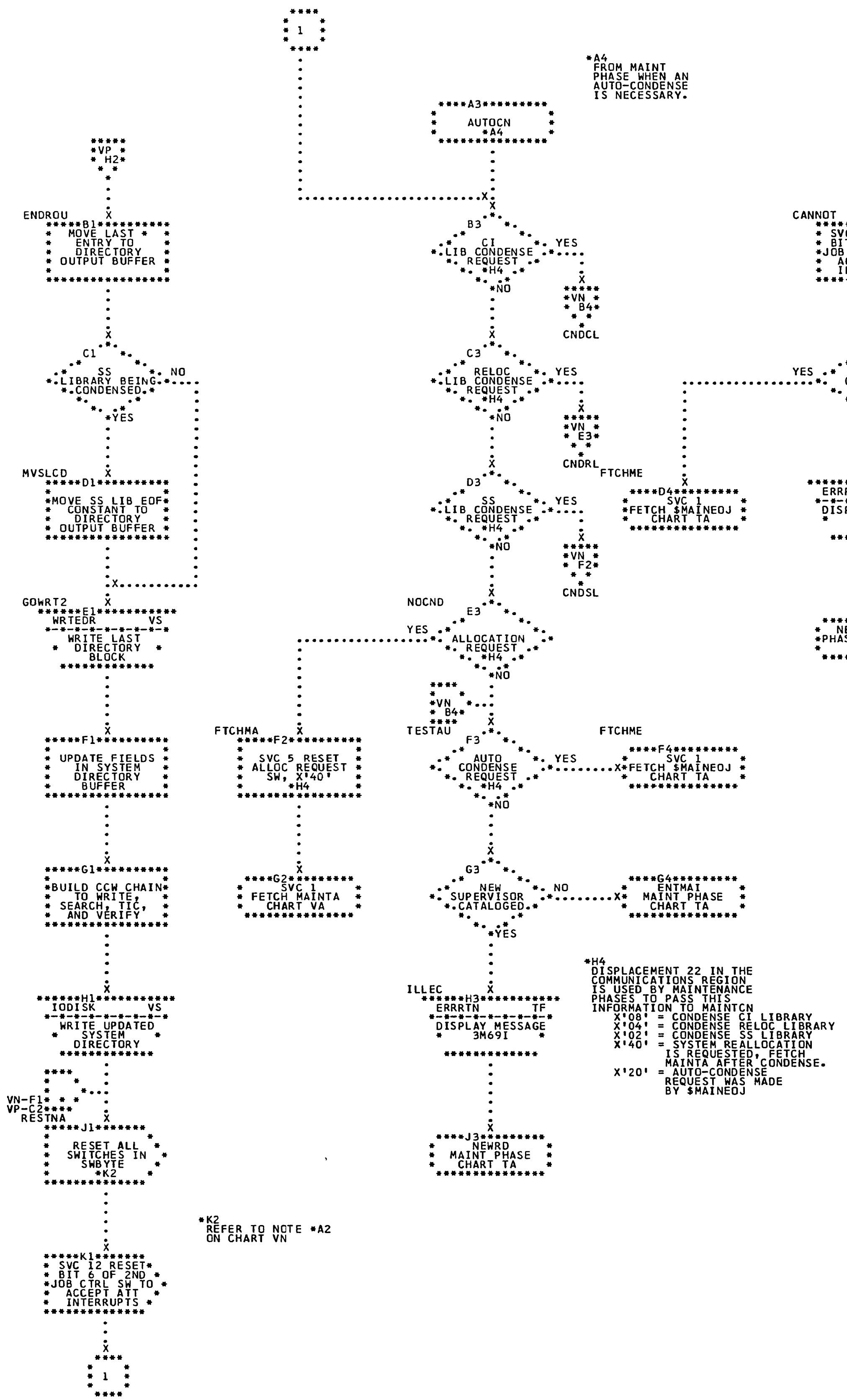


Chart VS. VERILI, IODISK, and WRTEDR Subroutines MAINTCN; Refer to Maintenance, Chart 44

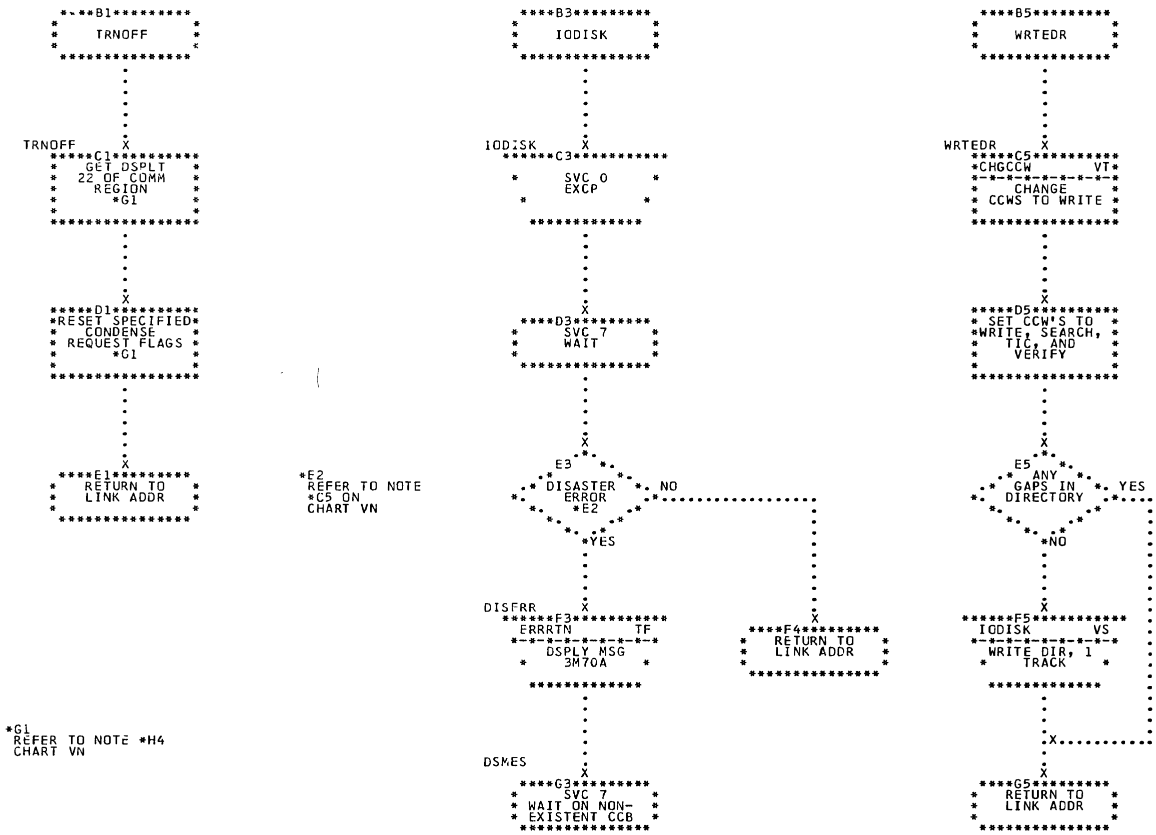


Chart VT. CHGCCW and ICRDAD Subroutines MAINTCN; Refer to Maintenance, Chart 44

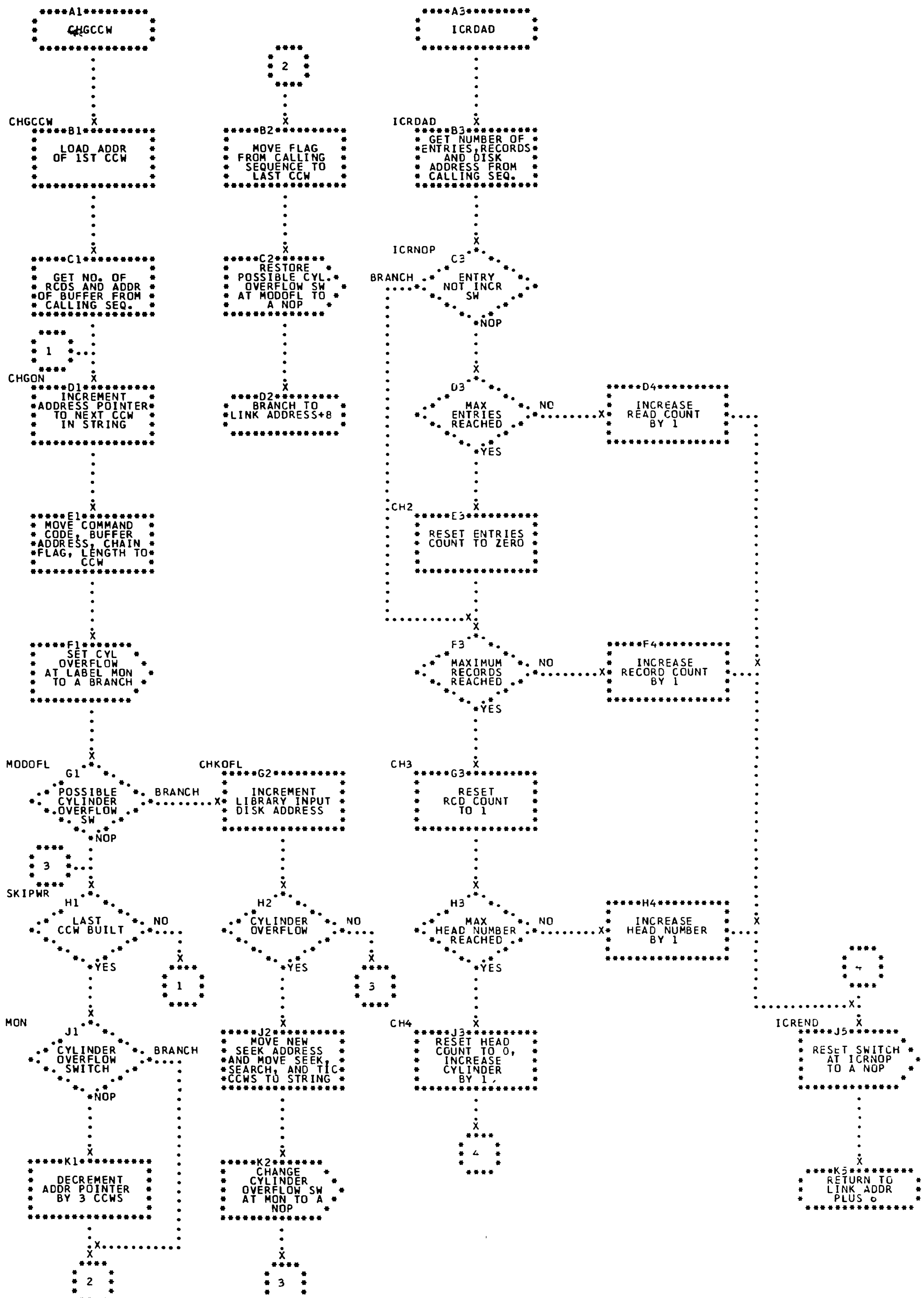


Chart VU. Set Condense Limits MAINTCL; Refer to Maintenance, Chart 45

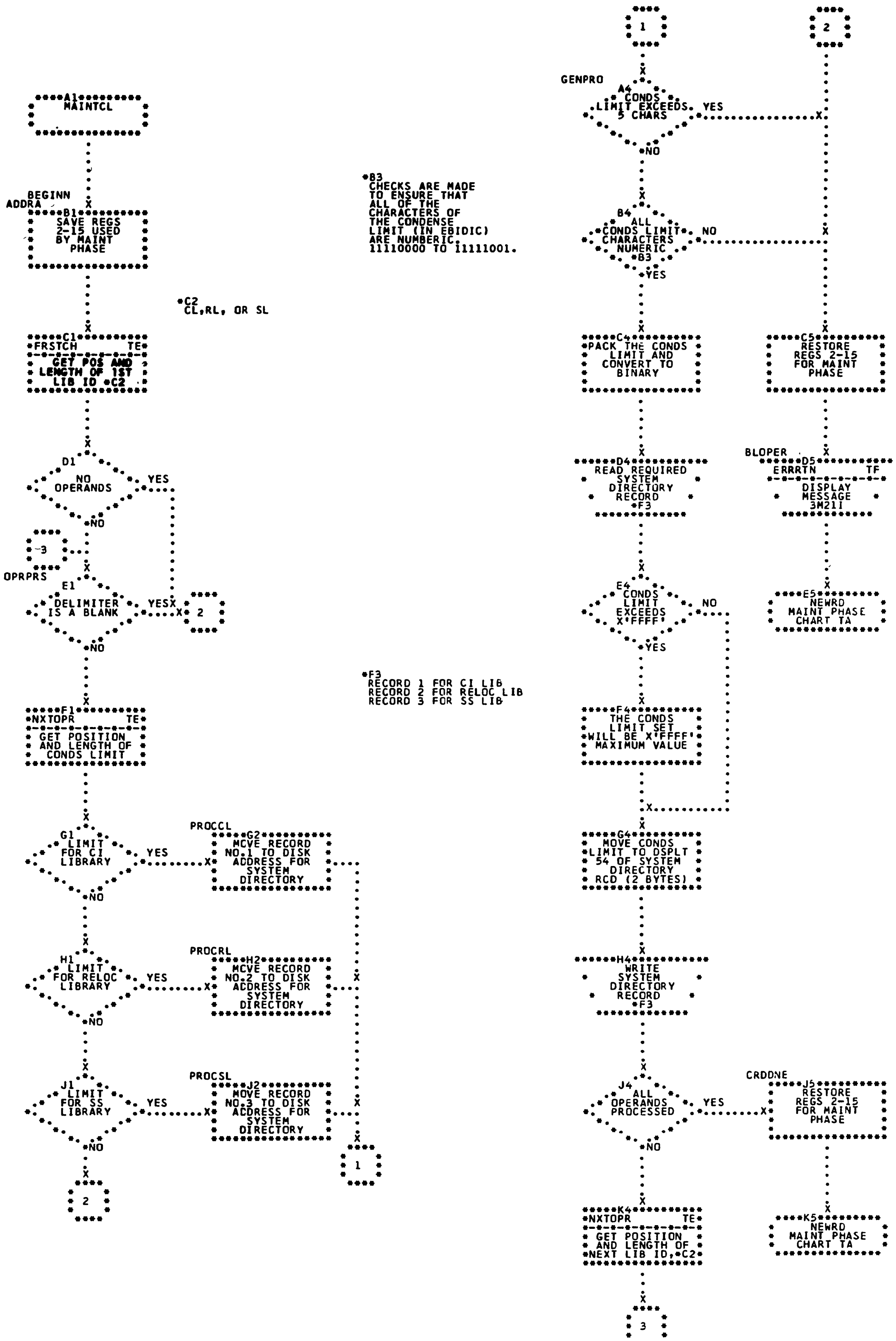


Chart VV. Print System Status Report and Update Subdirectories $MAINEOJ (Part 1 of 3); Refer to Maintenance, Chart 45

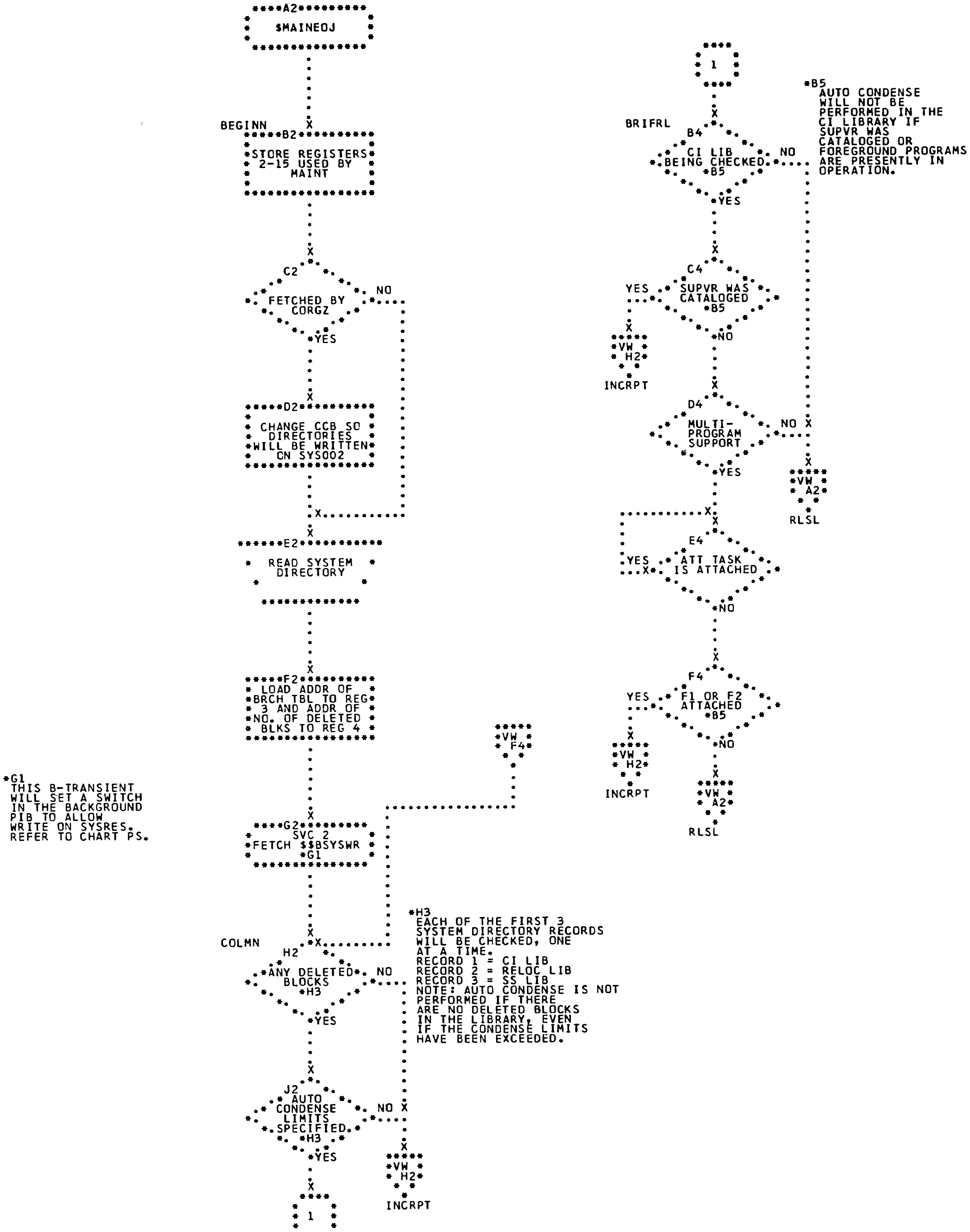


Chart VW. Print System Status Report and Update Subdirectories $MAINEOJ (Part 2 of 3); Refer to Maintenance, Chart 45

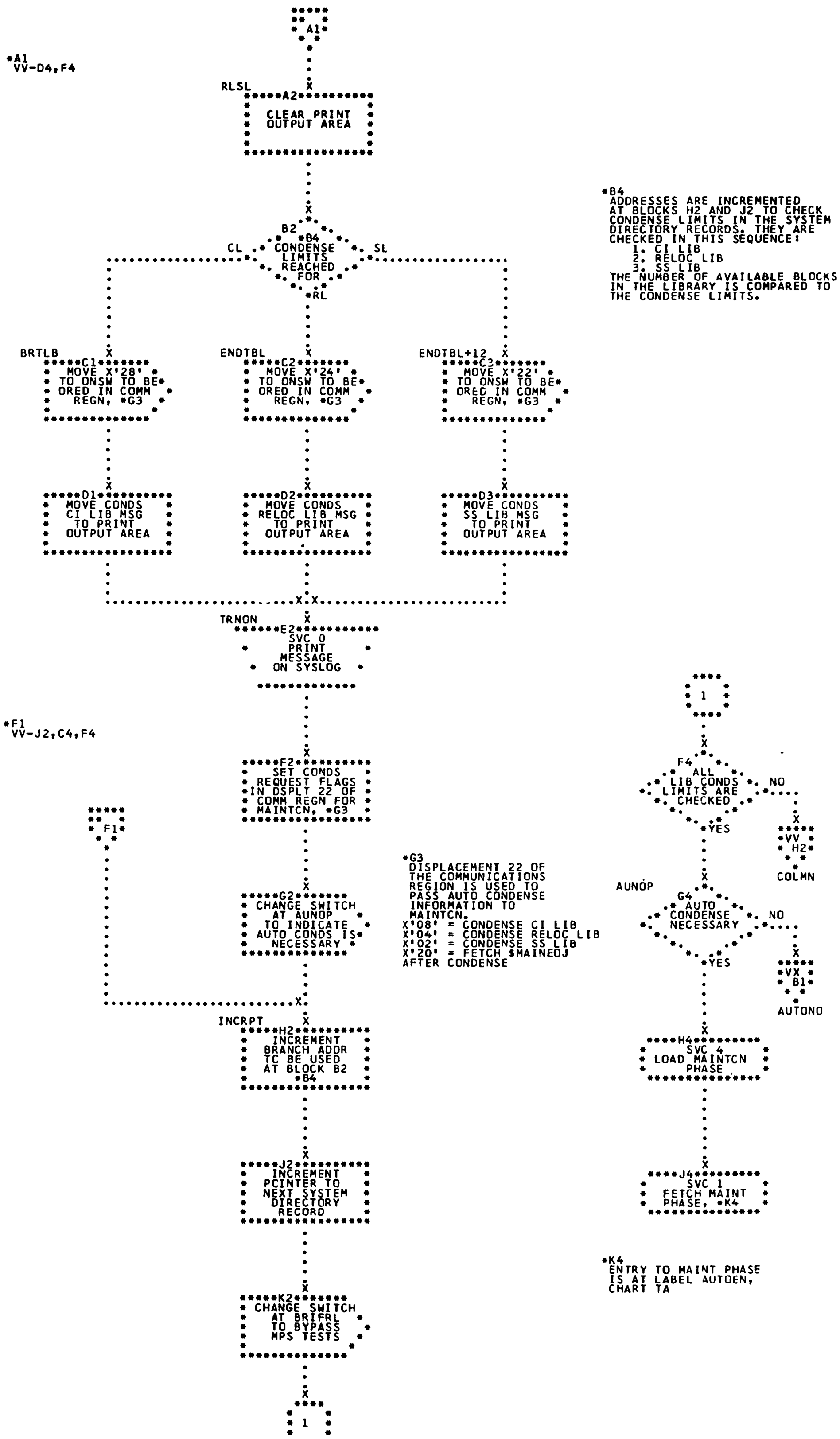


Chart VX. Print System Status Report and Update Subdirectories $MAINEOJ (Part 3 of 3); Refer to Maintenance, Chart 45

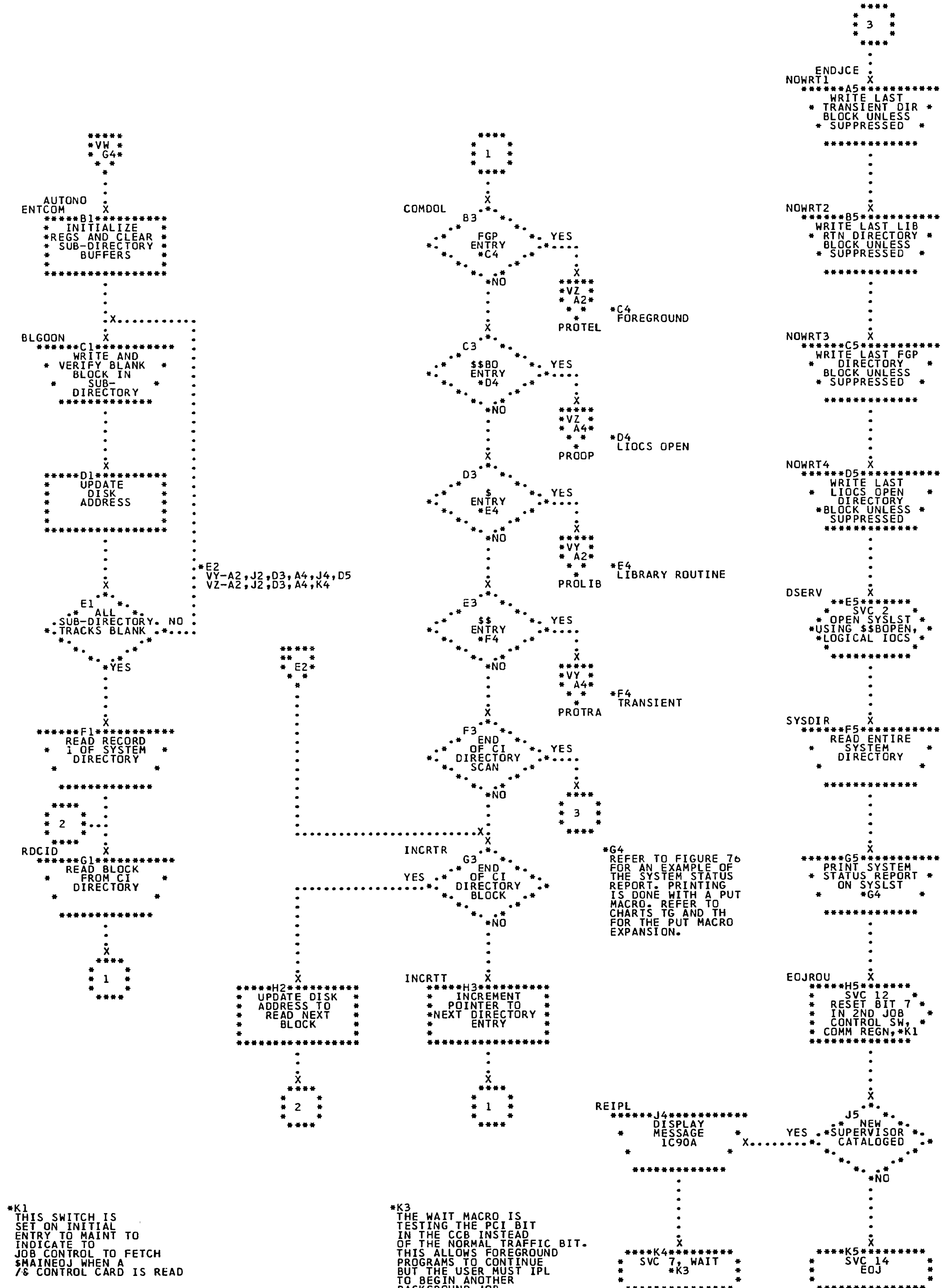


Chart VY. Build Library Routine and Transient Subdirectory Blocks $MAINEOJ; Refer to Maintenance, Chart 45

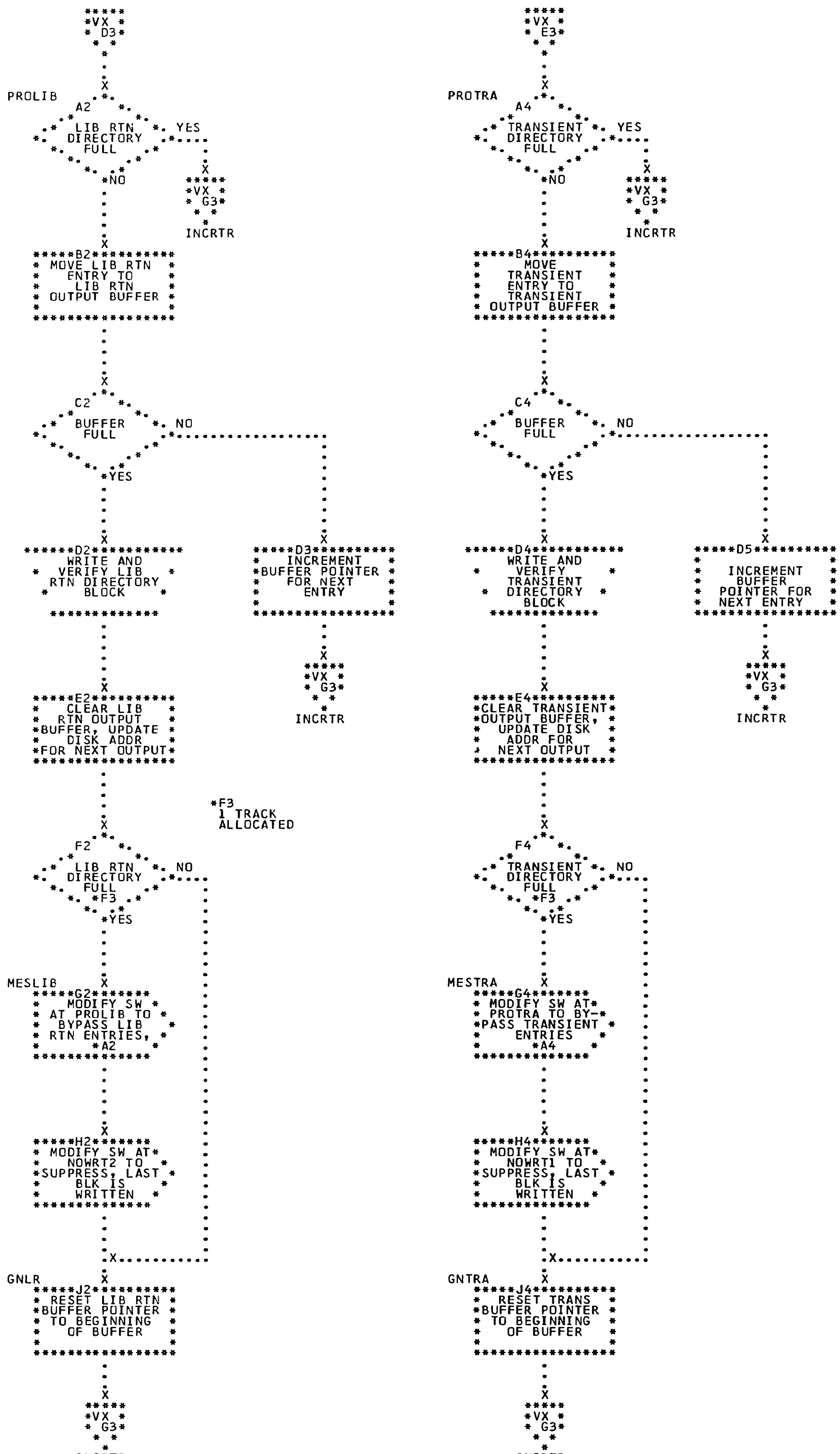
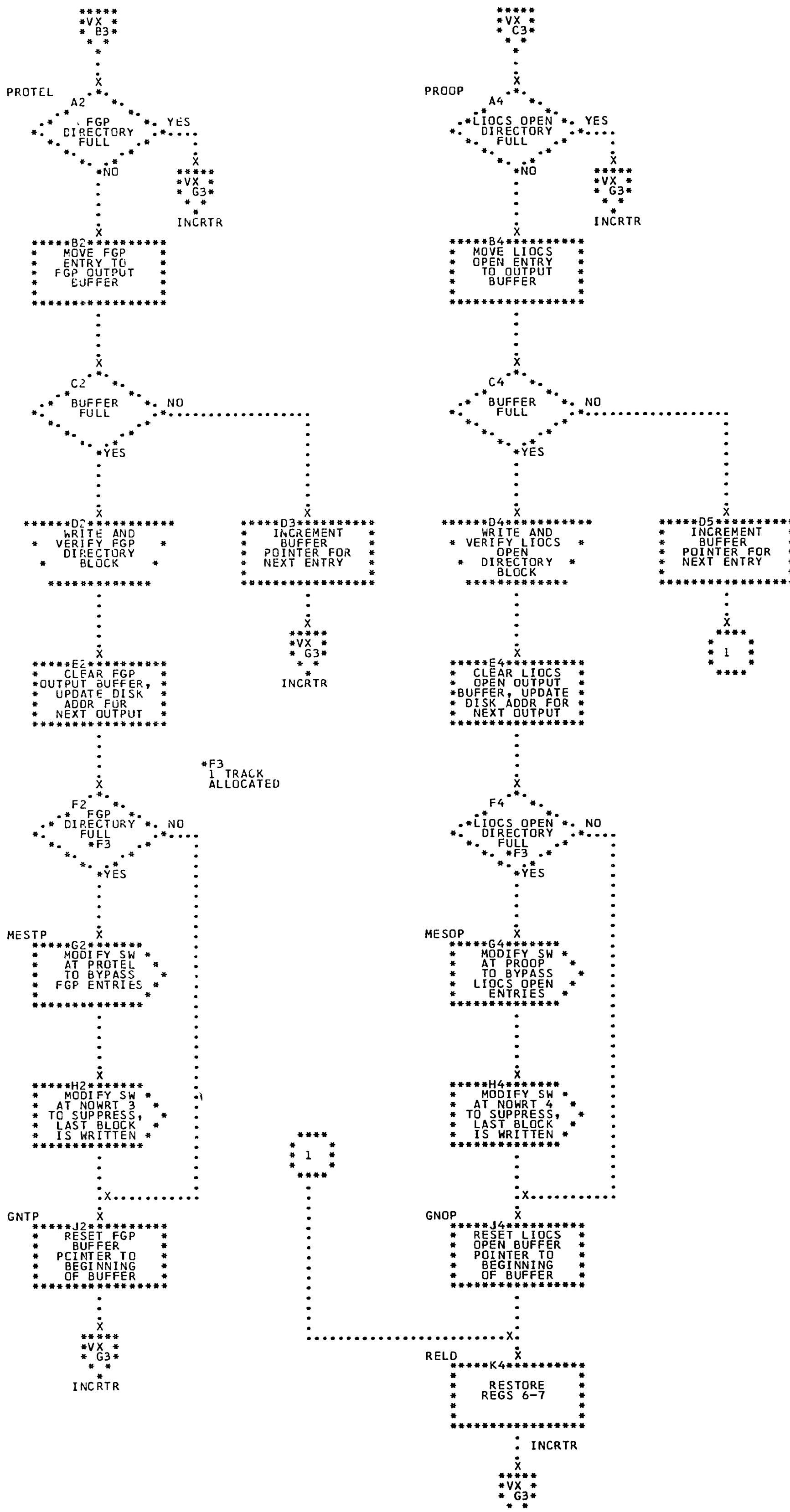


Chart VZ. Build FGP and LIOCS Open Subdirectory Blocks
$MAINEOJ; Refer to Maintenance, Chart 45



## Chart WA. Initialize Phase 1, Copy IPL, and Format Cylinder 0 of SYS002 CORGZ; Refer to Organization, Chart 46

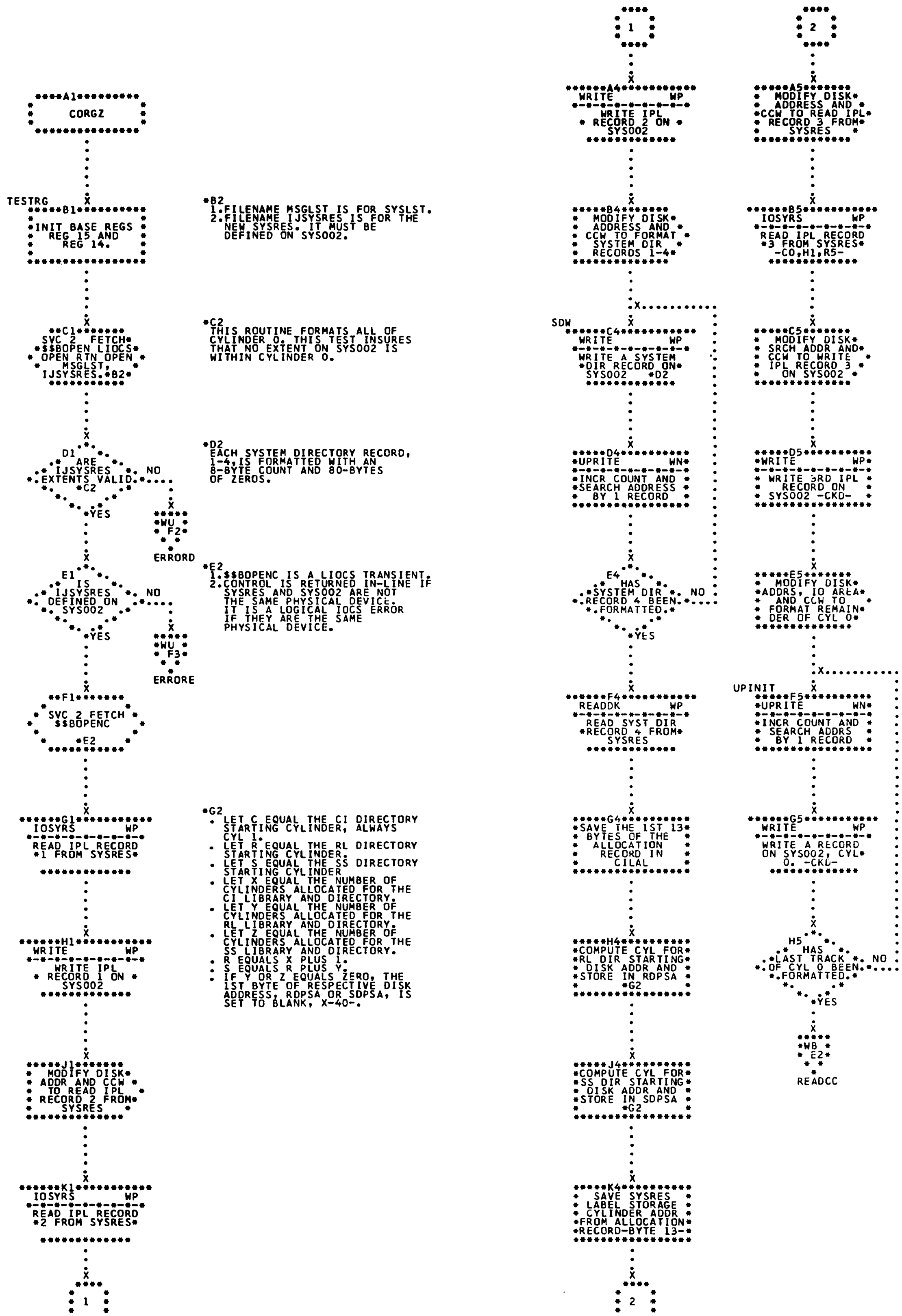


Chart WB. Read and Analyze Control Statement, Write System Directory Records CORGZ; Refer to Organization, Chart 46

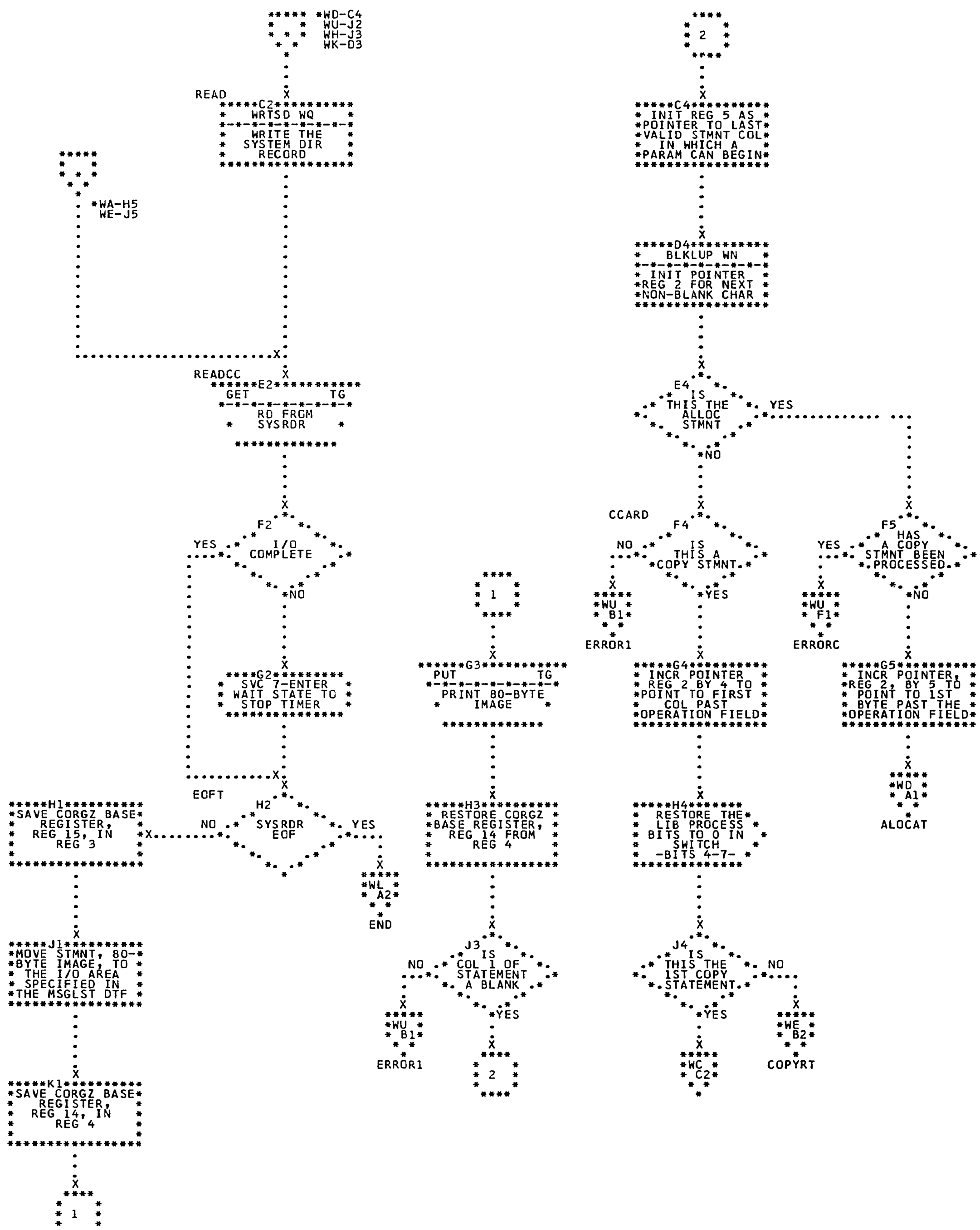


Chart WC. Build SYS002 System Directory Information CORGZ; Refer to Organization, Chart 46

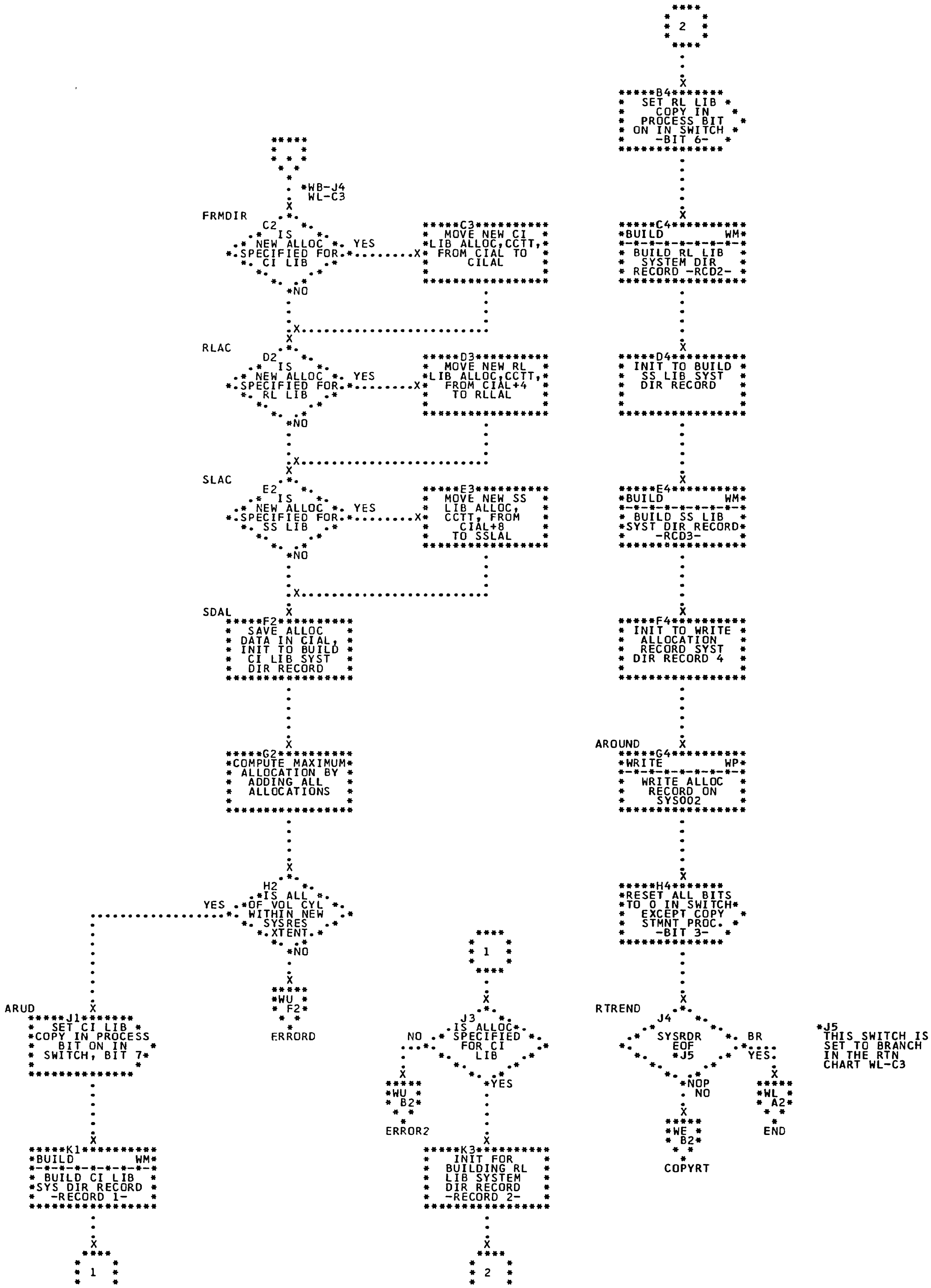


Chart WD. Process ALLOC Control Statement CORGZ; Refer to Organization, Chart 46

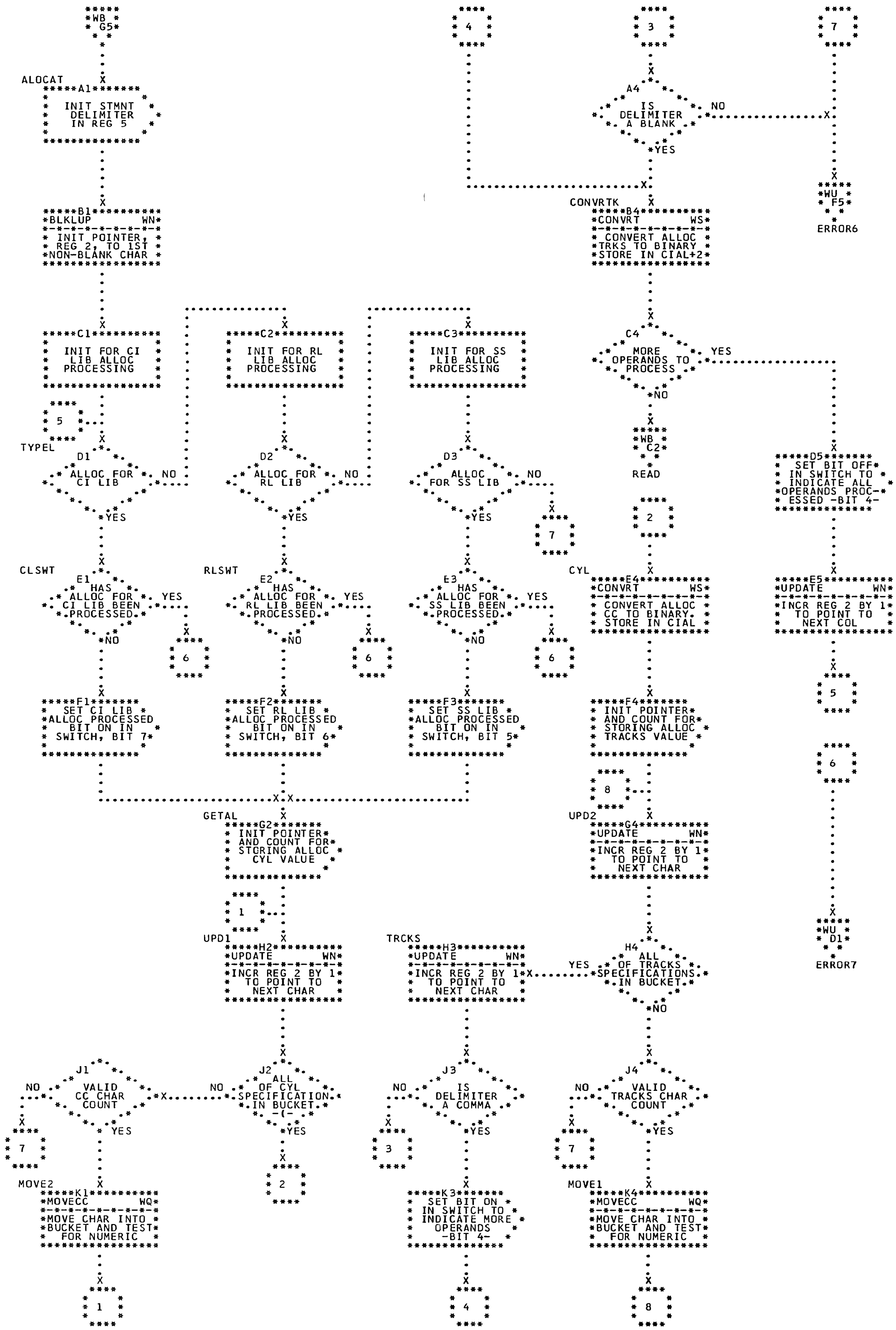


Chart WE. Analyze Copy Statement Type CORGZ; Refer to Organization, Chart 47

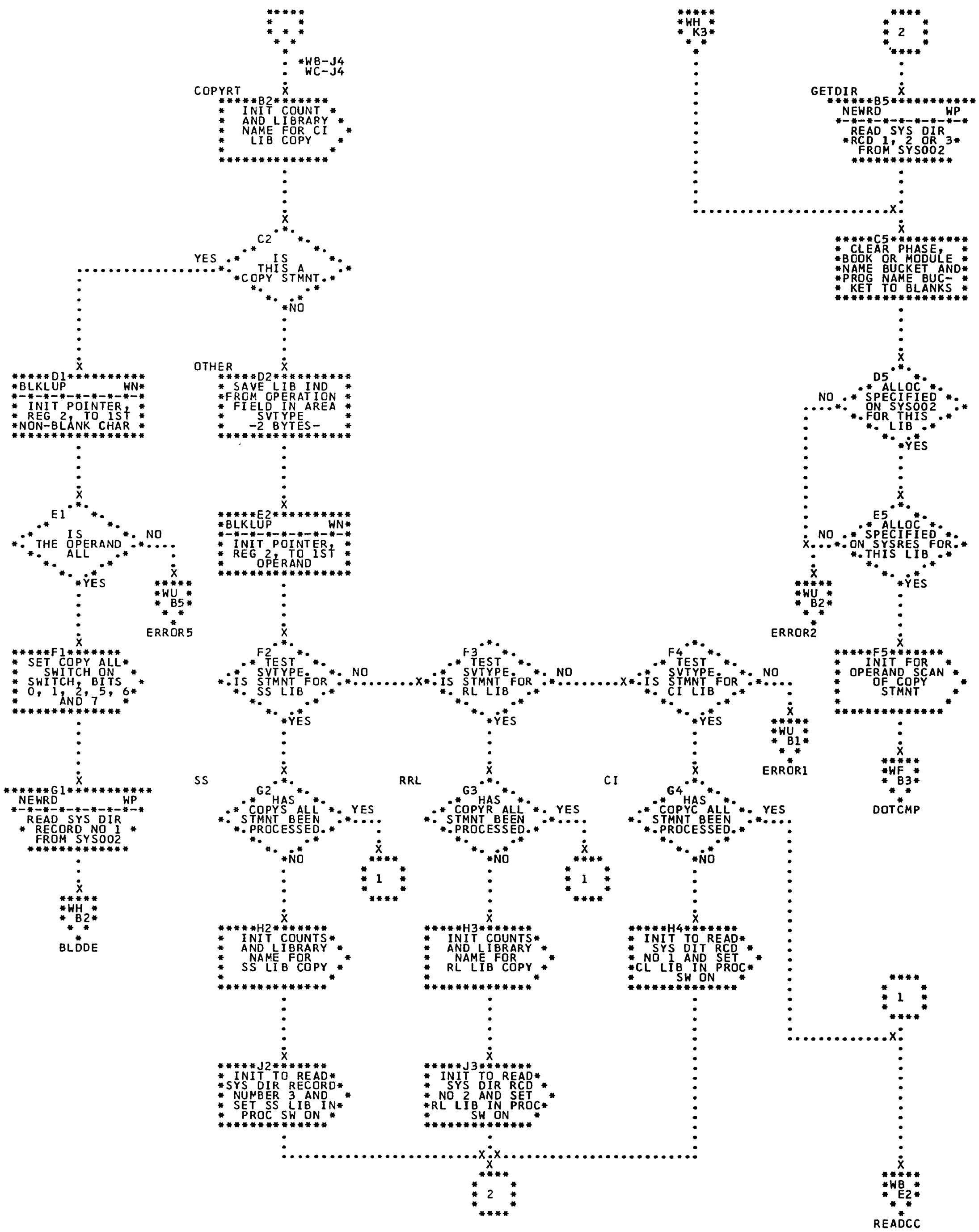


Chart WF. Scan Copy Statement Operands CORGZ; Refer to Organization, Chart 47

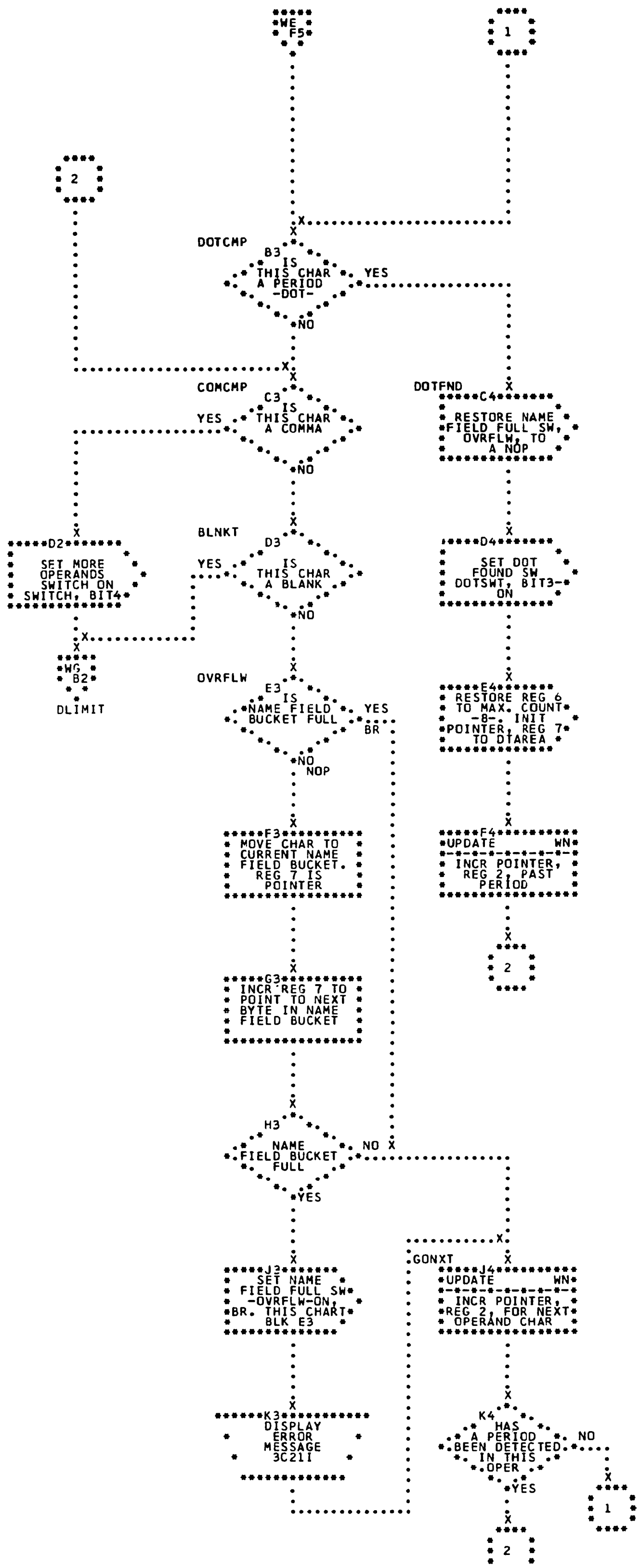


Chart WG. Initialize to Build Library Directories on SYS002
CORGZ; Refer to Organization, Chart 47

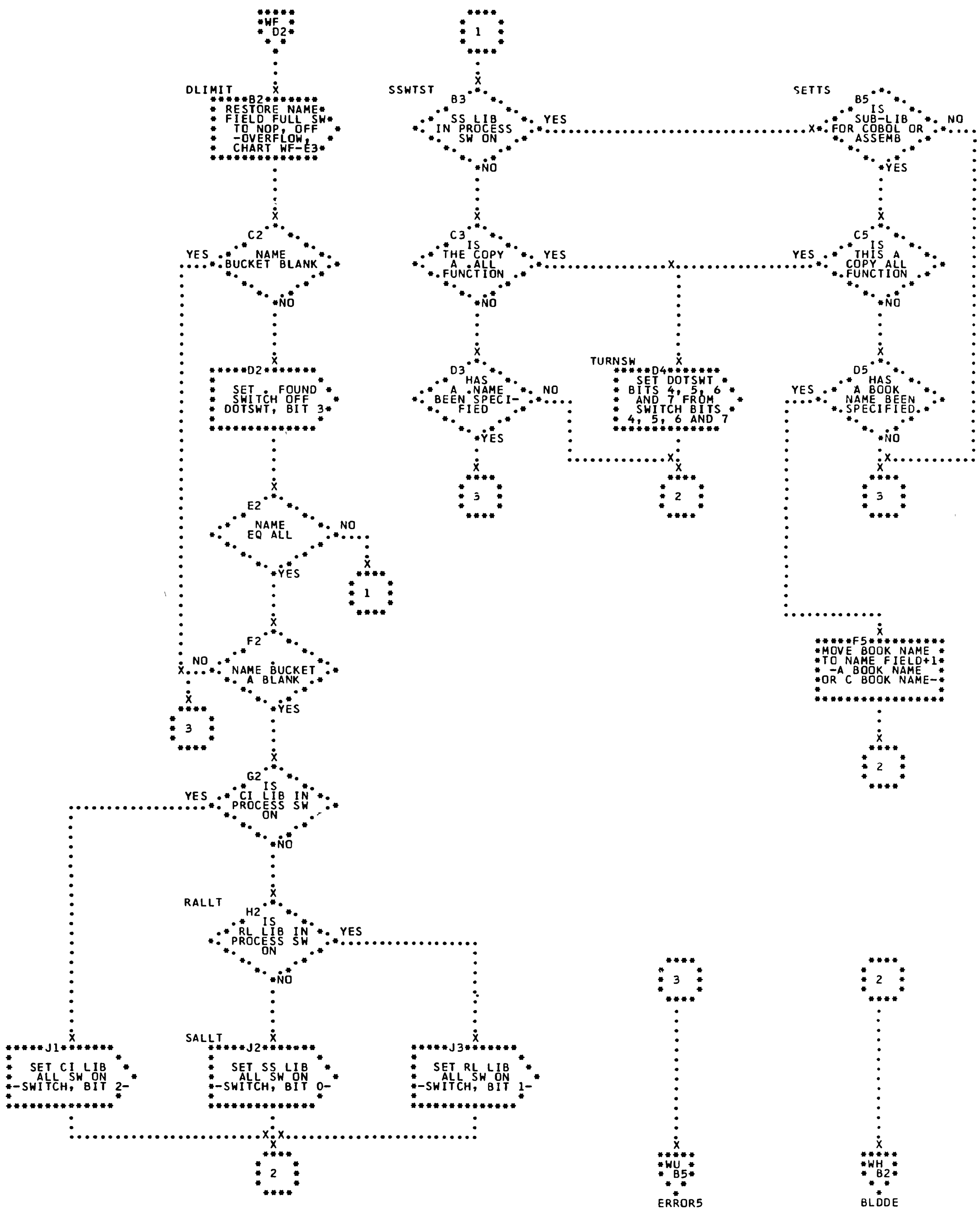


Chart WH. Build Core Image Library Directory on SYS002
CORGZ; Refer to Organization, Chart 47

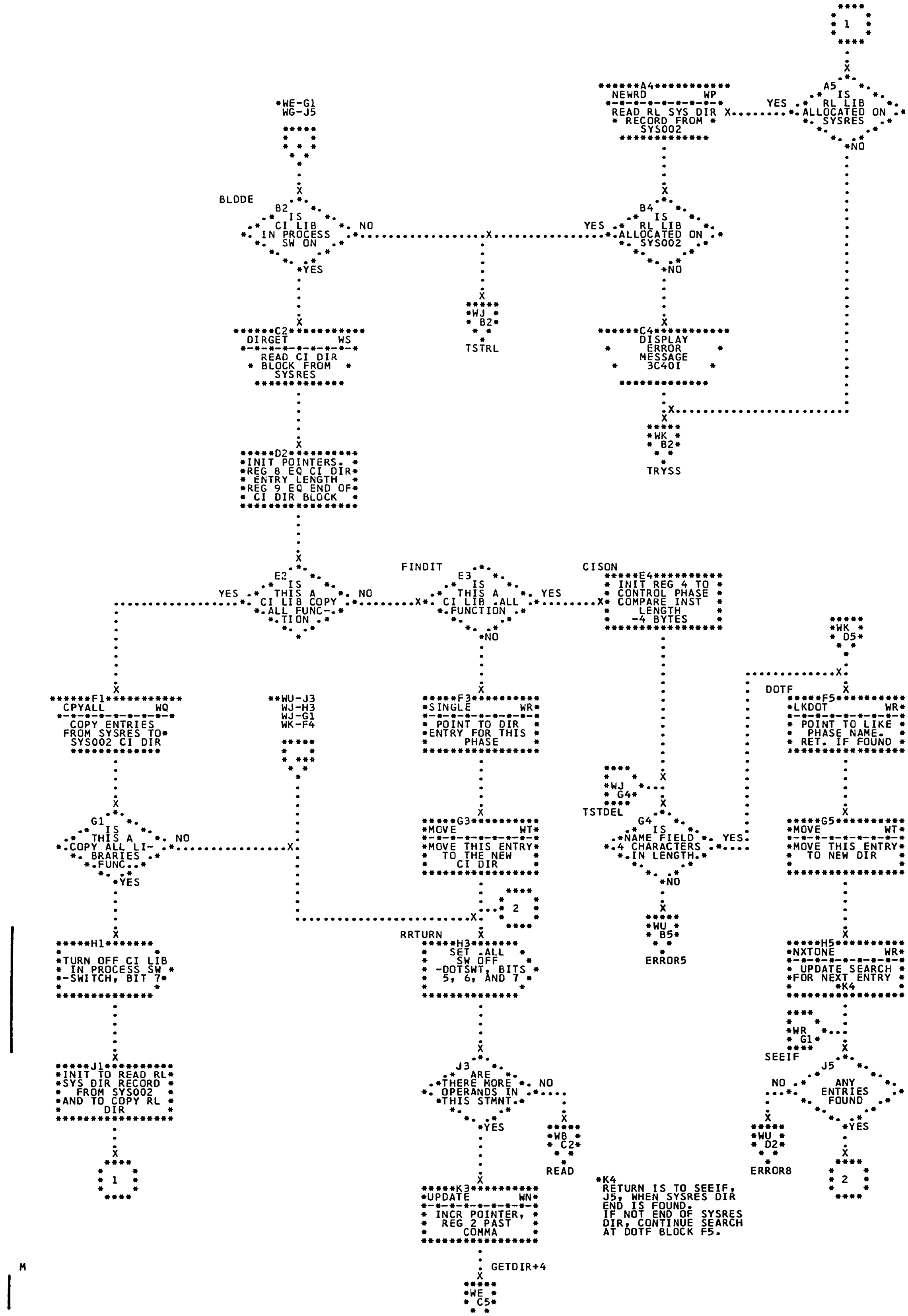


Chart WJ. Build Relocatable Library Directory on SYS002
CORGZ; Refer to Organization, Chart 47

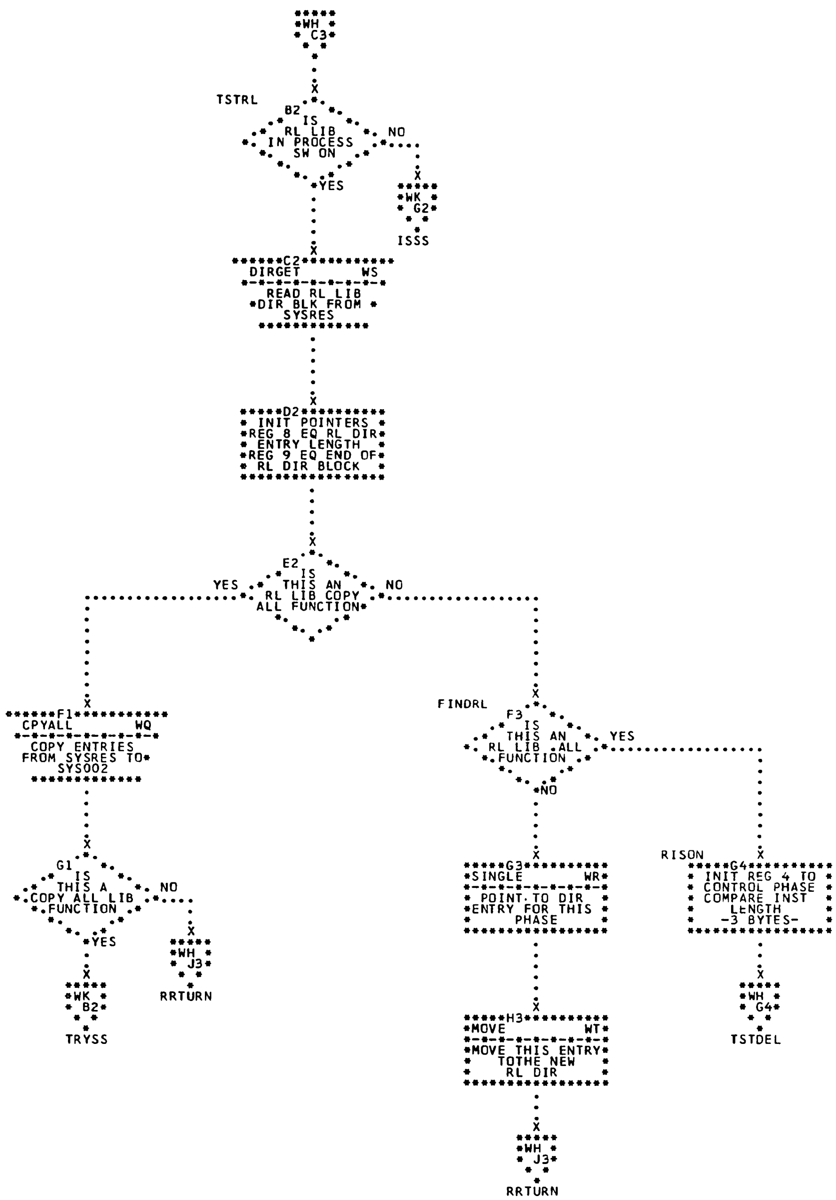


Chart WK. Build Source Statement Library Directory on SYS002 CORGZ; Refer to Organization, Chart 47

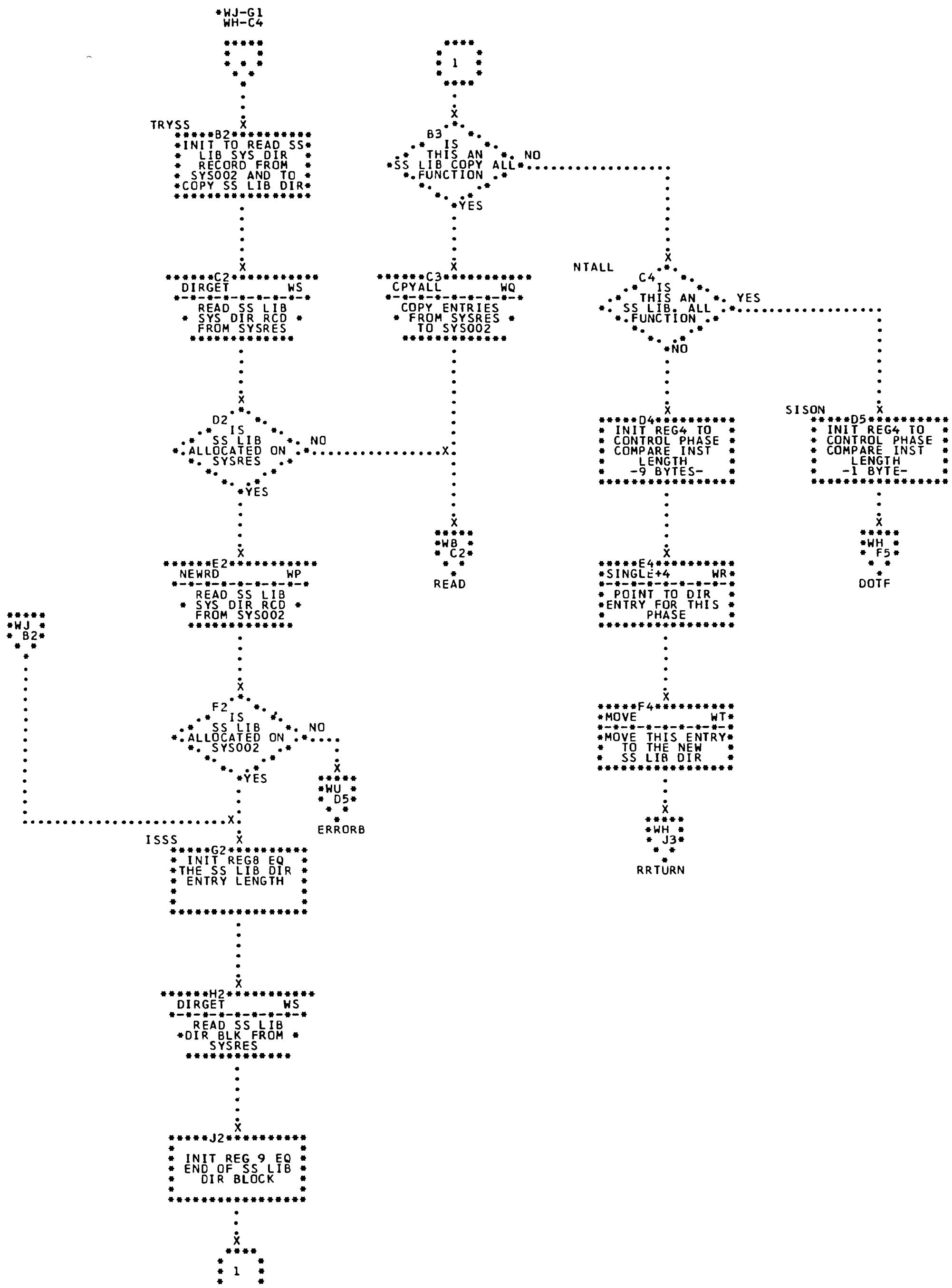


Chart WL. Build SYS002 Core Image Directory Entries for $ Programs CORGZ; Refer to Organization, Chart 46

```
*WB-H2
 WC-J4

A2  HAS ALL OF CI LIB BEEN COPIED     YES -> 2
    NO
B2  HAS A COPY STMNT BEEN PROCESSED   NO -> C3
    YES
C2  SET CI LIB IN PROCESS SW ON.
    -SWITCH,BIT7
D2  INIT TO CREATE CI DIR ENTRIES FOR ALL $ PROGRAMS   *D1
E2  NEWRD                                   WP
    READ CI LIB SYS DIR RCD SYS002
F2  INIT FOR CI LIB DIR BLK FROM SYSRES   *F1
G2  READDK                                  WP
    READ CI LIB DIR BLOCK FROM SYSRES
H2  INIT REGISTERS FOR DIRECTORY SEARCH   *H1
J2  EXCMP                                   WR
    SEARCH FOR $ DIR ENTRIES ON SYSRES
K2  MOVE                                    WT
    MOVE $ ENTRY TO SYS002 CI LIB DIR
    -> 1

C3  SET SW RTREND TO A BRANCH CHART WC-J4   *E5
    -> WC C2   FRMDIR

1
C4  NXTONE                                  WR
    UPDATE FOR NEXT ENTRY AND CON- TINUE SEARCH   *D5
    WR F2
STEXIT
D4  WRTSD                                   WQ
    WRITE THE CURRENT SYS DIR RECORD
2
EXIT
E4  LOAD REG13 WITH VOL CYL ADDRESS
F4  SVC 1
    FETCH CORGZ2
    CHART WW
```

*D1
1. SET NAME EQ $
2. SET DISK ADDR, NEWDSK, FOR CI LIB SYS DIR
3. SET CONTROL DATA, COUNT, FOR CI LIBRARY

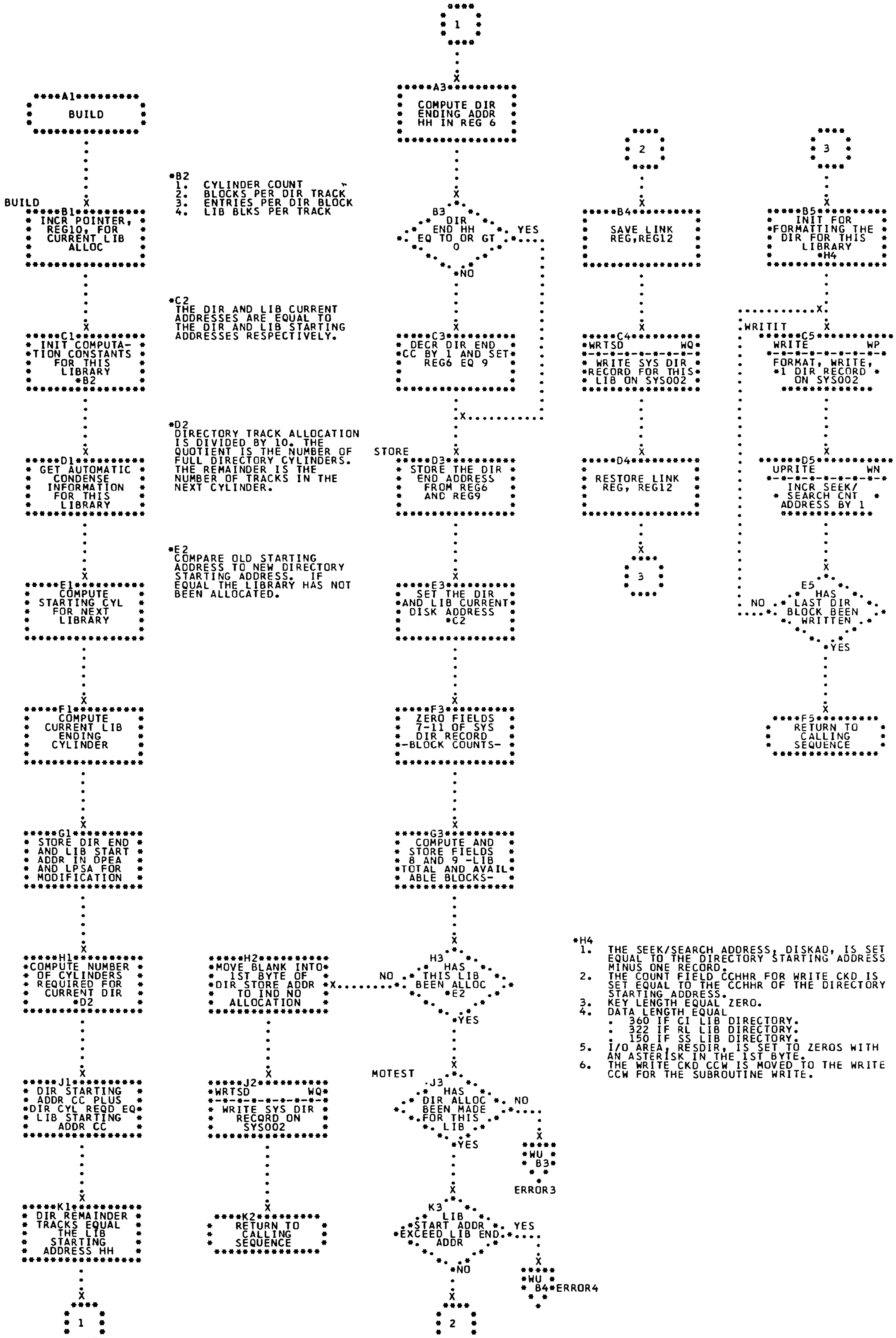
*F1
1. REG EQ ADDRESS OF SYSRES SYS DIR INPUT AREA
2. REG7 EQ SYSRES CI LIB DIR STARTING ADDRESS
3. DISK ADDRESS OF SYSRES CURRENT DIR RECORD IS IN OLDPSA.

*H1
1. REG 8 EQ DIR ENTRY LENGTH
2. REG 9 EQ END ADDRESS OF DIR BLOCK
3. REG 4 EQ. 0. BYTE COUNT TO CONTROL SEARCH COMPARE INST LENGTH
4. REG 5 EQ RETURN ADDRESS WHEN END OF DIR IS FOUND

*D5
RETURN IS TO STEXIT, D4, WHEN THE END OF THE DIR IS REACHED ON SYSRES

*E5
THE SWITCH RTREND, WHEN A BRANCH, FORCES A RE-ENTRY INTO THE END AT FRMDIR

Chart WM. Build System Directory Records and Format System Directory CORGZ; Refer to Organization, Chart 46



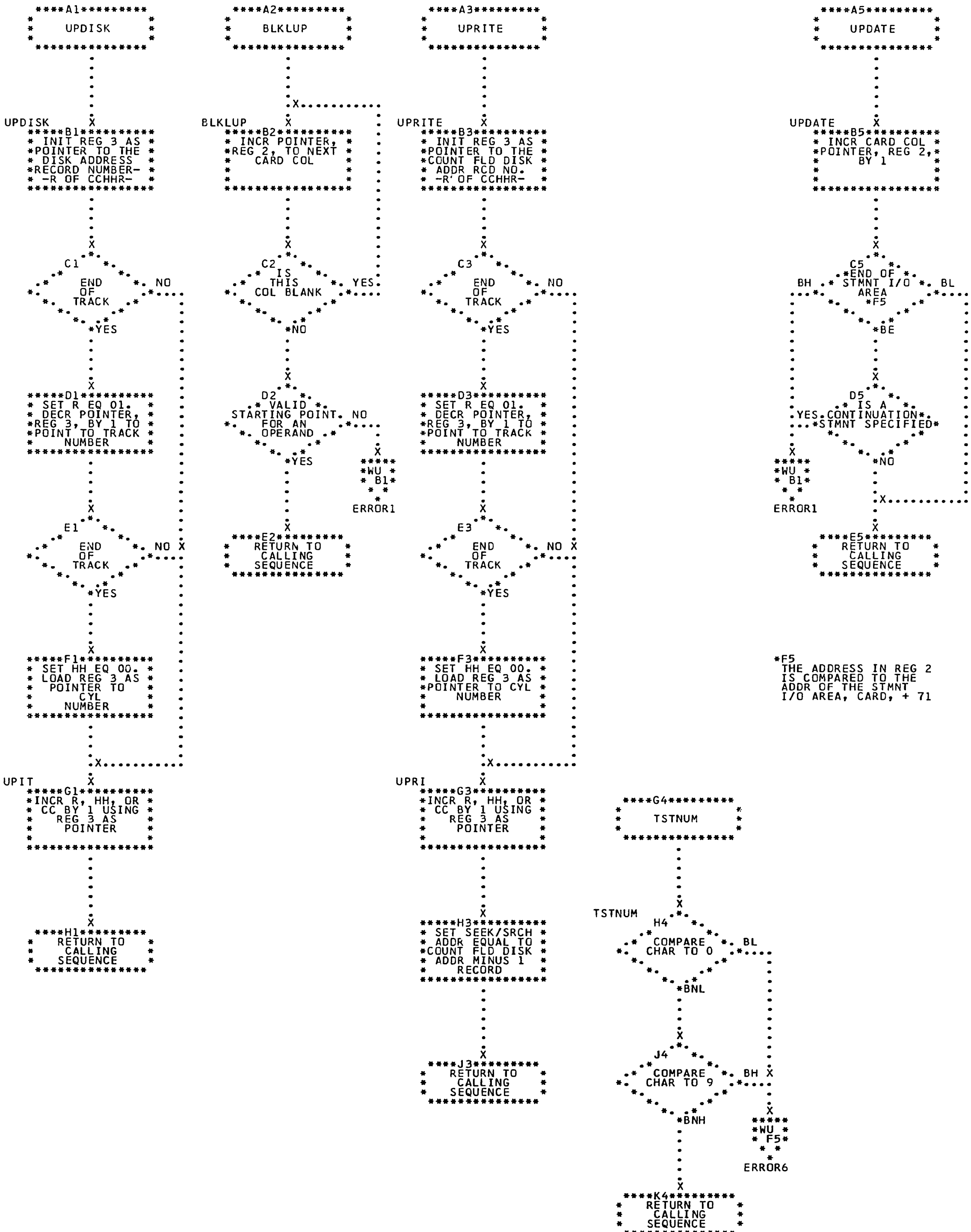

UPDISK
BLKLUP
UPRITE
UPDATE
INIT REG 3 AS POINTER TO THE DISK ADDRESS RECORD NUMBER- -R OF CCHHR-
INCR POINTER, REG 2, TO NEXT CARD COL
INIT REG 3 AS POINTER TO THE COUNT FLD DISK ADDR RCD NO. -R' OF CCHHR-
INCR CARD COL POINTER, REG 2, BY 1
END OF TRACK
IS THIS COL BLANK
END OF STMNT I/O AREA
SET R EQ 01. DECR POINTER, REG 3, BY 1 TO POINT TO TRACK NUMBER
VALID STARTING POINT FOR AN OPERAND
IS A CONTINUATION STMNT SPECIFIED
ERROR1
RETURN TO CALLING SEQUENCE
SET HH EQ 00. LOAD REG 3 AS POINTER TO CYL NUMBER
SET HH EQ 00. LOAD REG 3 AS POINTER TO CYL NUMBER
F5 THE ADDRESS IN REG 2 IS COMPARED TO THE ADDR OF THE STMNT I/O AREA, CARD, + 71
UPIT
UPRI
INCR R, HH, OR CC BY 1 USING REG 3 AS POINTER
TSTNUM
SET SEEK/SRCH ADDR EQUAL TO COUNT FLD DISK ADDR MINUS 1 RECORD
COMPARE CHAR TO 0
COMPARE CHAR TO 9
ERROR6

Chart WP. WRITE, NEWRD, IOSYSRS, and READDR Subroutines
CORGZ; Refer to Organization, Charts 46 and 47

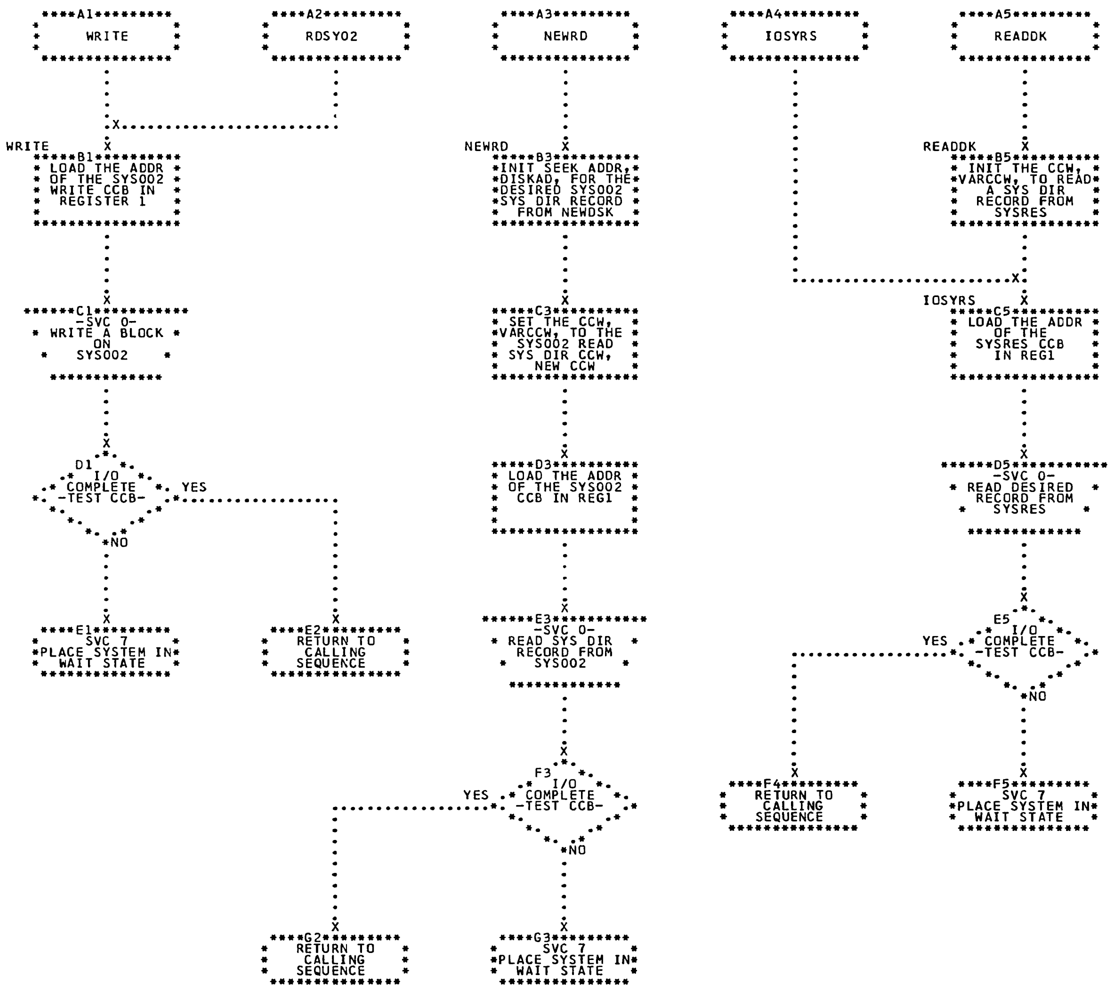


Chart WQ. MOVE2, MOVECC, CPYALL, and WRTSD Subroutines
CORGZ; Refer to Organization, Charts 46 and 47

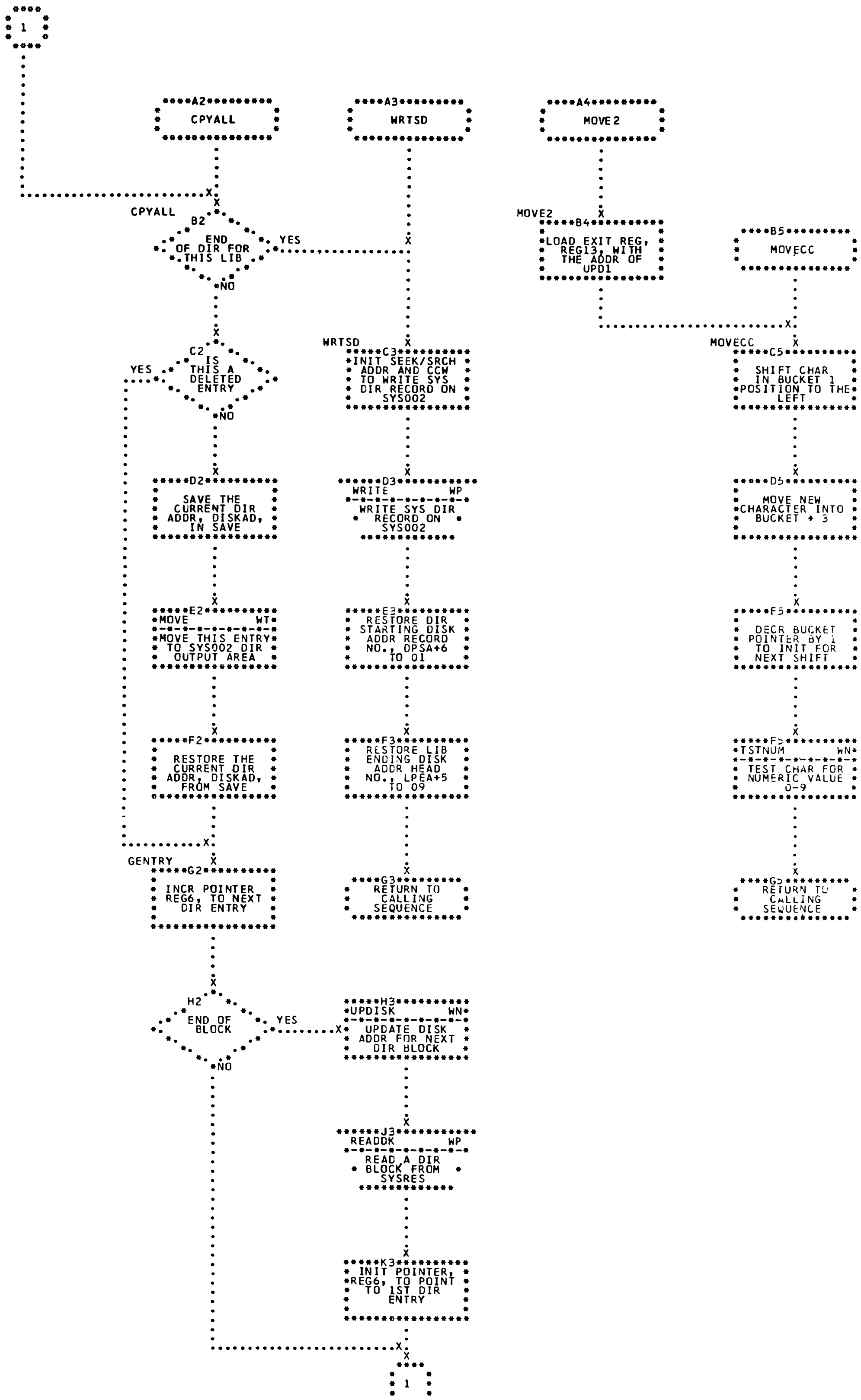


Chart WR. SINGLE, EXCMP, LKDOT and NXTONE Subroutines
CORGZ; Refer to Organization, Charts 46 and 47

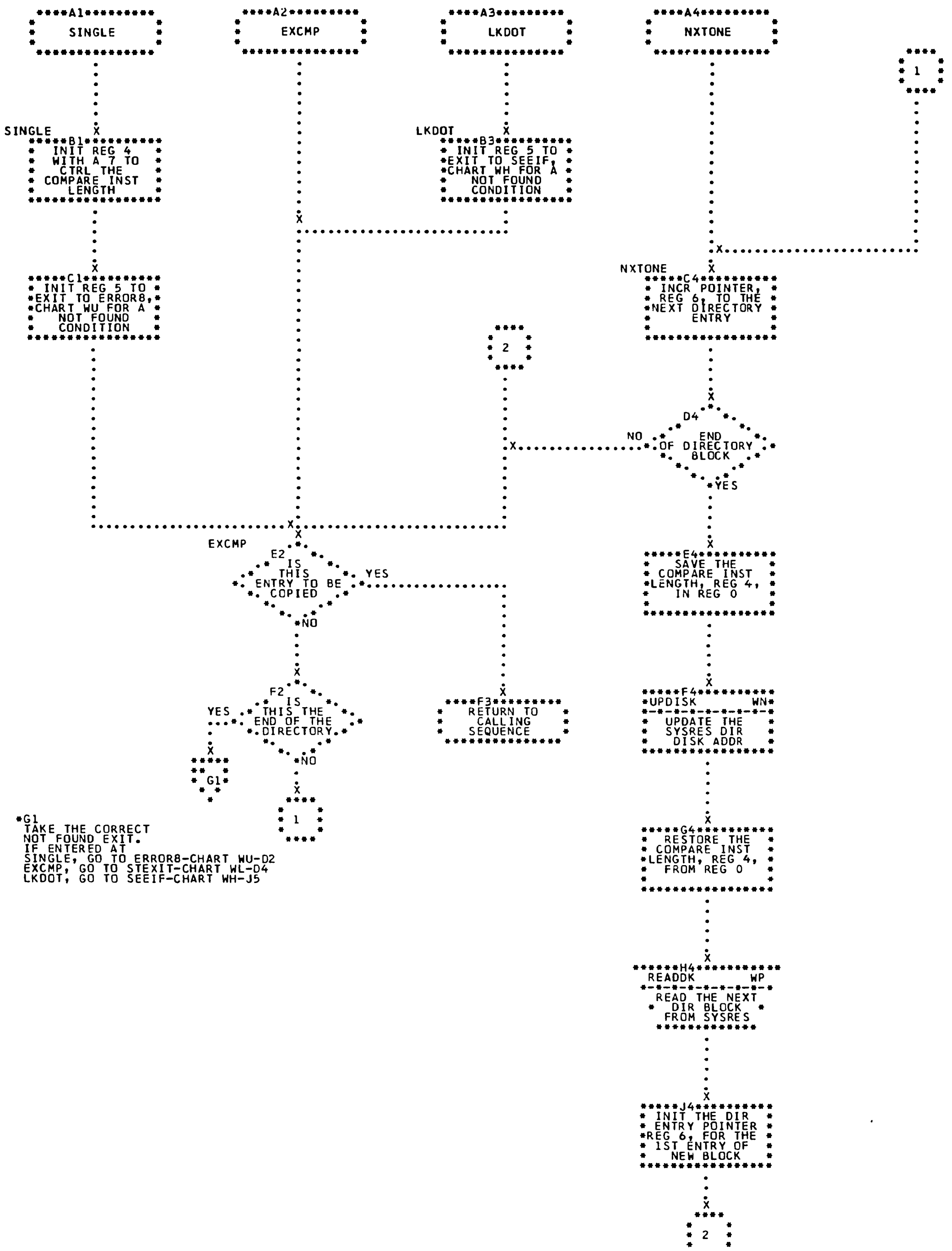


Chart WS. CONVRT and DIRGET Subroutines CORGZ; Refer to Organization, Charts 46 and 47

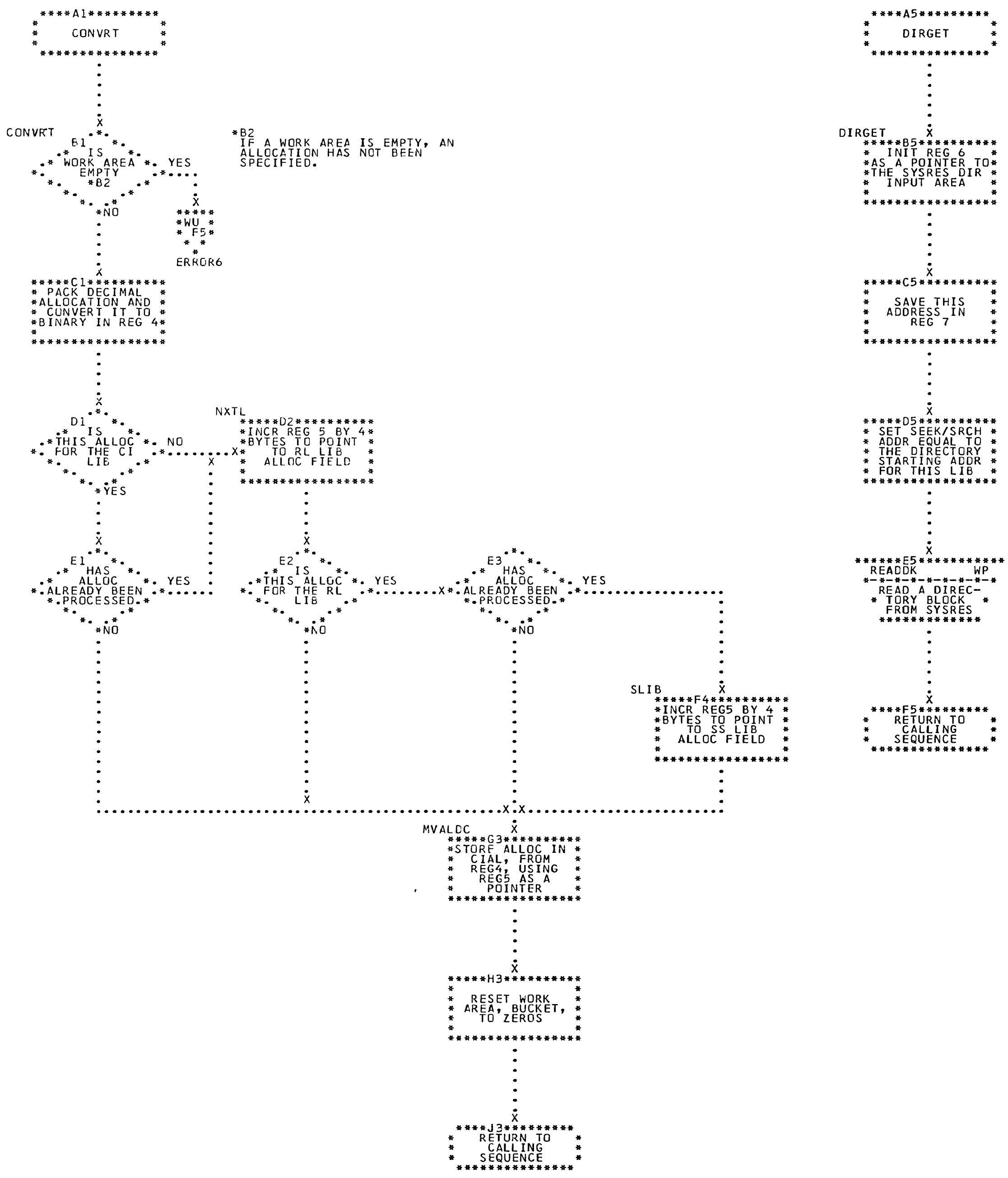


Chart WT. MOVE Subroutine CORGZ; Refer to Organization, Charts 46 and 47

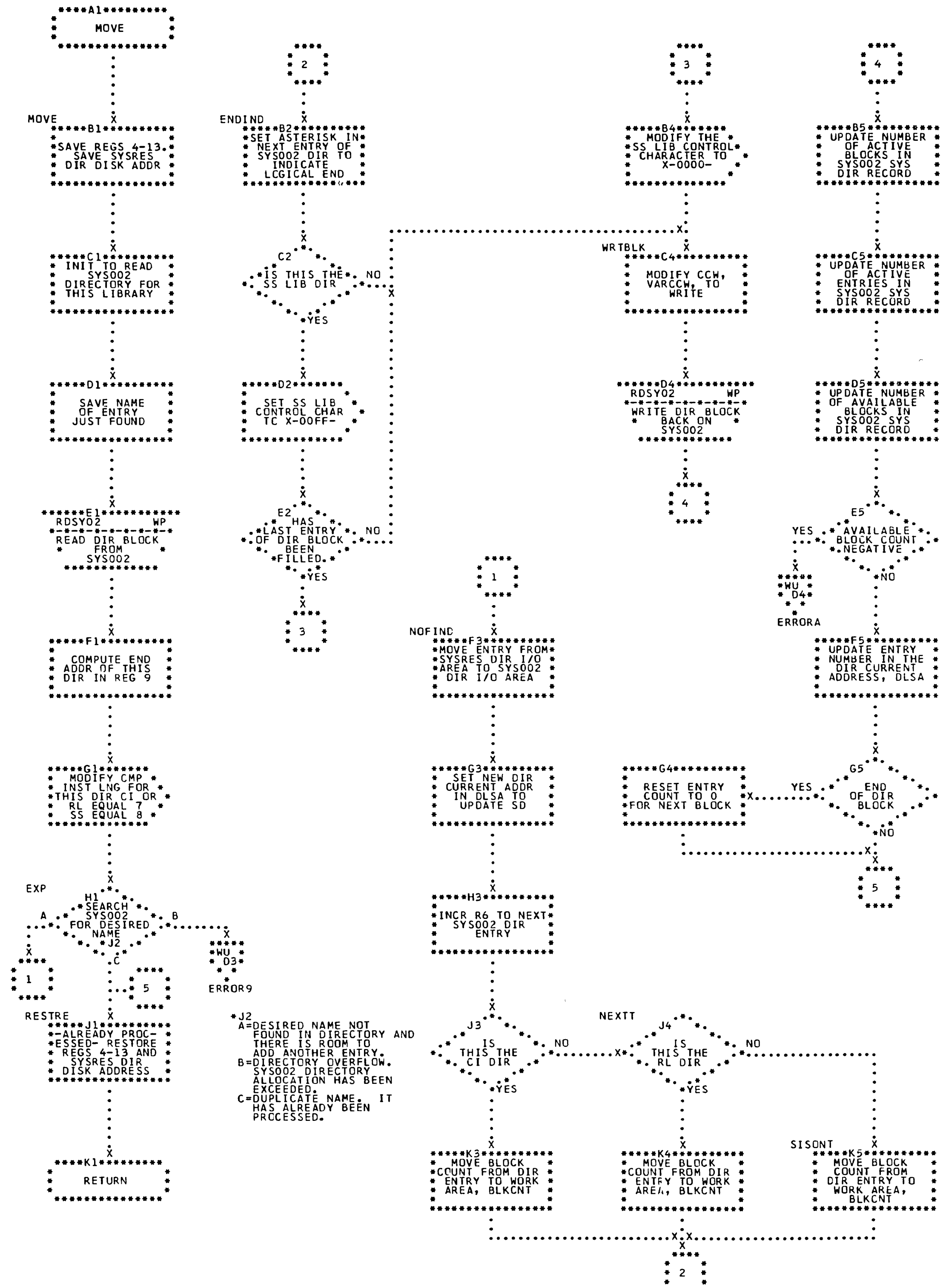
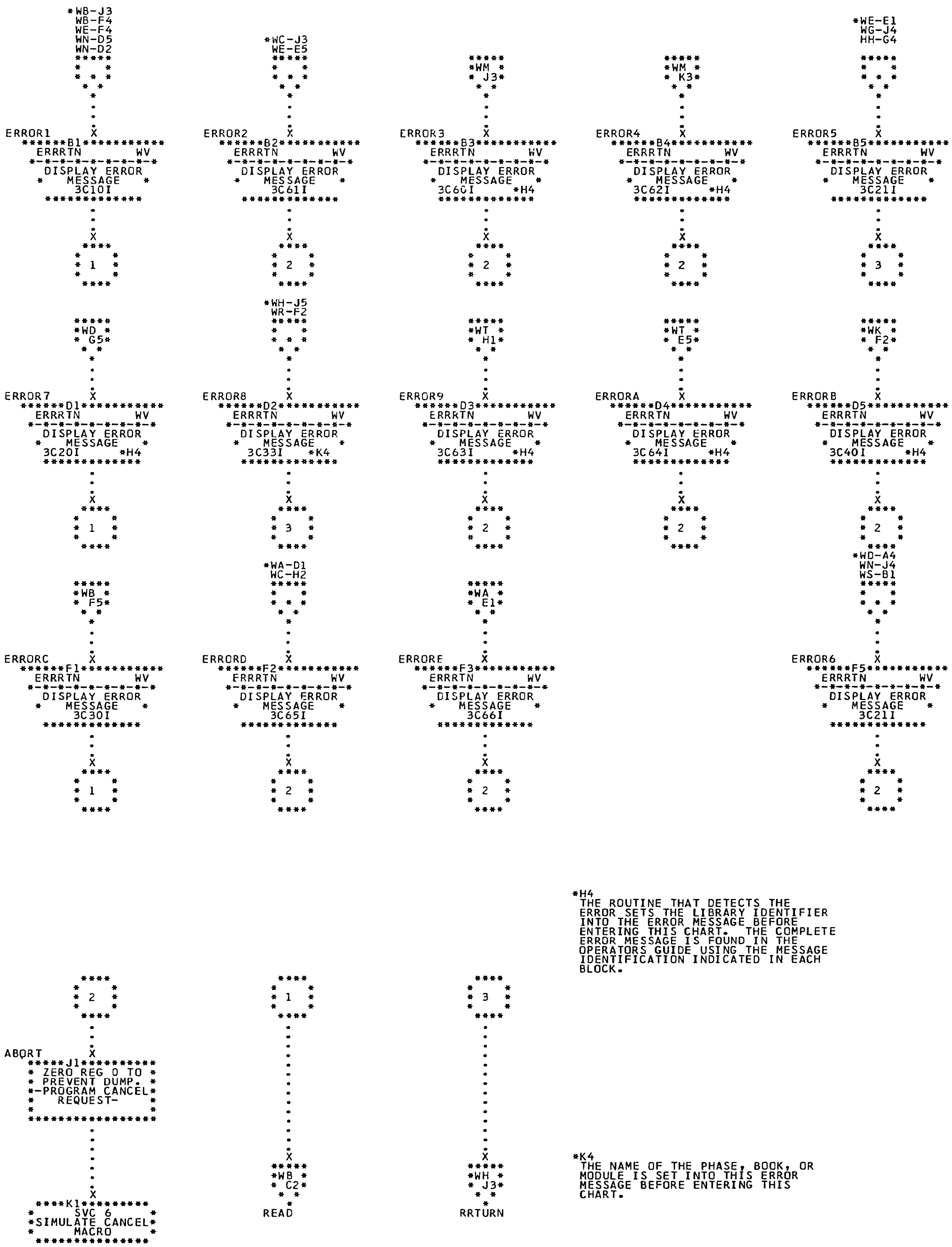


Chart WU. Phase 1 Error Message Routines CORGZ; Refer to Organization, Charts 46 and 47




*H4
THE ROUTINE THAT DETECTS THE ERROR SETS THE LIBRARY IDENTIFIER INTO THE ERROR MESSAGE BEFORE ENTERING THIS CHART. THE COMPLETE ERROR MESSAGE IS FOUND IN THE OPERATORS GUIDE USING THE MESSAGE IDENTIFICATION INDICATED IN EACH BLOCK.

*K4
THE NAME OF THE PHASE, BOOK, OR MODULE IS SET INTO THIS ERROR MESSAGE BEFORE ENTERING THIS CHART.

Chart WV. ERRRTN Error Subroutine CORGZ; Refer to Organization, Charts 46 and 47

```
                                       *A3
      ****A2*********                   THIS ROUTINE SERVES AS LINKAGE
     *    ERRRTN     *                  TO THE MAINTENANCE PROGRAM
     *               *                  COMMON ERROR ROUTINE.
     *      *A3      *                  THE COMMON ERROR ROUTINE,
      ***************                   THE MSGLST DTF, AND THE DTF
             *                          MODULE ARE INCLUDED IN CORGZ
             *                          AT LINK-EDIT TIME. THEY ARE
             *                          DISCUSSED FULLY IN THE COMMON
             *                          MAINTENANCE ROUTINE, MAINT.
             *
             *
ERRRTN       X
     *****B2**********
     * INIT REG 1 AS *
     * BRANCH REG TO *
     *  THE COMMON   *
     * ERROR ROUTINE *
     *               *
     ****** *********
             *
             *
             *
             *
             *
             X
     *****C2**********
     *  SAVE CORGZ   *
     *BASE REGISTER, *
     *  REG 15, IN   *
     *     REG 3     *
     *               *
     *****************
             *
             *
             *
             *
             *
             X
    ******D2***********
      ERRRTN       TF
     *-*-*-*-*-*-*-*-*
         DISPLAY
     *   DESIRED     *
         MESSAGE
       *************
             *
             *
             *
             *
             *
ERETRN       X
     *****E2**********
     * RESTORE CORGZ *
     *BASE REGISTER, *
     * REG 15, FROM  *
     *     REG 3     *
     *               *
     *****************
             *
             *
             *
             *
             *
             X
      ****F2*********
     *   RETURN TO   *
     *    CALLING    *
     *   SEQUENCE    *
      ***************
```

Chart WW. Copy Libraries from SYSRES to SYS002 CORGZ2 (Part 1 of 2); Refer to Organization, Chart 46

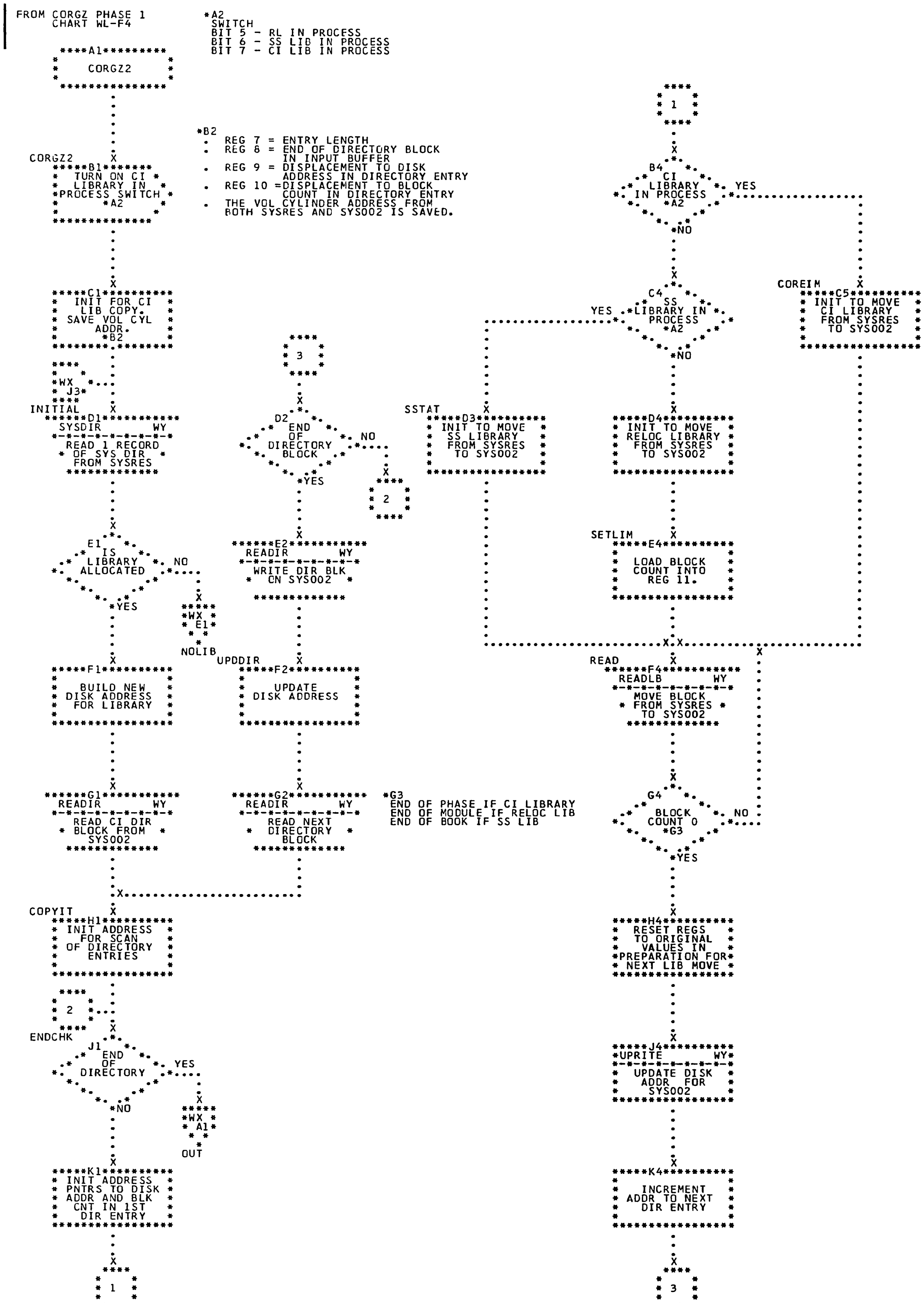


Chart WX. Copy Libraries from SYSRES to SYS002 CORGZ2 (Part 2 of 2); Refer to Organization, Chart 46

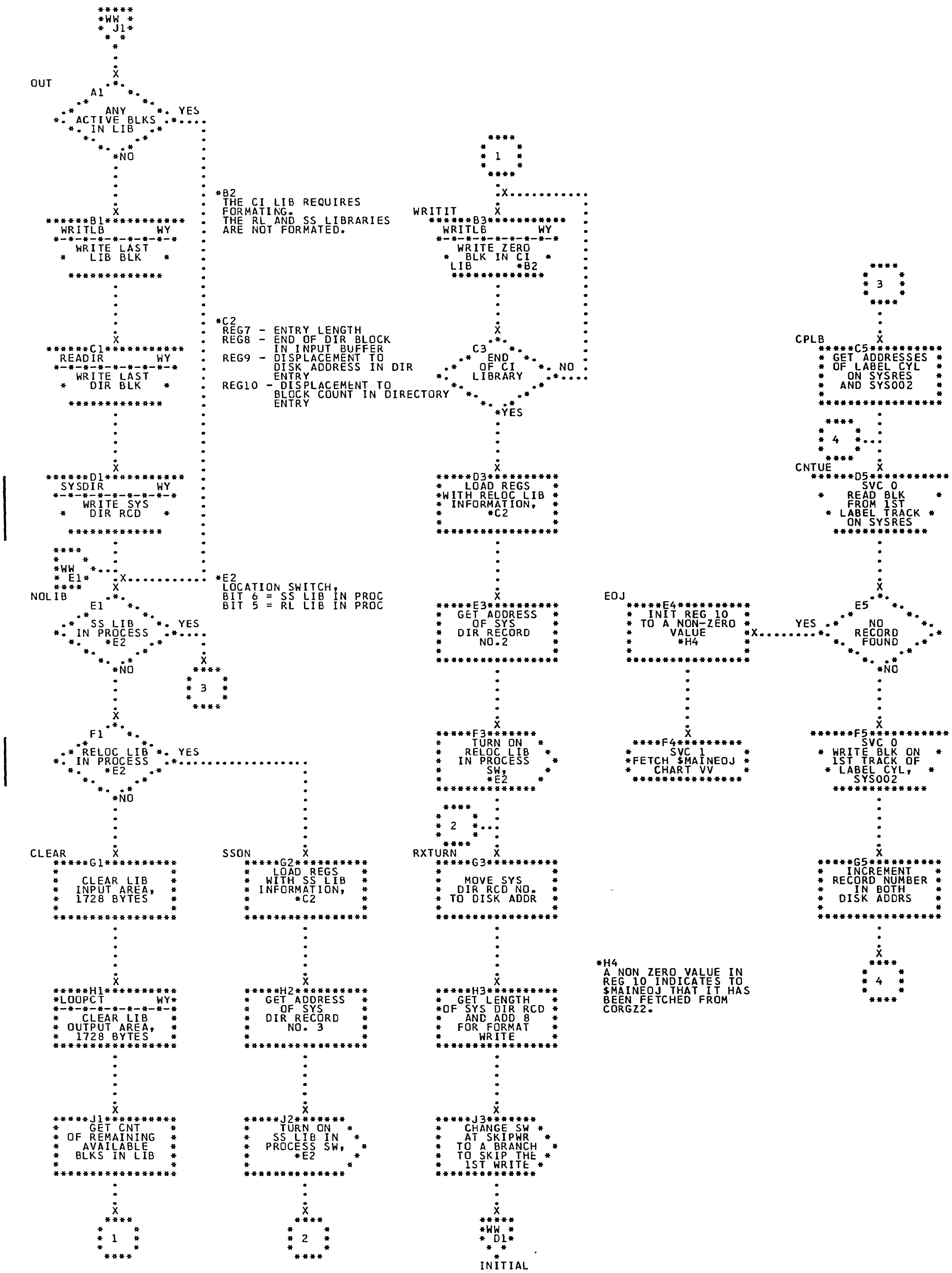


Chart WY. WRITLB, READLB, SYSDIR, READIR, LOOPCT, and UPRITE Subroutine CORGZ2; Refer to Organization, Chart 46

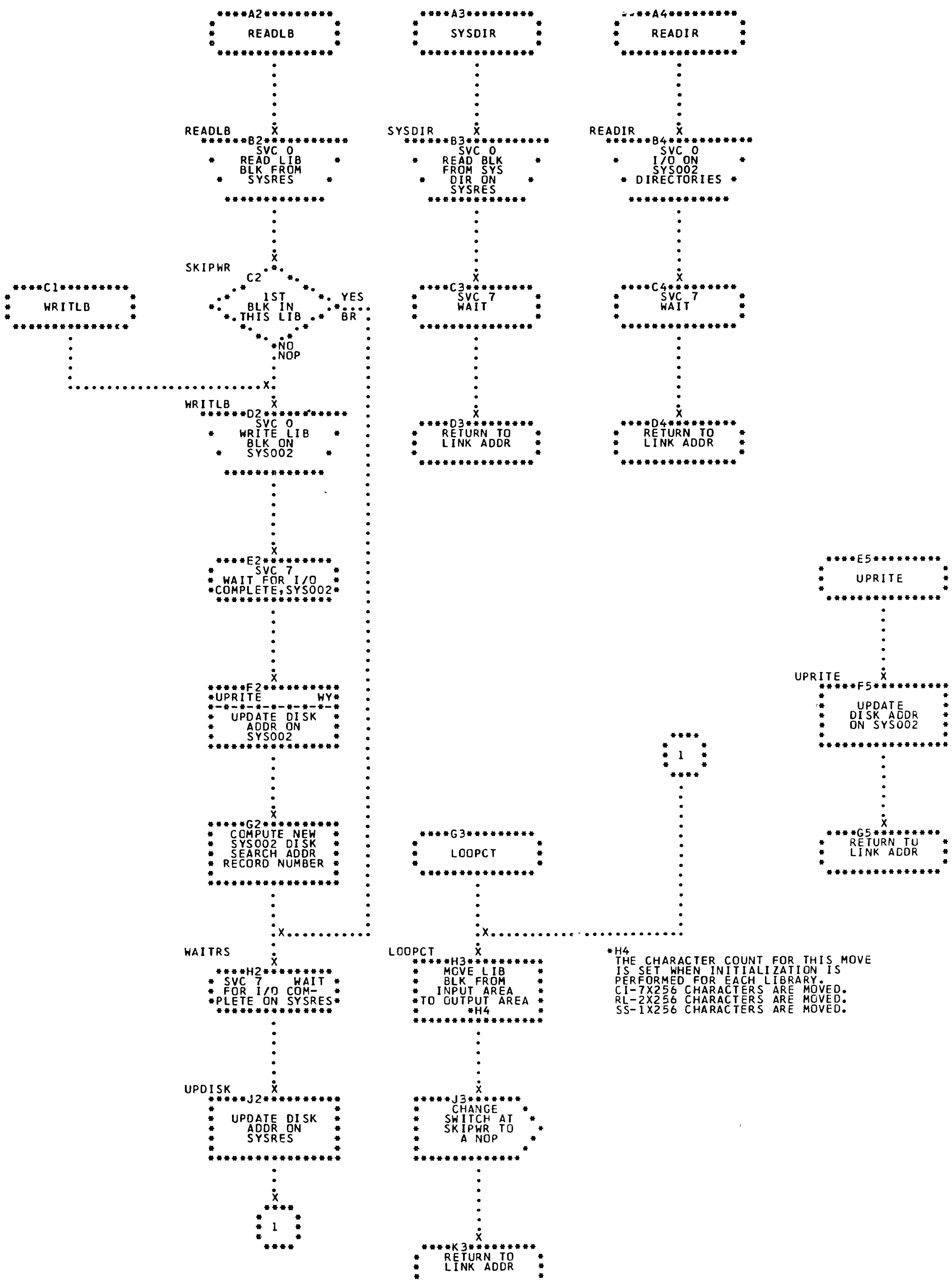


Chart XA. Read and Analyze Control Statements DSERV; Refer to Service, Chart 48

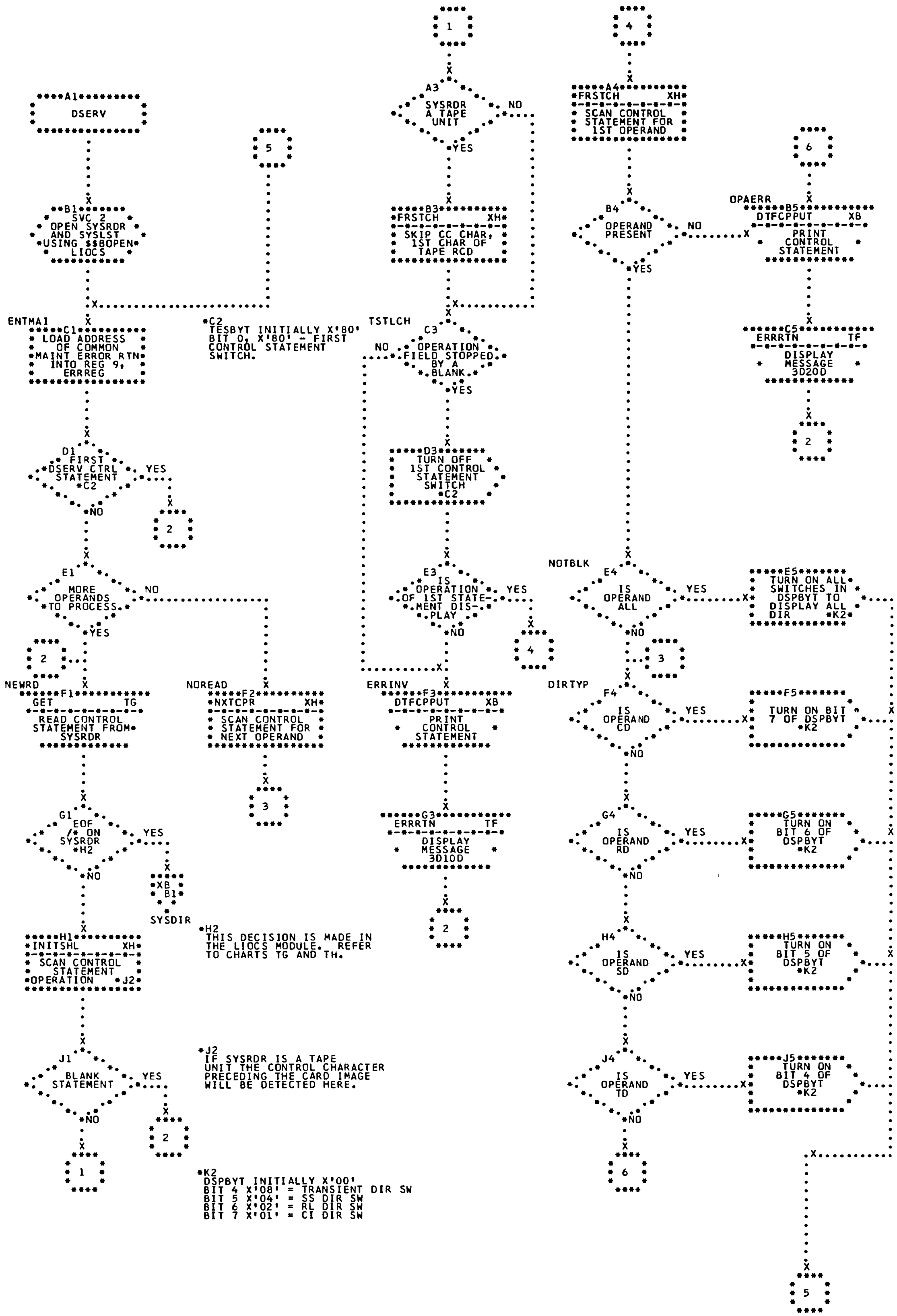


Chart XB. Print System Status Report DSERV; Refer to Service, Chart 48

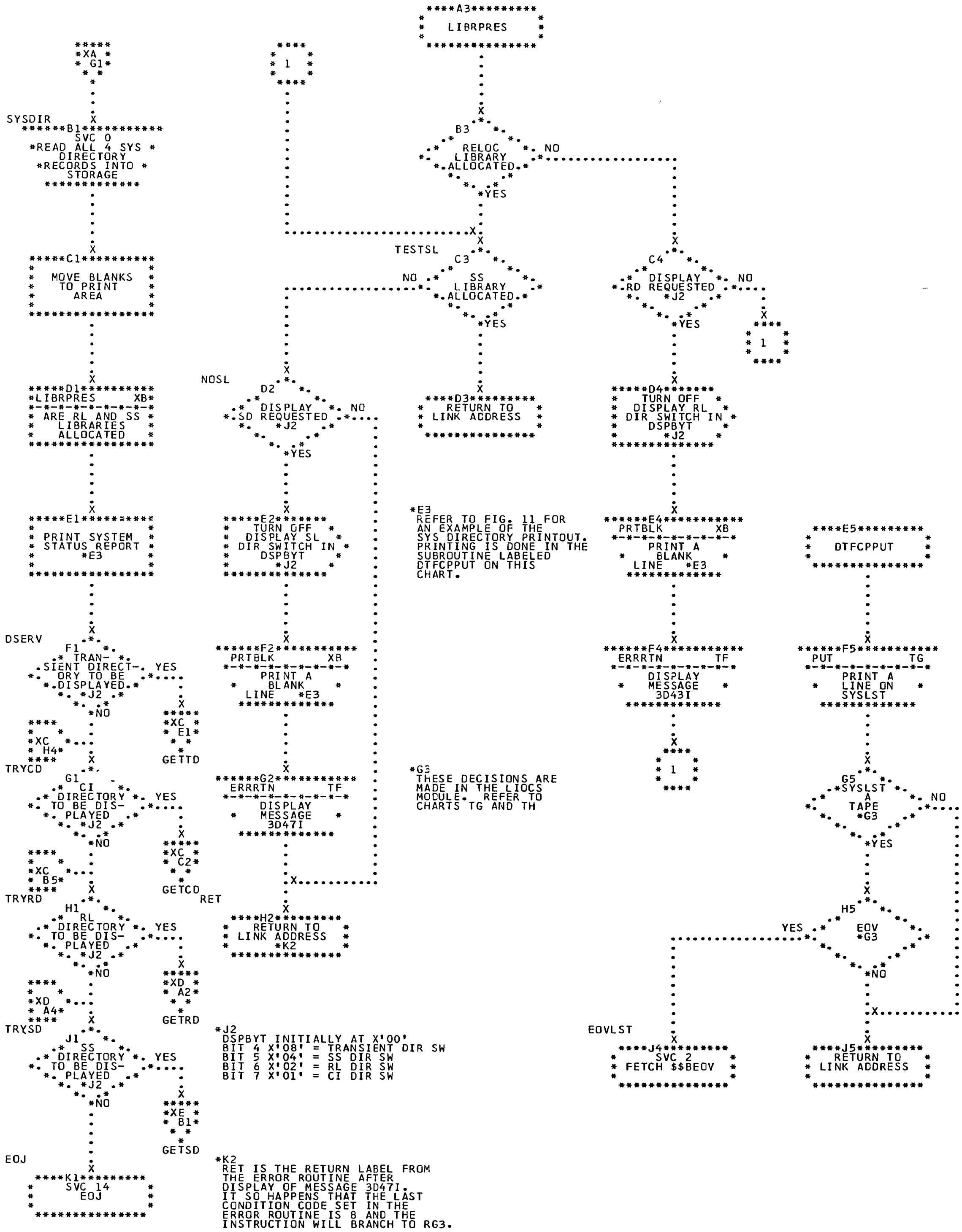


Chart XC. Print Transient and/or Core Image Directories
DSERV; Refer to Service, Chart 48

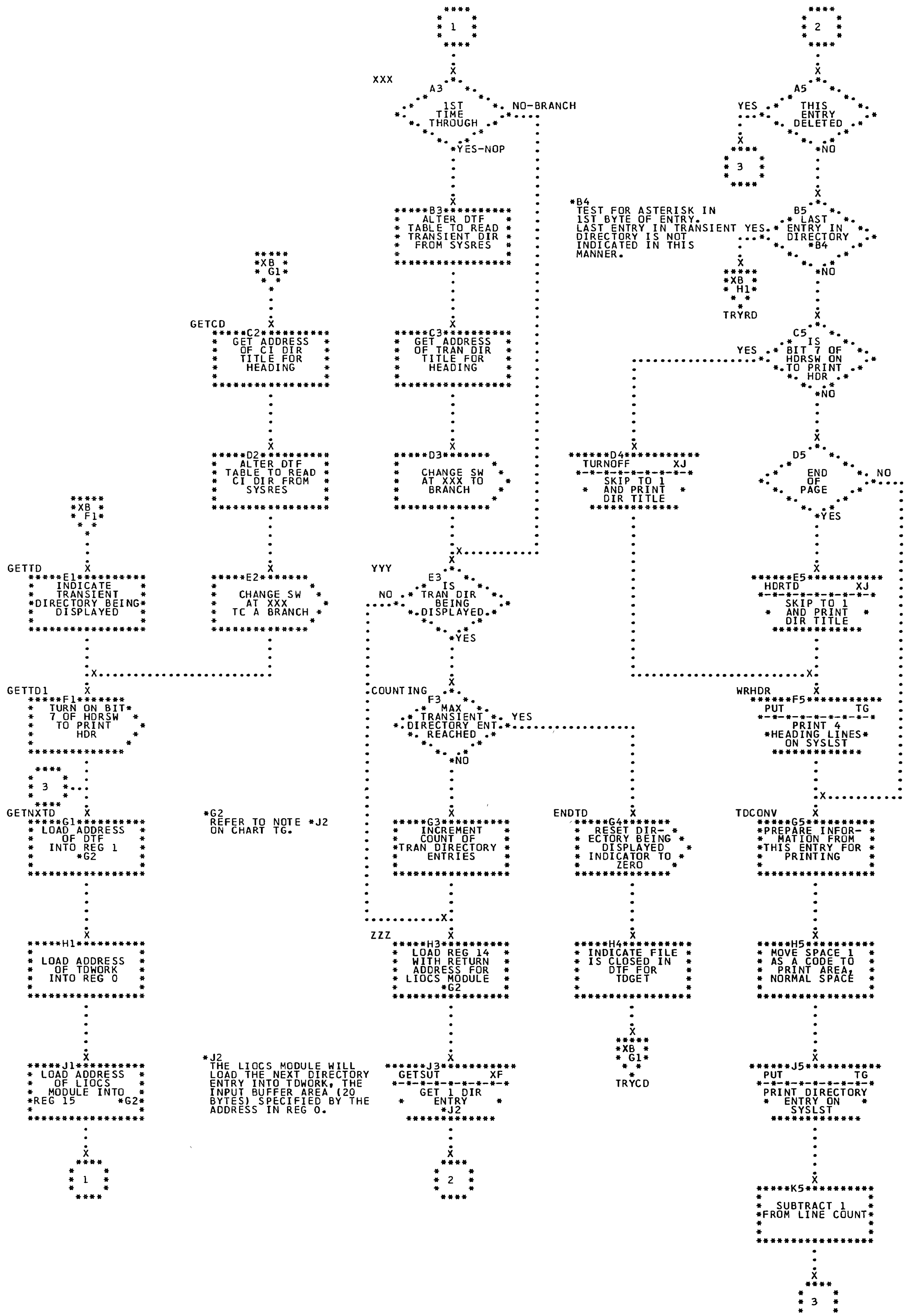


Chart XD. Print Relocatable Directory DSERV; Refer to Service, Chart 48

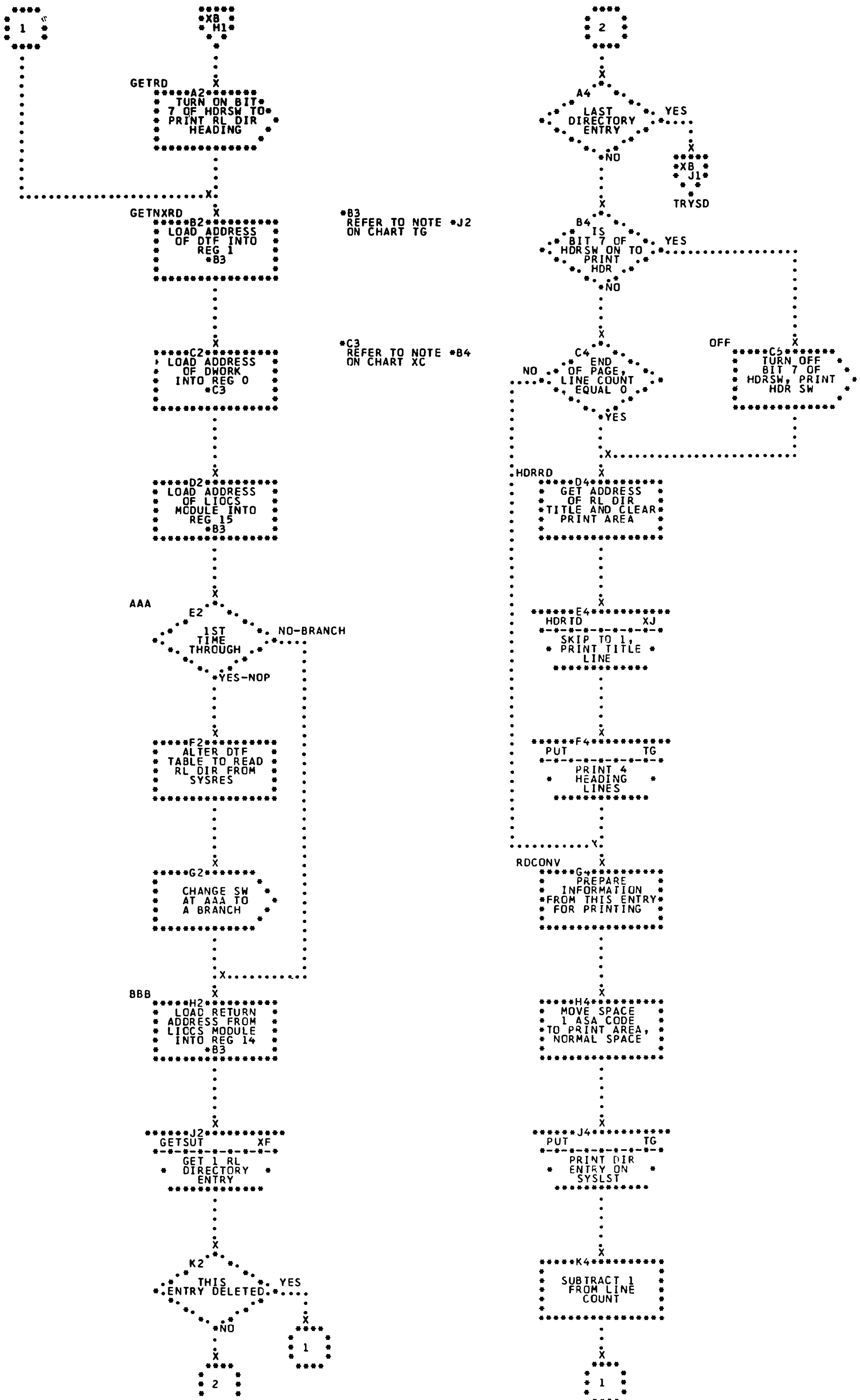


Chart XE. Print Source Statement Directory DSERV; Refer to Service, Chart 48

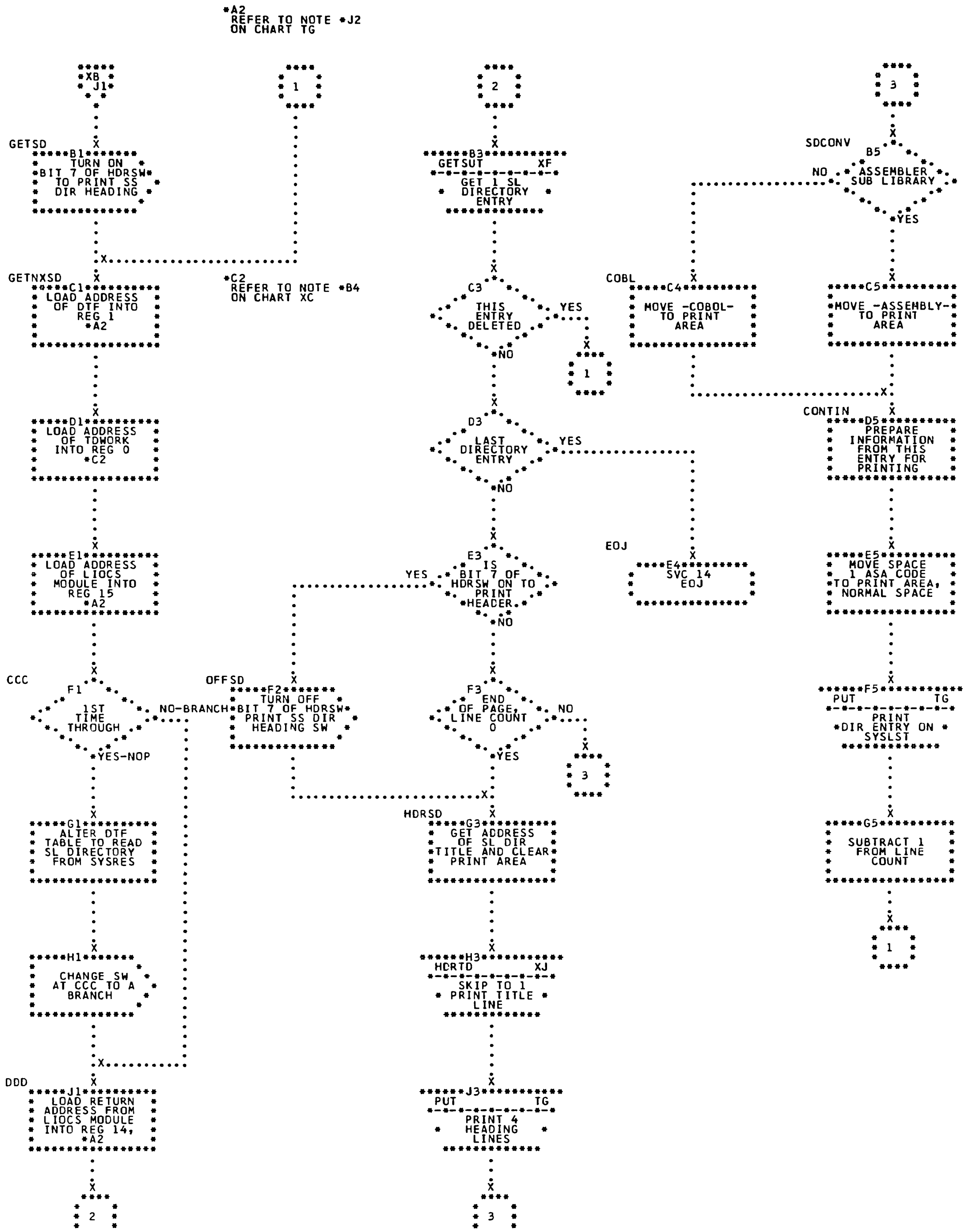


Chart XF. Get Next Directory Entry DSERV (Part 1 of 2); Refer to Service, Chart 48

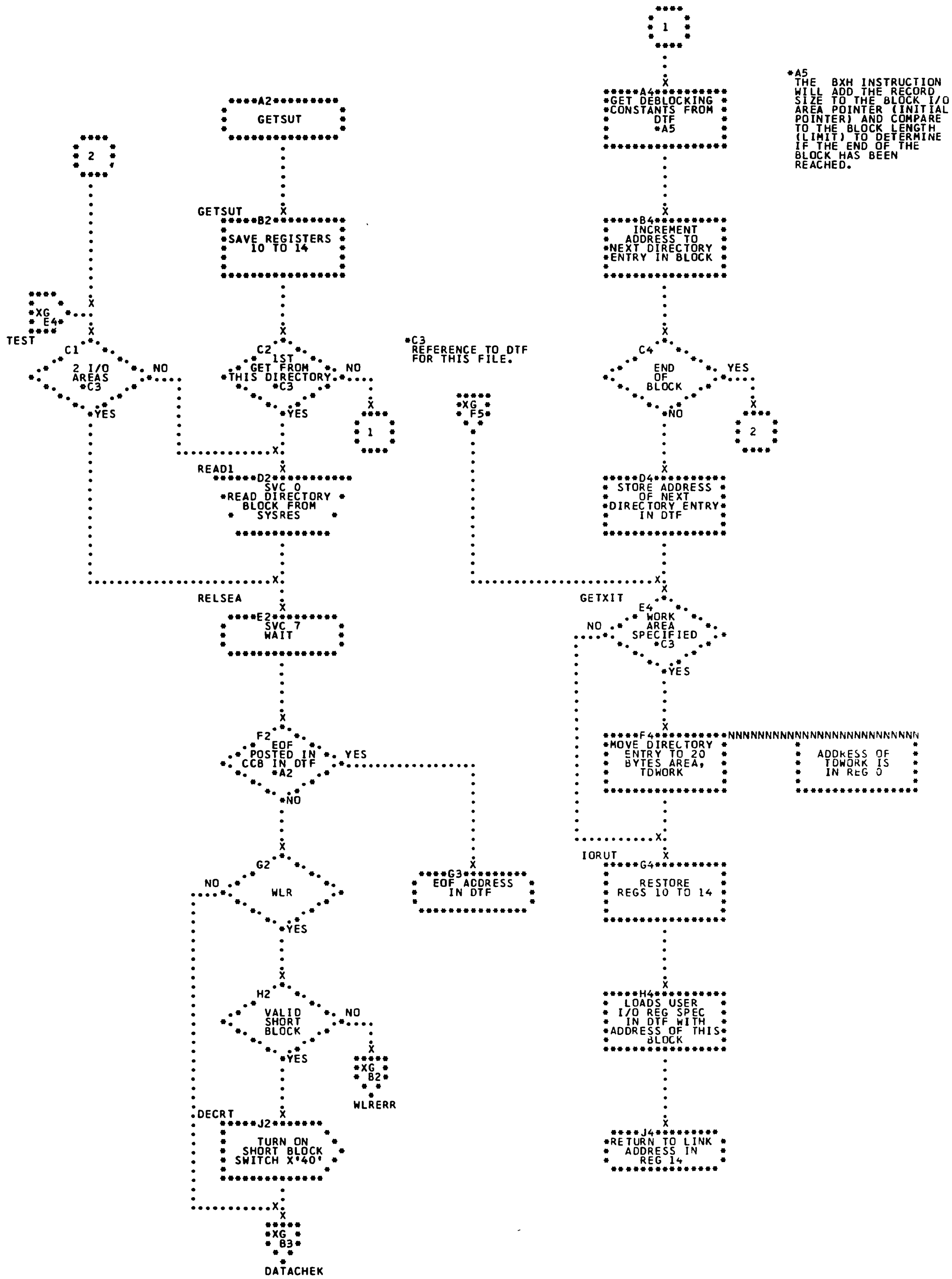


Chart XG. Get Next Directory Entry DSERV (Part 2 of 2); Refer to Service, Chart 48

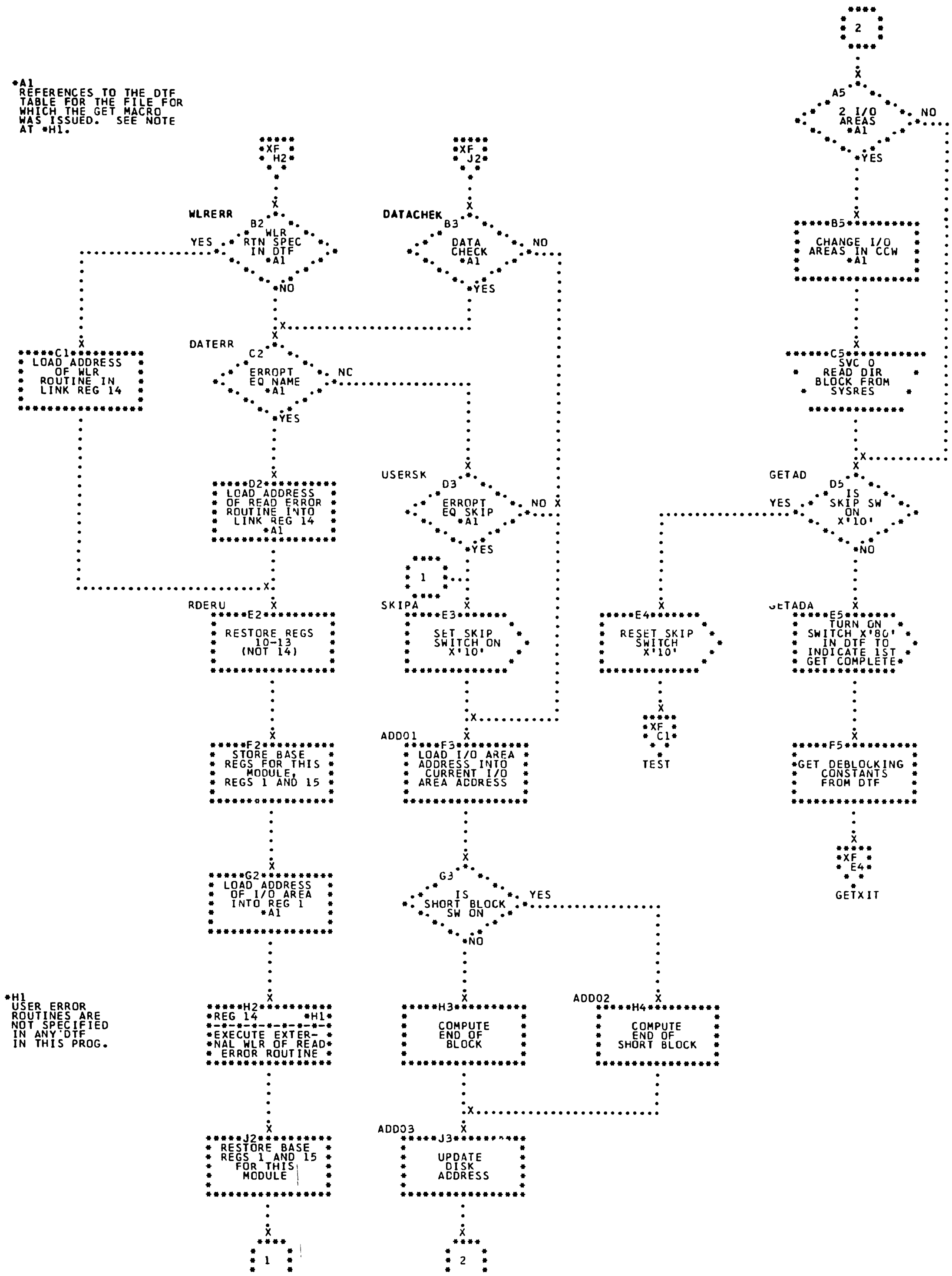


Chart XH. Scan Control Statements DSERV; Refer to Service, Chart 48

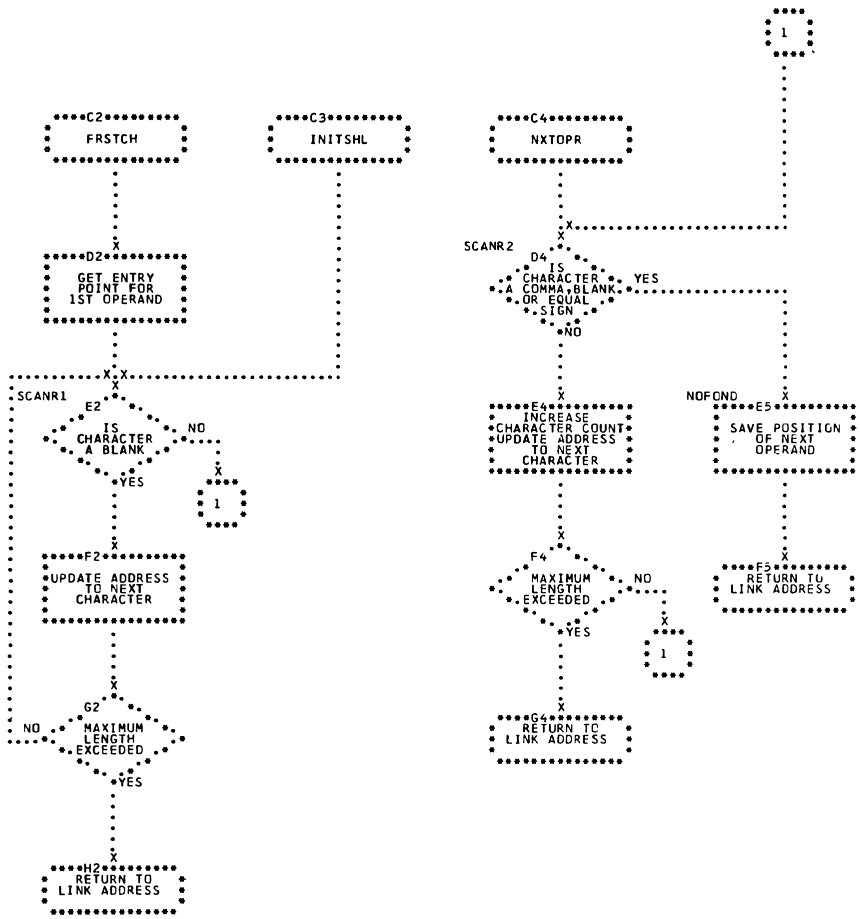


Chart XJ. Print Title Lines DSERV; Refer to Service, Chart 48

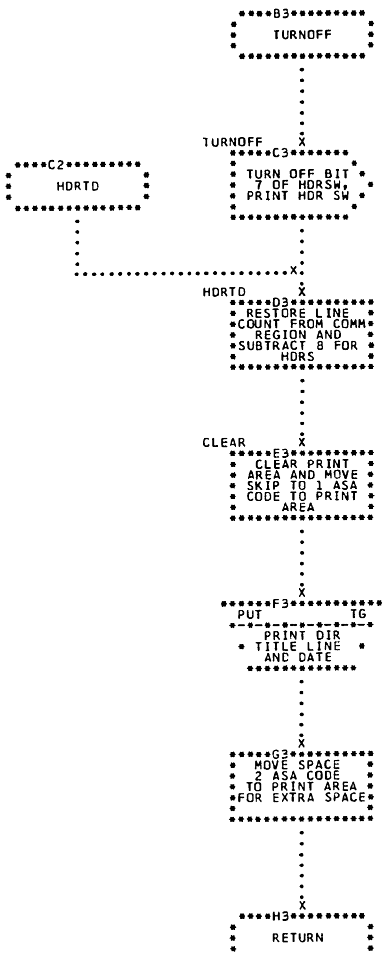


Chart YA. Analyze Control Statements RSERV; Refer to Service, Chart 49

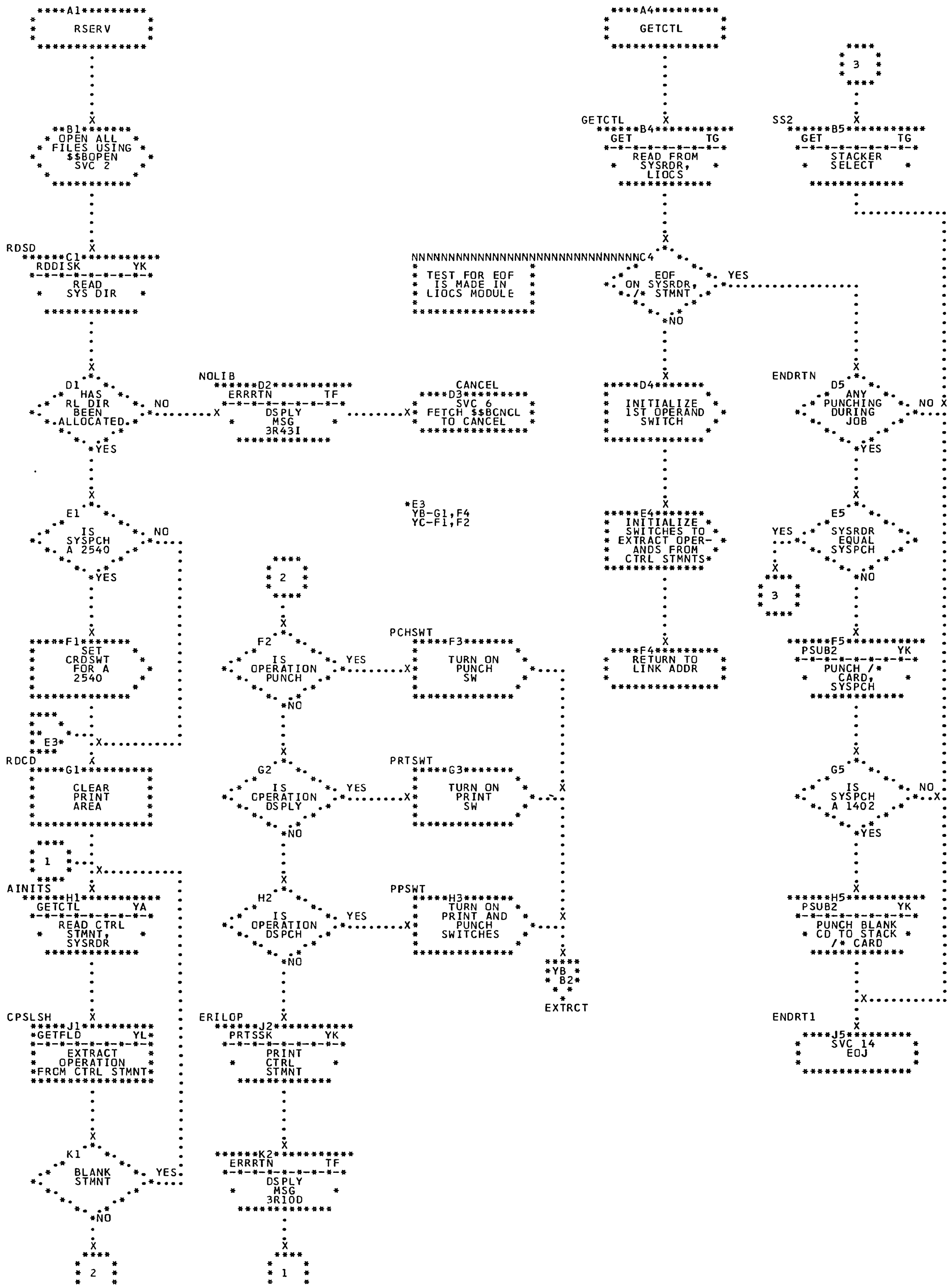


Chart YB. Analyze Control Statement Operands RSERV; Refer to Service, Chart 49

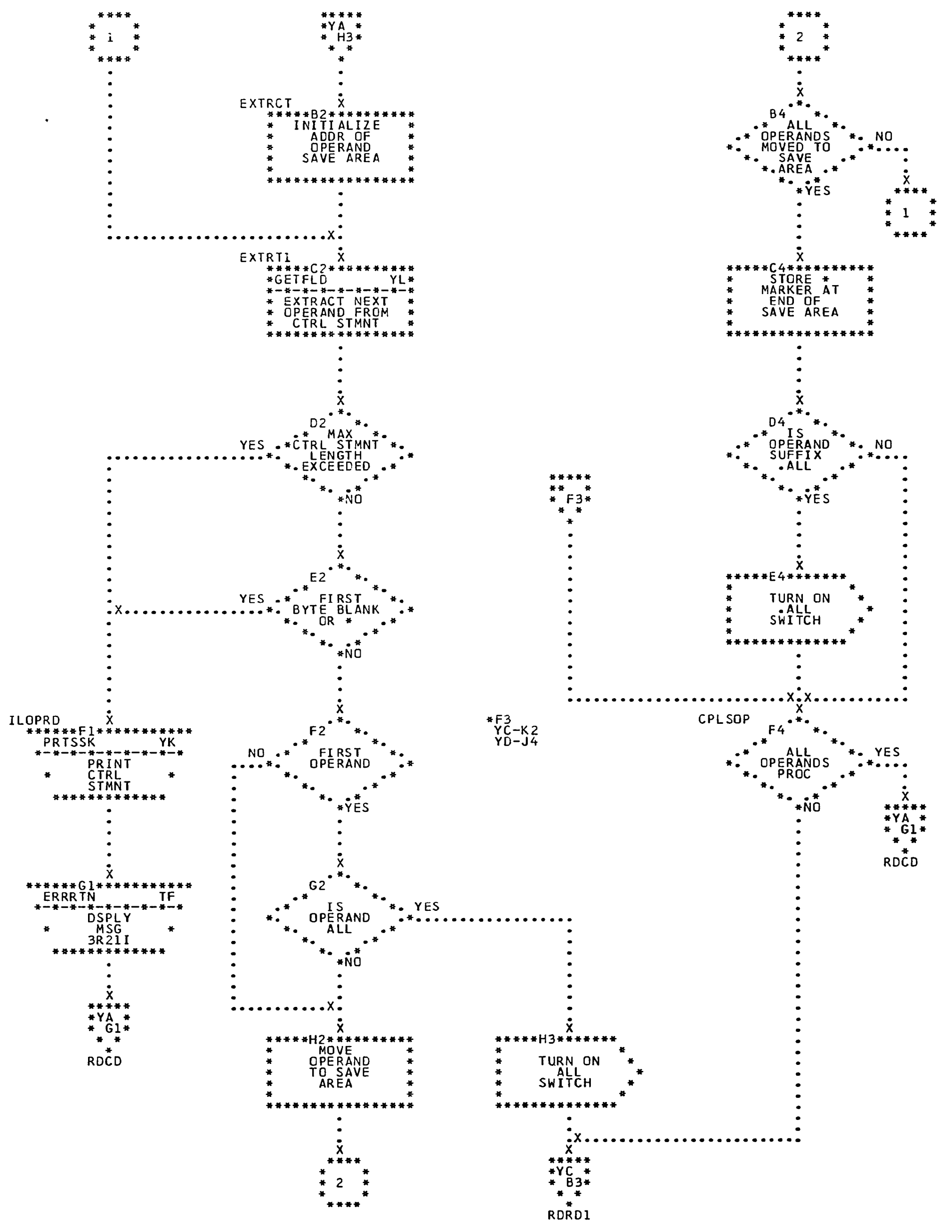


Chart YC. Read Directory Block and Scan for Module Name RSERV; Refer to Service, Chart 49

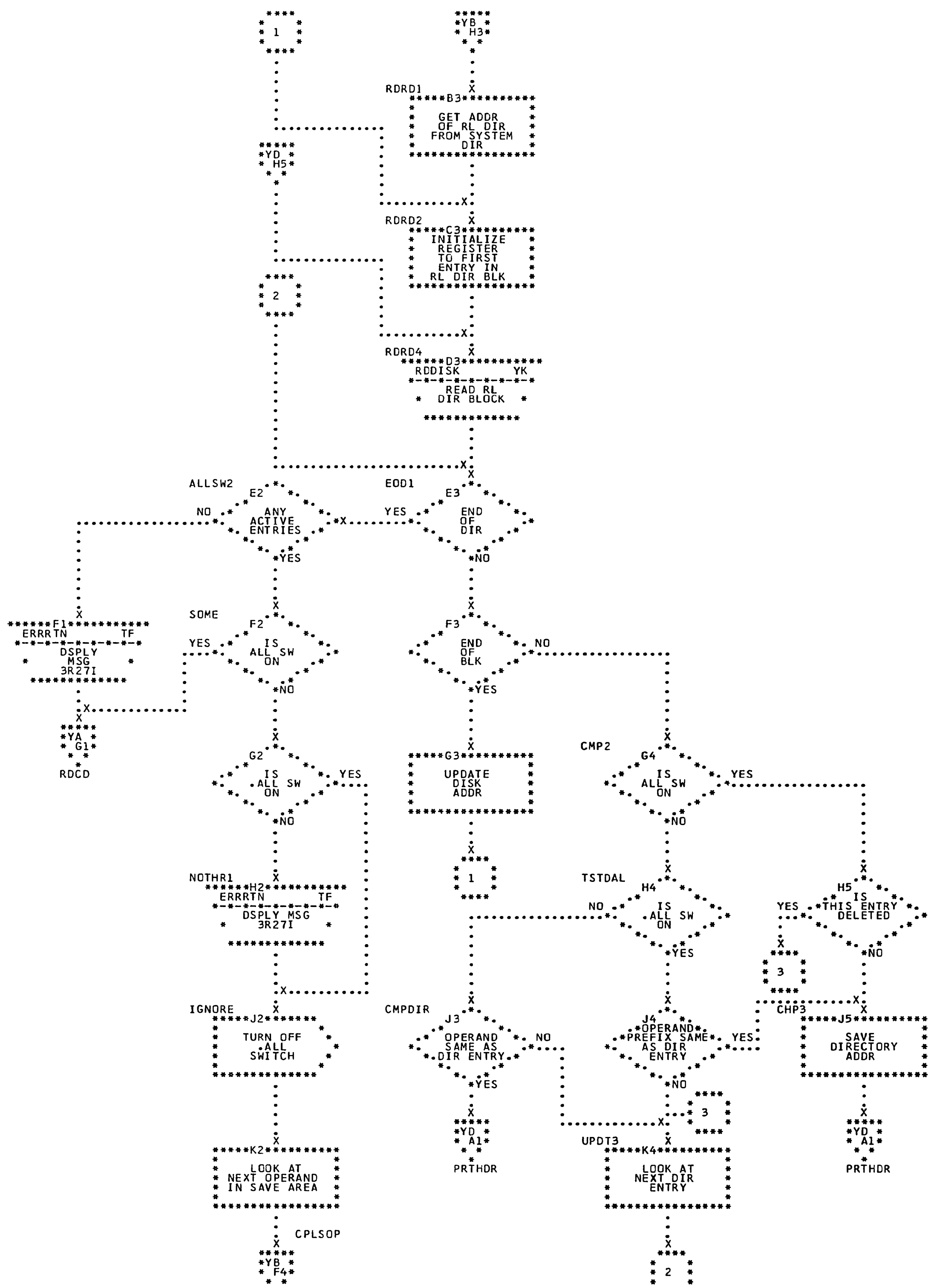


Chart YD. Read Blocks from Relocatable Library and Determine Type RSERV; Refer to Service, Chart 49

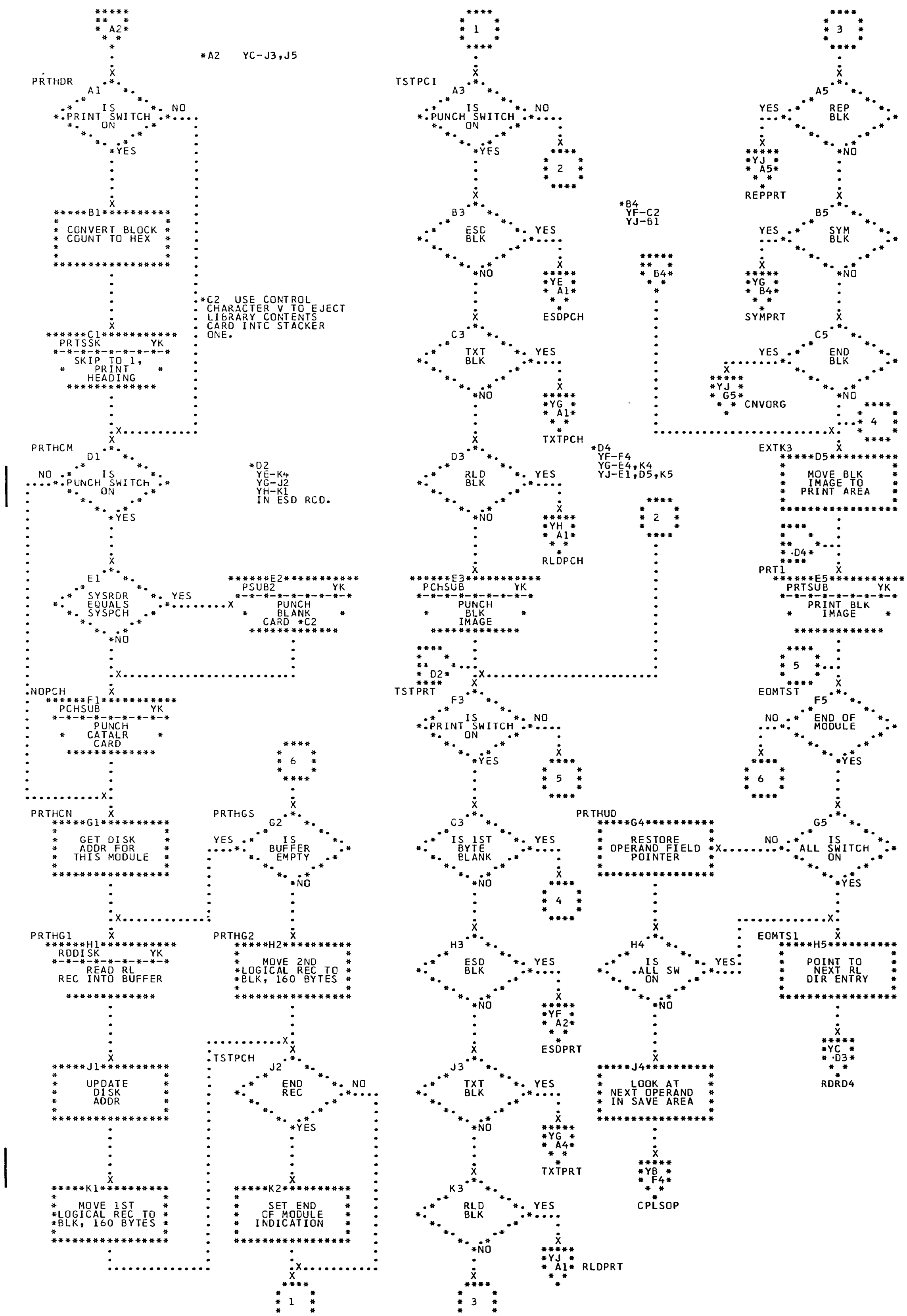


Chart YE. Punch ESD Record RSERV; Refer to Service, Chart 49

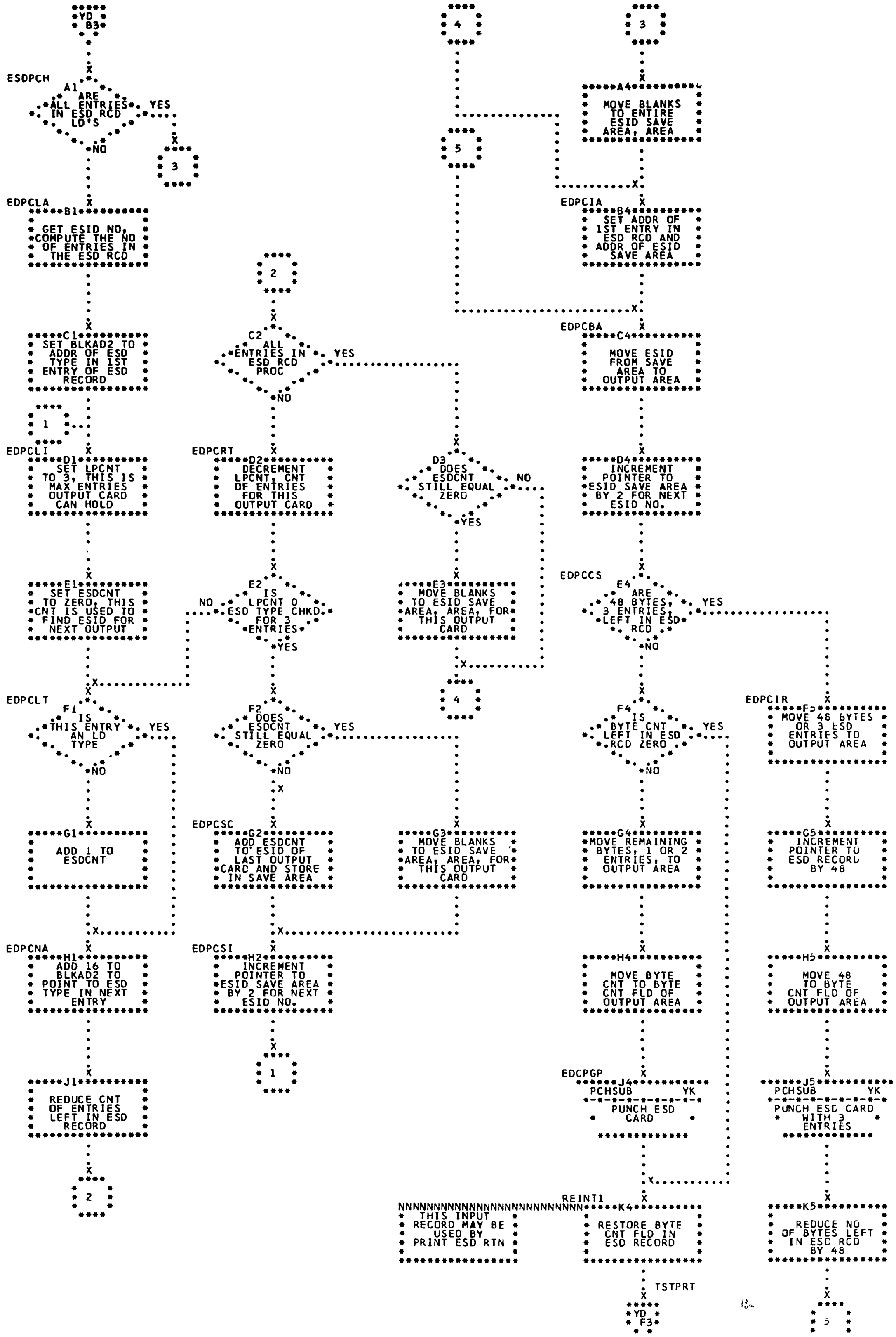


Chart YF. Print ESD Record RSERV; Refer to Service, Chart 49

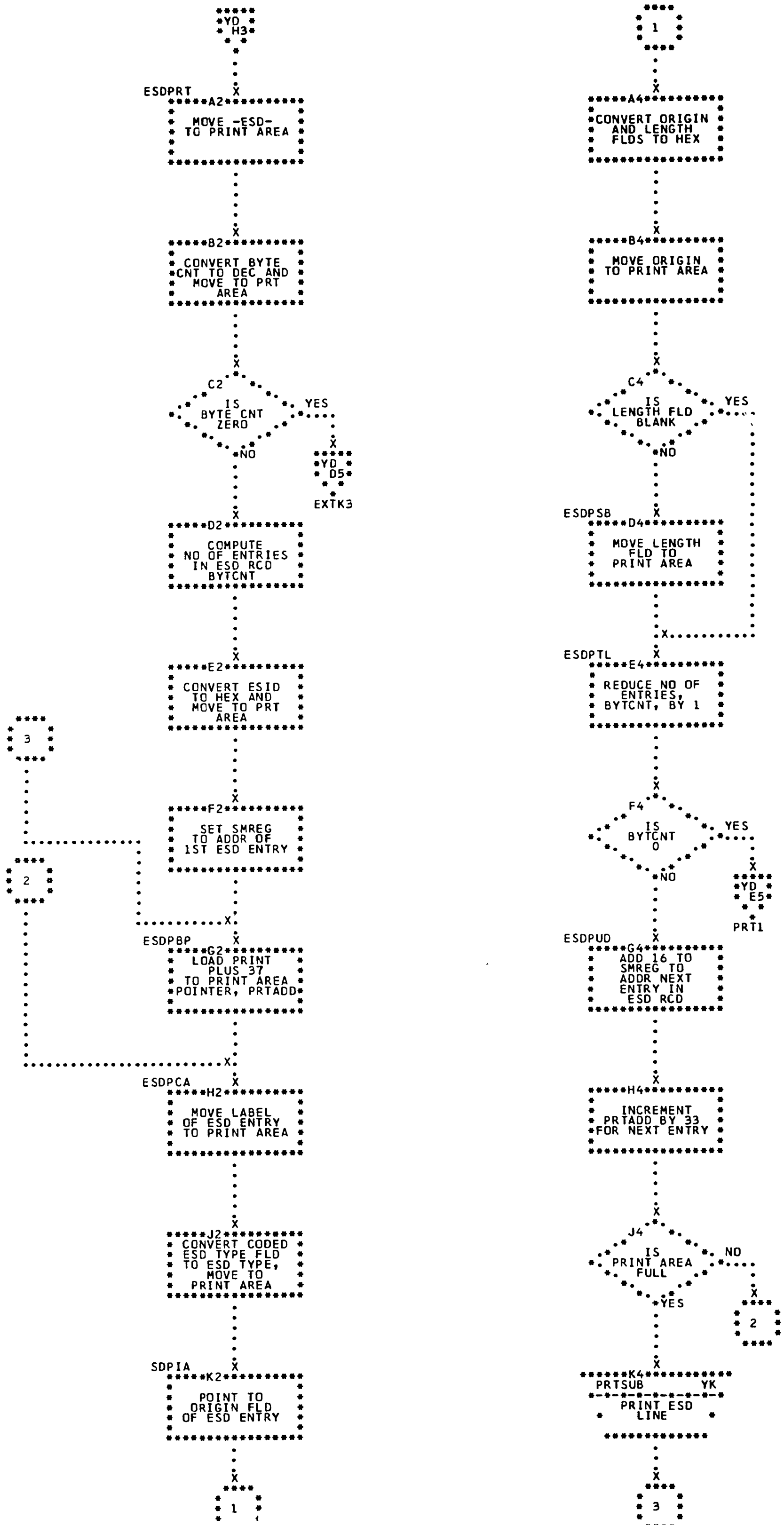


Chart YG. Punch and/or Print TXT Record RSERV; Refer to Service, Chart 49

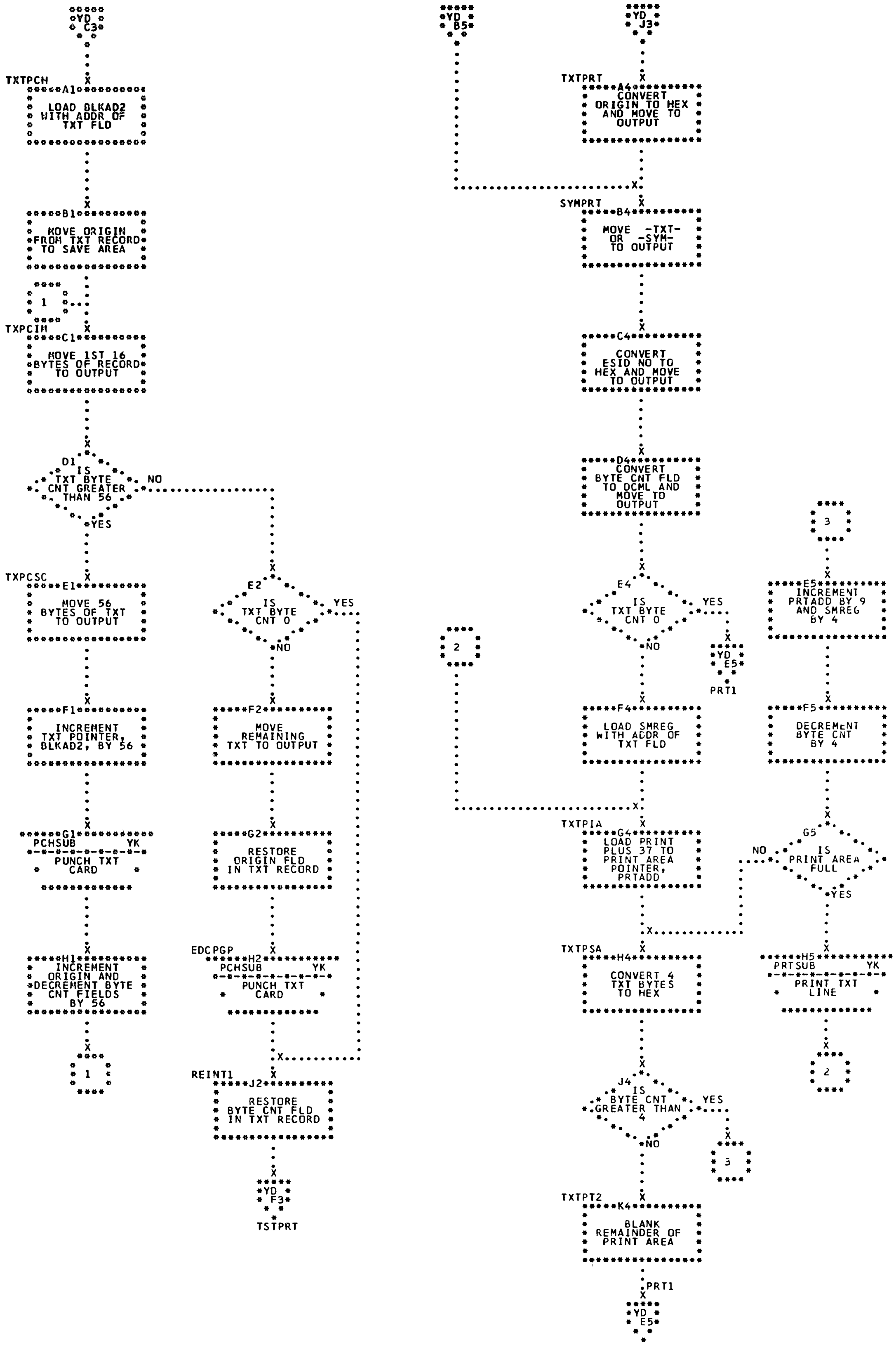


Chart YH. Punch RLD Record RSERV; Refer to Service, Chart 49

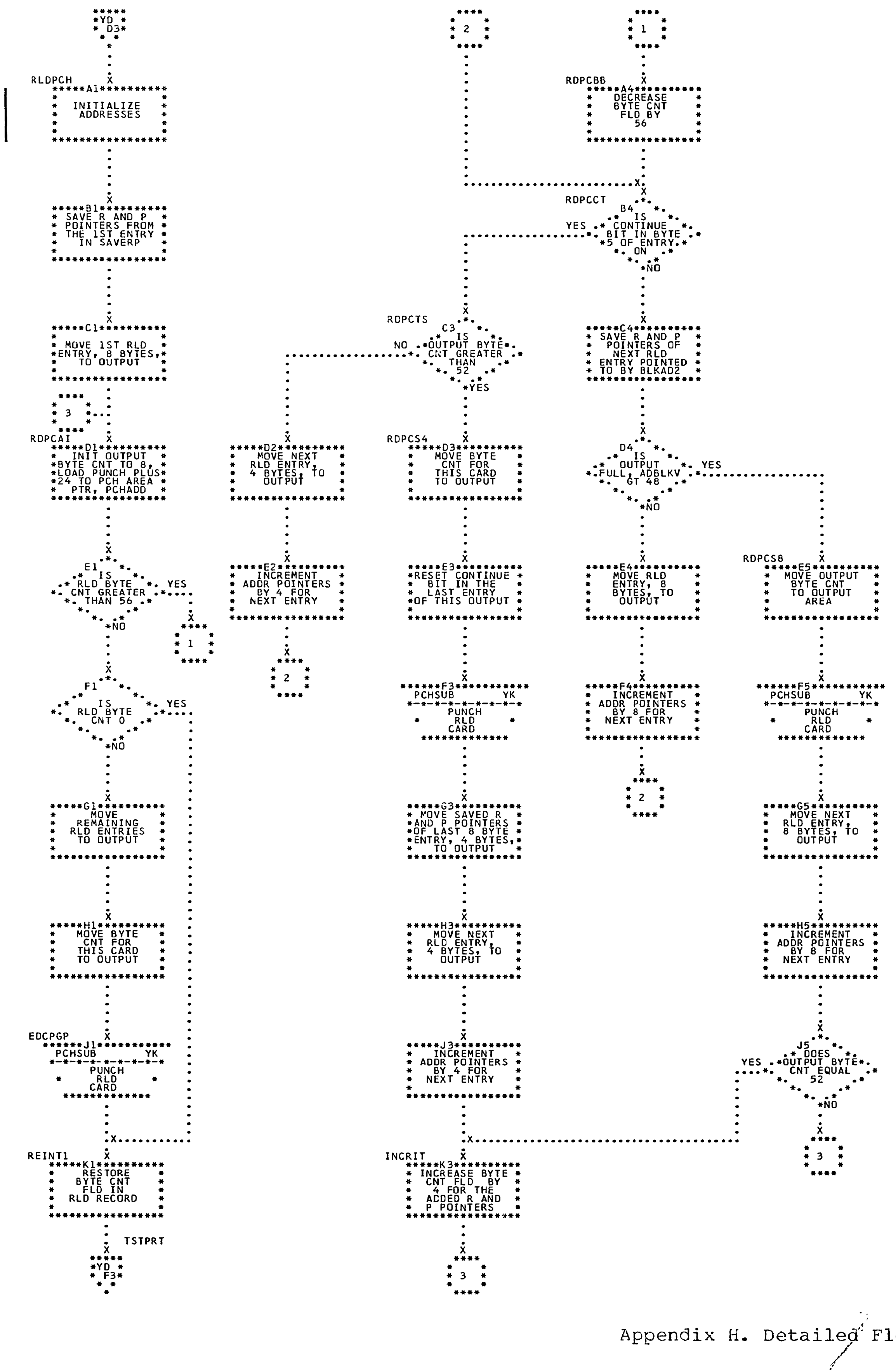
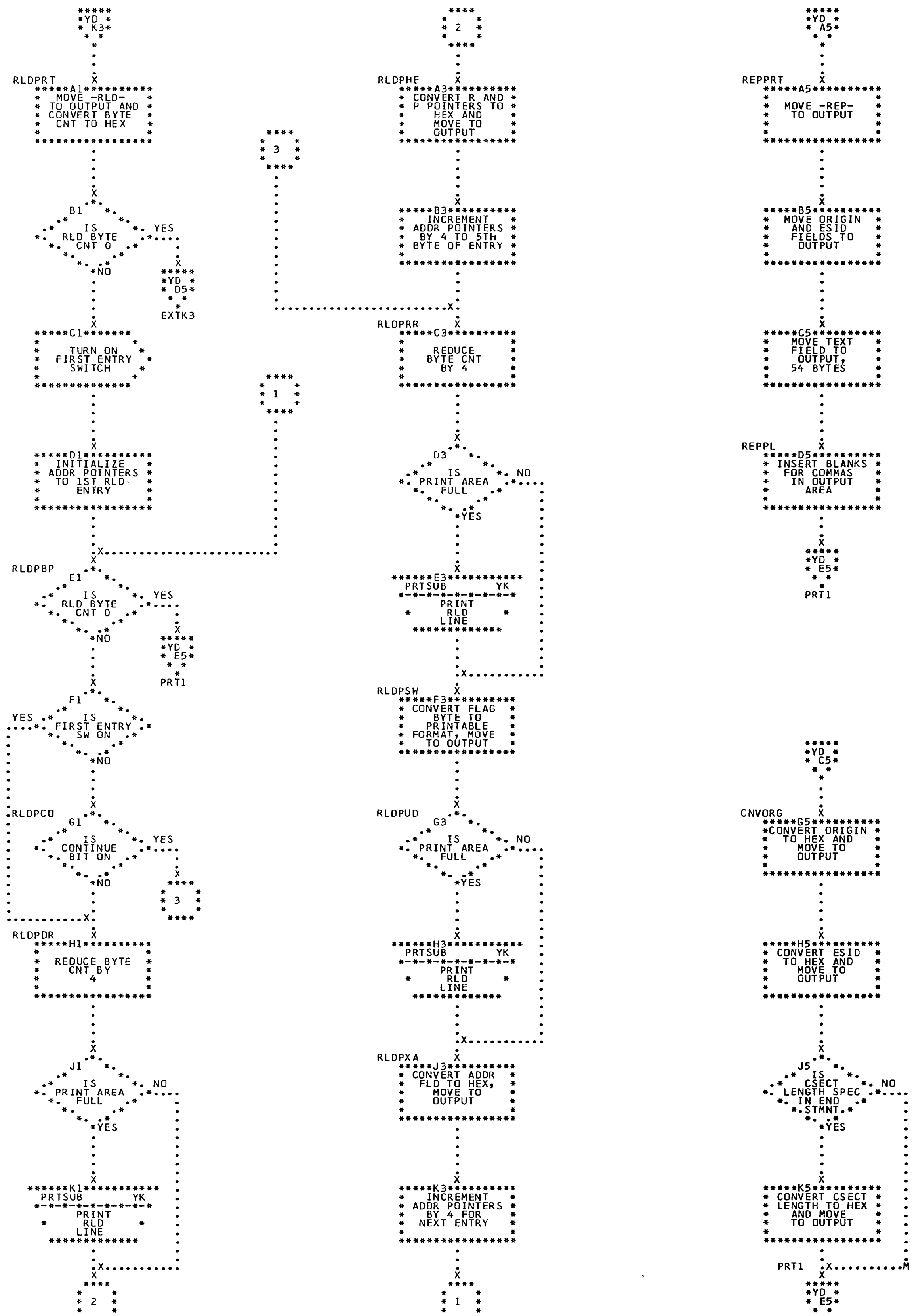


Chart YJ. Print RLD Record RSERV; Refer to Service, Chart 49



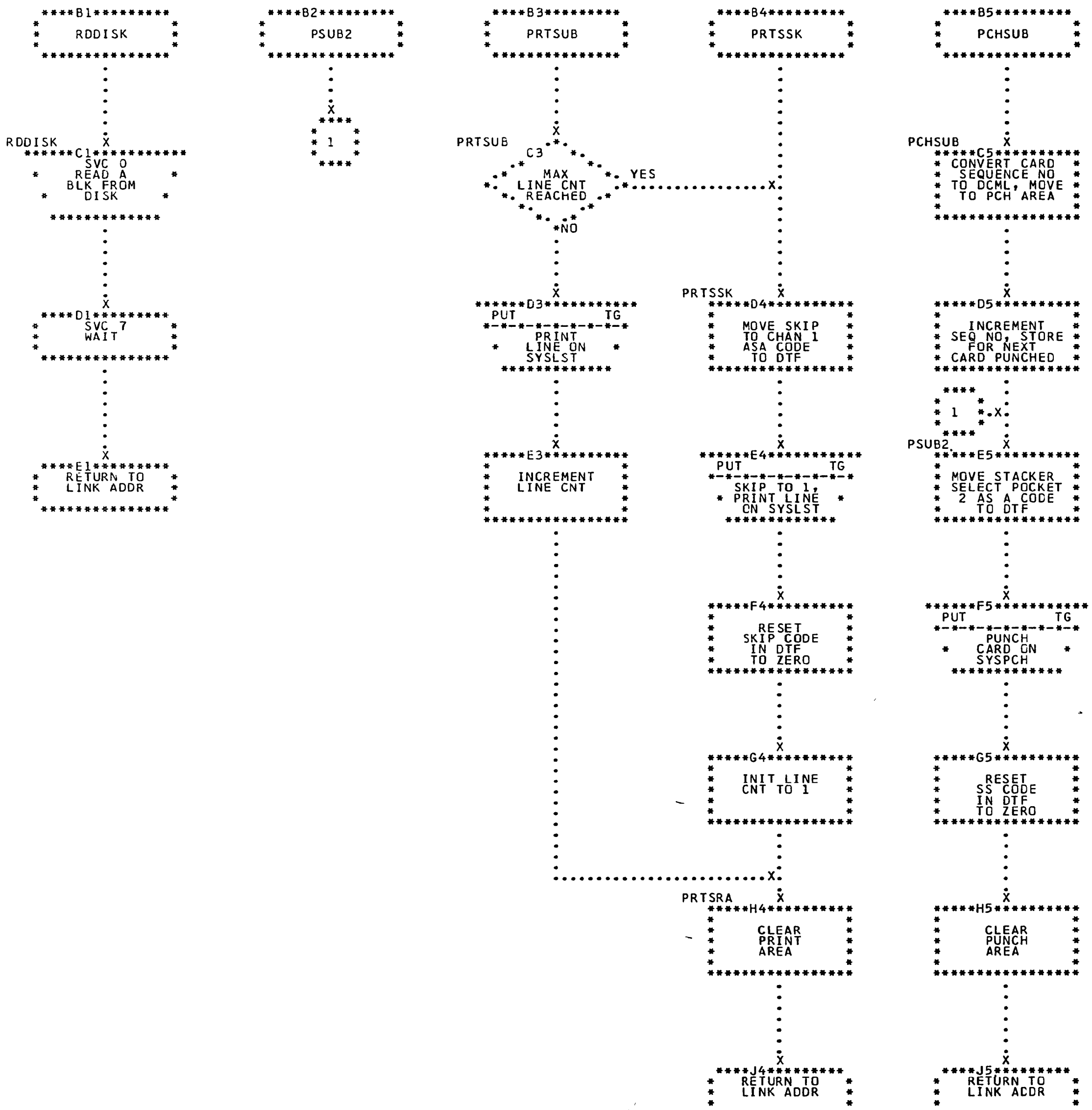

B1 RDDISK
RDDISK C1 SVC 0 READ A BLK FROM DISK
D1 SVC 7 WAIT
E1 RETURN TO LINK ADDR
B2 PSUB2
1
B3 PRTSUB
PRTSUB C3 MAX LINE CNT REACHED
YES
NO
D3 PUT TG PRINT LINE ON SYSLST
E3 INCREMENT LINE CNT
B4 PRTSSK
PRTSSK D4 MOVE SKIP TO CHAN 1 ASA CODE TO DTF
E4 PUT TG SKIP TO 1, PRINT LINE ON SYSLST
F4 RESET SKIP CODE IN DTF TO ZERO
G4 INIT LINE CNT TO 1
PRTSRA H4 CLEAR PRINT AREA
J4 RETURN TO LINK ADDR
B5 PCHSUB
PCHSUB C5 CONVERT CARD SEQUENCE NO TO DCML, MOVE TO PCH AREA
D5 INCREMENT SEQ NO, STORE FOR NEXT CARD PUNCHED
1
PSUB2 E5 MOVE STACKER SELECT POCKET 2 AS A CODE TO DTF
F5 PUT TG PUNCH CARD ON SYSPCH
G5 RESET SS CODE IN DTF TO ZERO
H5 CLEAR PUNCH AREA
J5 RETURN TO LINK ADDR

Chart YL. Scan Control Statements RSERV; Refer to Service, Chart 49

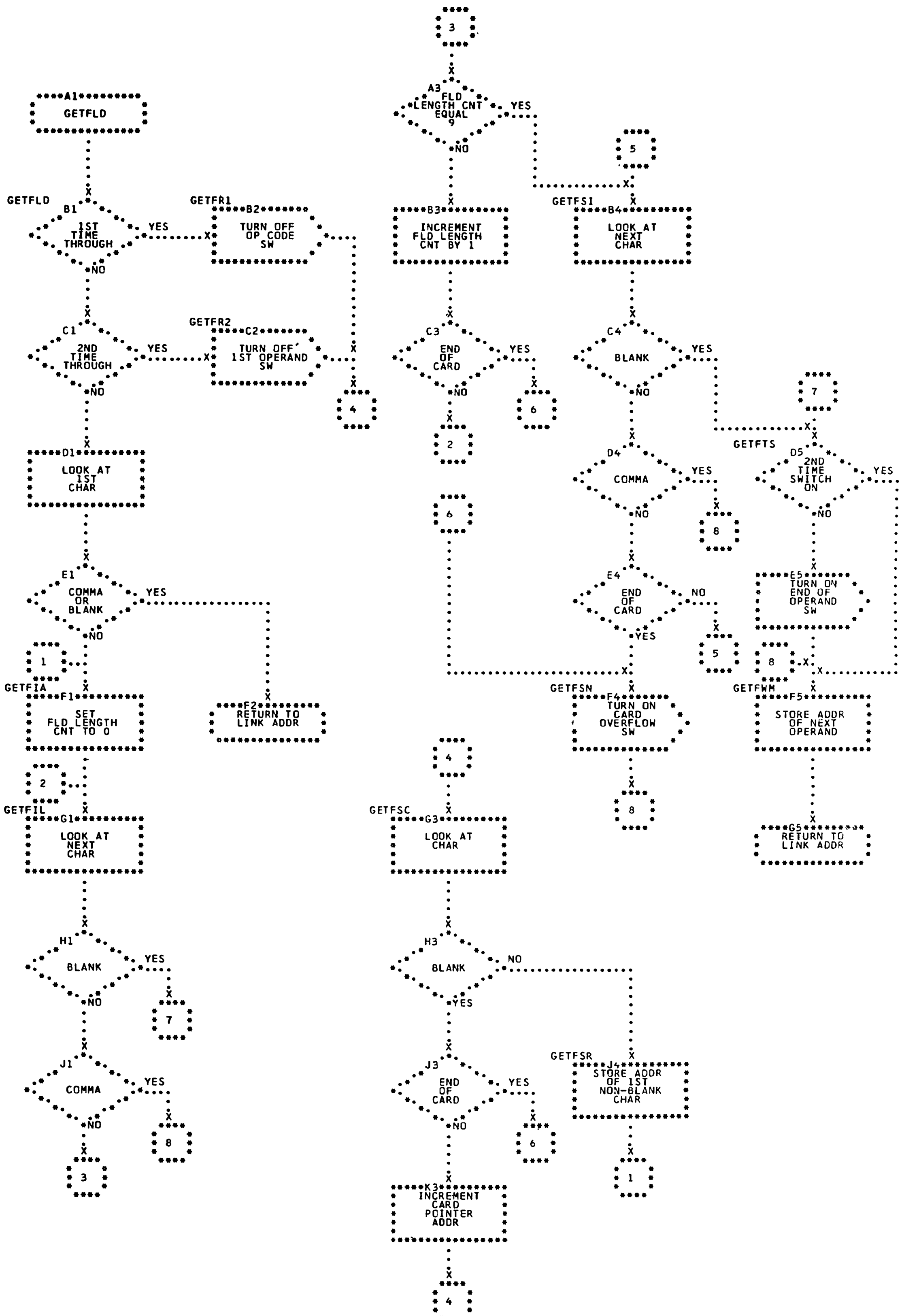


Chart ZA. Analyze Control Statements SSERV (Part 1 of 2); Refer to Service, Chart 50

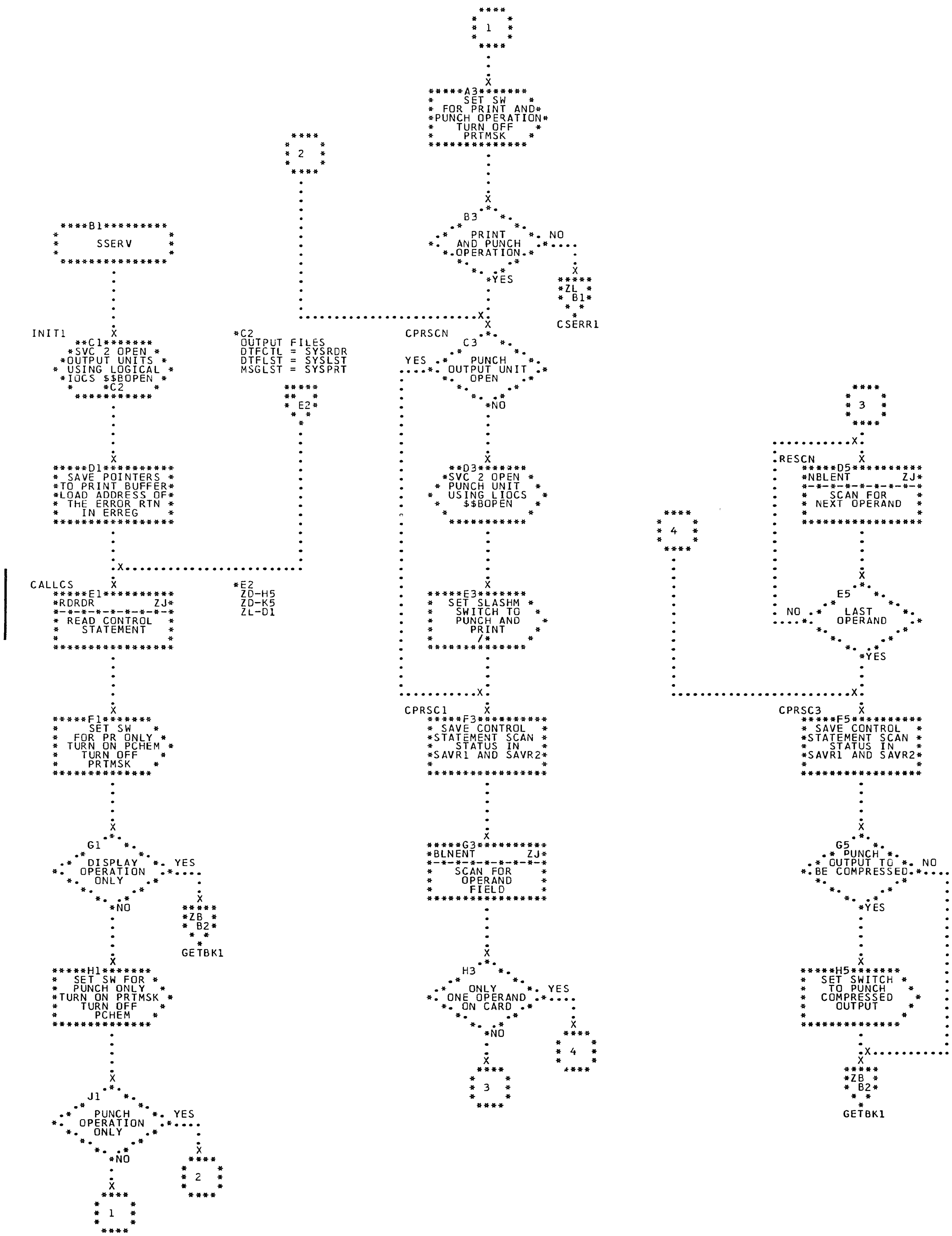


Chart ZB. Analyze Control Statements SSERV (Part 2 of 2); Refer to Service, Chart 50

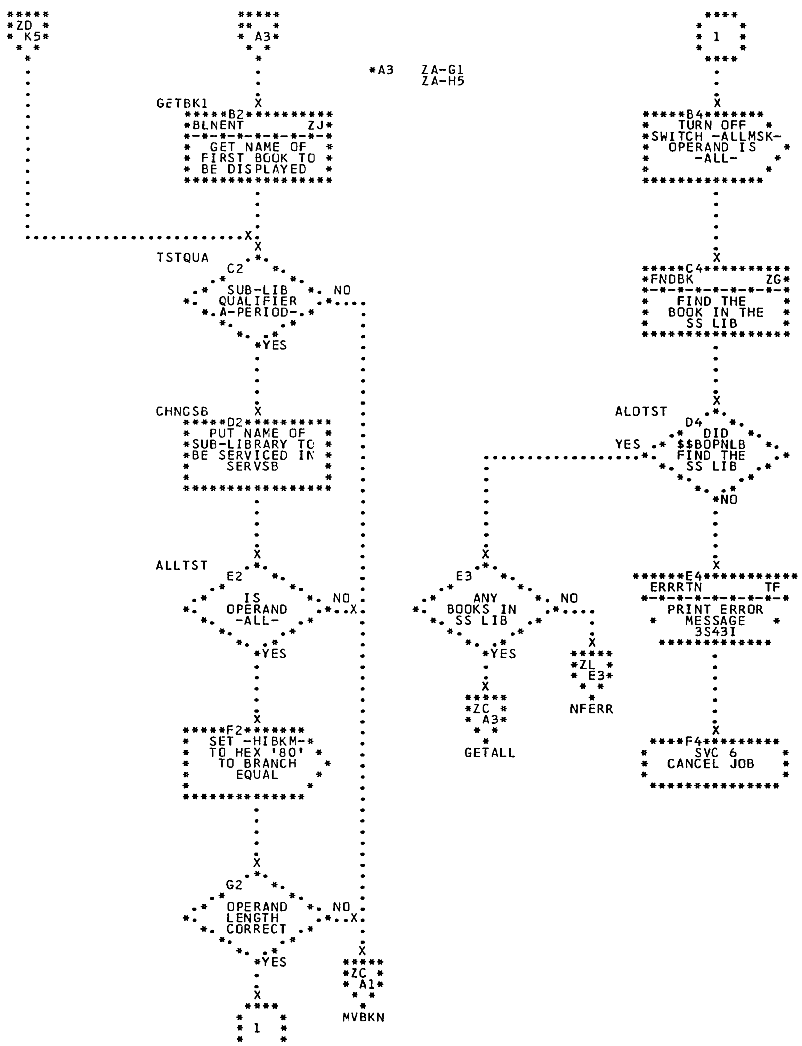


Chart ZC. Get Card Images and Load Output Buffers SSERV; Refer to Service, Chart 50

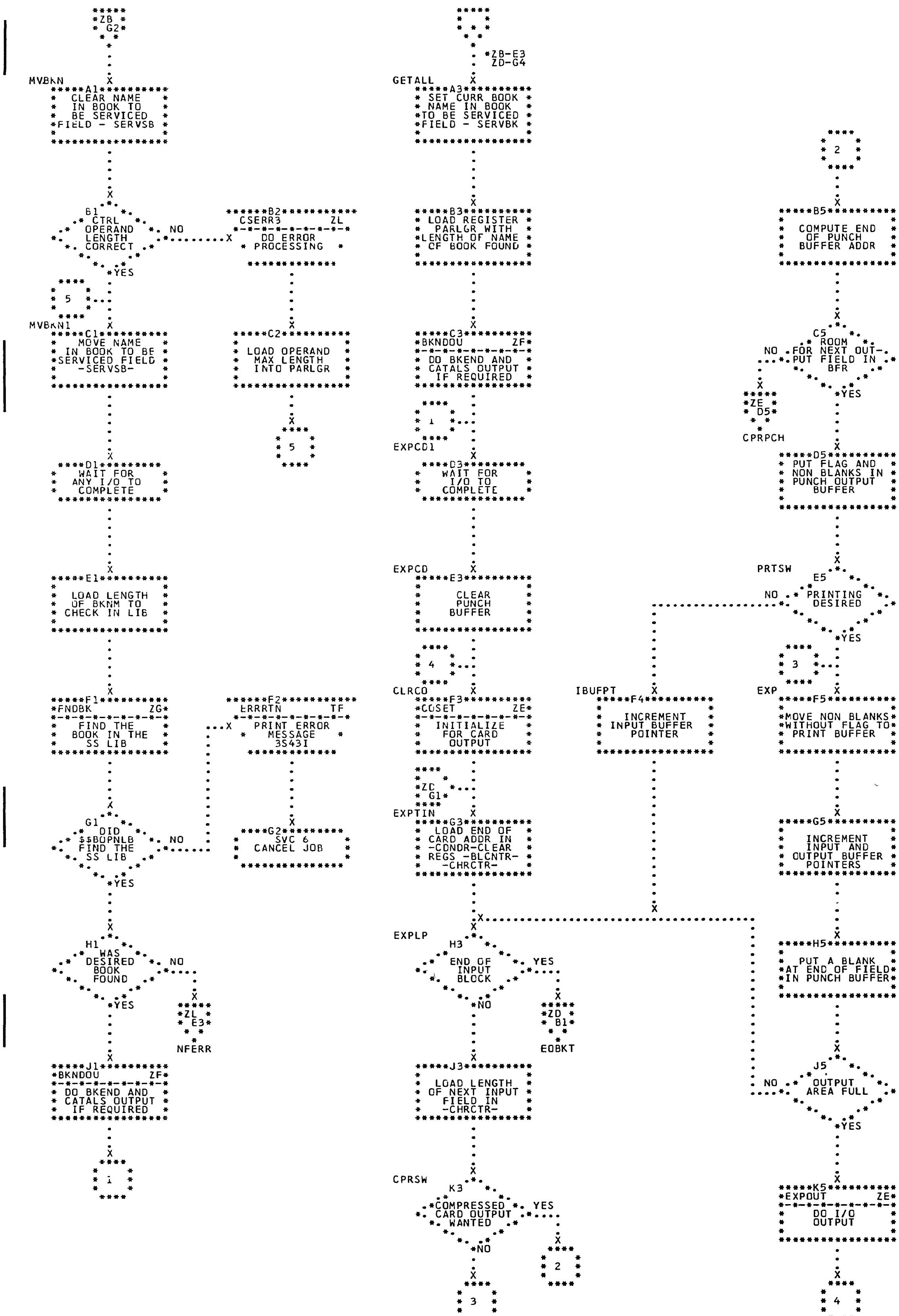


Chart ZD. I/O Input Control SSERV; Refer to Service, Chart 50

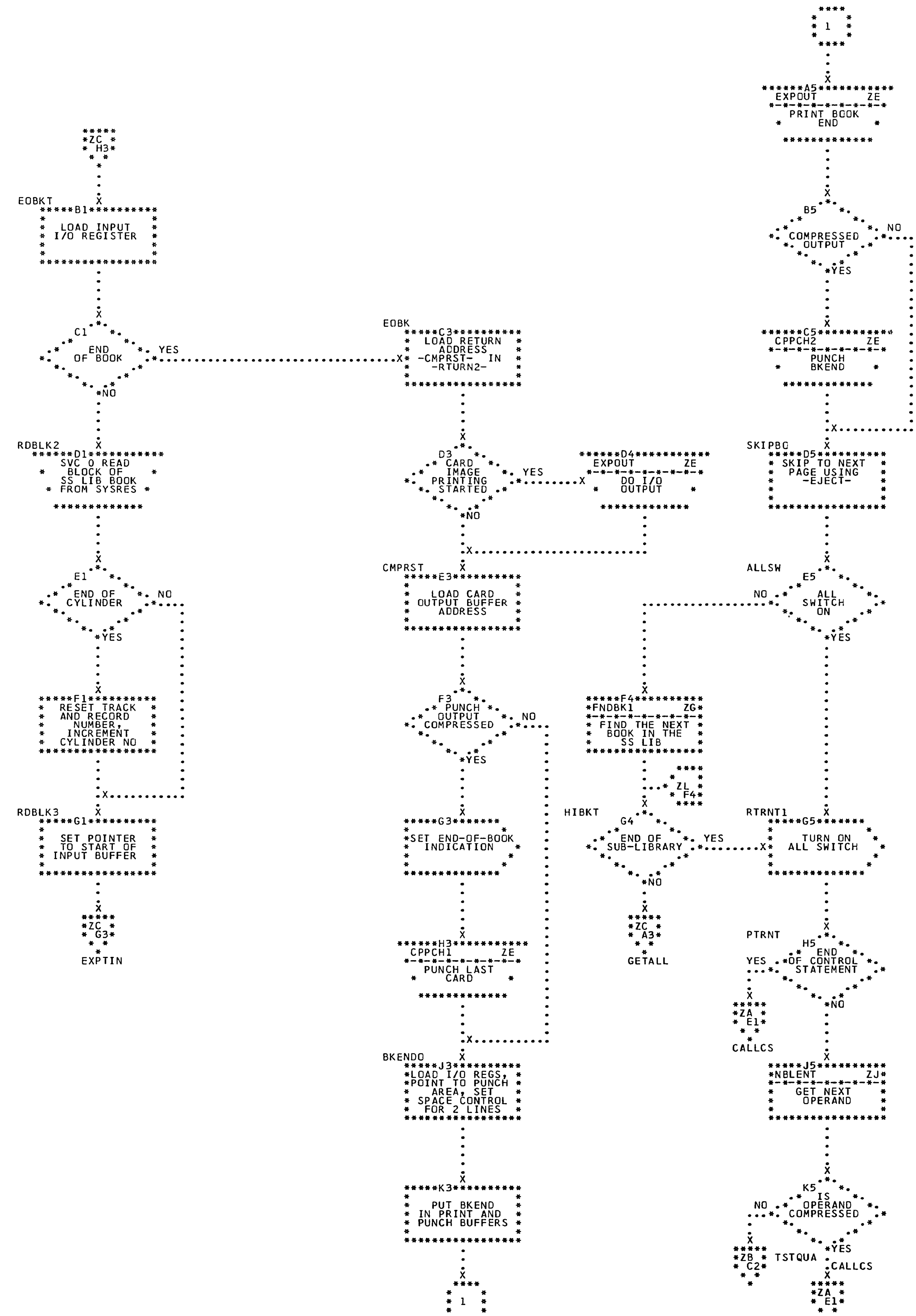


Chart ZE. Output SSERV; Refer to Service, Chart 50

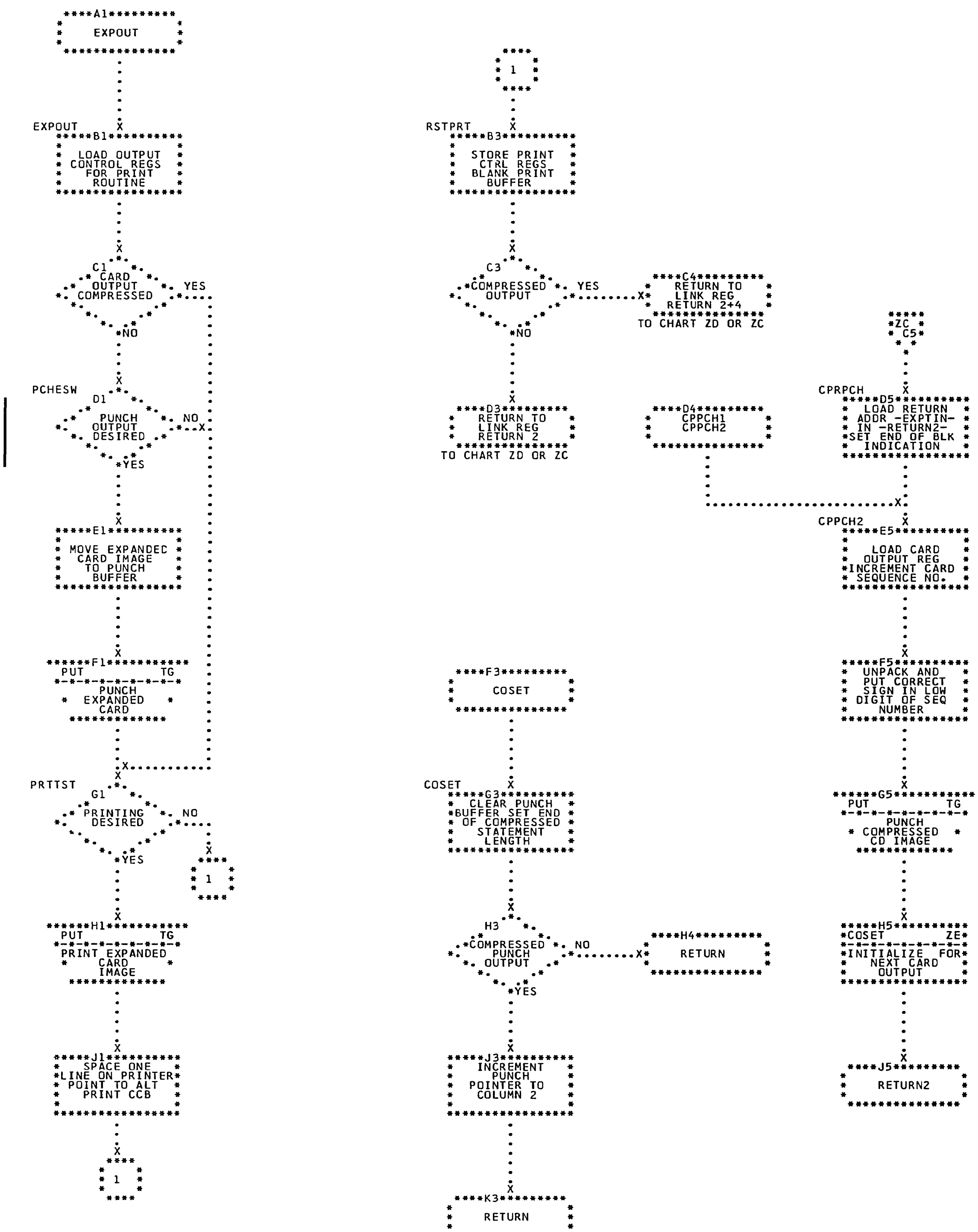


Chart ZF. Heading Control SSERV; Refer to Service, Chart 50

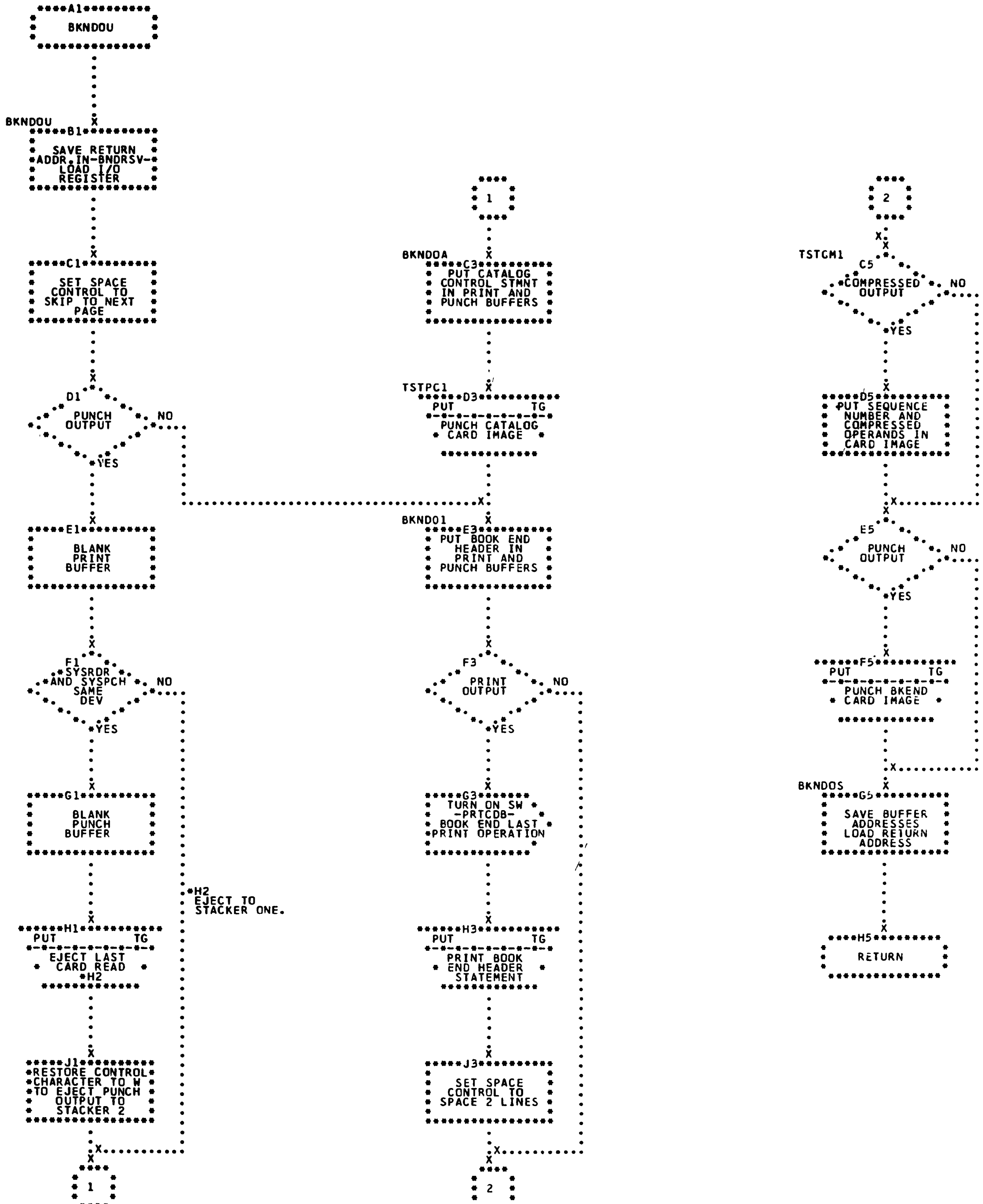


Chart ZG. Find Book SSERV; Refer to Service, Chart 50

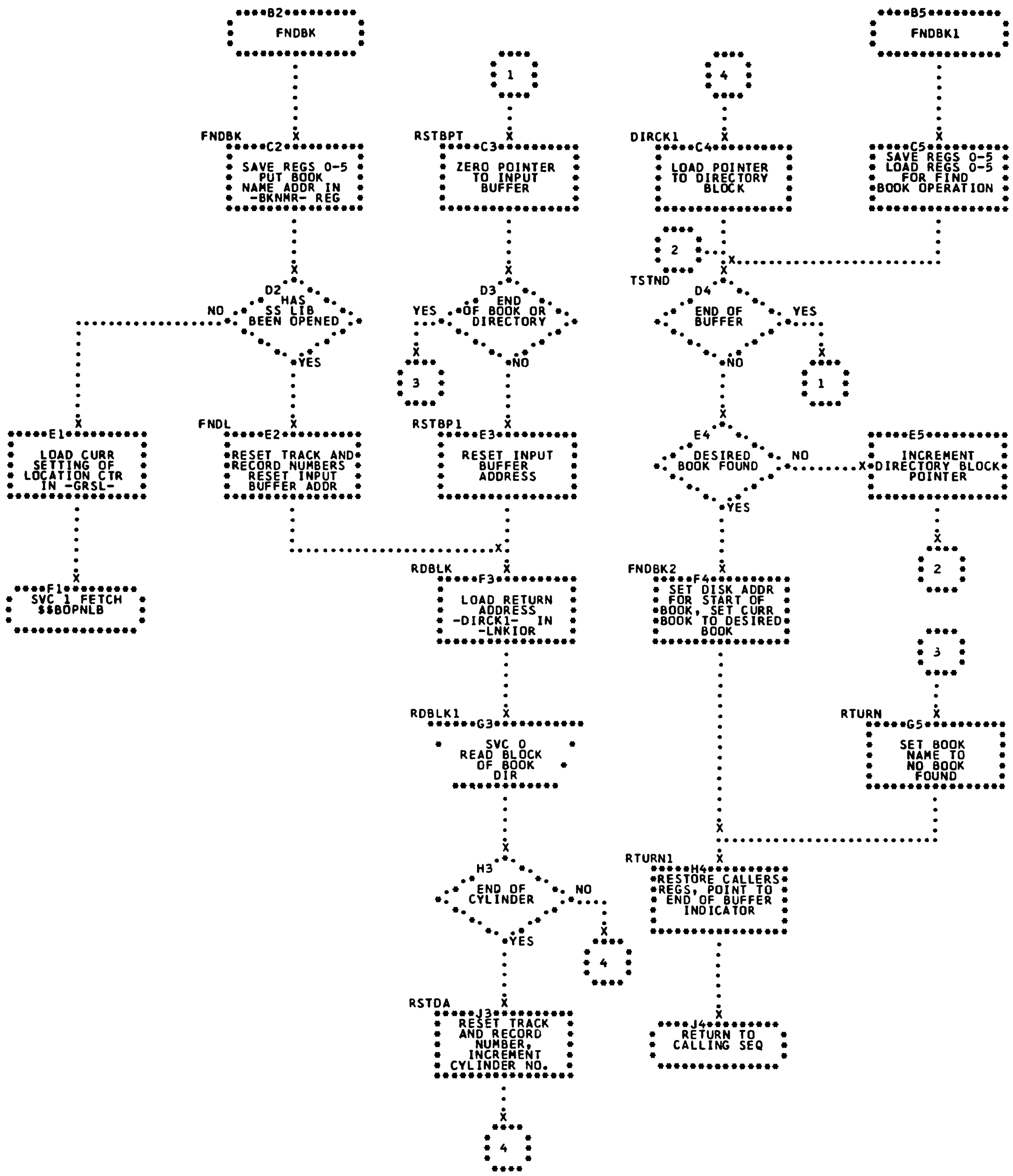


Chart ZH. $$BOPNLB Transient Program to Open Source Statement Library SSERV; Refer to Service, Chart 50

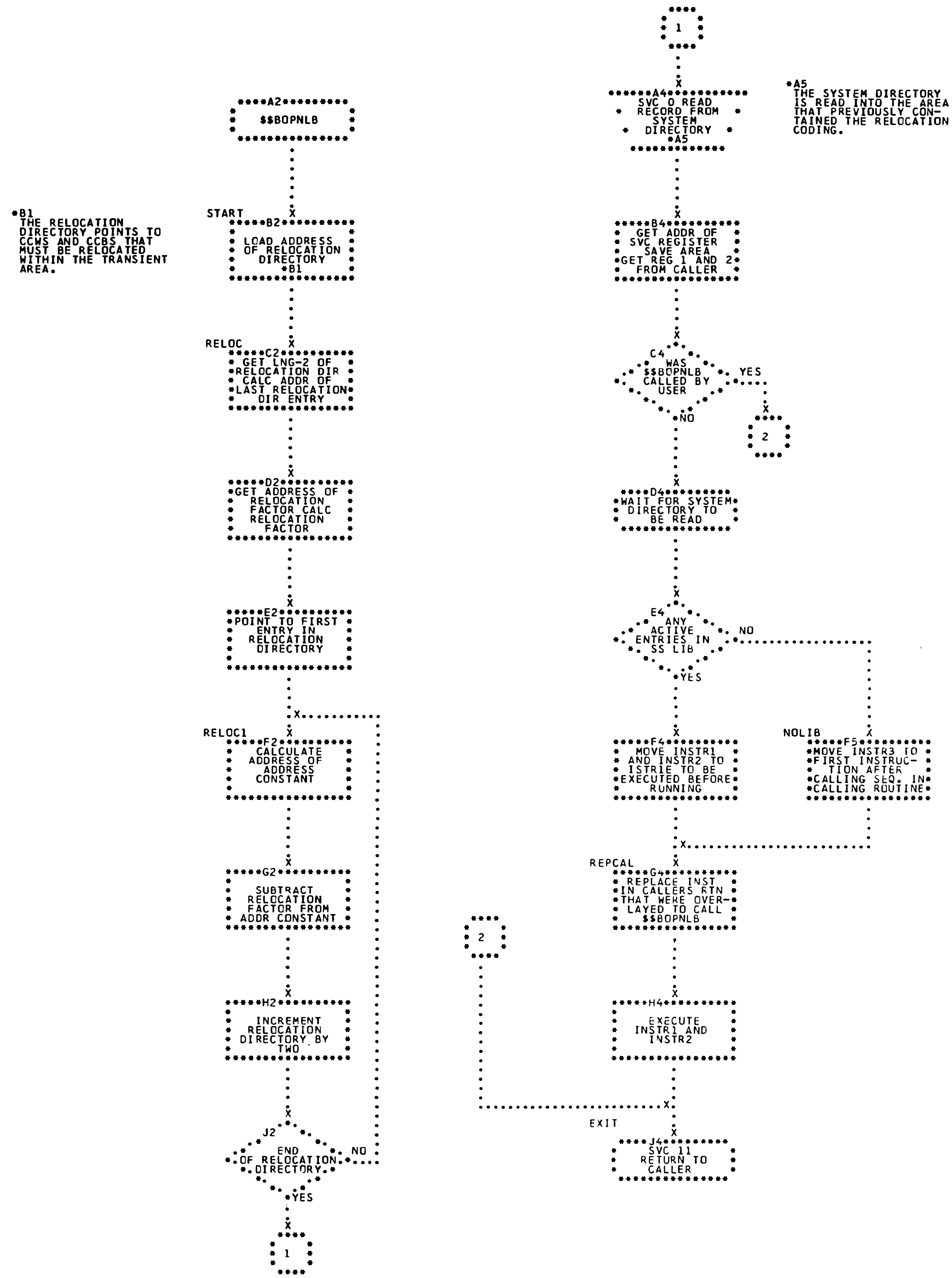


Chart ZJ. Read Control Statements and Scan for Operands
SSERV; Refer to Service, Chart 50

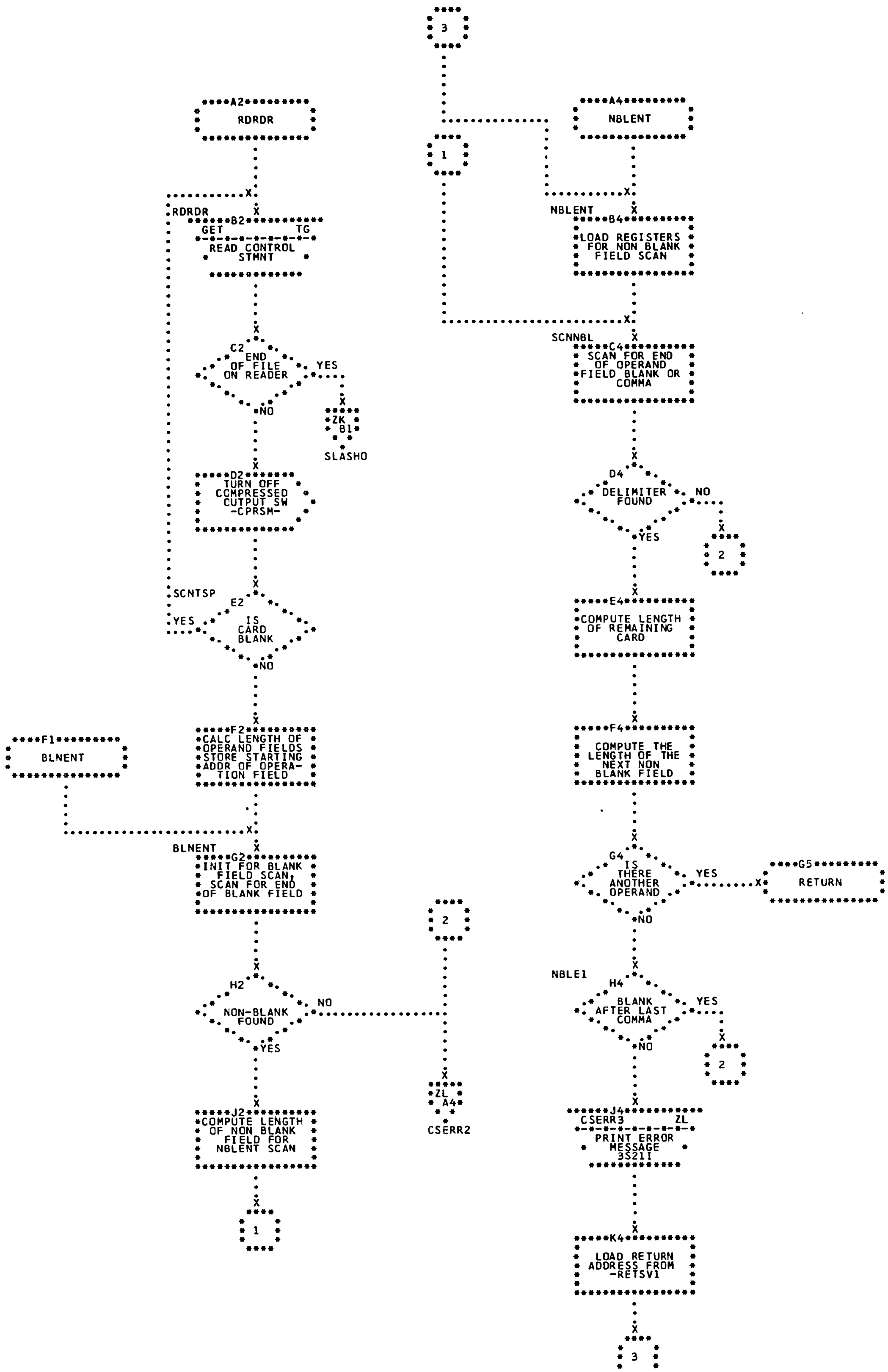


Chart ZK. EOF on SYSRDR, SYSLST, and SYSPCH SSERV; Refer to Service, Chart 50

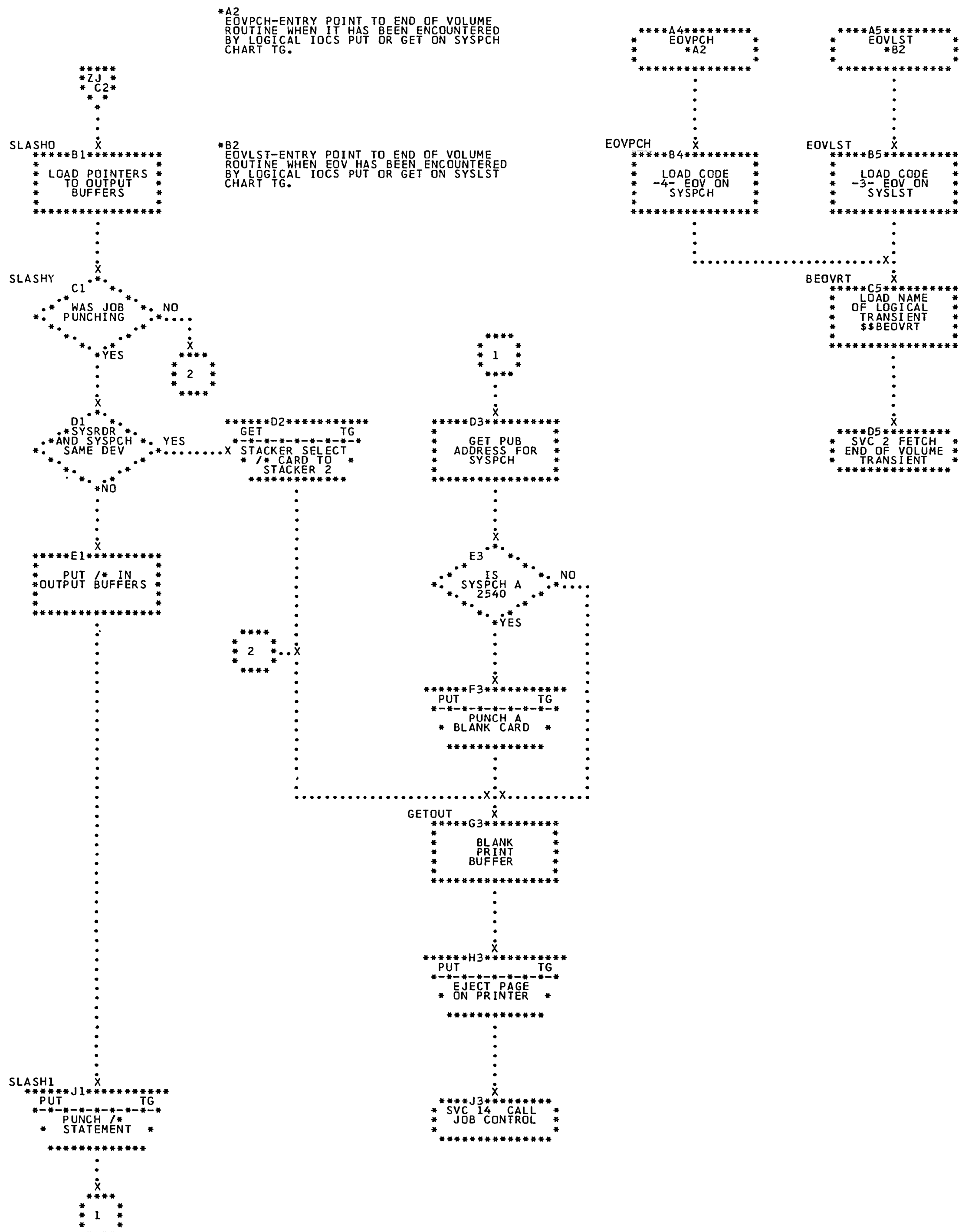
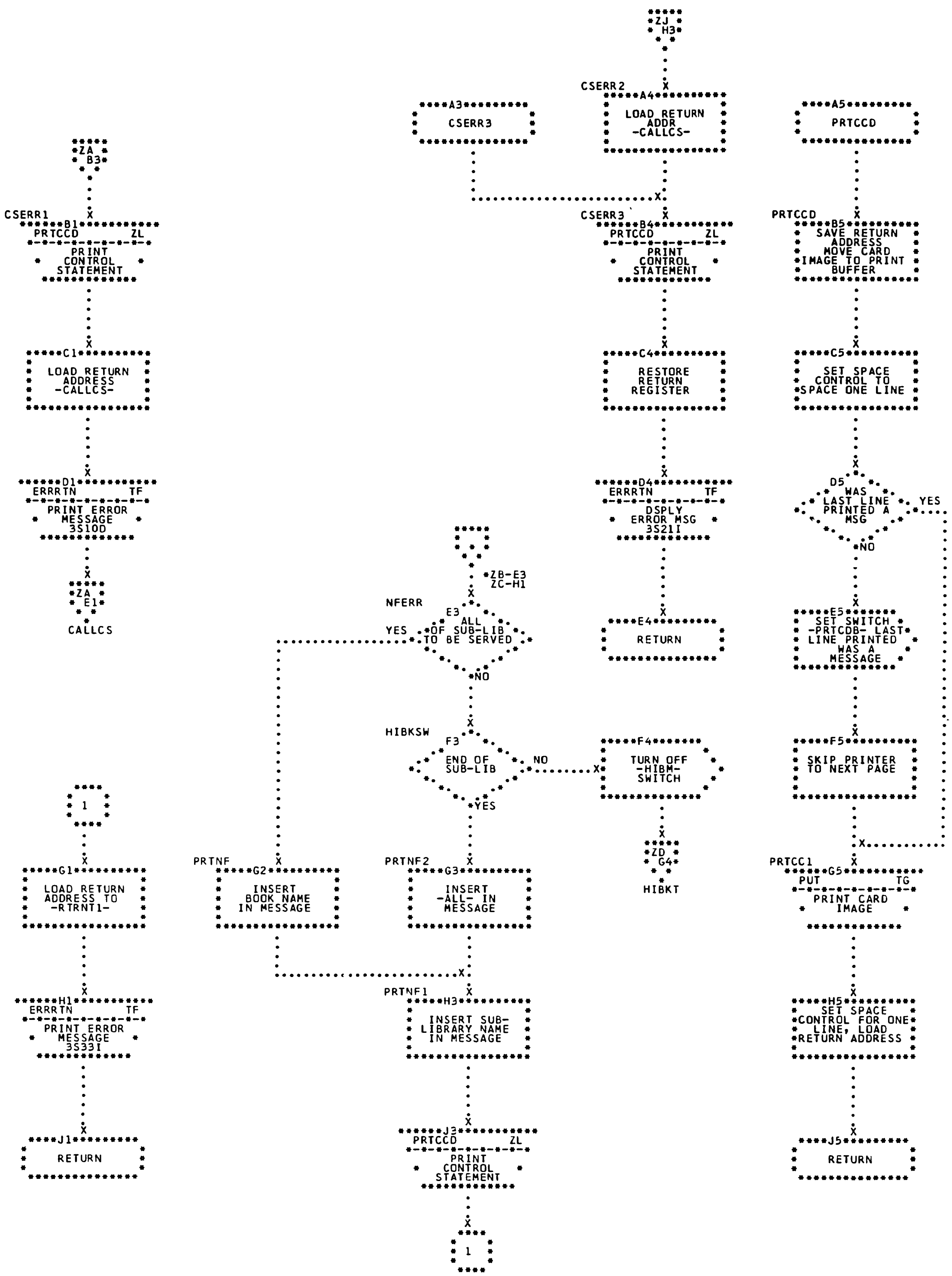


Chart ZL. Error Routines SSERV; Refer to Service, Chart 50



# APPENDIX I: MICROFICHE INDEX CROSS-REFERENCE LIST

This list can be used to relate core-image phase names to the labels used as identification on microfiche. The names are grouped by program type such as System Control, Autotest, FORTRAN, etc.

In some cases the program or portion of program displayed on one microfiche card may have no core image phase name; in other cases, it may have no relocatable module name. In every case, the microfiche identification is given and the user can relate this identification to either a core image phase name or a relocatable module name, or both. The last two numbers (10, 20) of some modules indicate the version and modification level and are subject to change.

Contents:



System Control Programs

Program Number 360N-CL-453

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| $$A$IPL1 | $$A$IPL1 | None |
| $$ANERRA | $$ANERRA | None |
| $$ANERRB | $$ANERRA | None |
| $$ANERRC | $$ANERRA | None |
| $$ANERRD | $$ANERRD | None |
| $$ANERRE | $$ANERRE | None |
| $$ANERRF | $$ANERRF | None |
| $$ANERRG | $$ANERRG | None |
| $$ANERRH | $$ANERRH | None |
| $$ANERRI | $$ANERRI | None |
| $$ANERRJ | $$ANERRJ | None |
| $$ANERRK | $$ANERRK | None |
| $$ANERRL | $$ANERRL | None |
| $$ANERRM | $$ANERRM | None |
| $$ANERRN | $$ANERRN | None |
| $$ANERRO | $$ANERRO | None |
| $$ANERRP | $$ANERRP | None |
| $$ANERRQ | $$ANERRQ | None |
| $$ANERRR | $$ANERRR | None |
| $$ANERRS | $$ANERRS | None |
| $$ANERRU | $$ANERRU | None |
| $$ANERRV | $$ANERRV | None |
| $$ANERRX | $$ANERRX | None |
| $$ANERRY | $$ANERRY | None |
| $$ANERRZ | $$ANERRY | None |
| $$ANERR0 | $$ANERRY | None |
| $$ANERR1 | $$ANERR1 | None |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name | Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|---|---|---|
| $$BATTNA | $$BATTNA | None | $$BDENDFF | $$BDENDFF | None |
| $$BATTNB | $$BATTNA | None | $$BDENDFL | $$BDENDFL | None |
| $$BATTNC | $$BATTNA | None | $$BEOJ | $$BEOJ | None |
| $$BATTND | $$BATTNA | None | $$BEOJ1 | $$BEOJ1 | None |
| $$BATTNE | $$BATTNA | None | $$BEOJ2 | $$BEOJ2 | None |
| $$BATTNF | $$BATTNA | None | $$BEOJ3 | $$BEOJ | None |
| $$BATTNG | $$BATTNA | None | $$BERRTN | $$BERRTN | None |
| $$BATTNH | $$BATTNA | None | $$BILSVC | $$BILSVC | None |
| $$BATTNI | $$BATTNI | None | $$BLSTIO | $$BLSTIO | None |
| $$BATTNJ | $$BATTNJ | None | $$BJCOPT | $$BJCOPT | None |
| $$BATTNK | $$BATTNK | None | $$BOCP01 | $$BOCP01 | None |
| $$BATTNL | $$BATTNL | None | $$BOCP02 | $$BOCP02 | None |
| $$BATTNM | $$BATTNM | None | $$BOCP11 | $$BOCP11 | None |
| $$BATTNN | $$BATTNN | None | $$BOCP12 | $$BOCP12 | None |
| $$BCEOV1 | $$BCEOV1 | None | $$BODAIN | $$BODAIN | None |
| $$BCHKPD | $$BCKPD | None | $$BODAO1 | $$BODAO1 | None |
| $$BCHKPE | $$BCHKPE | None | $$BODAO2 | $$BODAO2 | None |
| $$BCHKPT | $$BCHKPT | None | $$BODAO3 | $$BODAO3 | None |
| $$BCISOA | $$BCISOA | None | $$BODAU1 | $$BODAU1 | None |
| $$BCLOSE | $$BCLOSE | None | $$BODQUE | $$BODQUE | None |
| $$BCLOSP | $$BCLOSP | None | $$BODSPV | $$BODSPV | None |
| $$BCMT01 | $$BCMT01 | None | $$BODSPW | $$BODSPW | None |
| $$BCMT02 | $$BCMT02 | None | $$BOFLPT | $$BOFLPT | None |
| $$BCMT03 | $$BCMT03 | None | $$BOIS01 | $$BOIS01 | None |
| $$BCMT04 | $$BCMT04 | None | $$BOIS02 | $$BOIS02 | None |
| $$BCMT05 | $$BCMT05 | None | $$BOIS03 | $$BOIS03 | None |
| $$BCMT06 | $$BCMT06 | None | $$BOIS05 | $$BOIS05 | None |
| $$BCMT07 | $$BCMT07 | None | $$BOIS06 | $$BOIS06 | None |
| $$BDRSTR | $$BDRSTR | None | $$BOIS07 | $$BOIS07 | None |
| $$BDUMP | $$BDUMP | None | $$BOMSG1 | $$BOMSG1 | None |
| $$BDUMPB | $$BDUMPB | None | $$BOMSG2 | $$BOMSG2 | None |
| $$BDUMPD | $$BDUMPD | None | $$BOMSG3 | $$BOMSG3 | None |
| $$BDUMPF | $$BDUMPF | None | $$BOMSG4 | $$BOMSG4 | None |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name | Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|---|---|---|
| $$BOMSG5 | $$BOMSG5 | None | $$BPDUM1 | $$BPDUM1 | None |
| $$BOMT01 | $$BOMT01 | None | $$BPSW | $$BPSW | None |
| $$BOMT02 | $$BOMT02 | None | $$BRMSG1 | $$BRMSG1 | None |
| $$BOMT03 | $$BOMT03 | None | $$BRMSG2 | $$BRMSG2 | None |
| $$BOMT04 | $$BOMT04 | None | $$BRSTRT | $$BRSTRT | None |
| $$BOMT05 | $$BOMT05 | None | $$BRSTR2 | $$BRSTR2 | None |
| $$BOMT06 | $$BOMT06 | None | $$BSETFF | $$BSETFF | None |
| $$BOMT11 | $$BOMT11 | None | $$BSETFL | $$BSETFL | None |
| $$BOPEN | $$BOPEN | None | $$BSETL | $$BSETL | None |
| $$BOPENC | $$BOPENC | None | $$BSYSWR | $$BSYSWR | None |
| $$BOPENR | $$BOPENR | None | $$BTERM | $$BTERM | None |
| $$BOPEN2 | $$BOPEN2 | None | $$IPLRT2 | IJBIPL | IJBIPL |
| $$BOPNLB | $$BOPNLB | None | $JOBCTLA | IJBJC1 | IJBJC1 |
| $$BOSDC1 | $$BOSDC1 | None | $JOBCTLD | IJBJC2 | IJBJC2 |
| $$BOSDI1 | $$BOSDI1 | None | $JOBCTLG | IJBJC3 | IJBJC3 |
| $$BOSDI2 | $$BOSDI2 | None | $JOBCTLJ | IJBJC4 | IJBJC4 |
| $$BOSDI3 | $$BOSDI3 | None | $LNKEDT | IJBLE1 | IJBLE1 |
| $$BOSDO1 | $$BOSDO1 | None | $LNKEDTA | IJBLE1 | IJBLE1 |
| $$BOSDO2 | $$BOSDO2 | None | $LNKEDTC | IJBLBI | IJBLBI |
| $$BOSDO3 | $$BOSDO3 | None | $LNKEDT0 | IJBLE1 | IJBLE1 |
| $$BOSDO4 | $$BOSDO4 | None | $LNKEDT2 | IJBLE1 | IJBLE1 |
| $$BOSDO5 | $$BOSDO5 | None | $LNKEDT4 | IJBLE1 | IJBLE1 |
| $$BOSDO6 | $$BOSDO6 | None | $LNKEDT6 | IJBLE1 | IJBLE1 |
| $$BOSDO7 | $$BOSDO7 | None | $LNKEDT8 | IJBLE1 | IJBLE1 |
| $$BOSDO8 | $$BOSDO8 | None | $MAINEOJ | IJBLBH | IJBLBH |
| $$BOSDW1 | $$BOSDW1 | None | CORGZ | IJBLBJ | IJBLBJ |
| $$BOSDW2 | $$BOSDW2 | None | CORGZ2 | IJBLBK | IJBLBK |
| $$BOSDW3 | $$BOSDW3 | None | DSERV | IJBSL1 | IJBSL1 |
| $$BOSD00 | $$BOSD00 | None | IJBRSTRT | IJBRSTRT | IJBRSTRT |
| $$BOUR01 | $$BOUR01 | None | MAINT | IJBLBA | IJBLBA |
| $$BOVDMP | $$BOVDMP | None | | | IJJCPD0 |
| $$BPCHK | $$BPCHK | None | (Error Routine) | IJBLBC | IJBLBC |
| $$BPDUMP | $$BPDUMP | None | | | |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| MAINTA | IJBLBL | IJBLBL |
| MAINTCL | IJBLBM | IJBLBM |
| MAINTCN | IJBLBG | IJBLBG |
| MAINTC2 | IJBLBD | IJBLBD |
| MAINTR2 | IJBLBE | IJBLBE |
| MAINTS2 | IJBLBF | IJBLBF |
| RSERV | IJBSL3 | IJBSL3 |
| SSERV | IJBSL4 | IJBSL4 |

Autotest Program

Program Number 360N-PT-459

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| $$BATST1 | IJVSS1 | IJVSS110 |
| $$BATST3 | IJVSS3 | IJVSS310 |
| ATLECONT | IJVTA0 | IJVTA010 |
| ATLEDT | IJVLE | IJVLE |
| ATLEDT1A | IJVLE | IJVLE |
| ATLEDT1B | IJVLE | IJVLE |
| ATLEDT1C | IJVLE | IJVLE |
| ATLEDT10 | IJVLE | IJVLE |
| ATLEDT12 | IJVLE | IJVLE |
| ATLEDT14 | IJVLE | IJVLE |
| ATLEDT16 | IJVLE | IJVLE |
| ATLEDT18 | IJVLE | IJVLE |
| *(See Note) | IJVTAB | IJVTAB |
| ATLEFC1 | IJVTC1 | IJVTC110 |
| ATLEFC2 | IJVTC2 | IJVTC210 |
| ATLEFC3 | INVTC3 | IJVTC310 |
| ATLEFC4 | IJVTC4 | IJVTC410 |
| ATLEFC5 | IJVTC5 | IJVTC510 |
| ATLEFC7 | IJVTC7 | IJVTC710 |
| ATLEFD1 | IJVTD1 | IJVTD110 |
| ATLEFD2 | IJVTD2 | IJVTD210 |
| ATLEFE1 | IJVTE1 | IJVTE110 |
| ATLEFE2 | IJVTE2 | IJVTE210 |
| ATLEFF1 | IJVTF1 | IJVTF110 |
| ATLEFG1 | IJVTG1 | IJVTG110 |
| ATLEFH2 | IJVTH2 | IJVTH210 |
| ATLEFH3 | IJVTH3 | IJVTH310 |
| ATLEGO1 | IJVTI1 | IJVTI110 |
| ATLEJCTV | IJVTJ1 | IJVTJ110 |

*Can be included as part of other Autotest phases as required.

BTAM

Program Number 360N-CQ-469

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| $$ANERR2 | $$ANERR2 | None |
| $$BCTC01 | $$BCTC01 | None |
| $$BETPRT | $$BETPRT | None |
| $$BHDRCK | $$BHDRCK | None |
| $$BLEPRT | $$BLEPRT | None |
| $$BLOPEN | $$BLOPEN | None |
| $$BOTC01 | $$BOTC01 | None |
| $$BTCNCL | $$BTCNCL | None |
| $$BTMEBG | $$BTMEBG | None |
| $$BT1030 | $$BT1030 | None |
| $$BT1050 | $$BT1050 | None |
| $$BT1060 | $$BT1060 | None |
| $$BT2260 | $$BT2260 | None |
| $$BT2740 | $$BT2740 | None |
| $$BT2848 | $$BT2848 | None |
| IJLT2ALC | IJLT2ALC | None |
| IJLT2ROT | IJLT2ROT | None |
| IJLT2TLT | IJLT2TLT | None |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| IJLT2TWS | IJLT2TWS | None |
| IJLT3ALC | IJLT3ALC | None |
| IJLT3ROT | IJLT3ROT | None |
| IJLT3SLA | IJLT3SLA | None |
| IJLT3TLT | IJLT3TLT | None |
| IJLT3TWS | IJLT3TWS | None |
| IJLT5ALC | IJLT5ALC | None |
| IJLT5ROT | IJLT5ROT | None |
| IJLT5SLA | IJLT5SLA | None |
| IJLT5TLT | IJLT5TLT | None |
| IJLT5TWS | IJLT5TWS | None |
| IJLT6ALC | IJLT6ALC | None |
| IJLT6ROT | IJLT6ROT | None |
| IJLT6SLA | IJLT6SLA | None |
| IJLT6TLT | IJLT6TLT | None |
| IJLT6TWS | IJLT6TWS | None |
| None | IJL0AY | IJL0AY |
| None | IJL00Y | IJL00Y |
| None | IJL01Z | IJL01Z |
| None | IJL02Z | IJL02Z |
| None | IJL03Z | IJL03Z |
| None | IJL04Z | IJL04Z |
| None | IJL05Z | IJL05Z |
| None | IJL07Y | IJL07Y |
| None | IJL07Z | IJL07Z |
| None | IJL08M | IJL08M |
| None | IJL08P | IJL08P |
| None | IJL08Q | IJL08Q |
| None | IJL08R | IJL08R |
| None | IJL08U | IJL08U |
| None | IJL08X | IJL08X |
| None | IJL08Y | IJL08Y |
| None | IJL08Z | IJL08Z |
| None | IJL09Y | IJL09Y |

## MPS Utilities

Program Number 360N-UT-471

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| $$BMU100 | $$BMU100 | None |
| $$BMU200 | $$BMU200 | None |
| $$BMU300 | $$BMU300 | None |

## Disk Sort/Merge

Program Number 360N-SM-450

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| DSORT | IJOSM001 | IJOSM001 |
| DSORT002 | IJOSM002 | IJOSM002 |
| DSORT003 | IJOSM003 | IJOSM003 |
| DSORT004 | IJOSM004 | IJOSM004 |
| DSORT005 | IJOSM005 | IJOSM005 |
| DSORT006 | IJOSM006 | IJOSM006 |
| DSORT007 | IJOSM007 | IJOSM007 |
| DSORT008 | IJOSM008 | IJOSM008 |
| DSORT009 | IJOSM009 | IJOSM009 |
| DSORT010 | IJOSM010 | IJOSM010 |
| DSORT101 | IJOSM101 | IJOSM101 |
| DSORT102 | IJOSM102 | IJOSM102 |
| DSORT103 | IJOSM103 | IJOSM103 |
| DSORT104 | IJOSM104 | IJOSM104 |
| DSORT105 | IJOSM105 | IJOSM105 |
| DSORT201 | IJOSM201 | IJOSM201 |
| DSORT202 | IJOSM202 | IJOSM202 |
| DSORT203 | IJOSM203 | IJOSM203 |
| DSORT204 | IJOSM204 | IJOSM204 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| DSORT301 | IJOSM301 | IJOSM301 |
| DSORT302 | IJOSM302 | IJOSM302 |
| DSORT303 | IJOSM303 | IJOSM303 |
| DSORT304 | IJOSM304 | IJOSM304 |
| DSORT401 | IJOSM401 | IJOSM401 |
| DSORT402 | IJOSM402 | IJOSM402 |

DOS Sort/Merge (Tape)

Program Number 360N-SM-400

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| TSRTP001 | TSRTP001 | IJPSM001 |
| TSRTP002 | TSRTP001 | IJPSM001 |
| TSRTP003 | TSRTP001 | IJPSM001 |
| TSRTP004 | TSRTP001 | IJPSM001 |
| TSRTP005 | TSRTP001 | IJPSM001 |
| TSRTP006 | TSRTP001 | IJPSM001 |
| TSRTP007 | TSRTP001 | IJPSM001 |
| TSRTP008 | TSRTP001 | IJPSM001 |
| TSRTP101 | TSRTP101 | IJPSM002 |
| TSRTP102 | TSRTP101 | IJPSM002 |
| TSRTP103 | TSRTP101 | IJPSM002 |
| TSRTP104 | TSRTP101 | IJPSM002 |
| TSRTP105 | TSRTP101 | IJPSM002 |
| TSRTP201 | TSRTP201 | IJPSM003 |
| TSRTP202 | TSRTP201 | IJPSM003 |
| TSRTP203 | TSRTP201 | IJPSM003 |
| TSRTP204 | TSRTP201 | IJPSM003 |
| TSRTP301 | TSRTP301 | IJPSM004 |
| TSRTP302 | TSRTP301 | IJPSM004 |
| TSRTP303 | TSRTP301 | IJPSM004 |

Compiler I/O Modules

Program Number 360N-I-476

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IJCFAOI0 | IJCFAOI0 |
| None | IJCFAOI1 | IJCFAOI1 |
| None | IJCFAOI2 | IJCFAOI2 |
| None | IJCFAOI4 | IJCFAOI4 |
| None | IJCFAOZ0 | IJCFAOZ0 |
| None | IJCFAOZ1 | IJCFAOZ1 |
| None | IJCFAOZ2 | IJCFAOZ2 |
| None | IJCFAOZ4 | IJCFAOZ4 |
| None | IJCFCCZ0 | IJCFCCZ0 |
| None | IJCFCCZ1 | IJCFCCZ1 |
| None | IJCFCCZ2 | IJCFCCZ2 |
| None | IJCFCIZ0 | IJCFCIZ0 |
| None | IJCFCIZ1 | IJCFCIZ1 |
| None | IJCFCIZ2 | IJCFCIZ2 |
| None | IJCFYOZ0 | IJCFYOZ0 |
| None | IJCFYOZ1 | IJCFYOZ1 |
| None | IJCFYOZ2 | IJCFYOZ2 |
| None | IJCFZII0 | IJCFZII0 |
| None | IJCFZII1 | IJCFZII1 |
| None | IJCFZII2 | IJCFZII2 |
| None | IJCFZII3 | IJCFZII3 |
| None | IJCFZIZ0 | IJCFZIZ0 |
| None | IJCFZIZ1 | IJCFZIZ1 |
| None | IJCFZIZ2 | IJCFZIZ2 |
| None | IJCFZIZ3 | IJCFXIZ3 |
| None | IJCFZOI1 | IJCFZOI1 |
| None | IJCFZOI2 | IJCFZOI2 |
| None | IJCFZOI4 | IJCFZOI4 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name | Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|---|---|---|
| None | IJCFZOZ1 | IJCFZOZ1 | None | IJHAIZZZ | IJHAIZZZ |
| None | IJCFZOZ2 | IJCFZOZ2 | None | IJHBABZZ | IJHBABZZ |
| None | IJCFZOZ4 | IJCFZOZ4 | None | IJHBARZZ | IJHBARZZ |
| None | IJDFAPIZ | IJDFAPIZ | None | IJHBASZZ | IJHBASZZ |
| None | IJDFAPZZ | IJDFAPZZ | None | IJHBIZZZ | IJHBIZZZ |
| None | IJDFYPZZ | IJDFYPZZ | None | IJHUABZZ | IJHUABZZ |
| None | IJDFZPIZ | IJDFZPIZ | None | IJHUARZZ | IJHUARZZ |
| None | IJDFZPZZ | IJDFZPZZ | None | IJHUASZZ | IJHUASZZ |
| None | IJFFBCZZ | IJFFBCZZ | None | IJHUIZZZ | IJHUIZZZ |
| None | IJFFZCZZ | IJFFZCZZ | None | IJHZLZZZ | IJHZLZZZ |
| None | IJFUBCZZ | IJFUBCZZ | None | IJHZRBZZ | IJHZRBZZ |
| None | IJFVBCWZ | IJFVBCWZ | None | IJHZRRZZ | IJHZRRZZ |
| None | IJFVBCZZ | IJFVBCZZ | None | IJHZRSZZ | IJHZRSZZ |
| None | IJFVZCWZ | IJFVZCWZ | None | IJIBAIZZ | IJIBAIZZ |
| None | IJFWZNZZ | IJFWZNZZ | None | IJIBAZZZ | IJIBAZZZ |
| None | IJFWZZZZ | IJFWZZZZ | None | IJIBZIZZ | IJIBZIZZ |
| None | IJGFIEZZ | IJGFIEZZ | None | IJIBZZZZ | IJIBZZZZ |
| None | IJGFIZZZ | IJGFIZZZ | None | IJIFAIZZ | IJIFAIZZ |
| None | IJGFOZZZ | IJGFOZZZ | None | IJIFAZZZ | IJIFAZZZ |
| None | IJGFUZZZ | IJGFUZZZ | None | IJIFZIZZ | IJIFZIZZ' |
| None | IJGUIEZZ | IJGUIEZZ | None | IJIFZZZZ | IJIFZZZZ |
| None | IJGUIZZZ | IJGUIZZZ | None | IJJCPV | IJJCPV |
| None | IJGUOZZZ | IJGUOZZZ | None | IJJCP0 | IJJCP0 |
| None | IJGUUZZZ | IJGUUZZZ | None | IJJCP1 | IJJCP1 |
| None | IJGVIEZZ | IJGVIEZZ | None | IJJCP2 | IJJCP2 |
| None | IJGVIZZZ | IJGVIZZZ | None | IJJCP3 | IJJCP3 |
| None | IJGVOZZZ | IJGVOZZZ | None | IJJCPDV | IJJCPDV |
| None | IJGVUZZZ | IJGVUZZZ | None | IJJCPD0 | IJJCPD0 |
| None | IJGWZNZZ | IJGWZNZZ | None | IJJCPD1 | IJJCPD1 |
| None | IJGWZRZZ | IJGWZRZZ | None | IJJCPD2 | IJJCPD2 |
| None | IJHAABZZ | IJHAABZZ | None | IJJCPD3 | IJJCPD3 |
| None | IJHAARZZ | IJHAARZZ | None | IJJCPV1 | IJJCPV1 |
| None | IJHAASZZ | IJHAASZZ | None | IJJCPV2 | IJJCPV2 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IJJCP0N | IJJCP0N |
| None | IJJCP1N | IJJCP1N |
| None | IJJCPDV1 | IJJCPDV1 |
| None | IJJCPDV2 | IJJCPDV2 |
| None | IJJCPD0N | IJJCPD0N |
| None | IJJCPD1N | IJJCPD1N |

RPG

Program Number 360N-RG-460

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IJR000 | IJR000 |
| RPG10010 | IJR010 | IJR010 |
| RPG10020 | IJR020 | IJR020 |
| RPG10025 | IJR025 | IJR025 |
| RPG10030 | IJR030 | IJR030 |
| RPG10030 | IJR039 | IJR039 |
| RPG10040 | IJR040 | IJR040 |
| RPG10040 | IJR049 | IJR049 |
| RPG10050 | IJR050 | IJR050 |
| RPT10050 | IJR059 | IJR059 |
| RPG10060 | IJR060 | IJR060 |
| RPG10060 | IJR069 | IJR069 |
| RPG10070 | IJR070 | IJR070 |
| RPG10070 | IJR079 | IJR079 |
| RPG10080 | IJR080 | IJR080 |
| RPG10080 | IJR089 | IJR089 |
| RPG10090 | IJR090 | IJR090 |
| RPG10090 | IJR099 | IJR099 |
| RPG10100 | IJR100 | IJR100 |
| RPG10100 | IJR109 | IJR109 |
| RPG10110 | IJR110 | IJR110 |
| RPG10110 | IJR119 | IJR119 |
| RPG10120 | IJR120 | IJR120 |
| RPG10120 | IJR129 | IJR129 |
| RPG10130 | IJR130 | IJR130 |
| RPG10130 | IJR139 | IJR139 |
| RPG10140 | IJR140 | IJR140 |
| RPG10140 | IJR149 | IJR149 |
| RPG10150 | IJR150 | IJR150 |
| RPG10150 | IJR159 | IJR159 |
| RPG10160 | IJR160 | IJR160 |
| RPG10160 | IJR169 | IJR169 |
| RPG10170 | IJR170 | IJR170 |
| RPG10170 | IJR179 | IJR179 |
| RPG1018A | IJR18A | IJR18A |
| RPG1018A | IJR18F | IJR18F |
| RPG10180 | IJR180 | IJR180 |
| RPG10180 | IJR189 | IJR189 |
| RPG10190 | IJR190 | IJR190 |
| RPG10190 | IJR199 | IJR199 |
| RPG10200 | IJR200 | IJR200 |
| RPG10200 | IJR209 | IJR209 |
| RPG10210 | IJR210 | IJR210 |
| RPG10210 | IJR219 | IJR219 |
| RPG10220 | IJR220 | IJR220 |
| RPG10220 | IJR229 | IJR229 |
| RPG10230 | IJR230 | IJR230 |
| RPG10230 | IJR239 | IJR239 |
| RPG10230 | IJR240 | IJR240 |
| RPG10230 | IJR241 | IJR241 |
| RPG10230 | IJR242 | IJR242 |
| RPG10230 | IJR243 | IJR243 |
| RPG10230 | IJR244 | IJR244 |
| RPG10230 | IJR245 | IJR245 |
| RPG10230 | IJR246 | IJR246 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| RPG10230 | IJR247 | IJR247 |
| RPG10230 | IJR249 | IJR249 |

Assembler

Program Number 360N-AS-465

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| ASSEMBLY | ASSEMBLY | IJQ00020 |
| ASSEM00A | ASSEMBLY | IJQ00020 |
| ASSEM00B | ASSEMBLY | IJQ00020 |
| ASSEM02 | ASSEM02 | IJQ02$20 |
| ASSEM02A | ASSEM02A | IJQ02A20 |
| ASSEM03 | ASSEM03 | IJQ03$20 |
| ASSEM03A | ASSEM03A | IJQ03A20 |
| ASSEM04 | ASSEM04 | IJQ04$20 |
| ASSEM04A | ASSEM04A | IJQ04A20 |
| ASSEM04B | ASSEM04B | IJQ04B20 |
| ASSEM05 | ASSEM05 | IJQ05$20 |
| ASSEM05A | ASSEM05A | IJQ05A20 |
| ASSEM05B | ASSEM05B | IJQ05B20 |
| ASSEM06 | ASSEM06 | IJQ06X20 |
| ASSEN07 | ASSEN07 | IJQ07020 |
| ASSEN07A | ASSEN07 | IJQ07020 |
| ASSEN07B | ASSEN07 | IJQ07020 |
| ASSEN07C | ASSEN07 | IJQ07020 |
| ASSEN07I | ASSEN07 | IJQ07120 |
| ASSEN08 | ASSEN08 | IJQ08020 |
| ASSEN08A | ASSEN08 | IJQ08020 |
| ASSEN08B | ASSEN08 | IJQ08020 |
| ASSEN08C | ASSEN08 | IJQ08020 |
| ASSEN088 | ASSEN088 | IJQ08X20 |
| ASSEN09 | ASSEN09 | IJQ09020 |
| ASSEN09I | ASSEN09I | IJQ09I20 |
| ASSEN10 | ASSEN10 | IJQ10020 |
| ASSEN10B | ASSEN10B | IJQ10B20 |
| ASSEN11A | ASSEN11A | IJQ11A20 |
| ASSEN11B | ASSEN11B | IJQ11B20 |
| ASSEN11C | ASSEN11B | IJQ11B20 |
| ASSEN11D | ASSEN11B | IJQ11B20 |
| ASSEN11E | ASSEN11B | IJQ11B20 |
| ASSEN12 | ASSEN12 | IJQ12020 |
| ASSEN13 | ASSEN13 | IJQ13020 |
| ASSEN14 | ASSEN14 | IJQ14020 |

COBOL

Program Number 360N-CB-452

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| $$BCBLOP | $$BCBLOP | None |
| $$BCBODA | $$BCBODA | None |
| $$BCBUSR | $$BCBUSR | None |
| $$BCBUSW | $$BCBUSW | None |
| None | IHD00000 | IHD00000 |
| None | IHD00100 | IHD00100 |
| None | IHD00200 | IHD00200 |
| None | IHD00300 | IHD00300 |
| None | IHD00400 | IHD00400 |
| None | IHD00500 | IHD00500 |
| None | IHD00600 | IHD00600 |
| None | IHD00700 | IHD00700 |
| None | IHD00800 | IHD00800 |
| None | IHD00900 | IHD00900 |
| None | IHD01000 | IHD01000 |
| None | IHD01100 | IHD01100 |
| None | IHD01200 | IHD01200 |
| None | IHD01300 | IHD01300 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IHD01400 | IHD01400 |
| None | IHD01500 | IHD01500 |
| None | IHD01600 | IHD01600 |
| None | IHD01700 | IHD01700 |
| None | IHD01800 | IHD01800 |
| None | IHD01900 | IHD01900 |
| None | IHD02000 | IHD02000 |
| None | IHD02100 | IHD02100 |
| None | IHD02200 | IHD02200 |
| None | IHD02300 | IHD02300 |
| None | IHD02400 | IHD02400 |
| None | IHD02500 | IHD02500 |
| None | IHD02600 | IHD02600 |
| None | IHD02700 | IHD02700 |
| None | IHD02800 | IHD02800 |
| None | IHD02900 | IHD02900 |
| None | IHD03000 | IHD03000 |
| None | IHD03100 | IHD03100 |
| None | IHD03200 | IHD03200 |
| None | IHD03300 | IHD03300 |
| None | IHD03400 | IHD03400 |
| None | IHD03500 | IHD03500 |
| None | IHD03600 | IHD03600 |
| None | IHD03700 | IHD03700 |
| COBOL | IJSCBL01 | IJSCBL01 |
| COBOL000 | IJSCBL02 | IJSCBL02 |
| COBOL000 | IJSCBL03 | IJSCBL03 |
| COBOL001 | IJSCBL04 | IJSCBL04 |
| COBOL001 | IJSCBL05 | IJSCBL05 |
| COBOL002 | IJSCBL06 | IJSCBL06 |
| COBOL003 | IJSCBL07 | IJSCBL07 |
| COBOL004 | IJSCBL08 | IJSCBL08 |
| COBOL005 | IJSCBL09 | IJSCBL09 |
| COBOL006 | IJSCBL10 | IJSCBL10 |
| COBOL007 | IJSCBL11 | IJSCBL11 |
| COBOL008 | IJSCBL12 | IJSCBL12 |
| COBOL009 | IJSCBL13 | IJSCBL13 |
| COBOL010 | IJSCBL14 | IJSCBL14 |
| COBOL011 | IJSCBL15 | IJSCBL15 |
| COBOL012 | IJSCBL16 | IJSCBL16 |
| COBOL013 | IJSCBL17 | IJSCLB17 |
| COBOL014 | IJSCBL18 | IJSCBL18 |
| COBOL015 | IJSCBL19 | IJSCBL19 |
| COBOL016 | IJSCBL20 | IJSCBL20 |
| COBOL017 | IJSCBL21 | IJSCBL21 |
| COBOL018 | IJSCBL22 | IJSCBL22 |
| COBOL019 | IJSCBL23 | IJSCBL23 |
| COBOL020 | IJSCBL24 | IJSCBL24 |
| COBOL021 | IJSCBL25 | IJSCBL25 |
| COBOL022 | IJSCBL26 | IJSCBL26 |
| COBOL023 | IJSCBL27 | IJSCBL27 |
| COBOL024 | IJSCBL28 | IJSCBL28 |
| COBOL025 | IJSCBL29 | IJSCBL29 |
| COBOL027 | IJSCBL31 | IJSCBL31 |
| COBOL028 | IJSCBL32 | IJSCBL32 |
| COBOL028 | IJSCBL33 | IJSCBL33 |
| COBOL029 | IJSCBL34 | IJSCBL34 |
| COBOL030 | IJSCBL35 | IJSCBL35 |
| COBOL031 | IJSCBL36 | IJSCBL36 |
| COBOL032 | IJSCBL37 | IJSCBL37 |
| COBOL033 | IJSCBL38 | IJSCBL38 |
| COBOL034 | IJSCBL39 | IJSCBL39 |
| COBOL035 | IJSCBL40 | IJSCBL40 |
| COBOL036 | IJSCBL41 | IJSCBL41 |
| COBOL037 | IJSCBL42 | IJSCBL42 |
| COBOL038 | IJSCBL43 | IJSCBL43 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| COBOL039 | IJSCBL44 | IJSCBL44 |
| COBOL040 | IJSCBL45 | IJSCBL45 |
| COBOL041 | IJSCBL46 | IJSCBL46 |
| COBOL042 | IJSCBL47 | IJSCBL47 |
| COBOL043 | IJSCBL48 | IJSCBL48 |
| COBOL044 | IJSCBL49 | IJSCBL49 |
| COBOL050 | IJSCBL50 | IJSCBL50 |
| DEBUG | IJSCBL60 | IJSCBL60 |

FORTRAN IV

Program Number 360N-FO-451

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IJTAAFR | IJTAAFR |
| None | IJTACOM | IJTACOM |
| None | IJTACON | IJTACON |
| None | IJTADIR | IJTADIR |
| None | IJTADXD | IJTADXD |
| None | IJTADXI | IJTADXI |
| None | IJTAIXI | IJTAIXI |
| None | IJTAPST | IJTAPST |
| None | IJTARBE | IJTARBE |
| None | IJTARXI | IJTARXI |
| None | IJTARXR | IJTARXR |
| None | IJTDVCK | IJTDVCK |
| None | IJTEXPN | IJTEXPN |
| None | IJTFDMP | IJTFDMP |
| None | IJTFIOS | IJTFIOS |
| FORTRAN | IJTFO1 | IJTFO1 |
| FORTREL | IJTFO2 | IJTFO2 |
| FORTRGE | IJTFO3 | IJTFO3 |
| FORTRPU | IJTFO4 | IJTFO4 |
| None | IJTFXIT | IJRFXIT |
| None | IJTHEXC | IJTHEXC |
| None | IJTIFIX | IJTIFIX |
| None | IJTLEXP | IJTLEXP |
| None | IJTLLOG | IJTLLOG |
| None | IJTLSCN | IJTLSCN |
| None | IJTLSQT | IJTLSQT |
| None | IJTLTAN | IJTLTAN |
| None | IJTLTNH | IJTLTNH |
| None | IJTMAXD | IJTMAXD |
| None | IJTMODI | IJTMODI |
| None | IJTMODR | IJTMODR |
| None | IJTOVRF | IJTOVRF |
| None | IJTSINT | IJTSINT |
| None | IJTSLIT | IJTSLIT |
| None | IJTSLOG | IJTSLOG |
| None | IJTSMXO | IJTSMXO |
| None | IJTSMX1 | IJTSMX1 |
| None | IJTSSCN | IJTSSCN |
| None | IJTSSQT | IJTSSQT |
| None | IJTSTAN | IJTSTAN |
| None | IJTSTNH | IJTSTNH |

Vocabulary File Utility

Program Number 360N-UT-472

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| None | IJNVBL | IJNVBL |
| None | IJNVCT | IJNVCT |
| None | IJNVER | IJNVER |
| None | IJNVIO | IJNVIO |
| None | IJNVLI | IJNVLI |
| None | IJNVLO | IJNVLO |
| None | IJNVUP | IJNVUP |

## Utilities Group 1 (Unit Record and Disk)

### Program Number 360N-UT-461

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| CDDK | IJWCD1 | IJWCD1 |
| CDDK2 | IJWGEN | IJWGEN |
| CDDK3 | IJWCD3 | IJWCD3 |
| CDDK4 | IJWCD4 | IJWCD4 |
| CDDK5 | IJWLAB | IJWLAB |
| CDPP | IJWCP1 | IJWCP1 |
| CDPP2 | IJWGEN | IJWGEN |
| CDPP3 | IJWCP3 | IJWCP3 |
| CDPP4 | IJWCP4 | IJWCP4 |
| CDPP5 | IJWLAB | IJWLAB |
| CLRDSK | IJWCLD | IJWCLD |
| CLRD2 | IJWCLD2 | IJWCLD1 |
| CLRD3 | IJWCLD3 | IJWCLD2 |
| DKCD | IJWDC1 | IJWDC1 |
| DKCD2 | IJWGEN | IJWGEN |
| DKCD3 | IJWDC3 | IJWDC3 |
| DKCD4 | IJWDC4 | IJWDC4 |
| DKCD5 | IJWLAB | IJWLAB |
| DKDK | IJWDD1 | IJWDD1 |
| DKDK2 | IJWGEN | IJWGEN |
| DKDK3 | IJWDD3 | IJWDD3 |
| DKDK4 | IJWDD4 | IJWDD4 |
| DKDK5 | IJWLAB | IJWLAB |
| DKPR | IJWDP1 | IJWDP1 |
| DKPR2 | IJWGEN | IJWGEN |
| DKPR3 | IJWDP3 | IJWDP3 |
| DKPR4 | IJWDP4 | IJWDP4 |
| DKPR5 | IJWLAB | IJWLAB |

## Utilities Group 2 (Tape)

### Program Number 360N-UT-462

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| CDTP | IJWCT1 | IJWCT1 |
| CDTP2 | IJWGEN | IJWGEN |
| CDTP3 | IJWCT3 | IJWCT3 |
| CDTP4 | IJWCT4 | IJWCT4 |
| CDTP5 | IJWLAB | IJWLAB |
| DCTP | IJWMT1 | IJWMT1 |
| DCTP2 | IJWGEN | IJWGEN |
| DKTP | IJWDT1 | IJWDT1 |
| DKTP2 | IJWGEN | IJWGEN |
| DKTP3 | IJWDT3 | IJWDT3 |
| DKTP4 | IJWDT4 | IJWDT4 |
| DKTP5 | IJWLAB | IJWLAB |
| TPCD | IJWTC1 | IJWTC1 |
| TPCD2 | IJWGEN | IJWGEN |
| TPCD3 | IJWTC3 | IJWTC3 |
| TPCD4 | IJWTC4 | IJWTC4 |
| TPCD5 | IJWLAB | IJWLAB |
| TPCP | IJWTCP | IJWTCP |
| TPCP2 | IJWTCP2 | IJWTCP2 |
| TPCP3 | IJWTCP3 | IJWTCP3 |
| TPDC | IJWTM1 | IJWTM1 |
| TPDC2 | IJWGEN | IJWGEN |
| TPDK | IJWTD1 | IJWTD1 |
| TPDK2 | IJWGEN | IJWGEN |
| TPDK3 | IJWTD3 | IJWTD3 |
| TPDK4 | IJWTD4 | IJWTD4 |
| TPDK5 | IJWLAB | IJWLAB |
| TPPR | IJWTP1 | IJWTP1 |

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| TPPR2 | IJWGEN | IJWGEN |
| TPPR3 | IJWTP3 | IJWTP3 |
| TPPR4 | IJWTP4 | IJWTP4 |
| TPPR5 | IJWLAB | IJWLAB |
| TPTP | IJWTT1 | IJWTT1 |
| TPTP2 | IJWGEN | IJWGEN |
| TPTP3 | IJWTT3 | IJWTT3 |
| TPTP4 | IJWTT4 | IJWTT4 |
| TPTP5 | IJWLAB | IJWLAB |

Utilities Group 3 (Data Cell)

Program Number 360N-UT-463

| Core Image Phase Name | Microfiche Label | Relocatable Module Name |
|---|---|---|
| CLDC | IJWCLM1 | IJWCLM1 |
| CLDC2 | IJWGEN | IJWGEN |
| CLDC3 | IJWDD3 | IJWDD3 |
| CLDC4 | IJWDD4 | IJWDD4 |
| CLDC5 | IJWLAB | IJWLAB |
| DCDC | IJWMM1 | IJWMM1 |
| DCDC2 | IJWGEN | IJWGEN |
| DCDC3 | IJWDD3 | IJWDD3 |
| DCDC4 | IJWDD4 | IJWDD4 |
| DCDC5 | IJWLAB | IJWLAB |
| DCDK | IJWMD1 | IJWMD1 |
| DCDK2 | IJWGEN | IJWGEN |
| DCDK3 | IJWDD3 | IJWDD3 |
| DCDK4 | IJWDD4 | IJWDD4 |
| DCDK5 | IJWLAB | IJWLAB |
| DCPR | IJWMP1 | IJWMP1 |
| DCPR2 | IJWGEN | IJWGEN |
| DCPR3 | IJWDD3 | IJWDD3 |
| DCPR4 | IJWDD4 | IJWDD4 |
| DCPR5 | IJWLAB | IJWLAB |
| DKDC | IJWDM1 | IJWDM1 |
| DKDC2 | IJWGEN | IJWGEN |
| DKDC3 | IJWDD3 | IJWDD3 |
| DKDC4 | IJWDD4 | IJWDD4 |
| DKDC5 | IJWLAB | IJWLAB |

active: Any loaded program ready for execution.

attention routine: System routine activated by pressing the SYSLOG request key.

background program: In multiprogramming, the program with lowest priority. Background programs execute from a stacked job input.

BG: Background program.

F1: Foreground Program One.
Highest priority user program.

F2: Foreground Program Two.
Second highest priority user program.

foreground initiation: A set of system routines to process operator commands for initiating a foreground program.

foreground program: In multiprogramming, the program with the highest priority. Foreground programs do not execute from a job stack.

inactive 1. A program not loaded in the system is inactive.

2. A loaded program not ready for execution.

MPS: Multiprogramming System.

multiprogramming system: A system that controls more than one program simultaneously by interleaving their execution.

problem program: Any program invoked by an EXEC statement. (This is a general definition. The specific definition for use with this manual is found in Section 1.)

self-relocating program: A program able to run in any area of storage by having an initialization routine to modify all address constants at object time.

SYSIN: Name used when SYSRDR and SYSIPT are assigned to the same input device by one control statement. The assignment can be either standard or temporary.

SYSOUT: Name used when SYSLST and SYSPCH are assigned to the same tape file. This can only be a standard assignment. Separate file operation is re-established by submitting a standard ASSGN for either SYSLST or SYSPCH to a unit not currently in use by the combined file. A CLOSE command may be used to perform this function.

system inquiry: The function of operator-initiated communication to a problem program.

task selection: The supervisor mechanism for determining which program should gain control of CPU processing.

INDEX

Y24-5017-2

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

## READER'S COMMENT FORM

IBM System/360
Disk Operating System
System Control

Y24-5017-2

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

| | Yes | No |
|---|---|---|
| • Does this publication meet your needs? | ☐ | ☐ |
| • Did you find the material: | | |
| Easy to read and understand? | ☐ | ☐ |
| Organized for convenient use? | ☐ | ☐ |
| Complete? | ☐ | ☐ |
| Well illustrated? | ☐ | ☐ |
| Written for your technical level? | ☐ | ☐ |

- What is your occupation? ______
- How do you use this publication?

| | | | |
|---|---|---|---|
| As an introduction to the subject? | ☐ | As an instructor in a class? | ☐ |
| For advanced knowledge of the subject? | ☐ | As a student in a class? | ☐ |
| For information about operating procedures? | ☐ | As a reference manual? | ☐ |

Other ______

- Please give specific page and line references with your comments when appropriate.

## COMMENTS:

- Thank you for your cooperation. No postage necessary if mailed in the U. S. A.

Staple

Y24-5017-2

## YOUR COMMENTS, PLEASE . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

**IBM Corporation**
**P. O. Box 6**
**Endicott, N. Y. 13760**

Attention: Programming Publications, Dept. 157

Fold Fold

Cut Along Line

IBM S/360 Printed in U. S. A. Y24-5017-2

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Additional Comments: