SY33-8557-4
File No. S370/S4300-31

Systems

DOS/VSE Librarian Logic

Program Number 5745-SC-LBR

IBM

Summary of Amendments

Edition SY33-8557-4 documents:

- Fast CORGZ
- Extended COPYSERV
- Fixed Block Architecture (FBA) Direct Access Storage Devices IBM 3310 and 3370

Technical corrections and editorial changes have also been included. Changes in contents are indicated by a vertical bar to the left of the change.

__Fifth Edition__ (February 1979)

This is a major revision of, and obsoletes, SY33-8557-3 and Technical Newsletter SN33-9214.

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

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

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

IBM® Technical Newsletter

This Newsletter No.	SN33-9266
Date	April 30, 1979
Base Publication No.	SY33-8557-4
File No.	S370/S4300-31
Previous Newsletters	None

DOS/VSE Librarian Logic

This Technical Newsletter, a part of the Disk Operating System/Virtual Storage Extended, provides replacement pages for your publication. These replacement pages remain in effect for subsequent DOS/VSE releases unless specifically altered. Pages to be replaced are:

231 - 236
239, 240

A technical change to the text or to an illustration is indicated by a vertical line to the left of the change. Editorial changes are not indicated.

Summary of Amendments

This technical newsletter documents corrections to the library format.

Note: *Please insert this page in your publication to provide a record of changes.*

IBM Laboratory, Programming Publications Department, Boeblingen, Germany

Printed in U.S.A.

PREFACE

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

System/component name	To be read as
DOS/VS	DOS/VSE (see Note below)

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

This Program Logic Manual (PLM) is a detailed guide to the IBM Disk Operating System/Virtual Storage (DOS/VS) Librarian Organization, Maintenance, and Service programs. It supplements the program listings by providing descriptive text and flowcharts.

Prerequisite and related publications that will aid in the use of this manual follow.

PREREQUISITE

- DOS/VS System Control Statements, GC33-5376.
- OS/VS, DOS/VS, and VM/370 Assembler Language Guide, GC33-4010.

RELATED

- DOS/VS System Generation, GC33-5377.
- DOS/VS Messages, GC33-5379.
- DOS/VS LIOCS Volume 1, SY33-8559.

For overall system control logic description, this Program Logic Manual is to be used with six other PLMs:

- DOS/VS Supervisor Logic, SY33-8551.
- DOS/VS Error Recovery and Recording Transients Logic, SY33-8552.
- DOS/VS Logical Transients and Dump Phases Logic, SY33-8553.
- DOS/VS System Serviceability Aids Logic, SY33-8554.
- DOS/VS Initial Program Load and Job Control Logic, SY33-8555.
- DOS/VS Linkage Editor Logic, SY33-8556.

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

Publication Organization

This manual consists of five major sections. The first is an introduction briefly discussing the functions of librarian programs.

The nexc section, Method of Operation, shows the I/O flow in Librarian Programs and describes their function, control flow and partition layout if there is more than one phase, and sequence of operation.

The next section, Program Organization, contains numbered charts describing the program flow. Some of these charts fan out in more detailed flow charts identified by letters.

Then follows the section Data Areas which shows SYSRES formats, especially the libraries for CKD and fixed block devices. The last section, Diagnostics, lists labels, phases, error messages, and internal error codes as references for debugging.

TABLE OF CONTENTS

FIGURES. 8

INTRODUCTION 9

METHOD OF OPERATION. 11
I/O Access to Librarian Files. .11
Organization Programs. 13
The COPYSERV Program, Chart 01 .13
The CORGZ Program. 15
The Maintenance Program MAINT. .19
Calling Structure of MAINT 20
Partition Layout of MAINT. 21
Sequence of Operation of Individual MAINT Phases 21
The Service Phase $LIBSTAT, Chart 47 28
The Service Phase $MAINDIR or $MAINDIF, Charts 48 to 69. 29
Service Programs 32
Functions of the Service Programs. 32
The DSERV Program, Charts 70 to 76 33
The CSERV Program, Charts 77 to 78 36
The RSERV Program, Charts 79 to 81 36
The SSERV Program, Charts 82 to 83 37
The PSERV Program, Charts 84 to 85 37
The $$BSYSWR Transient, Chart 86 38
The $$BOPNLB Transient, Chart 87 38
The DTFSL and DTFPL Macros, Charts 88 to 90. 39
The Phase $IJBLBSL, Charts 91 to 93. 42

PROGRAM ORGANIZATION 43
General Charts 43
General Charts Conventions:. 43
Chart 01. COPYSERV (Detail Chart AA). 44
Chart 02. CORGZ - Root Phase (Part 1 of 2). 45
Chart 03. CORGZ - Root Phase (Part 2 of 2). 46
Chart 04. CORGZ3 (Part 1 of 3). 47
Chart 05. CORGZ3 (Part 2 of 3). 48
Chart 06. CORGZ3 (Part 3 of 3). 49
Chart 07. CORGZ6 (Part 1 of 3). 50
Chart 08. CORGZ6 (Part 2 of 3). 51
Chart 09. CORGZ6 (Part 3 of 3). 52
Chart 10. CORGZ7 (Part 1 of 2). 53
Chart 11. CORGZ7 (Part 2 of 2). 54
Chart 12. CORGZ3F (Part 1 of 3) 55
Chart 13. CORGZ3F (Part 2 of 3) 56
Chart 14. CORGZ3F (Part 3 of 3) 57
Chart 15. CORGZ6F (Part 1 of 3) 58
Chart 16. CORGZ6F (Part 2 of 3) 59
Chart 17. CORGZ6F (Part 3 of 3) 60
Chart 18. CORGZ7F (Part 1 of 2) 61
Chart 19. CORGZ7F (Part 2 of 2) 62
Chart 20. MAINT - Root Phase (Part 1 of 4). 63
Chart 21. MAINT - Root Phase (Part 2 of 4). 64
Chart 22. MAINT - Root Phase (Part 3 of 4). 65
Chart 23. MAINT - Root Phase (Part 4 of 4). 66
Chart 24. MAINTCL (Detail Charts EX-FA) 67
Chart 25. MAINTCN (Detail Charts FD-FM) 68
Chart 26. MAINTCNF (Part 1 of 4). 69
Chart 27. MAINTCNF (Part 2 of 4). 70
Chart 28. MAINTCNF (Part 3 of 4). 71
Chart 29. MAINTCNF (Part 4 of 4). 72
Chart 30. MAINTR2 73
Chart 31. MAINTR2F. 74

Chart 32. MAINTS2 75
Chart 33. MAINTS2F. 76
Chart 34. MAINTP2 77
Chart 35. MAINTP2F. 78
Chart 36. MAINTDR (Part 1 of 2) 79
Chart 37. MAINTDR (Part 2 of 2) 80
Chart 38. MAINTDRF (Part 1 of 2). 81
Chart 39. MAINTDRF (Part 2 of 2). 82
Chart 40. MAINTA (Detail Charts MA-MQ). 83
Chart 41. MAINTAF (Part 1 of 3) 84
Chart 42. MAINTAF (Part 2 of 3) 85
Chart 43. MAINTAF (Part 3 of 3) 86
Chart 44. MAINTUP/F (Part 1 of 3) (Detail Charts NA-PG) 87
Chart 45. MAINTUP/F (Part 2 of 3) (Detail Charts NA-PG) 88
Chart 46. MAINTUP/F (Part 3 of 3) (Detail Charts NA-PG) 89
Chart 47. $LIBSTAT. 90
Chart 48. $MAINDIR (Part 1 of 10) 91
Chart 49. $MAINDIR (Part 2 of 10) 92
Chart 50. $MAINDIR (Part 3 of 10) 93
Chart 51. $MAINDIR (Part 4 of 10) 94
Chart 52. $MAINDIR (Part 5 of 10) 95
Chart 53. $MAINDIR (Part 6 of 10) 96
Chart 54. $MAINDIR (Part 7 of 10) 97
Chart 55. $MAINDIR (Part 8 of 10) 98
Chart 56. $MAINDIR (Part 9 of 10) 99
Chart 57. $MAINDIR (Part 10 of 10). 100
Chart 58. $MAINDIF (Part 1 of 12) 101
Chart 59. $MAINDIF (Part 2 of 12) 102
Chart 60. $MAINDIF (Part 3 of 12) 103
Chart 61. $MAINDIF (Part 4 of 12) 104
Chart 62. $MAINDIF (Part 5 of 12) 105
Chart 63. $MAINDIF (Part 6 of 12) 106
Chart 64. $MAINDIF (Part 7 of 12) 107
Chart 65. $MAINDIF (Part 8 of 12) 108
Chart 66. $MAINDIF (Part 9 of 12) 109
Chart 67. $MAINDIF (Part 10 of 12). 110
Chart 68. $MAINDIF (Part 11 of 12). 111
Chart 69. $MAINDIF (Part 12 of 12). 112
Chart 70. DSERV 113
Chart 71. DSERVC/DSERVF 114
Chart 72. DSERV1 (Part 1 of 2). 115
Chart 73. DSERV1 (Part 2 of 2). 116
Chart 74. DSERV2/DSERV2F. 117
Chart 75. DSERV3/DSERV4/DSERV5. 118
Chart 76. DSERV3F/DSERV6. 119
Chart 77. CSERV 120
Chart 78. CSERVC/CSERVF 121
Chart 79. RSERV (Part 1 of 2) 122
Chart 80. RSERV (Part 2 of 2) 123
Chart 81. RSERVC/RSERVF 124
Chart 82. SSERV (Part 1 of 2) 125
Chart 83. SSERV (Part 2 of 2) 126
Chart 84. PSERV (Part 1 of 2) 127
Chart 85. PSERV (Part 2 of 2) 128
Chart 86. $$BSYSWR. 129
Chart 87. $$BOPNLB. 130
Chart 88. DTFSL Macro (Part 1 of 3) 131
Chart 89. DTFSL Macro (Part 2 of 3) 132
Chart 90. DTFSL Macro (Part 3 of 3) 133
Chart 91. $IJBLBSL (Part 1 of 3). 134
Chart 92. $IJBLBSL (Part 2 of 3). 135
Chart 93. $IJBLBSL (Part 3 of 3). 136
Detail Charts. 137
Chart AA. COPYSERV 138
Chart EX. MAINTCL. 139
Chart EY. MAINTCL. 140
Chart FA. MAINTCL. 141
Chart FD. MAINTCN - Initialize for Library Condense. 142

Chart FE. MAINTCN - Initialize for Library Condense. 143
Chart FF. MAINTCN - Condense Directories 144
Chart FG. MAINTCN - Condense Directories 145
Chart FH. MAINTCN - Write Condensed Library and Directory. . . 146
Chart FJ. MAINTCN - End Routine. Initiate Load for next Condense. RPS Routine 147
Chart FK. MAINTCN - I/O and I/O Error Subroutines. 148
Chart FL. MAINTCN - Destroy IPL Record 1, Block Count and Disk Address Extraction, ... 149
Chart FM. MAINTCN - Update Subroutines 150
Chart MA. MAINTA - Process Control Card, Store Allocation Information (Part 1 of 2) 151
Chart MB. MAINTA - Store Allocation Information (Part 2 of 2). 152
Chart MC. MAINTA - Read System Directory Records and Update ALLOC (Part 1 of 2) 153
Chart MD. MAINTA - Read System Directory Records and Update ALLOC (Part 2 of 2) 154
Chart ME. MAINTA - Build Directory and Library Reallocation Tables. 155
Chart MF. MAINTA - Compute Displacement for Directory and Library Movement. 156
Chart MG. MAINTA - Update System Directory Records 1, 2, and 3 157
Chart MH. MAINTA - Write Directory Records, Blank Cylinder 0, Tracks 5-9. 158
Chart MJ. MAINTA - Update Library Directories. 159
Chart MK. MAINTA - Relocate Libraries and Directories. 160
Chart ML. MAINTA - Format Unused Tracks (Part 1 of 2). 161
Chart MM. MAINTA - Format Unused Tracks (Part 2 of 2). 162
Chart MN. MAINTA - Error Routines. 163
Chart MP. MAINTA - Subroutines 164
Chart MQ. MAINTA - Subroutines 165
Chart NA. MAINTUP - Initialize I/O Table, CHeck for Operands (Part 1 of 3) 166
Chart NB. MAINTUP - Initialize I/O Table, Check for Operands (Part 2 of 3) 167
Chart NC. MAINTUP - Initialize I/O Table, Check for Operands (Part 3 of 3) 168
Chart ND. MAINTUP - Check Temporary Update Operand 169
Chart NE. MAINTUP - Process V.M Operand (Part 1 of 2). 170
Chart NF. MAINTUP - Process V.M Operand (Part 2 of 2). 171
Chart NG. MAINTUP - Process ADD Control Statement (Part 1 of 2) 172
Chart NH. MAINTUP - Process ADD Control Statement (Part 2 of 2) 173
Chart NJ. MAINTUP - Proceee REP Control Statement (Part 1 of 2) 174
Chart NK. MAINTUP - Process REP Control Statement (Part 2 of 2) 175
Chart NL. MAINTUP - Process DEL Control Statement. 176
Chart NM. MAINTUP - Process END Control Statement (Part 1 of 3) 177
Chart NN. MAINTUP - Process END Control Statement (Part 2 of 3) 178
Chart NP. MAINTUP - Process END Control Statement (Part 3 of 3) 179
Chart NQ. MAINTUP - Subroutines (Part 1 of 6). 180
Chart NR. MAINTUP - Subroutines (Part 2 of 6). 181
Chart NS. MAINTUP - Subroutines (Part 3 of 6). 182
Chart NT. MAINTUP - Subroutines (Part 4 of 6). 183
Chart NU. MAINTUP - Subroutines (Part 5 of 6). 184
Chart NV. MAINTUP - Subroutines (Part 6 of 6). 185
Chart NW. MAINTUP - Sequence Number Check (Part 1 of 4). . . . 186
Chart NX. MAINTUP - Sequence Number Check (Part 2 of 4). . . . 187
Chart NY. MAINTUP - Sequence Number Check (Part 3 of 4). . . . 188
Chart NZ. MAINTUP - Sequence Number Chart (Part 4 of 4). . . . 189
Chart PC. MAINTUPF - Initialization. 190
Chart PD. MAINTUPF - END Statem. Routine. Update Directory and Library Descriptor. 191

Chart PE. MAINTUPF - Update Directory and Library Descriptor . 192
Chart PF. MAINTUPF - Prepare Directory Scan. 193
Chart PG. MAINTUPF - Member Block Write Routine. 194
Chart RA. DSERV1 195
Chart RB. DSERV2 196
Chart RC. DSERV2F. 197
Chart RD. DSERV3 198
Chart RE. DSERV3F. 199
Chart RF. DSERV4 and DSERV5. 200
Chart RG. DSERV6 201

DATA AREAS 202
INITABLE from MAINT Root Phase 202
Reallocation Tables from MAINTA or MAINTAF 204
Library Status Table from $LIBSTAT 206
The Stow Table and the TABIN Array (for $MAINDIR/$MAINDIF) 206
CKD Formats in STOWTAB 206
Fixed Block Formats in STOWTAB 207
The TABIN Array. 209
Switches for Various Phases. 209
Switches for $MAINDIR. 209
Switches for $MAINDIF. 211
Switches for the DSERV Program 212
SYSRES Formats 214
SYSRES Overview. 214
The System Directory 217
Libraries on CKD Devices 219
Core Image Library 219
Relocatable Library. 222
Source Statement Library 228
Procedure Library. 230
Libraries on Fixed Block Devices 231
Directory Space on a Fixed Block Device. 231
Member Space on a Fixed Block Device 238
Private Libraries. 238
General Overview of Library Record Sizes 239

DIAGNOSTICS. 240
Label List for Charts 01-93. 240
Label List for Charts AA-RG. 243
Phase to Module Cross Reference. 248
Messages Cross Reference 249
Internal Librarian Error and Return Codes. 250
Return Codes with Message 3M17I: 250
Return Codes for $MAINDIR/$MAINDIF 250

INDEX. 251

FIGURES

Figure 1. I/O Flow in Librarian Phases 12
Figure 2. Printout Produced by COPYSERV 14
Figure 3. CORGZ Program Calling Structure 15
Figure 4. CORGZ Program Partition Layout 16
Figure 5. CORGZ I/O Flow 17
Figure 6. MAINT Program Calling Structure 20
Figure 7. MAINT Program Partition Layout 21
Figure 8. Reallocation of SYSRES by MAINTA 26
Figure 9. Reallocation of SYSRES by MAINTAF 27
Figure 10. Sequence of Operation in $MAINDIR and $MAINDIF 30
Figure 11. DSERV Program Calling Structure 34
Figure 12. DSERV Program Partition Layout 34
Figure 13. INITABLE from MAINT Root Phase 202
Figure 14. MAINTA Reallocation Table 204
Figure 15. MAINTAF Reallocation Table 205
Figure 16. Library Status Table 206
Figure 17. Layout of SYSRES on a CKD Device 215
Figure 18. Layout of SYSRES on a Fixed Block Device 216
Figure 19. System Directory on a CKD Device 218
Figure 20. System Directory on a Fixed Block Device 218
Figure 21. Core Image Library Member Space on a CKD Device 221
Figure 22. Relocatable Library Member Space on a CKD Device 223
Figure 23. Module in the Relocatable Library 224
Figure 24. Format of ESD Records 225
Figure 25. Format of TXT Records 226
Figure 26. Format of RLD Records 227
Figure 27. Source Statement Library Member Space on a CKD Device 229
Figure 28. Library Descriptor for Libraries on a Fixed Block Device 232
Figure 29. Library Record Size Overview 239

The Librarian is a group of programs which serve to organize and maintain the libraries of a DOS/VS resident system, and the private libraries attached to it.
It also contains service programs to display and punch libraries or parts of them or display their directories.

Libraries can reside, at the user's discretion, on FBA or CKD devices. The I/O function for different disk storage types is done in different phases called twin phases which differ in their I/O logic. The phase names follow this naming convention:

```
    phasename  -  function performed for CKD or for both types
    phasenameF -  function performed for FBA
```

For CKD phases in the librarian, rotational position sensing is provided. This support is activated or deactivated depending on supervisor option and device.

Organization Programs

COPYSERV is fetched by job control when the // EXEC COPYSERV statement is read. The main functions of this program are to compare the directory entries of the current libraries with those of the new libraries and to prepare corresponding copy statements automatically and in sorted order for inclusion in the CORGZ job stream. The use of COPYSERV is especially advantageous for the installation of a new release of DOS/VS. COPYSERV does not support FBA.

CORGZ is fetched by job control when the // EXEC CORGZ control statement is read. Its major functions are to:

-- create a new SYSRES,
-- create private libraries,
-- copy SYSRES either selectively or completely,
-- merge from one library to another, either selectively or completely.

After completing the copy run, the CORGZ program fetches $LIBSTAT to print the system status report of the new SYSRES or of the private files.

The Maintenance Program MAINT

The MAINT program is fetched by job control when the // EXEC MAINT statement is read. The various phases of this program catalog, delete, or rename elements and update, reallocate, or condense the libraries.

Service Programs

The librarian contains serveral programs to display and punch parts or all of the different private or system libraries or to display their directories. Following is a list of these programs and their respective functions.

DSERV: display the directories of system or private libraries either unsorted (DSPLY) or sorted in alphameric sequence (DSPLYS).

CSERV: display and/or punch phases, programs, or all, of a core image library.

RSERV: display and/or punch modules, programs, or all, of a relocatable library.

SSERV: display and/or punch books, sublibraries, or all, of a source statement library.

PSERV: display and/or punch procedures or all of the procedure library.

In addition there are some auxiliary programs used by other components when dealing with the libraries: the transients $$BOPNLB and $$BSYSWR, the phase $IJBLBSL, and numerous internal macros of which only two, DTFSL and DTFPL, contain executable code.

$$BOPNLB supplies to the calling program the disk address and status of the source statement and the procedure libraries.

$$BSYSWR updates the address of the label information area and of the procedure library after a reallocation of the system, in the label area ACBs.

DTFSL/DTFPL retrieve members from the source statement and from the procedure libraries.

$IJBLBSL accesses the source statement and the procedure library when requested by programs via the two preceding macros.

I/O ACCESS TO LIBRARIAN FILES

SYSIPT,SYSLST, and SYSPCH are accessed via DTFCP, GET, PUT. Libraries are opened for input or output via DTFCP or DTFPH and accessed within the phases by their own I/O. They support RPS where appropriate, depending on supervisor option and device. Following is an overview showing which libraries are serviced by which librarian phases:

	IPT	PCH	LST	LOG	RES	CLB	RLB	SLB	SYS	SYS	SYS	SYS
									000	001	002	003
COPYSERV		OUT	OUT		IN	IN	IN	IN				IN
CORGZ	IN		OUT		IN							
CORGZ3/F					I/O		OUT	OUT	IN	IN	I/O	
CORGZ6/F					I/O	OUT					I/O	IN
CORGZ7/F					IN		OUT	OUT			OUT	OUT
MAINT	IN		OUT	OUT	I/O	I/O	I/O	I/O				
MAINTCL	IN		OUT	OUT	I/O	I/O	I/O	I/O				
MAINTCN/F			OUT	OUT	I/O	I/O	I/O	I/O				
MAINTR2/F	IN		OUT	OUT	I/O		I/O					
MAINTS2/F	IN		OUT	OUT	I/O			I/O				
MAINTP2/F	IN		OUT	OUT	I/O							
MAINTDR/F			OUT		I/O		I/O	I/O				
MAINTA/F			OUT	OUT	I/O							
MAINTUP/F	IN		OUT	OUT	I/O			I/O				
$LIBSTAT			OUT		IN	IN	IN	IN				
$MAINDIR/DIF			OUT	OUT	I/O	I/O						
DSERV	IN		OUT		IN	IN	IN	IN				
CSERV	IN	OUT	OUT		IN	IN						
RSERV	IN	OUT	OUT		IN		IN					
SSERV	IN	OUT	OUT		IN			IN				
PSERV	IN	OUT	OUT		IN							
$$BSYSWR					IN							
$$BOPNLB					IN			IN				
DTFSL												
$IJBLBSL					IN			IN				

Figure 1. I/O Flow in Librarian Phases

ORGANIZATION PROGRAMS

This section presents the copy service program (COPYSERV) and the copy program (CORGZ).

THE COPYSERV PROGRAM, CHART 01

Function of COPYSERV

COPYSERV is a one-phase program that is fetched from the CIL when the // EXEC COPYSERV control statement is read by job control.

The program compares the directory of a current library with that of a new library and produces copy statements in sorted order on SYSPCH for any current library element not yet contained on the new library. These statements can then be used as input to the CORGZ program which merges the missing elements to the new library. This can also be done collectively for all system libraries together. The following control statements are produced, as required:

```
    COPYC phasename,phasename,...
    COPYR modname,modname,...
    COPYS bookname,bookname,...
    COPYP procname,procname,...
```

COPYSERV also records the results of the comparison on SYSLST for a printed output of the COPYSERV run.

Output of COPYSERV

The printout consists of copy statements for the elements which are not in the new system pack, the number of directory entries required, and the number of library blocks needed to accommodate the programs, modules, books, or procedures that are to be merged by CORGZ. Figure 2 shows a sample COPYSERV printout.

```
 COPYR CARDS FOR MERGE TO NEW SYSRLB PACK

// EXEC CORGZ
 MERGE RES,PRV
  COPYR $$A$IPLR,$$A$PLBF,$$A$PLBK,$$A$SUPS,$$A$SUPX,$$A$SVA,$$ABERAA
  COPYR $$ABERAB,$$ABERAC,$$ABERAD,$$ABERAE,$$ABERAF,$$ABERAG
  COPYR $$ABERAH,$$ABERAI,$$ABERAN,$$ABERAO,$$ABERAQ,$$ABERA2
   .
   .
   .
   .
   .
  COPYR IPKQA,IPKRA,IPKRB,IPKRC,IPKSA,IPKSB,IPKTA,IPKVA,IPKVB
  COPYR IPKVD,IPKVE,IPKVF,IPKVG,IPKVI,IPKVK,IPKVM,PPGPRINT,XJWSARST

3A03I RELOCATABLE LIBRARY       1,793 NEW DIRECTORY ENTRIES REQUIRED,
                               17,753 NEW LIBRARY BLOCKS REQUIRED
```

Figure 2. Printout Produced by COPYSERV

Besides copy statements, COPYSERV also produces the following statements for inclusion in a CORGZ jobstream:

```
// EXEC CORGZ
MERGE xxx,yyy
[Copy statements]
/*
/&
```

Sequence of Operation of COPYSERV

The COPYSERV program:

- Opens SYSLST and SYSPCH and the libraries involved in the comparison,
- Prints and punches the CORGZ and MERGE statements, the COPY(x) statements for the libraries involved,
- Compares private libraries, if there are any, in the same way,
- Closes all files.

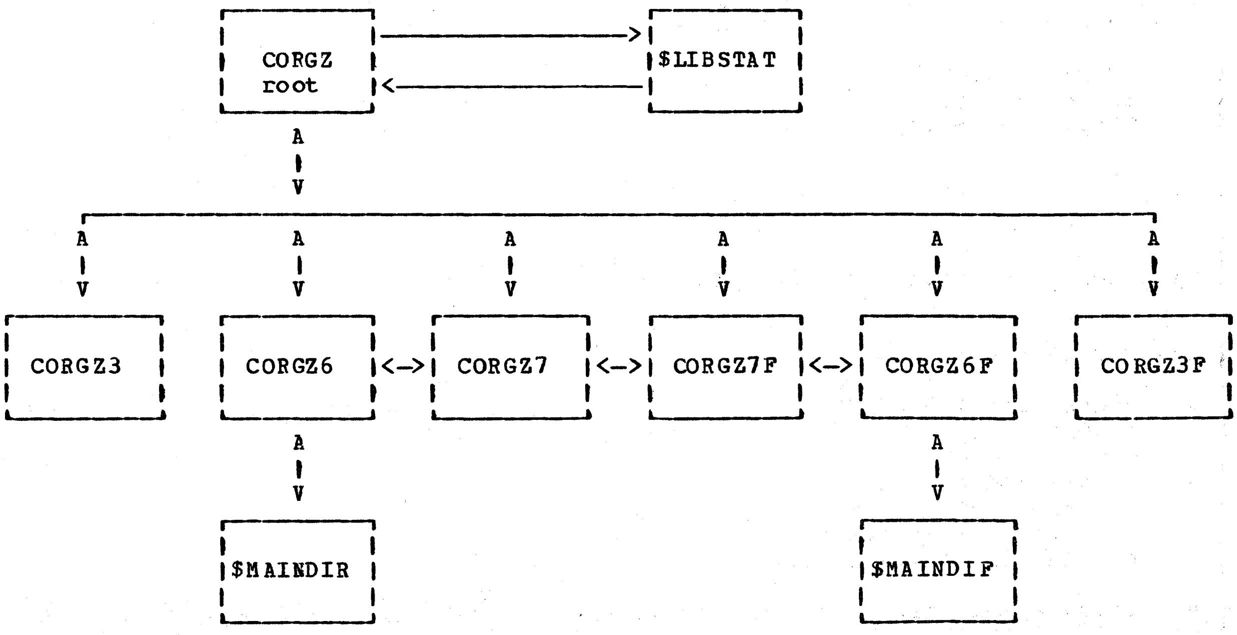
THE CORGZ PROGRAM

Function of CORGZ

The copy program selectively or completely copies the system residence onto another disk pack and can define the limits for the new disk pack (allocation). It also creates private core image, relocatable, and source statement libraries and can merge from one library to another either selectively or completely. All $ phases of the core image library, the partition standard (PARSTD), and the standard label **(STDLABEL) tracks of the label area, are first copied** automatically on an ALLOC statement.

Calling Structure of CORGZ

The CORGZ program consists of seven phases. Figure 3 shows the calling structure of those phases.



Note: There is communication between the phases CORGZ7 and CORGZ7F because these phases process the NEWVOL command which can have output to a device type different from the type of SYSRES. An FBA SYSRES can have output to a CKD disk and vice versa.

Figure 3. CORGZ Program Calling Structure

Partition Layout of CORGZ

CORGZ, the root phase, contains tables and switches necessary to the interface between its related processing phases. The partition layout for CORGZ phases is illustrated in Figure 4.

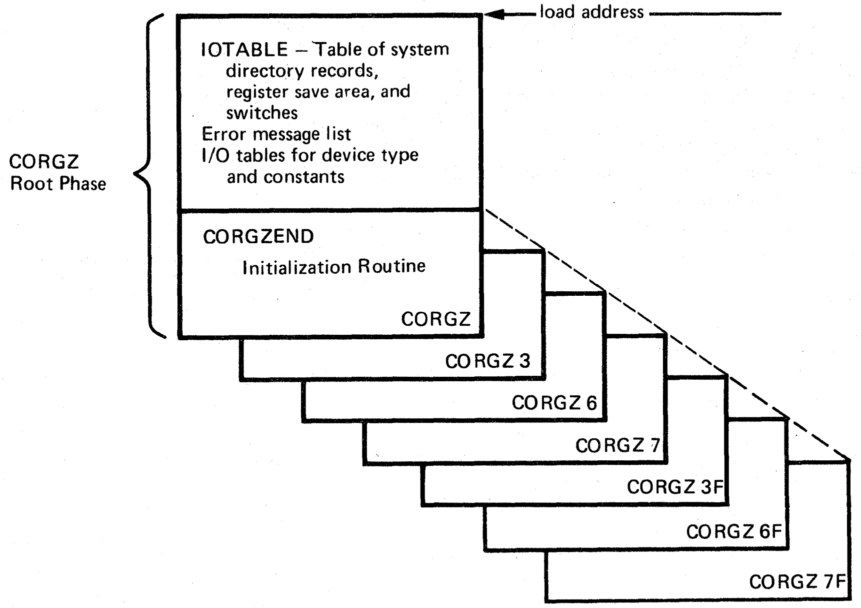


Figure 4. CORGZ Program Partition Layout

I/O Flow of CORGZ

Figure 5 shows the I/O flow for all CORGZ phases.

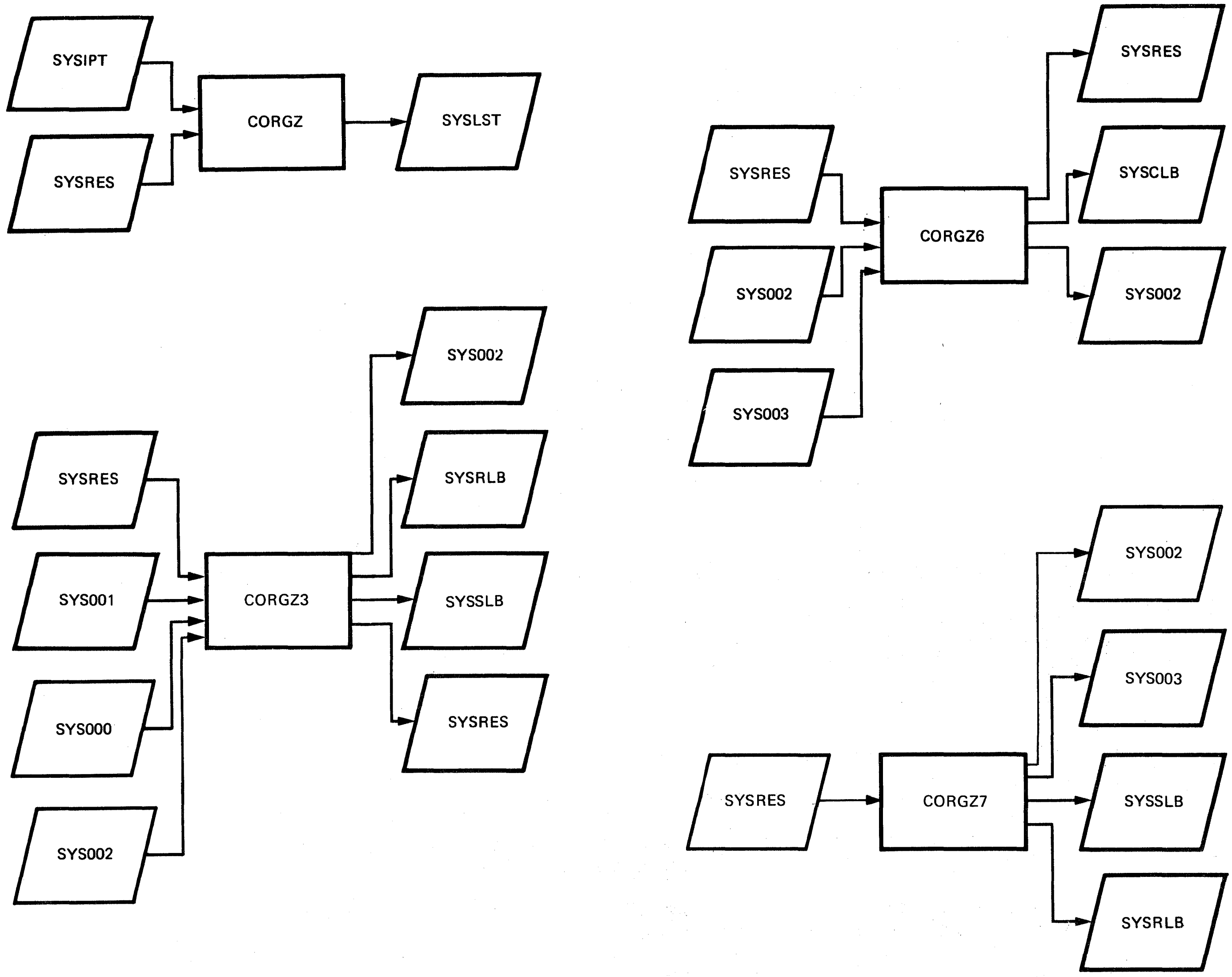


Figure 5. CORGZ I/O Flow

CCW Chaining Algorithm in the CORGZ Program

The following algorithm is used for reading and writing members of libraries on CKD devices.

Three buffers are used to permit parallel reading and writing operations. If sufficient space is availble, the buffer size used is equal to one track. If the 'FROM' and 'TO' devices have different track capacities, the buffer size is equal to the capacity of the larger track. While data is being read into one of the buffers, other data is being written out of a previous buffer. The process wraps around from the third to the first buffer.

CCW chains are built to transfer as much data as will fit into one full track. This is equivalent to one full buffer except when either the 'FROM' or 'TO' device has a smaller track capacity than the other device. In that case, the CCW chain for the smaller device will transfer less data than is required for one full buffer. Subsequent CCW chains will also transfer one full track of data although the data may occupy space in the adjoining buffer. The adjoining buffer may be the first buffer if the CCW chain began in the third buffer.

If the members being read are adjacent to one another on the same track, a single chain will transfer the complete track. Otherwise, separate chains are built for each member. This may happen either by previous deletion of individual members or in a selective copy run.

Sequence of Operation in the CORGZ Program

ROOT PHASE CORGZ, CHARTS 2 TO 3: The // EXEC CORGZ job control statement loads and executes the root phase CORGZ. The prime functions performed by this phase are:

- Open SYSIPT and SYSLST.
- Open requested files.
- Initialize tables to reflect the device type.
- Read system directory records for starting addresses of the core image, relocatable, source statement, and procedure directories.
- Read the library descriptor records (first record of a directory) for library and directory information.
- Read and analyze control statements and fetch the appropriate phases.
- Give a status table to $LIBSTAT to have the required status report printed.

PHASE CORGZ3/F, CHARTS 4 TO 6, 12 TO 14: CORGZ3 and CORGZ3F process COPYR,COPYS,COPYP, and COPYI statements as follows:

- Process the operands on the copy statement for the library concerned.
- Set up tables for correct library and directory copy.
- Set proper status report switch.
- Copy the desired elements from the library concerned.

PHASE CORGZ6/F, CHARTS 7 TO 9, 15 TO 17: CORGZ6 and CORGZ6F process the COPYC statement as follows:

- Set up TO and FROM file operands and check that the private core image library is not otherwise assigned
- Sort and copy all $-phases, after an ALLOC statement is processed
- Copy or merge the phases as requested in the copy statements
- Fetch $MAINDIR to update the 'TO' file directory
- Set the proper status report switches

PHASE CORGZ7/F, CHARTS 10 TO 11, 18 TO 19: CORGZ7 or CORGZ7F perform the following functions.

If creating a new system residence file:

- Process operands on ALLOC statement
- Format the core image library and all directories
- Build tracks 0 and 1 for SYS002 and the label area

If creating a private library:

- Process operands on NEWVOL statements
- Format new private library directories
- Generate system directory records at the beginning of private library directories

THE MAINTENANCE PROGRAM MAINT

Functions of the MAINT Program

The functions of the MAINT program are as follows:

- Condense function for all libraries
- Condense limit setting for all libraries
- Catalog function for all libraries (core image library via $MAINDIR/F called by $LINKEDT)
- Rename and Delete functions for all libraries
- Reallocate function for all system libraries
- Update statements in the source statement library

The phases of the MAINT program are presented in the following order:

MAINT	-	root phase
MAINTCL	-	set condense limits
MAINTCN/F	-	condense libraries
MAINTR2/F	-	catalog relocatable library
MAINTS2/F	-	catalog source statement library
MAINTP2/F	-	catalog procedure library
MAINTDR/F	-	rename or delete any library
MAINTA/F	-	re-allocate libraries
MAINTUP/F	-	source statement library single statement update
$LIBSTAT	-	print status report
$MAINDIR/DIF	-	maintain core image directory and SDLs. See the description of that phase for details.

For the relationship of twin phases (with or without final F) see the Introduction of this manual.

CALLING STRUCTURE OF MAINT

The program has the following calling structure:

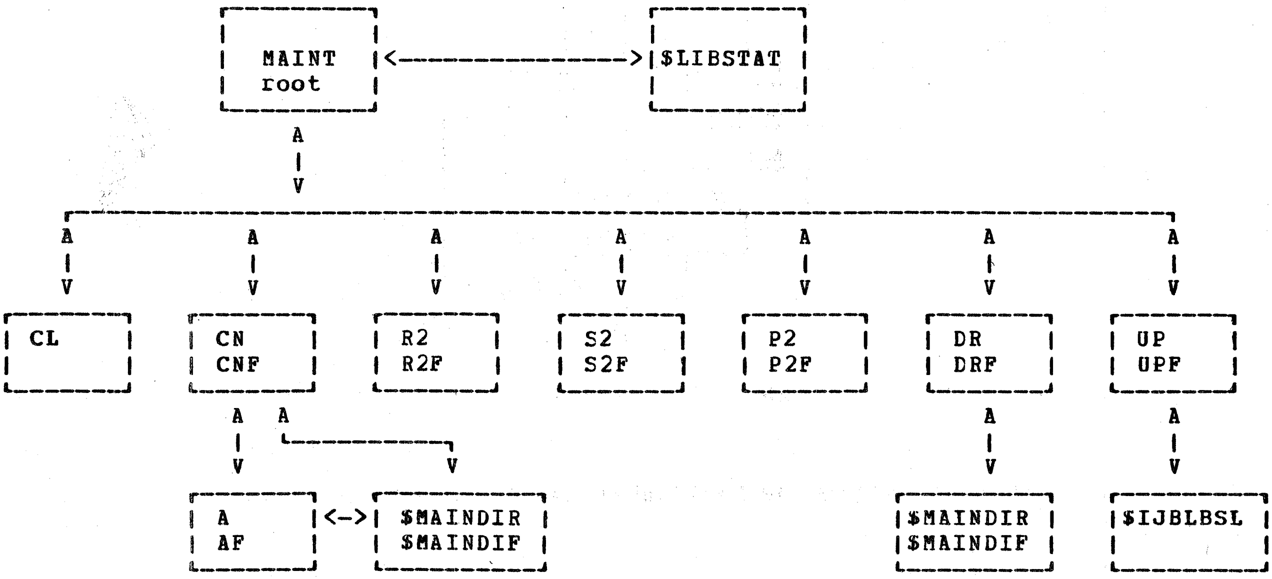


Figure 6. MAINT Program Calling Structure

PARTITION LAYOUT OF MAINT

Figure 7 shows the partition layout of the MAINT program where the root phase stays in core together with varying processing phases. $LIBSTAT and $MAINDIR/$MAINDIF as well as $IJBLBSL are located in the SVA and do not appear in the partition.

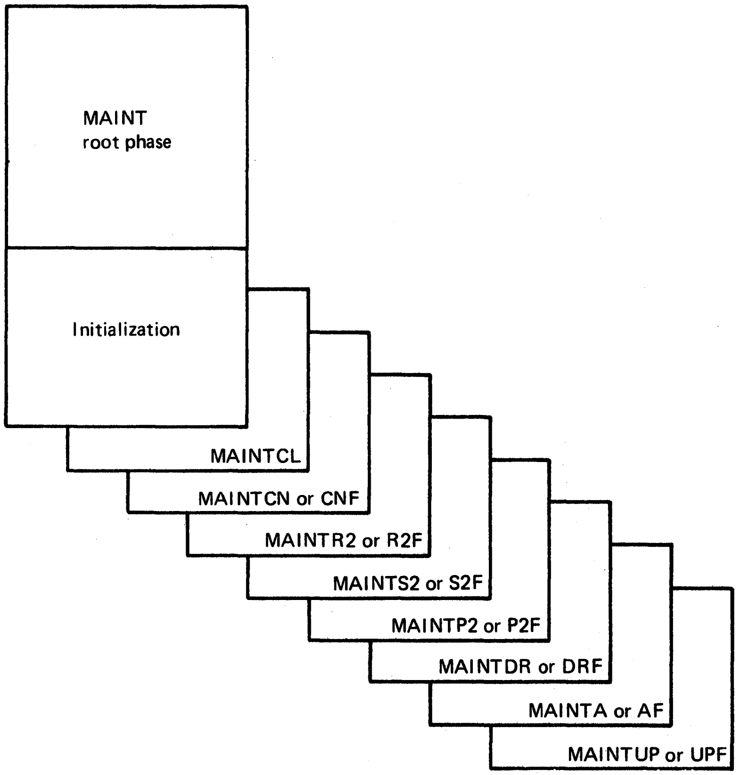


Figure 7. MAINT Program Partition Layout

SEQUENCE OF OPERATION OF INDIVIDUAL MAINT PHASES

MAINT Root Phase, Charts 20-23

The root phase consists of 9 modules:

1. IJBMIN Initialization (overlaid)
2. IJJCPD1N - LIOCS I/O module
3. IJBMUP - Update disk address
4. IJBMIO - Disk I/O
5. IJBMCS - Control statement read and scan
6. IJBLBC - Librarian error routine
7. IJBLBA - Analyze control statement and fetch processing phase
8. IJBMDU - Update directory
9. IJBMDS - Scan directory

In these it performs the following:

- Opens SYSIPT and SYSLST and the private libraries if there are any assigned.
- Determines the device type of the libraries and sets up INITABLE*.
- Reads the control statement.
- Displays the control statement on SYSLST.
- Analyzes the operation field of the control statement.
- Fetches or branches to required processing phases.

While the processing phases are operating, they make use of the following services located in the root phase:

- Scan directories (CKD phases only)
- Write error messages
- Perform disk I/O (CKD phases only)
- Update library descriptors and directory records. (CKD phases only)
- Update disk addresses for directory and member read and write. (CKD phases only)

After the processing phases have finished, the root phase:

- Branches to $LIBSTAT to have the status of updated libraries printed
- Gives EOJ

* The initialization table is specified in the module IJBLBA and contains device characteristics of the libraries which are filled in by the module IJBMIN. Other modules of the root-phase MAINT and all other phases of the MAINT program access this table. Figure 13 in the Data Areas section shows its format and contents.

Phase MAINTCL, Chart 24

The control statement which causes MAINTCL to be called is

```
    CONDL  CL=n,RL=n,SL=n,PL=n
```

```
where: n=5 digits for CKD
         9 digits for FBA
```

MAINTCL sets condense limits for all or any of the libraries in the respective library descriptors. If condense limits already exist from a previous CONDL statement these condense limits are changed to the limits specified in the new CONDL statement. The set condense limits function is performed for system and private libraries. MAINTCL returns control to the root phase at CSSTART after all operands of the control statement are processed or after certain error conditions have been met.

Phase MAINTCN or MAINTCNF, Charts 25-29

The phase is fetched by the root phase when a control statement requesting a condense or reallocation function is read. The phase condenses any or all of the libraries and their directories. The phase:

- Scans for library operands
- Initializes for the requested library condense
- Condenses the directory and member space
- Updates the library descriptor

Exits from MAINTCN or MAINTCNF are:

- Fetch MAINTA if an ALLOC was read,
- Return to MAINT to read the next statement if a CONDS statement was read.

During a condense or reallocation action all other access to the libraries has to be stopped. Therefore MAINTCN/F and MAINTA/F call PIOCS to:

- Mask attention if bit 6 of the linkage control byte (displacement 57 of the communications region) is on. This bit is turned on and off by both programs for the system CIL. For a private CIL, bit 5 of the job duration byte is set during condense.
- Enter the system into a 'hard wait' when an I/O error occurs on SYSRES, or when updating the core image library. The indication X'FF' is then set in register 11 and is stored in low real storage.

Phase MAINTR2 or MAINTR2F, Charts 30 to 31

MAINTR2 or MAINTR2F catalog modules to the relocatable library on SYSRES or SYSRLB as determined from INITABLE in the root phase.

To do this the phase:

- Finds out if the relocatable library is allocated
- Deletes possible duplicate modules in the library (for CKD only)
- Reads statements from SYSIPT
- Analyzes them for type
- Builds the respective records for each type
- Checks for more statements from SYSIPT
- Completes cataloging on finding the END statement
- Deletes possible duplicate modules in the library (for FBA only)
- Updates the library directory
- Returns to CSSTART in the root phase

Phase MAINTS2 or MAINTS2F, Charts 32 to 33

MAINTS2 or MAINRS2F catalog books to the source statement library on SYSRES or SYSRLB as determined from INITABLE in the root phase. To do this the phase:

- Finds out if the library is allocated and if library and directory are not full
- Scans the bookname and deletes possible duplicates in the library (only for CKD)
- Updates the library descriptor record
- Reads, compresses, and catalogs the book
- Deletes possible duplicates in the library (only for FBA)
- Updates the library directory
- Returns to either EOF or to read another statement.

Phase MAINTP2 or MAINTP2F, Charts 34 to 35

The phases catalog procedures into the procedure library when the root phase encounters a CATALP statement. The phase MAINTP2 does the following:

1. If the procedure library is allocated and not full,
2. scans the control statement
3. Looks for the procedure name in the directory
4. Deletes the directory entry (if there is one already)
5. If there is enough space in the library,
6. catalogs the procedure
7. Updates the directory when EOP is detected
8. Returns to CSSTART in the root phase.

For MAINTP2F, the sequence of steps is 1, 2, 5, 6, 3, 4, 7, 8.

Phase MAINTDR or MAINTDRF, Charts 36 to 39

The phase deletes and renames from the relocatable, source statement, and procedure library directories as determined from INITABLE in the root phase. For a core image library, the phase creates a Stow Table for updating the directory via phase $MAINDIR or $MAINDIF. For the format of the Stow Table see the section Data Areas.

Individual phases, modules, books, or procedures can be renamed in the core image, relocatable, source statement, or procedure libraries. The directories are always updated after a rename request. If, on the rename function, the new name is already in the directory or the old name is not in the directory, an error message is issued to SYSLST. On a valid pair of operands, the new name simply replaces the old name in the directory. In either case, a check is then made for more operands on the control statement. If there is another operand, processing continues in this phase. If there are no more operands, the program branches to CSSTART in the MAINT root phase to read another control statement.

Phase MAINTA or MAINTAF, Charts 40-43

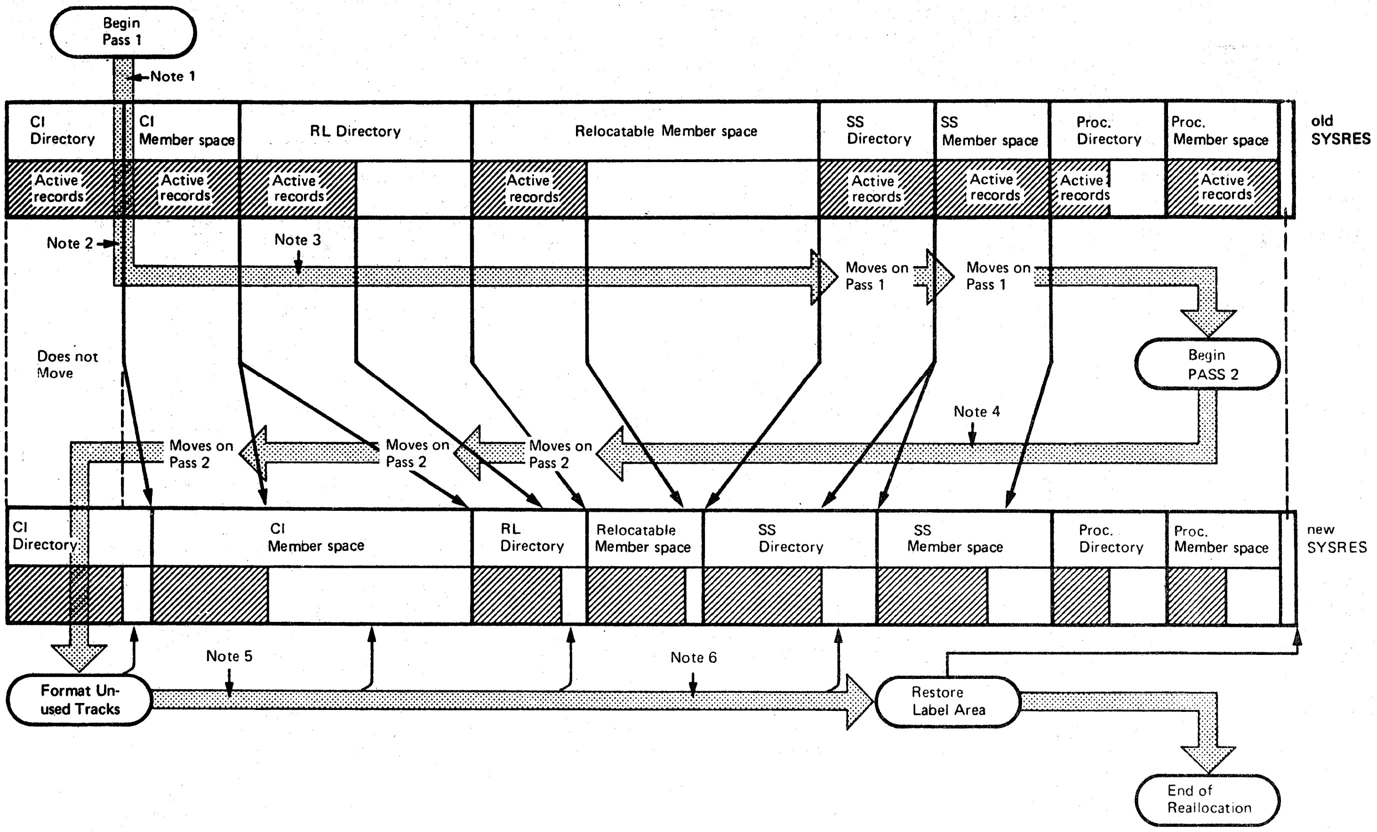
When a control statement requesting system reallocation is read by the root phase, MAINTCN or MAINTCNF is fetched to condense all system libraries before reallocation. Then SYSRES is reallocated by redefining the sizes of the libraries and their directories.

MAINTA uses the values specified in the ALLOC statement and subtracts the directory size specification from the space allocated to the library. This differs from the ALLOC statement used with CORGZ where the status report reflects the total directory space allocated to the library and the directory.

The phase:

- Builds the reallocation table
- Updates all directories
- Moves all libraries and the label information area
- Updates the label area control blocks in the SVA (via $$BSYSWR)
- Returns to module IJBMCS in the root phase.

For the reallocation tables, see the section Data Areas. Figures 8 and 9 show the method used to reallocate SYSRES.



In this example, reallocation is accomplished within the disk storage area allocated to SYSRES.

Figure 8. Reallocation of SYSRES by MAINTA

<u>Notes to Figure 8</u>:

1. Pass 1 is a forward scan of the directories and libraries beginning with the core image library.

2. The core image directory will never be moved from its predetermined starting disk address (Cyl 1, track 0) by MAINTA.

3. On pass 1, all libraries and directories that must be moved to a lower disk address are moved. Only active blocks are moved.

4. On pass 2, all libraries and directories to be moved to a higher disk address are moved. Only active blocks are moved.

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.

6. The relocatable library, the source statement library, and the procedure library are not formatted.

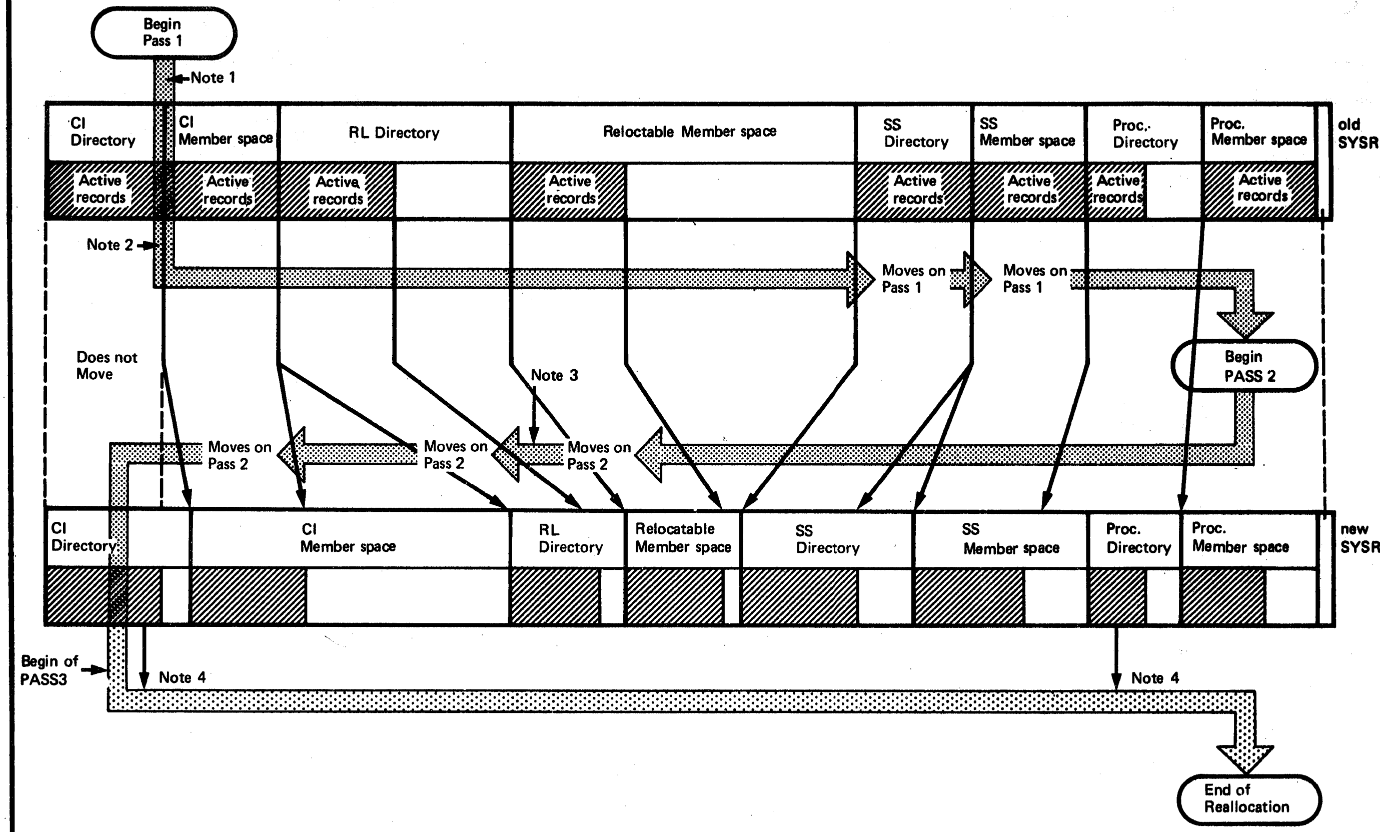


Figure 9. Reallocation of SYSRES by MAINTAF

Notes to Figure 9:

1. Pass 1 is a forward scan of the reallocation table, from lower disk addresses to higher disk addresses, beginning with the slot in the table for the core image directory. Any active blocks in the directories, or member space, or label area which must be moved to a lower disk address are moved.

2. The core image directory will never be moved from its predetermined starting disk address at block 10.

3. Pass 2 is a backward scan of the reallocation table, from higher disk addresses to lower disk addresses, beginning with the slot in the table for the label area. Any active blocks in the label area, or member spaces, or directories which must be moved to a higher disk address are moved.

4. Pass 3 is a scan of the reallocation table for newly created directories and for those directories whose member spaces have moved. Only directories are handled in this pass. No action is taken for member spaces or the label area.

Phase MAINTUP or MAINTUPF, Charts 44 to 46

The phase is fetched by MAINT root phase when an UPDATE statement is read. It adds statements to, deletes statements from, or replaces statements in books in a source statement library.

The phase:

- Checks if the source statement library is allocated, has entries, and is not full.
- Finds the book name in the source directory, first private and then system.
- Determines the kind of update requested
- Processes the subcontrol statements)ADD,)REP,)DEL,)END
- Returns to the MAINT root phase either after an error unsuccessfully, or after having updated the directory successfully.

THE SERVICE PHASE $LIBSTAT, CHART 47

Function of $LIBSTAT

The phase is called by the librarian programs and by the linkage editor to display the status of one, some, or all libraries. The SVA status can also be displayed. The calling program indicates which libraries should be displayed by means of the status table shown in the Data Areas section.

Sequence of Operation of $LIBSTAT

When $LIBSTAT is called the following parameters are transmitted:

```
Register 0 - address of status table
         1 - entry point of $LIBSTAT
        14 - return address
        13 - address of work area in user partition
```

All registers are saved and restored. $LIBSTAT is resident in the SVA. The user save area, workareas for variables and buffers are in the partition.

For $LIBSTAT, CKD and FBA code is in one phase.

The phase:

- Initializes control blocks for CKD or FBA
- Prints the header
- Reads the volume 1 label of the disk pack on which the library resides to obtain the volume serial number. All other data required for the status report is contained in the library descriptor record.
- Uses the logical unit and the disk address from the status table entry to read the library descriptor record of the indicated library.
- Builds and prints: Status line
  Directory line
  Library line
- Prints SVASTAT
- Returns to the caller

THE SERVICE PHASE $MAINDIR OR $MAINDIF. CHARTS 48 TO 69

Function of $MAINDIR/$MAINDIF

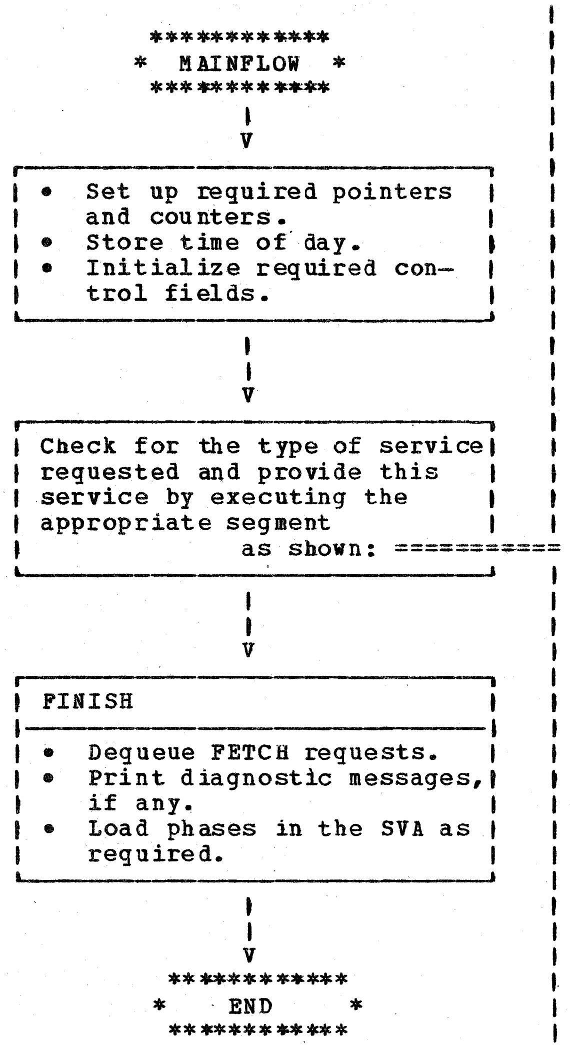
The phase is used to service core image libraries, their directories, and the system directory lists. Both versions of the program provide one, or any combination, of the following services:

- Build system directory list (SDL)
- Build second level directory (SLD) for FBA devices
- Re-initialize the library descriptor record and the SLD when a private core image library is deleted
- Build or update a core image directory or delete entries from it
- Update a core image library descriptor record
- Update the RAS load list in the supervisor.

The phase is used by librarian phases, IPL, job control, and the linkage editor.

Figure 10 gives an overview of the program structure for both phases.

Sequence of Operation of $MAINDIR/$MAINDIF



MAINFLOW
• Set up required pointers and counters.
• Store time of day.
• Initialize required control fields.
Check for the type of service requested and provide this service by executing the appropriate segment as shown:
FINISH
• Dequeue FETCH requests.
• Print diagnostic messages, if any.
• Load phases in the SVA as required.
END

The program executes with storage protection key 0.

Request code	SDLSVA
I, B, M, N	Build and update the SDL. Service is provided for IPL when SET SDL is issued or for job control when a phase is to be loaded in the SVA.
E	SCLVLDR ($MAINDIF only) Build a second level directory. Service is provided for // ASSGN SYSCLB to job control.
D	DELETALL ($MAINDIF only) Delete a complete PCIL. Service is provided for // EXEC MAINT DELETC ALL

C	MAINT
	Update the core image directory and, as necessary, the SLD, SDL, and RAS load list.
	Service is provided for
	// OPTION CATAL
	// EXEC LNKEDT
	or
	// EXEC MAINT

D	with DELETC ...
R	or with RENAMC ...
U	or with CONDS CL or // EXEC CORGZ
C	COPYC
X	REALLOC Update disk addresses in the core image directory, SDL, and RAS load list. Service is provided for // EXEC MAINT ALLOC CL= ...

L	OPTLINK
	Write one or more entries in last directory record and insert this record's address in the FETCH table.
	Service is provided for
	// OPTION LINK
	// EXEC LNKEDT

F	LCCALL Update the library descriptor on last call of $MAINDIF for a library condense request.

Figure 10. Sequence of Operation in $MAINDIR/$MAINDIF

Input/Output Operations

For phase $MAINDIR, reading from and writing to disk is done in procedures GETINP and PUTOUTP; for phase $MAINDIF, this is done in procedures RDDIRREC and WRDIRREC. The program uses the EXCP level interface to the physical I/O routines of the supervisor.

ACCESS TO INPUT/OUTPUT AREAS IN $MAINDIR. In order to read the descriptor record and the directory entries of a CIL from disk, the procedure GETINP uses the pointer:

INPTR giving the address of the start of the input area.

The size of the input area used is 17 blocks.

In order to write the descriptor record and the directory entries of a CIL onto disk, the procedure PUTOUTP uses the pointer:

OUTPTR giving the address of the block to be written.

Linkage:
Register 0 points to STOWTAB.*
Register 1 contains entrypoint.
Register 13 points to an area used for:
1. User save area (72 bytes)
2. Pseudo 'automatic' storage. Size about 7 K
3. Array 'TABIN'. max.size 1 K.
(These areas have to be aligned on doubleword)
Register 14 is return register.
Register 15 contains the return code.
See Diagnostics section.

ACCESS TO INPUT/OUTPUT AREAS IN $MAINDIF. For reading from disk, procedure RDDIRREC uses a pointer and control fields as follows:

RDBUFADR pointer to the read-buffer area.
RDBUFLEN length of the read-buffer area.
RDBLKNR relative block number of first record that is to be read from disk.

For writing onto disk, the corresponding pointer and control fields used by procedure WRDIRREC are:

WRBUFADR pointer to the write-buffer area.
WRBUFLEN length of the write-buffer area.
WRBLKNR relative block number of first record that is to be written onto disk.

Linkage:
Register 0 points to STOWTAB.*
Register 1 contains entrypoint.
Register 13 points to an area used for:
(max. total size about 24 K)
1. User save area (72 bytes)
2. Pseudo 'automatic' storage. (4 K)
3. Buffer Space (16 K)
4. Stow Table extract 'TABIN'.
Register 14 is return register.
Register 15 contains the return code.
See Diagnostics section.

* The Stow Table which is built by the calling routine contains all the information the phase needs to identify and execute the requested function.

The Stow Table has the following structure:

```
┌────────────────────┐
| Header (12 bytes)  |
|----------------------------------
|    entry (18 to 30 bytes)
|----------------------------------
|    entry
└----------------------------------
'        '
'        '
'        '
```

The header has the label STOWREG for CKD and OWPHDR for FBA. For formats of header and entry see the Data Areas section.

The Stow Table mentioned above is not suitable for the checking of entries for proper sequence by phase names and processing them in sequence. The program, therefore, builds and uses an array TABIN for this pupose. The section Data Areas shows the layout of a TABIN entry. The array contains one such entry for each entry in the Stow Table.

Other areas referenced or accessed by the phase are:

- The FETCH table
- The SDL and SLD
- The RAS load list, which is located at the beginning of the RAS monitor table.
- Core image library directory entry and descriptor record.

SERVICE PROGRAMS

FUNCTIONS OF THE SERVICE PROGRAMS

This section contains the programs which display and punch DOS/VS libraries and their directories. Some auxiliary transients and macros are also described.

These programs are presented in the following order:

- DSERV program: displays on SYSLST the contents of the directories of any or all libraries on SYSRES and the private directories as requested.

- CSERV program: displays and/or punches phases from the system and/or private core image libraries.

- RSERV program: displays and/or punches modules from system and/or private relocatable libraries.

- SSERV program: displays and/or punches blocks from the system and/or private source statement libraries.

- PSERV program: display and/or punches procedures from the procedure library.

- $$BSYSWR transient: allows the calling phase to write on SYSRES. Updates the label area control block in the SVA.

- $$BOPNLB transient: opens the source statement and the procedure libraries.

- DTFSL and DTFPL macros: allow programs like Assembler or COBOL to access a source statement library or procedure library.

- Phase $IJBLBSL: Allows access to source statement and procedure libraries. It is called via the DTFSL or DTFPL macros.

THE DSERV PROGRAM, CHARTS 70 TO 76

Calling Structure of DSERV

The program consists of the following phases:

DSERV -- root phase

DSERVC
DSERVF -- contain device specific subroutines used by the processing phases

DSERV1 -- analyzes control statements for library and function operands

DSERV2/F -- prints a core image directory

DSERV3 -- sorts relocatable, source, and procedure directories for CKD devices

DSERV4 -- prints relocatable and source directories for CKD devices

DSERV5 -- prints procedure directory for CKD devices

DSERV3F -- sorts and prints relocatable, source, and procedure directories for fixed block devices

DSERV6 -- prints the SDL

The calling structure between these phases is shown in Figure 11:

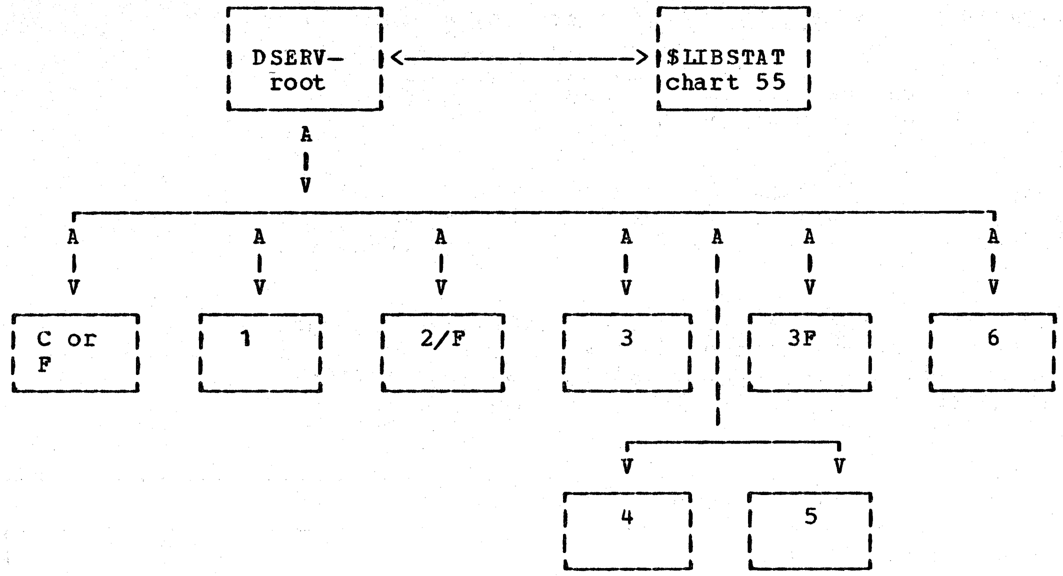


Figure 11. DSERV Program Calling Structure

Partition Layout of DSERV

The partition layout for the DSERV program is shown in Figure 12:

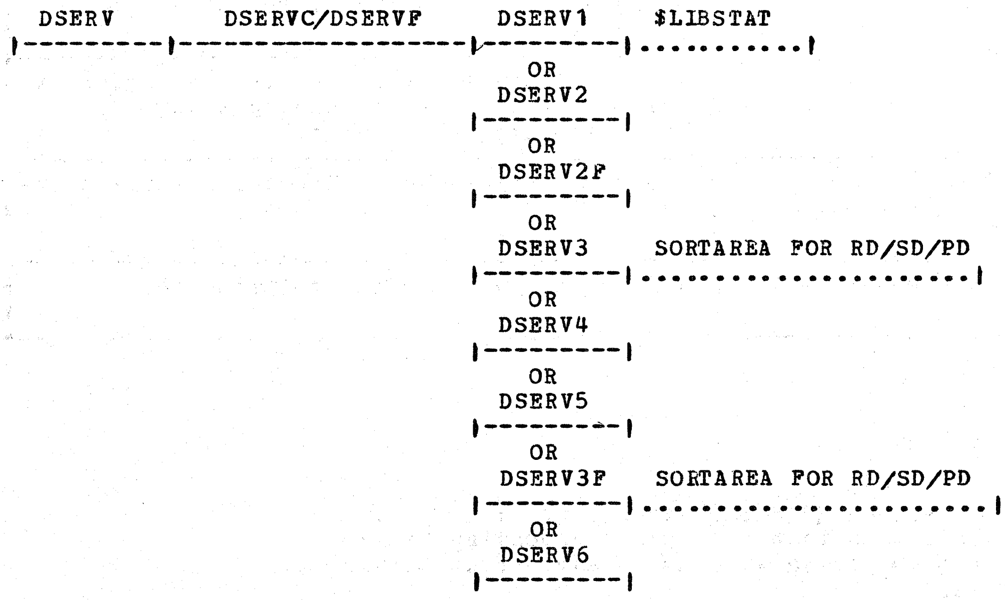


Figure 12. DSERV Program Partition Layout

Sequence of Operation of the DSERV Program

ROOT PHASE DSERV WITH DSERVC AND DSERVF: DSERV functions as root phase.

- It initializes areas, control fields, and pointers.
- Determines the load point for overlay phases.
- Opens SYSLST and SYSIN.
- Determines library device types.
- For a CKD device, loads DSERVC.
- For an FBA device, loads DSERVF.
- Fetches appropriate overlay phase (DSERV1 on first time through).

PHASE DSERV1

- Fetches $LIBSTAT to build the status table on first time through.
- Reads and analyzes the current control statement.
- Fetches a processing phase which builds and prints the directory output according to the following list:

DSERV phases producing directory list output:

	Phase name if library is on		
Operand	CKD device	FBA device	Summary of phase output
CD	DSERV2	DSERV2F	A list of phases cataloged in the pertinent core image library.
PD	DSERV3 and DSERV5	DSERV3F	A list of procedures cataloged in the procedure library, sorted if so requested.
RD	DSERV3 and DSERV4		A list of modules cataloged in the pertinent relocatable library, sorted if so requested.
SD	DSERV3 and DSERV4		A list of books cataloged in the pertinent source statement library, sorted if so requested.
TD	DSERV2	DSERV2F	A list of phases cataloged in the pertinent core image library and having a name starting with character $.
SDL	DSERV6 (For both CKD and FBA devices)		A list of phases whose directory entries are included in the SDL.

The above named phases use subroutine PRINT in root phase DSERV to print the directory list output.

I/O Flow of the DSERV Program

If a private library is assigned (SYSCLB, SYSRLB, or SYSSLB), its directory is printed rather than that of the corresponding library from SYSRES. If there are no active entries in any private library, an error message is printed.

Besides the control statements, which the program reads from SYSIPT (or SYSIN), the program uses as input the directories of the libraries for which a display has been requested.

The program builds its output records, including messages, in area PRINTB, using symbolic addressing. When a line is complete, the program passes the address of the area to logical IOCS (DTFCP) for printing that line on SYSLST.

The directory list output is always preceded by a library status report. The program retrieves the required information from the directories of the various libraries and uses this information to build the status table at area STATTAB and to print the report based on the information contained in that table.

THE CSERV PROGRAM, CHARTS 77 TO 78

Sequence of Operation

The CSERV program consists of the phases:

CSERV -- root phase which
- reads control statements
- calls processing phases
- prints and punches
- terminates the program.

CSERVC -- which do the device dependent reading from the libraries
or into storage from either CKD or FBA devices.
CSERVF

The two phases reside together in the partition.

I/O Flow of the CSERV Program

Control statements invoking the CSERV program are read from SYSIPT. CSERV will refer to the private relocatable library first and if the phase is not found there or SYSCLB is not assigned it will use the system core image library. Error messages and displayed phases are written to SYSLST. If punched output is desired a /* is provided.

THE RSERV PROGRAM, CHARTS 79 TO 81

Sequence of Operation

The RSERV program consists of two phases, the root phase RSERV and RSERVC or RSERVF whose distribution of function is the same as for the CSERV program (see above).

RSERV must analyze each record in the module as to type and convert the 160-byte records to smaller records by dividing the information in the variable length 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.

I/O Flow of the RSERV Program

Control statements invoking the RSERV program are read from SYSIPT. The search order of the libraries is the same as for CSERV, that is, first the private and then the system library.

All punched output is ejected into stacker 2. If SYSRDR and SYSPCH are assigned to the same device, a /* statement is not punched. The last RSERV control statement, which is a /* statement, is ejected into stacker 2.

THE SSERV PROGRAM, CHARTS 82 TO 83

Sequence of Operation

SSERV is a one-phase program. 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 SYSIPT. When an EOF (/*) condition is encountered on SYSIPT, the SSERV program is terminated.

I/O Flow of the SSERV Program

The I/O Flow follows the same rules for control statement input, library search and punched output as in the RSERV program (see above).

THE PSERV PROGRAM, CHARTS 84 TO 85

Sequence of Operation

The PSERV program is a one-phase program. The control statement may have multiple operands. Each time a procedure 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 SYSIPT. When an EOF (/*) condition is encountered on SYSIPT, the PSERV program is terminated.

I/O Flow of the PSERV program

The control statements invoking the PSERV prorgam are read from SYSIPT. PSERV prints or punches form the procedure library on SYSRES. All punched output is ejected into stacker 2 of SYSPCH. Printed output goes to SYSLST.

THE $$BSYSWR TRANSIENT, CHART 86

Sequence of Operation

This transient has two functions:
When invoked with register 0 containing 0 it allows to write on SYSRES from that partition. When invoked with register 0 not containing 0 it updates the label area control block in the SVA with the location and length of the label area and with the location of the procedure library.

$$BSYSWR tests for the function to be performed. For label area information update, it reads the system directory and places the information on the label area control block in GETVIS of the SVA. For writing on SYSRES, it sets a bit in the partition communication area. The transient also uses the system communication area and the SLA control block.

THE $$BOPNLB TRANSIENT, CHART 87

Sequence of Operation

This transient provides the disk address of the respective system library and the status of private and system libraries as to whether they contain active members or not. For private libraries, the address has to be provided via $$BOPEN.

With SVC 2, a control block address is passed which indicates for which library type the OPEN is to be done. For a private source statement library, the control block must also contain the address of the library. The phase reads the system directory to get the location of the system libraries. If the desired library is present the library descriptor record is read to obtain the member status.

Switches used:

```
OVLAYA    DSECT
BUCK1     DS    XL4              DISK ADDR OF PRIVATE LIBRARY C2C1H2R
*                                  (INPUT FRM CALLER)
BUCK2     DS    XL4              DISK ADDR OF SYSTEM LIBRARY C2C1H2R
*                                  (OUTPUT TO CALLER)
PRVSW     DS    X                SWITCH BYTE
PROCLIB   EQU   X'80'            BIT 0 = 0: OPEN SOURCE LIB
*                                      = 1: OPEN PROCLIB
*                                            (INPUT FROM CALLER)
PRIV      EQU   X'01'            BIT 7: PRIVATE LIBRARY OPENED
*                                  (INPUT FROM CALLER)
*                                  THIS BIT IS SET TO 0 IF A PRIVATE
*                                  LIBRARY HAS NO ACTIVE MEMBERS,
*                                  ELSE UNCHANGED (OUTPUT TO CALLER)
PRSSSW    DS    X                SWITCHES (OUTPUT TO CALLER)
ACTIVE    EQU   X'40'            BIT 1: SYSTEM LIBRARY
*                                  CONTAINS ACTIVE MEMBERS
```

THE DTFSL AND DTFPL MACROS, CHARTS 88 TO 90

Invocation

The following description applies to both macros although only DTFSL is mentioned specifically. They serve to access the source statement and the procedure libraries. The code of the macro is called by the internal imperative macros.

FNDSL   used in librarian and other programs
GETSL

NTSL   used only in conjunction with GETSL
PTSL

READSL   used in librarian programs only

The individual imperative macros which use the DTFSL code have the following functions:

FNDSL : Find a book and save its disk address if found.
A branch address must be specified where to branch to if the book is not found.
Register 1 points to a 9-byte book name.
The address of the found book is stored in the DTFSL control blocks for following GETs and READs.

GETSL : Retrieve a book sequentially (by one card with each GET request).

NTSL : If inner macros are encountered during GETSL processing. Note position where retrieving is to continue.

PTSL : After processing of inner macro restore that position.

READSL : Transfer a source statement record (160 bytes) to user buffer.

NTSL returns to the caller the position of retrieving. For releases prior to DOS/VSE, the position is contained in register 1. For DOS/VSE, register 1 contains the address of an entry into an internal stack called note word table. This table contains the disk and buffer addresses to which a GETSL macro returns after an inner macro has been executed.

The DTFSL macro works in two modes: For releases prior to DOS/VSE, it allocates all necessary blocks and buffers and executes in its own expansion.

For DOS/VSE, actual processing is done by the phase $IJBLBSL which resides in the SVA. This phase builds I/O buffers, access blocks, and the note word table. The DTFSL expansion then only sets up the save area and the basic request list and builds the request list.

The DTFSL is coded as follows:

$$\text{\&DTFSL}\quad\text{DTFSL}\quad\text{\&NOTEPNT=}\left\{\begin{matrix}\text{NO}\\\text{YES}\end{matrix}\right\}\text{,\&PRIVATE=}\left\{\begin{matrix}\text{NO}\\\text{YES}\end{matrix}\right\}\text{,\&ERROR=label,}$$

$$\text{\&LIBR=}\left\{\begin{matrix}\text{NO}\\\text{YES}\end{matrix}\right\}$$

where the parameters have the meaning:

NOTEPNT=NO .... Only FIND and GET allowed.
NOTEPNT=YES ... FIND, GET, NOTE and POINT allowed.
PRIVATE=NO .... Operates on system source library only.
PRIVATE=YES ... Operates on private source library too.
LBR=YES .... Special actions for librarian programs.
LBR=NO .... Normal processing
ERROR=LABEL ... Entry point of error routine.
(For I/O error and bad records).

Sequence of Operation:

The first macro call (FNDSL) is routed to a subroutine which sets the mode of operation (either prior to DOS/VSE or DOS/VSE mode). All following macro calls will work then in that mode. The first call (FNDSL) opens also the source statement library (private and/or system) assigned to the partition. GETSL de-compresses the 160 byte logical records in the source statement library to 80 byte card images. NTSL provides the position of current processing of the GETSL macro. PTSL restores that position.

Control Blocks and Switches used by DTFSL and DTFPL.

The control blocks for releases prior to DOS/VSE see in the expansion of the macros.

For DOS/VSE, the macros use:

the note word table	DTFSNWT
the request list	DTFSRQL
the access control block	DTFSACB
the first-time-entered-switch	DTFSL1ST

The note word table has the following format:

```
*----------------------------------------------------------------------
*        NOTE INFORMATION WORD STACK DSECT (NOT FOR PROCEDURE)
*----------------------------------------------------------------------

DTFSNWT  DSECT
         DS    0XL88
DTFSNWAV DS    XL4                    ADDRESS OF FIRST FREE
*                                     NOTE WORD IN STACK
         DS    0XL84                  STACK OF NOTE WORDS
DTFSNIW  DS    0XL14                  FIRST NOTE INFORMATION WORD
DTFSNWAD DS    XL2                    DISPLACEMENT INTO DTFSACB
DTFSNWCS DS    XL5                    CURRENT SEEK/ LOCATE ADDR.
DTFSNWCV DS    XL1                    CURRENT SECTOR VALUE
DRFSNWBN DS    XL4                    END ADDRESS OF BUFFER+1
DRFSNWBC DS    XL2                    CURRENT DISPLACEM.IN BUFFER
DTFSNIWE EQU   *
DTFSNIWL EQU   DTFSNIWE-DTFSNIW       LENGTH OF ONE NOTE WORD
         DS    5XL14                  OTHER NOTE WORDS IN STACK
DTFSNWSE EQU   *
DTFSNWSL EQU   DTFSNWSE-DTFSNWT       LENGTH OF STACK
         DS    0D                     ALOGN NEXT CONTROL BLOCK
DTDSNWTE EQU   *                      END OF NOTE WORD TABLE
DTFSNWTL EQU   DTFSNWTE-DTFSNWT       LENGTH OF NOTE TABLE SPACE
*----------------------------------------------------------------------
```

This table is set up when the first FNDSL is issued and it serves as interface between NTSL and PTSL.

The request list is passed from the DTFSL to the phase $IJBLBSL and contains information about the requests such as request type and pointers to resources.

The access control block has the following format:

```
DTFSACB  DSECT
         DS    0XL511

DTFSABB  DS    0XL28                 BUFFER CONTROL
DTFSABDC DS    XL4                   CURRENT ADDRESS IN DIR.BUF.
DTFSABDF DS    XL4                   ADDRESS OF DIRECTORY BUFFER
DTFSABDE DS    XL4                   END ADDRESS OF DIRECT. BUF.
DTFSABMC DS    XL4                   CURRENT ADDRESS IN MBR.BUF.
DTFSABMN DS    XL4                   END ADDR. OF MEMBER BUFF.+1
DTFSABMF DS    XL4                   ADDRESS OF MEMBER BUFFER
DTFSABME DS    XL4                   END ADDRESS OF MEMBER BUFF.

DTFSABFX DS    0XL9                  FIX LIST FOR PRIVATE AND
*                                    SYSTEM I/O REQUESTS
DTFSABFA DS    XL4                   BEGIN ADDR. IN FIX LIST
DTFSABFE DS    XL4                   END ADDRESS IN FIX LIST
DTFSABFT DS    XL1                   TERMINATING CODE FOR FIX L.

DTFSAB$B DS    0XL58                 $$BOPNLB COMPATIBILITY
DTFSAB$P DS    XL4                   ADDRESS OF PRIVATE SOURCE
DTFSAB$S DS    XL4                   ADDRESS OF SYSTEM SOURCE

DTFSAB$1 DS    XL1                   SWITCH BYTE 1
PROCLIB  EQU   X'80'                 $$BOPNLB CALLED FOR PROC.LB
PRIVACT  EQU   X'01'                 ACTIVE MEMBERS IN PRIVATE
IJBSL    EQU   X'02'                 $$BOPNLB CALLED BY $IJBLBSL
DTFSAB$2 DS    XL1                   SWITCH BYTE 2
SACTIV   EQU   X'40'                 ACTIVE MEMBERS IN SYSTEM

DTFABPSK DS    XL24                  CKD : SEEK ADDRESS OF BEGIN
*                                    OF PRIVATE LIBRARY
*                                    FBA : EXTENT/LOCATE WORDS
*                                    TO READ LIBRARY AND DIRECT.
DTFSABSE EQU   *
LDTFABSK EQU   DTFSABSE-DTFABPSK     LENGTH OF START ADDR. INFO.

DTFABSSK DS    XL24                  START ADDR. OF SYSTEM LIBR.

         DS    0D                    ALIGN PRIVATE PART
DTFSABP  DS    0XL208                CONTROL BLOCK PRIVATE SRCE
*                                    (FOR PROCEDURE UNUSED)
DTFSABPS DS    XL1                   SWITCH BYTE
DTFSFBA  EQU   X'80'                 LIBRARY IS ON FBA DEVICE
DTFSPRNO EQU   X'08'                 NO PRIVATE LIBRARY EXPANS.
DTFSPRUA EQU   X'04'                 PRIVATE LIBRARY IS UNASSG.
DTFSNOAC EQU   X'02'                 LIBRARY CONTAINS NO ACTIVE
*                                    MEMBERS

DTFABPIO DS    XL24                  IORB FOR PRIVATE LIBRARY

         DS    0D                    ALIGN ON DOUBLE WORD

DTFABPCD DS    XL56                  CCW CHAIN FOR PRIV. DIRECT.
DTFABPSD DS    XL24                  CURRENT ADDR. IN PRIV. DIR.
DTFABPRD DS    XL1                   CURRENT SECTOR VALUE PRIV.
         ORG   DTFABPRD
DTFABPBD DS    XL1                   NMBR FBA BLCKS/DIR. RECORD

         DS    0D                    ALIGN ON DOUBLE WORD
```

```
DTFABPCM DS    XL56                       CCW CHAIN FOR PRIV. MEMBER SP.
DTFABPSM DS    XL24                       CURRENT ADDR. IN PRIV. MEMBER SP.
DTFABPRM DS    XL1                        CURRENT SECTOR VALUE PRIV.
         ORG   DTFABPRM
DTFABPBM DS    XL1                        NMBR FBA BLCKS/MEMBER RECORD
DTFAMPAM DS    XL5                        CURRENT ADDR.-1 IN MEMBER
DTFABPVM DS    XL1                        CURR.SECTOR VAL.MEMBER(CKD)
DTFABPSV DS    XL1                        RPS VALUE FOR DEVICE (CKD)
DTFSABPE EQU   *
DTFSABPL EQU   DTFSABPE-DTFSABP           LENGTH OF PRIV. LIBRARY CNTL

         DS    0D                         ALIGN SYSTEM PART
DTFSABS  DS    0XL208                     CONTROL BLOCK SYSTEM SOURCE
*                                         OR PROCEDURE LIBRARY
DTFSABSS DS    XL1                        SWITCH BYTE
DTFABSIO DS    XL24                       IORB FOR SYSTEM LIBRARY
         DS    0D                         ALIGN SYSTEM PART
DTFABSCD DS    XL56                       CCW CHAIN FOR SYSTEM DIR.
DTFABSSD DS    XL24                       CURRENT ADDR. IN SYST. DIR.
DTFABSRD DS    XL1                        CURRENT SECTOR VALUE SYST.
         ORG   DTFABSRD
DTFABSBD DS    XL1                        NMBR FBA BLCKS/DIR. RECORD

         DS    0D                         ALIGN ON DOUBLE WORD
DTFABSCM DS    XL56                       CCW CHAIN FOR SYST.MEMBER SP.
DTFABSSM DS    XL24.                      CURRENT ADDR. IN SYST. MEMBER SP.
DTFABSRM DS    XL1                        CURRENT SECTOR VALUE SYSTEM
         ORG   DTFABSRM
DTFABSBM DS    XL1                        NMBR FBA BLCKS/MEMBER RECORD
DTFABSAM DS    XL5                        CURRENT ADDR.-1 IN MEMBER
DTFABSVM DS    XL1                        CURR. SECTOR VAL. MEMBER (CKD)
DTFABSSV DS    XL1                        RPS VALUE FOR DEVICE (CKD)

         DS    0D                         ALIGNMENT FOR NOTE TABLE
DTFSACBE EQU   *                          END OF DTFSACB
DTFSACBL EQU   DTFSACBE-DTFSACB           LENGTH OF DTFSACB
DTFSCBL  EQU   DTFSRQLL+DTFSACBL          CONTROL BLOCK LENGTH
```

THE PHASE $IJBLBSL, CHARTS 91 TO 93

Sequence of Operation

The requests to access the source and procedure libraries from a macro are passed in the form of a request list (DTFSRQL) which indicates the operation requested and points to all resources needed to complete a request. It works in two modes:

- when requested from compilers etc., it searches through the directory(ies) of the assigned library(ies) to find a book and moves card image of source statements to the user card I/O area.
- when requested from librarian programs, the phase searches the directory of a single library only and returns complete 160 byte records to the program issuing the request.

The phase resides in the SVA. A first call from a partition causes the library or libraries to be opened, that is, the library addresses, channel programs, buffer addresses to be filled into the access control block located in the partition GETVIS area. The directories are then searched for the bookname as described above.

GENERAL CHARTS

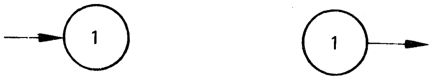
GENERAL CHARTS CONVENTIONS:

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

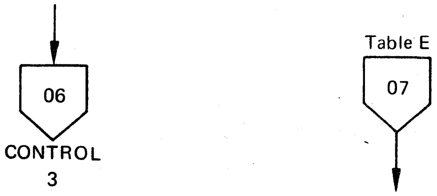
```
 xxx
┌─────────────────────────────────────────────────────┐
│yyy                                               zzz│
├─────────────────────────────────────────────────────┤
│1.                                                   │
│2.   step members                                    │
│3.                                                   │
.                                                     .
.                                                     .
.                                                     .
```

Where: xxx marks the label and routine name
yyy says shortly what the routine does
zzz is the reference to the detail chart(s), if any.
The step numbers are given from 1 to n within this routine only.

2. On-page connectors are such:



3. Off-page connectors are such



where: the number in the frame marks the chart from or to which we go. The word above (incoming) or below (outgoing) marks the label (routine) on that chart, the number under the word marks the step within the routine to which we go if it is not step 1.

Chart 01. COPYSERV (Detail Chart AA)

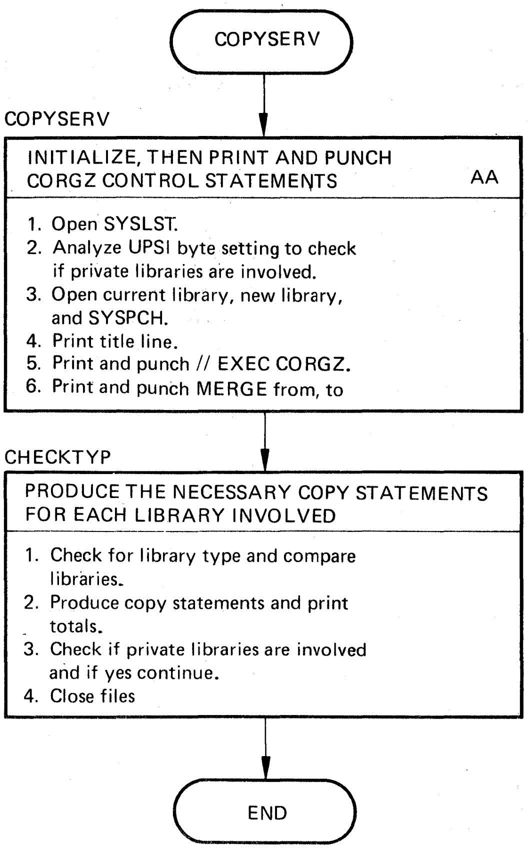
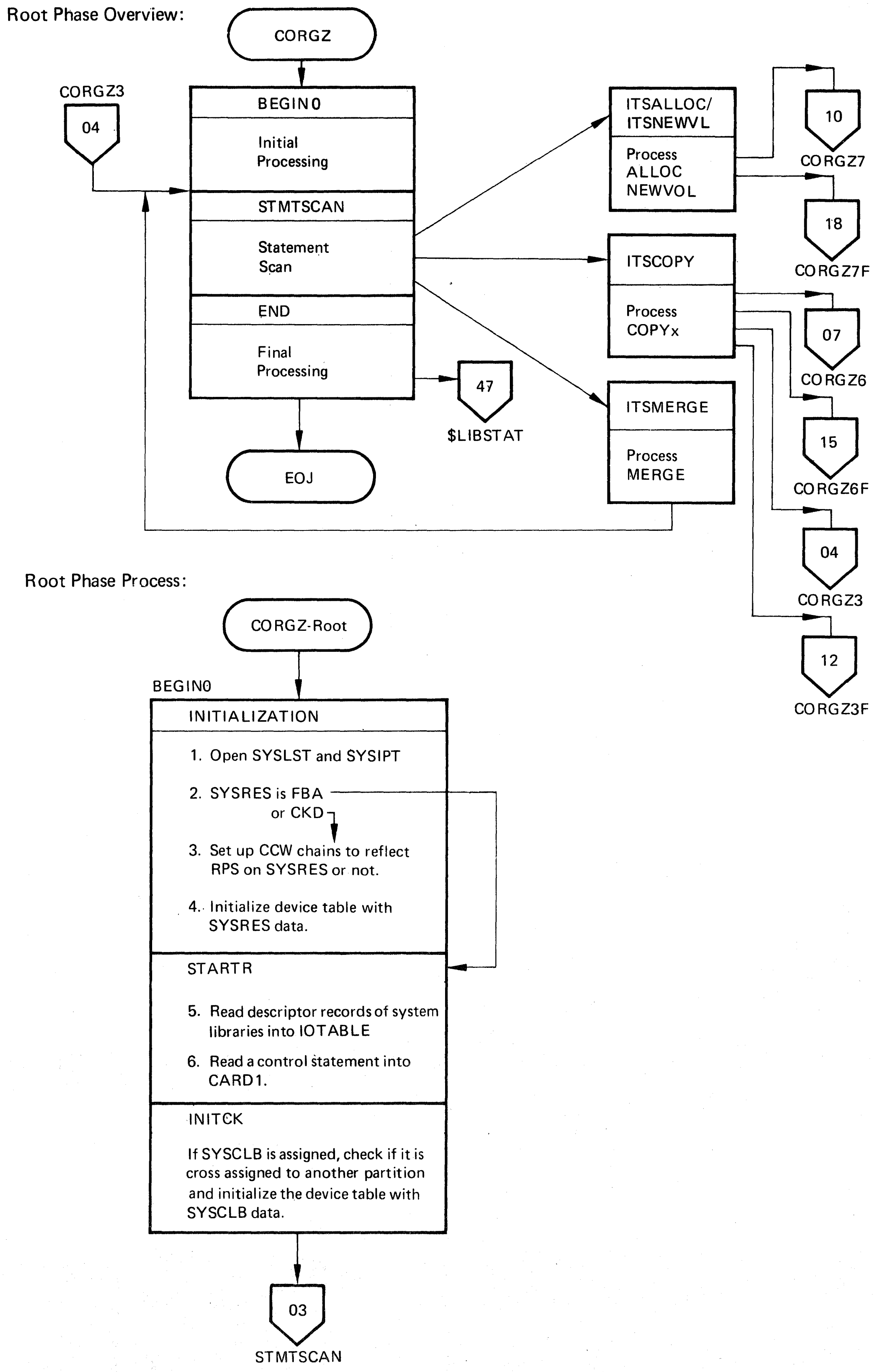

COPYSERV
COPYSERV
INITIALIZE, THEN PRINT AND PUNCH CORGZ CONTROL STATEMENTS
AA
1. Open SYSLST.
2. Analyze UPSI byte setting to check if private libraries are involved.
3. Open current library, new library, and SYSPCH.
4. Print title line.
5. Print and punch // EXEC CORGZ.
6. Print and punch MERGE from, to
CHECKTYP
PRODUCE THE NECESSARY COPY STATEMENTS FOR EACH LIBRARY INVOLVED
1. Check for library type and compare libraries.
2. Produce copy statements and print totals.
3. Check if private libraries are involved and if yes continue.
4. Close files
END

Chart 02. CORGZ - Root Phase (Part 1 of 2)



Root Phase Overview:

Root Phase Process:

Chart 03. CORGZ - Root Phase (Part 2 of 2)

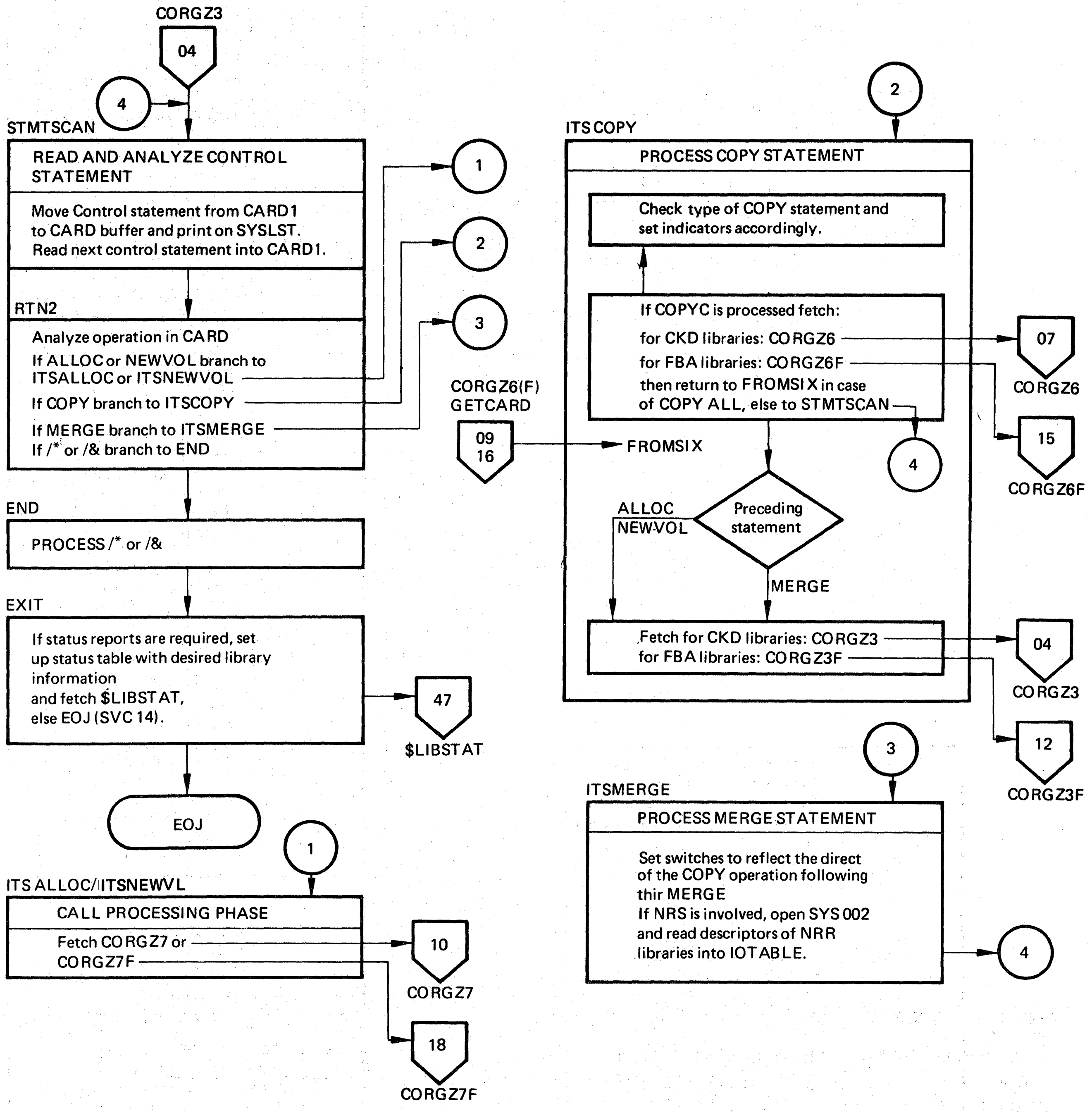

CORGZ3
04
4
STMTSCAN
READ AND ANALYZE CONTROL STATEMENT
Move Control statement from CARD1 to CARD buffer and print on SYSLST. Read next control statement into CARD1.
RTN2
Analyze operation in CARD
If ALLOC or NEWVOL branch to ITSALLOC or ITSNEWVOL
If COPY branch to ITSCOPY
If MERGE branch to ITSMERGE
If /* or /& branch to END
1
2
3
END
PROCESS /* or /&
EXIT
If status reports are required, set up status table with desired library information and fetch $LIBSTAT, else EOJ (SVC 14).
47
$LIBSTAT
EOJ
1
ITS ALLOC/ITSNEWVL
CALL PROCESSING PHASE
Fetch CORGZ7 or CORGZ7F
10
CORGZ7
18
CORGZ7F
2
ITS COPY
PROCESS COPY STATEMENT
Check type of COPY statement and set indicators accordingly.
If COPYC is processed fetch:
for CKD libraries: CORGZ6
for FBA libraries: CORGZ6F
then return to FROMSIX in case of COPY ALL, else to STMTSCAN
07
CORGZ6
15
CORGZ6F
4
CORGZ6(F)
GETCARD
09
16
FROMSIX
ALLOC
NEW-VOL
Preceding statement
MERGE
Fetch for CKD libraries: CORGZ3
for FBA libraries: CORGZ3F
04
CORGZ3
12
CORGZ3F
3
ITSMERGE
PROCESS MERGE STATEMENT
Set switches to reflect the direct of the COPY operation following thir MERGE
If NRS is involved, open SYS 002 and read descriptors of NRR libraries into IOTABLE.
4

Chart 04. CORGZ3 (Part 1 of 3)

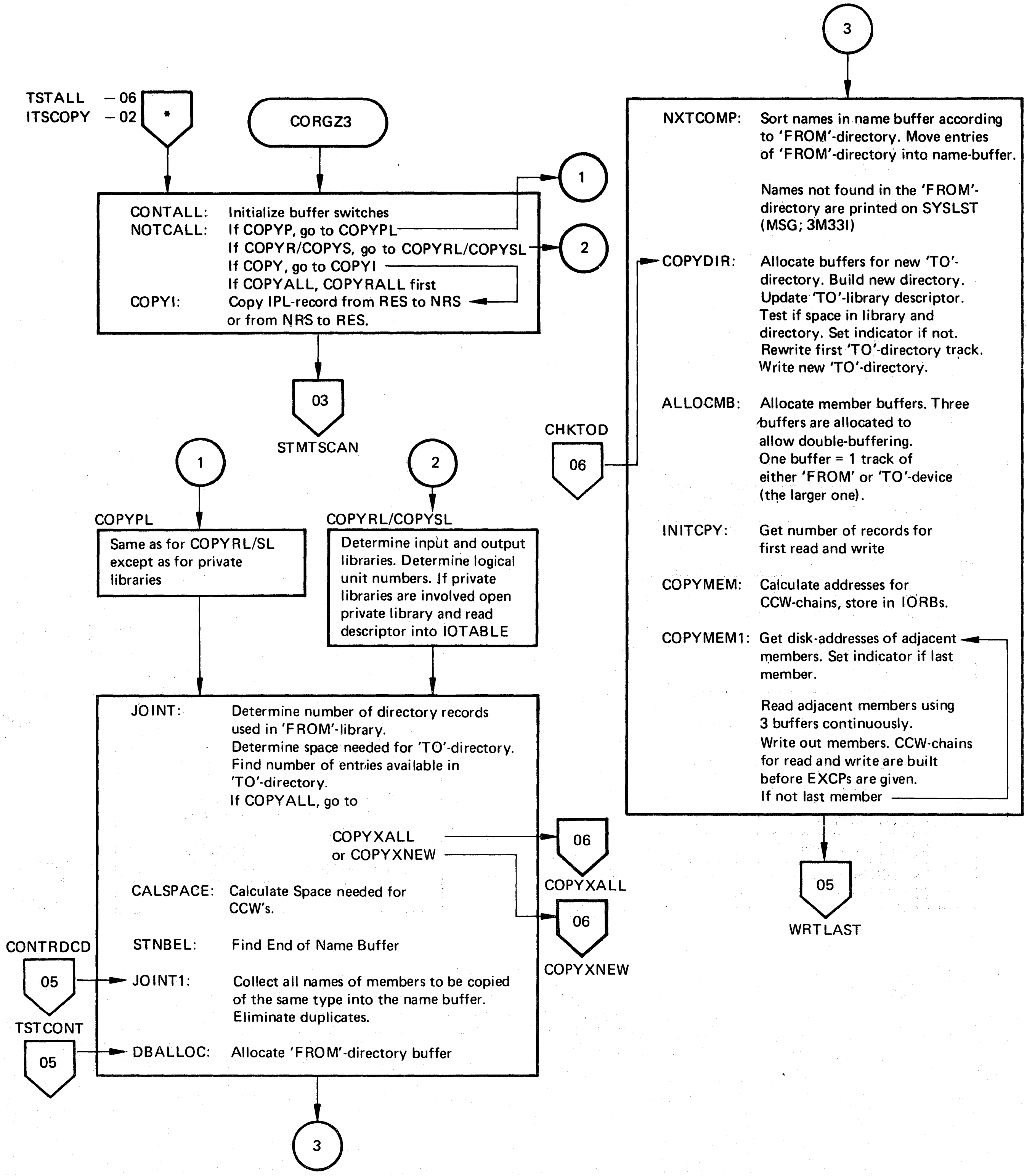

3
TSTALL -06
ITSCOPY -02
*
CORGZ3
1
2
CONTALL: Initialize buffer switches
NOTCALL: If COPYP, go to COPYPL
If COPYR/COPYS, go to COPYRL/COPYSL
If COPY, go to COPYI
If COPYALL, COPYRALL first
COPYI: Copy IPL-record from RES to NRS or from NRS to RES.
03
STMTSCAN
1
COPYPL
Same as for COPYRL/SL except as for private libraries
2
COPYRL/COPYSL
Determine input and output libraries. Determine logical unit numbers. If private libraries are involved open private library and read descriptor into IOTABLE
JOINT: Determine number of directory records used in 'FROM'-library. Determine space needed for 'TO'-directory. Find number of entries available in 'TO'-directory. If COPYALL, go to
COPYXALL
or COPYXNEW
06
COPYXALL
06
COPYXNEW
CALSPACE: Calculate Space needed for CCW's.
STNBEL: Find End of Name Buffer
CONTRDCD
05
JOINT1: Collect all names of members to be copied of the same type into the name buffer. Eliminate duplicates.
TSTCONT
05
DBALLOC: Allocate 'FROM'-directory buffer
3
CHKTOD
06
NXTCOMP: Sort names in name buffer according to 'FROM'-directory. Move entries of 'FROM'-directory into name-buffer.
Names not found in the 'FROM'-directory are printed on SYSLST (MSG; 3M33I)
COPYDIR: Allocate buffers for new 'TO'-directory. Build new directory. Update 'TO'-library descriptor. Test if space in library and directory. Set indicator if not. Rewrite first 'TO'-directory track. Write new 'TO'-directory.
ALLOCMB: Allocate member buffers. Three buffers are allocated to allow double-buffering. One buffer = 1 track of either 'FROM' or 'TO'-device (the larger one).
INITCPY: Get number of records for first read and write
COPYMEM: Calculate addresses for CCW-chains, store in IORBs.
COPYMEM1: Get disk-addresses of adjacent members. Set indicator if last member.
Read adjacent members using 3 buffers continuously. Write out members. CCW-chains for read and write are built before EXCPs are given. If not last member
05
WRTLAST

Chart 05. CORGZ3 (Part 2 of 3)

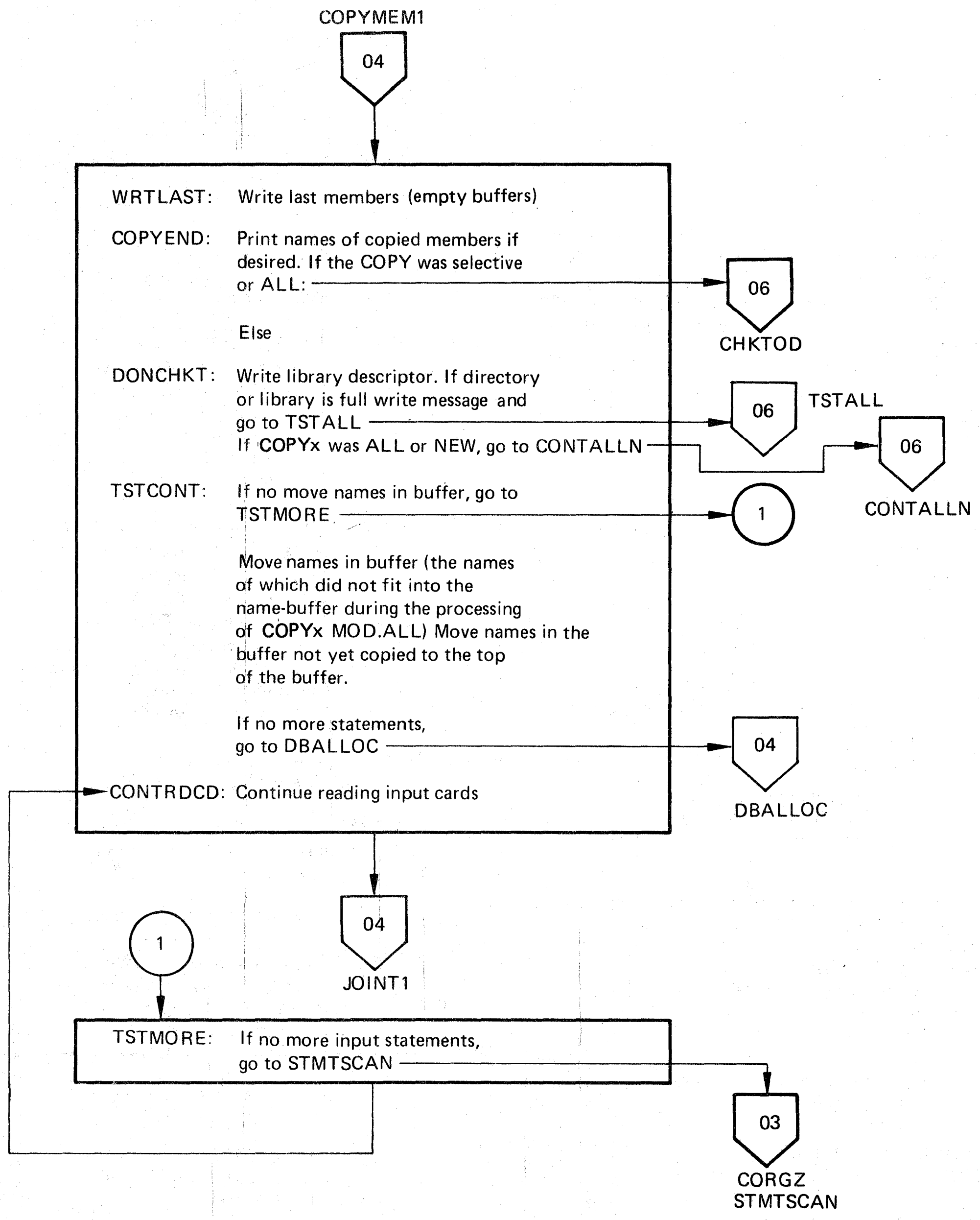

COPYMEM1
04
WRTLAST: Write last members (empty buffers)
COPYEND: Print names of copied members if desired. If the COPY was selective or ALL:
06
CHKTOD
Else
DONCHKT: Write library descriptor. If directory or library is full write message and go to TSTALL
06
TSTALL
If COPYx was ALL or NEW, go to CONTALLN
06
CONTALLN
TSTCONT: If no move names in buffer, go to TSTMORE
1
Move names in buffer (the names of which did not fit into the name-buffer during the processing of COPYx MOD.ALL) Move names in the buffer not yet copied to the top of the buffer.
If no more statements, go to DBALLOC
04
DBALLOC
CONTRDCD: Continue reading input cards
04
JOINT1
1
TSTMORE: If no more input statements, go to STMTSCAN
03
CORGZ
STMTSCAN

Chart 06. CORGZ3 (Part 3 of 3)

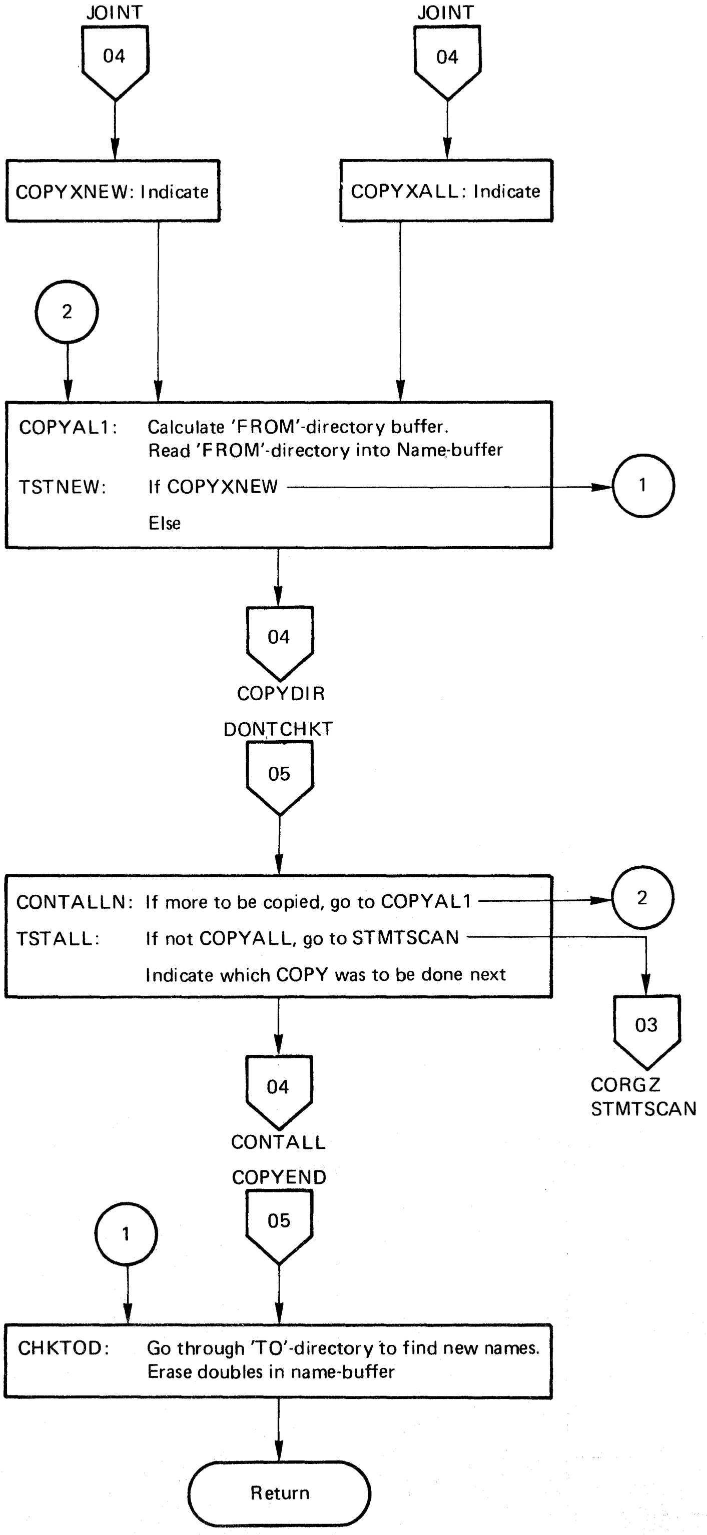

JOINT
04
JOINT
04
COPYXNEW: Indicate
COPYXALL: Indicate
2
COPYAL1: Calculate 'FROM'-directory buffer.
Read 'FROM'-directory into Name-buffer
TSTNEW: If COPYXNEW
Else
1
04
COPYDIR
DONTCHKT
05
CONTALLN: If more to be copied, go to COPYAL1
2
TSTALL: If not COPYALL, go to STMTSCAN
Indicate which COPY was to be done next
03
CORGZ
STMTSCAN
04
CONTALL
COPYEND
1
05
CHKTOD: Go through 'TO'-directory to find new names.
Erase doubles in name-buffer
Return

Chart 07. CORGZ6 (Part 1 of 3)

CORGZ6 Overview:

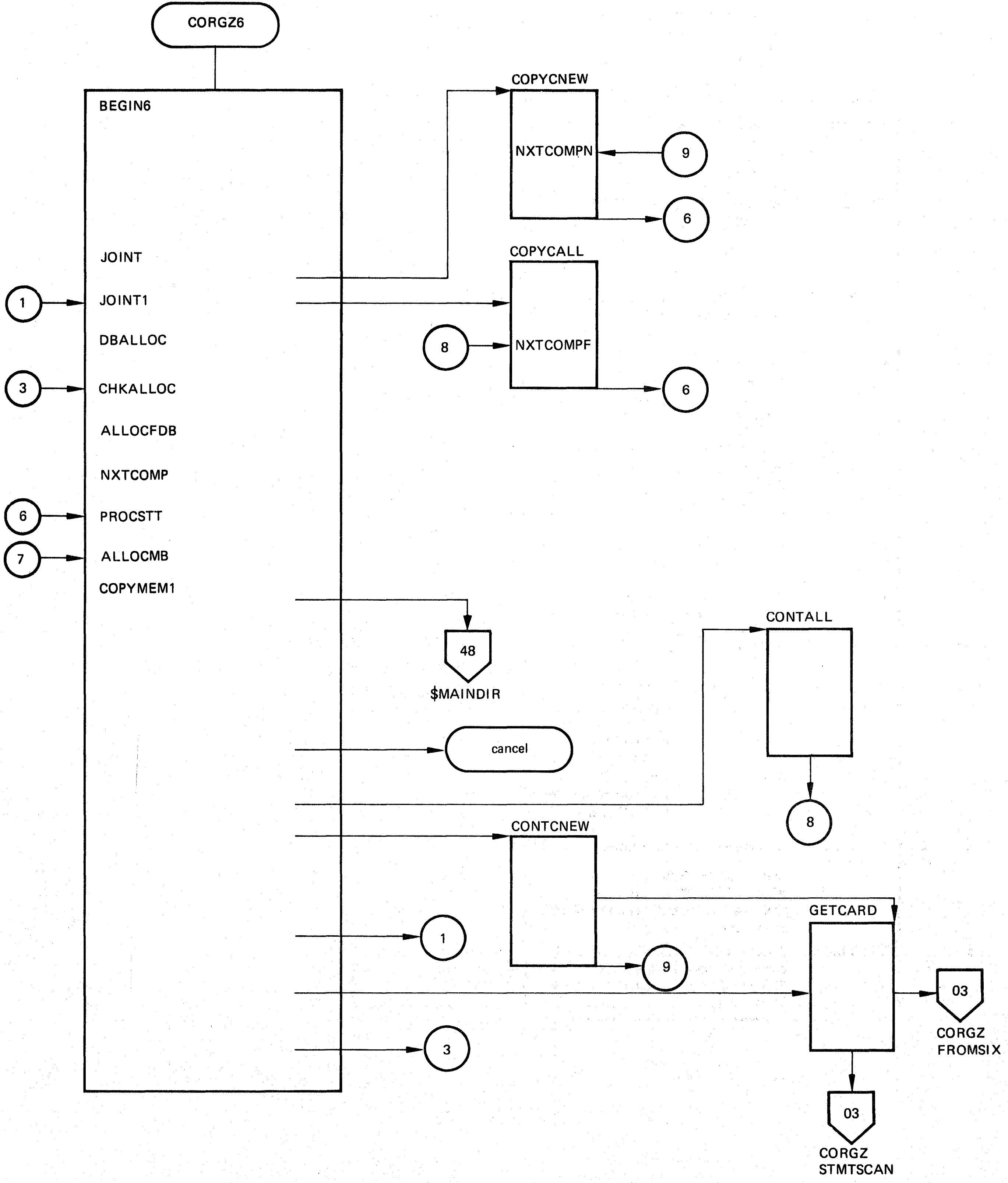
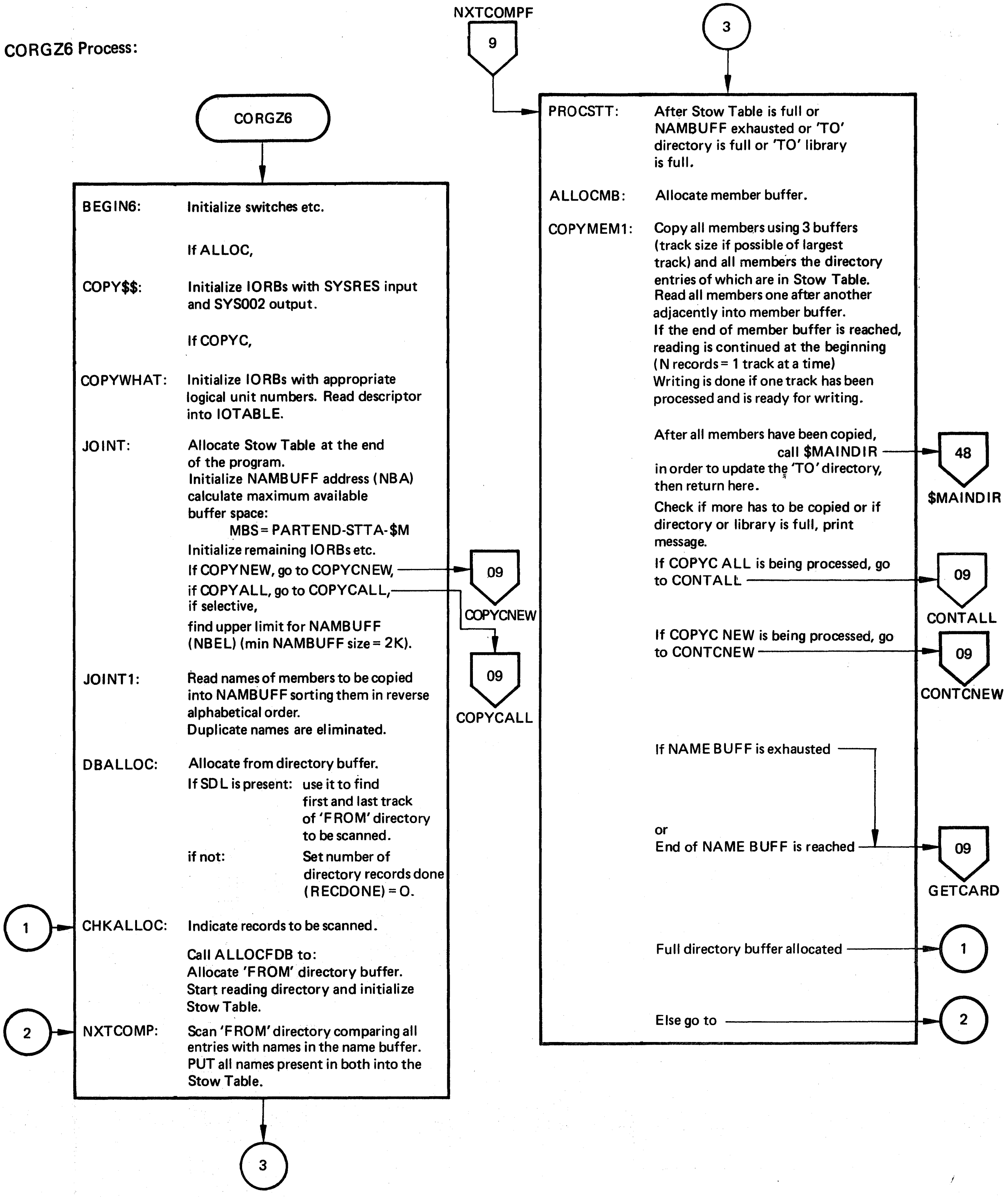


Chart 08. CORGZ6 (Part 2 of 3)

CORGZ6 Process:



BEGIN6: Initialize switches etc.

If ALLOC,

COPY$$: Initialize IORBs with SYSRES input and SYS002 output.

If COPYC,

COPYWHAT: Initialize IORBs with appropriate logical unit numbers. Read descriptor into IOTABLE.

JOINT: Allocate Stow Table at the end of the program.
Initialize NAMBUFF address (NBA)
calculate maximum available buffer space:
MBS = PARTEND-STTA-$M
Initialize remaining IORBs etc.
If COPYNEW, go to COPYCNEW,
if COPYALL, go to COPYCALL,
if selective,
find upper limit for NAMBUFF (NBEL) (min NAMBUFF size = 2K).

JOINT1: Read names of members to be copied into NAMBUFF sorting them in reverse alphabetical order.
Duplicate names are eliminated.

DBALLOC: Allocate from directory buffer.
If SDL is present: use it to find first and last track of 'FROM' directory to be scanned.
if not: Set number of directory records done (RECDONE) = O.

CHKALLOC: Indicate records to be scanned.

Call ALLOCFDB to:
Allocate 'FROM' directory buffer.
Start reading directory and initialize Stow Table.

NXTCOMP: Scan 'FROM' directory comparing all entries with names in the name buffer.
PUT all names present in both into the Stow Table.

PROCSTT: After Stow Table is full or NAMBUFF exhausted or 'TO' directory is full or 'TO' library is full.

ALLOCMB: Allocate member buffer.

COPYMEM1: Copy all members using 3 buffers (track size if possible of largest track) and all members the directory entries of which are in Stow Table.
Read all members one after another adjacently into member buffer.
If the end of member buffer is reached, reading is continued at the beginning (N records = 1 track at a time)
Writing is done if one track has been processed and is ready for writing.

After all members have been copied, call $MAINDIR in order to update the 'TO' directory, then return here.

Check if more has to be copied or if directory or library is full, print message.

If COPYC ALL is being processed, go to CONTALL

If COPYC NEW is being processed, go to CONTCNEW

If NAME BUFF is exhausted

or
End of NAME BUFF is reached

Full directory buffer allocated

Else go to

Chart 09. CORGZ6 (Part 3 of 3)

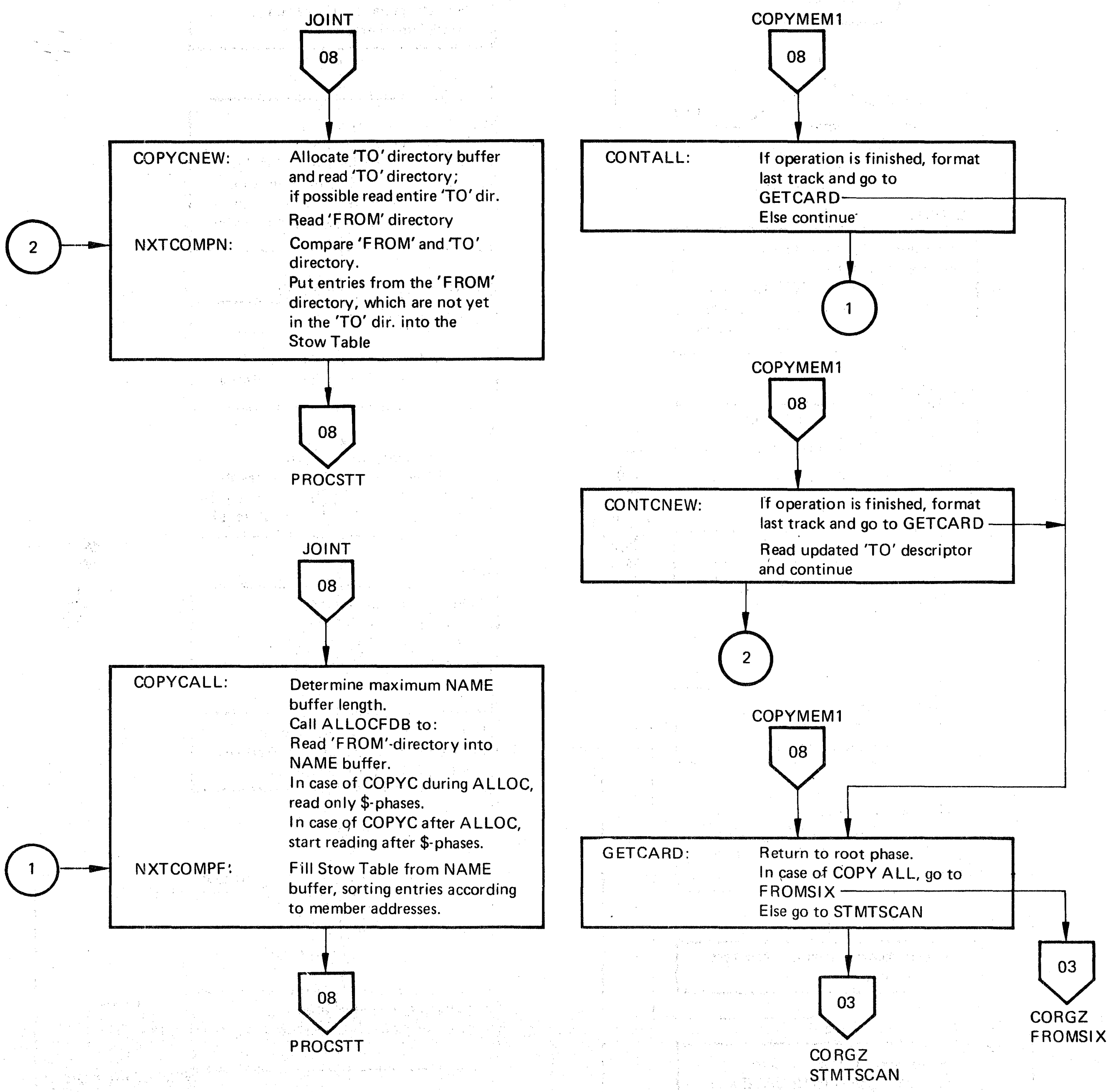

JOINT
08
COPYCNEW:
Allocate 'TO' directory buffer and read 'TO' directory; if possible read entire 'TO' dir.
Read 'FROM' directory
2
NXTCOMPN:
Compare 'FROM' and 'TO' directory.
Put entries from the 'FROM' directory, which are not yet in the 'TO' dir. into the Stow Table
08
PROCSTT
JOINT
08
COPYCALL:
Determine maximum NAME buffer length.
Call ALLOCFDB to:
Read 'FROM'-directory into NAME buffer.
In case of COPYC during ALLOC, read only $-phases.
In case of COPYC after ALLOC, start reading after $-phases.
1
NXTCOMPF:
Fill Stow Table from NAME buffer, sorting entries according to member addresses.
08
PROCSTT
COPYMEM1
08
CONTALL:
If operation is finished, format last track and go to GETCARD
Else continue
1
COPYMEM1
08
CONTCNEW:
If operation is finished, format last track and go to GETCARD
Read updated 'TO' descriptor and continue
2
COPYMEM1
08
GETCARD:
Return to root phase.
In case of COPY ALL, go to FROMSIX
Else go to STMTSCAN
03
CORGZ
STMTSCAN
03
CORGZ
FROMSIX

Chart 10. CORGZ7 (Part 1 of 2)

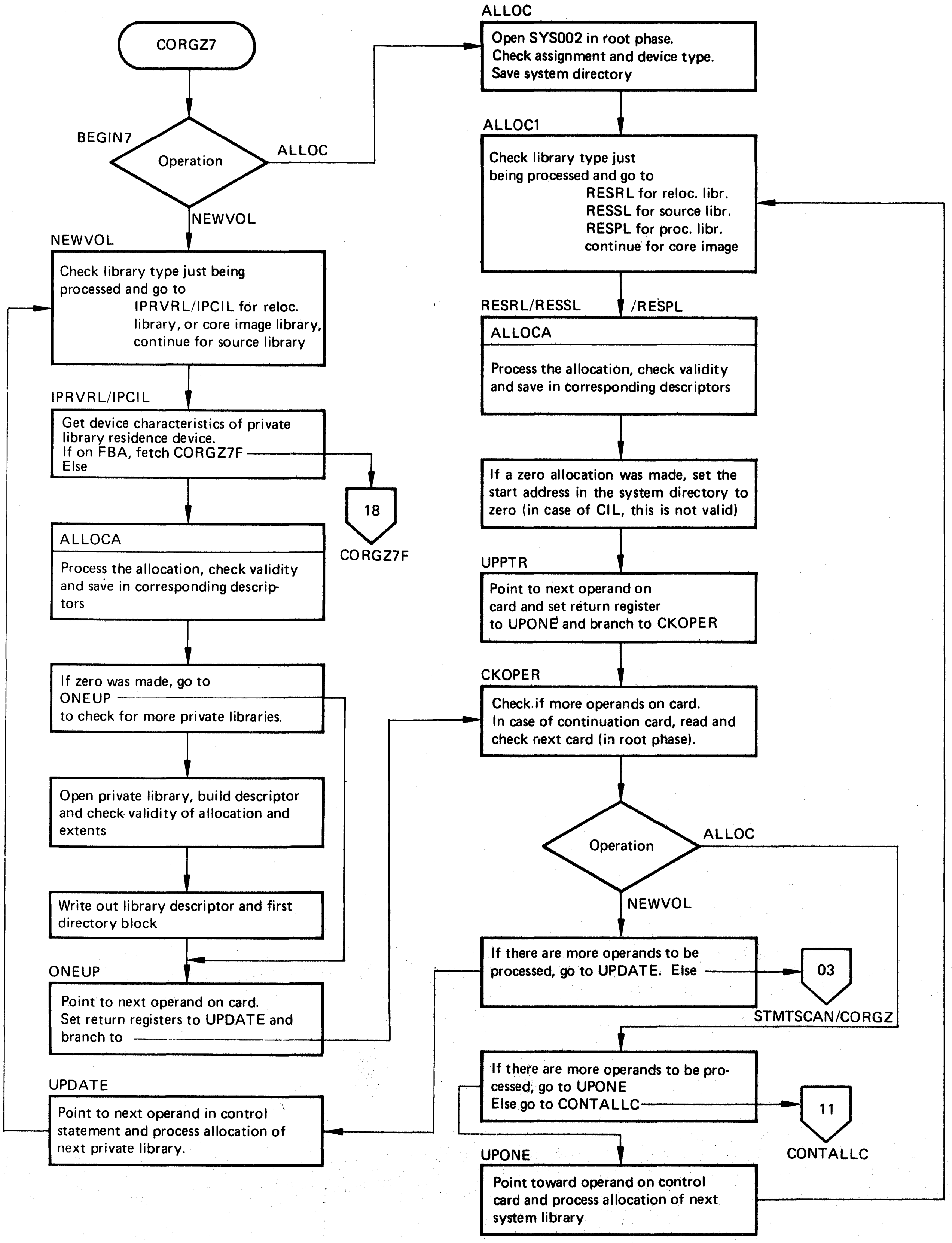
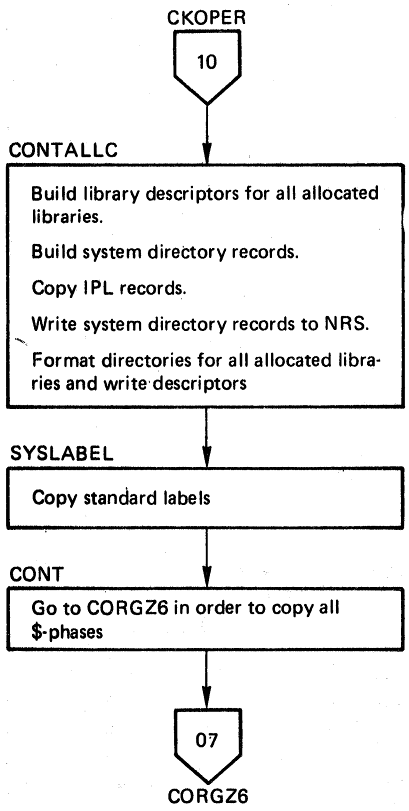

CORGZ7
BEGIN7
Operation
ALLOC
NEWVOL
NEWVOL
Check library type just being processed and go to IPRVRL/IPCIL for reloc. library, or core image library, continue for source library
IPRVRL/IPCIL
Get device characteristics of private library residence device. If on FBA, fetch CORGZ7F Else
18
CORGZ7F
ALLOCA
Process the allocation, check validity and save in corresponding descriptors
If zero was made, go to ONEUP to check for more private libraries.
Open private library, build descriptor and check validity of allocation and extents
Write out library descriptor and first directory block
ONEUP
Point to next operand on card. Set return registers to UPDATE and branch to
UPDATE
Point to next operand in control statement and process allocation of next private library.
ALLOC
Open SYS002 in root phase. Check assignment and device type. Save system directory
ALLOC1
Check library type just being processed and go to RESRL for reloc. libr. RESSL for source libr. RESPL for proc. libr. continue for core image
RESRL/RESSL
/RESPL
ALLOCA
Process the allocation, check validity and save in corresponding descriptors
If a zero allocation was made, set the start address in the system directory to zero (in case of CIL, this is not valid)
UPPTR
Point to next operand on card and set return register to UPONE and branch to CKOPER
CKOPER
Check if more operands on card. In case of continuation card, read and check next card (in root phase).
Operation
ALLOC
NEWVOL
If there are more operands to be processed, go to UPDATE. Else
03
STMTSCAN/CORGZ
If there are more operands to be processed, go to UPONE Else go to CONTALLC
11
CONTALLC
UPONE
Point toward operand on control card and process allocation of next system library

Chart 11. CORGZ7 (Part 2 of 2)



CKOPER
10
CONTALLC
Build library descriptors for all allocated libraries.
Build system directory records.
Copy IPL records.
Write system directory records to NRS.
Format directories for all allocated libraries and write descriptors
SYSLABEL
Copy standard labels
CONT
Go to CORGZ6 in order to copy all $-phases
07
CORGZ6

Chart 12. CORGZ3F (Part 1 of 3)

CORGZ3F Overview:

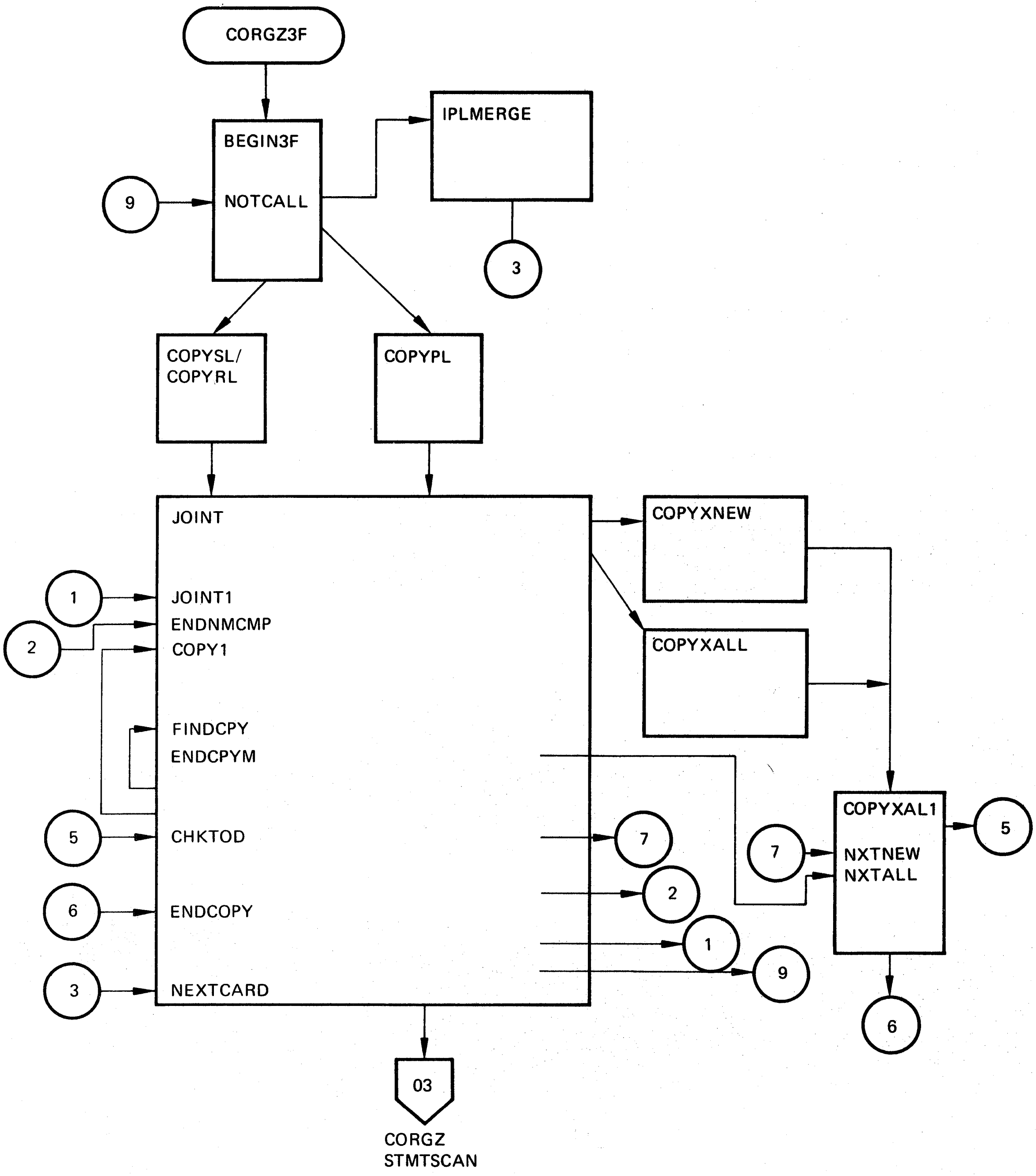
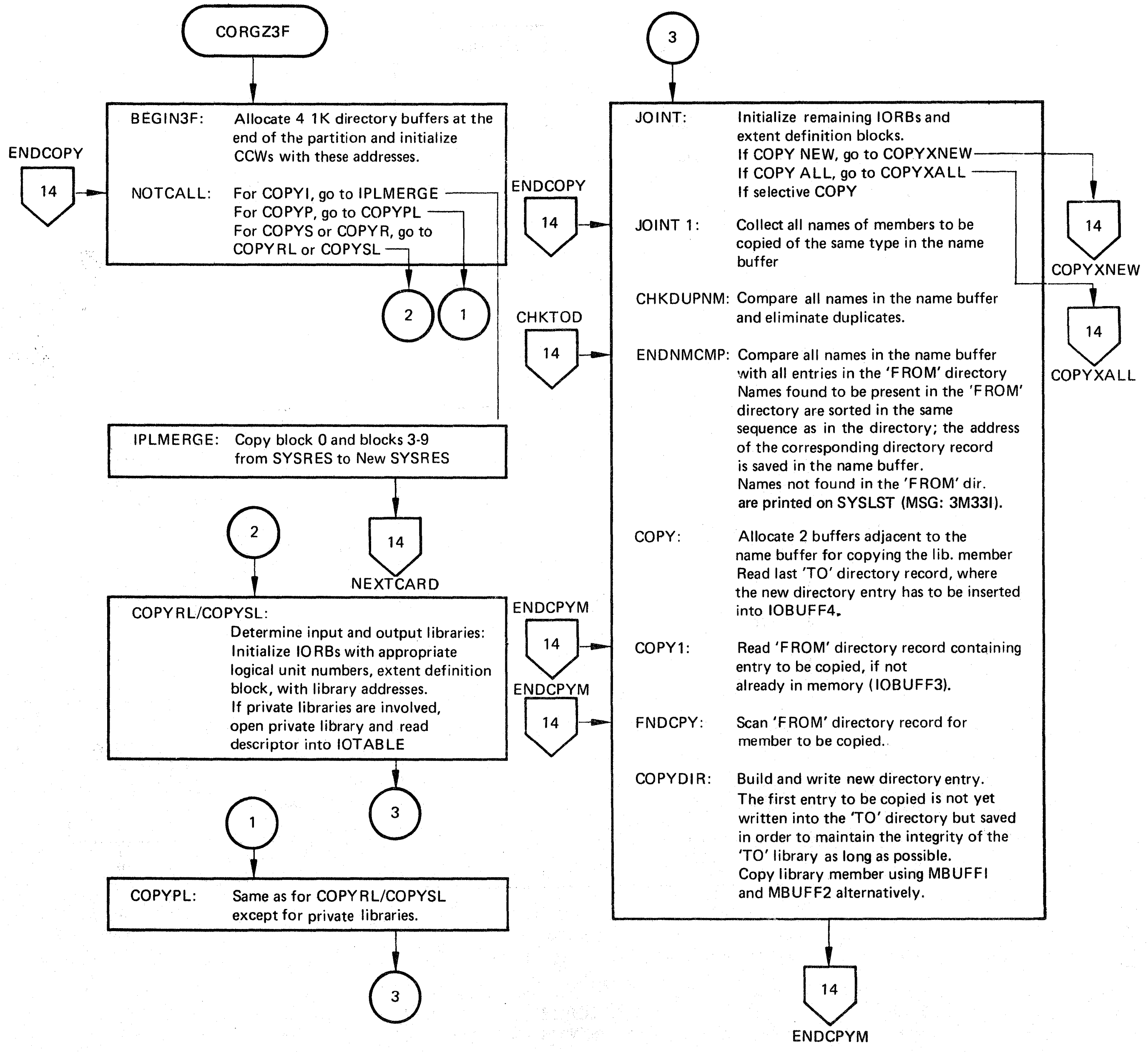


Chart 13. CORGZ3F (Part 2 of 3)

CORGZ3F Processing:



BEGIN3F: Allocate 4 1K directory buffers at the end of the partition and initialize CCWs with these addresses.

NOTCALL: For COPYI, go to IPLMERGE
For COPYP, go to COPYPL
For COPYS or COPYR, go to COPYRL or COPYSL

IPLMERGE: Copy block 0 and blocks 3-9 from SYSRES to New SYSRES

COPYRL/COPYSL: Determine input and output libraries: Initialize IORBs with appropriate logical unit numbers, extent definition block, with library addresses. If private libraries are involved, open private library and read descriptor into IOTABLE

COPYPL: Same as for COPYRL/COPYSL except for private libraries.

JOINT: Initialize remaining IORBs and extent definition blocks.
If COPY NEW, go to COPYXNEW
If COPY ALL, go to COPYXALL
If selective COPY

JOINT 1: Collect all names of members to be copied of the same type in the name buffer

CHKDUPNM: Compare all names in the name buffer and eliminate duplicates.

ENDNMCMP: Compare all names in the name buffer with all entries in the 'FROM' directory Names found to be present in the 'FROM' directory are sorted in the same sequence as in the directory; the address of the corresponding directory record is saved in the name buffer.
Names not found in the 'FROM' dir. are printed on SYSLST (MSG: 3M33I).

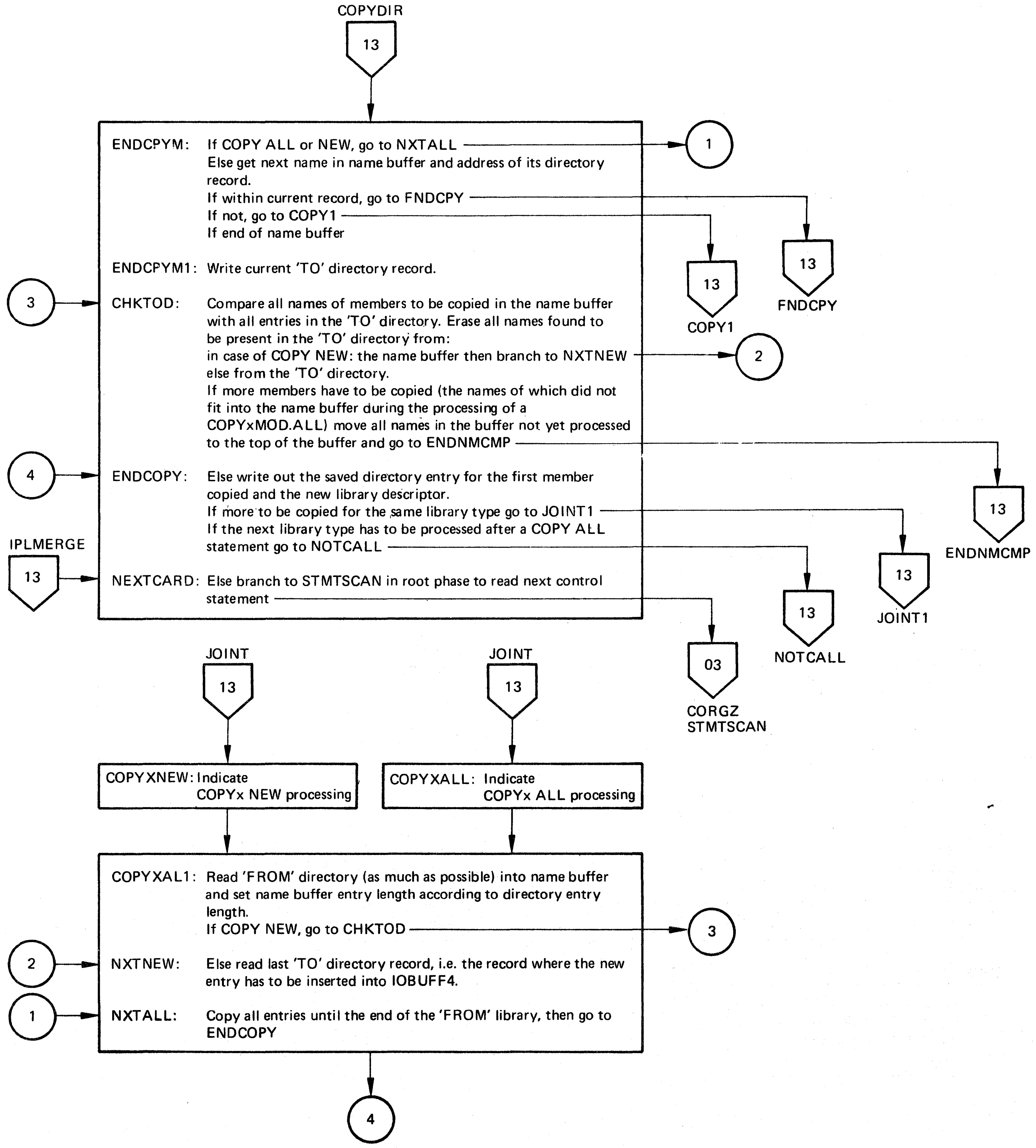
COPY: Allocate 2 buffers adjacent to the name buffer for copying the lib. member Read last 'TO' directory record, where the new directory entry has to be inserted into IOBUFF4.

COPY1: Read 'FROM' directory record containing entry to be copied, if not already in memory (IOBUFF3).

FNDCPY: Scan 'FROM' directory record for member to be copied.

COPYDIR: Build and write new directory entry. The first entry to be copied is not yet written into the 'TO' directory but saved in order to maintain the integrity of the 'TO' library as long as possible.
Copy library member using MBUFFI and MBUFF2 alternatively.

Chart 14. CORGZ3F (Part 3 of 3)



ENDCPYM: If COPY ALL or NEW, go to NXTALL
Else get next name in name buffer and address of its directory record.
If within current record, go to FNDCPY
If not, go to COPY1
If end of name buffer

ENDCPYM1: Write current 'TO' directory record.

CHKTOD: Compare all names of members to be copied in the name buffer with all entries in the 'TO' directory. Erase all names found to be present in the 'TO' directory from:
in case of COPY NEW: the name buffer then branch to NXTNEW
else from the 'TO' directory.
If more members have to be copied (the names of which did not fit into the name buffer during the processing of a COPYxMOD.ALL) move all names in the buffer not yet processed to the top of the buffer and go to ENDNMCMP

ENDCOPY: Else write out the saved directory entry for the first member copied and the new library descriptor.
If more to be copied for the same library type go to JOINT1
If the next library type has to be processed after a COPY ALL statement go to NOTCALL

NEXTCARD: Else branch to STMTSCAN in root phase to read next control statement

COPYXNEW: Indicate COPYx NEW processing

COPYXALL: Indicate COPYx ALL processing

COPYXAL1: Read 'FROM' directory (as much as possible) into name buffer and set name buffer entry length according to directory entry length.
If COPY NEW, go to CHKTOD

NXTNEW: Else read last 'TO' directory record, i.e. the record where the new entry has to be inserted into IOBUFF4.

NXTALL: Copy all entries until the end of the 'FROM' library, then go to ENDCOPY

Chart 15. CORGZ6F (Part 1 of 3)

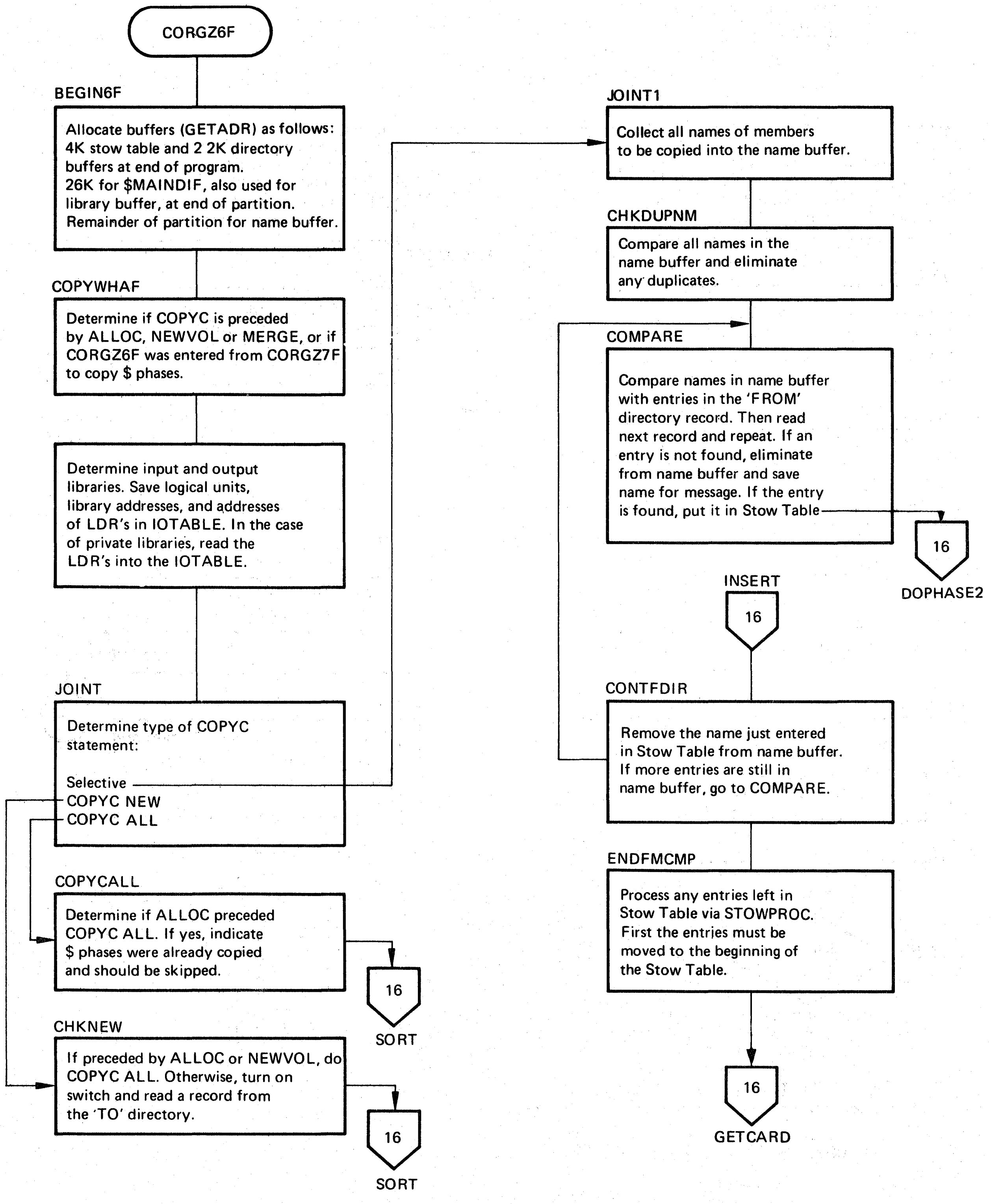

CORGZ6F
BEGIN6F
Allocate buffers (GETADR) as follows: 4K stow table and 2 2K directory buffers at end of program. 26K for $MAINDIF, also used for library buffer, at end of partition. Remainder of partition for name buffer.
COPYWHAF
Determine if COPYC is preceded by ALLOC, NEWVOL or MERGE, or if CORGZ6F was entered from CORGZ7F to copy $ phases.
Determine input and output libraries. Save logical units, library addresses, and addresses of LDR's in IOTABLE. In the case of private libraries, read the LDR's into the IOTABLE.
JOINT
Determine type of COPYC statement:
Selective
COPYC NEW
COPYC ALL
COPYCALL
Determine if ALLOC preceded COPYC ALL. If yes, indicate $ phases were already copied and should be skipped.
16
SORT
CHKNEW
If preceded by ALLOC or NEWVOL, do COPYC ALL. Otherwise, turn on switch and read a record from the 'TO' directory.
16
SORT
JOINT1
Collect all names of members to be copied into the name buffer.
CHKDUPNM
Compare all names in the name buffer and eliminate any duplicates.
COMPARE
Compare names in name buffer with entries in the 'FROM' directory record. Then read next record and repeat. If an entry is not found, eliminate from name buffer and save name for message. If the entry is found, put it in Stow Table
16
DOPHASE2
INSERT
16
CONTFDIR
Remove the name just entered in Stow Table from name buffer. If more entries are still in name buffer, go to COMPARE.
ENDFMCMP
Process any entries left in Stow Table via STOWPROC. First the entries must be moved to the beginning of the Stow Table.
16
GETCARD

Chart 16. CORGZ6F (Part 2 of 3)

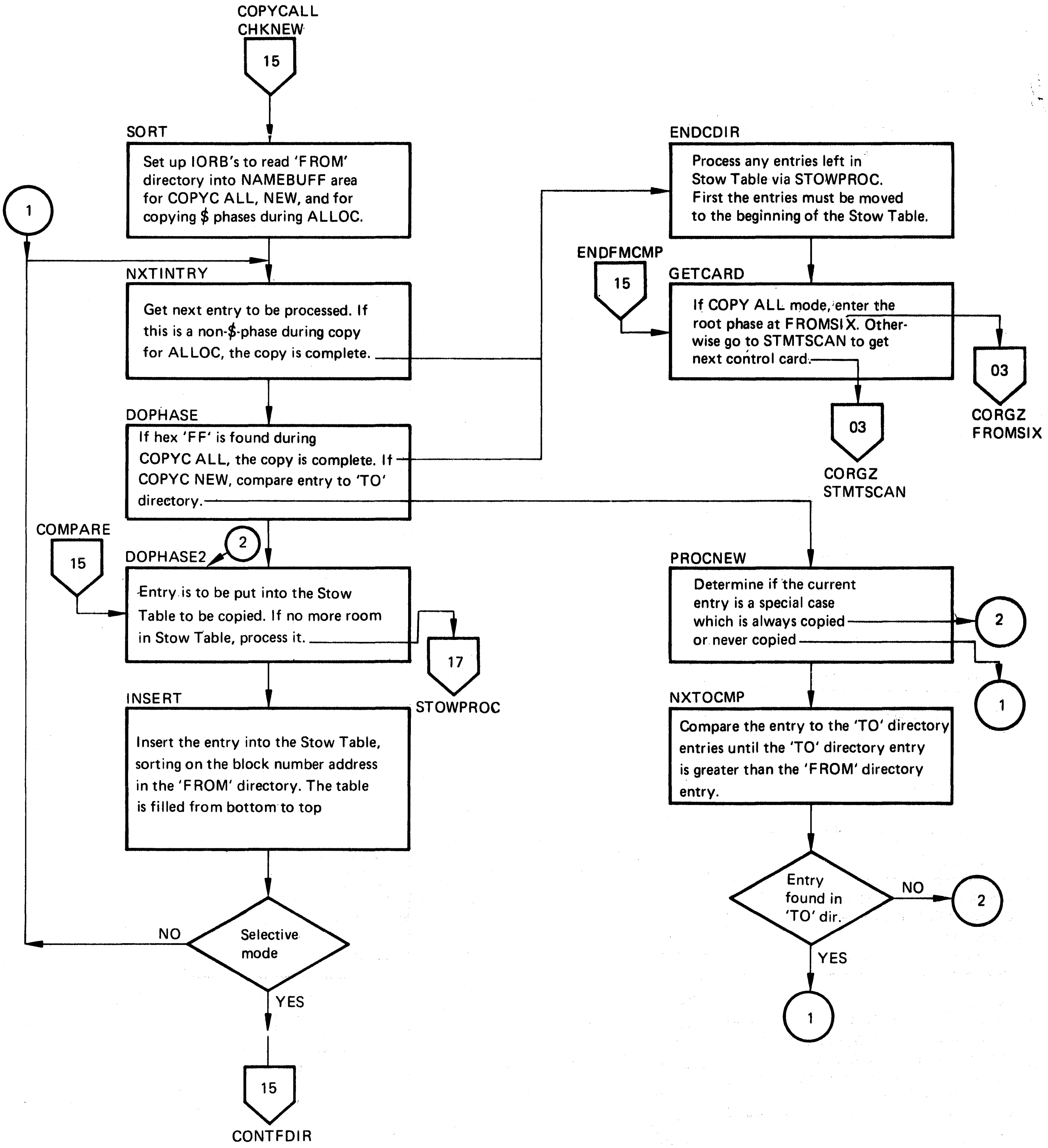

COPYCALL
CHKNEW
15
SORT
Set up IORB's to read 'FROM' directory into NAMEBUFF area for COPYC ALL, NEW, and for copying $ phases during ALLOC.
1
NXTINTRY
Get next entry to be processed. If this is a non-$-phase during copy for ALLOC, the copy is complete.
DOPHASE
If hex 'FF' is found during COPYC ALL, the copy is complete. If COPYC NEW, compare entry to 'TO' directory.
COMPARE
15
DOPHASE2
2
Entry is to be put into the Stow Table to be copied. If no more room in Stow Table, process it.
17
STOWPROC
INSERT
Insert the entry into the Stow Table, sorting on the block number address in the 'FROM' directory. The table is filled from bottom to top
NO
Selective mode
YES
15
CONTFDIR
ENDCDIR
Process any entries left in Stow Table via STOWPROC. First the entries must be moved to the beginning of the Stow Table.
ENDFMCMP
15
GETCARD
If COPY ALL mode, enter the root phase at FROMSIX. Otherwise go to STMTSCAN to get next control card.
03
CORGZ
FROMSIX
03
CORGZ
STMTSCAN
PROCNEW
Determine if the current entry is a special case which is always copied or never copied
2
1
NXTOCMP
Compare the entry to the 'TO' directory entries until the 'TO' directory entry is greater than the 'FROM' directory entry.
Entry found in 'TO' dir.
NO
2
YES
1

Chart 17. CORGZ6F (Part 3 of 3)

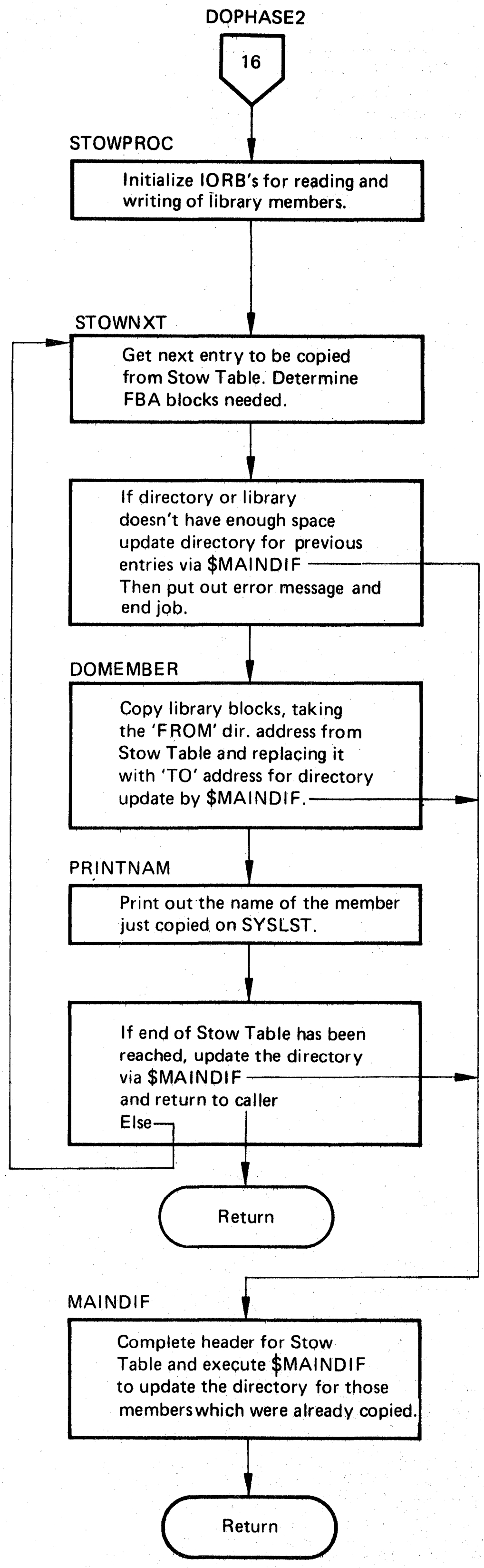

DOPHASE2
16
STOWPROC
Initialize IORB's for reading and writing of library members.
STOWNXT
Get next entry to be copied from Stow Table. Determine FBA blocks needed.
If directory or library doesn't have enough space update directory for previous entries via $MAINDIF
Then put out error message and end job.
DOMEMBER
Copy library blocks, taking the 'FROM' dir. address from Stow Table and replacing it with 'TO' address for directory update by $MAINDIF.
PRINTNAM
Print out the name of the member just copied on SYSLST.
If end of Stow Table has been reached, update the directory via $MAINDIF
and return to caller
Else
Return
MAINDIF
Complete header for Stow Table and execute $MAINDIF to update the directory for those members which were already copied.
Return

Chart 18. CORGZ7F (Part 1 of 2)

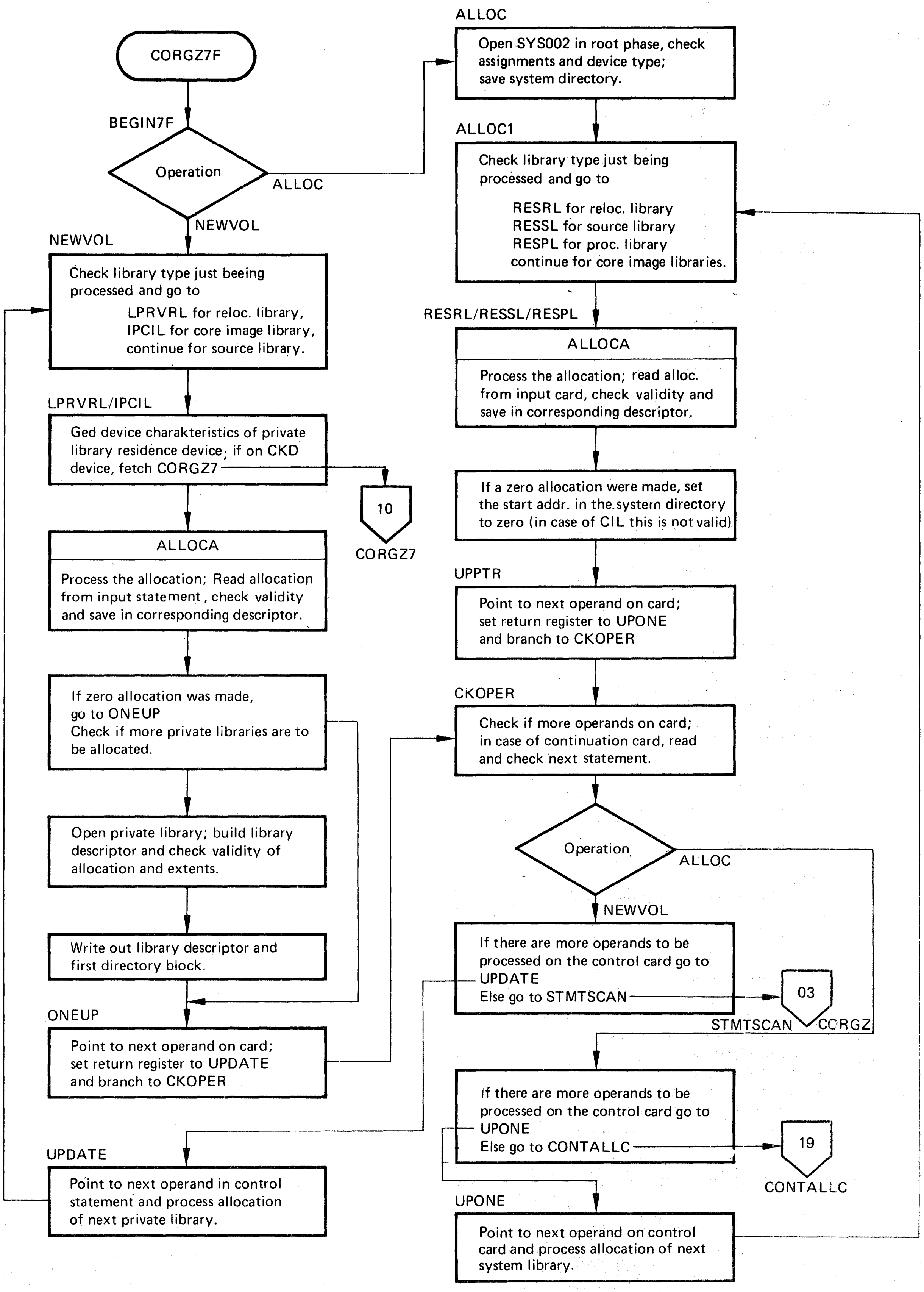

CORGZ7F
BEGIN7F
Operation
ALLOC
NEWVOL
NEWVOL
Check library type just beeing processed and go to
LPRVRL for reloc. library,
IPCIL for core image library,
continue for source library.
LPRVRL/IPCIL
Ged device charakteristics of private library residence device; if on CKD device, fetch CORGZ7
10
CORGZ7
ALLOCA
Process the allocation; Read allocation from input statement, check validity and save in corresponding descriptor.
If zero allocation was made, go to ONEUP
Check if more private libraries are to be allocated.
Open private library; build library descriptor and check validity of allocation and extents.
Write out library descriptor and first directory block.
ONEUP
Point to next operand on card; set return register to UPDATE and branch to CKOPER
UPDATE
Point to next operand in control statement and process allocation of next private library.
ALLOC
Open SYS002 in root phase, check assignments and device type; save system directory.
ALLOC1
Check library type just being processed and go to
RESRL for reloc. library
RESSL for source library
RESPL for proc. library
continue for core image libraries.
RESRL/RESSL/RESPL
ALLOCA
Process the allocation; read alloc. from input card, check validity and save in corresponding descriptor.
If a zero allocation were made, set the start addr. in the system directory to zero (in case of CIL this is not valid).
UPPTR
Point to next operand on card; set return register to UPONE and branch to CKOPER
CKOPER
Check if more operands on card; in case of continuation card, read and check next statement.
Operation
ALLOC
NEWVOL
If there are more operands to be processed on the control card go to UPDATE
Else go to STMTSCAN
03
STMTSCAN
CORGZ
If there are more operands to be processed on the control card go to UPONE
Else go to CONTALLC
19
CONTALLC
UPONE
Point to next operand on control card and process allocation of next system library.

Chart 19. CORGZ7F (Part 2 of 2)

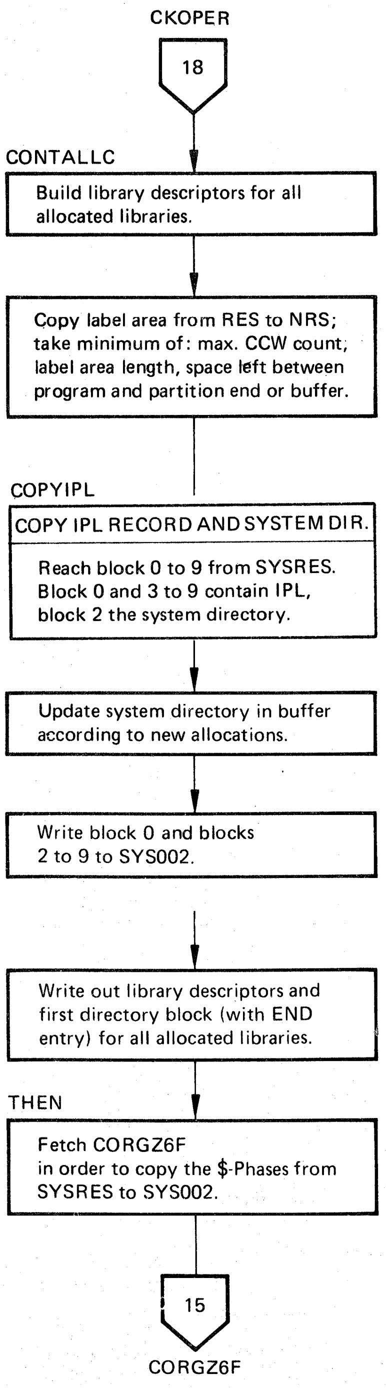

CKOPER
18
CONTALLC
Build library descriptors for all allocated libraries.
Copy label area from RES to NRS; take minimum of: max. CCW count, label area length, space left between program and partition end or buffer.
COPYIPL
COPY IPL RECORD AND SYSTEM DIR.
Reach block 0 to 9 from SYSRES. Block 0 and 3 to 9 contain IPL, block 2 the system directory.
Update system directory in buffer according to new allocations.
Write block 0 and blocks 2 to 9 to SYS002.
Write out library descriptors and first directory block (with END entry) for all allocated libraries.
THEN
Fetch CORGZ6F in order to copy the $-Phases from SYSRES to SYS002.
15
CORGZ6F

Chart 20. MAINT - Root Phase (Part 1 of 4)

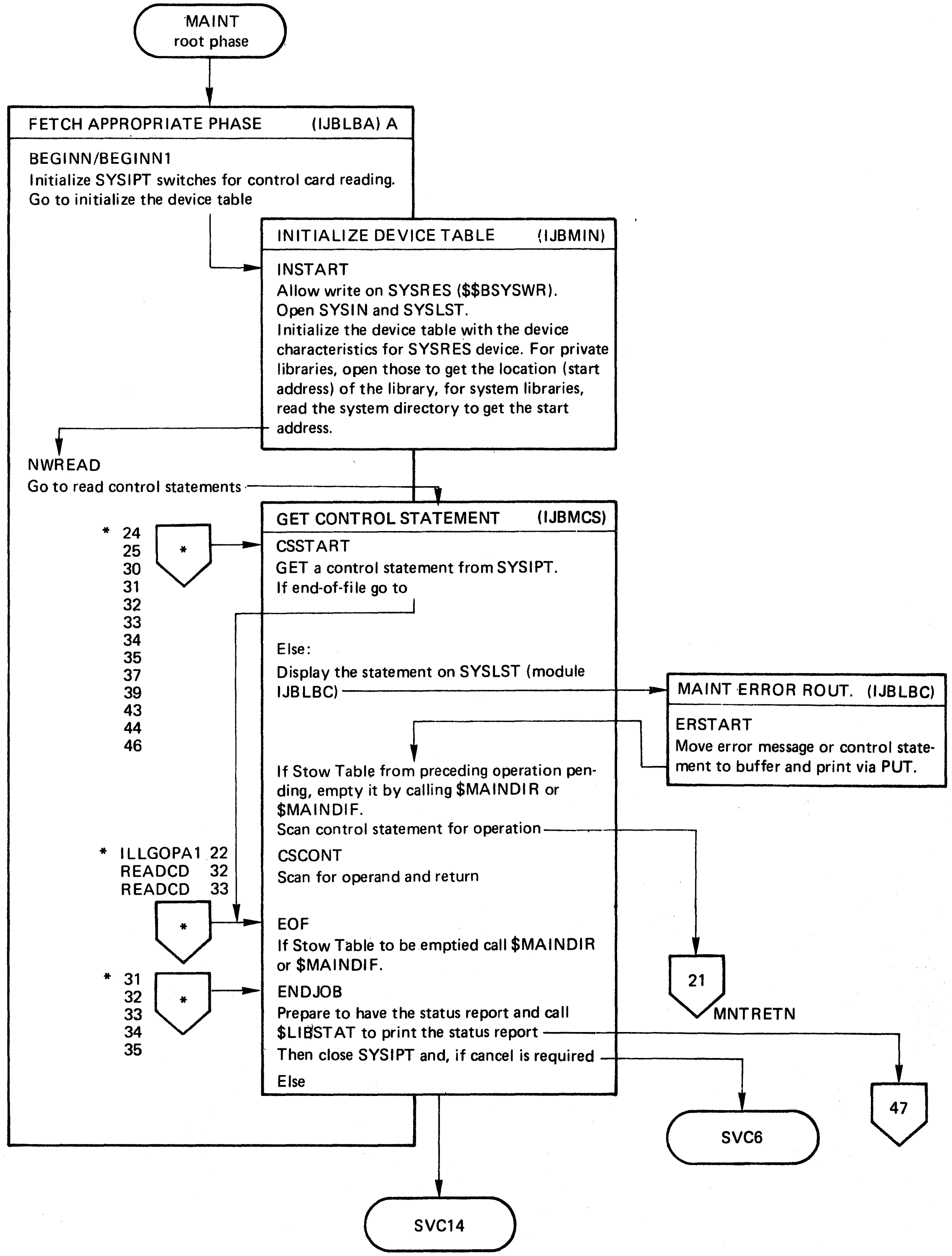
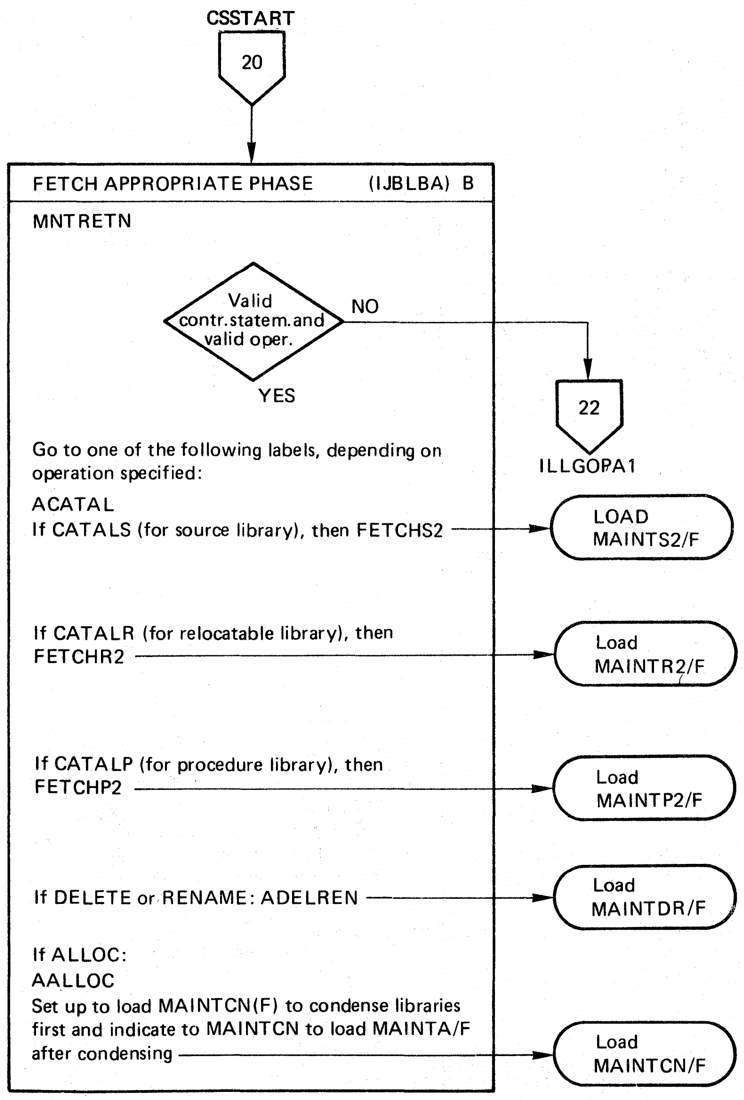

MAINT
root phase
FETCH APPROPRIATE PHASE (IJBLBA) A
BEGINN/BEGINN1
Initialize SYSIPT switches for control card reading.
Go to initialize the device table
INITIALIZE DEVICE TABLE (IJBMIN)
INSTART
Allow write on SYSRES ($$BSYSWR).
Open SYSIN and SYSLST.
Initialize the device table with the device characteristics for SYSRES device. For private libraries, open those to get the location (start address) of the library, for system libraries, read the system directory to get the start address.
NWREAD
Go to read control statements
GET CONTROL STATEMENT (IJBMCS)
* 24 25 30 31 32 33 34 35 37 39 43 44 46
*
CSSTART
GET a control statement from SYSIPT.
If end-of-file go to
Else:
Display the statement on SYSLST (module IJBLBC)
MAINT ERROR ROUT. (IJBLBC)
ERSTART
Move error message or control statement to buffer and print via PUT.
If Stow Table from preceding operation pending, empty it by calling $MAINDIR or $MAINDIF.
Scan control statement for operation
* ILLGOPA1 22
READCD 32
READCD 33
CSCONT
Scan for operand and return
*
EOF
If Stow Table to be emptied call $MAINDIR or $MAINDIF.
21
MNTRETN
* 31 32 33 34 35
*
ENDJOB
Prepare to have the status report and call $LIBSTAT to print the status report
Then close SYSIPT and, if cancel is required
Else
47
SVC6
SVC14

Chart 21. MAINT - Root Phase (Part 2 of 4)


CSSTART
20
FETCH APPROPRIATE PHASE (IJBLBA) B
MNTRETN
Valid contr.statem.and valid oper.
NO
YES
22
ILLGOPA1
Go to one of the following labels, depending on operation specified:
ACATAL
If CATALS (for source library), then FETCHS2
LOAD MAINTS2/F
If CATALR (for relocatable library), then FETCHR2
Load MAINTR2/F
If CATALP (for procedure library), then FETCHP2
Load MAINTP2/F
If DELETE or RENAME: ADELREN
Load MAINTDR/F
If ALLOC:
AALLOC
Set up to load MAINTCN(F) to condense libraries first and indicate to MAINTCN to load MAINTA/F after condensing
Load MAINTCN/F

Notes:

1. These phases are only loaded if they are not present from an earlier operation.
2. Loading is done via routine LDMNTPH1

Chart 22. MAINT - Root Phase (Part 3 of 4)

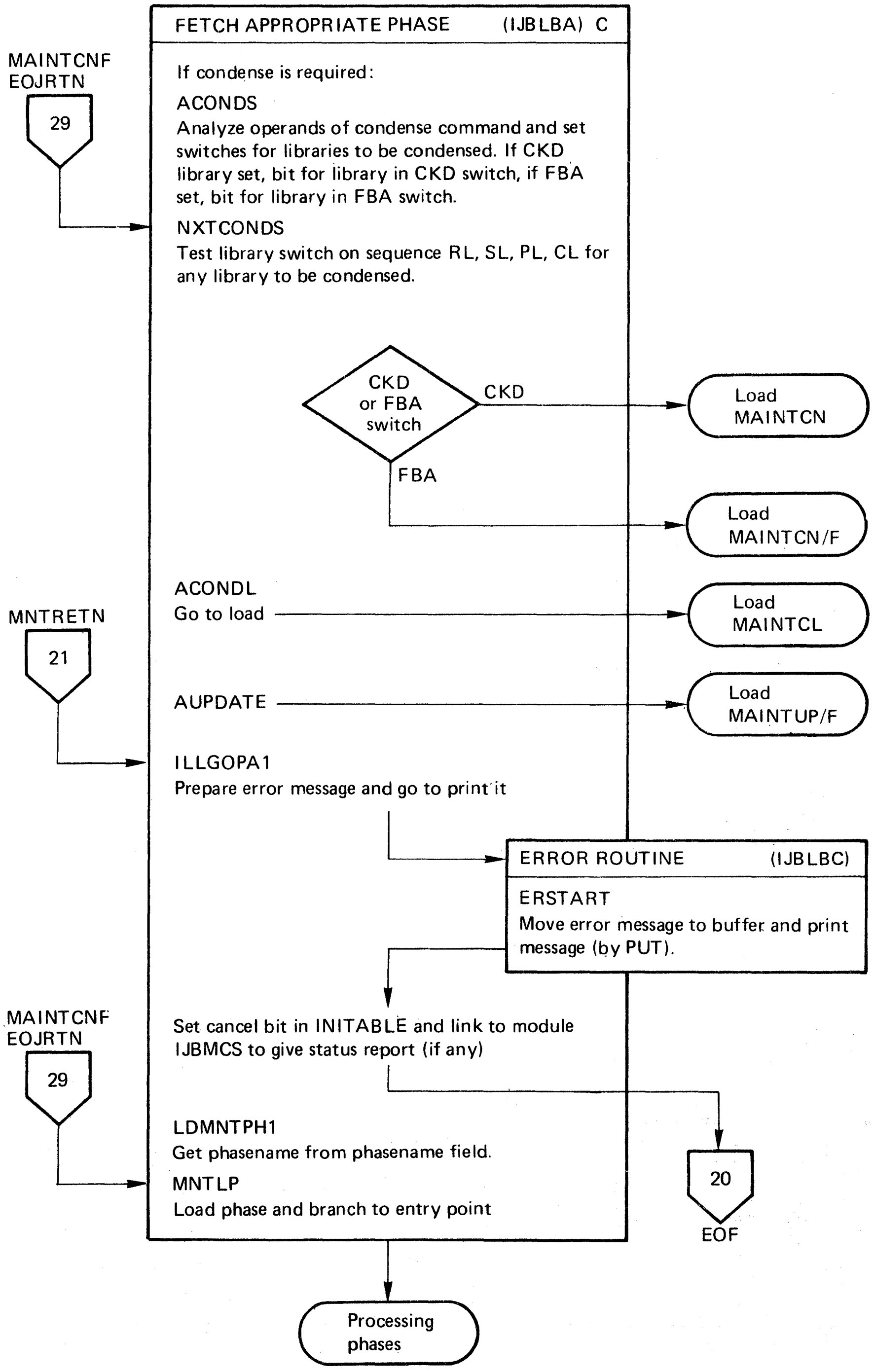

FETCH APPROPRIATE PHASE (IJBLBA) C
MAINTCNF
EOJRTN
29
If condense is required:
ACONDS
Analyze operands of condense command and set switches for libraries to be condensed. If CKD library set, bit for library in CKD switch, if FBA set, bit for library in FBA switch.
NXTCONDS
Test library switch on sequence RL, SL, PL, CL for any library to be condensed.
CKD or FBA switch
CKD
Load MAINTCN
FBA
Load MAINTCN/F
ACONDL
Go to load
Load MAINTCL
MNTRETN
21
AUPDATE
Load MAINTUP/F
ILLGOPA1
Prepare error message and go to print it
ERROR ROUTINE (IJBLBC)
ERSTART
Move error message to buffer and print message (by PUT).
Set cancel bit in INITABLE and link to module IJBMCS to give status report (if any)
MAINTCNF
EOJRTN
29
LDMNTPH1
Get phasename from phasename field.
MNTLP
Load phase and branch to entry point
20
EOF
Processing phases

Chart 23. MAINT - Root Phase (Part 4 of 4)

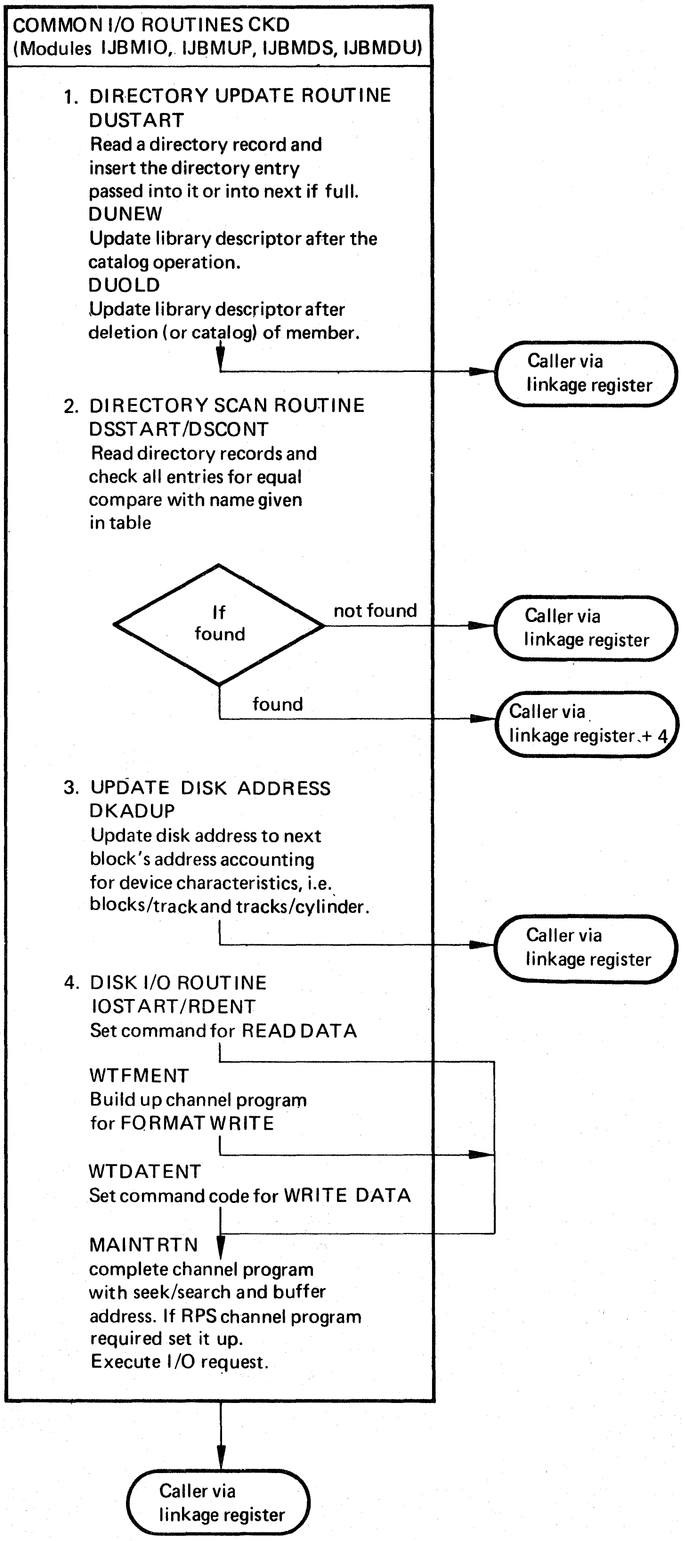

COMMON I/O ROUTINES CKD
(Modules IJBMIO, IJBMUP, IJBMDS, IJBMDU)
1. DIRECTORY UPDATE ROUTINE
DUSTART
Read a directory record and insert the directory entry passed into it or into next if full.
DUNEW
Update library descriptor after the catalog operation.
DUOLD
Update library descriptor after deletion (or catalog) of member.
Caller via linkage register
2. DIRECTORY SCAN ROUTINE
DSSTART/DSCONT
Read directory records and check all entries for equal compare with name given in table
If found
not found
Caller via linkage register
found
Caller via linkage register + 4
3. UPDATE DISK ADDRESS
DKADUP
Update disk address to next block's address accounting for device characteristics, i.e. blocks/track and tracks/cylinder.
Caller via linkage register
4. DISK I/O ROUTINE
IOSTART/RDENT
Set command for READ DATA
WTFMENT
Build up channel program for FORMAT WRITE
WTDATENT
Set command code for WRITE DATA
MAINTRTN
complete channel program with seek/search and buffer address. If RPS channel program required set it up.
Execute I/O request.
Caller via linkage register

Chart 24. MAINTCL (Detail Charts EX-FA)

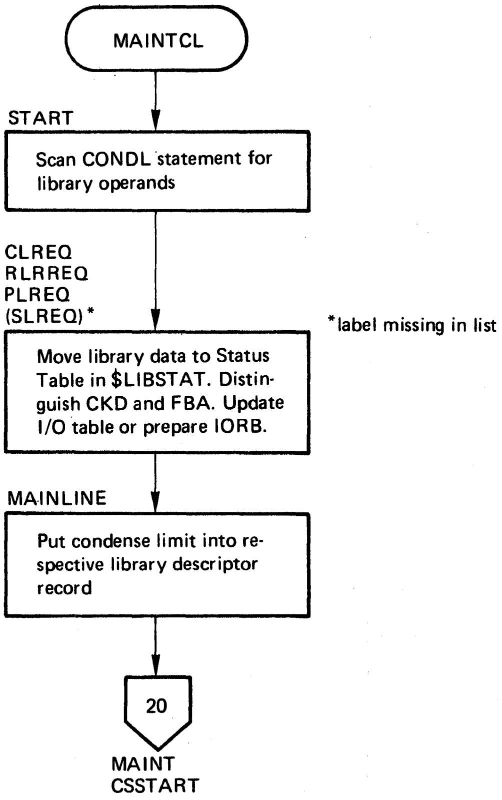

MAINTCL
START
Scan CONDL statement for library operands
CLREQ
RLRREQ
PLREQ
(SLREQ)*
*label missing in list
Move library data to Status Table in $LIBSTAT. Distinguish CKD and FBA. Update I/O table or prepare IORB.
MAINLINE
Put condense limit into respective library descriptor record
20
MAINT
CSSTART

Chart 25. MAINTCN (Detail Charts FD-FM)

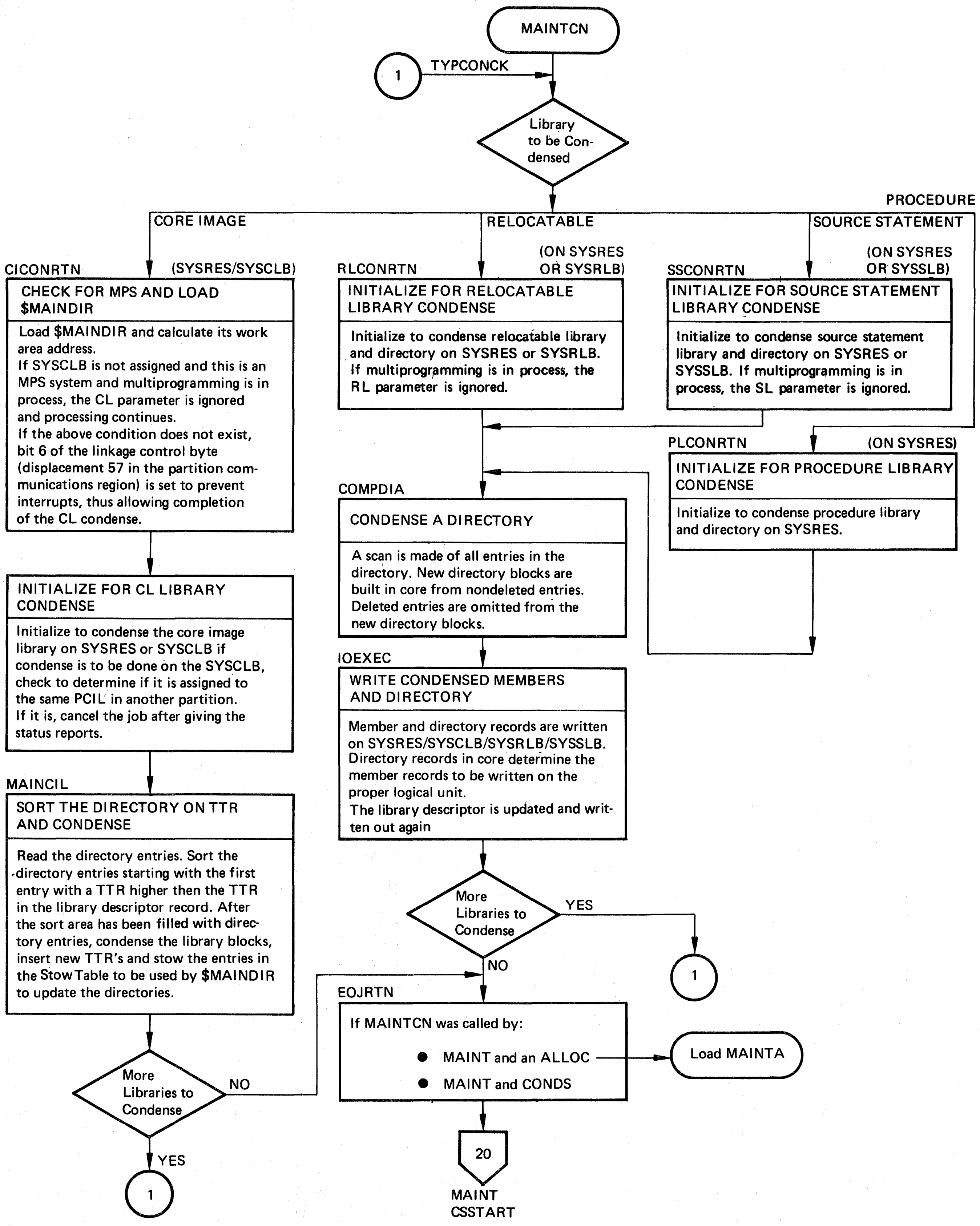
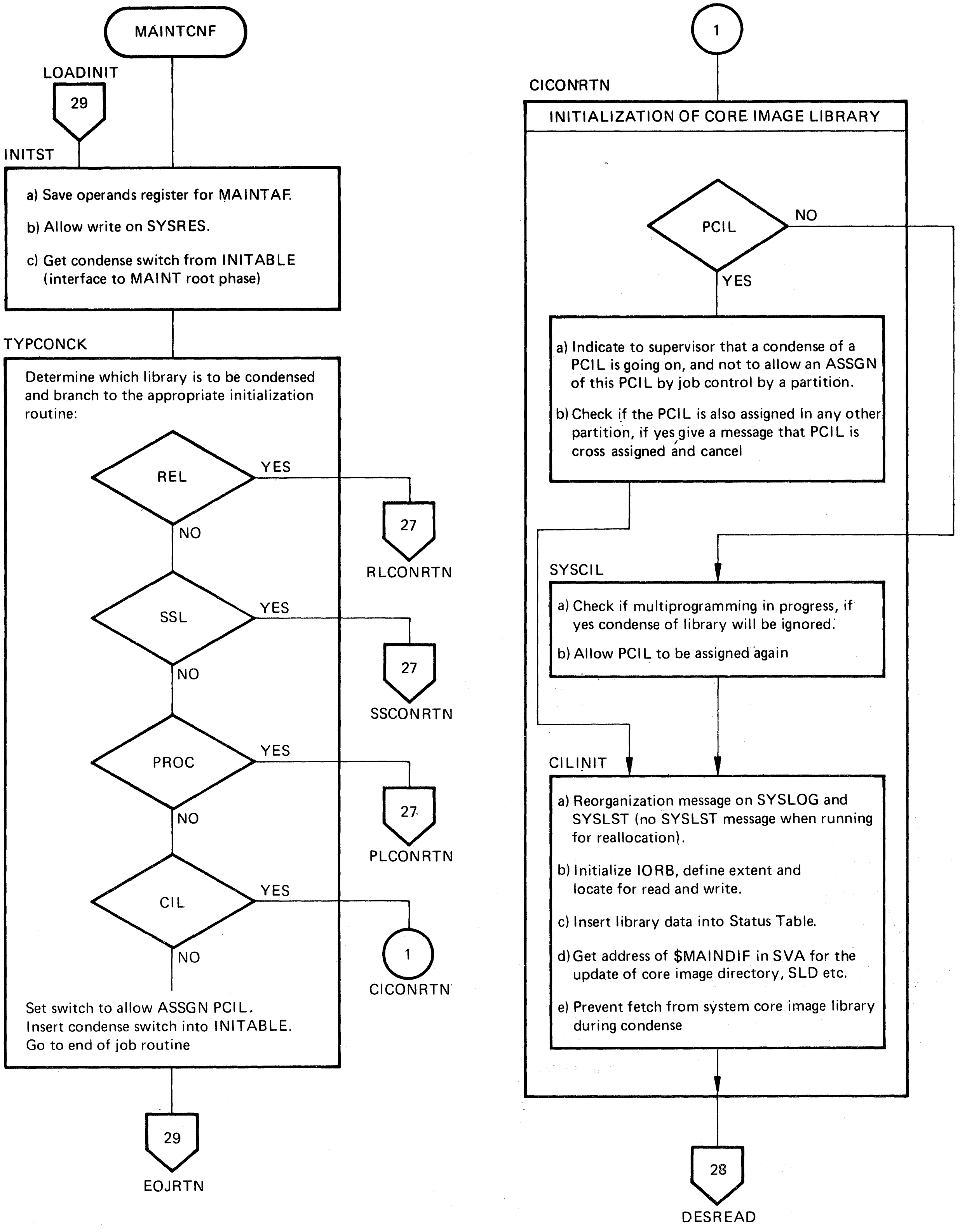

MAINTCN
1
TYPCONCK
Library to be Condensed
CORE IMAGE
RELOCATABLE
PROCEDURE
SOURCE STATEMENT
CICONRTN
(SYSRES/SYSCLB)
CHECK FOR MPS AND LOAD $MAINDIR
Load $MAINDIR and calculate its work area address.
If SYSCLB is not assigned and this is an MPS system and multiprogramming is in process, the CL parameter is ignored and processing continues.
If the above condition does not exist, bit 6 of the linkage control byte (displacement 57 in the partition communications region) is set to prevent interrupts, thus allowing completion of the CL condense.
RLCONRTN
(ON SYSRES OR SYSRLB)
INITIALIZE FOR RELOCATABLE LIBRARY CONDENSE
Initialize to condense relocatable library and directory on SYSRES or SYSRLB. If multiprogramming is in process, the RL parameter is ignored.
SSCONRTN
(ON SYSRES OR SYSSLB)
INITIALIZE FOR SOURCE STATEMENT LIBRARY CONDENSE
Initialize to condense source statement library and directory on SYSRES or SYSSLB. If multiprogramming is in process, the SL parameter is ignored.
PLCONRTN
(ON SYSRES)
INITIALIZE FOR PROCEDURE LIBRARY CONDENSE
Initialize to condense procedure library and directory on SYSRES.
COMPDIA
CONDENSE A DIRECTORY
A scan is made of all entries in the directory. New directory blocks are built in core from nondeleted entries. Deleted entries are omitted from the new directory blocks.
INITIALIZE FOR CL LIBRARY CONDENSE
Initialize to condense the core image library on SYSRES or SYSCLB if condense is to be done on the SYSCLB, check to determine if it is assigned to the same PCIL in another partition.
If it is, cancel the job after giving the status reports.
IOEXEC
WRITE CONDENSED MEMBERS AND DIRECTORY
Member and directory records are written on SYSRES/SYSCLB/SYSRLB/SYSSLB.
Directory records in core determine the member records to be written on the proper logical unit.
The library descriptor is updated and written out again
MAINCIL
SORT THE DIRECTORY ON TTR AND CONDENSE
Read the directory entries. Sort the directory entries starting with the first entry with a TTR higher then the TTR in the library descriptor record. After the sort area has been filled with directory entries, condense the library blocks, insert new TTR's and stow the entries in the StowTable to be used by $MAINDIR to update the directories.
More Libraries to Condense
YES
1
NO
EOJRTN
If MAINTCN was called by:
● MAINT and an ALLOC
● MAINT and CONDS
Load MAINTA
More Libraries to Condense
NO
YES
1
20
MAINT
CSSTART

Chart 26. MAINTCNF (Part 1 of 4)



a) Save operands register for MAINTAF.

b) Allow write on SYSRES.

c) Get condense switch from INITABLE (interface to MAINT root phase)

Determine which library is to be condensed and branch to the appropriate initialization routine:

Set switch to allow ASSGN PCIL.
Insert condense switch into INITABLE.
Go to end of job routine

INITIALIZATION OF CORE IMAGE LIBRARY

a) Indicate to supervisor that a condense of a PCIL is going on, and not to allow an ASSGN of this PCIL by job control by a partition.

b) Check if the PCIL is also assigned In any other partition, if yes give a message that PCIL is cross assigned and cancel

a) Check if multiprogramming in progress, if yes condense of library will be ignored.

b) Allow PCIL to be assigned again

a) Reorganization message on SYSLOG and SYSLST (no SYSLST message when running for reallocation).

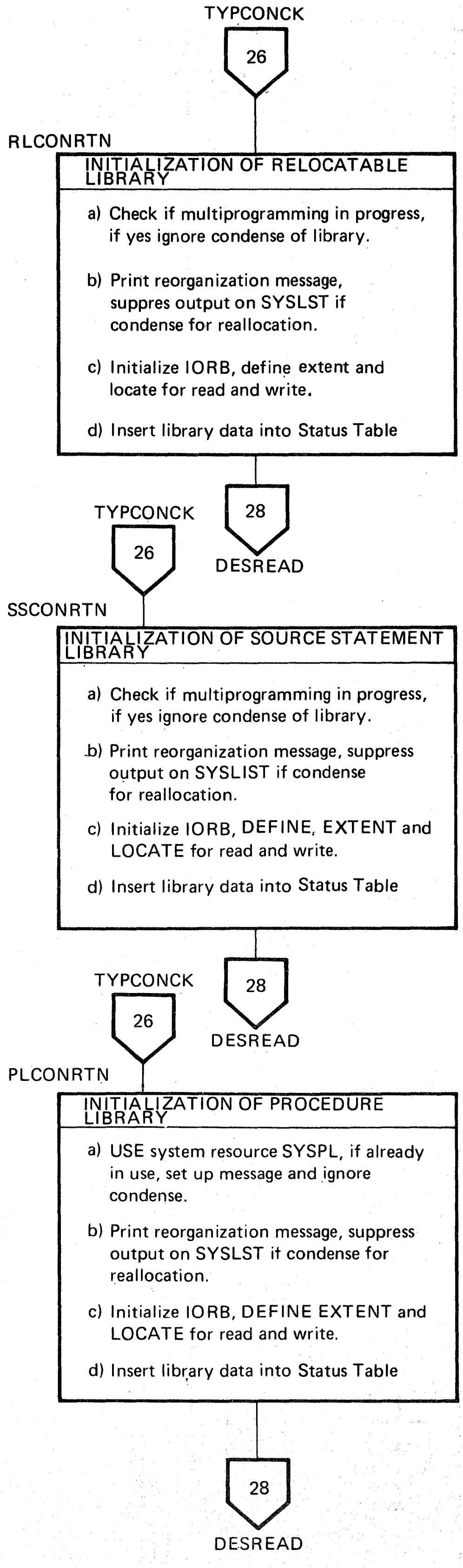
b) Initialize IORB, define extent and locate for read and write.

c) Insert library data into Status Table.

d) Get address of $MAINDIF in SVA for the update of core image directory, SLD etc.

e) Prevent fetch from system core image library during condense

Chart 27. MAINTCNF (Part 2 of 4)



INITIALIZATION OF RELOCATABLE LIBRARY

a) Check if multiprogramming in progress, if yes ignore condense of library.

b) Print reorganization message, suppres output on SYSLST if condense for reallocation.

c) Initialize IORB, define extent and locate for read and write.

d) Insert library data into Status Table

INITIALIZATION OF SOURCE STATEMENT LIBRARY

a) Check if multiprogramming in progress, if yes ignore condense of library.

b) Print reorganization message, suppress output on SYSLIST if condense for reallocation.

c) Initialize IORB, DEFINE, EXTENT and LOCATE for read and write.

d) Insert library data into Status Table

INITIALIZATION OF PROCEDURE LIBRARY

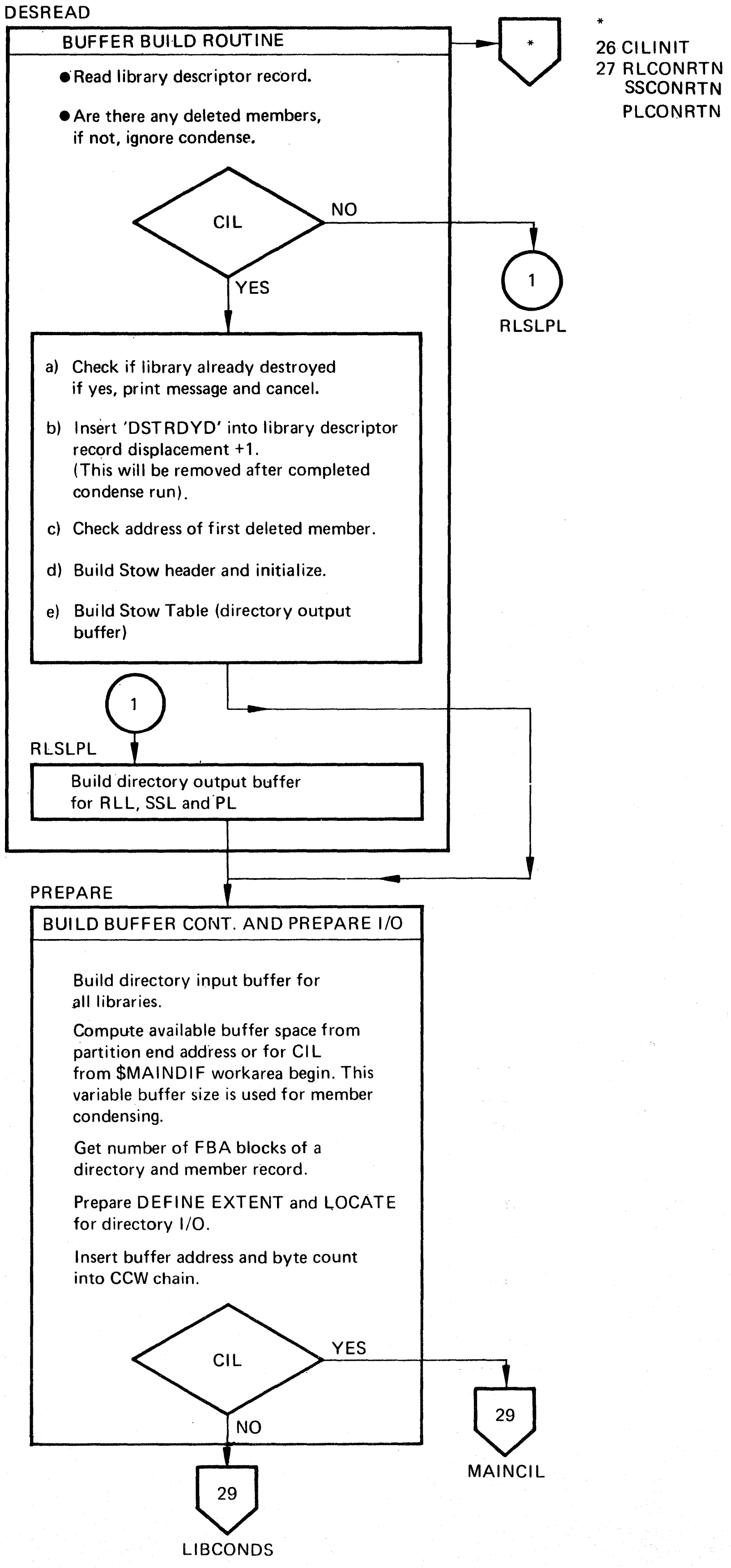
a) USE system resource SYSPL, if already in use, set up message and ignore condense.

b) Print reorganization message, suppress output on SYSLST if condense for reallocation.

c) Initialize IORB, DEFINE EXTENT and LOCATE for read and write.

d) Insert library data into Status Table

Chart 28. MAINTCNF (Part 3 of 4)



- Read library descriptor record.
- Are there any deleted members, if not, ignore condense.

a) Check if library already destroyed if yes, print message and cancel.

b) Insert 'DSTRDYD' into library descriptor record displacement +1. (This will be removed after completed condense run).

c) Check address of first deleted member.

d) Build Stow header and initialize.

e) Build Stow Table (directory output buffer)

Build directory output buffer for RLL, SSL and PL

Build directory input buffer for all libraries.

Compute available buffer space from partition end address or for CIL from $MAINDIF workarea begin. This variable buffer size is used for member condensing.

Get number of FBA blocks of a directory and member record.

Prepare DEFINE EXTENT and LOCATE for directory I/O.

Insert buffer address and byte count into CCW chain.

Chart 29. MAINTCNF (Part 4 of 4)

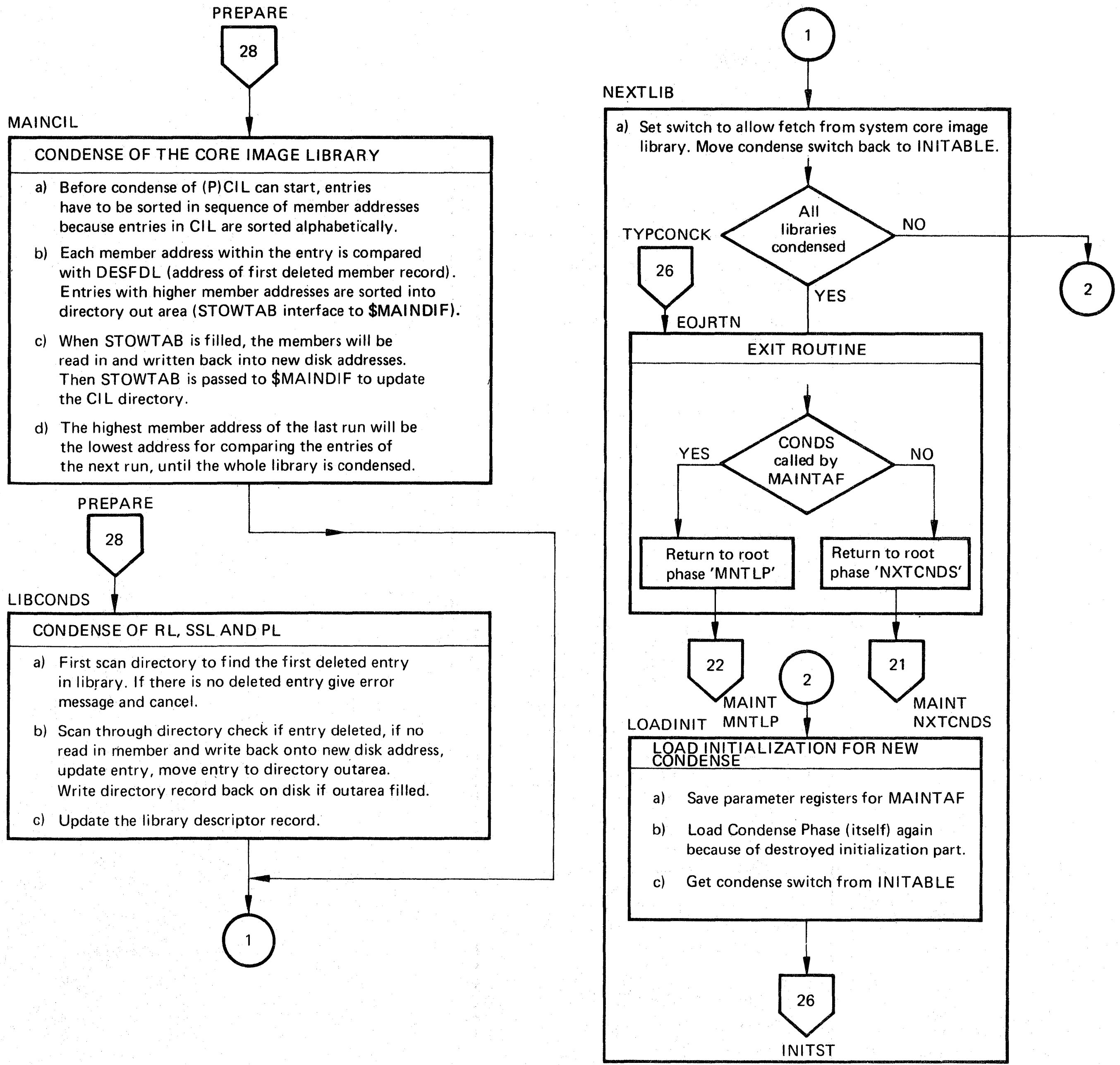
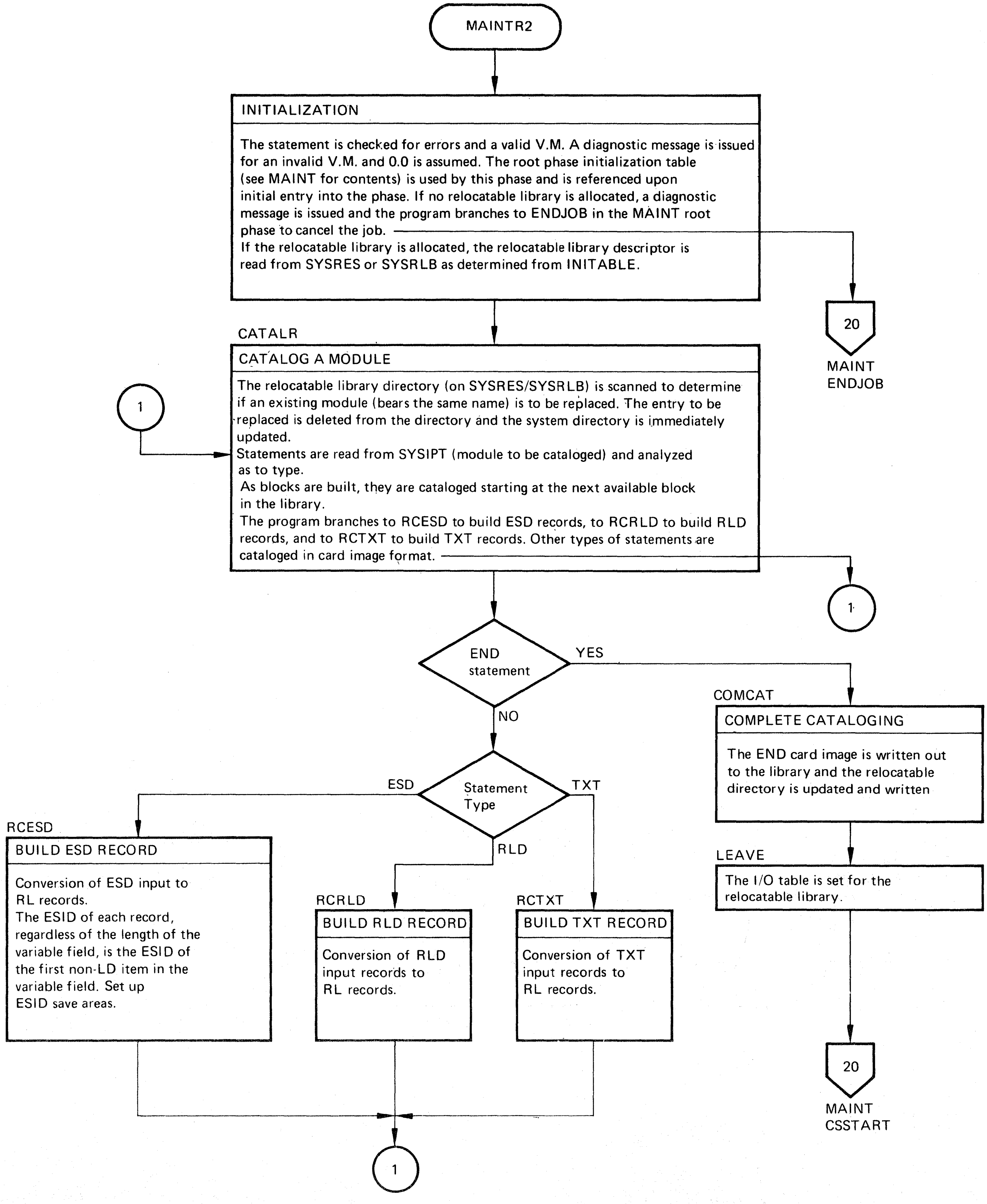

PREPARE
28
MAINCIL
CONDENSE OF THE CORE IMAGE LIBRARY
a) Before condense of (P)CIL can start, entries have to be sorted in sequence of member addresses because entries in CIL are sorted alphabetically.
b) Each member address within the entry is compared with DESFDL (address of first deleted member record). Entries with higher member addresses are sorted into directory out area (STOWTAB interface to $MAINDIF).
c) When STOWTAB is filled, the members will be read in and written back into new disk addresses. Then STOWTAB is passed to $MAINDIF to update the CIL directory.
d) The highest member address of the last run will be the lowest address for comparing the entries of the next run, until the whole library is condensed.
PREPARE
28
LIBCONDS
CONDENSE OF RL, SSL AND PL
a) First scan directory to find the first deleted entry in library. If there is no deleted entry give error message and cancel.
b) Scan through directory check if entry deleted, if no read in member and write back onto new disk address, update entry, move entry to directory outarea. Write directory record back on disk if outarea filled.
c) Update the library descriptor record.
1
1
NEXTLIB
a) Set switch to allow fetch from system core image library. Move condense switch back to INITABLE.
TYPCONCK
All libraries condensed
NO
2
YES
26
EOJRTN
EXIT ROUTINE
YES
CONDS called by MAINTAF
NO
Return to root phase 'MNTLP'
Return to root phase 'NXTCNDS'
22
MAINT MNTLP
2
21
MAINT NXTCNDS
LOADINIT
LOAD INITIALIZATION FOR NEW CONDENSE
a) Save parameter registers for MAINTAF
b) Load Condense Phase (itself) again because of destroyed initialization part.
c) Get condense switch from INITABLE
26
INITST

Chart 30. MAINTR2



The statement is checked for errors and a valid V.M. A diagnostic message is issued for an invalid V.M. and 0.0 is assumed. The root phase initialization table (see MAINT for contents) is used by this phase and is referenced upon initial entry into the phase. If no relocatable library is allocated, a diagnostic message is issued and the program branches to ENDJOB in the MAINT root phase to cancel the job.

If the relocatable library is allocated, the relocatable library descriptor is read from SYSRES or SYSRLB as determined from INITABLE.

The relocatable library directory (on SYSRES/SYSRLB) is scanned to determine if an existing module (bears the same name) is to be replaced. The entry to be replaced is deleted from the directory and the system directory is immediately updated.

Statements are read from SYSIPT (module to be cataloged) and analyzed as to type.

As blocks are built, they are cataloged starting at the next available block in the library.

The program branches to RCESD to build ESD records, to RCRLD to build RLD records, and to RCTXT to build TXT records. Other types of statements are cataloged in card image format.

The END card image is written out to the library and the relocatable directory is updated and written

The I/O table is set for the relocatable library.

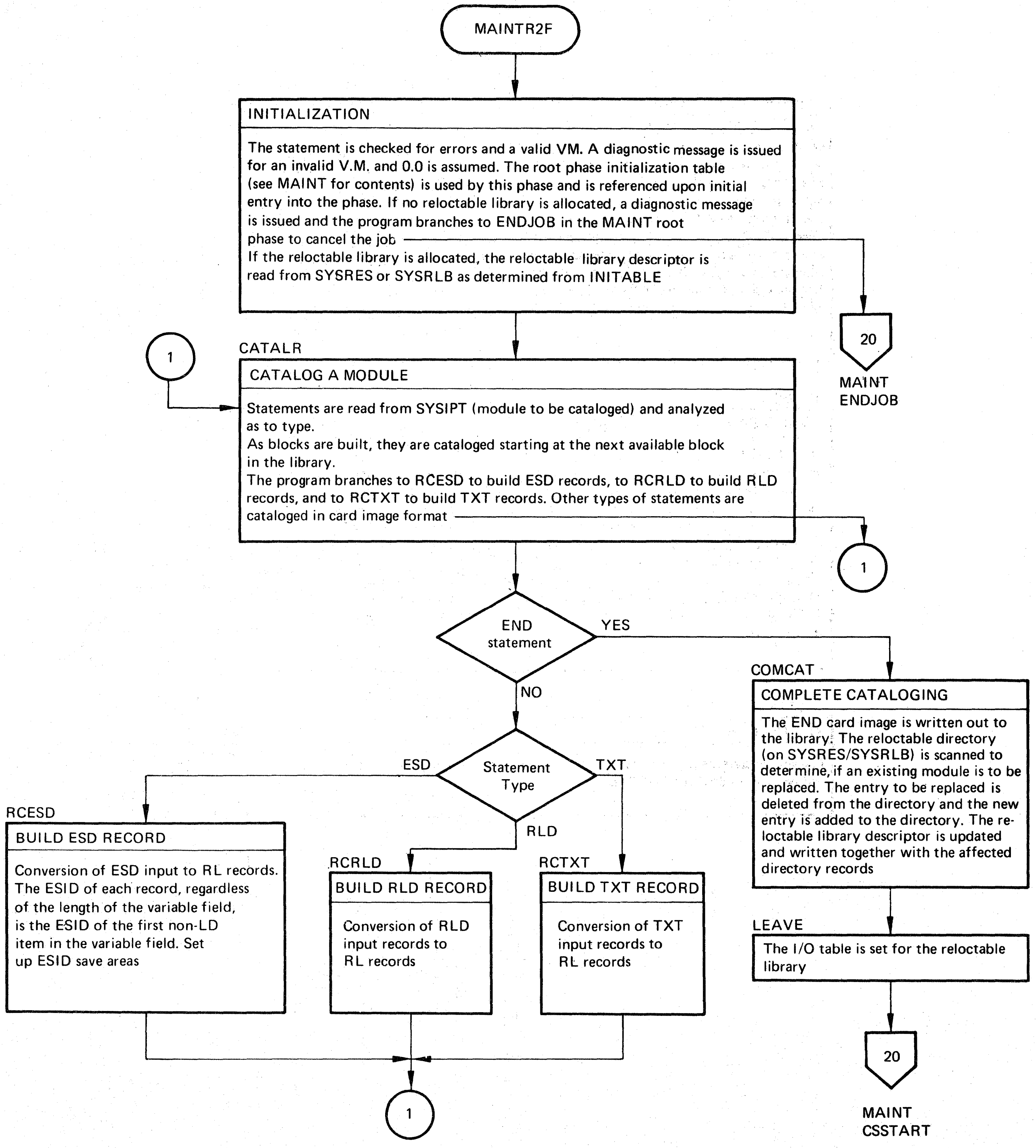
Conversion of ESD input to RL records.

The ESID of each record, regardless of the length of the variable field, is the ESID of the first non-LD item in the variable field. Set up ESID save areas.

Conversion of RLD input records to RL records.

Conversion of TXT input records to RL records.

Chart 31. MAINTR2F



INITIALIZATION

The statement is checked for errors and a valid VM. A diagnostic message is issued for an invalid V.M. and 0.0 is assumed. The root phase initialization table (see MAINT for contents) is used by this phase and is referenced upon initial entry into the phase. If no reloctable library is allocated, a diagnostic message is issued and the program branches to ENDJOB in the MAINT root phase to cancel the job

If the reloctable library is allocated, the reloctable library descriptor is read from SYSRES or SYSRLB as determined from INITABLE

CATALOG A MODULE

Statements are read from SYSIPT (module to be cataloged) and analyzed as to type.

As blocks are built, they are cataloged starting at the next available block in the library.

The program branches to RCESD to build ESD records, to RCRLD to build RLD records, and to RCTXT to build TXT records. Other types of statements are cataloged in card image format

COMPLETE CATALOGING

The END card image is written out to the library. The reloctable directory (on SYSRES/SYSRLB) is scanned to determine, if an existing module is to be replaced. The entry to be replaced is deleted from the directory and the new entry is added to the directory. The reloctable library descriptor is updated and written together with the affected directory records

BUILD ESD RECORD

Conversion of ESD input to RL records. The ESID of each record, regardless of the length of the variable field, is the ESID of the first non-LD item in the variable field. Set up ESID save areas

BUILD RLD RECORD

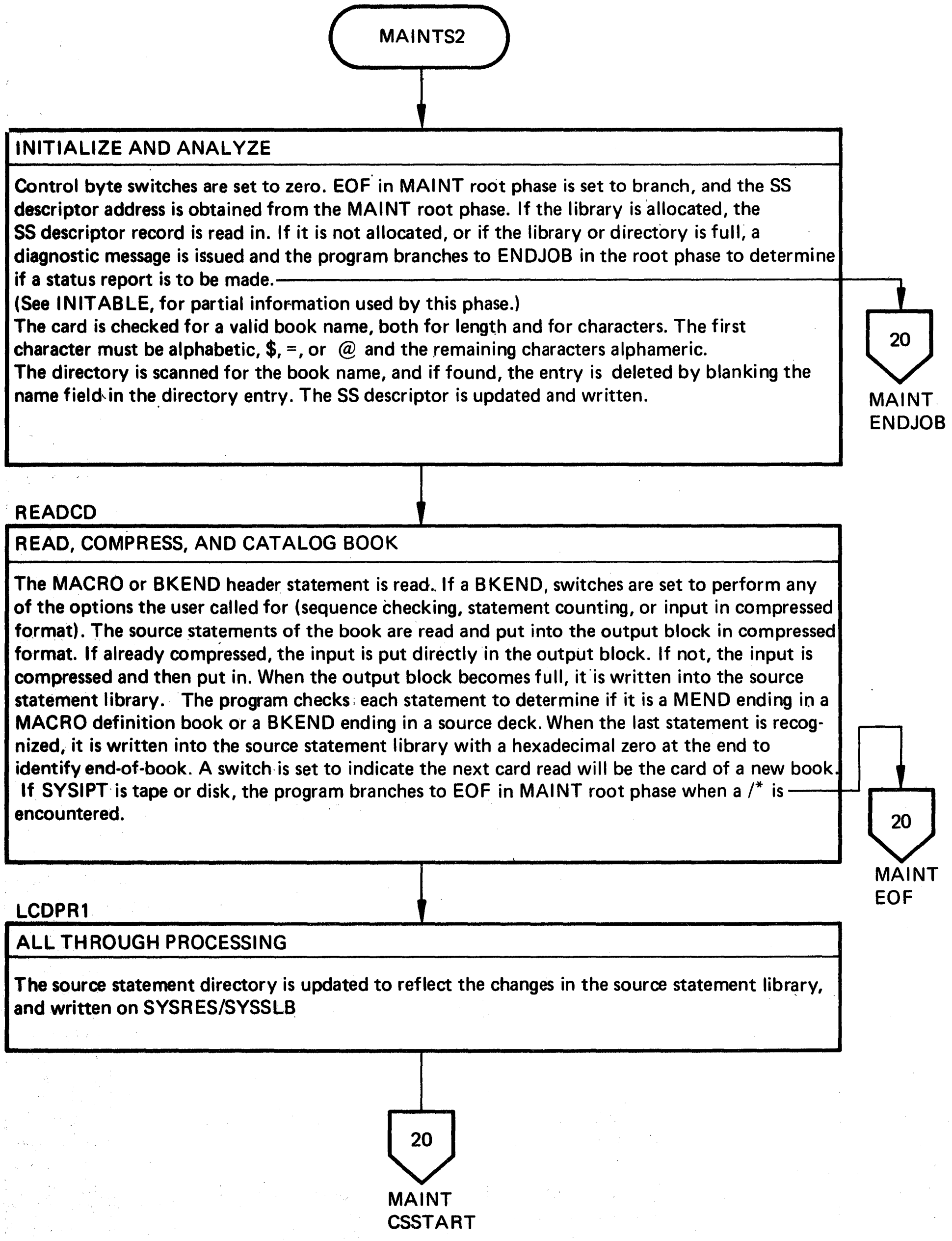
Conversion of RLD input records to RL records

BUILD TXT RECORD

Conversion of TXT input records to RL records

The I/O table is set for the reloctable library

Chart 32. MAINTS2



INITIALIZE AND ANALYZE

Control byte switches are set to zero. EOF in MAINT root phase is set to branch, and the SS descriptor address is obtained from the MAINT root phase. If the library is allocated, the SS descriptor record is read in. If it is not allocated, or if the library or directory is full, a diagnostic message is issued and the program branches to ENDJOB in the root phase to determine if a status report is to be made.
(See INITABLE, for partial information used by this phase.)
The card is checked for a valid book name, both for length and for characters. The first character must be alphabetic, $, =, or @ and the remaining characters alphameric.
The directory is scanned for the book name, and if found, the entry is deleted by blanking the name field in the directory entry. The SS descriptor is updated and written.

READCD

READ, COMPRESS, AND CATALOG BOOK

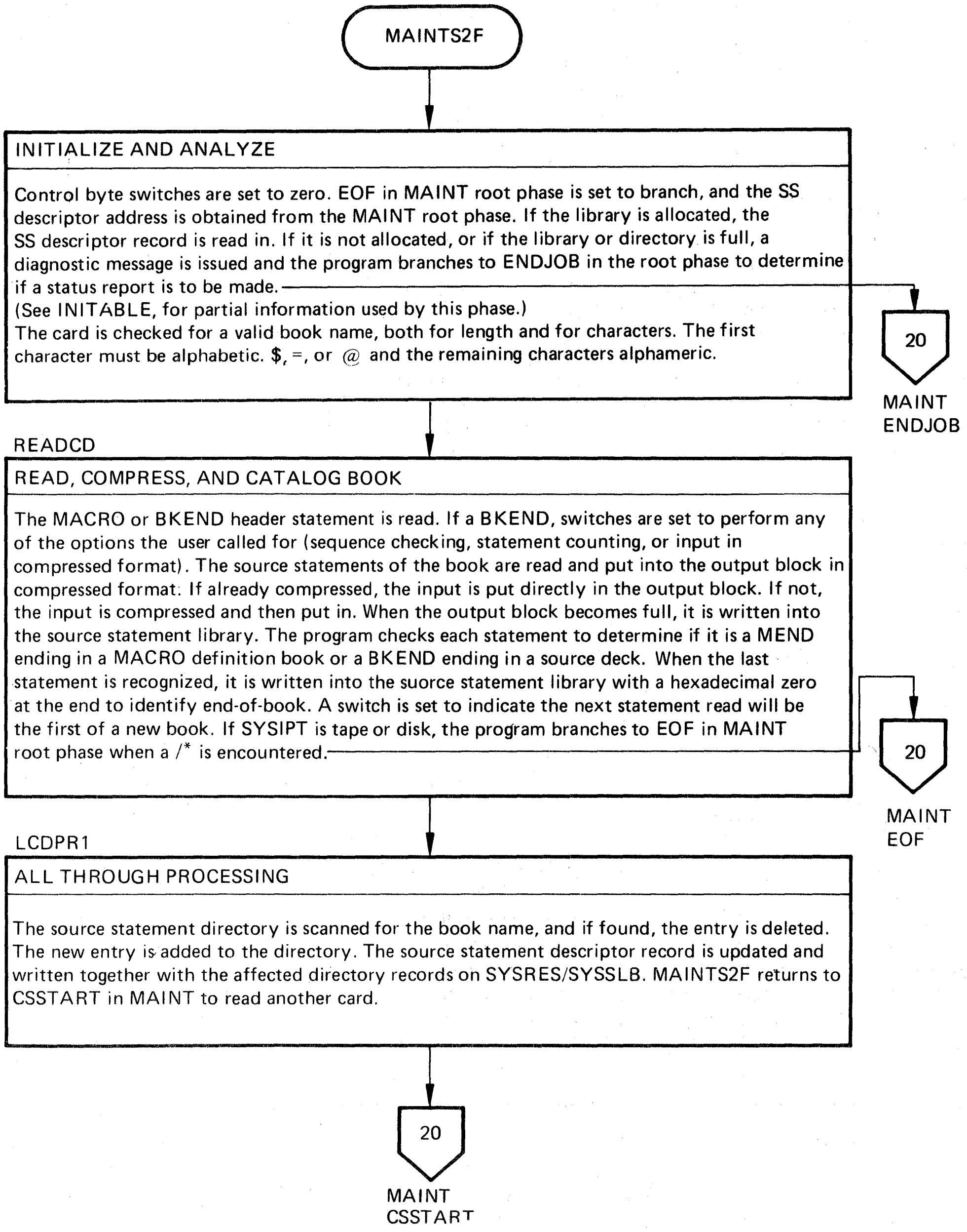
The MACRO or BKEND header statement is read. If a BKEND, switches are set to perform any of the options the user called for (sequence checking, statement counting, or input in compressed format). The source statements of the book are read and put into the output block in compressed format. If already compressed, the input is put directly in the output block. If not, the input is compressed and then put in. When the output block becomes full, it is written into the source statement library. The program checks each statement to determine if it is a MEND ending in a MACRO definition book or a BKEND ending in a source deck. When the last statement is recognized, it is written into the source statement library with a hexadecimal zero at the end to identify end-of-book. A switch is set to indicate the next card read will be the card of a new book. If SYSIPT is tape or disk, the program branches to EOF in MAINT root phase when a /* is encountered.

LCDPR1

ALL THROUGH PROCESSING

The source statement directory is updated to reflect the changes in the source statement library, and written on SYSRES/SYSSLB

Chart 33. MAINTS2F



INITIALIZE AND ANALYZE

Control byte switches are set to zero. EOF in MAINT root phase is set to branch, and the SS descriptor address is obtained from the MAINT root phase. If the library is allocated, the SS descriptor record is read in. If it is not allocated, or if the library or directory is full, a diagnostic message is issued and the program branches to ENDJOB in the root phase to determine if a status report is to be made.
(See INITABLE, for partial information used by this phase.)
The card is checked for a valid book name, both for length and for characters. The first character must be alphabetic. $, =, or @ and the remaining characters alphameric.

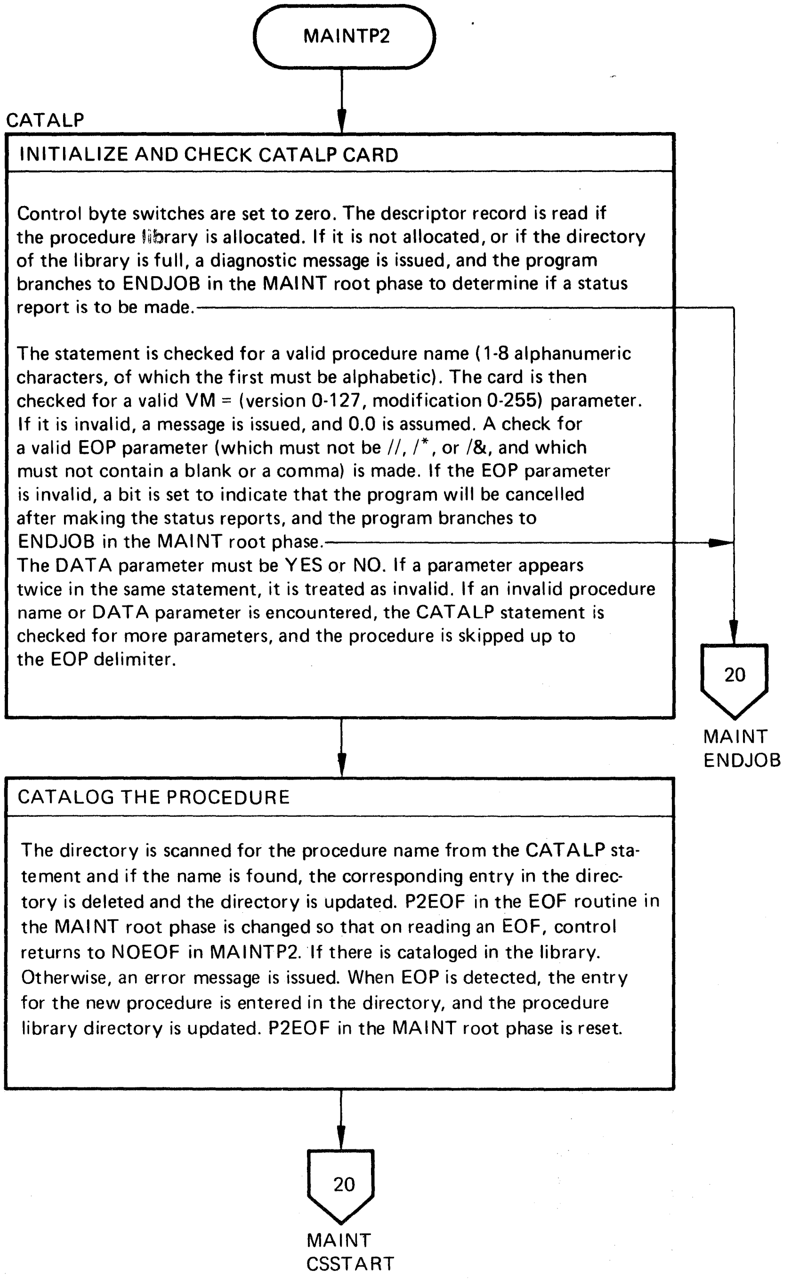
READ, COMPRESS, AND CATALOG BOOK

The MACRO or BKEND header statement is read. If a BKEND, switches are set to perform any of the options the user called for (sequence checking, statement counting, or input in compressed format). The source statements of the book are read and put into the output block in compressed format. If already compressed, the input is put directly in the output block. If not, the input is compressed and then put in. When the output block becomes full, it is written into the source statement library. The program checks each statement to determine if it is a MEND ending in a MACRO definition book or a BKEND ending in a source deck. When the last statement is recognized, it is written into the suorce statement library with a hexadecimal zero at the end to identify end-of-book. A switch is set to indicate the next statement read will be the first of a new book. If SYSIPT is tape or disk, the program branches to EOF in MAINT root phase when a /* is encountered.

ALL THROUGH PROCESSING

The source statement directory is scanned for the book name, and if found, the entry is deleted. The new entry is added to the directory. The source statement descriptor record is updated and written together with the affected directory records on SYSRES/SYSSLB. MAINTS2F returns to CSSTART in MAINT to read another card.

Chart 34. MAINTP2



INITIALIZE AND CHECK CATALP CARD

Control byte switches are set to zero. The descriptor record is read if the procedure library is allocated. If it is not allocated, or if the directory of the library is full, a diagnostic message is issued, and the program branches to ENDJOB in the MAINT root phase to determine if a status report is to be made.

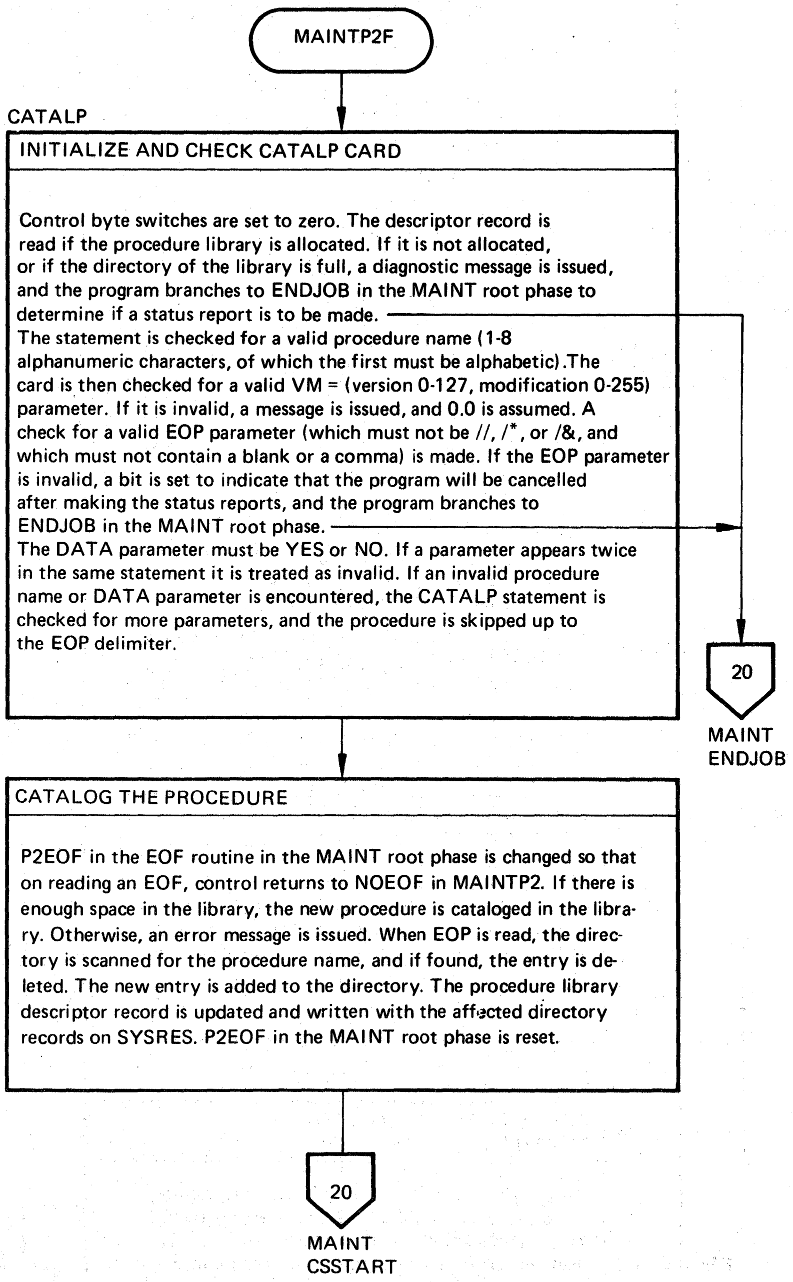
The statement is checked for a valid procedure name (1-8 alphanumeric characters, of which the first must be alphabetic). The card is then checked for a valid VM = (version 0-127, modification 0-255) parameter. If it is invalid, a message is issued, and 0.0 is assumed. A check for a valid EOP parameter (which must not be //, /*, or /&, and which must not contain a blank or a comma) is made. If the EOP parameter is invalid, a bit is set to indicate that the program will be cancelled after making the status reports, and the program branches to ENDJOB in the MAINT root phase.

The DATA parameter must be YES or NO. If a parameter appears twice in the same statement, it is treated as invalid. If an invalid procedure name or DATA parameter is encountered, the CATALP statement is checked for more parameters, and the procedure is skipped up to the EOP delimiter.

CATALOG THE PROCEDURE

The directory is scanned for the procedure name from the CATALP statement and if the name is found, the corresponding entry in the directory is deleted and the directory is updated. P2EOF in the EOF routine in the MAINT root phase is changed so that on reading an EOF, control returns to NOEOF in MAINTP2. If there is cataloged in the library. Otherwise, an error message is issued. When EOP is detected, the entry for the new procedure is entered in the directory, and the procedure library directory is updated. P2EOF in the MAINT root phase is reset.

Chart 35. MAINTP2F



INITIALIZE AND CHECK CATALP CARD

Control byte switches are set to zero. The descriptor record is read if the procedure library is allocated. If it is not allocated, or if the directory of the library is full, a diagnostic message is issued, and the program branches to ENDJOB in the MAINT root phase to determine if a status report is to be made.

The statement is checked for a valid procedure name (1-8 alphanumeric characters, of which the first must be alphabetic).The card is then checked for a valid VM = (version 0-127, modification 0-255) parameter. If it is invalid, a message is issued, and 0.0 is assumed. A check for a valid EOP parameter (which must not be //, /*, or /&, and which must not contain a blank or a comma) is made. If the EOP parameter is invalid, a bit is set to indicate that the program will be cancelled after making the status reports, and the program branches to ENDJOB in the MAINT root phase.

The DATA parameter must be YES or NO. If a parameter appears twice in the same statement it is treated as invalid. If an invalid procedure name or DATA parameter is encountered, the CATALP statement is checked for more parameters, and the procedure is skipped up to the EOP delimiter.

CATALOG THE PROCEDURE

P2EOF in the EOF routine in the MAINT root phase is changed so that on reading an EOF, control returns to NOEOF in MAINTP2. If there is enough space in the library, the new procedure is cataloged in the library. Otherwise, an error message is issued. When EOP is read, the directory is scanned for the procedure name, and if found, the entry is deleted. The new entry is added to the directory. The procedure library descriptor record is updated and written with the affected directory records on SYSRES. P2EOF in the MAINT root phase is reset.

Chart 36. MAINTDR (Part 1 of 2)

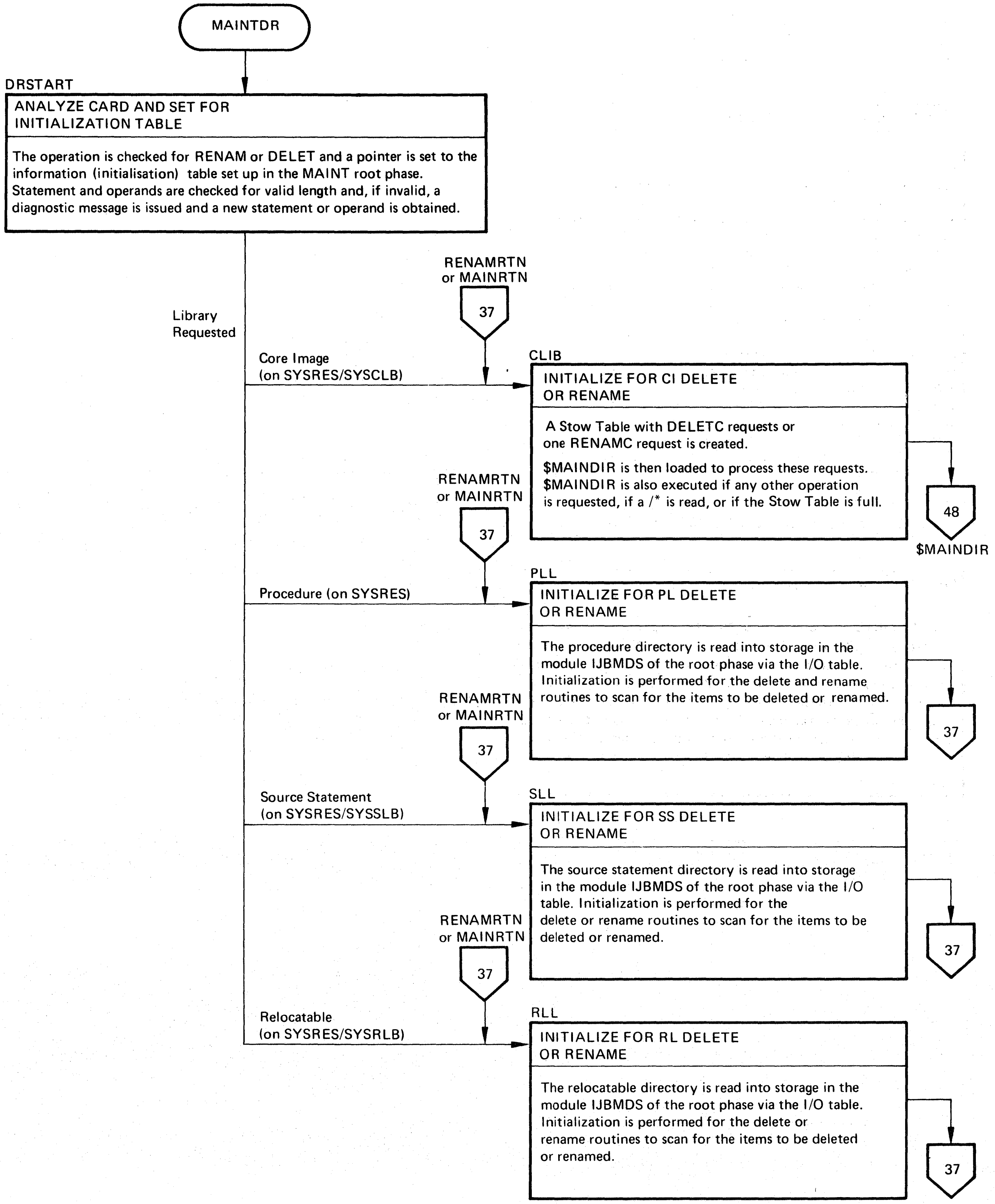
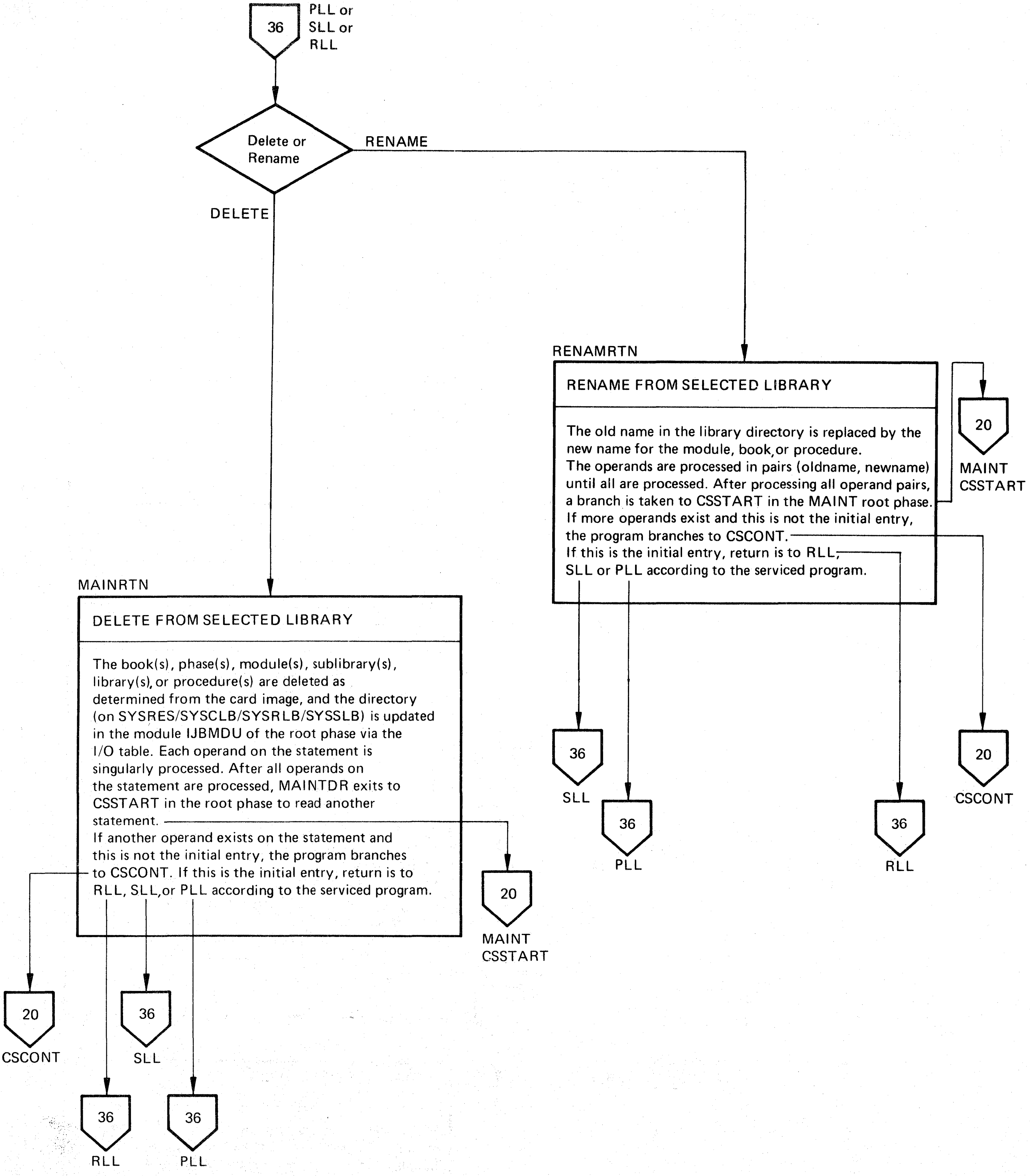

MAINTDR
DRSTART
ANALYZE CARD AND SET FOR INITIALIZATION TABLE
The operation is checked for RENAM or DELET and a pointer is set to the information (initialisation) table set up in the MAINT root phase. Statement and operands are checked for valid length and, if invalid, a diagnostic message is issued and a new statement or operand is obtained.
Library Requested
RENAMRTN or MAINRTN
37
Core Image (on SYSRES/SYSCLB)
CLIB
INITIALIZE FOR CI DELETE OR RENAME
A Stow Table with DELETC requests or one RENAMC request is created.
$MAINDIR is then loaded to process these requests. $MAINDIR is also executed if any other operation is requested, if a /* is read, or if the Stow Table is full.
48
$MAINDIR
RENAMRTN or MAINRTN
37
Procedure (on SYSRES)
PLL
INITIALIZE FOR PL DELETE OR RENAME
The procedure directory is read into storage in the module IJBMDS of the root phase via the I/O table. Initialization is performed for the delete and rename routines to scan for the items to be deleted or renamed.
37
RENAMRTN or MAINRTN
37
Source Statement (on SYSRES/SYSSLB)
SLL
INITIALIZE FOR SS DELETE OR RENAME
The source statement directory is read into storage in the module IJBMDS of the root phase via the I/O table. Initialization is performed for the delete or rename routines to scan for the items to be deleted or renamed.
37
RENAMRTN or MAINRTN
37
Relocatable (on SYSRES/SYSRLB)
RLL
INITIALIZE FOR RL DELETE OR RENAME
The relocatable directory is read into storage in the module IJBMDS of the root phase via the I/O table. Initialization is performed for the delete or rename routines to scan for the items to be deleted or renamed.
37

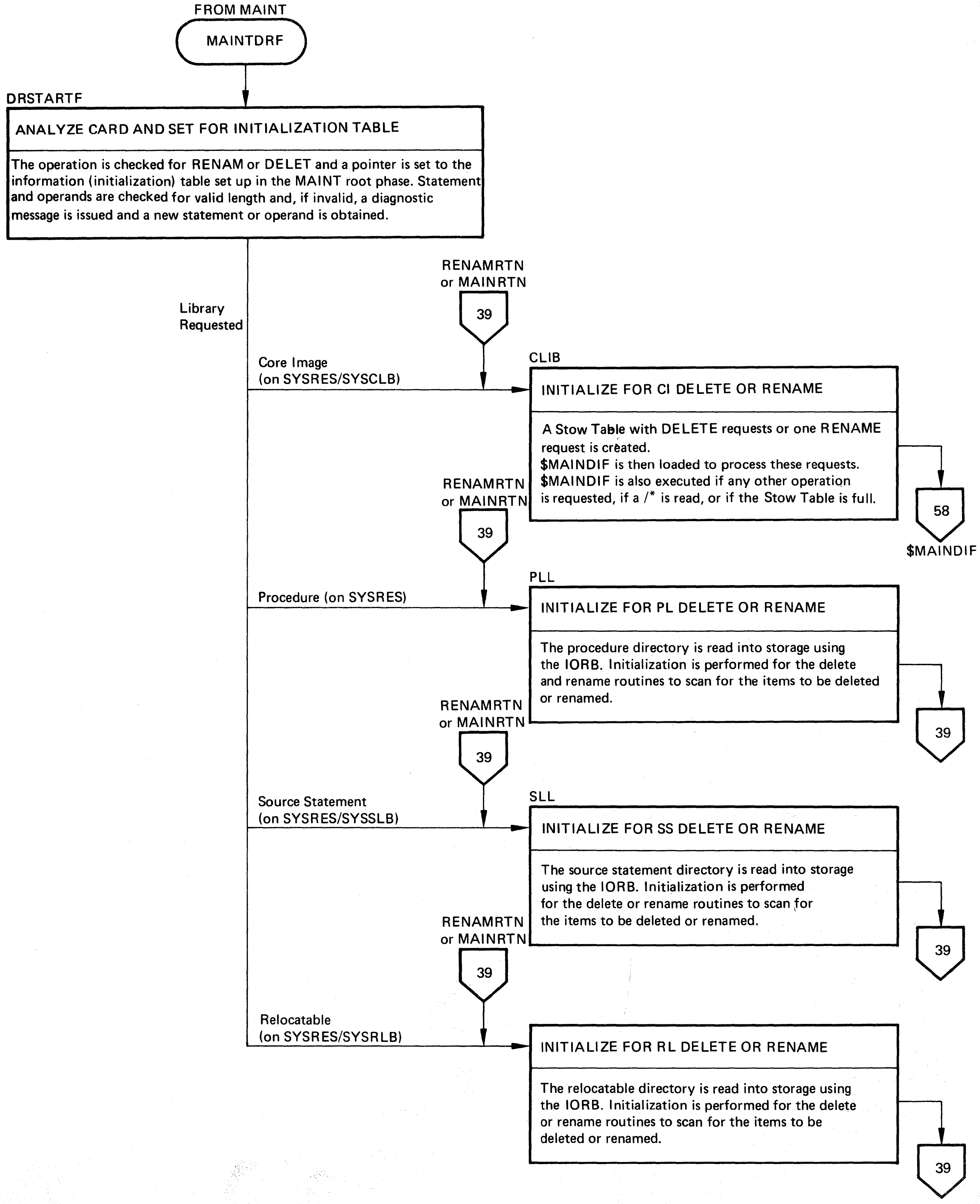
Chart 37. MAINTDR (Part 2 of 2)



The old name in the library directory is replaced by the new name for the module, book, or procedure.
The operands are processed in pairs (oldname, newname) until all are processed. After processing all operand pairs, a branch is taken to CSSTART in the MAINT root phase.
If more operands exist and this is not the initial entry, the program branches to CSCONT.
If this is the initial entry, return is to RLL, SLL or PLL according to the serviced program.

The book(s), phase(s), module(s), sublibrary(s), library(s), or procedure(s) are deleted as determined from the card image, and the directory (on SYSRES/SYSCLB/SYSRLB/SYSSLB) is updated in the module IJBMDU of the root phase via the I/O table. Each operand on the statement is singularly processed. After all operands on the statement are processed, MAINTDR exits to CSSTART in the root phase to read another statement.
If another operand exists on the statement and this is not the initial entry, the program branches to CSCONT. If this is the initial entry, return is to RLL, SLL, or PLL according to the serviced program.

Chart 38. MAINTDRF (Part 1 of 2)



ANALYZE CARD AND SET FOR INITIALIZATION TABLE

The operation is checked for RENAM or DELET and a pointer is set to the information (initialization) table set up in the MAINT root phase. Statement and operands are checked for valid length and, if invalid, a diagnostic message is issued and a new statement or operand is obtained.

INITIALIZE FOR CI DELETE OR RENAME

A Stow Table with DELETE requests or one RENAME request is created. $MAINDIF is then loaded to process these requests. $MAINDIF is also executed if any other operation is requested, if a /* is read, or if the Stow Table is full.

INITIALIZE FOR PL DELETE OR RENAME

The procedure directory is read into storage using the IORB. Initialization is performed for the delete and rename routines to scan for the items to be deleted or renamed.

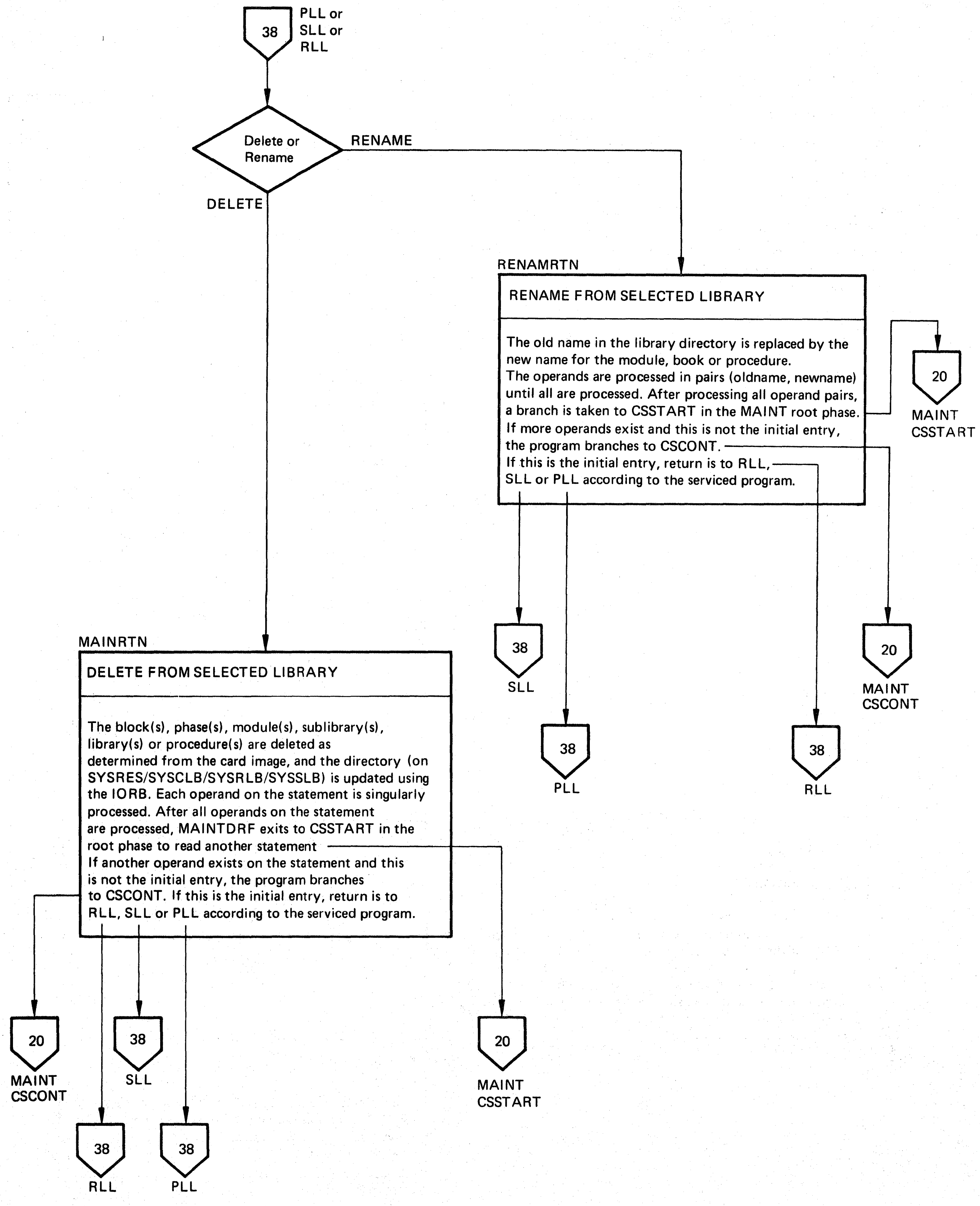
INITIALIZE FOR SS DELETE OR RENAME

The source statement directory is read into storage using the IORB. Initialization is performed for the delete or rename routines to scan for the items to be deleted or renamed.

INITIALIZE FOR RL DELETE OR RENAME

The relocatable directory is read into storage using the IORB. Initialization is performed for the delete or rename routines to scan for the items to be deleted or renamed.

Chart 39. MAINTDRF (Part 2 of 2)



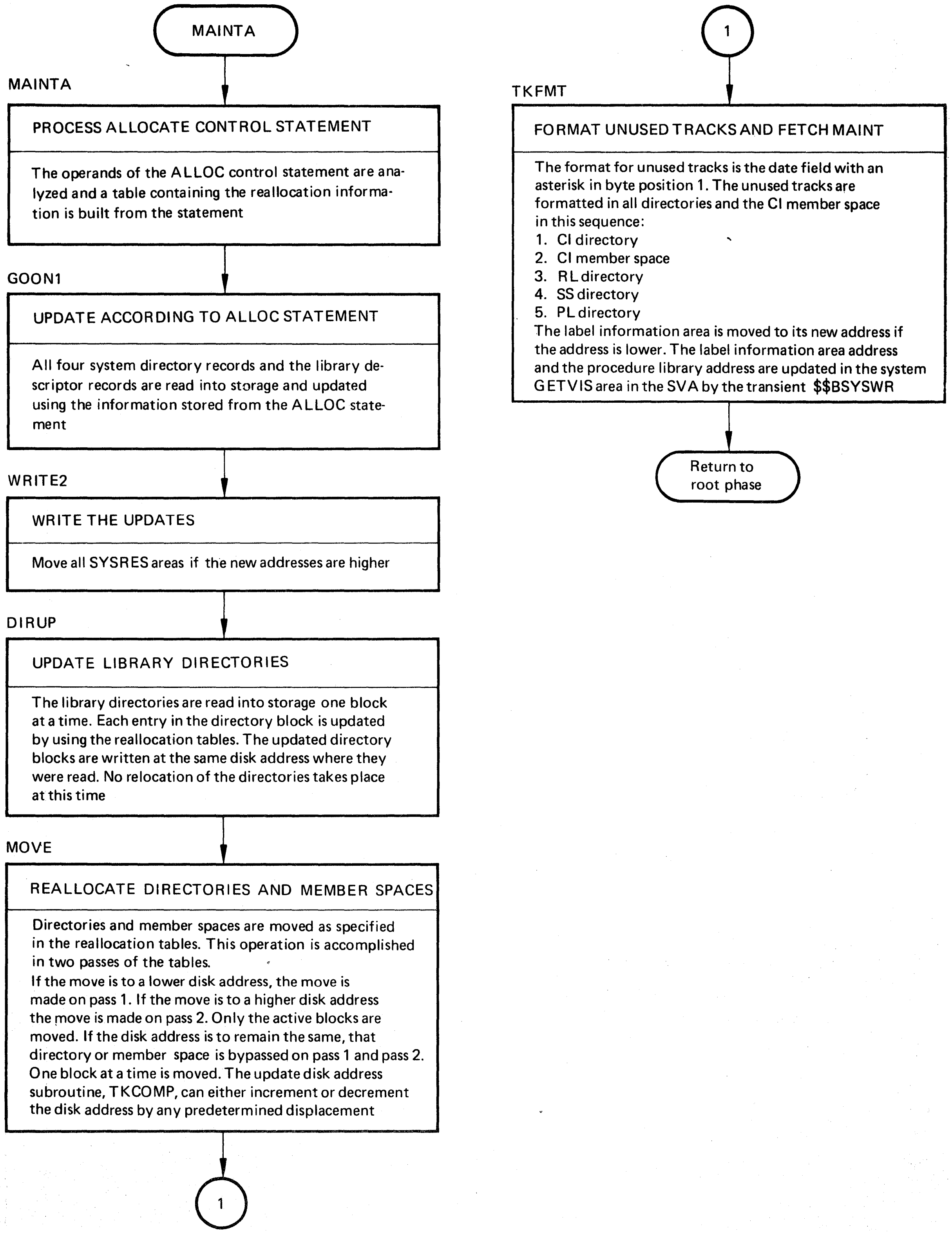
RENAME FROM SELECTED LIBRARY

The old name in the library directory is replaced by the new name for the module, book or procedure.
The operands are processed in pairs (oldname, newname) until all are processed. After processing all operand pairs, a branch is taken to CSSTART in the MAINT root phase.
If more operands exist and this is not the initial entry, the program branches to CSCONT.
If this is the initial entry, return is to RLL, SLL or PLL according to the serviced program.

DELETE FROM SELECTED LIBRARY

The block(s), phase(s), module(s), sublibrary(s), library(s) or procedure(s) are deleted as determined from the card image, and the directory (on SYSRES/SYSCLB/SYSRLB/SYSSLB) is updated using the IORB. Each operand on the statement is singularly processed. After all operands on the statement are processed, MAINTDRF exits to CSSTART in the root phase to read another statement
If another operand exists on the statement and this is not the initial entry, the program branches to CSCONT. If this is the initial entry, return is to RLL, SLL or PLL according to the serviced program.

Chart 40. MAINTA (Detail Charts MA-MQ)



MAINTA

MAINTA

PROCESS ALLOCATE CONTROL STATEMENT

The operands of the ALLOC control statement are analyzed and a table containing the reallocation information is built from the statement

GOON1

UPDATE ACCORDING TO ALLOC STATEMENT

All four system directory records and the library descriptor records are read into storage and updated using the information stored from the ALLOC statement

WRITE2

WRITE THE UPDATES

Move all SYSRES areas if the new addresses are higher

DIRUP

UPDATE LIBRARY DIRECTORIES

The library directories are read into storage one block at a time. Each entry in the directory block is updated by using the reallocation tables. The updated directory blocks are written at the same disk address where they were read. No relocation of the directories takes place at this time

MOVE

REALLOCATE DIRECTORIES AND MEMBER SPACES

Directories and member spaces are moved as specified in the reallocation tables. This operation is accomplished in two passes of the tables.
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 member space 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

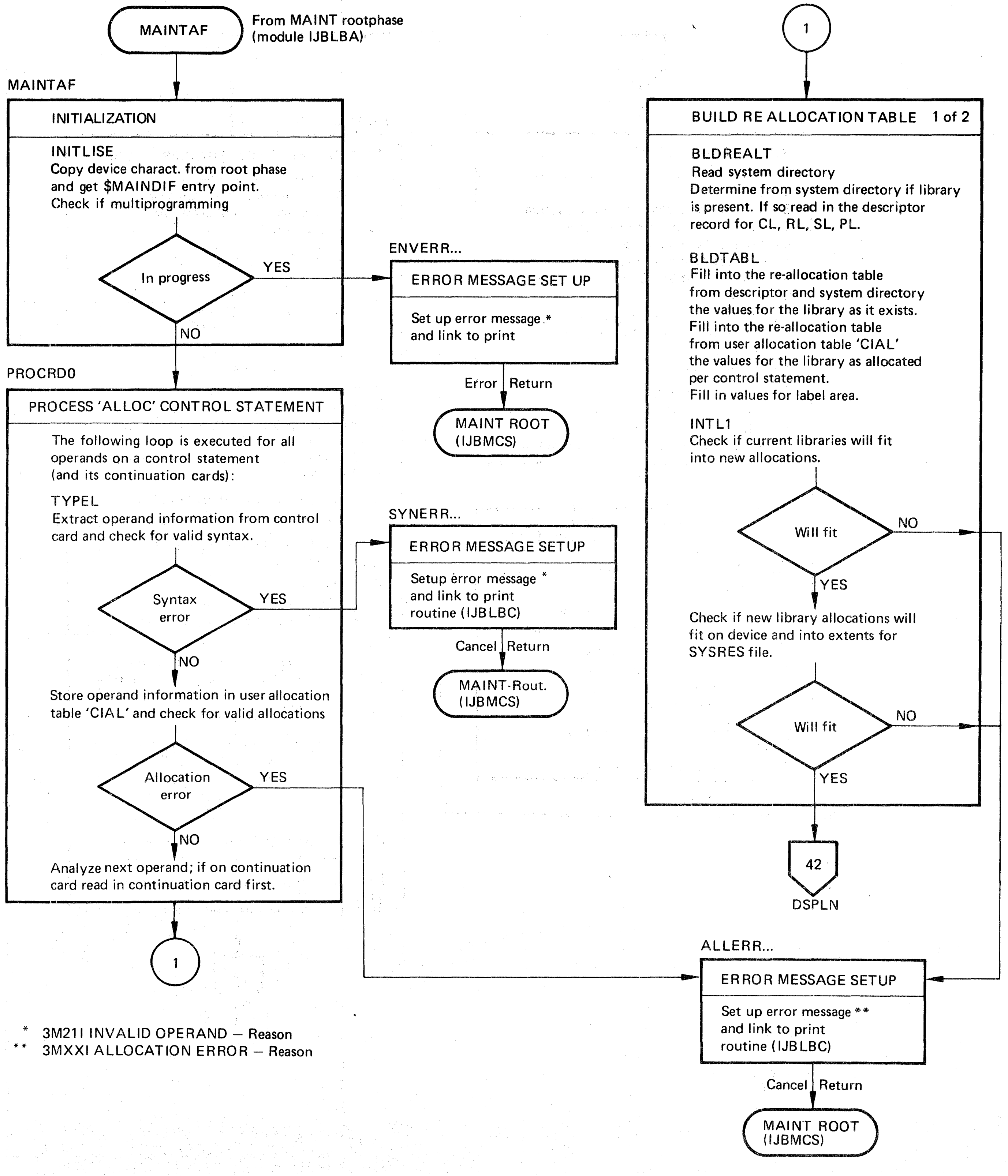
TKFMT

FORMAT UNUSED TRACKS AND FETCH MAINT

The format for unused tracks is the date field with an asterisk in byte position 1. The unused tracks are formatted in all directories and the CI member space in this sequence:
1. CI directory
2. CI member space
3. RL directory
4. SS directory
5. PL directory
The label information area is moved to its new address if the address is lower. The label information area address and the procedure library address are updated in the system GETVIS area in the SVA by the transient $$BSYSWR

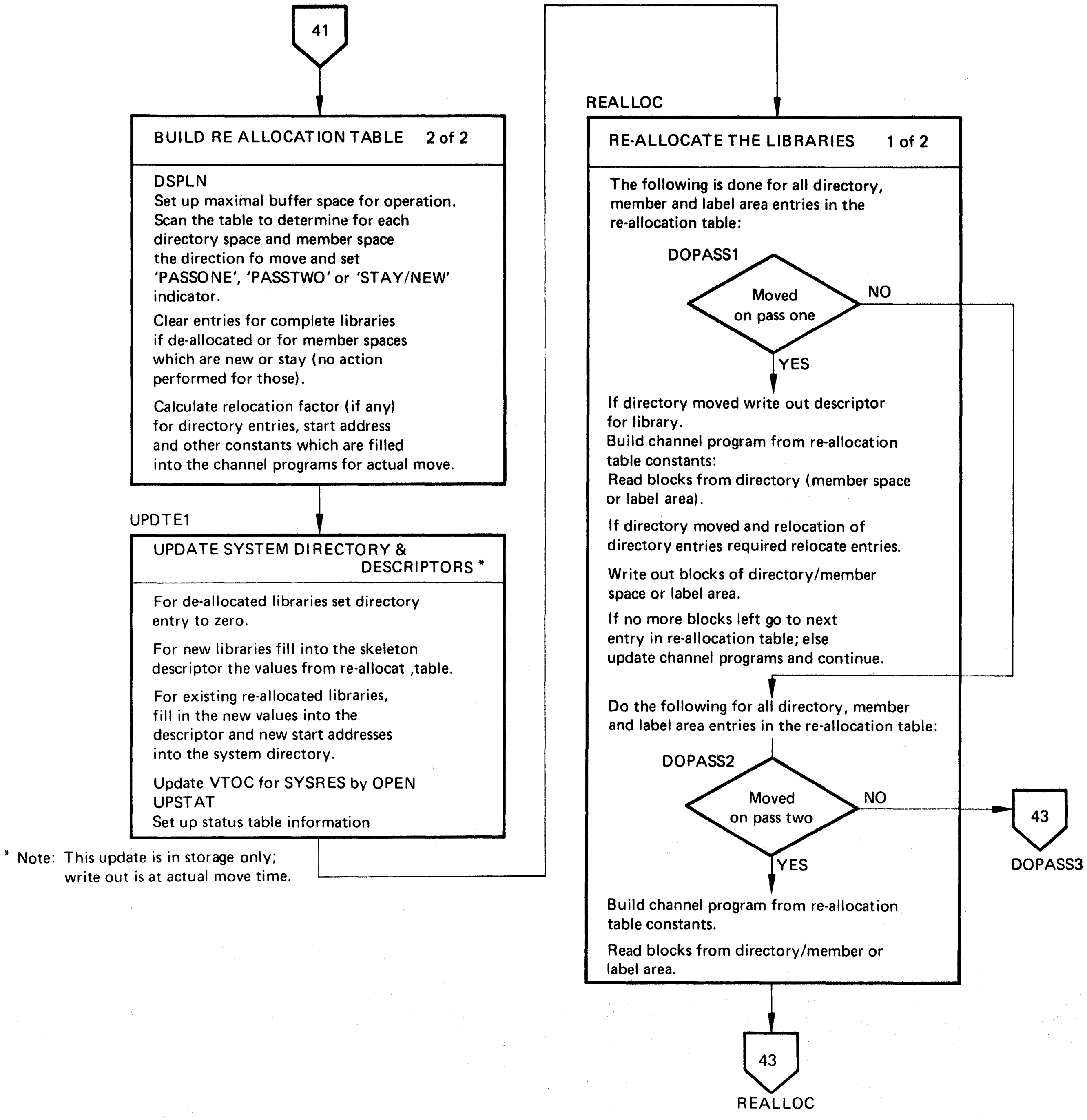
Return to root phase

Chart 41. MAINTAF (Part 1 of 3)


MAINTAF
From MAINT rootphase (module IJBLBA)
MAINTAF
INITIALIZATION
INITLISE
Copy device charact. from root phase and get $MAINDIF entry point.
Check if multiprogramming
In progress
YES
NO
ENVERR...
ERROR MESSAGE SET UP
Set up error message * and link to print
Error
Return
MAINT ROOT (IJBMCS)
PROCRDO
PROCESS 'ALLOC' CONTROL STATEMENT
The following loop is executed for all operands on a control statement (and its continuation cards):
TYPEL
Extract operand information from control card and check for valid syntax.
Syntax error
YES
NO
SYNERR...
ERROR MESSAGE SETUP
Setup error message * and link to print routine (IJBLBC)
Cancel
Return
MAINT-Rout. (IJBMCS)
Store operand information in user allocation table 'CIAL' and check for valid allocations
Allocation error
YES
NO
Analyze next operand; if on continuation card read in continuation card first.
1
1
BUILD RE ALLOCATION TABLE 1 of 2
BLDREALT
Read system directory
Determine from system directory if library is present. If so read in the descriptor record for CL, RL, SL, PL.
BLDTABL
Fill into the re-allocation table from descriptor and system directory the values for the library as it exists.
Fill into the re-allocation table from user allocation table 'CIAL' the values for the library as allocated per control statement.
Fill in values for label area.
INTL1
Check if current libraries will fit into new allocations.
Will fit
NO
YES
Check if new library allocations will fit on device and into extents for SYSRES file.
Will fit
NO
YES
42
DSPLN
ALLERR...
ERROR MESSAGE SETUP
Set up error message ** and link to print routine (IJBLBC)
Cancel
Return
MAINT ROOT (IJBMCS)

* 3M21I INVALID OPERAND – Reason
** 3MXXI ALLOCATION ERROR – Reason

Chart 42. MAINTAF (Part 2 of 3)



BUILD RE ALLOCATION TABLE 2 of 2

DSPLN
Set up maximal buffer space for operation.
Scan the table to determine for each directory space and member space the direction fo move and set 'PASSONE', 'PASSTWO' or 'STAY/NEW' indicator.

Clear entries for complete libraries if de-allocated or for member spaces which are new or stay (no action performed for those).

Calculate relocation factor (if any) for directory entries, start address and other constants which are filled into the channel programs for actual move.

UPDTE1

UPDATE SYSTEM DIRECTORY & DESCRIPTORS *

For de-allocated libraries set directory entry to zero.

For new libraries fill into the skeleton descriptor the values from re-allocat ,table.

For existing re-allocated libraries, fill in the new values into the descriptor and new start addresses into the system directory.

Update VTOC for SYSRES by OPEN
UPSTAT
Set up status table information

* Note: This update is in storage only; write out is at actual move time.

REALLOC

RE-ALLOCATE THE LIBRARIES 1 of 2

The following is done for all directory, member and label area entries in the re-allocation table:

If directory moved write out descriptor for library.
Build channel program from re-allocation table constants:
Read blocks from directory (member space or label area).

If directory moved and relocation of directory entries required relocate entries.

Write out blocks of directory/member space or label area.

If no more blocks left go to next entry in re-allocation table; else update channel programs and continue.

Do the following for all directory, member and label area entries in the re-allocation table:

Build channel program from re-allocation table constants.

Read blocks from directory/member or label area.

Chart 43. MAINTAF (Part 3 of 3)

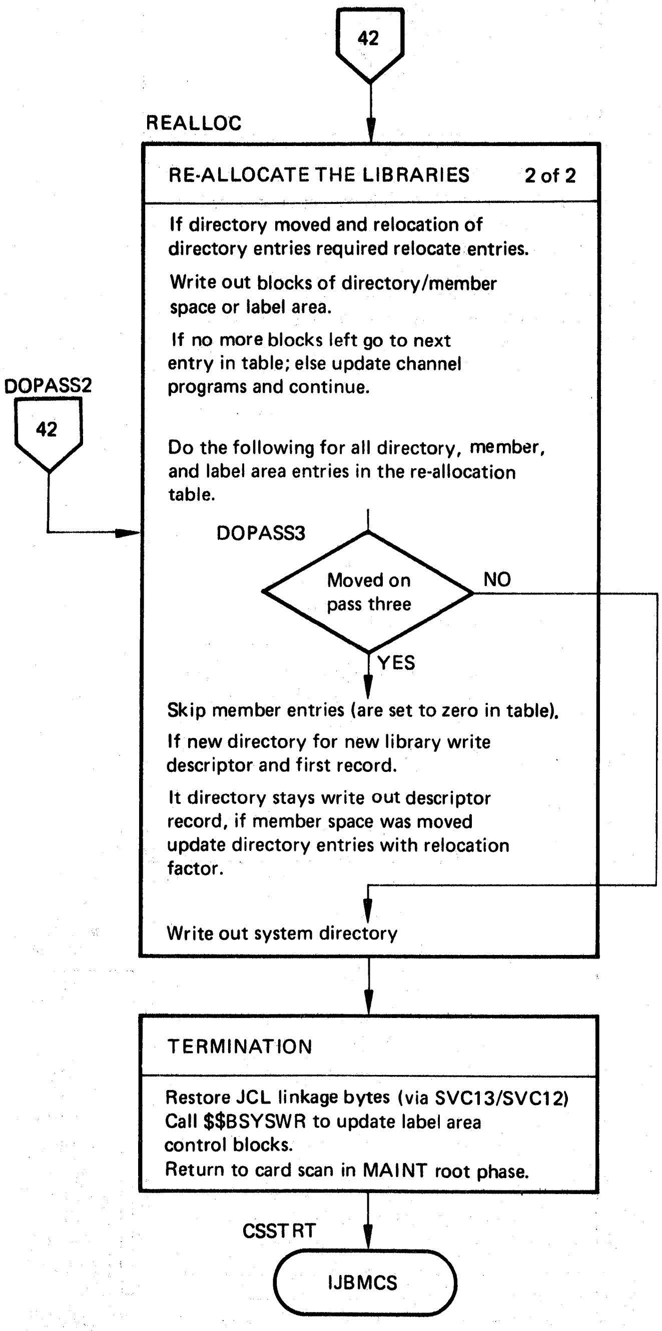
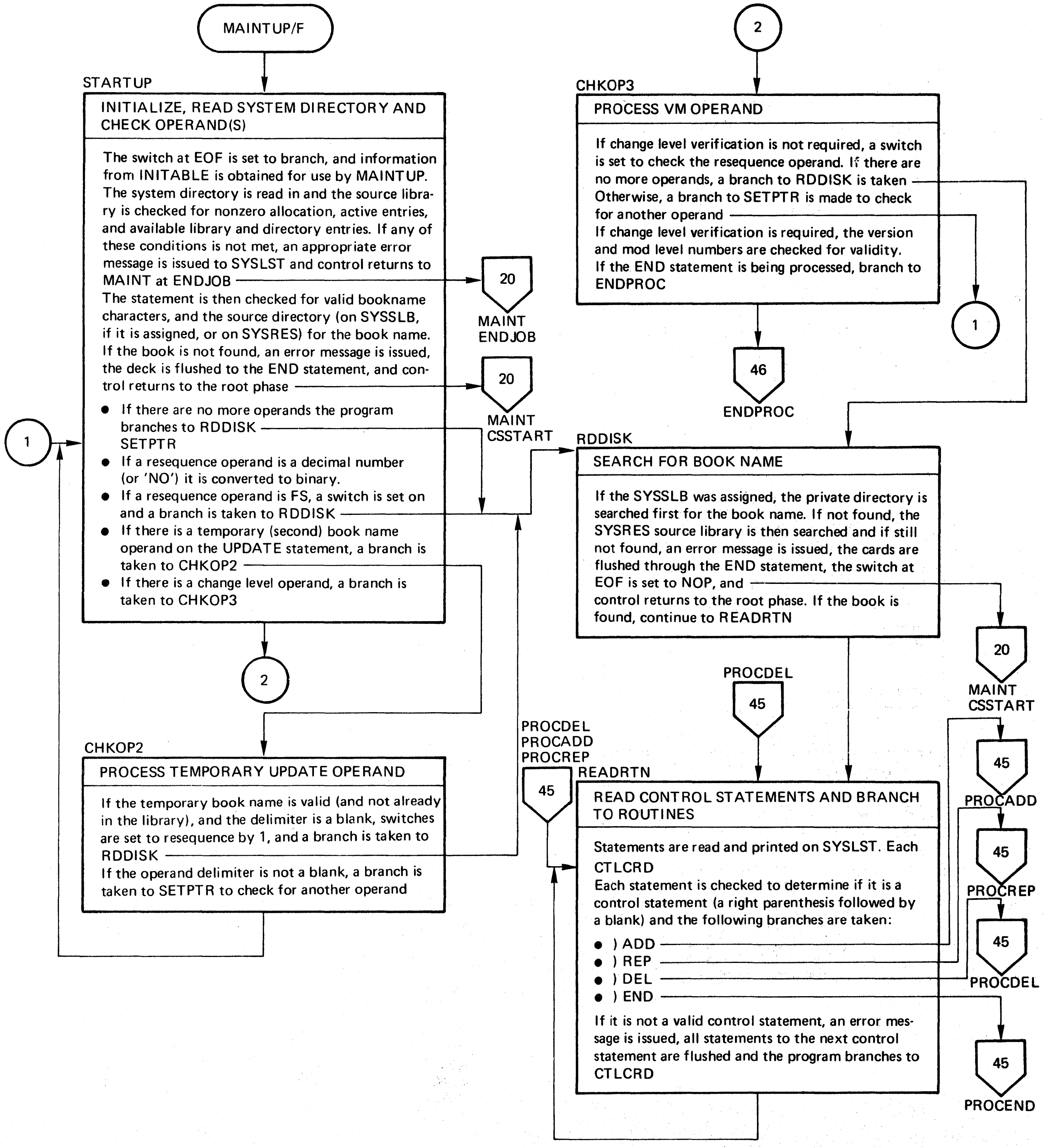

42
REALLOC
RE-ALLOCATE THE LIBRARIES 2 of 2
If directory moved and relocation of directory entries required relocate entries.
Write out blocks of directory/member space or label area.
If no more blocks left go to next entry in table; else update channel programs and continue.
DOPASS2
42
Do the following for all directory, member, and label area entries in the re-allocation table.
DOPASS3
Moved on pass three
NO
YES
Skip member entries (are set to zero in table).
If new directory for new library write descriptor and first record.
It directory stays write out descriptor record, if member space was moved update directory entries with relocation factor.
Write out system directory
TERMINATION
Restore JCL linkage bytes (via SVC13/SVC12)
Call $$BSYSWR to update label area control blocks.
Return to card scan in MAINT root phase.
CSSTRT
IJBMCS

Chart 44. MAINTUP/F (Part 1 of 3) (Detail Charts NA-PG)



INITIALIZE, READ SYSTEM DIRECTORY AND CHECK OPERAND(S)

The switch at EOF is set to branch, and information from INITABLE is obtained for use by MAINTUP. The system directory is read in and the source library is checked for nonzero allocation, active entries, and available library and directory entries. If any of these conditions is not met, an appropriate error message is issued to SYSLST and control returns to MAINT at ENDJOB

The statement is then checked for valid bookname characters, and the source directory (on SYSSLB, if it is assigned, or on SYSRES) for the book name. If the book is not found, an error message is issued, the deck is flushed to the END statement, and control returns to the root phase

- If there are no more operands the program branches to RDDISK
SETPTR
- If a resequence operand is a decimal number (or 'NO') it is converted to binary.
- If a resequence operand is FS, a switch is set on and a branch is taken to RDDISK
- If there is a temporary (second) book name operand on the UPDATE statement, a branch is taken to CHKOP2
- If there is a change level operand, a branch is taken to CHKOP3

PROCESS TEMPORARY UPDATE OPERAND

If the temporary book name is valid (and not already in the library), and the delimiter is a blank, switches are set to resequence by 1, and a branch is taken to RDDISK

If the operand delimiter is not a blank, a branch is taken to SETPTR to check for another operand

PROCESS VM OPERAND

If change level verification is not required, a switch is set to check the resequence operand. If there are no more operands, a branch to RDDISK is taken

Otherwise, a branch to SETPTR is made to check for another operand

If change level verification is required, the version and mod level numbers are checked for validity.

If the END statement is being processed, branch to ENDPROC

SEARCH FOR BOOK NAME

If the SYSSLB was assigned, the private directory is searched first for the book name. If not found, the SYSRES source library is then searched and if still not found, an error message is issued, the cards are flushed through the END statement, the switch at EOF is set to NOP, and

control returns to the root phase. If the book is found, continue to READRTN

READ CONTROL STATEMENTS AND BRANCH TO ROUTINES

Statements are read and printed on SYSLST. Each

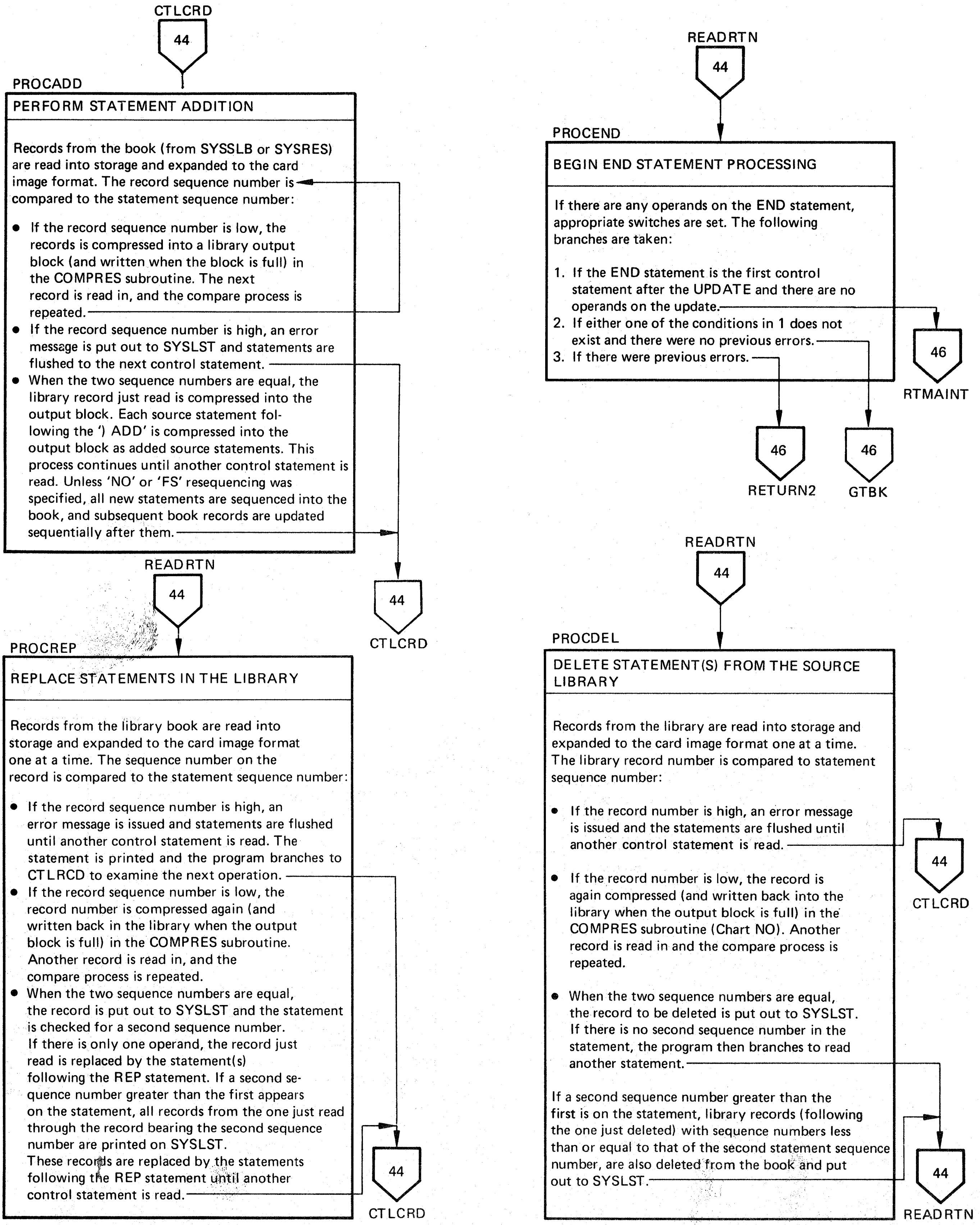
CTLCRD

Each statement is checked to determine if it is a control statement (a right parenthesis followed by a blank) and the following branches are taken:

-) ADD
-) REP
-) DEL
-) END

If it is not a valid control statement, an error message is issued, all statements to the next control statement are flushed and the program branches to CTLCRD

Chart 45. MAINTUP/F (Part 2 of 3) (Detail Charts NA-PG)



PERFORM STATEMENT ADDITION

Records from the book (from SYSSLB or SYSRES) are read into storage and expanded to the card image format. The record sequence number is compared to the statement sequence number:

- If the record sequence number is low, the records is compressed into a library output block (and written when the block is full) in the COMPRES subroutine. The next record is read in, and the compare process is repeated.
- If the record sequence number is high, an error message is put out to SYSLST and statements are flushed to the next control statement.
- When the two sequence numbers are equal, the library record just read is compressed into the output block. Each source statement following the ') ADD' is compressed into the output block as added source statements. This process continues until another control statement is read. Unless 'NO' or 'FS' resequencing was specified, all new statements are sequenced into the book, and subsequent book records are updated sequentially after them.

BEGIN END STATEMENT PROCESSING

If there are any operands on the END statement, appropriate switches are set. The following branches are taken:

1. If the END statement is the first control statement after the UPDATE and there are no operands on the update.
2. If either one of the conditions in 1 does not exist and there were no previous errors.
3. If there were previous errors.

REPLACE STATEMENTS IN THE LIBRARY

Records from the library book are read into storage and expanded to the card image format one at a time. The sequence number on the record is compared to the statement sequence number:

- If the record sequence number is high, an error message is issued and statements are flushed until another control statement is read. The statement is printed and the program branches to CTLRCD to examine the next operation.
- If the record sequence number is low, the record number is compressed again (and written back in the library when the output block is full) in the COMPRES subroutine. Another record is read in, and the compare process is repeated.
- When the two sequence numbers are equal, the record is put out to SYSLST and the statement is checked for a second sequence number. If there is only one operand, the record just read is replaced by the statement(s) following the REP statement. If a second sequence number greater than the first appears on the statement, all records from the one just read through the record bearing the second sequence number are printed on SYSLST. These records are replaced by the statements following the REP statement until another control statement is read.

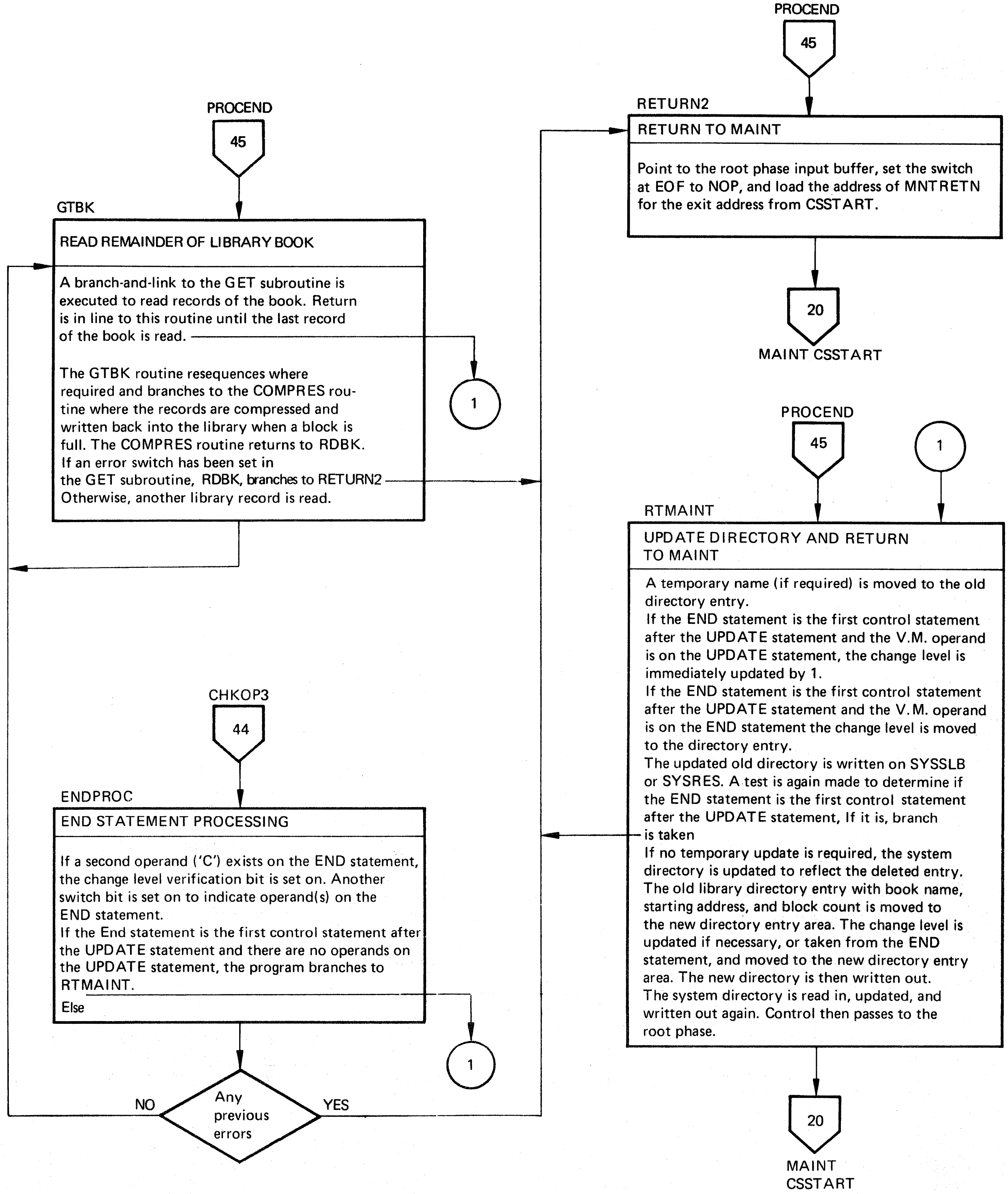
DELETE STATEMENT(S) FROM THE SOURCE LIBRARY

Records from the library are read into storage and expanded to the card image format one at a time. The library record number is compared to statement sequence number:

- If the record number is high, an error message is issued and the statements are flushed until another control statement is read.
- If the record number is low, the record is again compressed (and written back into the library when the output block is full) in the COMPRES subroutine (Chart NO). Another record is read in and the compare process is repeated.
- When the two sequence numbers are equal, the record to be deleted is put out to SYSLST. If there is no second sequence number in the statement, the program then branches to read another statement.

If a second sequence number greater than the first is on the statement, library records (following the one just deleted) with sequence numbers less than or equal to that of the second statement sequence number, are also deleted from the book and put out to SYSLST.

Chart 46. MAINTUP/F (Part 3 of 3) (Detail Charts NA-PG)



READ REMAINDER OF LIBRARY BOOK

A branch-and-link to the GET subroutine is executed to read records of the book. Return is in line to this routine until the last record of the book is read.

The GTBK routine resequences where required and branches to the COMPRES routine where the records are compressed and written back into the library when a block is full. The COMPRES routine returns to RDBK. If an error switch has been set in the GET subroutine, RDBK, branches to RETURN2 Otherwise, another library record is read.

END STATEMENT PROCESSING

If a second operand ('C') exists on the END statement, the change level verification bit is set on. Another switch bit is set on to indicate operand(s) on the END statement.
If the End statement is the first control statement after the UPDATE statement and there are no operands on the UPDATE statement, the program branches to RTMAINT.
Else

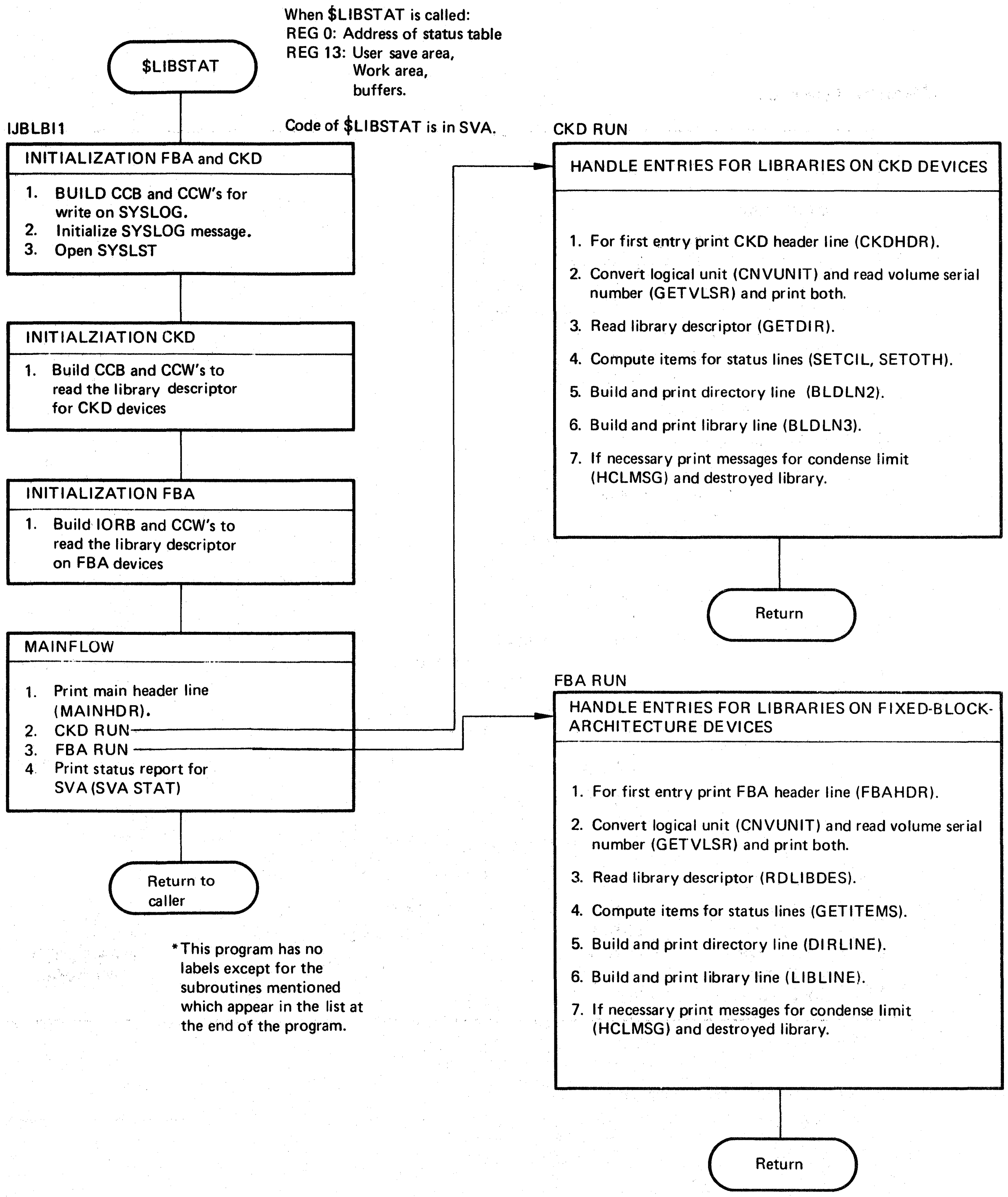
RETURN TO MAINT

Point to the root phase input buffer, set the switch at EOF to NOP, and load the address of MNTRETN for the exit address from CSSTART.

UPDATE DIRECTORY AND RETURN TO MAINT

A temporary name (if required) is moved to the old directory entry.
If the END statement is the first control statement after the UPDATE statement and the V.M. operand is on the UPDATE statement, the change level is immediately updated by 1.
If the END statement is the first control statement after the UPDATE statement and the V.M. operand is on the END statement the change level is moved to the directory entry.
The updated old directory is written on SYSSLB or SYSRES. A test is again made to determine if the END statement is the first control statement after the UPDATE statement, If it is, branch is taken
If no temporary update is required, the system directory is updated to reflect the deleted entry. The old library directory entry with book name, starting address, and block count is moved to the new directory entry area. The change level is updated if necessary, or taken from the END statement, and moved to the new directory entry area. The new directory is then written out.
The system directory is read in, updated, and written out again. Control then passes to the root phase.

Chart 47. $LIBSTAT



When $LIBSTAT is called:
REG 0: Address of status table
REG 13: User save area,
Work area,
buffers.

Code of $LIBSTAT is in SVA.

$LIBSTAT

IJBLBI1

INITIALIZATION FBA and CKD

1. BUILD CCB and CCW's for write on SYSLOG.
2. Initialize SYSLOG message.
3. Open SYSLST

INITIALZIATION CKD

1. Build CCB and CCW's to read the library descriptor for CKD devices

INITIALIZATION FBA

1. Build IORB and CCW's to read the library descriptor on FBA devices

MAINFLOW

1. Print main header line (MAINHDR).
2. CKD RUN
3. FBA RUN
4. Print status report for SVA (SVA STAT)

Return to caller

*This program has no labels except for the subroutines mentioned which appear in the list at the end of the program.

CKD RUN

HANDLE ENTRIES FOR LIBRARIES ON CKD DEVICES

1. For first entry print CKD header line (CKDHDR).
2. Convert logical unit (CNVUNIT) and read volume serial number (GETVLSR) and print both.
3. Read library descriptor (GETDIR).
4. Compute items for status lines (SETCIL, SETOTH).
5. Build and print directory line (BLDLN2).
6. Build and print library line (BLDLN3).
7. If necessary print messages for condense limit (HCLMSG) and destroyed library.

Return

FBA RUN

HANDLE ENTRIES FOR LIBRARIES ON FIXED-BLOCK-ARCHITECTURE DEVICES

1. For first entry print FBA header line (FBAHDR).
2. Convert logical unit (CNVUNIT) and read volume serial number (GETVLSR) and print both.
3. Read library descriptor (RDLIBDES).
4. Compute items for status lines (GETITEMS).
5. Build and print directory line (DIRLINE).
6. Build and print library line (LIBLINE).
7. If necessary print messages for condense limit (HCLMSG) and destroyed library.

Return

Chart 48. $MAINDIR (Part 1 of 10)

$MAINDIR Overview:

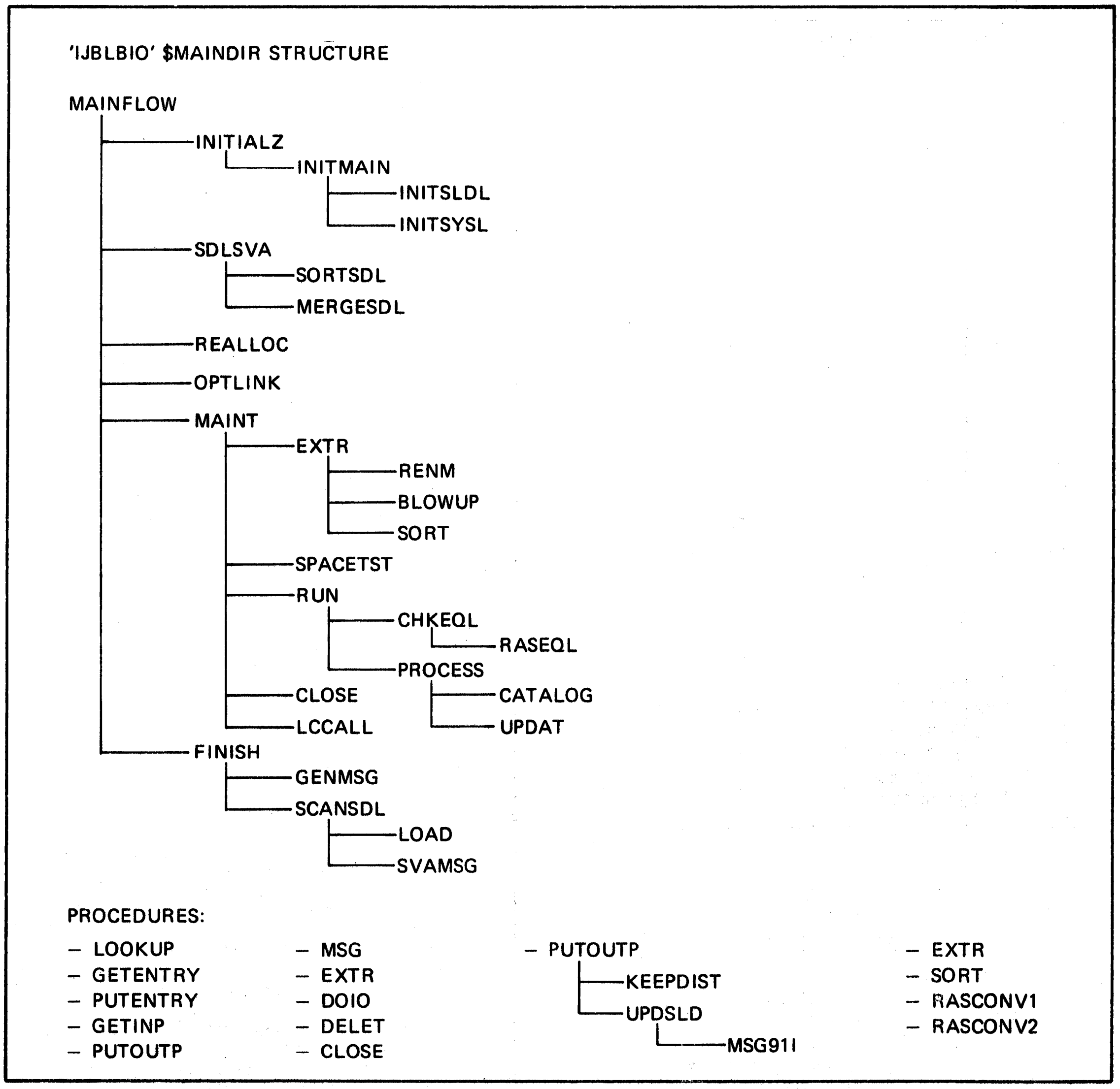


Chart 49. $MAINDIR (Part 2 of 10)

$MAINDIR Processing:

```
START OF MAINFLOW

INITIALIZATION:
                    INCLUDE 'INITIALZ'
CALLED BY IPL OR JCL:
                    INCLUDE 'SDLSVA'
                    GOTO FINI
CALLED BY LINKAGE EDITOR
FOR //OPTION LINK:
                    INCLUDE 'OPTLINK'
                    GOTO FINI
CALLED BY MAINT FOR REALLOCATION:
                    INCLUDE 'REALLOC'
                    GOTO FINI
TERMINATION OF CONDENSE:
                    INCLUDE 'LCCALL'
                    GOTO FINI
CALLED BY LINKAGE EDITOR FOR //OPTION CATAL, OR
BY MAINT FOR DELETC, RENAMC, OR
BY MAINT FOR CONDENSE OR
BY CORGZ FOR COPY:
                    INCLUDE 'MAINT'
                    GOTO FINI
FINI:
CLEANUP, GENERATION OF MESSAGES,
LOAD INTO SVA:
                    INCLUDE 'FINISH'
                    RETURN

END OF MAINFLOW
```

INITIALZ

INITIALIZATION FOR $MAINDIR

1. PROVIDE ADDRESSABILITY OF STOWTABLE AND OF ARRAY 'TABIN' WHICH IS USED TO SORT THE STOWTABLE.
2. MODIFY LAST BYTE USED TO FORCE PROTECTION EXCEPTION IF PARTITION IS TOO SMALL.
3. OBTAIN STORAGE KEY 0 WHICH ALLOWS TO WRITE INTO THE SUPERVISOR (NECESSARY FOR SLD AND FETCHTABLE UPDATE).
4. STORE TIME OF DAY CLOCK CONTENTS IN 'TIME'.
6. INITIALIZE COUNTER FOR MESSAGES AND SVA LOAD REQUESTS.

Chart 50. $MAINDIR (Part 3 of 10)

INITMAIN

MAIN INITIALIZATION

1. CLEAR ALL SWITCHES, INITIALIZE POINTERS FOR SDL,SLD AND RAS LOADLIST.
2. GET POINTERS TO SECOND LEVEL DIRECTORY AND SYSTEM DIRECTORY LIST ('INITSLDL').
3. FOR IPL GOTO 6.
4. INSPECT FIRST STOWTABLE ENTRY. IF TYPE IS 'LINK' OR 'CATAL' THEN INSERT THIS PHASENAME INTO THE GETVIS AREA FOR AN '// EXEC ' USE.
5. READ LIBRARY DESCRIPTOR RECORD.
6. CHECK FOR SYSLST ASSIGNMENT ('INITSYSL').
7. INITIALIZE RETURNCODE 'RTCODE' TO ZERO.

INITSLDL

GET POINTERS TO SECOND LEVEL DIRECTORY AND SYSTEM DIRECTORY LIST

1. MAKE CORRESPONDING FETCHABLE ENTRY ADDRESSABLE.
2. IF WORKING ON SYSTEM LIBRARY 'SYSRES', THEN GET SLD AND SDL POINTERS FROM BG-FETCHABLE ENTRY. FOR A CONDENSE REQUEST INCREASE CONDENSE COUNTER BG-FETCHTABLE ENTRY.
3. FOR PRIVATE CORE IMAGE LIBRARY OBTAIN POINTER TO PRIVATE SECOND LEVEL DIRECTORY (SLD). FOR A CONDENSE REQUEST INCREASE CONDENSE COUNTER IN PARTITION FETCHTABLE ENTRY.

INITSYSL

CHECK FOR SYSLST ASSIGNMENT

1. CHECK IF SYSLST IS ASSIGNED AND SET 'SWASGN' TO 'YES' OR 'NO' ACCORDINGLY.

SDLSVA

SDL AND SVA BRINGUP FOR IPL AND JCL

IPL REQUEST:

IPL STORES THE DIRECTORY ENTRIES FOR THE SYSTEM PHASES IN SORTED AND COMPLETE FORM IN THE SDL AREA. IPL COMPUTES THE SPACE REQUIRED FOR THE FINAL AND COMPLETE SDL AND INITIALIZES THE HEADER OF SDL.

$MAINDIR LOADS ALL REQUIRED PHASES IN THE SVA, UPDATES THE SDL ENTRIES, AND GENERATES MESSAGES IF NECESSARY.

JCL REQUEST:

THE COMPLETE SDL WILL BE SORTED, SDL ENTRIES WILL BE COMPLETED BY INFORMATION FROM THE DISK AND THE PHASES WILL BE LOADED'

IF NECESSARY MESSAGES WILL BE GENERATED, THE SDL ITSELF SERVES AS INDICATOR WHICH MESSAGES SHOULD BE GENERATED AFTER A SDL BUILD OR MAINTENANCE RUN.

Chart 51. $MAINDIR (Part 4 of 10)

SORTSDL

SORT AND COMPLETE THE SYSTEM DIRECTORY LIST

CHECK IF THE NEW ADDED ENTRIES ARE ALREADY IN SDL. IF THEY EXIST ALREADY MAKE THEM TO END ENTRIES.

SORT NEW JCL ENTRIES TOGETHER WITH ALREADY EXISTING IPL AND JCL ENTRIES ON PHASENAME.

AFTER SORT DELETE DUPLICATE END ENTRIES.

INPUT: THE NUMBER OF ELEMENTS IN THE ARRAY IS FOUND IMMEDIATELY BEFORE THE SVA STOWTABLE HEADER IN THE FIELD 'TBSNE'.

MERGESDL

COMPLETE SDL SKELETON BY FILLING IN DIRECTORY INFORMATION

REALLOC

UPDATE OF DIRECTORIES AFTER A REALLOCATION OF THE SYSTEM CORE IMAGE LIBRARY

1. RE-WRITE SYSTEM CORE IMAGE DIRECTORY COMPLETELY, UPDATING ALL TTR'S WITH THE REALLOCATION FACTOR SUPPLIED BY MAINT IN A STOWTABLE WITH ONLY 1 ENTRY (TYPE 'REALLOC').
2. UPDATE THE TTR'S OF ALL SYSTEM DIRECTORY LIST ENTRIES.
3. UPDATE RAS LOADLIST IN THE SUPERVISOR.

OPTLINK

WRITE ENTRIES IN LAST DIRECTORY TRACK FOR '// OPTION LINK, // EXEC LNKEDT'

SERVES A REQUEST FROM THE LINKAGE EDITOR FOR A TEMPORARY LINK'
THE LINKAREA IS AT THE END OF THE CIL DIRECTORY. IT CONSISTS OF 8 BLOCKS, STARTING ON TRACK BOUNDARY. ONE BLOCK IS 256 BYTES LONG, IT STARTS WITH 2 BYTES WHICH GIVE THE NUMBER OF BYTES USED IN THE BLOCK.
(THIS NUMBER INCLUDES THE LENGTH OF THE 2 BYTES.)
AFTER THIS 2 BYTES THERE CAN BE UP TO 8 ENTRIES, EACH ONE WITH A CONSTANT LENGTH OF 30 BYTES.

THE BUFFER AREA USED FOR THIS PROCESS IS AN OVERLAY TO THE INPUT BUFFER AREA, WHICH IS NOT NEEDED FOR OPTION LINK AFTER THE 'TABIN' ENTRIES HAVE BEEN BUILT UP.

LCCALL

LAST CALL OF CONDENSE TO HAVE THE LIBRARY DESCRIPTOR WRITTEN OUT

1. CLEAR THE 'DESTRYD' IN THE LIBRARY DESCRIPTOR AND WRITE IT ON DISK.

Chart 52. $MAINDIR (Part 5 of 10)

MAINT

UPDATE ALL DIRECTORIES

BUILD SORTED ARRAY 'TABIN', CONTAINING PHASENAMES, TYPECODES AND POINTERS TO STOWTABLE ENTRIES.
START READING THE DIRECTORY AT THE FIRST BLOCK WITH A KEY HIGHER THEN OR EQUAL TO THE FIRST ENTRY IN ARRAY 'TABIN'.
FROM THIS POINT ON THE DIRECTORY WILL BE REWRITTEN COMPLETELY.
'MERGE' THE ARRAY 'TABIN' WITH THE DIRECTORY TAKING THE FOLLOWING ACTIONS:

1. IN CASE OF EQUALITY BETWEEN STOWTABLE AND DIRECTORY:

TYPE ACTION

CATAL – ADD STOWTABLE ENTRY TO OUTPUT RECORD, UPDATE LIBRARY DESCRIPTOR ENTRY, BUMP DIRECTORY INPUT POINTER. UPDATE RAS LOADLIST, UPDATE SYSTEM DIRECTORY LIST AND SHARED VIRTUAL AREA IF APPLICABLE.

UPDATE – ADD STOWTABLE ENTRY TO OUTPUT RECORD, BUMP DIRECTORY INPUT POINTER. UPDATE RAS LOADLIST AND SYSTEM DIRECTORY LIST IF APPLICABLE.

DELETE – UPDATE LIBRARY DESCRIPTOR ENTRY AND, IF APPLICABLE, THE RAS LOADLIST, THE SYSTEM DIRECTORY LIST AND THE SVA.
THEN BUMP DIRECTORY INPUT POINTER.

RENAME – SAME AS DELETE (SEE INTRODUCTORY COMMENTS OF SEGMENT EXTR), BUT WITHOUT CHANGING THE DESCRIPTOR ENTRY.

2. IF NO EQUALITY BETWEEN STOWTABLE AND DIRECTORY:

CATAL – SAME AS WITH EQUALITY, BUT WITHOUT BUMPING INPUT DIRECTORY POINTER, AND WITH OTHER UPDATING OF LIBRARY DESCRIPTOR.

UPDATE – ERROR. GENERATE DUMP.

DELETE – ERROR. ISSUE MESSAGE PHASE NOT FOUND.

SPECIAL – SAME AS CATAL, BUT WITHOUT CHANGING THE LIBRARY DESCRIPTOR ENTRY (SEE INTRODUCTORY COMMENTS OF SEGMENT 'EXTR').

RENM

FOR // EXEC MAINT RENAMC

BLOWUP

DELETE PRIVATE CORE IMAGE LIBRARY COMPLETELY, AS REQUESTED BY A 'DELETC ALL' COMMAND.

Chart 53. $MAINDIR (Part 6 of 10)

SPACETST

CHECK IF THERE IS ENOUGH SPACE FOR ENTRIES TO BE ADDED

WHEN ENTRIES ARE ADDED TO THE CORE IMAGE DIRECTORY AN OVERFLOW CAN OCCUR.

TO ENSURE INTEGRITY A FICTIVE UPDATE WILL BE PERFORMED TO TEST IF THERE IS ENOUGH SPACE IN THE DIRECTORY (THIS IS CALLED DRY RUN, THE DIRECTORY WILL NOT BE MODIFIED IN ANY WAY).

IT IS NOT POSSIBLE TO COMPUTE THE SPACE NEEDED IN ADVANCE BECAUSE THE DIRECTORY ENTRIES HAVE VARIABLE LENGTH.
BECAUSE THE DIRECTORY IS SORTED ON PHASENAMES ONE DOESN'T KNOW HOW THE FINAL LAYOUT WILL BE WITHOUT THIS DRY RUN.

RUN

UPDATE THE CORE IMAGE DIRECTORY WITH THE INFORMATION FROM THE STOWTABLE

NOTE: AFTER INITIALIZATION ALL INPUT BUFFERS HAVE BEEN FILLED. 'INPTR' POINTS TO THE FIRST APPLICABLE INPUT ENTRY. 'OUTPTR' POINTS TO THE AREA WHERE TO MOVE FIRST APPLICABLE OUTPUT ENTRY.

CHKEQL

CHECK EQUALITY BETWEEN CURRENT STOWTABLE ENTRY AND DIRECTORY INPUT, SYSTEM DIRECTORY LIST AND RAS LOADLIST.
INFORMATION ABOUT EQUALITY IS STORED IN 3 SWITCHES.

RASEQL

CHECK FOR RAS TRANSIENT

1. CHECK IF THIS IS A RAS TRANSIENT.
2. CALCULATE RAS LOADLIST INDEX.
3. CONVERT TTR FROM STOWTABLE ENTRY TO C–H–R DISK ADDRESS.

PROCESS

SELECT 'CATALOG', 'UPDAT' OR DELET' FOR EXECUTION.

Chart 54. $MAINDIR (Part 7 of 10)

CATALOG

ADD 1 STOWTABLE ENTRY TO DIRECTORY OUTPUT.

UPDAT

UPDATE DIRECTORIES FOR CONDENSE

1. IF PHASENAME NOT IN DIRECTORY GENERATE DUMP (SYSTEM ERROR).
2. REPLACE DIRECTORY ENTRY BY NEW ONE FROM STOWTABLE.

FINISH

CLEANUP BEFORE RETURN

1. DEQUEUE FETCH REQUESTS, UNLESS CONDENSE OR REALLOCATION IN PROCESS.
2. GENERATE MESSAGES AFTER A 'MAINT' RUN.
3. LOAD PHASES IN THE SHARED VIRTUAL AREA (IF NECESSARY).
4. RESTORE STORAGE KEY TO ORIGINAL VALUE.

SCANSDL

SCAN SYSTEM DIRECTORY LIST TO LOAD PHASES INTO THE SVA.

LOAD

LOAD 1 PHASE IN THE SHARED VIRTUAL AREA (SVA).

SVAMSG

GENERATE SDL AND SVA MESSAGES

GENERATE MESSAGES FOR ERROR SITUATIONS DETECTED DURING SDL UPDATE OR SVA BUILD PROCESS.

INPUT: SDL ENTRIES WITH FLAGS IN SWITCH BYTE.
SWITCH 'SWSVAFUL' AND SAVED NAME IN 'ARGUMENT'

OUTPUT: MESSAGES ON SYSLOG OR SYSLST.

GENMSG

GENERATE MESSAGES

GENERATE THESE MESSAGES WHICH COULD NOT BE DISPLAYED BECAUSE FETCH WAS NOT POSSIBLE.

INPUT: MESSAGE BITS IN 'TABIN'.
MSGCOUNT.

OUTPUT: MESSAGES ON SYSLOG OR SYSLST.

FUNCTION: SCAN 'TABIN' AND GENERATE MESSAGES.

Chart 55. $MAINDIR (Part 8 of 10)

DELET

FUNCTION: HONOUR DELETE REQUEST BY SKIPPING INPUT DIRECTORY ENTRY AND UPDATE LIBRARY DESCRIPTOR ACCORDINGLY. ISSUE DIAGNOSTIC MESSAGE IF THE APPLICABLE ENTRY IS NOT PRESENT IN INPUT.

NOTE 1: FOR A DELETE 'PROG.ALL' REQUEST MORE THAN ONE ENTRY MAY BE SKIPPED.

NOTE 2: IF AN ENTRY IN THE SYSTEM DIRECTORY LIST EXISTS FOR THIS PHASE IT IS CLEARED TO BINARY ZEROES. ONLY THE NAME REMAINS THERE, THE NOT FOUND BIT IS TURNED ON, AND THE STOWTYPE IS SAVED.

NOTE 3: IF A RAS TRANSIENT IS DELETED (PHASENAMES IN RANGE $$RAST00 – $$RAST99), THE DISK ADDRESS IN THE RAS LOADLIST IN THE SUPERVISOR IS MADE INVALID.

MSG

DISPLAY 'MSGAREA' ON SYSLST IF ASSIGNED, OTHERWISE ON SYSLOG.

LOOKUP

SCAN DIRECTORY TO FIND A PHASENAME

1. READ FIRST DIRECTORY BLOCK WITH KEY HIGHER THAN OR EQUAL TO 'ARGUMENT'.
2. CHECK IF THE PHASENAME IN 'ARGUMENT' EXISTS IN THE CORE IMAGE DIRECTORY. 'FOUND' IN 'SWITCHES' IS SET TO 'YES' OR 'NO'.
3. 'DIRPTR' WILL POINT TO THE DIRECTORY ENTRY WHERE THE SCAN STOPPED.

EXTR

BUILD ARRAY 'TABIN', CONTAINING PHASENAMES, TYPECODES AND POINTERS TO STOWTABLE ENTRIES.

INPUT: STOWTABLE

LAYOUT OF TABIN: ENTRIES CONSISTING OF
8 BYTES PHASENAME
1 BYTE TYPECODE
3 BYTES POINTER TO ORIGINAL ENTRY
2 BYTES INFORMATION WHICH MESSAGES SHOULD BE DISPLAYED
ENTRIES SORTED ON PHASENAME.

LOCATION OF TABIN: THE AREA POINTED TO BY TABREG AS INITIALIZED BY THE GETMAIN PROGRAMMER MACRO EXPANSION.

SPECIAL CASE: IN ARRAY 'TABIN' TWO ENTRIES ARE CREATED FOR A 'RENAME' TYPE 'STOWTAB' ENTRY.
1. 'RENAME' WITH THE OLD NAME
2. 'SPECIAL' WITH THE NEW NAME
BOTH POINTING TO THE SAME 'STOWTAB' ENTRY.

NOTES ABOUT 'RENAME' AND 'SPECIAL' PROCESSING:
BEFORE THESE 2 ENTRIES ARE CREATED 2 CHECKS ARE MADE:
1. 'OLDNAME' MUST BE IN THE DIRECTORY, OTHERWISE A DIAGNOSTIC MESSAGE WILL BE GIVEN: PHASE 'OLDNAME' NOT IN LIBRARY, AND THE RENAME REQUEST IS NOT PROCESSED.
2. 'NEWNAME' MUST NOT BE IN THE DIRECTORY, OTHERWISE A DIAGNOSTIC MESSAGE WILL BE GIVEN: PHASE 'NEWNAME' ALREADY IN LIBRARY, AND THE RENAME REQUEST WILL NOT BE PROCESSED.

THE INFOR FROM THE 'OLDNAME' DIRECTORY ENTRY IS SAVED IN THE STOWTABLE ENTRY, WHICH HAS ALWAYS THE MAXIMUM SIZE (30 BYTES).

Chart 56. $MAINDIR (Part 9 of 10)

SORT

SORT ARRAY 'TABIN' ON PHASENAME.

METHOD: SHELLSORT.
INPUT: THE NUMBER OF ELEMENTS IN THE ARRAY IS FOUND IN THE STOWTABLE HEADER FIELD 'NROFENTR'.

GETENTRY

1. PROVIDE AN ADDRESS IN 'INPTR' OF AN INPUT DIRECTORY ENTRY.
2. IF SWFIRST='YES' THEN INITIALIZE DIRECTORY INPUT.

PUTENTRY

1. WHEN CALLED THE FIRST TIME:
   SET UP DIRECTORY OUTPUT CCB AND CCW'S, INITIALIZE POINTERS TO CCW STRING AND OUTPUT BUFFER, SET UP DATA FIELD WITH BINARY ZEROES AND A BYTE COUNT OF 2, CALCULATE RELATIVE BLOCKNUMBER-1 OF FIRST DIRECTORY BLOCK READ AND INSERT IT INTO SAVED LIBRARY DESCRIPTOR ENTRY (FIELD 'DESDU'), MAKE FIELD 'DESDA' (# OF BLOCKS AVAILABLE) CONSISTENT WITH 'DESDU' BY INSERTING THE DIFFERENCE BETWEEN TOTAL # OF BLOCKS AND 'DESDU', INITIALIZE OUTPUT DISK ADDRESS WITH ADDRESS OF FIRST INPUT BLOCK.
2. FOR NORMAL CALLS:
   IF PAST END OF BUFFER THEN WRITE THE BLOCK WITH A WRITE KEY AND DATA COMMAND, CHAINED TO A READ COUNT MULTIPLE TRACK, BUMP 'DESDU' BY 1 AND 'DESDA' BY −1, SET UP DATA FIELD WITH BINARY ZEROES AND A BYTE COUNT OF 2. RESET 'OUTPTR' TO START OF BLOCK. UPDATA SLD IF APPLICABLE.

   DEPENDING ON SWITCH 'SWINPTR' MOVE CURRENT INPUT ENTRY OR CURRENT STOWTABLE ENTRY TO OUTPUT BUFFER, MOVE PHASENAME TO 'KEYOUT' AND BUMP BYTECOUNT BY LENGTH OF OUTPUT ENTRY. INCREASE 'OUTPTR' WITH LENGTH OF CURRENT DIRECTORY ENTRY.

   FOR THE LAST (END) DIRECTORY ENTRY: WRITE OUT THE BLOCK WITH THE END ENTRY AND FILL REST OF TRACK WITH EMPTY BLOCKS, ALL HAVING A KEY OF 16 X'F'.

GETINP

1. ONLY THE FIRST TIME: BUILD CCW STRING TO FILL ALL INPUT BUFFERS.
2. INITIALIZE CCB AND FILL ALL EMPTY BUFFERS.

PUTOUTP

1. WRITE ONE OUTPUT BLOCK (KEY AND DATA).
2. FOR LAST BLOCK OF TRACK: UPDATE SECOND LEVEL DIRECTORY.

CLOSE

WRITE UPDATED LIBRARY DESCRIPTOR

1. READ FIRST DIRECTORY BLOCK AGAIN.
2. REPLACE OLD LIBRARY DESCRIPTOR ENTRY BY NEW VERSION.
3. RE-WRITE FIRST DIRECTORY BLOCK.

Chart 57. $MAINDIR (Part 10 of 10)

RASCONV1

CONVERT THE 2-BYTE FIELD 'RASARG1' FROM CHARACTER FORMAT TO BINARY FORMAT AND ADD 1 TO MAKE IT USABLE AS INDEX INTO THE RAS LOADLIST IN THE SUPERVISOR.

RASCONV2

CONVERT TTR TO PRE-RELEASE 29 'CHR' FORMAT.
INPUT AND OUTPUT IN FIELD 'RASARG2'.
THE 'R' PART OF THE FIELD IS NOT TOUCHED.

DOIO

INPUT: REGISTER 1 POINTS TO A CCB, FOR WHICH I/O SHOULD BE PERFORMED.

FUNCTION: DO READ OR WRITE INCLUDING ALL CHECKS.

FOR CONDENSE OR REALLOCATION THE SEARCH ARGUMENT HAS TO BE ALWAYS PRESENT. THEREFORE A RETRY ON NO RECORD FOUND WILL BE DONE. AFTER 10 TIMES OF RETRY AN ERROR WILL INDICATED BY A RETURN CODE OF 16.

KEEPDIST

KEEP DISTANCE

WHEN LARGE NUMBERS OF PHASES ARE CATALOGED IT WILL BE NECESSARY SOMETIMES TO WRITE A DIRECTORY BLOCK BEFORE IT HAS BEEN READ. AS THIS WOULD CAUSE PART OF THE DIRECTORY TO BE DUPLICATED, AND ANOTHER PART TO BE SKIPPED, PRECAUTIONS MUST BE TAKEN.

1. SHIFT THE CURRENT INPUT BUFFER AND ALL BUFFERS FOLLOWING OVER ALL INPUT BUFFERS WHICH HAVE BEEN PROCESSED ALREADY.
2. FILL INPUT BUFFERS FREED THIS WAY WITH NEW DIRECTORY INFO.

UPDSLD

UPDATE SECOND LEVEL DIRECTORY

1. INSERT KEY OF LAST DIRECTORY BLOCK ON A TRACK IN THE APPLICABLE SECOND LEVEL DIRECTORY ENTRY.
2. INSERT 16 X'F' IN LAST SLD ENTRY.
3. ISSUE WARNING MESSAGE 3M91I IF MORE DIRECTORY TRACKS ARE USED THAN SLD ENTRIES EXIST.

MSG91I

PREPARE WARNING MESSAGE 3M91I ONLY ONCE PER JOBSTEP.

Chart 58. $MAINDIF (Part 1 of 12)

$MAINDIF Overview:

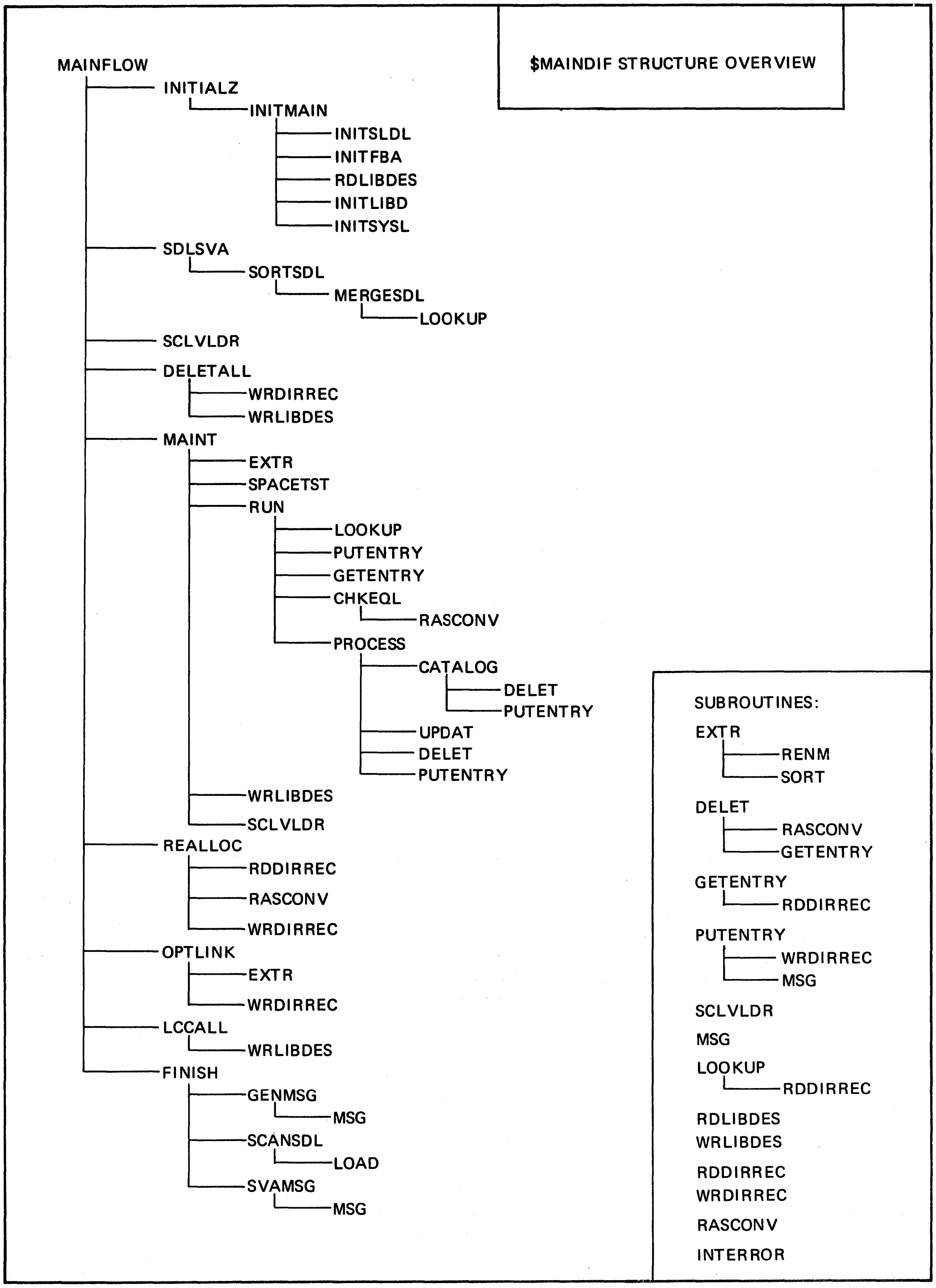


Chart 59. $MAINDIF (Part 2 of 12)

$MAINDIF Processing:

INITIALZ

INITIALIZATION FOR $MAINDIF

1. SET UP LOCAL STOW TABLE AND BUFFER POINTERS.
2. MODIFY LAST BYTE USED TO FORCE PROTECTION EXCEPTION.
3. OBTAIN STORAGE KEY 0.
4. STORE TIME OF DAY CLOCK CONTENTS IN 'TIME'.
5. MAIN INITIALIZATION ('INITMAIN').
6. INITIALIZE SDL-POINTER AND COUNTERS.

INITMAIN

MAIN INITIALIZATION

1. CLEAR ALL SWITCHES AND INITIALIZE POINTERS.
2. GET POINTERS TO SECOND LEVEL DIRECTORY AND SYSTEM DIRECTORY LIST ('INITSLDL').
3. FOR IPL GO TO 9.
4. INSPECT FIRST STOW TABLE ENTRY. IF TYPE IS 'LINK' OR 'CATAL' THEN INSERT THIS PHASENAME INTO THE GETVIS AREA FOR AN '// EXEC' USE.
5. RELOCATION AND INITIALIZATION OF I/O CONTROL BLOCKS AND COMMANDS ('INITFBA').
6. READ LIBRARY DESCRIPTOR RECORD ('RDLIBDES').
7. INITIALIZATION OF CONTROL BLOCKS WHICH NEED VALUES OF THE LIBRARY DESCRIPTOR RECORD ('INITLIBD').
8. INITIALIZE BUFFER VALUES.
9. INITIALIZE RETURN CODE 'RTCODE' TO ZERO.
10. CHECK FOR SYSLST ASSIGNMENT ('INITSYSL').

INITSYSL

CHECK FOR SYSLST ASSIGNMENT

1. CHECK IF SYSLST IS ASSIGNED AND SET 'SWASGN' TO 'YES' OR 'NO' ACCORDINGLY.

INITSLDL

GET POINTERS TO SECOND LEVEL DIRECTORY AND SYSTEM DIRECTORY LIST

1. MAKE CORRESPONDING FETCHTABLE ENTRY ADDRESSABLE.
2. IF WORKING ON SYSTEM LIBRARY 'SYSRES', THEN GET SLD AND SDL POINTERS FROM BG-FETCHTABLE ENTRY. FOR A CONDENSE REQUEST INCREASE CONDENSE COUNTER BG-FETCH-TABLE ENTRY.
3. FOR PRIVATE CORE IMAGE LIBRARY OBTAIN POINTER TO PRIVATE SECOND LEVEL DIRECTORY (SLD). FOR A CONDENSE REQUEST INCREASE CONDENSE COUNTER IN PARTITION FETCHTABLE ENTRY.

INITFBA

RELOCATION AND INITIALIZATION OF ALL I/O CONTROL BLOCKS AND COMMANDS

1. TRANSFER 'STATIC LOCAL' DATA INTO 'AUTOMATIC' AREA.
2. RELOCATE ADDRESSES AND INITIALIZE IORB'S AND CCW'S.

INITLIBD

INITIALIZATION OF CONTROL BLOCKS WHICH NEED VALUES OF THE LIBRARY DESCRIPTOR RECORD.

Chart 60. $MAINDIF (Part 3 of 12)

SDLSVA

SDL AND SVA BRINGUP FOR IPL AND JCL

IPL REQUEST:

IPL STORES THE DIRECTORY ENTRIES FOR THE SYSTEM SVA PHASES IN SORTED AND COMPLETE FORM IN THE SDL AREA' IPL COMPUTES THE SPACE REQUIRED FOR THE FINAL AND COMPLETE SDL AND STORES IT IN THE HEADER OF THE SVA.

THIS SPACE HAS TO BE 12 BYTES LONGER THAN THE COMPUTER VALUE. THESE 12 BYTES ARE TEMPORARY BYTES USED FOR THE IPL STOW TABLE AND FOR JCL.

$MAINDIF LOADS ALL REQUIRED PHASES INTO THE SVA AND UPDATES THE HEADER OF THE SVA ACCORDINGLY IN SEGMENT 'LOAD'.

JCL REQUEST:

SORT COMPLETE SYSTEM DIRECTORY LIST.
MERGE ENTRIES WITH DIRECTORY, FILLING IN DIRECTORY INFORMATION, AND INCREMENT 'SDLBCNT' IF REQUEST ENCOUNTERED TO LOAD PHASE INTO VIRTUAL LIBRARY.

1. INSERT AND ENTRY (FF..FF) AT END OF THE USED SDL.
2. SORT AND COMPLETE THE SYSTEM DIRECTORY LIST FOR JCL ('SORTSDL').
3. FOR IPL SAVE NUMBER OF PHASES LATER TO BE LOADED INTO SVA.

SORTSDL

SORT AND COMPLETE THE SYSTEM DIRECTORY LIST

CHECK IF THE NEW ADDED ENTRIES ARE ALREADY IN SDL. IF THEY EXIST ALREADY MAKE THEM TO END ENTRIES.

SORT NEW JCL ENTRIES TOGETHER WITH ALREADY EXISTING IPL AND JCL ENTRIES ON PHASENAME.

DELETE DUPLICATE END ENTRIES.

IF NEW ADDED ENTRIES EXIST' INCLUDE 'MERGESDL'.

INPUT: THE NUMBER OF ELEMENTS IN THE ARRAY IS FOUND IN THE 'TBSNE'.
JCL STORES ITS ENTRIES AFTER THE EXISTING ENTRIES IN SDL.
ONE STOW TABLE HEADER IS USED FOR JCL AND IPL (INITIALIZED BY IPL, AFTER 'TBSNE').

MERGESDL

MERGE SDL WITH CORE IMAGE DIRECTORY

COMPLETE SDL SKELETON BY FILLING IN DIRECTORY INFORMATION AND INCREMENT 'SDLBCNT' IF A BUILD REQUEST WAS ISSUED.

Chart 61. $MAINDIF (Part 4 of 12)

REALLOC

UPDATE DIRECTORIES AFTER A REALLOCATION OF THE SYSTEM CORE IMAGE LIBRARY

1. RE-WRITE SYSTEM CORE IMAGE DIRECTORY COMPLETELY, UPDATING ALL ADDRESSES WITH THE REALLOCATION DISPL. SUPPLIED BY MAINT IN A STOW TABLE WITH ONLY 1 ENTRY (TYPE 'REALLOC') BY 'RDDIRREC' AND 'WRDIRREC'.
2. UPDATE THE ADDRESSES OF ALL SYSTEM DIRECTORY LIST ENTRIES (FOR FBA ALL ADDRESSES ARE BLOCKNUMBERS).
3. UPDATE RAS LOADLIST IN THE SUPERVISOR ('RASCONV').

OPTLINK

WRITE ENTRIES IN LAST DIRECTORY RECORD FOR '// OPTION LINK'

THE LINKAGE EDITOR CALLS TO BUILD A LINK DIRECTORY. A NUMBER OF PHASES WILL BE TEMPORARILY STORED IN THE SYSTEM OR PRIVATE LIBRARY. THE LINK DIRECTORY FOR THESE PHASES IS IN ONE DIRECTORY RECORD AND CONSISTS ONLY OF ENTRIES, WITHOUT A DESCRIPTOR RECORD. THIS DIRECTORY IS SORTED BY ALPHABET.

THE LINK DIRECTORY IS ALWAYS THE LAST RECORD IN THE DIRECTORY SPACE ALLOCATED.

1. SPECIAL HANDLING FOR ONLY ONE PHASE.
2. BUILD STOW TABLE EXTRACT IN TABIN ('EXTR').
3. WRITE LINK DIRECTORY RECORDS ONTO DISK ('WRDIRREC').
4. INSERT THE BEGIN OF THE LINK DIRECTORY IN THE FETCH TABLE.

LCCALL

FINAL CALL OF CONDENSE TO HAVE THE LIBRARY DESCRIPTOR WRITTEN OUT

1. RESET 'DSTROYD' TO ZERO.
2. UPDATE LIBRARY DESCRIPTOR.
3. WRITE LIBRARY DESCRIPTOR ONTO DISK ('WRLIBDES').
4. SET RETURN CODE.

DELETALL

DELETE PRIVATE CORE IMAGE LIBRARY COMPLETELY, AS REQUESTED BY A 'DELETC ALL' COMMAND

1. WRITE AN END ENTRY (DIRNME=FFFFFFFF) IN FIRST DIRECTORY BLOCK AND INITIALIZE U BYTES.
2. UPDATE LIBRARY DESCRIPTOR RECORD.
3. WRITE LIBRARY DESCRIPTOR RECORD ONTO DISK.
4. SET RETURN CODE.

Chart 62. $MAINDIF (Part 5 of 12)

MAINT

UPDATE ALL DIRECTORIES

BUILD SORTED ARRAY 'TABIN', CONTAINING PHASENAMES, STOWTYPES AND POINTERS TO STOW TABLE ENTRIES.
START READING THE DIRECTORY AT THE FIRST BLOCK WITH A NAME HIGHER THAN OR EQUAL TO THE FIRST ENTRY IN ARRAY 'TABIN'.
FROM THIS POINT ON THE DIRECTORY WILL BE REWRITTEN COMPLETELY.
'MERGE' THE ARRAY 'TABIN' WITH THE DIRECTORY TAKING THE FOLLOWING ACTIONS:

1. IF NAME IN 'TABIN' ENTRY EXISTS IN A DIRECTORY ENTRY:

CATAL:

ADD STOW TABLE ENTRY TO OUTPUT RECORD, UPDATE LIBRARY DESCRIPTOR RECORD, BUMP DIRECTORY INPUT POINTER.
UPDATE RAS LOADLIST, UPDATE SYSTEM DIRECTORY LIST AND SHARED VIRTUAL AREA IF APPLICABLE.

UPDATE:

ADD STOW TABLE ENTRY TO OUTPUT RECORD, BUMP DIRECTORY INPUT POINTER, UPDATE RAS LOADLIST AND SYSTEM DIRECTORY LIST IF APPLICABLE.

DELETE:

UPDATE LIBRARY DESCRIPTOR RECORD AND, IF APPLICABLE, THE RAS LOADLIST, THE SYSTEM DIRECTORY LIST AND THE SVA.
THEN BUMP DIRECTORY INPUT POINTER.

RENAME:

SAME AS DELETE (SEE INTRODUCTORY COMMENTS OF EXTR), BUT WITHOUT CHANGING THE DESCRIPTOR RECORD.

2. IF NAME IN 'TABIN' ENTRY NOT FOUND IN THE DIRECTORY:

CATAL:

SAME AS WITH EQUALITY, BUT WITHOUT BUMPING INPUT DIRECTORY POINTER' AND WITH OTHER UPDATING OF LIBRARY DESCRIPTOR.

UPDATE – SYSTEM ERROR. GENERATE DUMP.

DELETE – ERROR. ISSUE MESSAGE PHASE NOT FOUND.

SPECIAL:

SAME AS CATAL, BUT WITHOUT CHANGING THE LIBRARY DESCRIPTOR ENTRY (SEE INTRODUCTORY COMMENTS OF 'EXTR').
THE STOW TABLE HAS TO BE TERMINATED BY AN END ENTRY, OTHERWISE THE ALGORITHM OF 'MAINT' WILL NOT WORK.

1. INITIALIZE NUMBER OF MESSAGES TO BE GENERATED AFTER THE MAINT RUN.
2. CALL 'EXTR' TO BUILD ARRAY 'TABIN'.
3. TEST IF THERE IS ENOUGH SPACE FOR NEW ENTRIES TO BE ADDED (DRYRUN CAN BE NECESSARY).
4. UPDATE THE CORE IMAGE DIRECTORY WITH THE INFORMATION FROM THE STOW TABLE ('RUN').
5. WRITE THE UPDATED LIBRARY DESCRIPTOR ONTO DISK ('WRLIBDES').
6. IF NECESSARY REFORMAT THE SECOND LEVEL DIRECTORY ('SCLVLDR').

Chart 63. $MAINDIF (Part 6 of 12)

RENM

RENAME OF PHASE IN CORE IMAGE LIBRARY

FOR FUNCTIONS SEE INITIAL COMMENTS IN 'EXTR'.

SORT

SORT ARRAY 'TABIN' ON PHASENAME

METHOD: SHELLSORT.
INPUT: THE NUMBER OF ELEMENTS IN THE ARRAY IS FOUND 'TABNOEN'.

SPACETST

TEST IF THERE IS ENOUGH SPACE FOR NEW ENTRIES TO BE ADDED

WHEN ENTRIES WILL BE ADDED ONE HAS TO ASSURE THAT THEY FIT INTO THE DIRECTORY.

THE NUMBER OF AVAILABLE ENTRIES IN THE DIRECTORY WILL BE COMPUTED (THIS IS POSSIBLE, BECAUSE ALL ENTRIES HAVE THE SAME LENGTH) AND COMPARED WITH THE NUMBER OF ENTRIES IN THE STOW TABLE.
IF THERE ARE ENOUGH ENTRIES IN THE DIRECTORY AVAILABLE, THE UPDATE CAN START.

IF THERE ARE NOT ENOUGH ENTRIES AVAILABLE IT CAN BE THAT PHASES WILL BE CATALOGED WHICH ARE ALREADY IN THE DIRECTORY, SO THESE ENTRIES WILL BE AVAILABLE TOO.

A METHOD OF CHECKING IF THERE IS ENOUGH SPACE IS THE 'DRYRUN'. THE DIRECTORY UPDATE IS SIMULATED IN THE DRYRUN (A SWITCH PREVENTS FROM REAL UPDATING).
THERE IS NO MODIFICATION OF THE DIRECTORY DURING THE DRYRUN.

IF THE DIRECTORY BECOMES FULL IN THE DRYRUN ('DIRECTORY TOO SMALL') THE JOB WILL BE CANCELLED, OTHERWISE THE RUN STEP PERFORMED.

RUN

UPDATE THE CORE IMAGE DIRECTORY WITH THE INFORMATION FROM THE STOW TABLE

0. QUEUE FETCH REQUESTS, BECAUSE DIRECTORY WILL BE MODIFIED. NO FETCH IS THEREFORE ALLOWED.
1. READ DIRECTORY RECORD CONTAINING ENTRY WITH NAME OF FIRST 'TABIN' ENTRY ('LOOKUP').
2. WRITE INAREA UNTIL 'DIRPTR' TO OUTPUT AREA ('PUTENTRY', 'GETENTRY').
3. 'INPTR' POINTS TO THE FIRST APPLICABLE INPUT ENTRY. 'OUTPTR' POINTS TO THE AREA WHERE TO MOVE THE FIRST APPLICABLE ENTRY.
4. CHECK FOR EQUALITY BETWEEN CURRENT STOW TABLE ENTRY AN DIRECTORY INPUT ('CHKEQL').
5. IF MERGE NOT COMPLETED CONTINUE WITH NEXT 'TABIN' ENTRY UNTIL ALL ENTRIES PROCESSED ('PROCESS').

Chart 64. $MAINDIF (Part 7 of 12)

CHKEQL

CHECK EQUALITY BETWEEN CURRENT STOW TABLE ENTRY AND DIRECTORY INPUT, SYSTEM DIRECTORY LIST, AND RAS LOADLIST.

PROCESS

SELECT 'CATALOG', 'UPDAT', OR 'DELET' FOR EXECUTION

CATALOG

ADD 1 STOW TABLE ENTRY TO DIRECTORY OUTPUT

UPDAT

HANDLE A NORMAL CONDENSE REQUEST

HANDLES A NORMAL CONDENSE REQUEST. 'SWEQUAL' SHOULD ALWAYS BE 'YES', BECAUSE EACH ENTRY TO BE UPDATED EXISTS IN THE DIRECTORY (AT LEAST BEFORE THE CONDENSE, CONDENSE ONLY CHANGES THE ADDRESS IN DIRECTORY ENTRY).

1. IF PHASENAME NOT IN DIRECTORY GENERATE A DUMP, SYSTEM ERROR DURING CONDENSE.
2. REPLACE DIRECTORY ENTRY BY NEW ONE FROM STOW TABLE.

FINISH

CLEANUP BEFORE RETURN

1. DEQUEUE FETCH REQUESTS.
2. GENERATE MESSAGES FOR ERROR SITUATIONS DETECTED DURING EXECUTION OF 'MAINT' ('GENMSG').
3. SCAN THE SYSTEM DIRECTORY LIST TO LOAD PHASES INTO THE SVA ('SCANSDL').
4. GENERATE MESSAGES FOR ERROR SITUATIONS DETECTED DURING SDL UPDATE OR SVA BUILD ('SVAMSG').
5. RESTORE STORAGE KEY TO ORIGINAL VALUE.

GENMSG

GENERATE MESSAGES

GENERATE MESSAGES FOR ERROR SITUATIONS DETECTED DURING EXECUTION OF 'MAINT'.

INPUT: MSGCOUNT .. TOTAL NUMBER OF MESSAGES.
TABIN .. MESSAGE BITS.

OUTPUT: MESSAGES ON SYSLOG OR SYSLST.

FUNCTION: SCAN TABIN AND GENERATE MESSAGES BY 'MSG'.

Chart 65. $MAINDIF (Part 8 of 12)

SVAMSG

GENERATE SDL AND SVA MESSAGES

GENERATE MESSAGES FOR ERROR SITUATIONS DETECTED DURING SDL UPDATE OR SVA BUILD.

INPUT: SDL ENTRIES.
SWSVAFUL.

OUTPUT: MESSAGES ON SYSLOG OR SYSLST.

FUNCTION: SCAN SDL AND GENERATE MESSAGES BY 'MSG'.

SCANSDL

SCAN THE SYSTEM DIRECTORY LIST TO LOAD PHASES INTO THE SVA

1. SCAN SYSTEM DIRECTORY LIST AND
2. LOAD PHASES IN THE SHARED VIRTUAL AREA (SVA) BY 'LOAD'.

LOAD

LOAD 1 PHASE INTO SHARED VIRTUAL AREA (SVA)

INTERNAL SUBROUTINES

SCLVLDR

BUILD SECOND LEVEL DIRECTORY (SLD, ALWAYS PRESENT FOR FBA CORE IMAGE LIBRARIES) FOR PRIVATE CORE IMAGE LIBRARY

REFORMATTING OF ALL SLD'S WHEN SCOPE OF SLD HAS BEEN EXCEEDED.

THE NUMBER OF ENTRIES IN SLD (SLDNE) IS ALREADY INITIATED DURING SUPERVISOR GENERATION.

1. COMPUTE NUMBER OF FBA-BLOCKS WITHIN ONE GROUP.
2. READ THE LAST RECORD WITHIN GROUP, GET LAST ENTRY (VIA U-BYTES) AND STORE ITS NAME IN THE SLD.

Chart 66. $MAINDIF (Part 9 of 12)

EXTR

BUILD ARRAY 'TABIN', CONTAINING PHASENAMES, STOWTYPES AND POINTERS TO STOW TABLE ENTRIES.

INPUT: STOW TABLE

LAYOUT OF TABIN:
ENTRIES CONSISTING OF
8 BYTES PHASENAME
1 BYTE STOWTYPE
3 BYTES POINTER TO ORIGINAL ENTRY IN STOW TABLE

ENTRIES WILL BE SORTED ON PHASENAME. AT THE END AN END ENTRY (F..F) IS ADDED.

LOCATION OF TABIN:
THE AREA POINTED TO BY IOREG, AS INITIALIZED BY THE GETMAIN PROGRAMMER MACRO EXPANSION.

SPECIAL CASE:
IN ARRAY 'TABIN' TWO ENTRIES ARE CREATED FOR A 'RENAME' TYPE 'STOWTAB' ENTRY.
1. 'RENAME' WITH THE OLD NAME
2. 'SPECIAL' WITH THE NEW NAME
BOTH POINTING TO THE SAME 'STOWTAB' ENTRY.

NOTES ABOUT 'RENAME' AND 'SPECIAL' PROCESSING:
BEFORE THESE 2 ENTRIES ARE CREATED 2 CHECKS ARE MADE:
1. OLD NAME MUST BE IN THE DIRECTORY, OTHERWISE A DIAGNOSTIC MESSAGE WILL BE GIVEN: PHASE 'OLDNAME' NOT IN LIBRARY, AND THE RENAME REQUEST IS NOT PROCESSED.
2. NEW NAME MUST NOT BE IN THE DIRECTORY, OTHERWISE A DIAGNOSTIC MESSAGE WILL BE GIVEN: PHASE 'NEWNAME' ALREADY IN LIBRARY, AND THE RENAME REQUEST WILL NOT BE PROCESSED.

THE INFO FROM THE 'OLDNAME' DIRECTORY ENTRY IS SAVED IN THE STOW TABLE ENTRY.

1. INITIALIZE VALUES FOR 'TABIN'.
2. EXTRACT FROM STOW TABLE INFORMATION FOR 'TABIN'.
   FOR A RENAME ENTRY INCLUDE 'RENM'.
3. INSERT END ENTRY OF 'TABIN'.
4. IF 'TABIN' IS NOT SORTED' INCLUDE 'SORT'.

DELET

SKIP INPUT DIRECTORY ENTRY AND UPDATE LIBRARY DESCRIPTOR ACCORDINGLY.
ISSUE DIAGNOSTIC MESSAGE IF THE APPLICABLE ENTRY IS NOT PRESENT IN INPUT.

NOTE 1:
FOR A DELETE 'PROG.ALL' REQUEST MORE THAN ONE ENTRY MAY BE SKIPPED'

NOTE 2:
IF AN ENTRY IN THE SYSTEM DIRECTORY LIST EXISTS FOR THIS PHASE IT IS CLEARED TO BINARY ZEROES. ONLY THE NAME REMAINS THERE, THE NOT FOUND BIT IS TURNED ON, AND THE STOWTYPE IS SAVED.

NOTE 3:
IF A RAS TRANSIENT IS DELETED (PHASENAMES IN RANGE $$RAST00 - $$RAST99), THE DISK ADDRESS IN THE RAS LOADLIST IN THE SUPERVISOR IS MADE INVALID.

MSG

DISPLAY 'MSGAREA' ON SYSLST IF ASSIGNED, OTHERWISE ON SYSLOG.

Chart 67. $MAINDIF (Part 10 of 12)

LOOKUP

SCAN DIRECTORY TO FIND 'LOOKNAME'

INPUT: VALUE IN 'LOOKNAME'.
LIBRARY DESCRIPTOR RECORD IN 'DESCRPT'.
LIBRARY DIRECTORY BLOCKS ON DISK.
SECOND LEVEL DIRECTORY.

FUNCTION: CHECK IF THE PHASENAME EXISTS IN THE CORE IMAGE LIBRARY AND SET 'FOUND' TO YES OR NO ACCORDINGLY.

OUTPUT:
- FOUND SWITCH
- DIRECTORY RECORD IN INPUT BUFFER.
- 'INPTR' IS POINTING TO FOUND ENTRY OR TO ENTRY WITH A NAME HIGHER OR = TO 'FF..FF' IF NOT FOUND.
- 'INBEG' CONTAINS BLOCKNUMBER OF FIRST RECORD READ IN BUFFER.

GETENTRY

PROVIDES A DIRECTORY ENTRY FOR A MERGE OPERATION DURING 'MAINT'

INITIALIZATION:
FILL ALL INPUT BUFFERS AND SET 'INPTR' TO BEGIN OF BUFFER AREA AFTER U-BYTES (IN 'RUN').
IN 'INBEG' IS BLOCKNUMBER OF RECORD FIRST READ IN (BY 'LOOKUP').

INPUT:

INPTR: POINTING TO ENTRY IN BUFFER TO BE PROCESSED.
INAREA: BUFFER.
INBEG: BLOCKNUMBER OF RECORD FIRST READ IN. DIRECTORY ON DISK.

FUNCTION:

1. BUMP 'INPTR' TO NEXT INPUT DIRECTORY ENTRY.
2. IF END OF RECORD REACHED' BUMP TO FIRST ENTRY IN NEXT RECORD.
3. IF END OF INPUT BUFFER REACHED REFILL INPUT BUFFER AND SET 'INPTR' TO BEGIN OF THE INPUT BUFFER AREA.

OUTPUT:

INPTR: BUMPED TO NEXT ENTRY, IF END OF BUFFER REACHED SET TO BEGIN OF BUFFER JUST FILLED WITH NEW RECORDS.
INRECPTR: POINTS TO BEGIN OF RECORD IN BUFFER.
INAREA: UNCHANGED OR FILLED WITH NEW RECORDS.
INBEG: UNCHANGED OR INCREASED BY LENGTH OF BUFFER.

Chart 68. $MAINDIF (Part 11 of 12)

PUTENTRY

WRITES A DIRECTORY ENTRY TO THE OUTPUT BUFFER AND WRITES THE OUTPUT BUFFER ONTO DISK WHEN IT IS FULL, OR WHEN THEY ARE COMPLETE

DURING DRYRUN NOTHING WILL BE MODIFIED.

INITIALIZATION:
ORECPTR = OPTR
OUTPTR = OPTR + DESLUBYT
OUTBEG = INBEG

INPUT:

INPTR: POINTS TO ENTRY TO BE TRANSFERRED TO OUT BUFFER (OUTFUNC = INA).
TABACT: ACTUAL INDEX OF 'TABIN' (OUTFUNC = TAB).
OUTFUNC: FUNCTION CODE.

FUNCTION:
1. STORE THE DIRECTORY ENTRY BASED BY INPTR OR TABACT THE OUTAREA BEGINNING AT 'OUTPTR'.
2. WHEN THE LAST OUTPUT BUFFER IS FILLED WRITE ALL OUTPUT BUFFERS ONTO DISK AND SET 'OUTPTR' TO BEGIN OF FIRST OUTPUT BUFFER.
3. UPDATE THE LIBRARY DESCRIPTOR RECORD.
4. TEST FOR FULL DIRECTORY.

RDLIBDES

READ LIBRARY DESCRIPTOR

INPUT: ADDRESS OF LIBRARY.
LIBRARY DESCRIPTOR RECORD ON DISK.

OUTPUT: DESCRIPTOR RECORD IN 'DESCRPT'.
MESSAGE FOR ERROR CASE.

WRLIBDES

WRITE LIBRARY DESCRIPTOR ONTO DISK

INPUT: LIBRARY DESCRIPTOR RECORD IN CORE.

OUTPUT: LIBRARY DESCRIPTOR ON DISK.

RDDIRREC

READ A UNIT OF DIRECTORY RECORDS INTO THE BUFFER AREA

INPUT:

RDBUFADR: ADDRESS OF BUFFER AREA USED.
RDBUFLEN: LENGTH OF BUFFER AREA IN RECORDS, NUMBER OF RECORDS READ IN.
RDBLKNR: RELATIVE BLOCKNUMBER OF FIRST RECORD TO BE READ IN.

OUTPUT: UNIT OF DIRECTORY RECORDS IN BUFFER AREA.

Chart 69. $MAINDIF (Part 12 of 12)

WRDIRREC

WRITE A UNIT OF DIRECTORY RECORDS FROM THE BUFFER AREA ONTO DISK

INPUT:

WRBUFADR: ADDRESS OF BUFFER AREA USED.
WRBUFLEN: LENGTH OF BUFFER AREA IN RECORDS, NUMBER OF RECORDS WRITTEN OUT.
WRBLKNR: RELATIVE BLOCKNUMBER WHERE FIRST RECORD HAS TO BE WRITTEN OUT.

OUTPUT: UNIT OF DIRECTORY RECORDS ON DISK AND RETURN CODE.

RASCONV

CONVERT THE 2-BYTE FIELD 'RASNR' FROM CHARACTER FORMAT TO BINARY FORMAT AND ADD 1 TO MAKE IT USABLE AS INDEX INTO THE RAS LOADLIST IN THE SUPERVISOR.

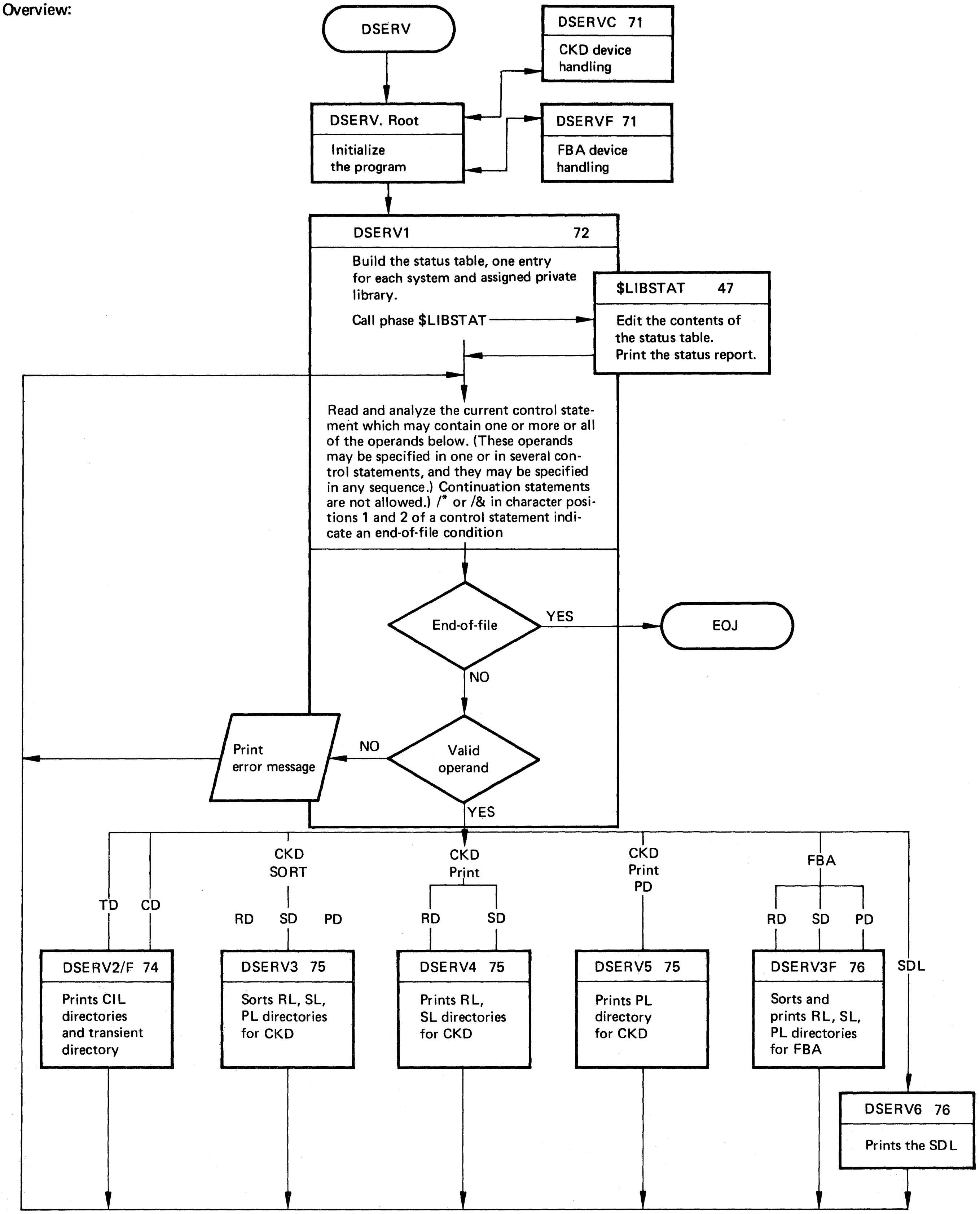
INTERROR

FOR SYSTEM ERRORS GENERATE A DUMP

INPUT: VALUE TO BE STORED IN REG. 15.

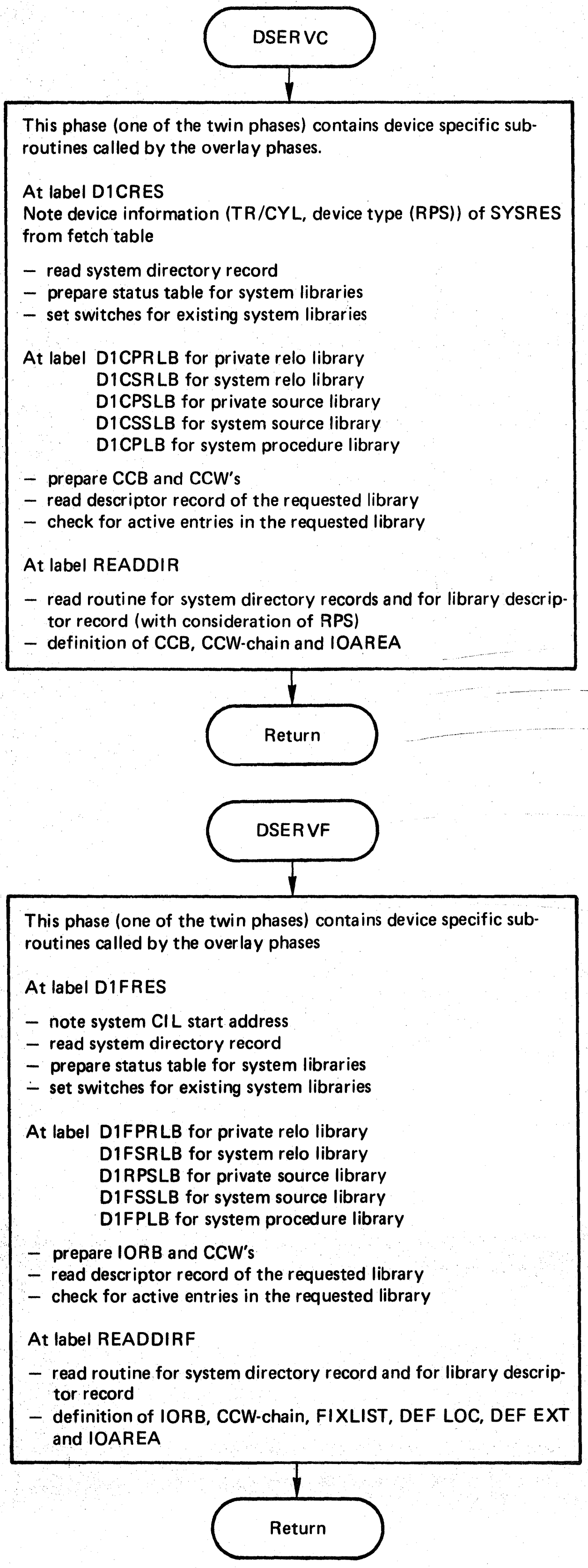
Chart 70. DSERV

Overview:



DSERV
DSERVC 71
CKD device handling
DSERV. Root
Initialize the program
DSERVF 71
FBA device handling
DSERV1 72
Build the status table, one entry for each system and assigned private library.
Call phase $LIBSTAT
$LIBSTAT 47
Edit the contents of the status table.
Print the status report.
Read and analyze the current control statement which may contain one or more or all of the operands below. (These operands may be specified in one or in several control statements, and they may be specified in any sequence.) Continuation statements are not allowed.) /* or /& in character positions 1 and 2 of a control statement indicate an end-of-file condition
End-of-file
YES
EOJ
NO
Valid operand
NO
Print error message
YES
TD
CD
CKD SORT
RD
SD
PD
CKD Print
RD
SD
CKD Print PD
FBA
RD
SD
PD
SDL
DSERV2/F 74
Prints CIL directories and transient directory
DSERV3 75
Sorts RL, SL, PL directories for CKD
DSERV4 75
Prints RL, SL directories for CKD
DSERV5 75
Prints PL directory for CKD
DSERV3F 76
Sorts and prints RL, SL, PL directories for FBA
DSERV6 76
Prints the SDL

Chart 71. DSERVC/DSERVF



DSERVC

This phase (one of the twin phases) contains device specific subroutines called by the overlay phases.

At label D1CRES
Note device information (TR/CYL, device type (RPS)) of SYSRES from fetch table

- read system directory record
- prepare status table for system libraries
- set switches for existing system libraries

At label D1CPRLB for private relo library
D1CSRLB for system relo library
D1CPSLB for private source library
D1CSSLB for system source library
D1CPLB for system procedure library

- prepare CCB and CCW's
- read descriptor record of the requested library
- check for active entries in the requested library

At label READDIR

- read routine for system directory records and for library descriptor record (with consideration of RPS)
- definition of CCB, CCW-chain and IOAREA

Return

DSERVF

This phase (one of the twin phases) contains device specific subroutines called by the overlay phases

At label D1FRES

- note system CIL start address
- read system directory record
- prepare status table for system libraries
- set switches for existing system libraries

At label D1FPRLB for private relo library
D1FSRLB for system relo library
D1RPSLB for private source library
D1FSSLB for system source library
D1FPLB for system procedure library

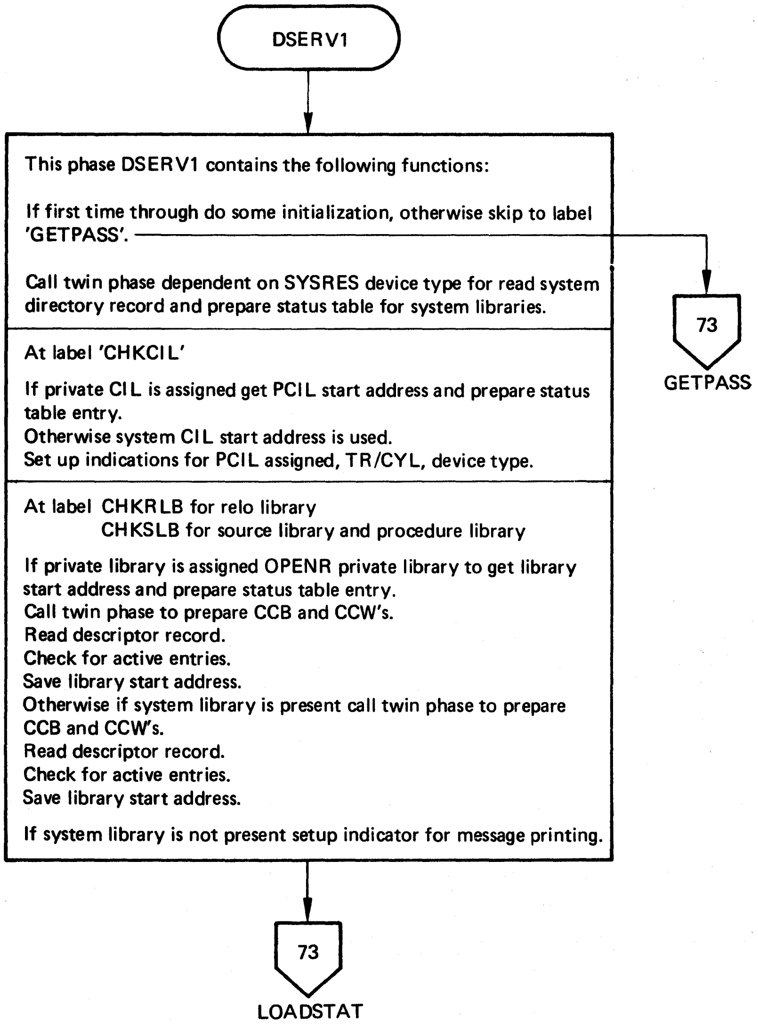
- prepare IORB and CCW's
- read descriptor record of the requested library
- check for active entries in the requested library

At label READDIRF

- read routine for system directory record and for library descriptor record
- definition of IORB, CCW-chain, FIXLIST, DEF LOC, DEF EXT and IOAREA

Return

Chart 72. DSERV1 (Part 1 of 2)
Detail Chart RA.



This phase DSERV1 contains the following functions:

If first time through do some initialization, otherwise skip to label 'GETPASS'.

Call twin phase dependent on SYSRES device type for read system directory record and prepare status table for system libraries.

At label 'CHKCIL'

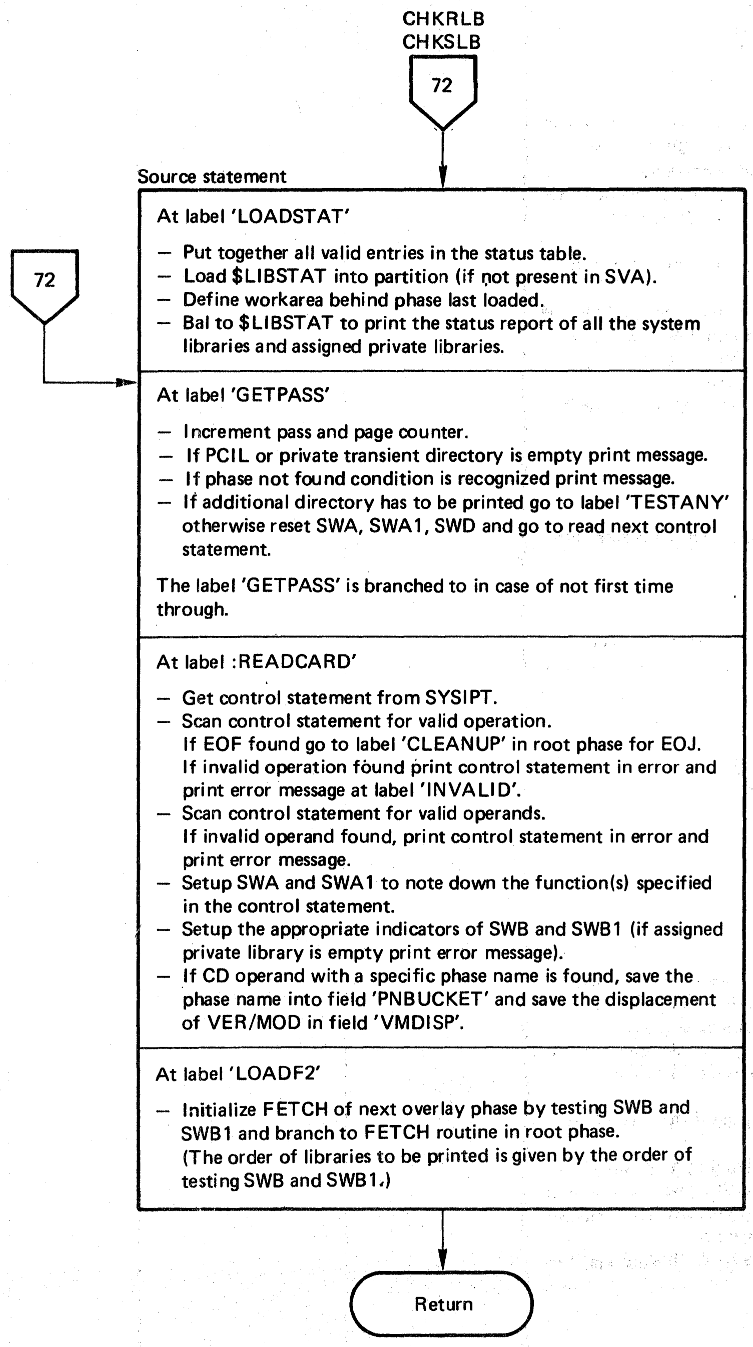
If private CIL is assigned get PCIL start address and prepare status table entry.
Otherwise system CIL start address is used.
Set up indications for PCIL assigned, TR/CYL, device type.

At label CHKRLB for relo library
CHKSLB for source library and procedure library

If private library is assigned OPENR private library to get library start address and prepare status table entry.
Call twin phase to prepare CCB and CCW's.
Read descriptor record.
Check for active entries.
Save library start address.
Otherwise if system library is present call twin phase to prepare CCB and CCW's.
Read descriptor record.
Check for active entries.
Save library start address.

If system library is not present setup indicator for message printing.

Chart 73. DSERV1 (Part 2 of 2)
Detail Chart RA.



At label 'LOADSTAT'

- Put together all valid entries in the status table.
- Load $LIBSTAT into partition (if not present in SVA).
- Define workarea behind phase last loaded.
- Bal to $LIBSTAT to print the status report of all the system libraries and assigned private libraries.

At label 'GETPASS'

- Increment pass and page counter.
- If PCIL or private transient directory is empty print message.
- If phase not found condition is recognized print message.
- If additional directory has to be printed go to label 'TESTANY' otherwise reset SWA, SWA1, SWD and go to read next control statement.

The label 'GETPASS' is branched to in case of not first time through.

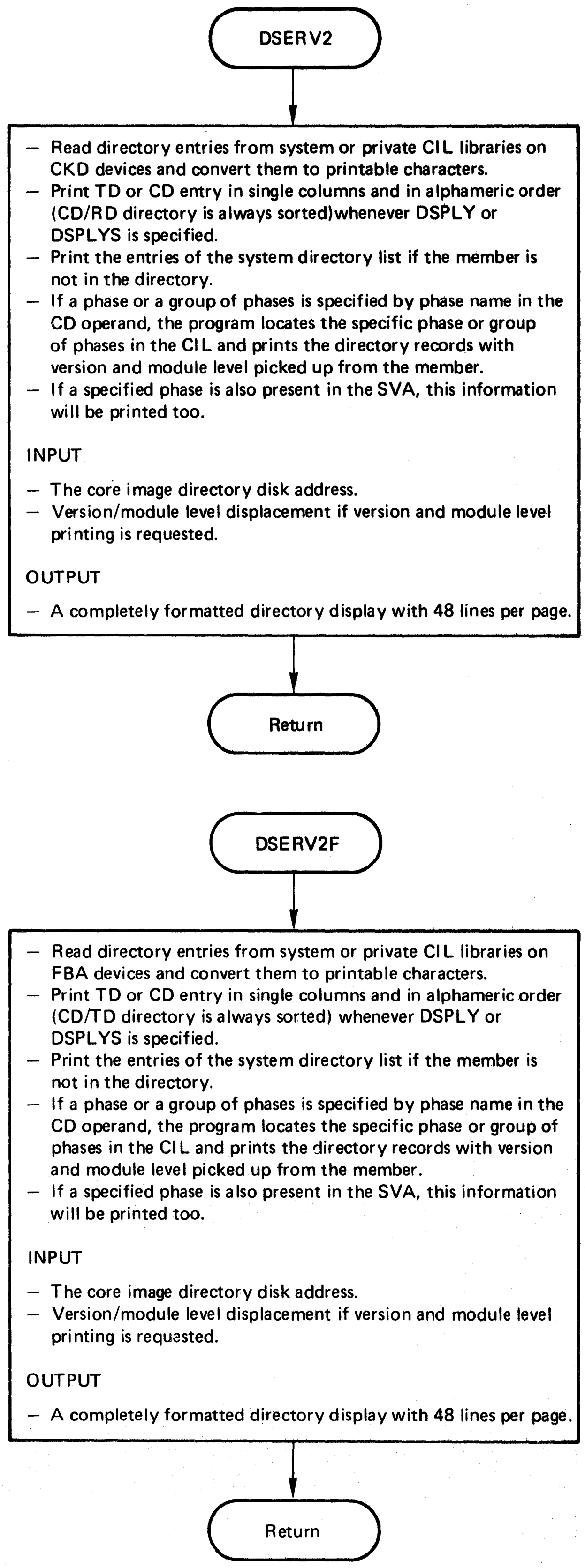
At label :READCARD'

- Get control statement from SYSIPT.
- Scan control statement for valid operation.
  If EOF found go to label 'CLEANUP' in root phase for EOJ.
  If invalid operation found print control statement in error and print error message at label 'INVALID'.
- Scan control statement for valid operands.
  If invalid operand found, print control statement in error and print error message.
- Setup SWA and SWA1 to note down the function(s) specified in the control statement.
- Setup the appropriate indicators of SWB and SWB1 (if assigned private library is empty print error message).
- If CD operand with a specific phase name is found, save the phase name into field 'PNBUCKET' and save the displacement of VER/MOD in field 'VMDISP'.

At label 'LOADF2'

- Initialize FETCH of next overlay phase by testing SWB and SWB1 and branch to FETCH routine in root phase.
  (The order of libraries to be printed is given by the order of testing SWB and SWB1.)

Chart 74. DSERV2/DSERV2F
Detail Charts RB-RC.



- Read directory entries from system or private CIL libraries on CKD devices and convert them to printable characters.
- Print TD or CD entry in single columns and in alphameric order (CD/RD directory is always sorted)whenever DSPLY or DSPLYS is specified.
- Print the entries of the system directory list if the member is not in the directory.
- If a phase or a group of phases is specified by phase name in the CD operand, the program locates the specific phase or group of phases in the CIL and prints the directory records with version and module level picked up from the member.
- If a specified phase is also present in the SVA, this information will be printed too.

INPUT

- The core image directory disk address.
- Version/module level displacement if version and module level printing is requested.

OUTPUT

- A completely formatted directory display with 48 lines per page.

- Read directory entries from system or private CIL libraries on FBA devices and convert them to printable characters.
- Print TD or CD entry in single columns and in alphameric order (CD/TD directory is always sorted) whenever DSPLY or DSPLYS is specified.
- Print the entries of the system directory list if the member is not in the directory.
- If a phase or a group of phases is specified by phase name in the CD operand, the program locates the specific phase or group of phases in the CIL and prints the directory records with version and module level picked up from the member.
- If a specified phase is also present in the SVA, this information will be printed too.

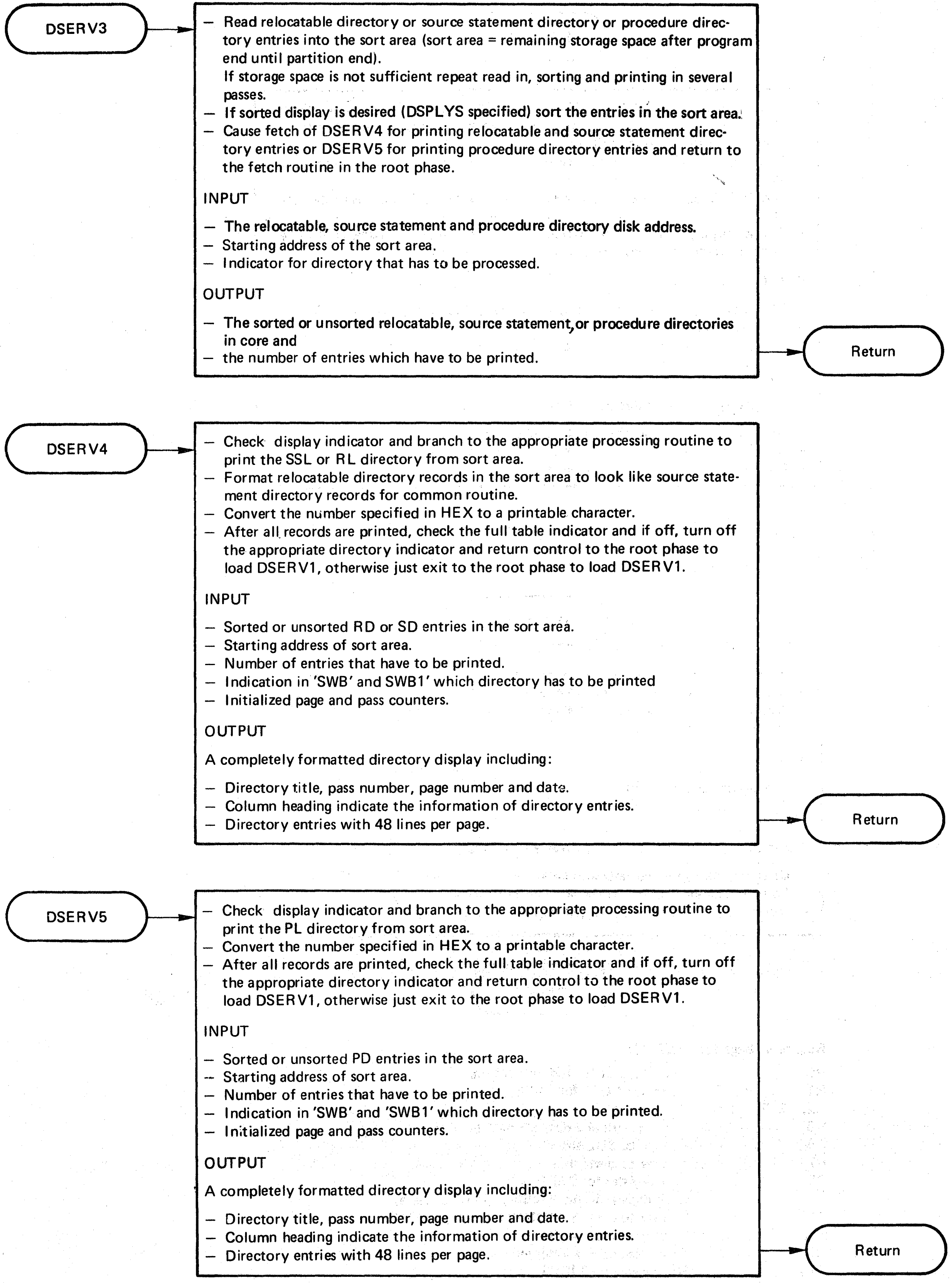
INPUT

- The core image directory disk address.
- Version/module level displacement if version and module level printing is requested.

OUTPUT

- A completely formatted directory display with 48 lines per page.

Chart 75. DSERV3/DSERV4/DSERV5
Detail Charts RD and RF.



DSERV3

- Read relocatable directory or source statement directory or procedure directory entries into the sort area (sort area = remaining storage space after program end until partition end).
  If storage space is not sufficient repeat read in, sorting and printing in several passes.
- If sorted display is desired (DSPLYS specified) sort the entries in the sort area.
- Cause fetch of DSERV4 for printing relocatable and source statement directory entries or DSERV5 for printing procedure directory entries and return to the fetch routine in the root phase.

INPUT

- The relocatable, source statement and procedure directory disk address.
- Starting address of the sort area.
- Indicator for directory that has to be processed.

OUTPUT

- The sorted or unsorted relocatable, source statement,or procedure directories in core and
- the number of entries which have to be printed.

Return

DSERV4

- Check display indicator and branch to the appropriate processing routine to print the SSL or RL directory from sort area.
- Format relocatable directory records in the sort area to look like source statement directory records for common routine.
- Convert the number specified in HEX to a printable character.
- After all records are printed, check the full table indicator and if off, turn off the appropriate directory indicator and return control to the root phase to load DSERV1, otherwise just exit to the root phase to load DSERV1.

INPUT

- Sorted or unsorted RD or SD entries in the sort area.
- Starting address of sort area.
- Number of entries that have to be printed.
- Indication in 'SWB' and SWB1' which directory has to be printed
- Initialized page and pass counters.

OUTPUT

A completely formatted directory display including:

- Directory title, pass number, page number and date.
- Column heading indicate the information of directory entries.
- Directory entries with 48 lines per page.

Return

DSERV5

- Check display indicator and branch to the appropriate processing routine to print the PL directory from sort area.
- Convert the number specified in HEX to a printable character.
- After all records are printed, check the full table indicator and if off, turn off the appropriate directory indicator and return control to the root phase to load DSERV1, otherwise just exit to the root phase to load DSERV1.

INPUT

- Sorted or unsorted PD entries in the sort area.
- Starting address of sort area.
- Number of entries that have to be printed.
- Indication in 'SWB' and 'SWB1' which directory has to be printed.
- Initialized page and pass counters.

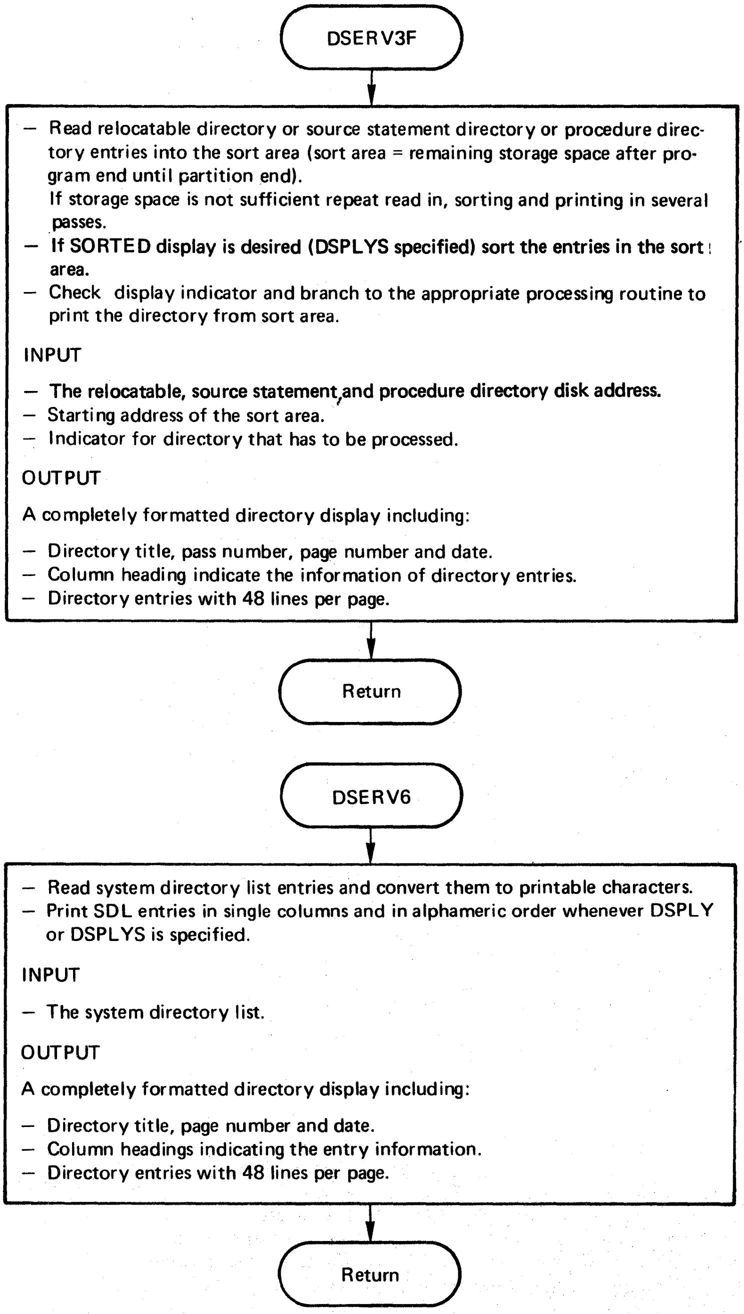
OUTPUT

A completely formatted directory display including:

- Directory title, pass number, page number and date.
- Column heading indicate the information of directory entries.
- Directory entries with 48 lines per page.

Return

Chart 76. DSERV3F/DSERV6
Detail Charts RE and RG.



- Read relocatable directory or source statement directory or procedure directory entries into the sort area (sort area = remaining storage space after program end until partition end).
  If storage space is not sufficient repeat read in, sorting and printing in several passes.
- If SORTED display is desired (DSPLYS specified) sort the entries in the sort area.
- Check display indicator and branch to the appropriate processing routine to print the directory from sort area.

INPUT

- The relocatable, source statement,and procedure directory disk address.
- Starting address of the sort area.
- Indicator for directory that has to be processed.

OUTPUT

A completely formatted directory display including:

- Directory title, pass number, page number and date.
- Column heading indicate the information of directory entries.
- Directory entries with 48 lines per page.

- Read system directory list entries and convert them to printable characters.
- Print SDL entries in single columns and in alphameric order whenever DSPLY or DSPLYS is specified.

INPUT

- The system directory list.

OUTPUT

A completely formatted directory display including:

- Directory title, page number and date.
- Column headings indicating the entry information.
- Directory entries with 48 lines per page.

Registers Usage for DSERV6:

R0	– Work register for EX instruction.
R1	– Address of CCB for EXCP.
R2, R3, R4, R10	– Work registers.
R5	– Entry length and displacement to VM.
R6	– Pointer to SDL entry.
R7	– Pointer to print area.
R8	– Base register for DSERV6.
R9	– Link register to print routine (in root phase).
R11	– Pointer to system directory list entry in SVA.
R12	– Root phase base register.
R13	– Save area address (reserved for ...).
R14	– Put register for LIOCS.
R15	– LIOCS base register.

Chart 77. CSERV

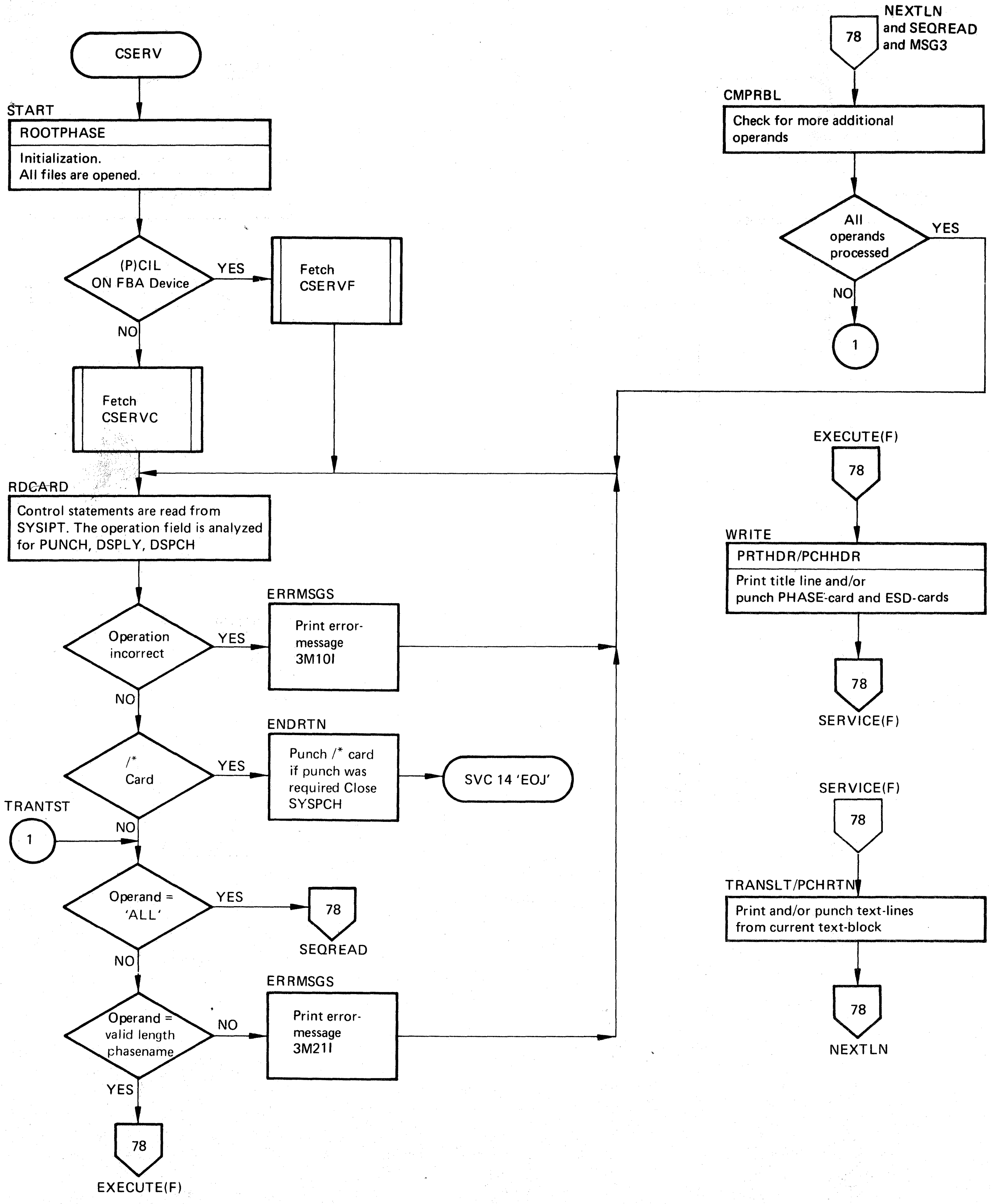

CSERV
START
ROOTPHASE
Initialization.
All files are opened.
(P)CIL
ON FBA Device
YES
NO
Fetch
CSERVF
Fetch
CSERVC
RDCARD
Control statements are read from SYSIPT. The operation field is analyzed for PUNCH, DSPLY, DSPCH
ERRMSGS
Operation incorrect
YES
NO
Print error-message 3M10I
ENDRTN
/* Card
YES
Punch /* card if punch was required Close SYSPCH
SVC 14 'EOJ'
TRANTST
1
NO
Operand = 'ALL'
YES
78
SEQREAD
NO
ERRMSGS
Operand = valid length phasename
NO
Print error-message 3M21I
YES
78
EXECUTE(F)
78
NEXTLN and SEQREAD and MSG3
CMPRBL
Check for more additional operands
All operands processed
YES
NO
1
EXECUTE(F)
78
WRITE
PRTHDR/PCHHDR
Print title line and/or punch PHASE-card and ESD-cards
78
SERVICE(F)
SERVICE(F)
78
TRANSLT/PCHRTN
Print and/or punch text-lines from current text-block
78
NEXTLN

Chart 78. CSERVC/CSERVF

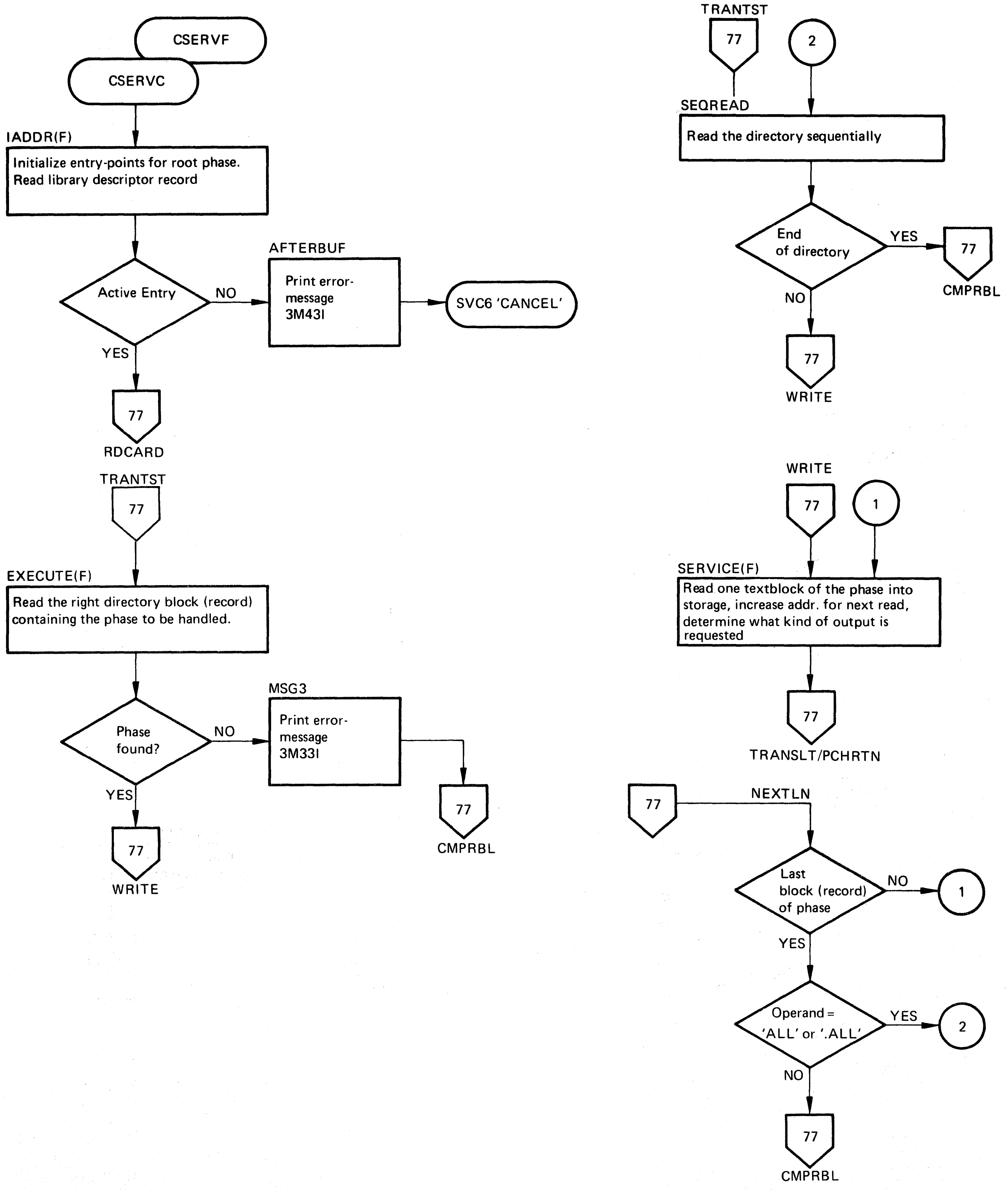

CSERVF
CSERVC
IADDR(F)
Initialize entry-points for root phase. Read library descriptor record
Active Entry
NO
AFTERBUF
Print error-message 3M431
SVC6 'CANCEL'
YES
77
RDCARD
TRANTST
77
EXECUTE(F)
Read the right directory block (record) containing the phase to be handled.
Phase found?
NO
MSG3
Print error-message 3M331
YES
77
WRITE
77
CMPRBL
TRANTST
77
2
SEQREAD
Read the directory sequentially
End of directory
YES
77
CMPRBL
NO
77
WRITE
WRITE
77
1
SERVICE(F)
Read one textblock of the phase into storage, increase addr. for next read, determine what kind of output is requested
77
TRANSLT/PCHRTN
77
NEXTLN
Last block (record) of phase
NO
1
YES
Operand = 'ALL' or '.ALL'
YES
2
NO
77
CMPRBL

Chart 79. RSERV (Part 1 of 2)

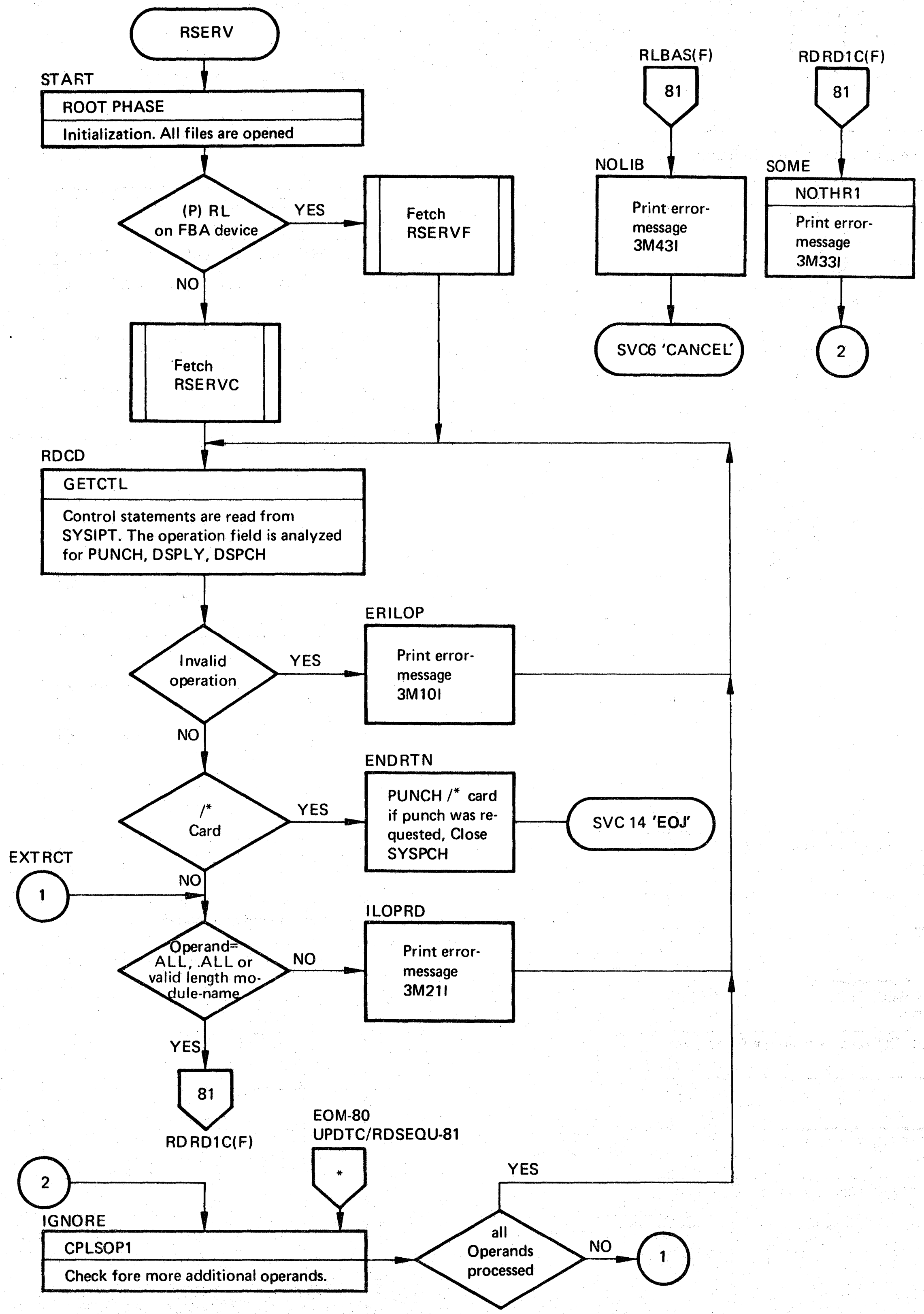

RSERV
START
ROOT PHASE
Initialization. All files are opened
(P) RL on FBA device
YES
NO
Fetch RSERVF
Fetch RSERVC
RLBAS(F)
81
NOLIB
Print error-message 3M431
SVC6 'CANCEL'
RDRD1C(F)
81
SOME
NOTHR1
Print error-message 3M331
2
RDCD
GETCTL
Control statements are read from SYSIPT. The operation field is analyzed for PUNCH, DSPLY, DSPCH
Invalid operation
YES
ERILOP
Print error-message 3M101
NO
/* Card
YES
ENDRTN
PUNCH /* card if punch was re-quested, Close SYSPCH
SVC 14 'EOJ'
EXTRCT
1
NO
Operand= ALL, .ALL or valid length mo-dule-name
NO
ILOPRD
Print error-message 3M211
YES
81
RDRD1C(F)
EOM-80
UPDTC/RDSEQU-81
*
2
IGNORE
CPLSOP1
Check fore more additional operands.
all Operands processed
YES
NO
1

Chart 80. RSERV (Part 2 of 2)

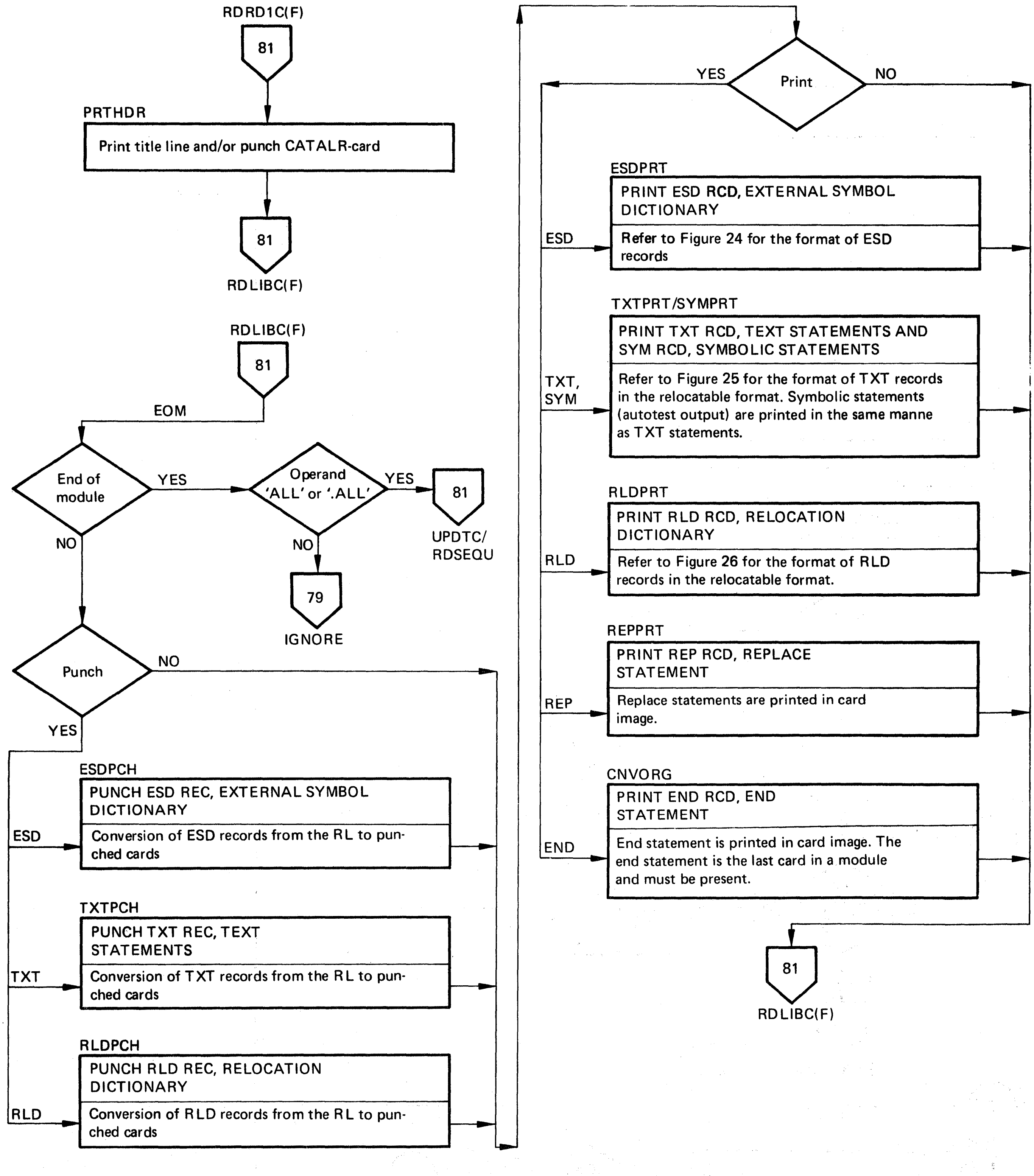

RDRD1C(F)
81
PRTHDR
Print title line and/or punch CATALR-card
81
RDLIBC(F)
RDLIBC(F)
81
EOM
End of module
YES
Operand 'ALL' or '.ALL'
YES
81
UPDTC/ RDSEQU
NO
NO
79
IGNORE
Punch
NO
YES
ESDPCH
PUNCH ESD REC, EXTERNAL SYMBOL DICTIONARY
ESD
Conversion of ESD records from the RL to punched cards
TXTPCH
PUNCH TXT REC, TEXT STATEMENTS
TXT
Conversion of TXT records from the RL to punched cards
RLDPCH
PUNCH RLD REC, RELOCATION DICTIONARY
RLD
Conversion of RLD records from the RL to punched cards
YES
Print
NO
ESDPRT
PRINT ESD RCD, EXTERNAL SYMBOL DICTIONARY
ESD
Refer to Figure 24 for the format of ESD records
TXTPRT/SYMPRT
PRINT TXT RCD, TEXT STATEMENTS AND SYM RCD, SYMBOLIC STATEMENTS
TXT, SYM
Refer to Figure 25 for the format of TXT records in the relocatable format. Symbolic statements (autotest output) are printed in the same manne as TXT statements.
RLDPRT
PRINT RLD RCD, RELOCATION DICTIONARY
RLD
Refer to Figure 26 for the format of RLD records in the relocatable format.
REPPRT
PRINT REP RCD, REPLACE STATEMENT
REP
Replace statements are printed in card image.
CNVORG
PRINT END RCD, END STATEMENT
END
End statement is printed in card image. The end statement is the last card in a module and must be present.
81
RDLIBC(F)

Chart 81. RSERVC/RSERVF

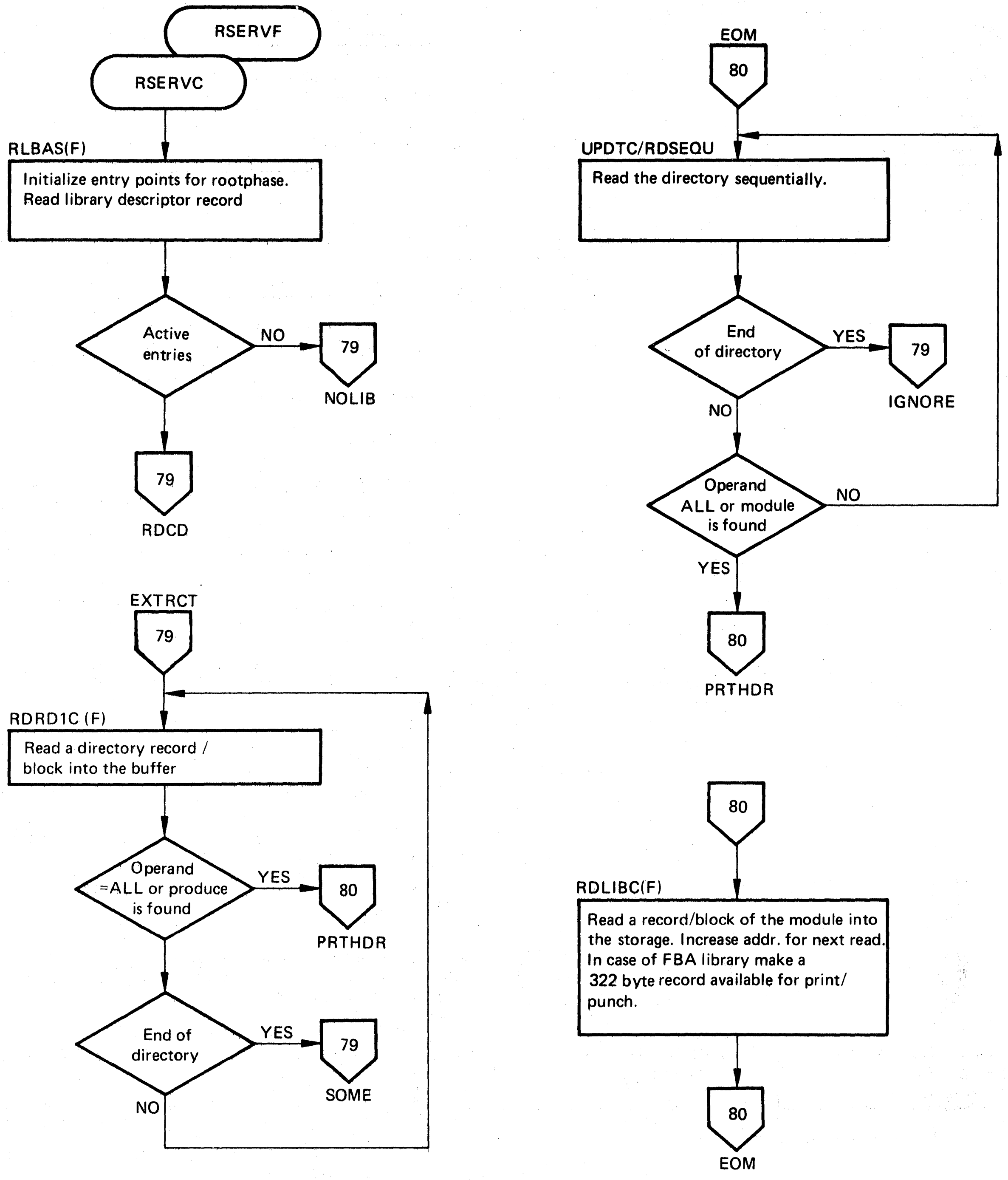

RSERVF
RSERVC
RLBAS(F)
Initialize entry points for rootphase. Read library descriptor record
Active entries
NO
79
NOLIB
79
RDCD
EXTRCT
79
RDRD1C (F)
Read a directory record / block into the buffer
Operand =ALL or produce is found
YES
80
PRTHDR
End of directory
YES
79
SOME
NO
EOM
80
UPDTC/RDSEQU
Read the directory sequentially.
End of directory
YES
79
IGNORE
NO
Operand ALL or module is found
NO
YES
80
PRTHDR
80
RDLIBC(F)
Read a record/block of the module into the storage. Increase addr. for next read. In case of FBA library make a 322 byte record available for print/ punch.
80
EOM

Chart 82. SSERV (Part 1 of 2)

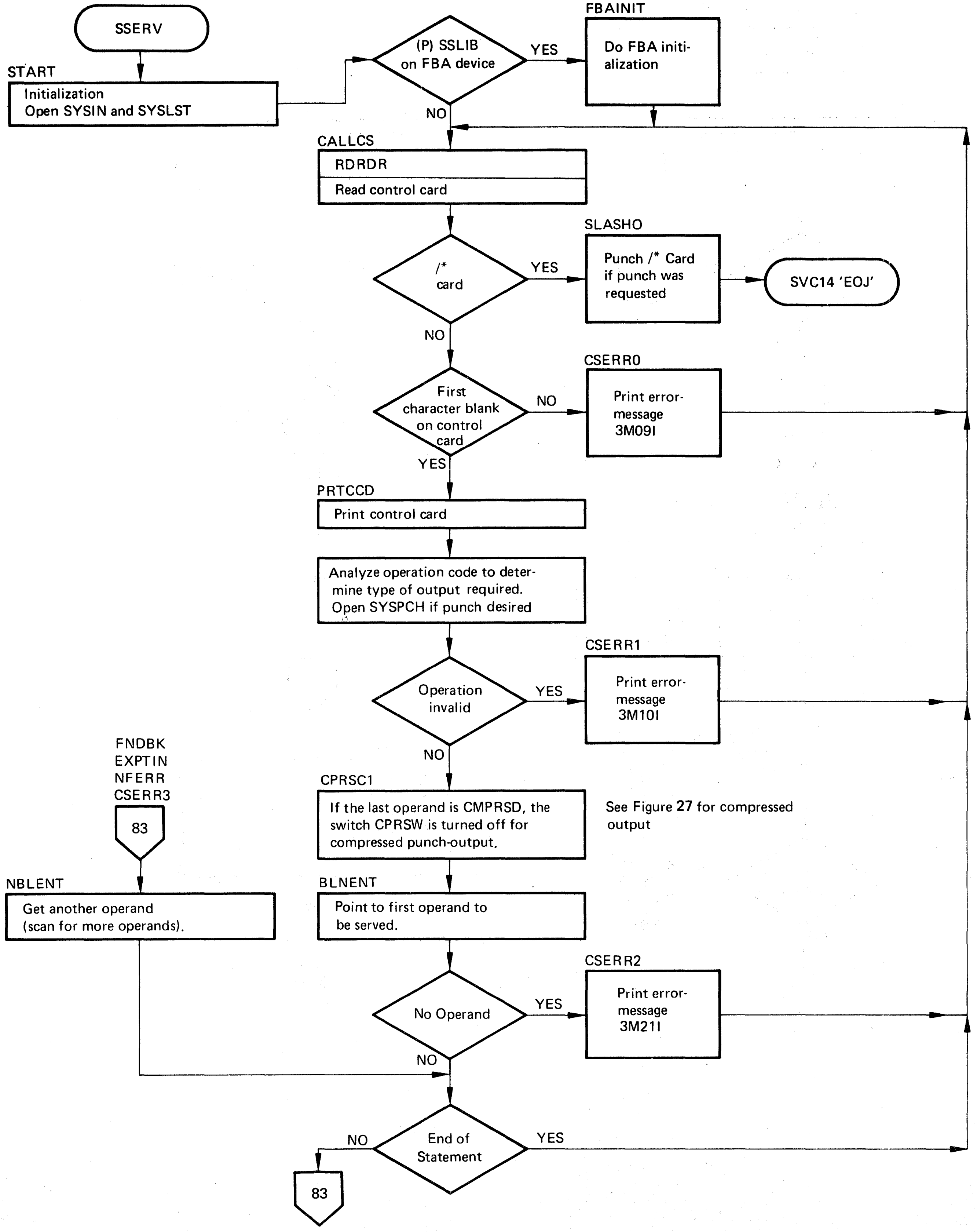

SSERV
START
Initialization
Open SYSIN and SYSLST
(P) SSLIB on FBA device
YES
NO
FBAINIT
Do FBA initialization
CALLCS
RDRDR
Read control card
/* card
YES
NO
SLASHO
Punch /* Card if punch was requested
SVC14 'EOJ'
First character blank on control card
NO
YES
CSERR0
Print error-message 3M091
PRTCCD
Print control card
Analyze operation code to determine type of output required.
Open SYSPCH if punch desired
Operation invalid
YES
NO
CSERR1
Print error-message 3M101
FNDBK
EXPTIN
NFERR
CSERR3
83
CPRSC1
If the last operand is CMPRSD, the switch CPRSW is turned off for compressed punch-output.
See Figure 27 for compressed output
NBLENT
Get another operand (scan for more operands).
BLNENT
Point to first operand to be served.
No Operand
YES
NO
CSERR2
Print error-message 3M211
End of Statement
NO
YES
83

Chart 83. SSERV (Part 2 of 2)

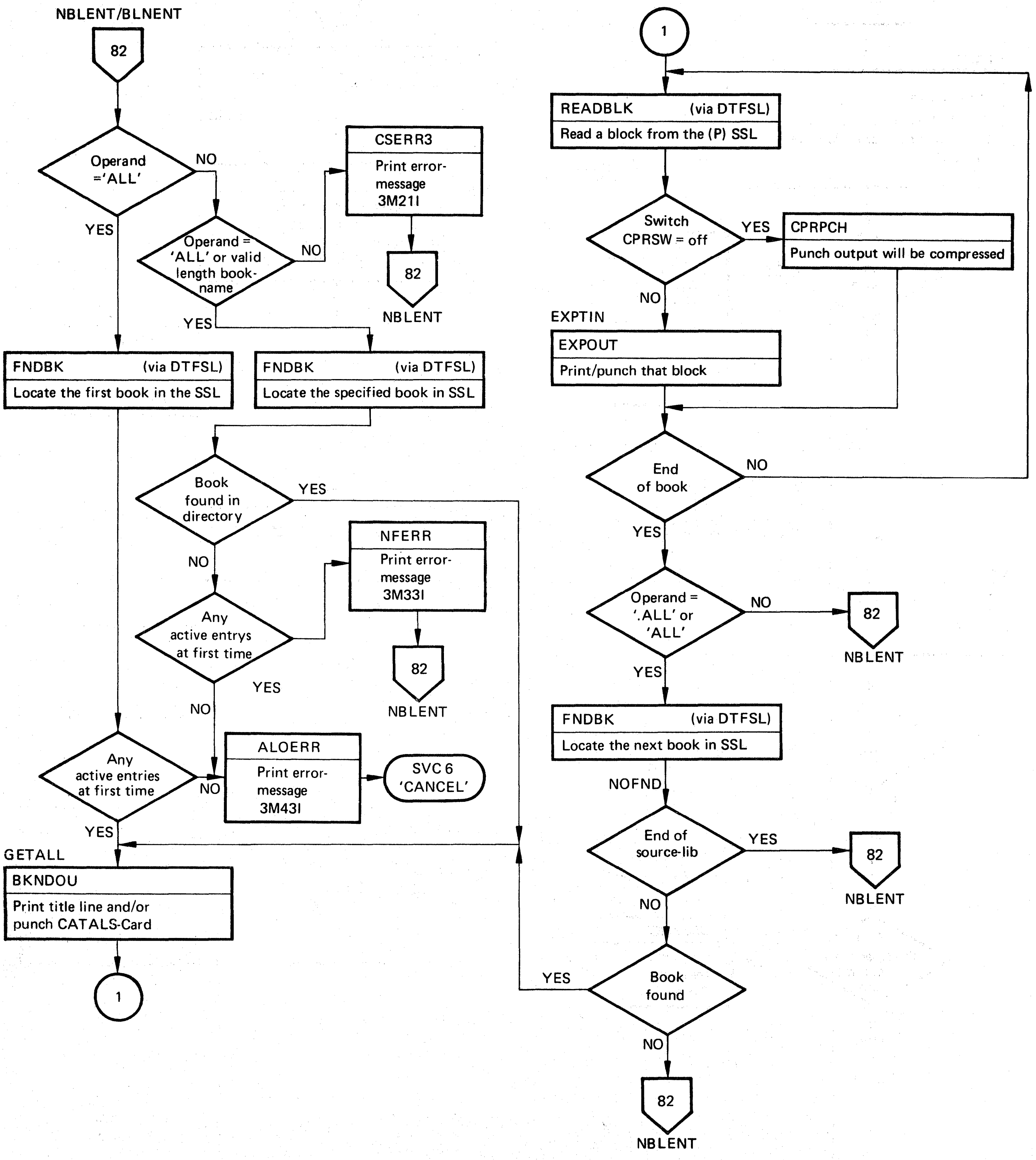

NBLENT/BLNENT
82
Operand ='ALL'
NO
YES
Operand = 'ALL' or valid length book-name
NO
YES
CSERR3
Print error-message 3M21I
82
NBLENT
FNDBK (via DTFSL)
Locate the first book in the SSL
FNDBK (via DTFSL)
Locate the specified book in SSL
Book found in directory
YES
NO
Any active entrys at first time
YES
NO
NFERR
Print error-message 3M33I
82
NBLENT
Any active entries at first time
NO
YES
ALOERR
Print error-message 3M43I
SVC 6 'CANCEL'
GETALL
BKNDOU
Print title line and/or punch CATALS-Card
1
1
READBLK (via DTFSL)
Read a block from the (P) SSL
Switch CPRSW = off
YES
CPRPCH
Punch output will be compressed
NO
EXPTIN
EXPOUT
Print/punch that block
End of book
NO
YES
Operand = '.ALL' or 'ALL'
NO
82
NBLENT
YES
FNDBK (via DTFSL)
Locate the next book in SSL
NOFND
End of source-lib
YES
82
NBLENT
NO
Book found
YES
NO
82
NBLENT

Chart 84. PSERV (Part 1 of 2)

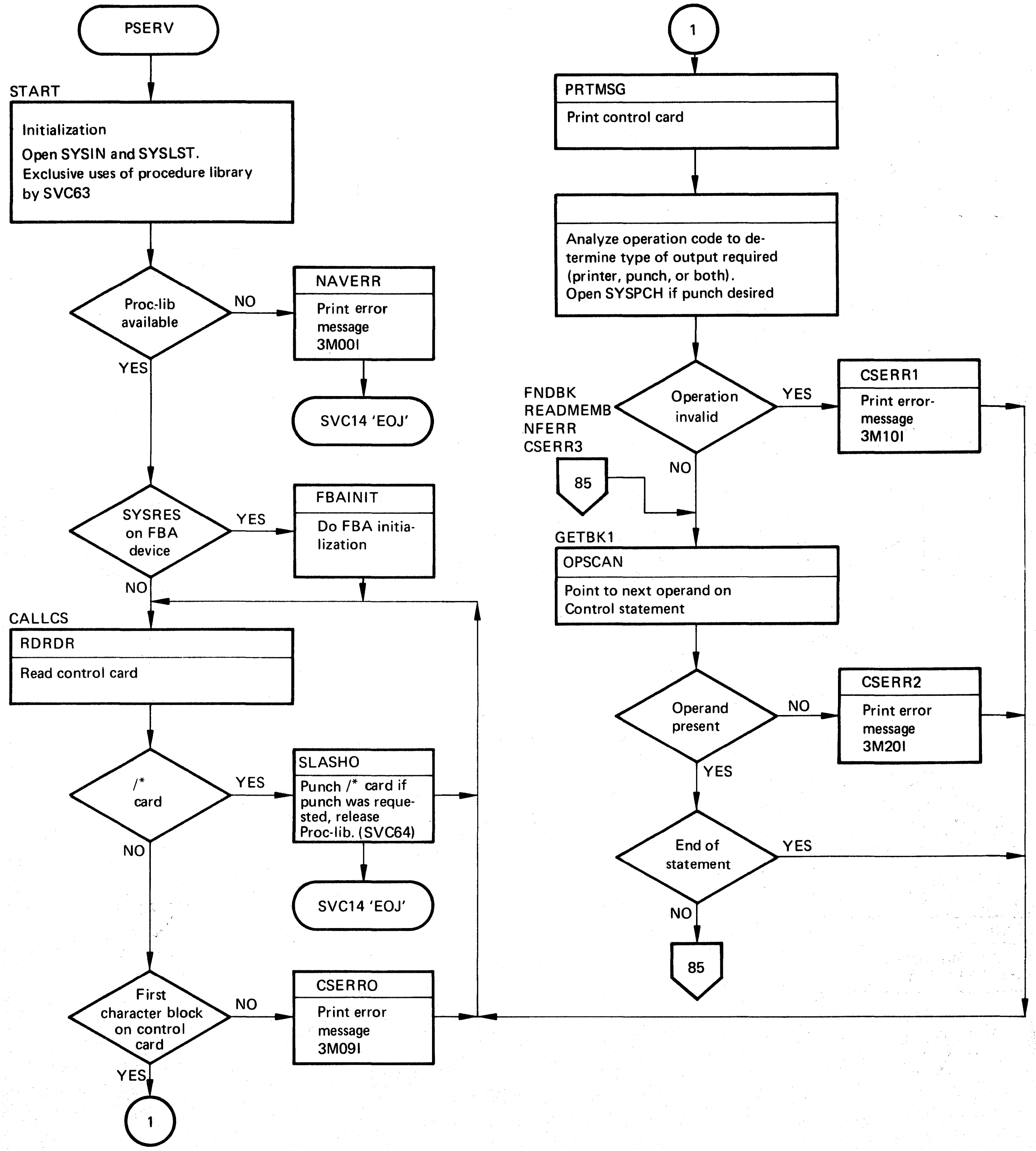

PSERV
START
Initialization
Open SYSIN and SYSLST.
Exclusive uses of procedure library by SVC63
Proc-lib available
NO
NAVERR
Print error message 3M001
SVC14 'EOJ'
YES
SYSRES on FBA device
YES
FBAINIT
Do FBA initialization
NO
CALLCS
RDRDR
Read control card
/* card
YES
SLASHO
Punch /* card if punch was requested, release Proc-lib. (SVC64)
SVC14 'EOJ'
NO
First character block on control card
NO
CSERRO
Print error message 3M091
YES
1
1
PRTMSG
Print control card
Analyze operation code to determine type of output required (printer, punch, or both).
Open SYSPCH if punch desired
FNDBK
READMEMB
NFERR
CSERR3
Operation invalid
YES
CSERR1
Print error-message 3M101
85
NO
GETBK1
OPSCAN
Point to next operand on Control statement
Operand present
NO
CSERR2
Print error message 3M201
YES
End of statement
YES
NO
85

Chart 85. PSERV (Part 2 of 2)

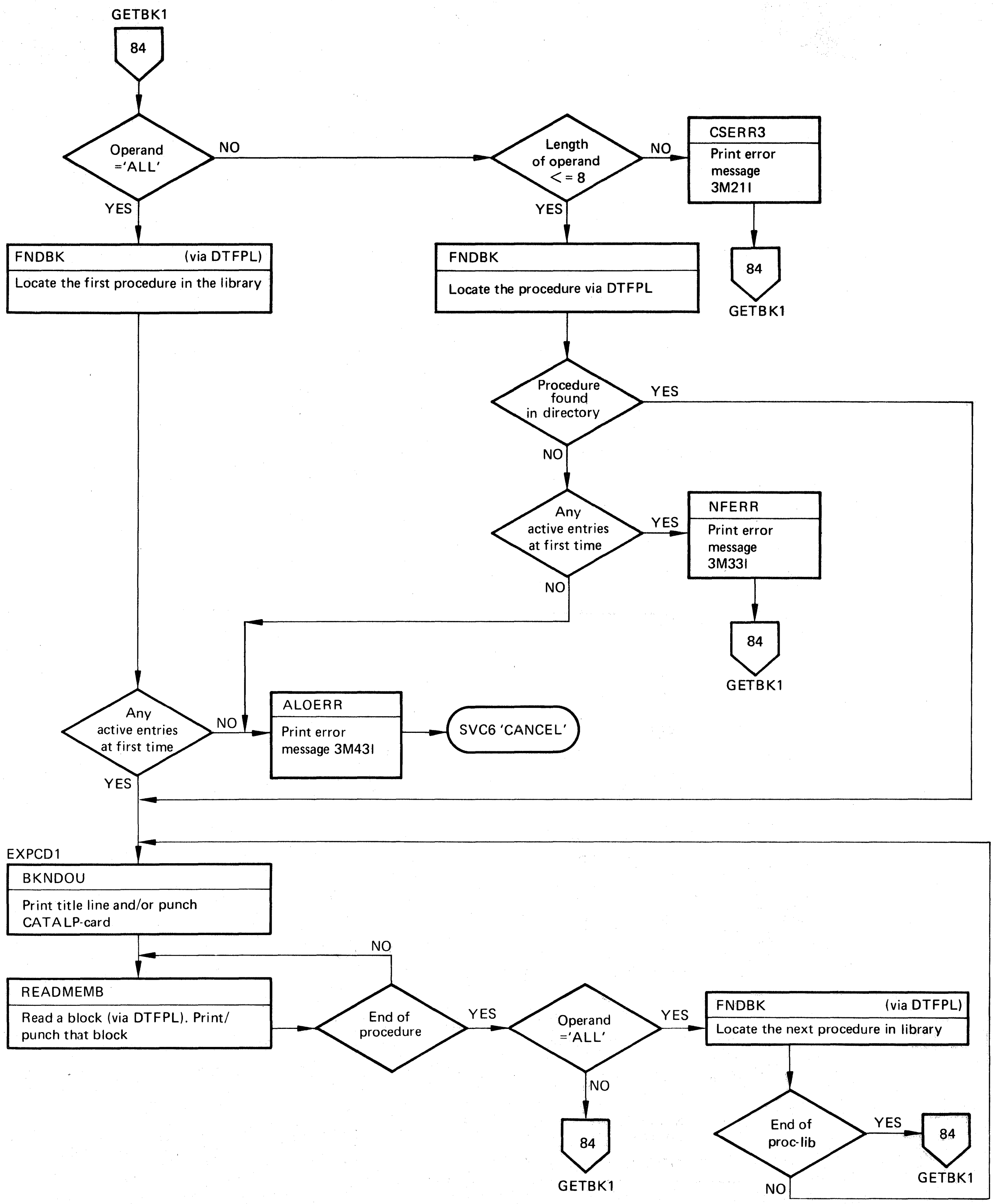

GETBK1
84
Operand ='ALL'
NO
YES
Length of operand <= 8
NO
YES
CSERR3
Print error message 3M21I
84
GETBK1
FNDBK
(via DTFPL)
Locate the first procedure in the library
FNDBK
Locate the procedure via DTFPL
Procedure found in directory
YES
NO
Any active entries at first time
YES
NO
NFERR
Print error message 3M33I
84
GETBK1
Any active entries at first time
NO
YES
ALOERR
Print error message 3M43I
SVC6 'CANCEL'
EXPCD1
BKNDOU
Print title line and/or punch CATALP-card
NO
READMEMB
Read a block (via DTFPL). Print/ punch that block
End of procedure
YES
Operand ='ALL'
YES
NO
84
GETBK1
FNDBK
(via DTFPL)
Locate the next procedure in library
End of proc-lib
YES
NO
84
GETBK1

Chart 86. $$BSYSWR

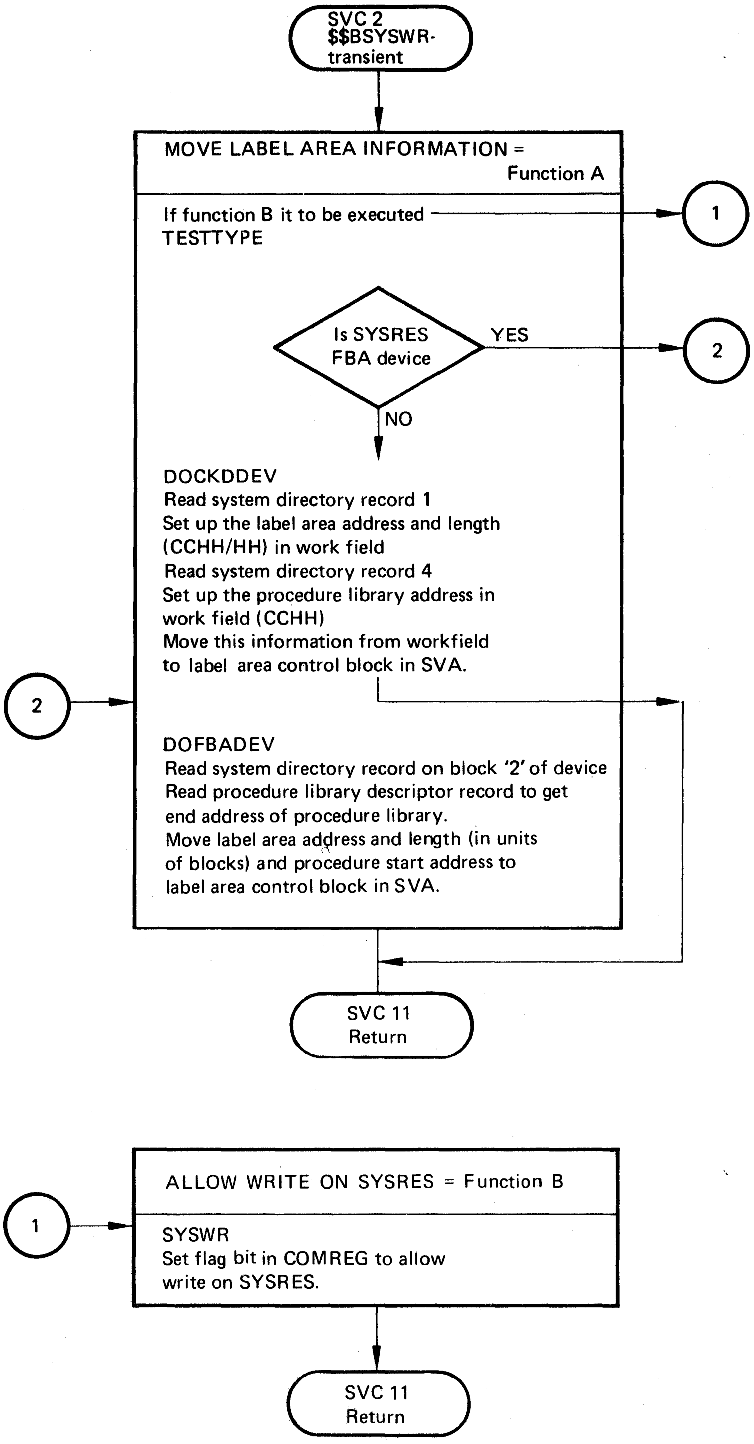
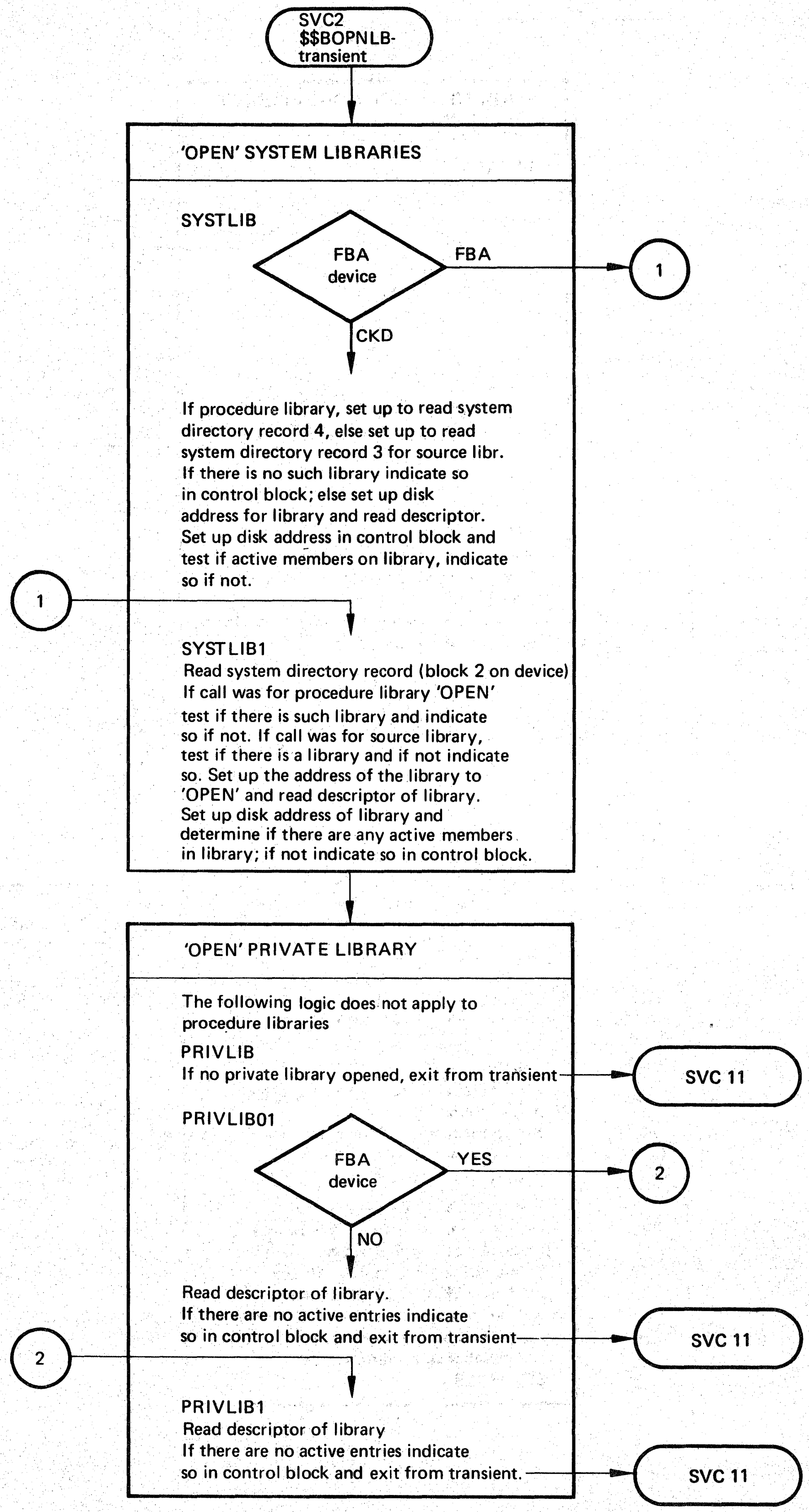

SVC 2
$$BSYSWR-
transient
MOVE LABEL AREA INFORMATION =
Function A
If function B it to be executed
TESTTYPE
1
Is SYSRES
FBA device
YES
2
NO
DOCKDDEV
Read system directory record 1
Set up the label area address and length
(CCHH/HH) in work field
Read system directory record 4
Set up the procedure library address in
work field (CCHH)
Move this information from workfield
to label area control block in SVA.
2
DOFBADEV
Read system directory record on block '2' of device
Read procedure library descriptor record to get
end address of procedure library.
Move label area address and length (in units
of blocks) and procedure start address to
label area control block in SVA.
SVC 11
Return
1
ALLOW WRITE ON SYSRES = Function B
SYSWR
Set flag bit in COMREG to allow
write on SYSRES.
SVC 11
Return

Chart 87. $$BOPNLB


SVC2
$$BOPNLB-
transient
'OPEN' SYSTEM LIBRARIES
SYSTLIB
FBA device
FBA
1
CKD
If procedure library, set up to read system directory record 4, else set up to read system directory record 3 for source libr. If there is no such library indicate so in control block; else set up disk address for library and read descriptor. Set up disk address in control block and test if active members on library, indicate so if not.
1
SYSTLIB1
Read system directory record (block 2 on device)
If call was for procedure library 'OPEN' test if there is such library and indicate so if not. If call was for source library, test if there is a library and if not indicate so. Set up the address of the library to 'OPEN' and read descriptor of library. Set up disk address of library and determine if there are any active members in library; if not indicate so in control block.
'OPEN' PRIVATE LIBRARY
The following logic does not apply to procedure libraries
PRIVLIB
If no private library opened, exit from transient
SVC 11
PRIVLIB01
FBA device
YES
2
NO
Read descriptor of library. If there are no active entries indicate so in control block and exit from transient
SVC 11
2
PRIVLIB1
Read descriptor of library
If there are no active entries indicate so in control block and exit from transient.
SVC 11

hart 88. DTFSL Macro (Part 1 of 3)

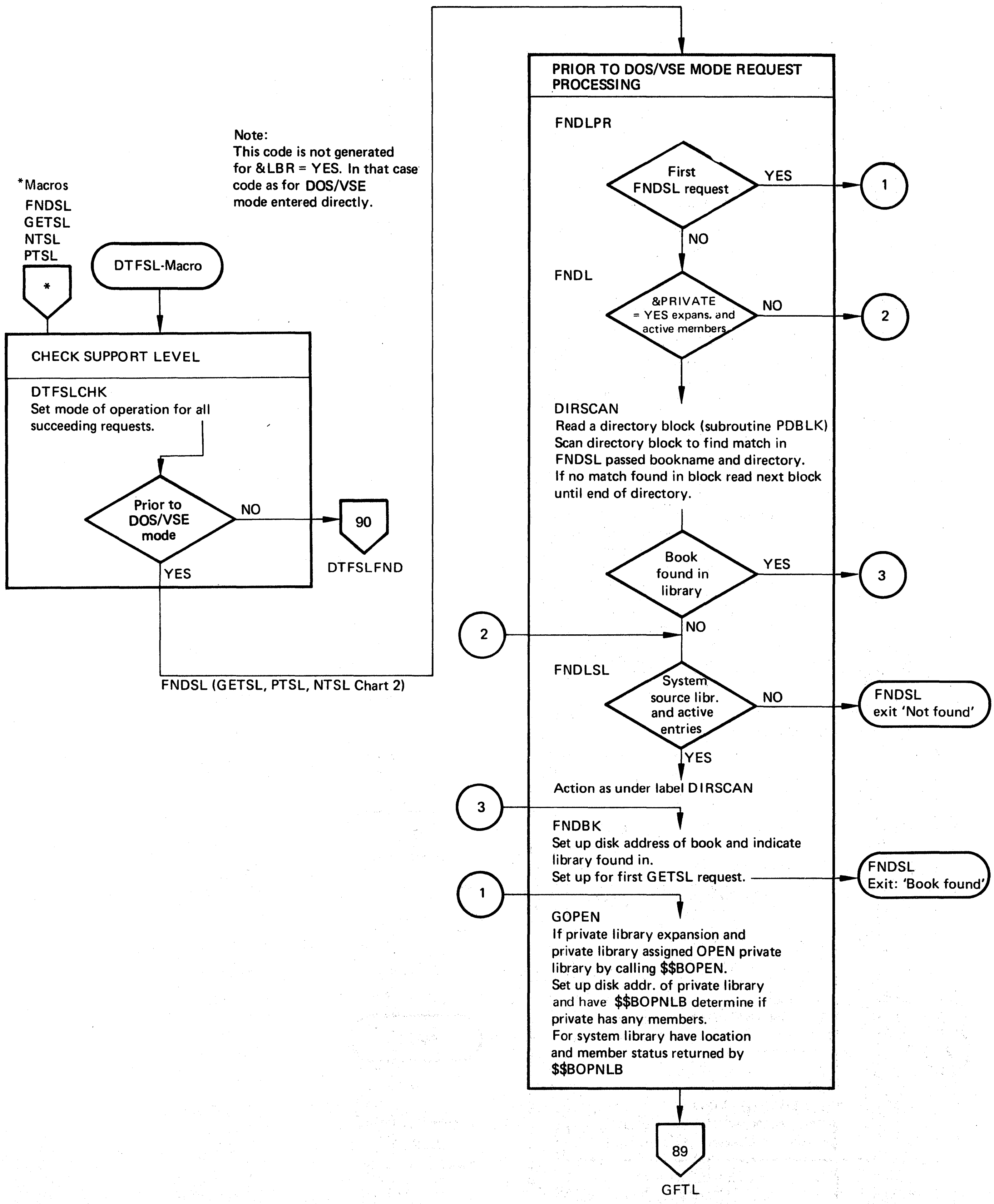

*Macros
FNDSL
GETSL
NTSL
PTSL
*
DTFSL-Macro
Note:
This code is not generated for &LBR = YES. In that case code as for DOS/VSE mode entered directly.
CHECK SUPPORT LEVEL
DTFSLCHK
Set mode of operation for all succeeding requests.
Prior to DOS/VSE mode
NO
90
DTFSLFND
YES
FNDSL (GETSL, PTSL, NTSL Chart 2)
PRIOR TO DOS/VSE MODE REQUEST PROCESSING
FNDLPR
First FNDSL request
YES
1
NO
FNDL
&PRIVATE = YES expans. and active members
NO
2
DIRSCAN
Read a directory block (subroutine PDBLK)
Scan directory block to find match in FNDSL passed bookname and directory.
If no match found in block read next block until end of directory.
Book found in library
YES
3
NO
2
FNDLSL
System source libr. and active entries
NO
FNDSL exit 'Not found'
YES
Action as under label DIRSCAN
3
FNDBK
Set up disk address of book and indicate library found in.
Set up for first GETSL request.
FNDSL Exit: 'Book found'
1
GOPEN
If private library expansion and private library assigned OPEN private library by calling $$BOPEN.
Set up disk addr. of private library and have $$BOPNLB determine if private has any members.
For system library have location and member status returned by $$BOPNLB
89
GFTL

Chart 89. DTFSL Macro (Part 2 of 3)

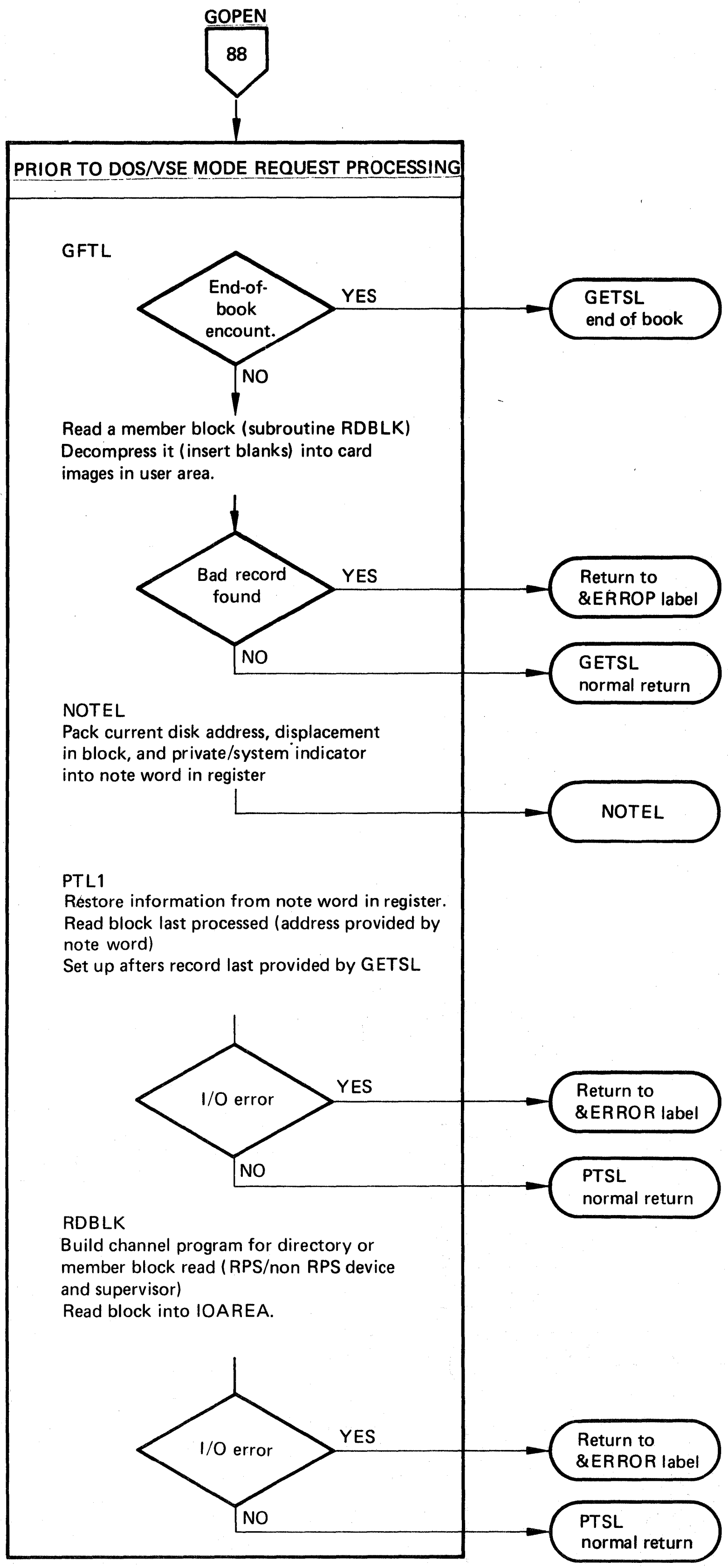

GOPEN
88
PRIOR TO DOS/VSE MODE REQUEST PROCESSING
GFTL
End-of-book encount.
YES
GETSL end of book
NO
Read a member block (subroutine RDBLK) Decompress it (insert blanks) into card images in user area.
Bad record found
YES
Return to &ERROP label
NO
GETSL normal return
NOTEL
Pack current disk address, displacement in block, and private/system indicator into note word in register
NOTEL
PTL1
Restore information from note word in register.
Read block last processed (address provided by note word)
Set up afters record last provided by GETSL
I/O error
YES
Return to &ERROR label
NO
PTSL normal return
RDBLK
Build channel program for directory or member block read (RPS/non RPS device and supervisor)
Read block into IOAREA.
I/O error
YES
Return to &ERROR label
NO
PTSL normal return

Chart 90. DTFSL Macro (Part 3 of 3)

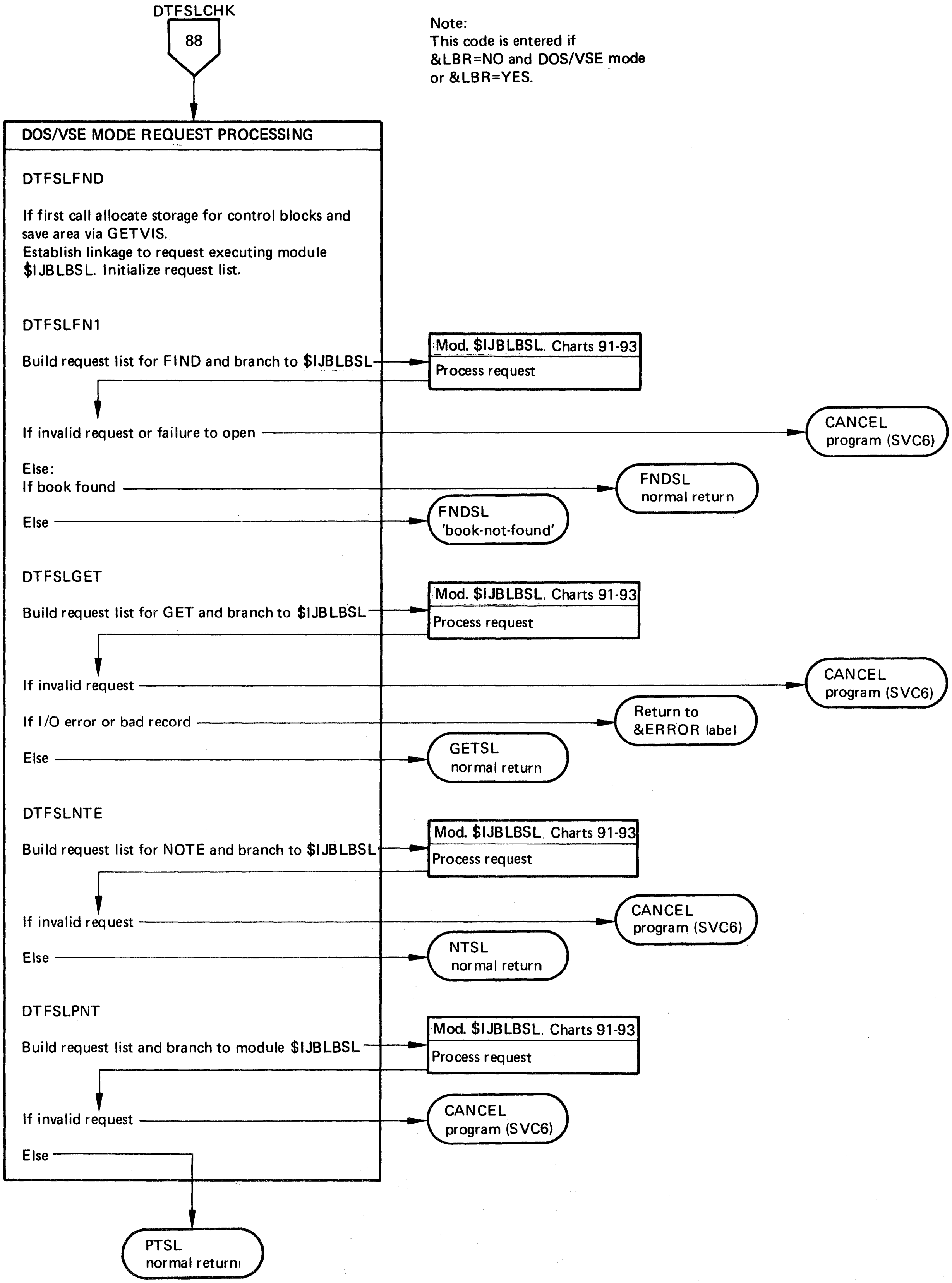
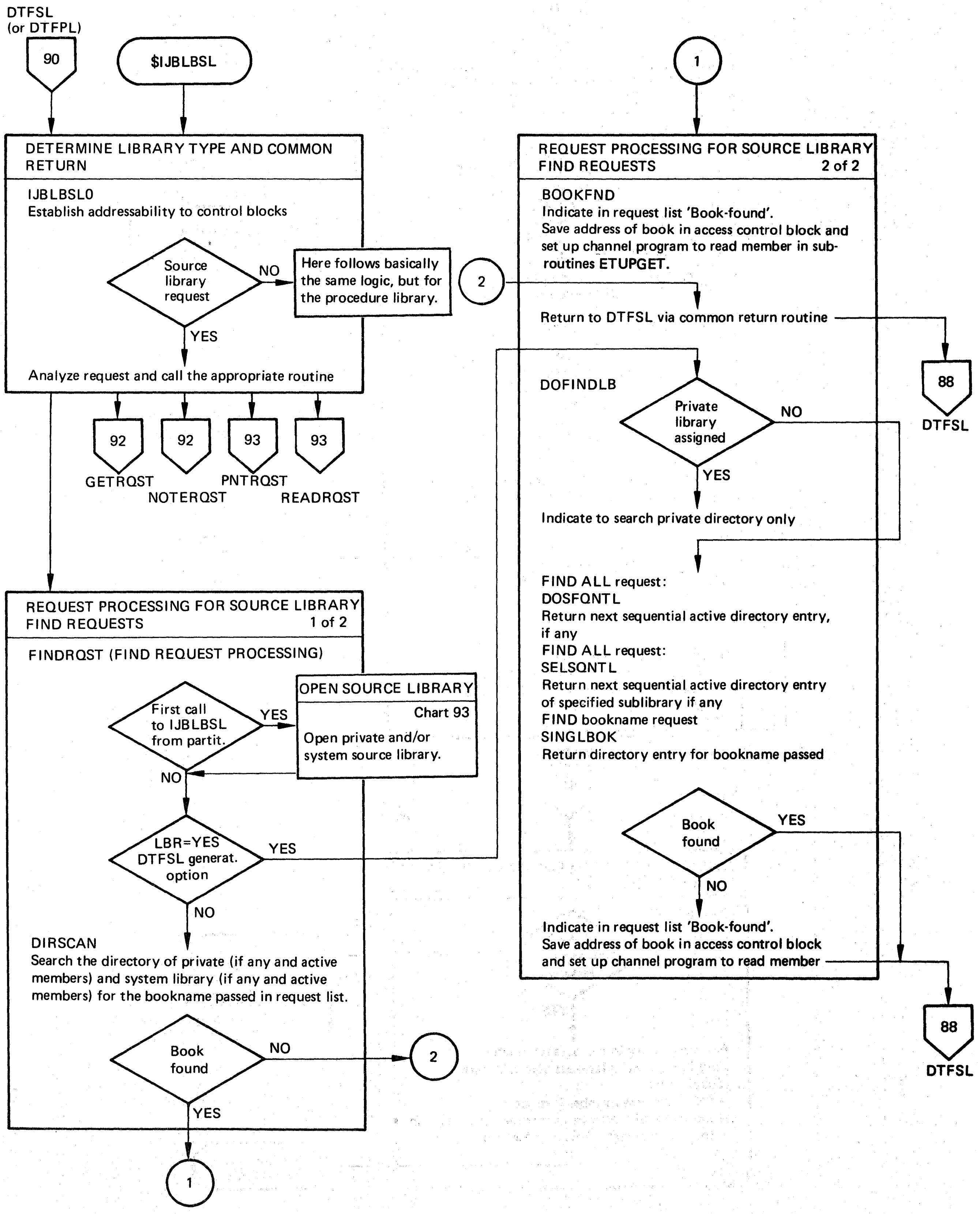

DTFSLCHK
88
Note:
This code is entered if
&LBR=NO and DOS/VSE mode
or &LBR=YES.
DOS/VSE MODE REQUEST PROCESSING
DTFSLFND
If first call allocate storage for control blocks and save area via GETVIS.
Establish linkage to request executing module $IJBLBSL. Initialize request list.
DTFSLFN1
Build request list for FIND and branch to $IJBLBSL
Mod. $IJBLBSL. Charts 91-93
Process request
If invalid request or failure to open
CANCEL
program (SVC6)
Else:
If book found
FNDSL
normal return
Else
FNDSL
'book-not-found'
DTFSLGET
Build request list for GET and branch to $IJBLBSL
Mod. $IJBLBSL. Charts 91-93
Process request
If invalid request
CANCEL
program (SVC6)
If I/O error or bad record
Return to
&ERROR label
Else
GETSL
normal return
DTFSLNTE
Build request list for NOTE and branch to $IJBLBSL
Mod. $IJBLBSL. Charts 91-93
Process request
If invalid request
CANCEL
program (SVC6)
Else
NTSL
normal return
DTFSLPNT
Build request list and branch to module $IJBLBSL
Mod. $IJBLBSL. Charts 91-93
Process request
If invalid request
CANCEL
program (SVC6)
Else
PTSL
normal return

Chart 91. $IJBLBSL (Part 1 of 3)



DTFSL
(or DTFPL)
90
$IJBLBSL
DETERMINE LIBRARY TYPE AND COMMON RETURN
IJBLBSLO
Establish addressability to control blocks
Source library request
NO
Here follows basically the same logic, but for the procedure library.
2
YES
Analyze request and call the appropriate routine
92
92
93
93
GETRQST
NOTERQST
PNTRQST
READRQST
REQUEST PROCESSING FOR SOURCE LIBRARY
FIND REQUESTS 1 of 2
FINDRQST (FIND REQUEST PROCESSING)
First call to IJBLBSL from partit.
YES
OPEN SOURCE LIBRARY
Chart 93
Open private and/or system source library.
NO
LBR=YES DTFSL generat. option
YES
NO
DIRSCAN
Search the directory of private (if any and active members) and system library (if any and active members) for the bookname passed in request list.
Book found
NO
2
YES
1
1
REQUEST PROCESSING FOR SOURCE LIBRARY
FIND REQUESTS 2 of 2
BOOKFND
Indicate in request list 'Book-found'.
Save address of book in access control block and set up channel program to read member in subroutines ETUPGET.
Return to DTFSL via common return routine
88
DTFSL
DOFINDLB
Private library assigned
NO
YES
Indicate to search private directory only
FIND ALL request:
DOSFQNTL
Return next sequential active directory entry, if any
FIND ALL request:
SELSQNTL
Return next sequential active directory entry of specified sublibrary if any
FIND bookname request
SINGLBOK
Return directory entry for bookname passed
Book found
YES
NO
Indicate in request list 'Book-found'.
Save address of book in access control block and set up channel program to read member
88
DTFSL

Chart 92. $IJBLBSL (Part 2 of 3)

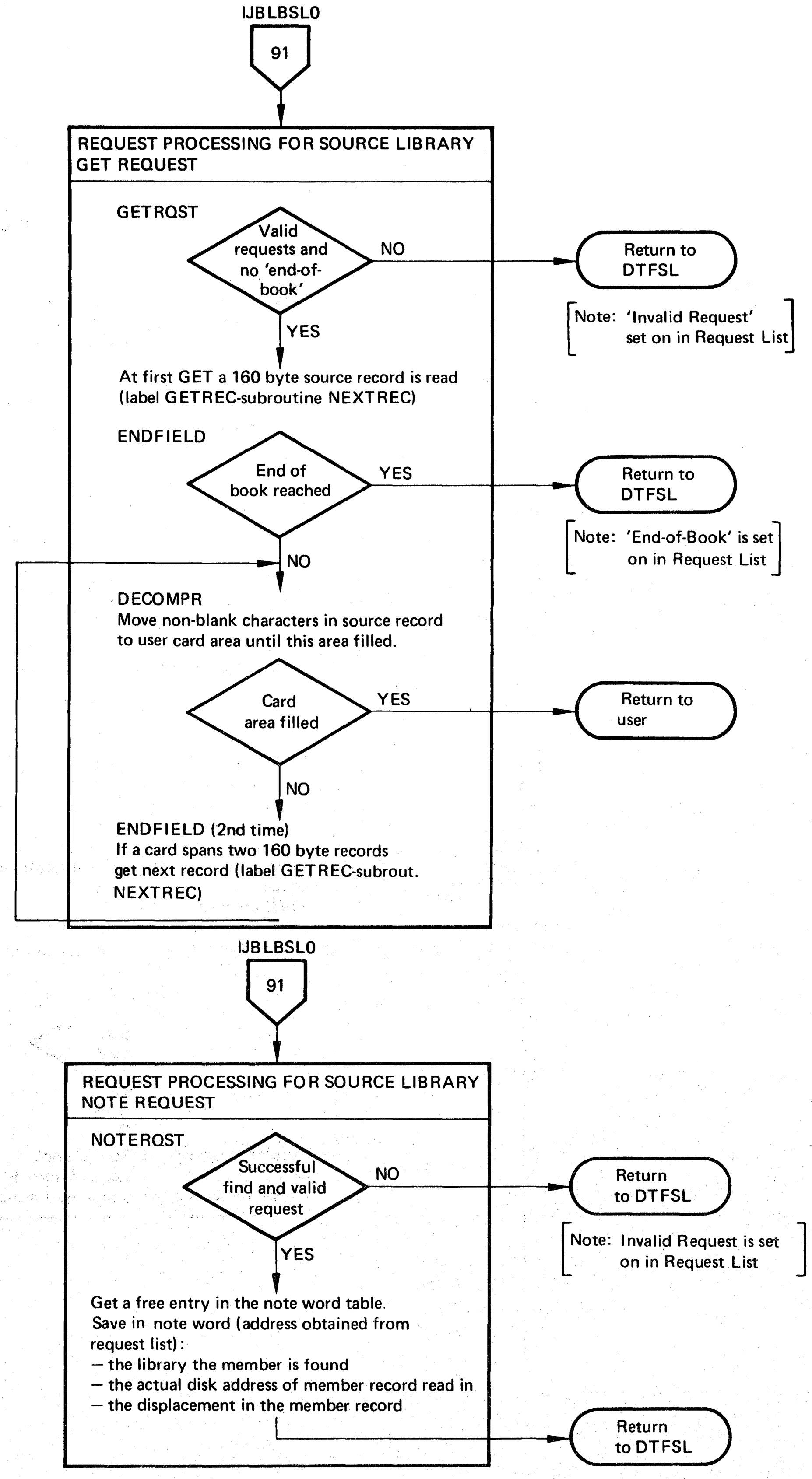
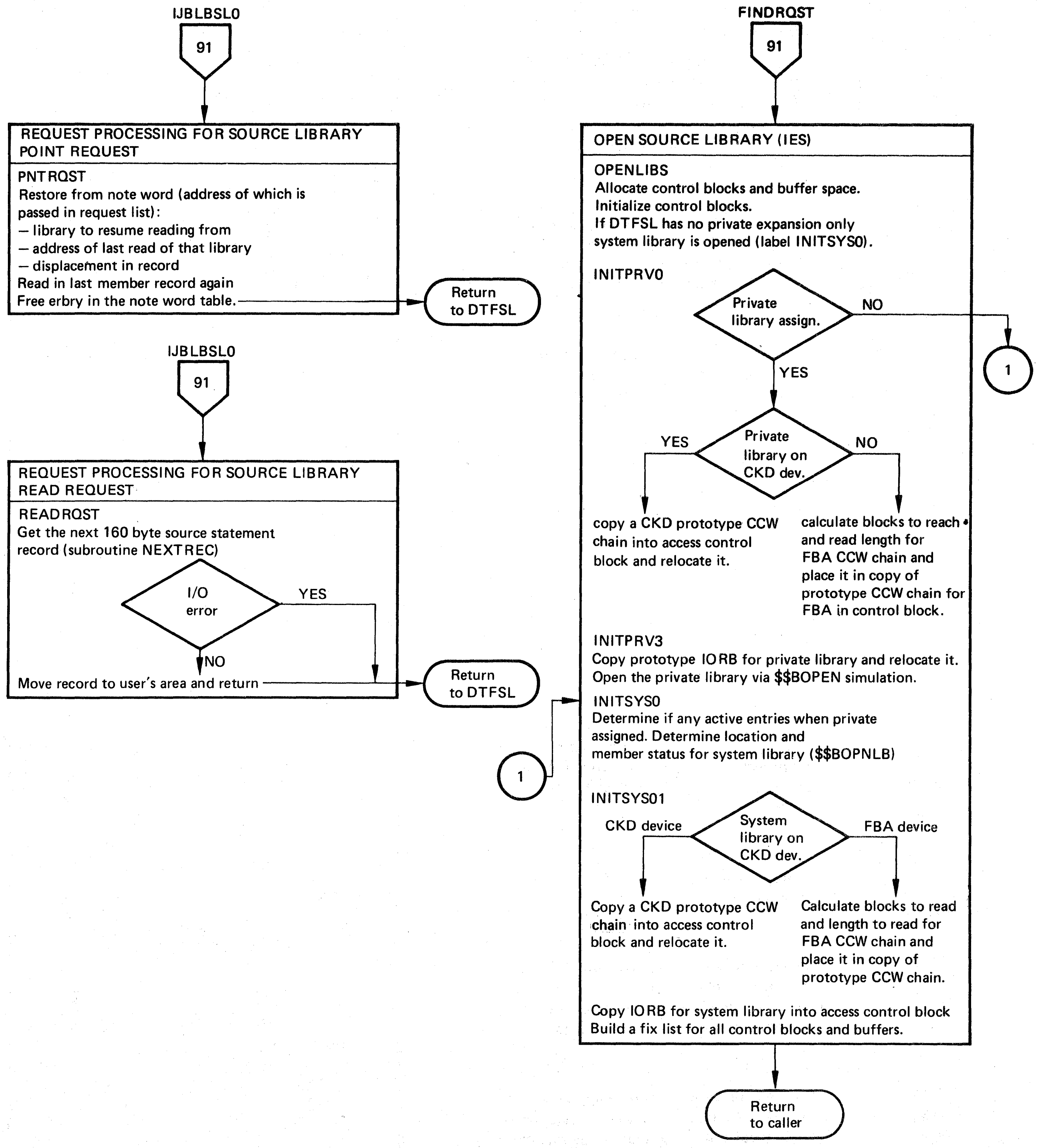

IJBLBSLO
91
REQUEST PROCESSING FOR SOURCE LIBRARY GET REQUEST
GETRQST
Valid requests and no 'end-of-book'
NO
Return to DTFSL
Note: 'Invalid Request' set on in Request List
YES
At first GET a 160 byte source record is read (label GETREC-subroutine NEXTREC)
ENDFIELD
End of book reached
YES
Return to DTFSL
Note: 'End-of-Book' is set on in Request List
NO
DECOMPR
Move non-blank characters in source record to user card area until this area filled.
Card area filled
YES
Return to user
NO
ENDFIELD (2nd time)
If a card spans two 160 byte records get next record (label GETREC-subrout. NEXTREC)
IJBLBSLO
91
REQUEST PROCESSING FOR SOURCE LIBRARY NOTE REQUEST
NOTERQST
Successful find and valid request
NO
Return to DTFSL
Note: Invalid Request is set on in Request List
YES
Get a free entry in the note word table.
Save in note word (address obtained from request list):
– the library the member is found
– the actual disk address of member record read in
– the displacement in the member record
Return to DTFSL

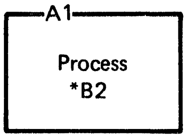
Chart 93. $IJBLBSL (Part 3 of 3)


IJBLBSLO
91
REQUEST PROCESSING FOR SOURCE LIBRARY POINT REQUEST
PNTRQST
Restore from note word (address of which is passed in request list):
– library to resume reading from
– address of last read of that library
– displacement in record
Read in last member record again
Free erbry in the note word table.
Return to DTFSL
IJBLBSLO
91
REQUEST PROCESSING FOR SOURCE LIBRARY READ REQUEST
READRQST
Get the next 160 byte source statement record (subroutine NEXTREC)
I/O error
YES
NO
Move record to user's area and return
Return to DTFSL
1
FINDRQST
91
OPEN SOURCE LIBRARY (IES)
OPENLIBS
Allocate control blocks and buffer space.
Initialize control blocks.
If DTFSL has no private expansion only system library is opened (label INITSYSO).
INITPRVO
Private library assign.
NO
1
YES
Private library on CKD dev.
YES
NO
copy a CKD prototype CCW chain into access control block and relocate it.
calculate blocks to reach and read length for FBA CCW chain and place it in copy of prototype CCW chain for FBA in control block.
INITPRV3
Copy prototype IORB for private library and relocate it.
Open the private library via $$BOPEN simulation.
INITSYSO
Determine if any active entries when private assigned. Determine location and member status for system library ($$BOPNLB)
INITSYSO1
CKD device
System library on CKD dev.
FBA device
Copy a CKD prototype CCW chain into access control block and relocate it.
Calculate blocks to read and length to read for FBA CCW chain and place it in copy of prototype CCW chain.
Copy IORB for system library into access control block
Build a fix list for all control blocks and buffers.
Return to caller

DETAIL CHARTS

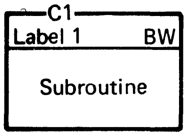
Explanation of Flowchart Symbols

DESCRIPTION

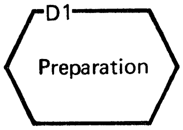


A group of program instructions that perform a processing function of the program. The label, if any, is shown above the block.

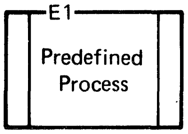
*B2
If any additional explanation is required, its location on the chart is identified by an asterisk and the block ID.



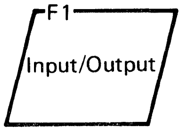
Description of a subroutine. The starting label of the routine appears above the stripe. If the subroutine is documented in detail on another flowchart, the ID of this flowchart is also shown.



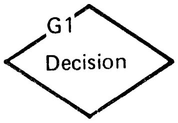
An instruction, or group of instructions, that changes portions of a routine or initializes a routine for given conditions.



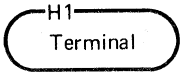
A group of operations not detailed in the flowcharts in this manual, such as user's routines.



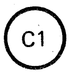
Any function of an input/output device or program, usually branching to an I/O routine to perform the function stated in the block.



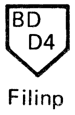
Points where the program branches to alternate processing, based upon variable conditions such as program switch settings and test results.



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



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



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

EXAMPLE

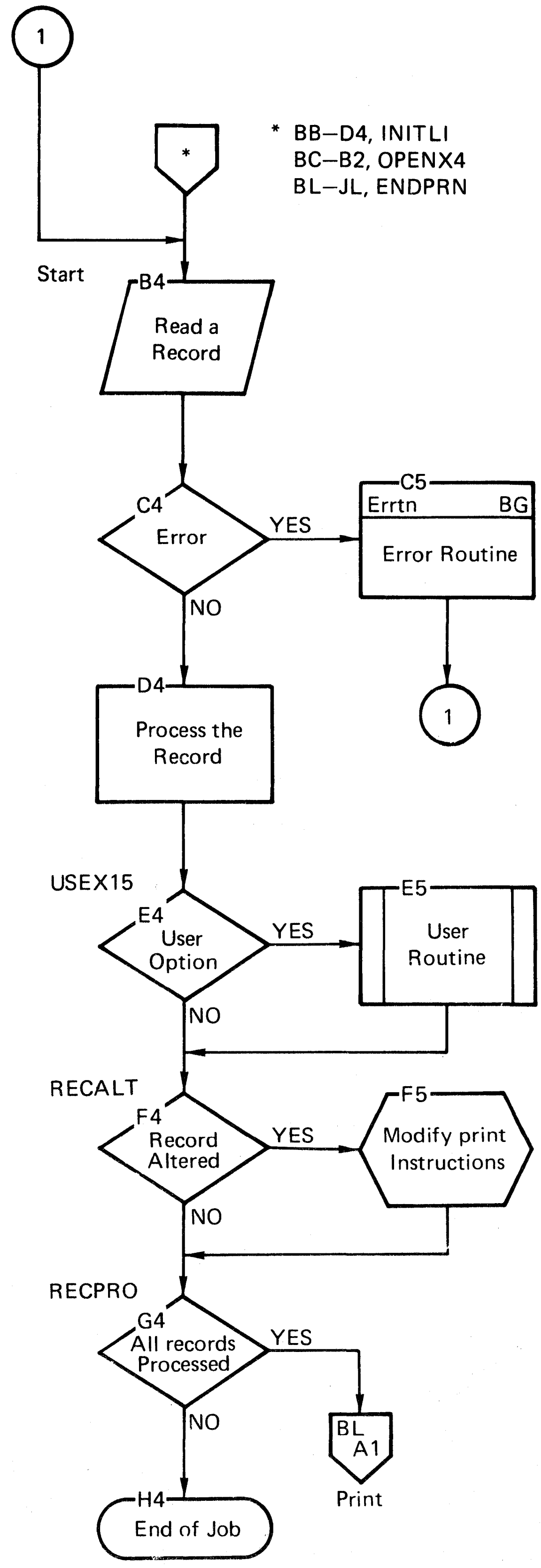


Chart AA. COPYSERV
Refer to Chart 01

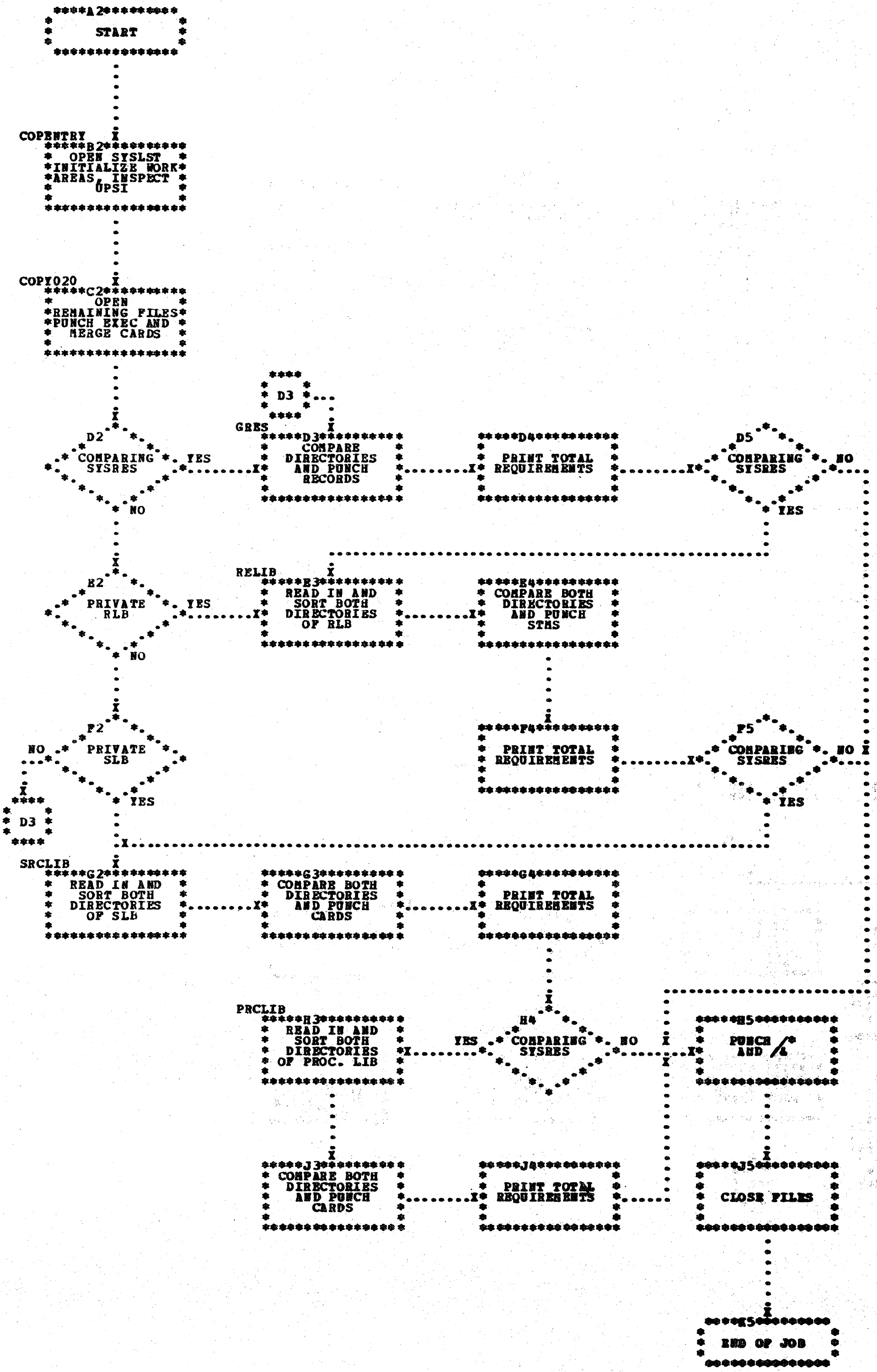

A2
START
COPENTRY
B2
OPEN SYSLST INITIALIZE WORK AREAS, INSPECT UPSI
COPY020
C2
OPEN REMAINING FILES PUNCH EXEC AND MERGE CARDS
D3
D2
COMPARING SYSRES
YES
NO
GRES
D3
COMPARE DIRECTORIES AND PUNCH RECORDS
D4
PRINT TOTAL REQUIREMENTS
D5
COMPARING SYSRES
NO
YES
E2
PRIVATE RLB
YES
NO
RELIB
E3
READ IN AND SORT BOTH DIRECTORIES OF RLB
E4
COMPARE BOTH DIRECTORIES AND PUNCH STMS
F2
PRIVATE SLB
NO
YES
D3
F4
PRINT TOTAL REQUIREMENTS
F5
COMPARING SYSRES
NO
YES
SRCLIB
G2
READ IN AND SORT BOTH DIRECTORIES OF SLB
G3
COMPARE BOTH DIRECTORIES AND PUNCH CARDS
G4
PRINT TOTAL REQUIREMENTS
PRCLIB
H3
READ IN AND SORT BOTH DIRECTORIES OF PROC. LIB
H4
COMPARING SYSRES
YES
NO
H5
PUNCH /* AND /&
J3
COMPARE BOTH DIRECTORIES AND PUNCH CARDS
J4
PRINT TOTAL REQUIREMENTS
J5
CLOSE FILES
K5
END OF JOB

Chart EX. MAINTCL
Refer to Chart 24

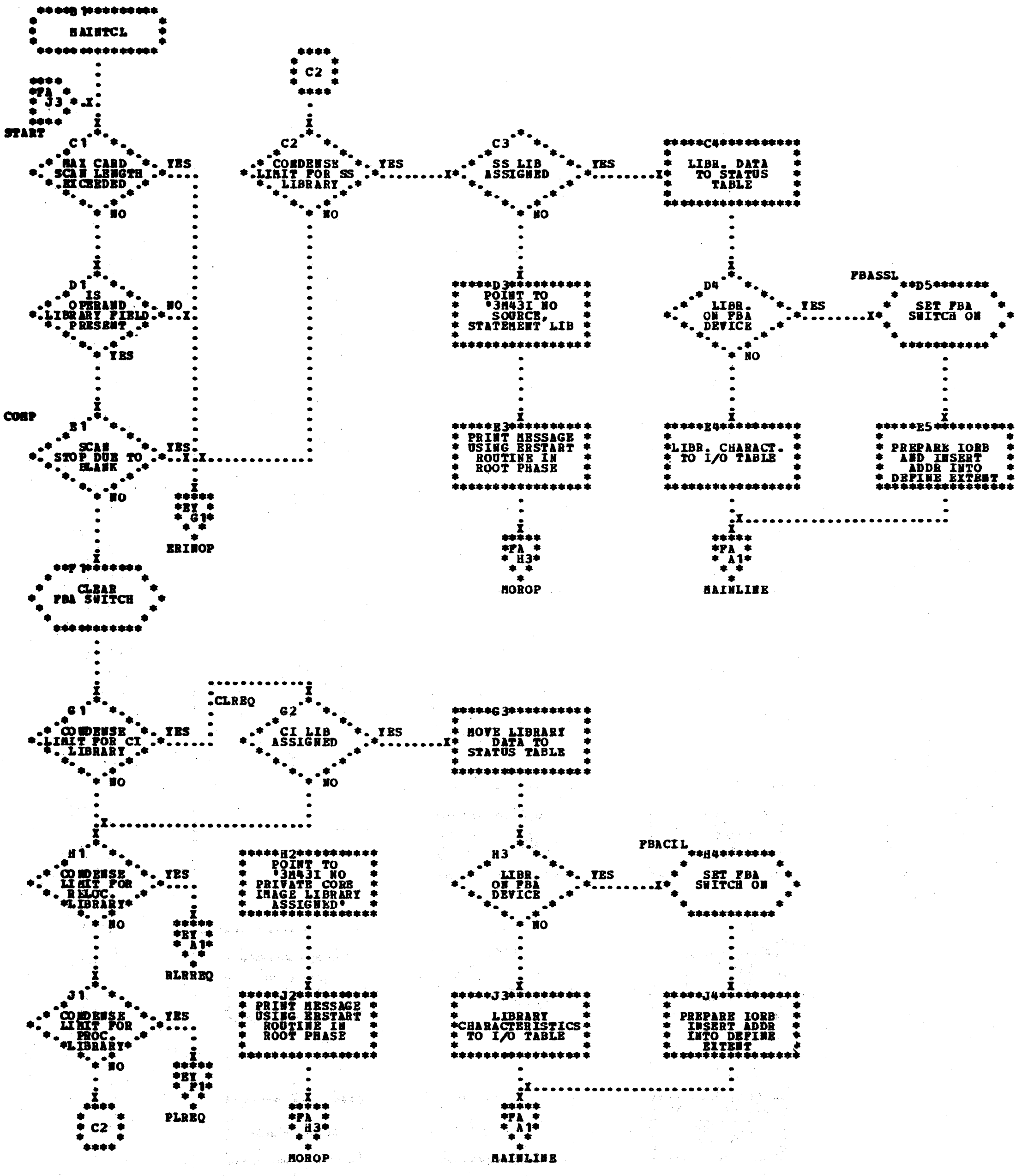

B1
MAINTCL
FA J3
START
C1 MAX CARD SCAN LENGTH EXCEEDED
YES
NO
D1 IS OPERAND LIBRARY FIELD PRESENT
NO
YES
COMP
E1 SCAN STOP DUE TO BLANK
YES
NO
EY G1
ERINOP
F1 CLEAR FBA SWITCH
C2
C2 CONDENSE LIMIT FOR SS LIBRARY
YES
NO
C3 SS LIB ASSIGNED
YES
NO
C4 LIBR. DATA TO STATUS TABLE
D3 POINT TO '3M43I NO SOURCE, STATEMENT LIB
E3 PRINT MESSAGE USING ERSTART ROUTINE IN ROOT PHASE
FA H3
MOROP
D4 LIBR. ON FBA DEVICE
YES
NO
FBASSL
D5 SET FBA SWITCH ON
E4 LIBR. CHARACT. TO I/O TABLE
E5 PREPARE IORB AND INSERT ADDR INTO DEFINE EXTENT
FA A1
MAINLINE
G1 CONDENSE LIMIT FOR CI LIBRARY
YES
NO
CLREQ
G2 CI LIB ASSIGNED
YES
NO
G3 MOVE LIBRARY DATA TO STATUS TABLE
H1 CONDENSE LIMIT FOR RELOC. LIBRARY
YES
NO
EY A1
RLRREQ
H2 POINT TO '3M43I NO PRIVATE CORE IMAGE LIBRARY ASSIGNED'
J1 CONDENSE LIMIT FOR PROC. LIBRARY
YES
NO
EY F1
PLREQ
C2
J2 PRINT MESSAGE USING ERSTART ROUTINE IN ROOT PHASE
FA H3
MOROP
H3 LIBR. ON FBA DEVICE
YES
NO
FBACIL
H4 SET FBA SWITCH ON
J3 LIBRARY CHARACTERISTICS TO I/O TABLE
J4 PREPARE IORB INSERT ADDR INTO DEFINE EXTENT
FA A1
MAINLINE

Chart EY. MAINTCL
Refer to Chart 24

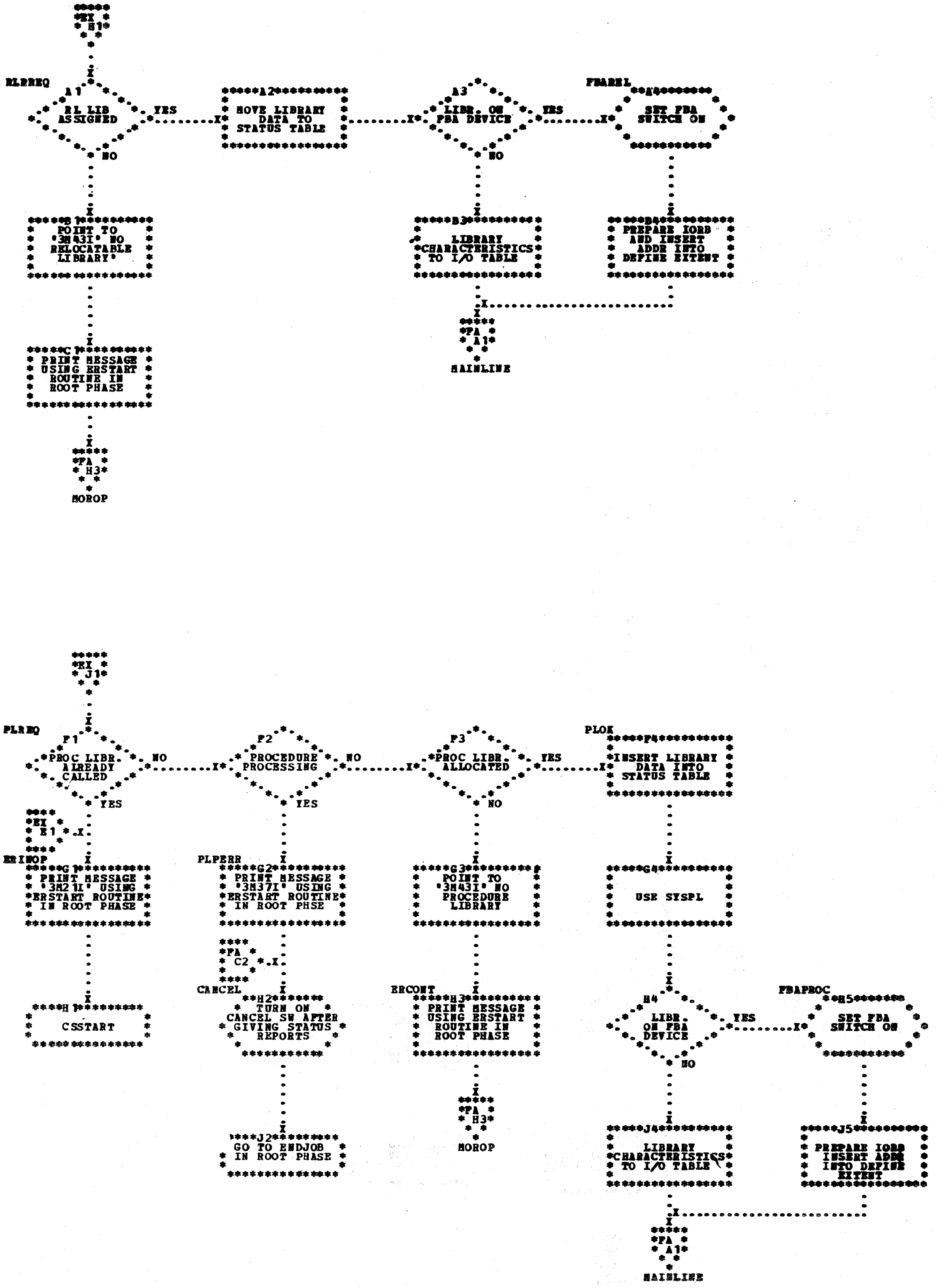

EY H1
RLREQ
A1
RL LIB ASSIGNED
YES
NO
A2
MOVE LIBRARY DATA TO STATUS TABLE
A3
LIBR. ON FBA DEVICE
YES
NO
FBAREL
A4
SET FBA SWITCH ON
B1
POINT TO '3M43I' NO RELOCATABLE LIBRARY'
B3
LIBRARY CHARACTERISTICS TO I/O TABLE
B4
PREPARE IORB AND INSERT ADDR INTO DEFINE EXTENT
FA A1
MAINLINE
C1
PRINT MESSAGE USING ERSTART ROUTINE IN ROOT PHASE
FA H3
NOROP
EY J1
PLREQ
F1
PROC LIBR. ALREADY CALLED
NO
YES
F2
PROCEDURE PROCESSING
NO
YES
F3
PROC LIBR. ALLOCATED
YES
NO
PLOK
F4
INSERT LIBRARY DATA INTO STATUS TABLE
EY E1
ERINOP
G1
PRINT MESSAGE '3M21I' USING ERSTART ROUTINE IN ROOT PHASE
PLPERR
G2
PRINT MESSAGE '3M37I' USING ERSTART ROUTINE IN ROOT PHSE
G3
POINT TO '3M43I' NO PROCEDURE LIBRARY
G4
USE SYSPL
FA C2
H1
CSSTART
CANCEL
H2
TURN ON CANCEL SW AFTER GIVING STATUS REPORTS
ERCONT
H3
PRINT MESSAGE USING ERSTART ROUTINE IN ROOT PHASE
H4
LIBR. ON FBA DEVICE
YES
NO
FBAPROC
H5
SET FBA SWITCH ON
J2
GO TO ENDJOB IN ROOT PHASE
FA H3
NOROP
J4
LIBRARY CHARACTERISTICS TO I/O TABLE
J5
PREPARE IORB INSERT ADDR INTO DEFINE EXTENT
FA A1
MAINLINE

Chart FA. MAINTCL
Refer to Chart 24

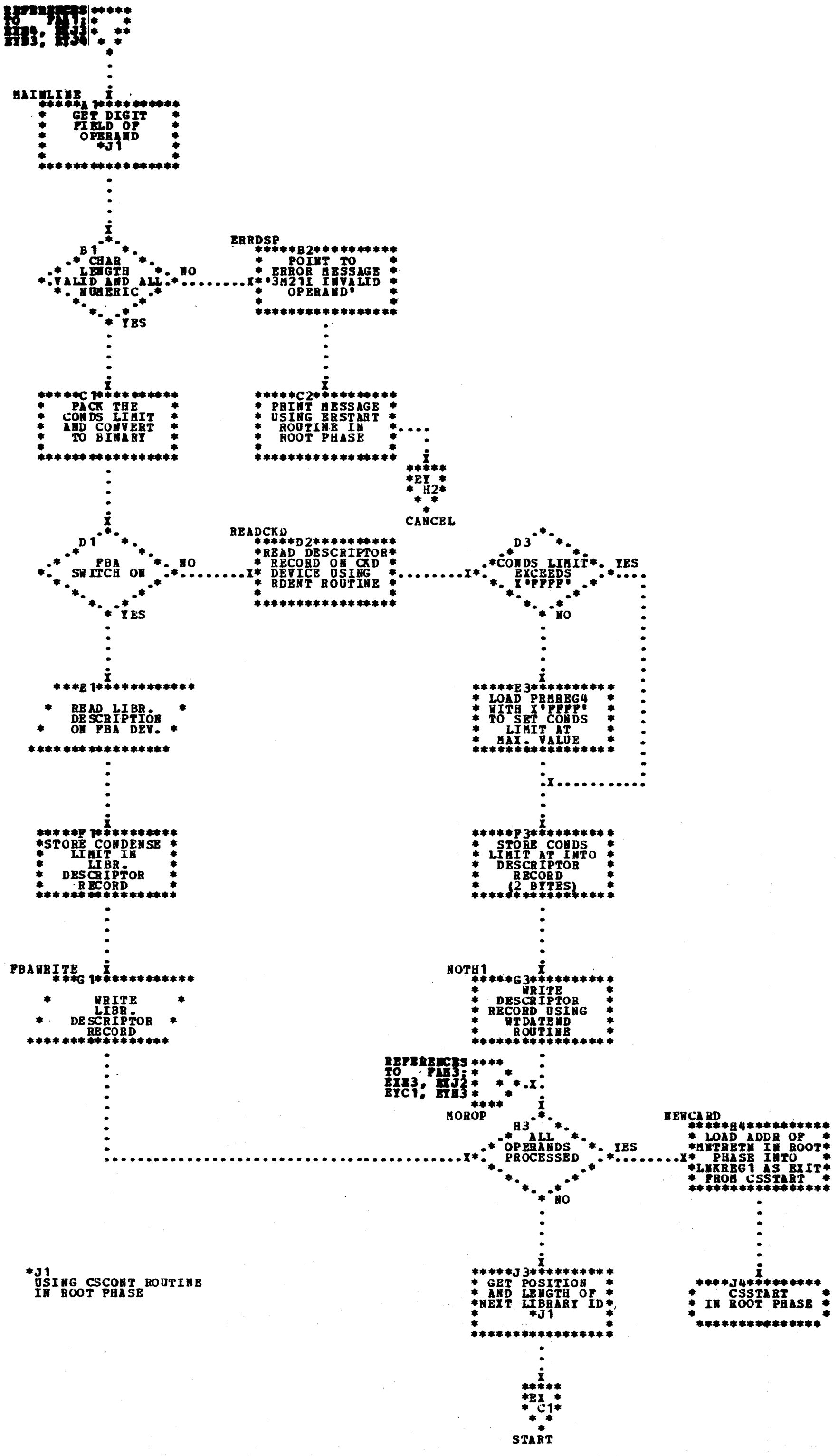

MAINLINE
A1
GET DIGIT FIELD OF OPERAND *J1
B1
CHAR LENGTH VALID AND ALL NUMERIC
NO
YES
ERRDSP
B2
POINT TO ERROR MESSAGE '3M21I INVALID OPERAND'
C1
PACK THE CONDS LIMIT AND CONVERT TO BINARY
C2
PRINT MESSAGE USING ERSTART ROUTINE IN ROOT PHASE
EY H2
CANCEL
D1
FBA SWITCH ON
NO
YES
READCKD
D2
READ DESCRIPTOR RECORD ON CKD DEVICE USING RDENT ROUTINE
D3
CONDS LIMIT EXCEEDS X'FFFF'
YES
NO
E1
READ LIBR. DESCRIPTION ON FBA DEV.
E3
LOAD PRMREG4 WITH X'FFFF' TO SET CONDS LIMIT AT MAX. VALUE
F1
STORE CONDENSE LIMIT IN LIBR. DESCRIPTOR RECORD
F3
STORE CONDS LIMIT AT INTO DESCRIPTOR RECORD (2 BYTES)
FBAWRITE
G1
WRITE LIBR. DESCRIPTOR RECORD
NOTH1
G3
WRITE DESCRIPTOR RECORD USING WTDATEND ROUTINE
REFERENCES TO
MOROP
H3
ALL OPERANDS PROCESSED
YES
NO
NEWCARD
H4
LOAD ADDR OF MNTRETN IN ROOT PHASE INTO LNKREG1 AS EXIT FROM CSSTART
J3
GET POSITION AND LENGTH OF NEXT LIBRARY ID *J1
J4
CSSTART IN ROOT PHASE
EX C1
START
*J1
USING CSCONT ROUTINE IN ROOT PHASE

Chart FD. MAINTCN - Initialize for Library Condense
Refer to Chart 25

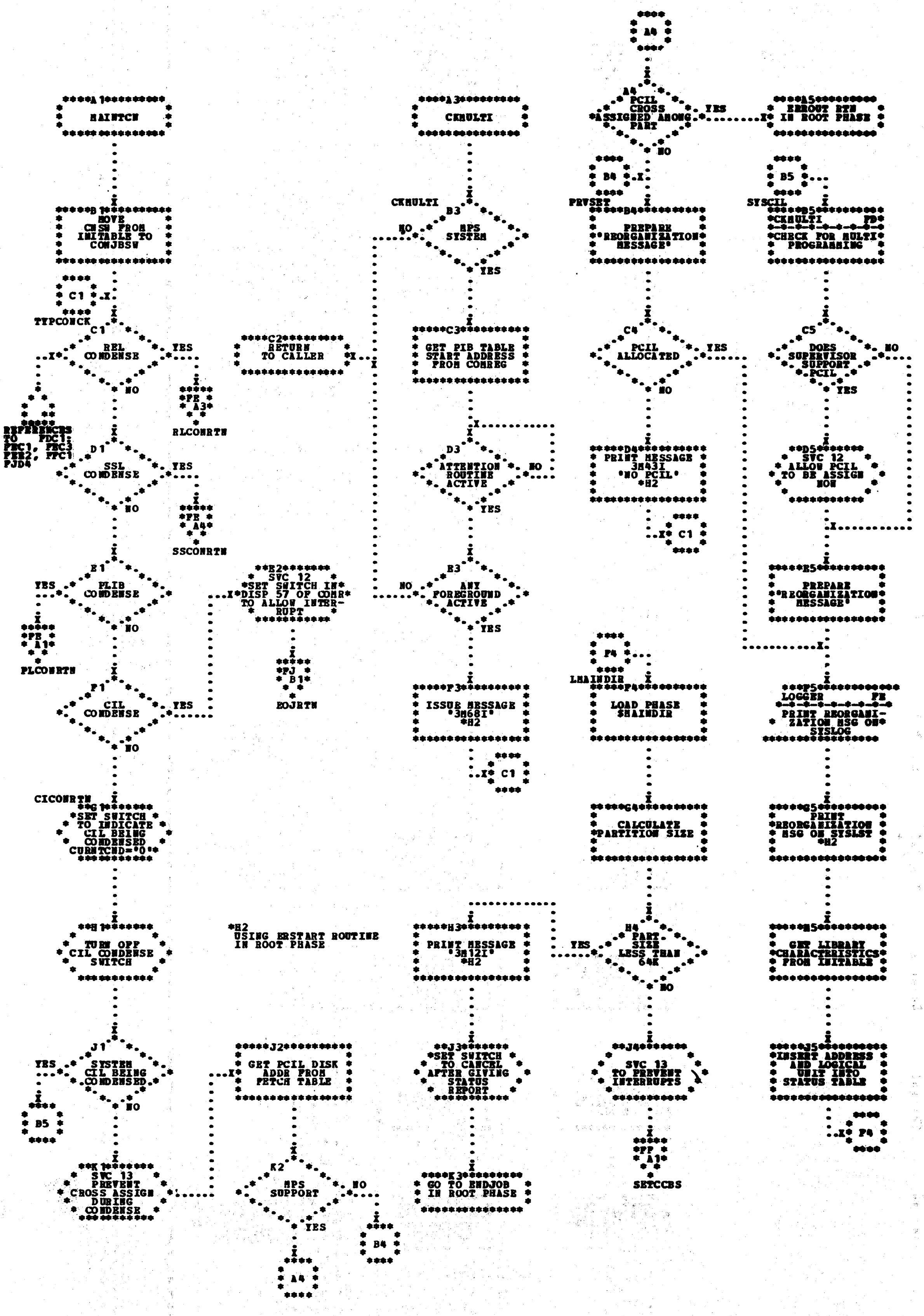

A1 MAINTCN
B1 MOVE CNSW FROM INITABLE TO CONJBSW
C1
TYPCONCK
C1 REL CONDENSE
YES
NO
FE A3
RLCONRTN
REFERENCES TO FDC1: FBC1, FBC3 FEB2, FFC1 FJD4
D1 SSL CONDENSE
YES
NO
FE A4
SSCONRTN
E1 PLIB CONDENSE
YES
NO
FE A1
PLCONRTN
F1 CIL CONDENSE
YES
NO
CICONRTN
G1 SET SWITCH TO INDICATE CIL BEING CONDENSED CURNTCND='0'
H1 TURN OFF CIL CONDENSE SWITCH
J1 SYSTEM CIL BEING CONDENSED
YES
NO
B5
K1 SVC 13 PREVENT CROSS ASSIGN DURING CONDENSE
C2 RETURN TO CALLER
E2 SVC 12 SET SWITCH IN DISP 57 OF COMR TO ALLOW INTERRUPT
FJ B1
EOJRTN
*H2 USING ERSTART ROUTINE IN ROOT PHASE
J2 GET PCIL DISK ADDR FROM FETCH TABLE
K2 MPS SUPPORT
NO
YES
B4
A4
A3 CKMULTI
CKMULTI
B3 MPS SYSTEM
NO
YES
C3 GET PIB TABLE START ADDRESS FROM COMREG
D3 ATTENTION ROUTINE ACTIVE
NO
YES
E3 ANY FOREGROUND ACTIVE
NO
YES
F3 ISSUE MESSAGE '3M68I' *H2
C1
H3 PRINT MESSAGE '3M12I' *H2
J3 SET SWITCH TO CANCEL AFTER GIVING STATUS REPORT
K3 GO TO ENDJOB IN ROOT PHASE
A4
A4 PCIL CROSS ASSIGNED AMONG PART
YES
NO
B4
PRVSET
B4 PREPARE 'REORGANIZATION MESSAGE'
C4 PCIL ALLOCATED
YES
NO
D4 PRINT MESSAGE 3M43I 'NO PCIL' *H2
C1
F4
LMAINDIR
F4 LOAD PHASE $MAINDIR
G4 CALCULATE PARTITION SIZE
H4 PART SIZE LESS THAN 64K
YES
NO
J4 SVC 13 TO PREVENT INTERRUPTS
FF A1
SETCCBS
A5 ERROUT RTN IN ROOT PHASE
B5
SYSCIL
B5 CKMULTI FD CHECK FOR MULTI PROGRAMMING
C5 DOES SUPERVISOR SUPPORT PCIL
NO
YES
D5 SVC 12 ALLOW PCIL TO BE ASSIGN NOW
E5 PREPARE 'REORGANIZATION MESSAGE'
F5 LOGGER FE PRINT REORGANIZATION MSG ON SYSLOG
G5 PRINT REORGANIZATION MSG ON SYSLST *H2
H5 GET LIBRARY CHARACTERISTICS FROM INITABLE
J5 INSERT ADDRESS AND LOGICAL UNIT INTO STATUS TABLE
F4

Chart FE. MAINTCN - Initialize for Library Condense
Refer to Chart 25

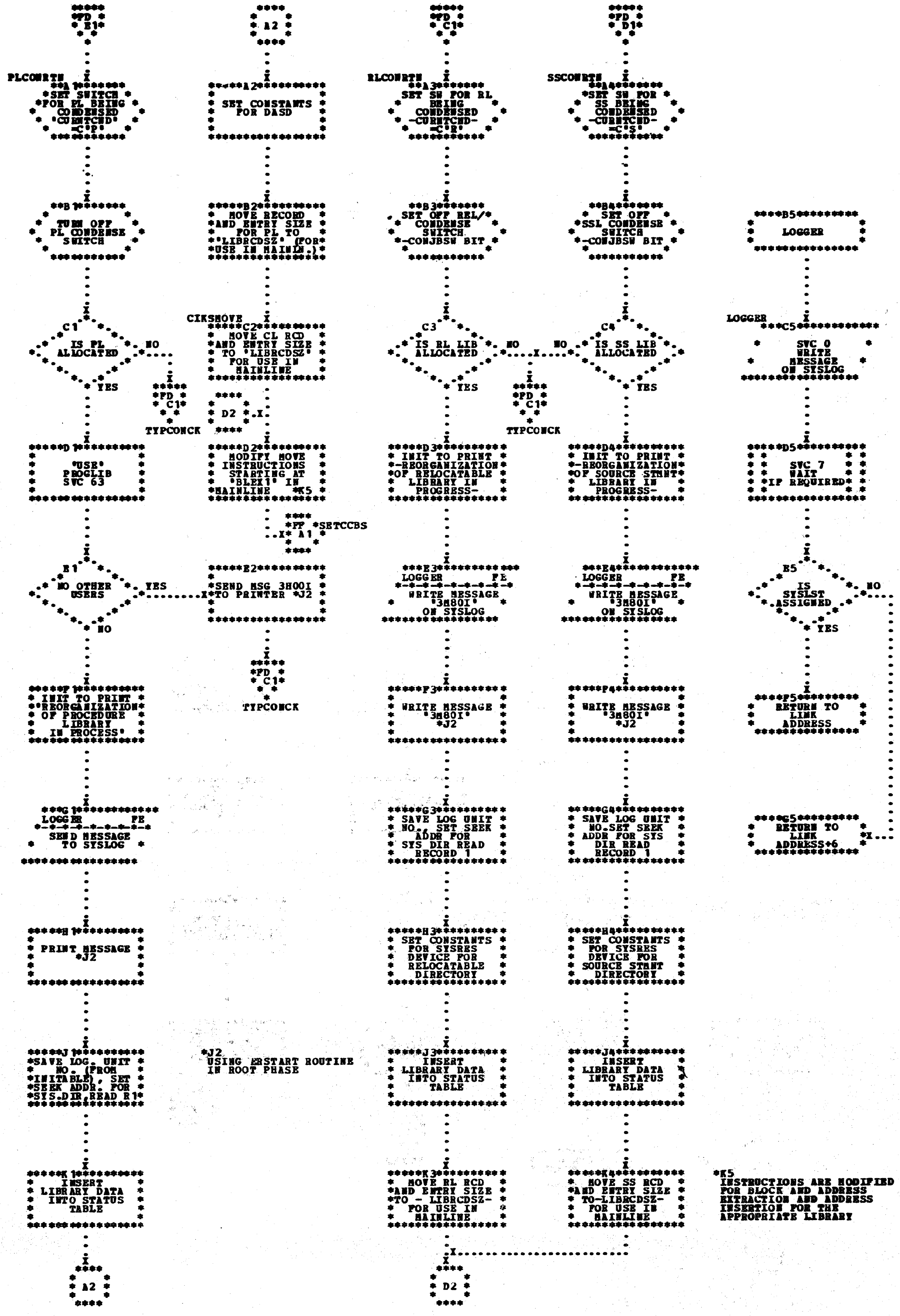

FD B1
A2
FD C1
FD D1
PLCONRTN
A1 SET SWITCH FOR PL BEING CONDENSED 'CURNTCND' =C'P'
A2 SET CONSTANTS FOR DASD
RLCONRTN
A3 SET SW FOR RL BEING CONDENSED -CURNTCND- =C'R'
SSCONRTN
A4 SET SW FOR SS BEING CONDENSED -CURNTCND- =C'S'
B1 TURN OFF PL CONDENSE SWITCH
B2 MOVE RECORD AND ENTRY SIZE FOR PL TO 'LIBRCDSZ' (FOR USE IN MAINLN.)
B3 SET OFF REL/ CONDENSE SWITCH -CONJBSW BIT
B4 SET OFF SSL CONDENSE SWITCH -CONJBSW BIT
B5 LOGGER
C1 IS PL ALLOCATED
NO
YES
FD C1 TYPCONCK
CIKSMOVE
C2 MOVE CL RCD AND ENTRY SIZE TO 'LIBRCDSZ' FOR USE IN MAINLINE
D2
C3 IS RL LIB ALLOCATED
NO
YES
FD C1 TYPCONCK
C4 IS SS LIB ALLOCATED
NO
YES
LOGGER
C5 SVC 0 WRITE MESSAGE ON SYSLOG
D1 'USE' PROGLIB SVC 63
D2 MODIFY MOVE INSTRUCTIONS STARTING AT 'BLEX1' IN MAINLINE *K5
D3 INIT TO PRINT -REORGANIZATION OF RELOCATABLE LIBRARY IN PROGRESS-
D4 INIT TO PRINT -REORGANIZATION OF SOURCE STMNT LIBRARY IN PROGRESS-
D5 SVC 7 WAIT IF REQUIRED
FF A1 SETCCBS
E1 NO OTHER USERS
YES
NO
E2 SEND MSG 3H00I TO PRINTER *J2
E3 LOGGER FE WRITE MESSAGE '3H80I' ON SYSLOG
E4 LOGGER FE WRITE MESSAGE '3H80I' ON SYSLOG
E5 IS SYSLST ASSIGNED
NO
YES
FD C1 TYPCONCK
F1 INIT TO PRINT 'REORGANIZATION OF PROCEDURE LIBRARY IN PROCESS'
F3 WRITE MESSAGE '3H80I' *J2
F4 WRITE MESSAGE '3H80I' *J2
F5 RETURN TO LINK ADDRESS
G1 LOGGER FE SEND MESSAGE TO SYSLOG
G3 SAVE LOG UNIT NO., SET SEEK ADDR FOR SYS DIR READ RECORD 1
G4 SAVE LOG UNIT NO.SET SEEK ADDR FOR SYS DIR READ RECORD 1
G5 RETURN TO LINK ADDRESS+6
H1 PRINT MESSAGE *J2
H3 SET CONSTANTS FOR SYSRES DEVICE FOR RELOCATABLE DIRECTORY
H4 SET CONSTANTS FOR SYSRES DEVICE FOR SOURCE STMNT DIRECTORY
J1 SAVE LOG. UNIT NO. (FROM INITABLE), SET SEEK ADDR. FOR SYS.DIR.READ R1
*J2 USING ERSTART ROUTINE IN ROOT PHASE
J3 INSERT LIBRARY DATA INTO STATUS TABLE
J4 INSERT LIBRARY DATA INTO STATUS TABLE
K1 INSERT LIBRARY DATA INTO STATUS TABLE
K3 MOVE RL RCD AND ENTRY SIZE TO - LIBRCDSZ- FOR USE IN MAINLINE
K4 MOVE SS RCD AND ENTRY SIZE TO-LIBRCDSZ- FOR USE IN MAINLINE
*K5 INSTRUCTIONS ARE MODIFIED FOR BLOCK AND ADDRESS EXTRACTION AND ADDRESS INSERTION FOR THE APPROPRIATE LIBRARY
A2
D2

Chart FF. MAINTCN - Condense Directories
Refer to Chart 25

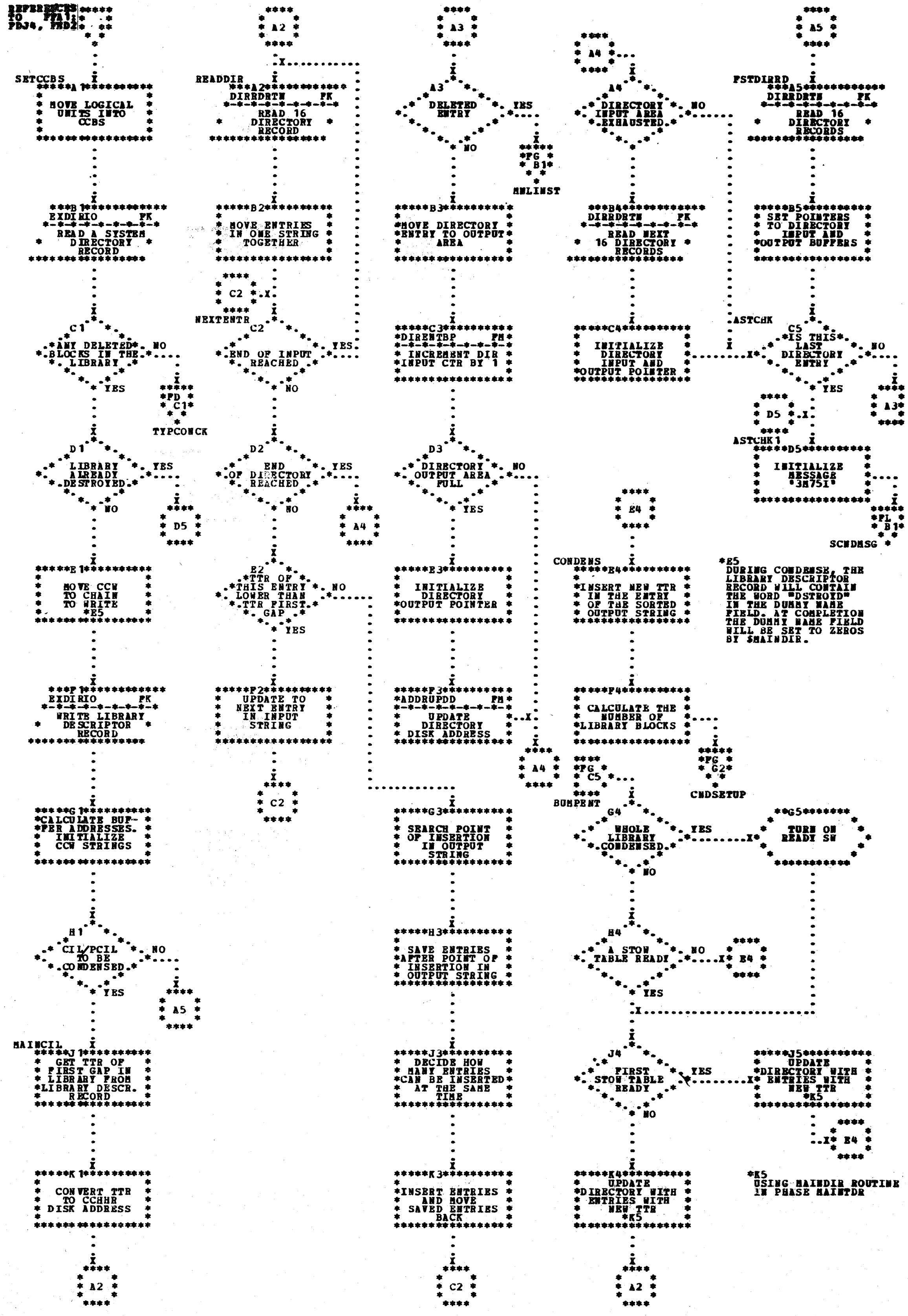

REFERENCES TO FFA1, FDJ4, FHD2
A2
A3
A4
A5
SETCCBS
A1
MOVE LOGICAL UNITS INTO CCBS
B1
EXDIRIO PK
READ A SYSTEM DIRECTORY RECORD
C1
ANY DELETED BLOCKS IN THE LIBRARY
NO
YES
FD C1
TYPCONCK
D1
LIBRARY ALREADY DESTROYED
YES
NO
D5
E1
MOVE CCW TO CHAIN TO WRITE *E5
F1
EXDIRIO PK
WRITE LIBRARY DESCRIPTOR RECORD
G1
CALCULATE BUFFER ADDRESSES. INITIALIZE CCW STRINGS
H1
CIL/PCIL TO BE CONDENSED
NO
YES
A5
MAINCIL
J1
GET TTR OF FIRST GAP IN LIBRARY FROM LIBRARY DESCR. RECORD
K1
CONVERT TTR TO CCHHR DISK ADDRESS
A2
READDIR
A2
DIRRDRTN PK
READ 16 DIRECTORY RECORD
B2
MOVE ENTRIES IN ONE STRING TOGETHER
C2
NEXTENTR
C2
END OF INPUT REACHED
YES
NO
D2
END OF DIRECTORY REACHED
YES
NO
A4
E2
TTR OF THIS ENTRY LOWER THAN TTR FIRST GAP
NO
YES
F2
UPDATE TO NEXT ENTRY IN INPUT STRING
C2
A3
DELETED ENTRY
YES
NO
FG B1
MNLINST
B3
MOVE DIRECTORY ENTRY TO OUTPUT AREA
C3
DIRENTBP PM
INCREMENT DIR INPUT CTR BY 1
D3
DIRECTORY OUTPUT AREA FULL
NO
YES
E3
INITIALIZE OUTPUT POINTER
F3
ADDRUPDD PM
UPDATE DIRECTORY DISK ADDRESS
A4
G3
SEARCH POINT OF INSERTION IN OUTPUT STRING
H3
SAVE ENTRIES AFTER POINT OF INSERTION IN OUTPUT STRING
J3
DECIDE HOW MANY ENTRIES CAN BE INSERTED AT THE SAME TIME
K3
INSERT ENTRIES AND MOVE SAVED ENTRIES BACK
C2
A4
A4
DIRECTORY INPUT AREA EXHAUSTED
NO
B4
DIRRDRTN PK
READ NEXT 16 DIRECTORY RECORDS
C4
INITIALIZE DIRECTORY INPUT AND OUTPUT POINTER
E4
CONDENS
E4
INSERT NEW TTR IN THE ENTRY OF THE SORTED OUTPUT STRING
F4
CALCULATE THE NUMBER OF LIBRARY BLOCKS
FG G2
CNDSETUP
FG C5
BUMPENT
G4
WHOLE LIBRARY CONDENSED
YES
NO
G5
TURN ON READY SW
H4
A STOW TABLE READY
NO
E4
YES
J4
FIRST STOW TABLE READY
YES
NO
K4
UPDATE DIRECTORY WITH ENTRIES WITH NEW TTR *K5
A2
A5
FSTDIRRD
A5
DIRRDRTN PK
READ 16 DIRECTORY RECORDS
B5
SET POINTERS TO DIRECTORY INPUT AND OUTPUT BUFFERS
ASTCHK
C5
IS THIS LAST DIRECTORY ENTRY
NO
A3
YES
D5
ASTCHK1
D5
INITIALIZE MESSAGE '3N75I'
FL B1
SCNDMSG
*E5
DURING CONDENSE, THE LIBRARY DESCRIPTOR RECORD WILL CONTAIN THE WORD "DSTROYD" IN THE DUMMY NAME FIELD. AT COMPLETION THE DUMMY NAME FIELD WILL BE SET TO ZEROS BY $MAINDIR.
J5
UPDATE DIRECTORY WITH ENTRIES WITH NEW TTR *K5
E4
*K5
USING MAINDIR ROUTINE IN PHASE MAINTDR

Chart FG. MAINTCN - Condense Directories
Refer to Chart 25

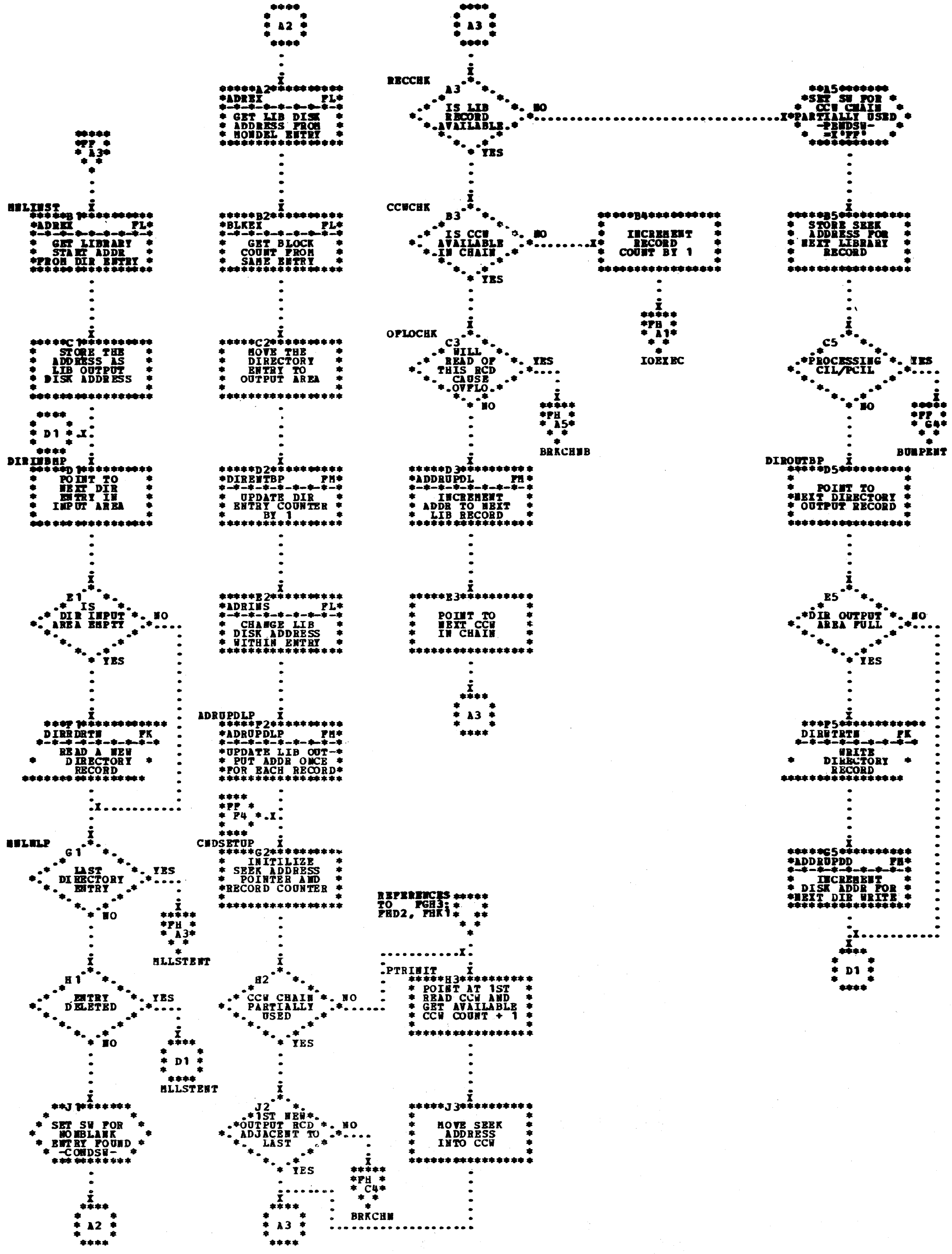
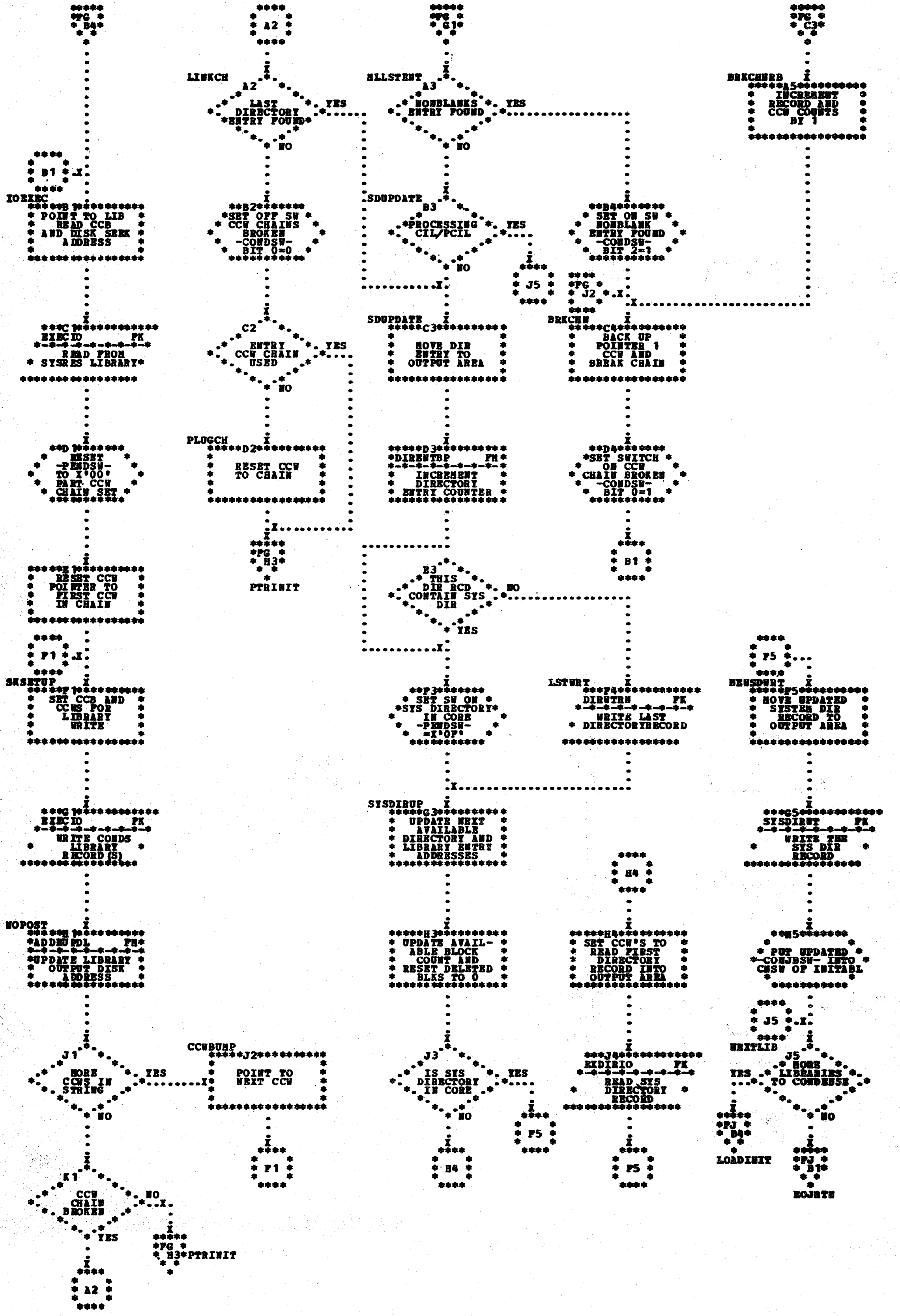
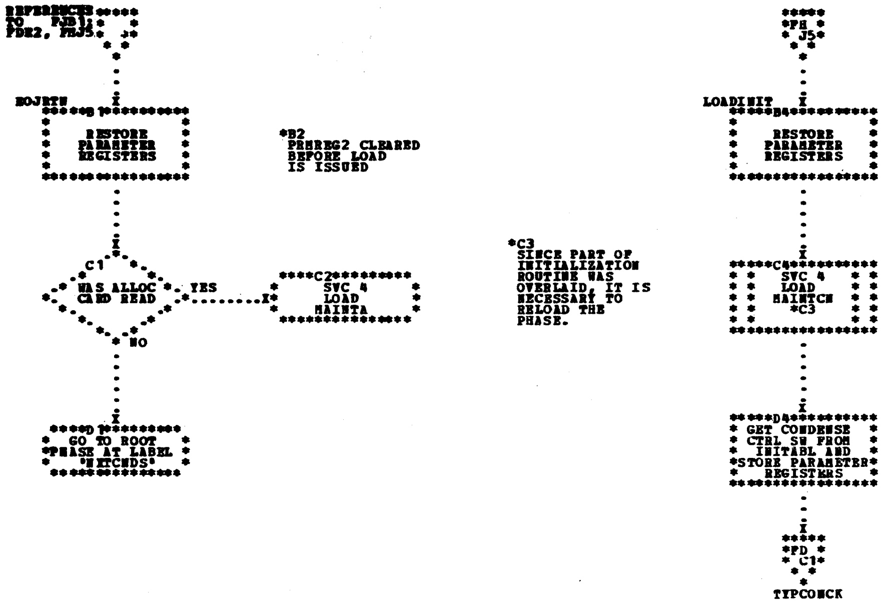

A2
A2
ADREX PL
GET LIB DISK ADDRESS FROM NONDEL ENTRY
FF A3
NNLINST
B1
ADREX PL
GET LIBRARY START ADDR FROM DIR ENTRY
B2
BLKEX PL
GET BLOCK COUNT FROM SAME ENTRY
C1
STORE THE ADDRESS AS LIB OUTPUT DISK ADDRESS
C2
MOVE THE DIRECTORY ENTRY TO OUTPUT AREA
D1
DIRINBMP
D1
POINT TO NEXT DIR ENTRY IN INPUT AREA
D2
DIRENTBP FM
UPDATE DIR ENTRY COUNTER BY 1
E1
IS DIR INPUT AREA EMPTY
NO
YES
E2
ADRINS PL
CHANGE LIB DISK ADDRESS WITHIN ENTRY
F1
DIRRDRTN FK
READ A NEW DIRECTORY RECORD
ADRUPDLP
F2
ADRUPDLP FM
UPDATE LIB OUT-PUT ADDR ONCE FOR EACH RECORD
FF F4
NNLNLP
G1
LAST DIRECTORY ENTRY
YES
NO
FH A3
NLLSTENT
CNDSETUP
G2
INITILIZE SEEK ADDRESS POINTER AND RECORD COUNTER
H1
ENTRY DELETED
YES
NO
D1
NLLSTENT
H2
CCW CHAIN PARTIALLY USED
NO
YES
J1
SET SW FOR NONBLANK ENTRY FOUND -CONDSW-
A2
J2
1ST NEW OUTPUT RCD ADJACENT TO LAST
NO
YES
FH C4
BRKCHN
A3
A3
RECCHK
A3
IS LIB RECORD AVAILABLE
NO
YES
CCWCHK
B3
IS CCW AVAILABLE IN CHAIN
NO
YES
B4
INCREMENT RECORD COUNT BY 1
FH A1
IOEXEC
OFLOCHK
C3
WILL READ OF THIS RCD CAUSE OVFLO
YES
NO
FH A5
BRKCHNB
D3
ADDRUPDL FM
INCREMENT ADDR TO NEXT LIB RECORD
E3
POINT TO NEXT CCW IN CHAIN
A3
REFERENCES TO FGH3, FHD2, FHK1
PTRINIT
H3
POINT AT 1ST READ CCW AND GET AVAILABLE CCW COUNT + 1
J3
MOVE SEEK ADDRESS INTO CCW
A5
SET SW FOR CCW CHAIN PARTIALLY USED -PENDSW- =X'FF'
B5
STORE SEEK ADDRESS FOR NEXT LIBRARY RECORD
C5
PROCESSING CIL/PCIL
YES
NO
FF G4
BUMPENT
DIROUTBP
D5
POINT TO NEXT DIRECTORY OUTPUT RECORD
E5
DIR OUTPUT AREA FULL
NO
YES
F5
DIRWTRTN FK
WRITE DIRECTORY RECORD
G5
ADDRUPDD FM
INCREMENT DISK ADDR FOR NEXT DIR WRITE
D1

Chart FH. MAINTCN - Write Condensed Library and Directory
Refer to Chart 25



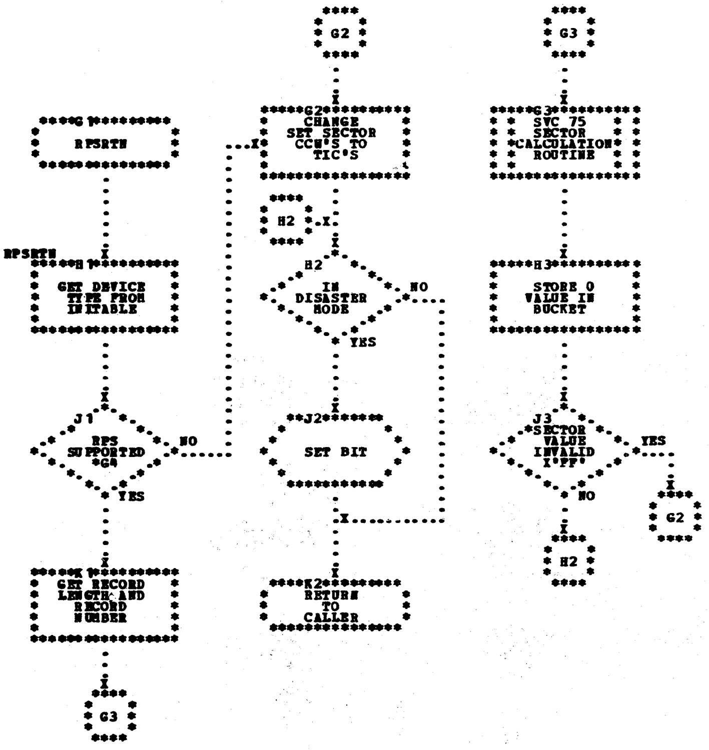
FG B4
A2
FG G1
FG C3
LINKCH A2
LAST DIRECTORY ENTRY FOUND
YES
NO
NLLSTENT A3
NONBLANKS ENTRY FOUND
YES
NO
BRKCHNRB A5
INCREMENT RECORD AND CCW COUNTS BY 1
B1
IOEXEC B1
POINT TO LIB READ CCB AND DISK SEEK ADDRESS
B2
SET OFF SW CCW CHAINS BROKEN -CONDSW- BIT 0=0
SDUPDATE B3
PROCESSING CIL/PCIL
YES
NO
J5
B4
SET ON SW NONBLANK ENTRY FOUND -CONDSW- BIT 2=1
FG J2
C1
EXECIO PK
READ FROM SYSRES LIBRARY
C2
ENTRY CCW CHAIN USED
YES
NO
SDUPDATE C3
MOVE DIR ENTRY TO OUTPUT AREA
BRKCHN C4
BACK UP POINTER 1 CCW AND BREAK CHAIN
D1
RESET -PENDSW- TO X'00' PART CCW CHAIN SET
PLUGCH D2
RESET CCW TO CHAIN
D3
DIRENTBP FH
INCREMENT DIRECTORY ENTRY COUNTER
D4
SET SWITCH ON CCW CHAIN BROKEN -CONDSW- BIT 0=1
FG H3
PTRINIT
B1
E1
RESET CCW POINTER TO FIRST CCW IN CHAIN
E3
THIS DIR RCD CONTAIN SYS DIR
NO
YES
F1
SKSETUP F1
SET CCB AND CCWS FOR LIBRARY WRITE
F3
SET SW ON SYS DIRECTORY IN CORE -PENDSW- =X'0F'
LSTWRT F4
DIRWTRN PK
WRITE LAST DIRECTORYRECORD
F5
NEWSDWRT F5
MOVE UPDATED SYSTEM DIR RECORD TO OUTPUT AREA
G1
EXECIO PK
WRITE CONDS LIBRARY RECORD(S)
SYSDIRUP G3
UPDATE NEXT AVAILABLE DIRECTORY AND LIBRARY ENTRY ADDRESSES
G5
SYSDIRWT PK
WRITE THE SYS DIR RECORD
H4
NOPOST H1
ADDNUPDL FH
UPDATE LIBRARY OUTPUT DISK ADDRESS
H3
UPDATE AVAIL- ABLE BLOCK COUNT AND RESET DELETED BLKS TO 0
H4
SET CCW'S TO READ FIRST DIRECTORY RECORD INTO OUTPUT AREA
H5
PUT UPDATED -CONJBSW- INTO CNSW OF INITABL
J5
J1
MORE CCWS IN STRING
YES
NO
CCWBUMP J2
POINT TO NEXT CCW
J3
IS SYS DIRECTORY IN CORE
YES
NO
J4
EXDIRIO PK
READ SYS DIRECTORY RECORD
NEXTLIB J5
MORE LIBRARIES TO CONDENSE
YES
NO
FJ B4
LOADINIT
FJ B1
EOJRTN
F1
H4
F5
F5
K1
CCW CHAIN BROKEN
NO
YES
FG H3 PTRINIT
A2

Chart FJ. MAINTCN - End Routine. Initiate Load for next Condense. RPS Routine
Refer to Chart 25


REFERENCES TO FJB1, FDK2, FHJ5
EOJRTN
B1
RESTORE PARAMETER REGISTERS
C1
WAS ALLOC CARD READ
YES
NO
C2
SVC 4 LOAD MAINTA
D1
GO TO ROOT PHASE AT LABEL 'NXTCMDS'
FH J5
LOADINIT
B4
RESTORE PARAMETER REGISTERS
C4
SVC 4 LOAD MAINTCN *C3
D4
GET CONDENSE CTRL SW FROM INITABL AND STORE PARAMETER REGISTERS
FD C1
TYPCONCK

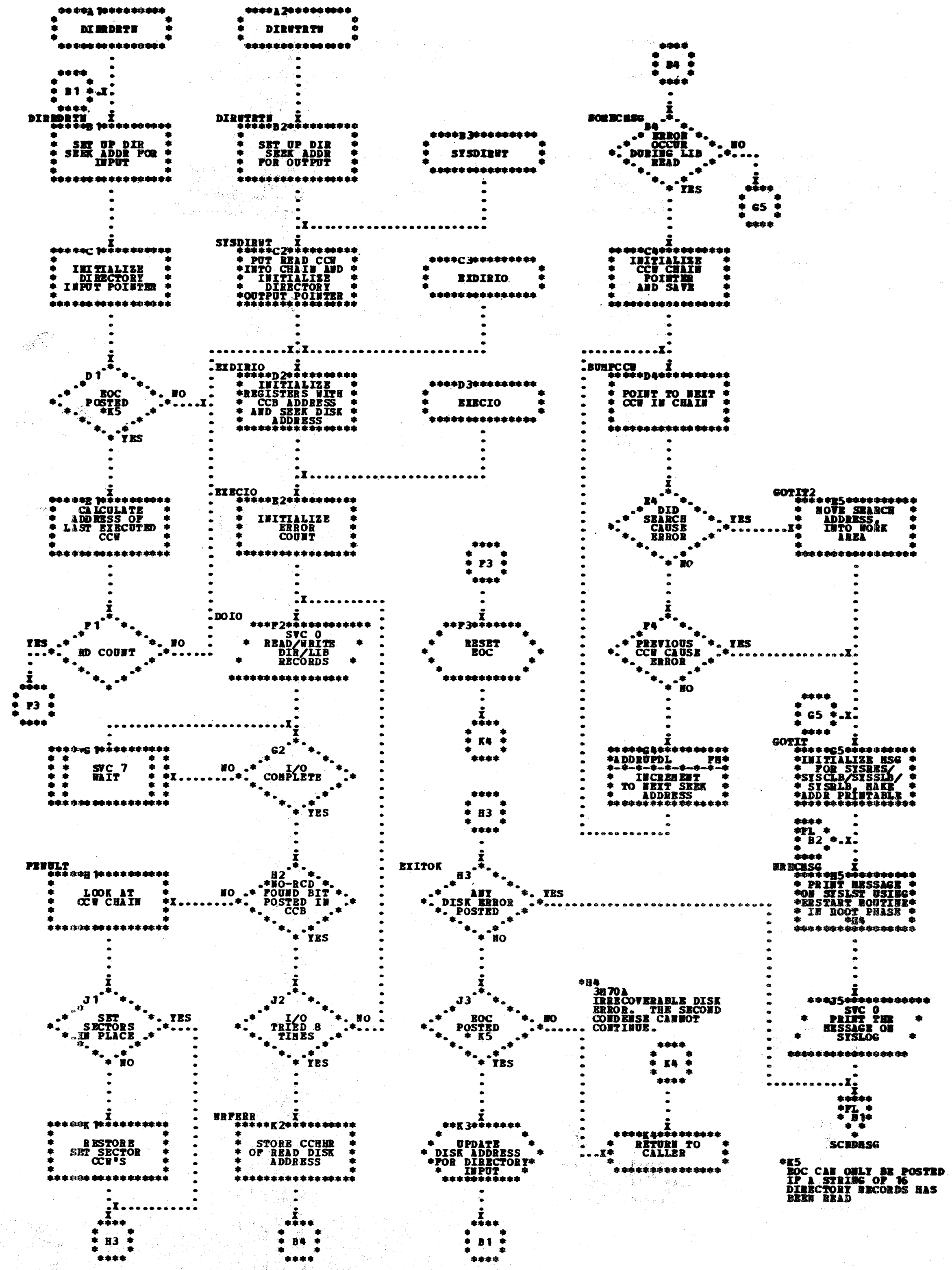
*B2
PRMREG2 CLEARED BEFORE LOAD IS ISSUED

*C3
SINCE PART OF INITIALIZATION ROUTINE WAS OVERLAID, IT IS NECESSARY TO RELOAD THE PHASE.


G1
RPSRTN
RPSRTN
H1
GET DEVICE TYPE FROM INITABLE
J1
RPS SUPPORTED *G4
NO
YES
K1
GET RECORD LENGTH AND RECORD NUMBER
G3
G2
G2
CHANGE SET SECTOR CCW'S TO TIC'S
H2
H2
IN DISASTER MODE
NO
YES
J2
SET BIT
K2
RETURN TO CALLER
G3
G3
SVC 75 SECTOR CALCULATION ROUTINE
H3
STORE 0 VALUE IN BUCKET
J3
SECTOR VALUE INVALID X'FF'
YES
NO
G2
H2

*G4
IF CONDENSING CI LIBRARY RPS WILL BE USED ONLY FOR CONDENSE OF PCIL, SINCE THIS IS THE ONLY CASE THAT MULTIPROGRAMMING IS ALLOWED

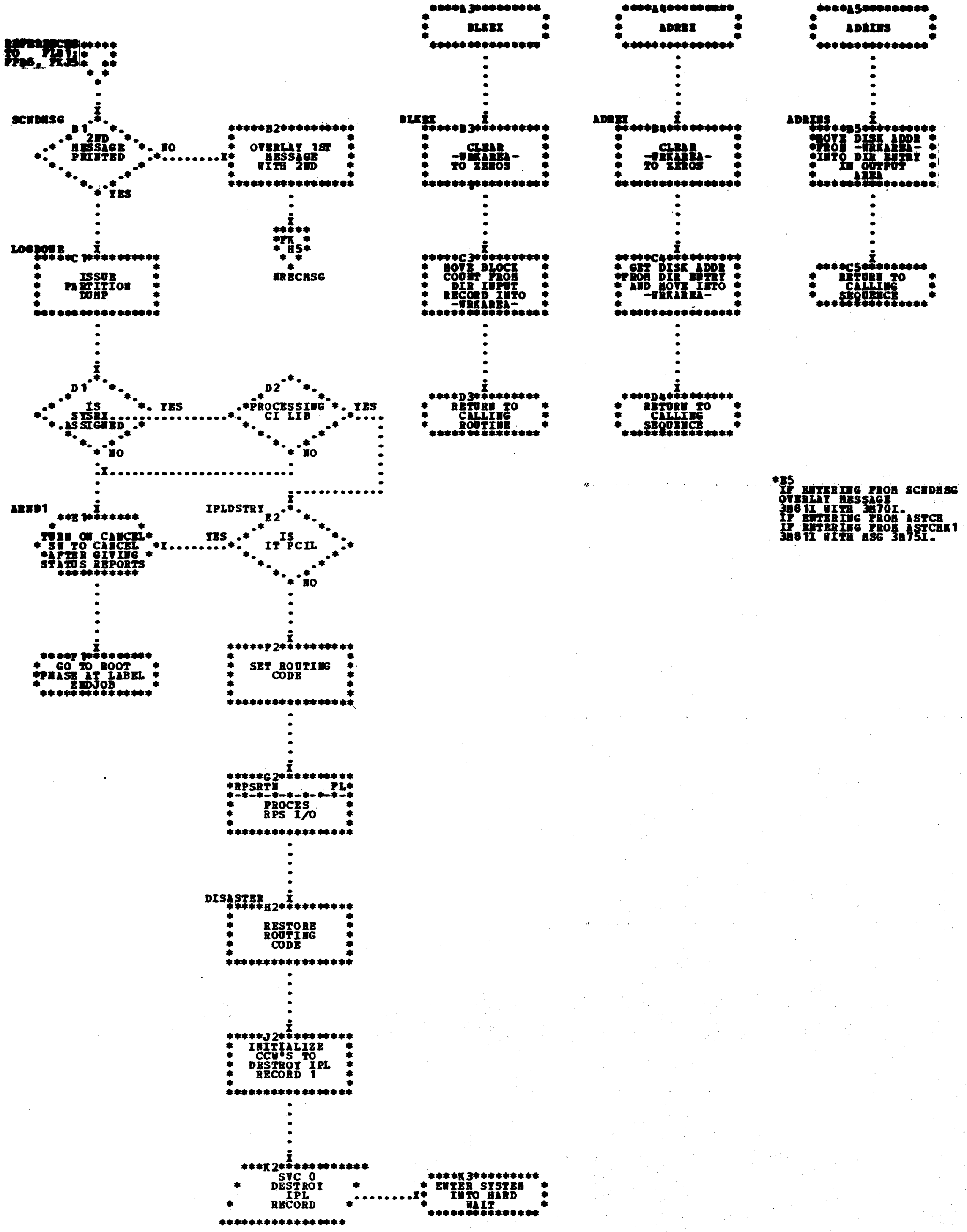
Chart FK. MAINTCN - I/O and I/O Error Subroutines
Refer to Chart 25


A1
DIRRDRTN
A2
DIRWTRTN
B1
DIRRDRTN
B1
SET UP DIR SEEK ADDR FOR INPUT
DIRWTRTN
B2
SET UP DIR SEEK ADDR FOR OUTPUT
B3
SYSDIRWT
B4
NORECMSG
B4
ERROR OCCUR DURING LIB READ
NO
YES
G5
C1
INITIALIZE DIRECTORY INPUT POINTER
SYSDIRWT
C2
PUT READ CCW INTO CHAIN AND INITIALIZE DIRECTORY OUTPUT POINTER
C3
EXDIRIO
C4
INITIALIZE CCW CHAIN POINTER AND SAVE
D1
EOC POSTED *K5
NO
YES
EXDIRIO
D2
INITIALIZE REGISTERS WITH CCB ADDRESS AND SEEK DISK ADDRESS
D3
EXECIO
BUMPCCW
D4
POINT TO NEXT CCW IN CHAIN
E1
CALCULATE ADDRESS OF LAST EXECUTED CCW
EXECIO
E2
INITIALIZE ERROR COUNT
E4
DID SEARCH CAUSE ERROR
YES
NO
GOTIT2
E5
MOVE SEARCH ADDRESS, INTO WORK AREA
F3
F1
RD COUNT
YES
NO
P3
DOIO
F2
SVC 0 READ/WRITE DIR/LIB RECORDS
F3
RESET EOC
K4
F4
PREVIOUS CCW CAUSE ERROR
YES
NO
G5
G1
SVC 7 WAIT
G2
I/O COMPLETE
NO
YES
H3
G4
ADDRUPDL PH
INCREMENT TO NEXT SEEK ADDRESS
GOTIT
G5
INITIALIZE MSG FOR SYSRES/ SYSCLB/SYSSLB/ SYSRLB, MAKE ADDR PRINTABLE
FL B2
PENULT
H1
LOOK AT CCW CHAIN
H2
NO-RCD FOUND BIT POSTED IN CCB
NO
YES
EXITOK
H3
ANY DISK ERROR POSTED
YES
NO
NRECMSG
H5
PRINT MESSAGE ON SYSLST USING ERSTART ROUTINE IN ROOT PHASE *H4
J1
SET SECTORS IN PLACE
YES
NO
J2
I/O TRIED 8 TIMES
NO
YES
J3
EOC POSTED * K5
NO
YES
J5
SVC 0 PRINT THE MESSAGE ON SYSLOG
K4
K1
RESTORE SET SECTOR CCW'S
WRFERR
K2
STORE CCHHR OF READ DISK ADDRESS
K3
UPDATE DISK ADDRESS FOR DIRECTORY INPUT
K4
RETURN TO CALLER
FL B1
SCNDMSG
H3
B4
B1

*H4
3M70A
IRRECOVERABLE DISK ERROR. THE SECOND CONDENSE CANNOT CONTINUE.

*K5
EOC CAN ONLY BE POSTED IF A STRING OF 16 DIRECTORY RECORDS HAS BEEN READ

Chart FL. MAINTCN - Destroy IPL Record 1, Block Count and Disk Address Extraction, and Disk Address Insertion Subroutines
Refer to Chart 25


REFERENCE TO FLB1, FPD5, FKJ5
SCNDMSG
B1
2ND MESSAGE PRINTED
NO
YES
B2
OVERLAY 1ST MESSAGE WITH 2ND
FK H5
MRECMSG
LOGDONE
C1
ISSUE PARTITION DUMP
D1
IS SYSRX ASSIGNED
YES
NO
D2
PROCESSING CI LIB
YES
NO
ARND1
E1
TURN ON CANCEL SW TO CANCEL AFTER GIVING STATUS REPORTS
IPLDSTRY
E2
IS IT PCIL
YES
NO
F1
GO TO ROOT PHASE AT LABEL ENDJOB
F2
SET ROUTING CODE
G2
RPSRTN FL
PROCES RPS I/O
DISASTER
H2
RESTORE ROUTING CODE
J2
INITIALIZE CCW'S TO DESTROY IPL RECORD 1
K2
SVC 0 DESTROY IPL RECORD
K3
ENTER SYSTEM INTO HARD WAIT
A3
BLKEX
BLKEX
B3
CLEAR -WRKAREA- TO ZEROS
C3
MOVE BLOCK COUNT FROM DIR INPUT RECORD INTO -WRKAREA-
D3
RETURN TO CALLING ROUTINE
A4
ADREX
ADREX
B4
CLEAR -WRKAREA- TO ZEROS
C4
GET DISK ADDR FROM DIR ENTRY AND MOVE INTO -WRKAREA-
D4
RETURN TO CALLING SEQUENCE
A5
ADRINS
ADRINS
B5
MOVE DISK ADDR FROM -WRKAREA- INTO DIR ENTRY IN OUTPUT AREA
C5
RETURN TO CALLING SEQUENCE

*E5
IF ENTERING FROM SCNDMSG
OVERLAY MESSAGE
3M81I WITH 3M70I.
IF ENTERING FROM ASTCH
IF ENTERING FROM ASTCHK1
3M81I WITH MSG 3M75I.

Chart FM. MAINTCN - Update Subroutines
Refer to Chart 25

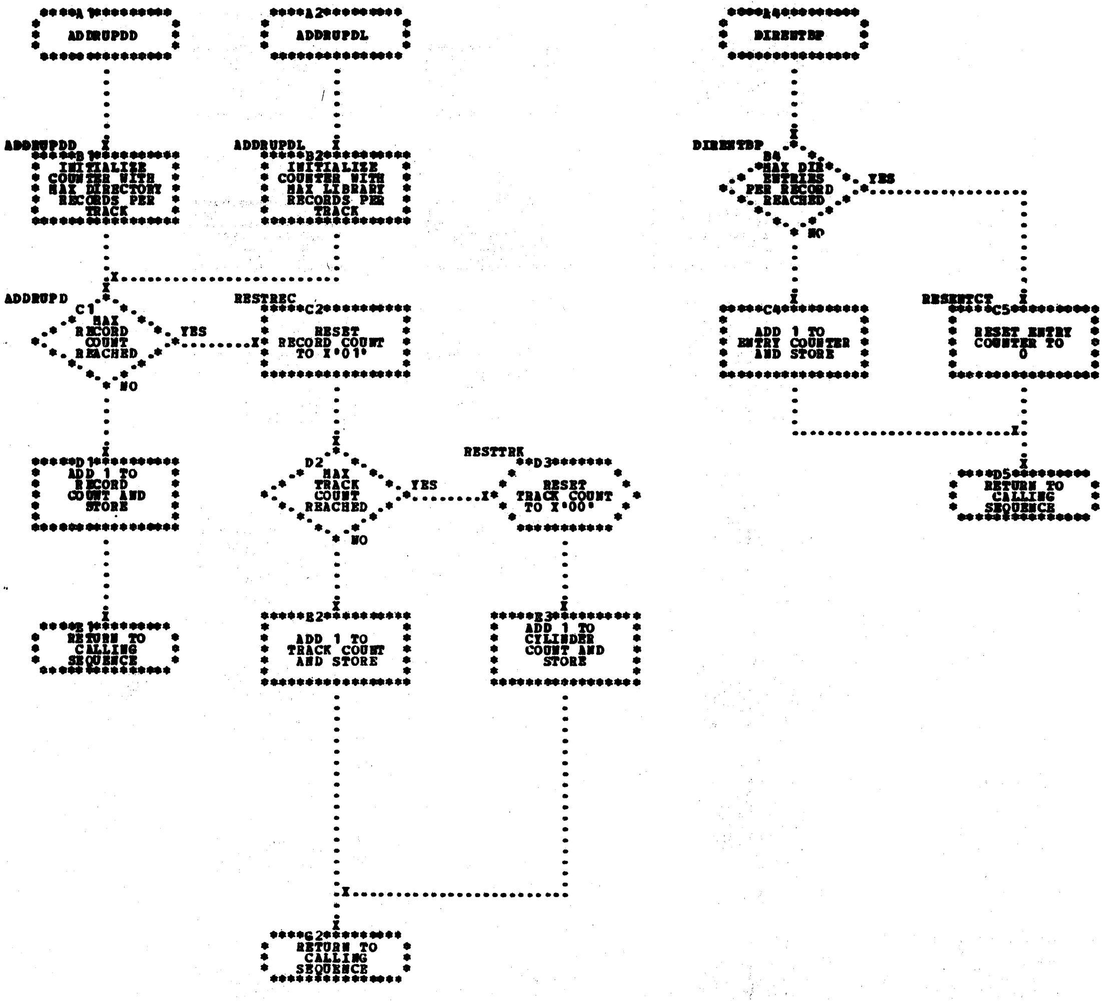

A1
ADDRUPDD
A2
ADDRUPDL
A4
DIRENTDP
ADDRUPDD
B1
INITIALIZE COUNTER WITH MAX DIRECTORY RECORDS PER TRACK
ADDRUPDL
B2
INITIALIZE COUNTER WITH MAX LIBRARY RECORDS PER TRACK
DIRENTDP
B4
MAX DIR ENTRIES PER RECORD REACHED
YES
NO
ADDRUPD
C1
MAX RECORD COUNT REACHED
YES
NO
RESTREC
C2
RESET RECORD COUNT TO X'01'
C4
ADD 1 TO ENTRY COUNTER AND STORE
RESENTCT
C5
RESET ENTRY COUNTER TO 0
D1
ADD 1 TO RECORD COUNT AND STORE
D2
MAX TRACK COUNT REACHED
YES
NO
RESTTRK
D3
RESET TRACK COUNT TO X'00'
D5
RETURN TO CALLING SEQUENCE
E1
RETURN TO CALLING SEQUENCE
E2
ADD 1 TO TRACK COUNT AND STORE
E3
ADD 1 TO CYLINDER COUNT AND STORE
G2
RETURN TO CALLING SEQUENCE

Chart MA. MAINTA - Process Control Card, Store Allocation Information (Part 1 of 2)
Refer to Chart 40

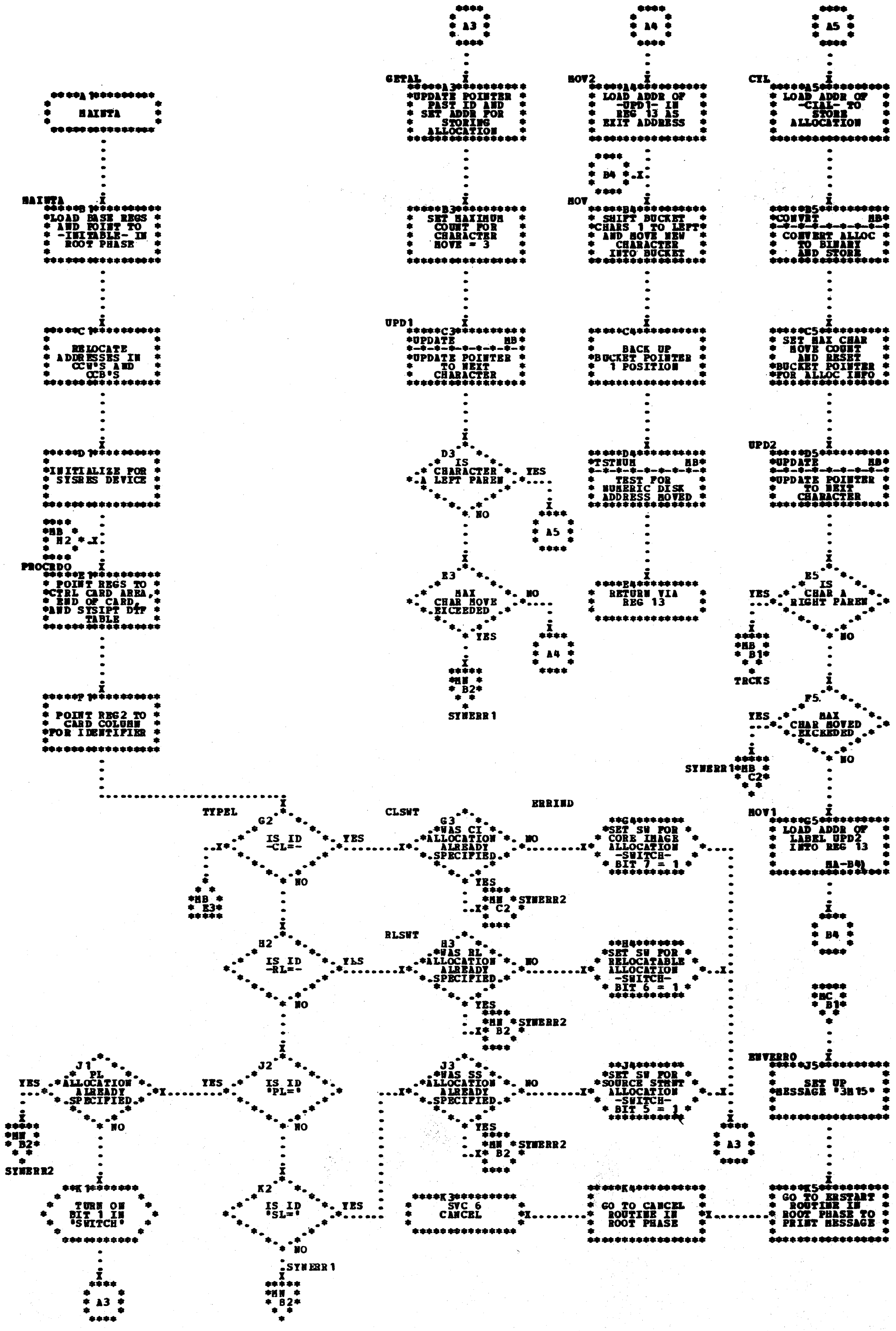

A3
A4
A5
A1 MAINTA
MAINTA B1 LOAD BASE REGS AND POINT TO -INITABLE- IN ROOT PHASE
C1 RELOCATE ADDRESSES IN CCW'S AND CCB'S
D1 INITIALIZE FOR SYSRES DEVICE
MB H2
PROCRDO E1 POINT REGS TO CTRL CARD AREA, END OF CARD, AND SYSIPT DTF TABLE
F1 POINT REG2 TO CARD COLUMN FOR IDENTIFIER
GETAL A3 UPDATE POINTER PAST ID AND SET ADDR FOR STORING ALLOCATION
B3 SET MAXIMUM COUNT FOR CHARACTER MOVE = 3
UPD1 C3 UPDATE MB UPDATE POINTER TO NEXT CHARACTER
D3 IS CHARACTER A LEFT PAREN
YES
NO
A5
E3 MAX CHAR MOVE EXCEEDED
NO
A4
YES
MN B2
SYNERR1
MOV2 A4 LOAD ADDR OF -UPD1- IN REG 13 AS EXIT ADDRESS
B4
MOV B4 SHIFT BUCKET CHARS 1 TO LEFT AND MOVE NEW CHARACTER INTO BUCKET
C4 BACK UP BUCKET POINTER 1 POSITION
D4 TSTNUM MB TEST FOR NUMERIC DISK ADDRESS MOVED
E4 RETURN VIA REG 13
CYL A5 LOAD ADDR OF -CYAL- TO STORE ALLOCATION
B5 CONVRT MB CONVERT ALLOC TO BINARY AND STORE
C5 SET MAX CHAR MOVE COUNT AND RESET BUCKET POINTER FOR ALLOC INFO
UPD2 D5 UPDATE MB UPDATE POINTER TO NEXT CHARACTER
E5 IS CHAR A RIGHT PAREN
YES
MB B1
TRCKS
NO
F5 MAX CHAR MOVED EXCEEDED
YES
SYNERR1 MB C2
NO
TYPEL G2 IS ID -CL=-
YES
NO
MB E3
CLSWT G3 WAS CI ALLOCATION ALREADY SPECIFIED
NO
YES
MN C2 SYNERR2
ERRIND
G4 SET SW FOR CORE IMAGE ALLOCATION -SWITCH- BIT 7 = 1
MOV1 G5 LOAD ADDR OF LABEL UPD2 INTO REG 13 MA-B4
B4
MC B1
H2 IS ID -RL=-
YES
NO
RLSWT H3 WAS RL ALLOCATION ALREADY SPECIFIED
NO
YES
MN B2 SYNERR2
H4 SET SW FOR RELOCATABLE ALLOCATION -SWITCH- BIT 6 = 1
J1 PL ALLOCATION ALREADY SPECIFIED
YES
MN B2
SYNERR2
NO
J2 IS ID 'PL='
YES
NO
J3 WAS SS ALLOCATION ALREADY SPECIFIED
NO
YES
MN B2 SYNERR2
J4 SET SW FOR SOURCE STMT ALLOCATION -SWITCH- BIT 5 = 1
A3
ENVERRO J5 SET UP MESSAGE '3H15'
K1 TURN ON BIT 1 IN 'SWITCH'
A3
K2 IS ID 'SL='
YES
NO
SYNERR1
MN B2
K3 SVC 6 CANCEL
K4 GO TO CANCEL ROUTINE IN ROOT PHASE
K5 GO TO ERSTART ROUTINE IN ROOT PHASE TO PRINT MESSAGE

Chart MB. MAINTA - Store Allocation Information (Part 2 of 2)
Refer to Chart 40

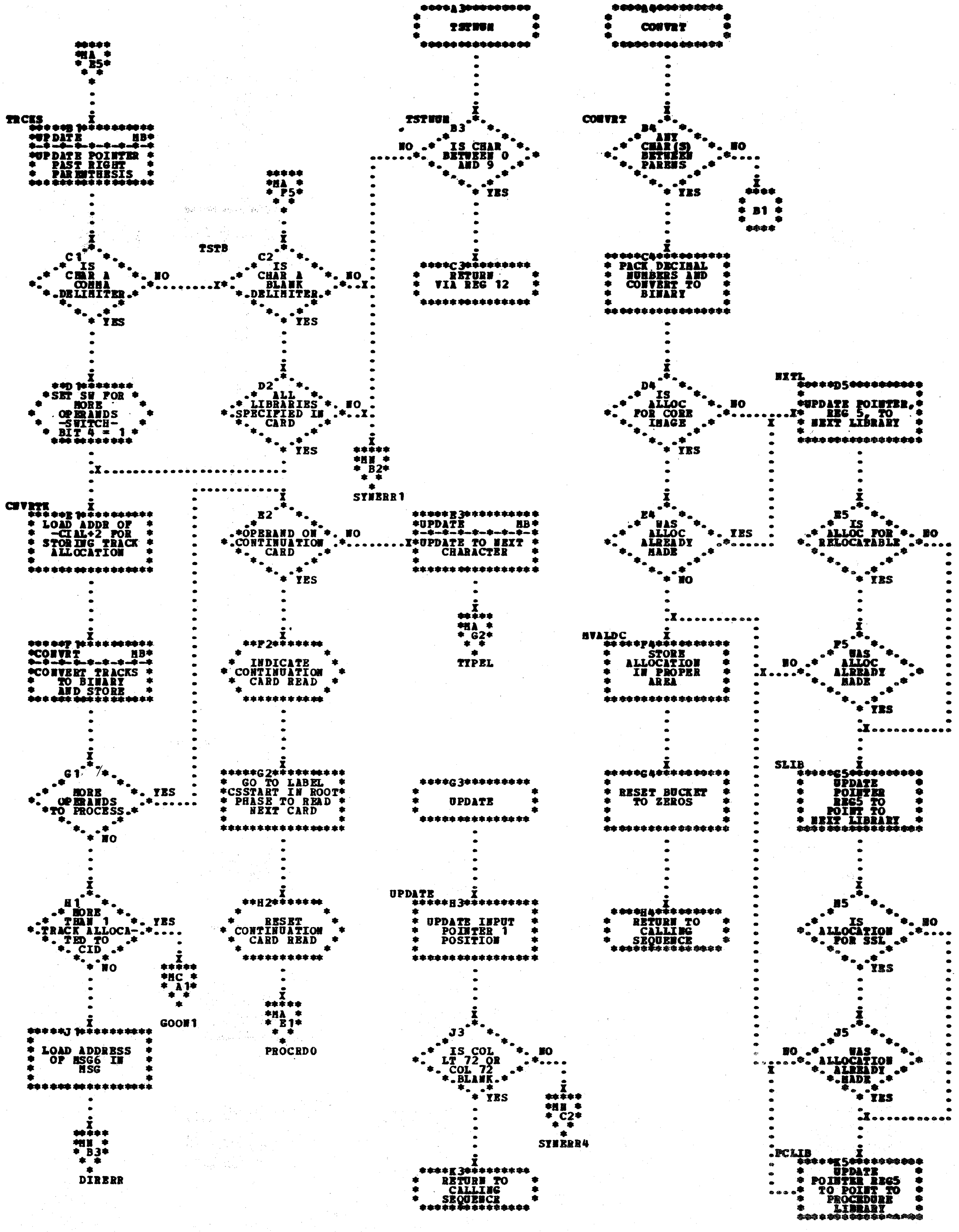
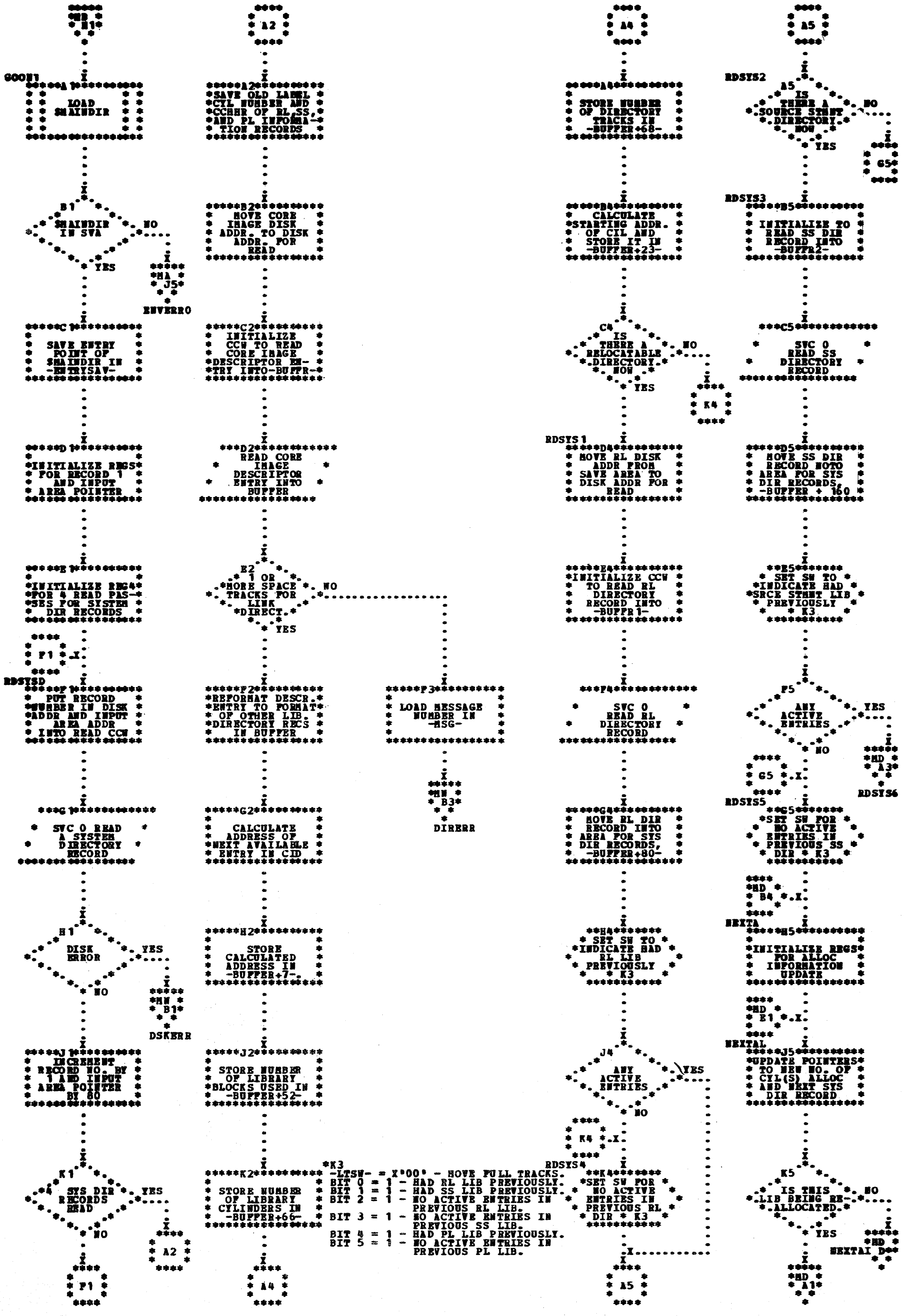
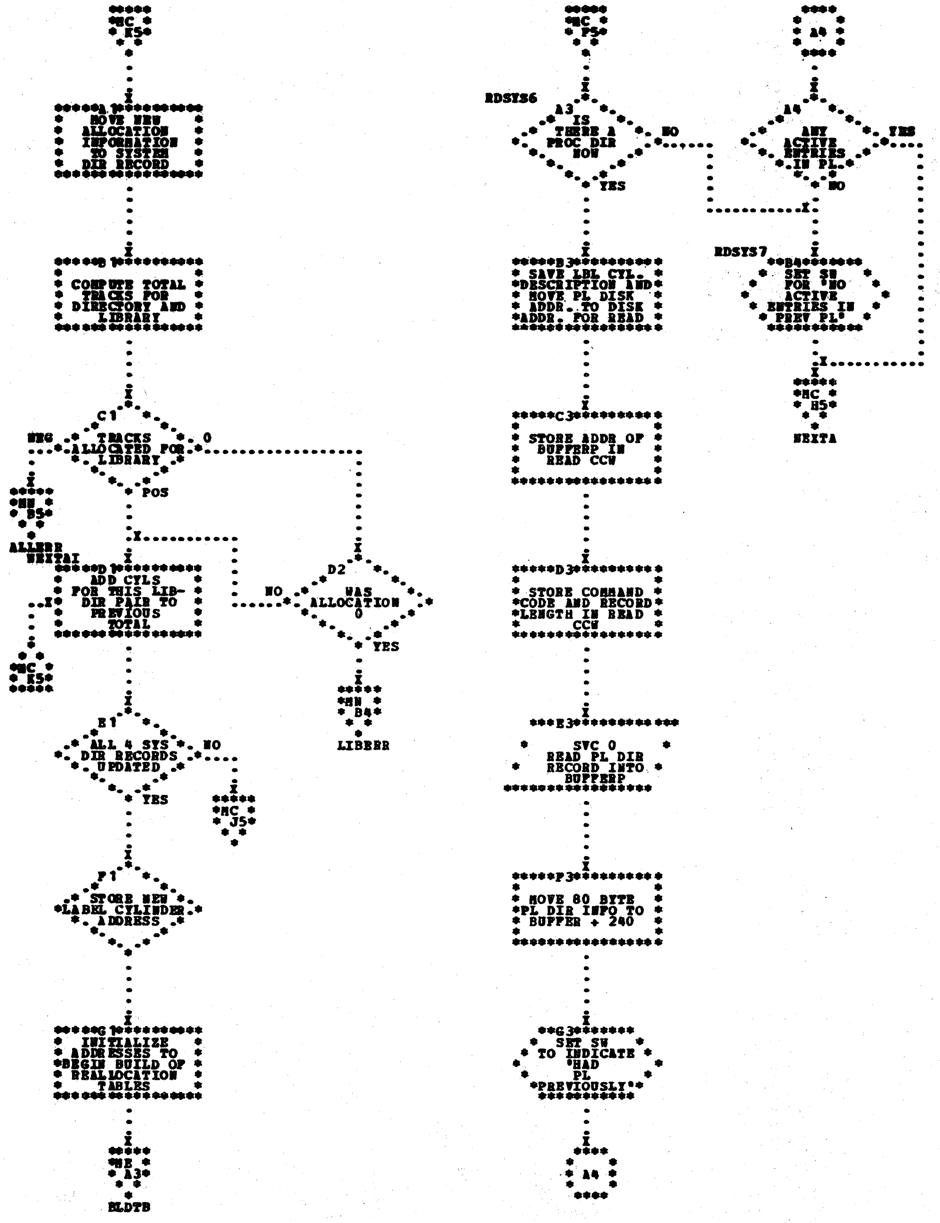

MA B5
TRCKS
B1 UPDATE MB
UPDATE POINTER PAST RIGHT PARENTHESIS
C1 IS CHAR A COMMA DELIMITER
NO
YES
D1 SET SW FOR MORE OPERANDS -SWITCH- BIT 4 = 1
CNVRTK
E1 LOAD ADDR OF -CIAL+2 FOR STORING TRACK ALLOCATION
F1 CONVRT MB
CONVERT TRACKS TO BINARY AND STORE
G1 MORE OPERANDS TO PROCESS
YES
NO
H1 MORE THAN 1 TRACK ALLOCATED TO CID
YES
NO
MC A1
GOON1
J1 LOAD ADDRESS OF MSG6 IN MSG
MN B3
DIRERR
MA F5
TSTB
C2 IS CHAR A BLANK DELIMITER
NO
YES
D2 ALL LIBRARIES SPECIFIED IN CARD
NO
YES
MN B2
SYNERR1
E2 OPERAND ON CONTINUATION CARD
NO
YES
F2 INDICATE CONTINUATION CARD READ
G2 GO TO LABEL CSSTART IN ROOT PHASE TO READ NEXT CARD
H2 RESET CONTINUATION CARD READ
MA E1
PROCRDO
A3 TSTNUM
TSTNUM
B3 IS CHAR BETWEEN 0 AND 9
NO
YES
C3 RETURN VIA REG 12
E3 UPDATE MB
UPDATE TO NEXT CHARACTER
MA G2
TYPEL
G3 UPDATE
UPDATE
H3 UPDATE INPUT POINTER 1 POSITION
J3 IS COL LT 72 OR COL 72 BLANK
NO
YES
MN C2
SYNERR4
K3 RETURN TO CALLING SEQUENCE
A4 CONVRT
CONVRT
B4 ANY CHAR(S) BETWEEN PARENS
NO
YES
B1
C4 PACK DECIMAL NUMBERS AND CONVERT TO BINARY
D4 IS ALLOC FOR CORE IMAGE
NO
YES
E4 WAS ALLOC ALREADY MADE
YES
NO
NVALDC
F4 STORE ALLOCATION IN PROPER AREA
G4 RESET BUCKET TO ZEROS
H4 RETURN TO CALLING SEQUENCE
NXTL
D5 UPDATE POINTER, REG 5, TO NEXT LIBRARY
E5 IS ALLOC FOR RELOCATABLE
NO
YES
F5 WAS ALLOC ALREADY MADE
NO
YES
SLIB
G5 UPDATE POINTER REG5 TO POINT TO NEXT LIBRARY
H5 IS ALLOCATION FOR SSL
NO
YES
J5 WAS ALLOCATION ALREADY MADE
NO
YES
PCLIB
K5 UPDATE POINTER REG5 TO POINT TO PROCEDURE LIBRARY

Chart MC. MAINTA - Read System Directory Records and Update ALLOC (Part 1 of 2)
Refer to Chart 40


MB
H1
A2
A4
A5
GOONT
A1
LOAD $MAINDIR
A2
SAVE OLD LABEL CYL NUMBER AND CCHHR OF RL,SS, AND PL INFORMA-TION RECORDS
A4
STORE NUMBER OF DIRECTORY TRACKS IN -BUFFER+68-
RDSYS2
A5
IS THERE A SOURCE STMNT DIRECTORY NOW
NO
YES
G5
B1
$MAINDIR IN SVA
NO
YES
MA
J5
ENVERRO
B2
MOVE CORE IMAGE DISK ADDR. TO DISK ADDR. FOR READ
B4
CALCULATE STARTING ADDR. OF CIL AND STORE IT IN -BUFFER+23-
RDSYS3
B5
INITIALIZE TO READ SS DIR RECORD INTO -BUFFR2-
C1
SAVE ENTRY POINT OF $MAINDIR IN -ENTRYSAV-
C2
INITIALIZE CCW TO READ CORE IMAGE DESCRIPTOR EN-TRY INTO-BUFFR-
C4
IS THERE A RELOCATABLE DIRECTORY NOW
NO
YES
K4
C5
SVC 0 READ SS DIRECTORY RECORD
D1
INITIALIZE REGS FOR RECORD 1 AND INPUT AREA POINTER
D2
READ CORE IMAGE DESCRIPTOR ENTRY INTO BUFFER
RDSYS1
D4
MOVE RL DISK ADDR FROM SAVE AREA TO DISK ADDR FOR READ
D5
MOVE SS DIR RECORD NOTO AREA FOR SYS DIR RECORDS, -BUFFER + 160
E1
INITIALIZE REGS FOR 4 READ PAS-SES FOR SYSTEM DIR RECORDS
E2
1 OR MORE SPACE TRACKS FOR LINK DIRECT.
NO
YES
E4
INITIALIZE CCW TO READ RL DIRECTORY RECORD INTO -BUFFR1-
E5
SET SW TO INDICATE HAD SRCE STMNT LIB PREVIOUSLY * K3
F1
RDSYSD
F1
PUT RECORD NUMBER IN DISK ADDR AND INPUT AREA ADDR INTO READ CCW
F2
REFORMAT DESCR. ENTRY TO FORMAT OF OTHER LIB. DIRECTORY RECS IN BUFFER
F3
LOAD MESSAGE NUMBER IN -MSG-
MN
B3
DIRERR
F4
SVC 0 READ RL DIRECTORY RECORD
F5
ANY ACTIVE ENTRIES
YES
NO
MD
A3
RDSYS6
G5
RDSYS5
G1
SVC 0 READ A SYSTEM DIRECTORY RECORD
G2
CALCULATE ADDRESS OF NEXT AVAILABLE ENTRY IN CID
G4
MOVE RL DIR RECORD INTO AREA FOR SYS DIR RECORDS, -BUFFER+80-
G5
SET SW FOR NO ACTIVE ENTRIES IN PREVIOUS SS DIR * K3
MD
B4
NEXTA
H1
DISK ERROR
YES
NO
MN
B1
DSKERR
H2
STORE CALCULATED ADDRESS IN -BUFFER+7-.
H4
SET SW TO INDICATE HAD RL LIB PREVIOUSLY * K3
H5
INITIALIZE REGS FOR ALLOC INFORMATION UPDATE
MD
E1
NEXTAL
J1
INCREMENT RECORD NO. BY 1 AND INPUT AREA POINTER BY 80
J2
STORE NUMBER OF LIBRARY BLOCKS USED IN -BUFFER+52-
J4
ANY ACTIVE ENTRIES
YES
NO
J5
UPDATE POINTERS TO NEW NO. OF CYL(S) ALLOC AND NEXT SYS DIR RECORD
K4
K1
SYS DIR RECORDS READ
YES
NO
A2
K2
STORE NUMBER OF LIBRARY CYLINDERS IN -BUFFER+66-
RDSYS4
K4
SET SW FOR NO ACTIVE ENTRIES IN PREVIOUS RL DIR * K3
K5
IS THIS LIB BEING RE-ALLOCATED.
NO
YES
MD
NEXTAI D
P1
A4
A5
MD
A1

```
*K3
-LTSW- = X'00' - MOVE FULL TRACKS.
BIT 0 = 1 - HAD RL LIB PREVIOUSLY.
BIT 1 = 1 - HAD SS LIB PREVIOUSLY.
BIT 2 = 1 - NO ACTIVE ENTRIES IN
            PREVIOUS RL LIB.
BIT 3 = 1 - NO ACTIVE ENTRIES IN
            PREVIOUS SS LIB.
BIT 4 = 1 - HAD PL LIB PREVIOUSLY.
BIT 5 = 1 - NO ACTIVE ENTRIES IN
            PREVIOUS PL LIB.
```

Chart MD. MAINTA - Read System Directory Records and Update ALLOC (Part 2 of 2)
Refer to Chart 40



MC K5
A1
MOVE NEW ALLOCATION INFORMATION TO SYSTEM DIR RECORD
B1
COMPUTE TOTAL TRACKS FOR DIRECTORY AND LIBRARY
C1
TRACKS ALLOCATED FOR LIBRARY
NEG
0
POS
MN B5
ALLERR
NEXTA1
D1
ADD CYLS FOR THIS LIB-DIR PAIR TO PREVIOUS TOTAL
MC K5
D2
WAS ALLOCATION 0
NO
YES
MN B4
LIBERR
E1
ALL 4 SYS DIR RECORDS UPDATED
NO
YES
MC J5
F1
STORE NEW LABEL CYLINDER ADDRESS
G1
INITIALIZE ADDRESSES TO BEGIN BUILD OF REALLOCATION TABLES
ME A3
BLDTB
MC F5
RDSYS6
A3
IS THERE A PROC DIR NOW
NO
YES
B3
SAVE LBL CYL. DESCRIPTION AND MOVE PL DISK ADDR. TO DISK ADDR. FOR READ
C3
STORE ADDR OF BUFFERP IN READ CCW
D3
STORE COMMAND CODE AND RECORD LENGTH IN READ CCW
E3
SVC 0 READ PL DIR RECORD INTO BUFFERP
F3
MOVE 80 BYTE PL DIR INFO TO BUFFER + 240
G3
SET SW TO INDICATE 'HAD PL PREVIOUSLY'
A4
A4
ANY ACTIVE ENTRIES IN PL
YES
NO
RDSYS7
B4
SET SW FOR 'NO ACTIVE ENTRIES IN PREV PL'
MC H5
NEXTA

Chart ME. MAINTA - Build Directory and Library Reallocation Tables
Refer to Chart 40

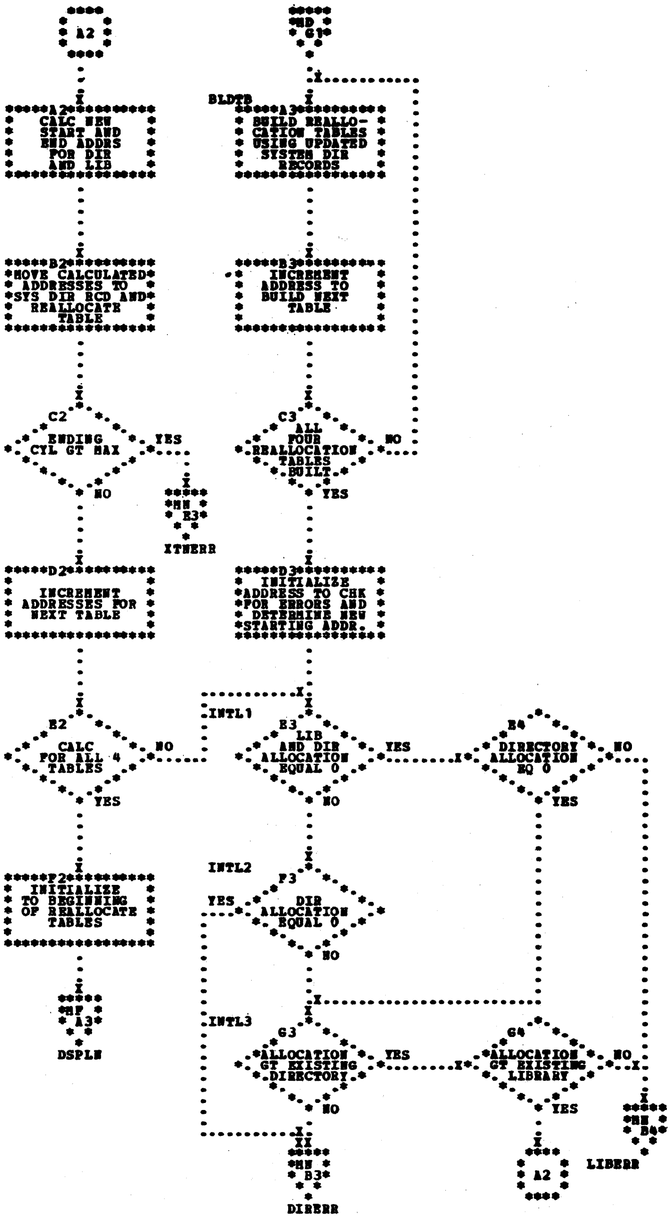
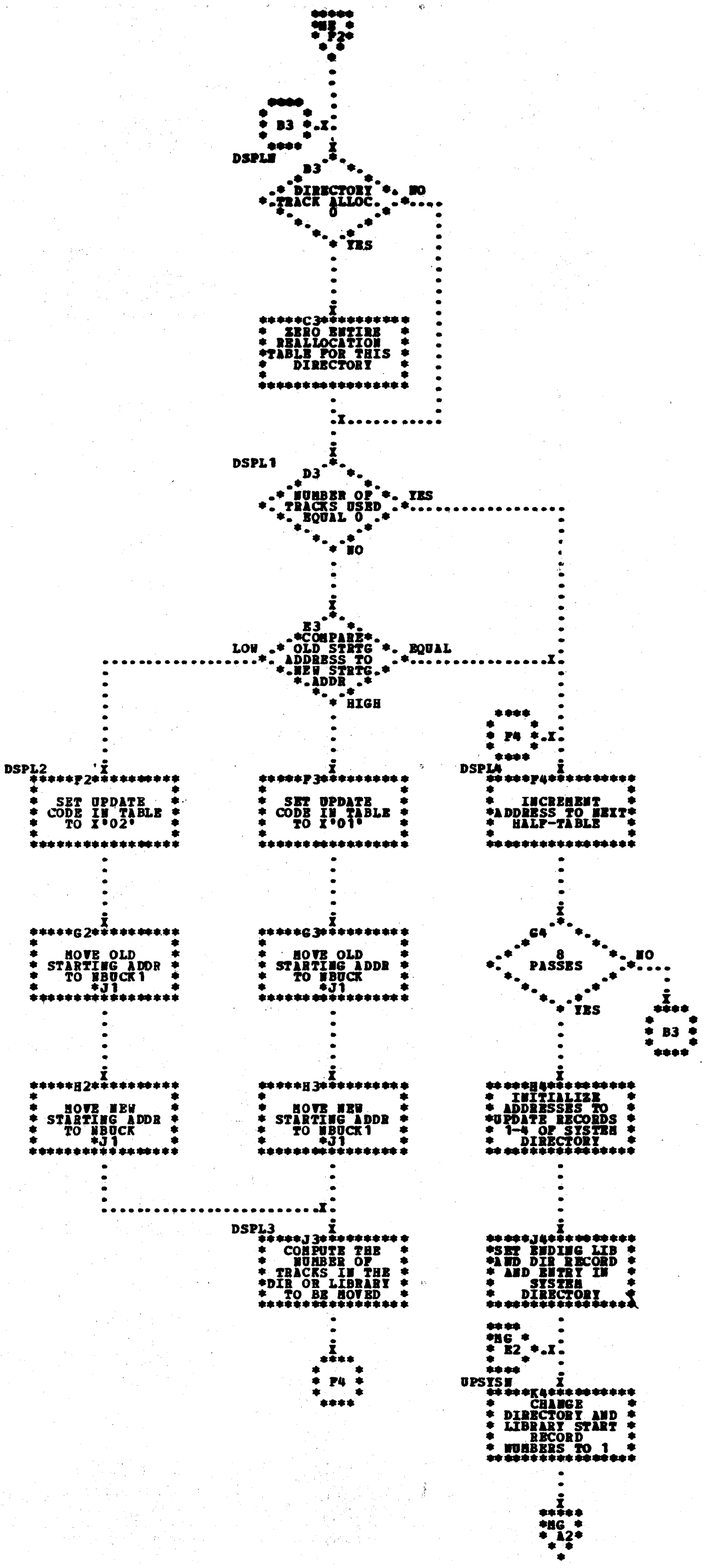

A2
MD G1
A2
CALC NEW START AND END ADDRS FOR DIR AND LIB
BLDTB
A3
BUILD REALLO-CATION TABLES USING UPDATED SYSTEM DIR RECORDS
B2
MOVE CALCULATED ADDRESSES TO SYS DIR RCD AND REALLOCATE TABLE
B3
INCREMENT ADDRESS TO BUILD NEXT TABLE
C2
ENDING CYL GT MAX
YES
NO
MN B3
XTNERR
C3
ALL FOUR REALLOCATION TABLES BUILT
NO
YES
D2
INCREMENT ADDRESSES FOR NEXT TABLE
D3
INITIALIZE ADDRESS TO CHK FOR ERRORS AND DETERMINE NEW STARTING ADDR.
INTL1
E2
CALC FOR ALL 4 TABLES
NO
YES
E3
LIB AND DIR ALLOCATION EQUAL 0
YES
NO
E4
DIRECTORY ALLOCATION EQ 0
NO
YES
INTL2
F2
INITIALIZE TO BEGINNING OF REALLOCATE TABLES
F3
DIR ALLOCATION EQUAL 0
YES
NO
MF A3
DSPLN
INTL3
G3
ALLOCATION GT EXISTING DIRECTORY
YES
NO
G4
ALLOCATION GT EXISTING LIBRARY
NO
YES
MN B4
LIBERR
MN B3
DIRERR
A2

Chart MF. MAINTA - Compute Displacement for Directory and Library Movement
Refer to Chart 40


ME
F2
B3
DSPLN
B3
DIRECTORY TRACK ALLOC. 0
NO
YES
C3
ZERO ENTIRE REALLOCATION TABLE FOR THIS DIRECTORY
DSPL1
D3
NUMBER OF TRACKS USED EQUAL 0
YES
NO
E3
COMPARE OLD STRTG ADDRESS TO NEW STRTG ADDR
LOW
EQUAL
HIGH
F4
DSPL2
F2
SET UPDATE CODE IN TABLE TO X'02'
F3
SET UPDATE CODE IN TABLE TO X'01'
DSPL4
F4
INCREMENT ADDRESS TO NEXT HALF-TABLE
G2
MOVE OLD STARTING ADDR TO NBUCK1 *J1
G3
MOVE OLD STARTING ADDR TO NBUCK *J1
G4
8 PASSES
NO
YES
B3
H2
MOVE NEW STARTING ADDR TO NBUCK *J1
H3
MOVE NEW STARTING ADDR TO NBUCK1 *J1
H4
INITIALIZE ADDRESSES TO UPDATE RECORDS 1-4 OF SYSTEM DIRECTORY
DSPL3
J3
COMPUTE THE NUMBER OF TRACKS IN THE DIR OR LIBRARY TO BE MOVED
J4
SET ENDING LIB AND DIR RECORD AND ENTRY IN SYSTEM DIRECTORY
F4
MG
E2
UPSYSN
K4
CHANGE DIRECTORY AND LIBRARY START RECORD NUMBERS TO 1
MG
A2

*J1
ADDRESSES IN NBUCK
AND NBUCK1 DETERMINE
THE DIRECTION IN WHICH
THE DIRECTORY OR LIBRARY
IS TO BE MOVED.

Chart MG. MAINTA - Update System Directory Records 1, 2, and 3
Refer to Chart 40

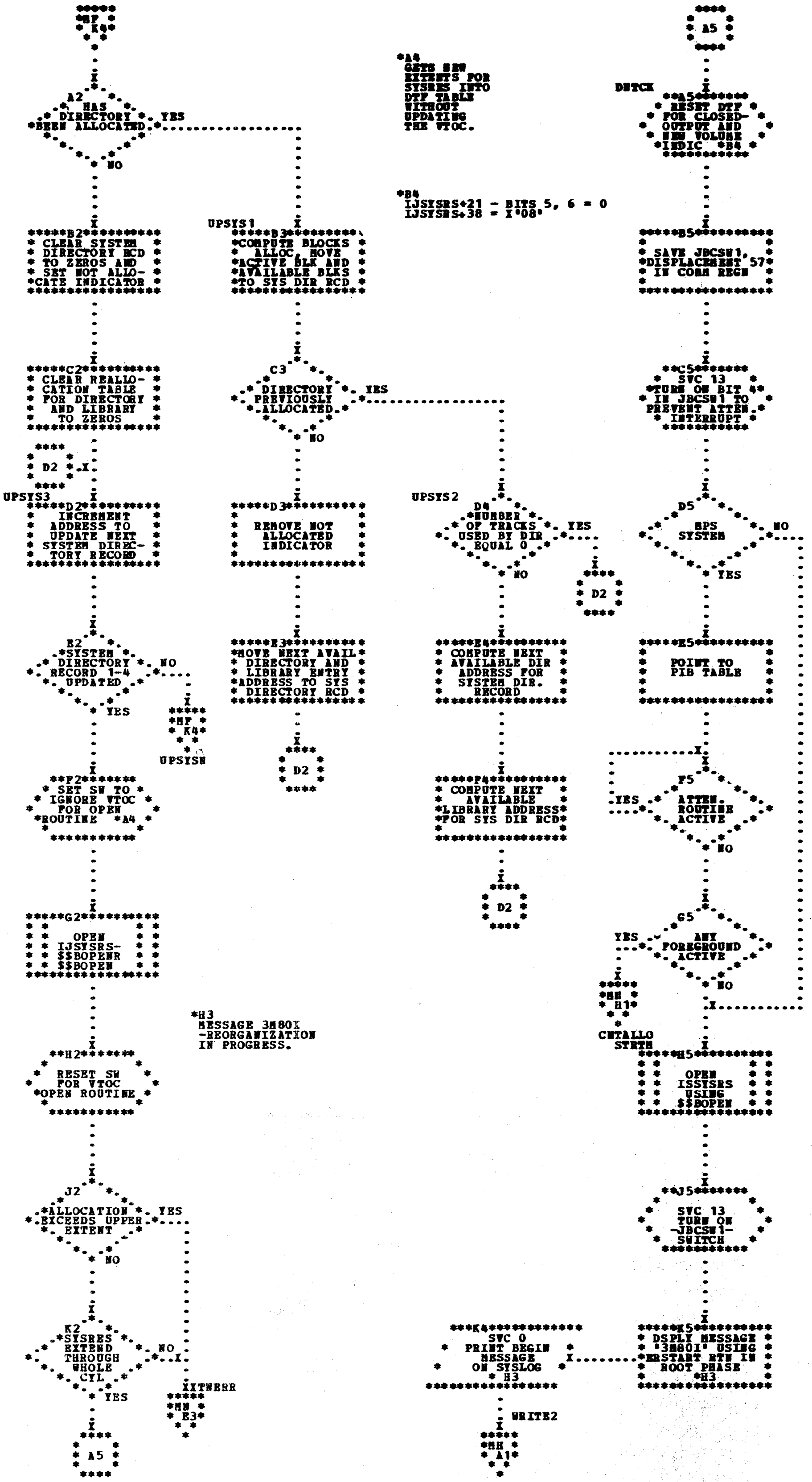

MF K4
A2 HAS DIRECTORY BEEN ALLOCATED
YES
NO
B2 CLEAR SYSTEM DIRECTORY RCD TO ZEROS AND SET NOT ALLOCATE INDICATOR
UPSYS1
B3 COMPUTE BLOCKS ALLOC, MOVE ACTIVE BLK AND AVAILABLE BLKS TO SYS DIR RCD
C2 CLEAR REALLOCATION TABLE FOR DIRECTORY AND LIBRARY TO ZEROS
C3 DIRECTORY PREVIOUSLY ALLOCATED
YES
NO
D2
UPSYS3
D2 INCREMENT ADDRESS TO UPDATE NEXT SYSTEM DIRECTORY RECORD
D3 REMOVE NOT ALLOCATED INDICATOR
E2 SYSTEM DIRECTORY RECORD 1-4 UPDATED
NO
YES
MF K4
UPSYSN
E3 MOVE NEXT AVAIL DIRECTORY AND LIBRARY ENTRY ADDRESS TO SYS DIRECTORY RCD
D2
F2 SET SW TO IGNORE VTOC FOR OPEN ROUTINE *A4
G2 OPEN IJSYSRS- $$BOPENR $$BOPEN
*H3 MESSAGE 3M80I -REORGANIZATION IN PROGRESS.
H2 RESET SW FOR VTOC OPEN ROUTINE
J2 ALLOCATION EXCEEDS UPPER EXTENT
YES
NO
K2 SYSRES EXTEND THROUGH WHOLE CYL
NO
YES
XXTNERR
MN E3
A5
*A4 GETS NEW EXTENTS FOR SYSRES INTO DTF TABLE WITHOUT UPDATING THE VTOC.
*B4 IJSYSRS+21 - BITS 5, 6 = 0
IJSYSRS+38 = X'08'
UPSYS2
D4 NUMBER OF TRACKS USED BY DIR EQUAL 0
YES
NO
D2
E4 COMPUTE NEXT AVAILABLE DIR ADDRESS FOR SYSTEM DIR. RECORD
F4 COMPUTE NEXT AVAILABLE LIBRARY ADDRESS FOR SYS DIR RCD
D2
K4 SVC 0 PRINT BEGIN MESSAGE ON SYSLOG * H3
WRITE2
MH A1
A5
DNTCK
A5 RESET DTF FOR CLOSED-OUTPUT AND NEW VOLUME INDIC *B4
B5 SAVE JBCSW1, DISPLACEMENT 57 IN COMM REGN
C5 SVC 13 TURN ON BIT 4 IN JBCSW1 TO PREVENT ATTEN. INTERRUPT
D5 BPS SYSTEM
NO
YES
E5 POINT TO PIB TABLE
F5 ATTEN. ROUTINE ACTIVE
YES
NO
G5 ANY FOREGROUND ACTIVE
YES
NO
MH H1
CNTALLO STRTN
H5 OPEN ISSYSRS USING $$BOPEN
J5 SVC 13 TURN ON -JBCSW1- SWITCH
K5 DSPLY MESSAGE '3M80I' USING ERSTART RTN IN ROOT PHASE *H3

Chart MH. MAINTA - Write Directory Records, Blank Cylinder 0, Tracks 5-9
Refer to Chart 40

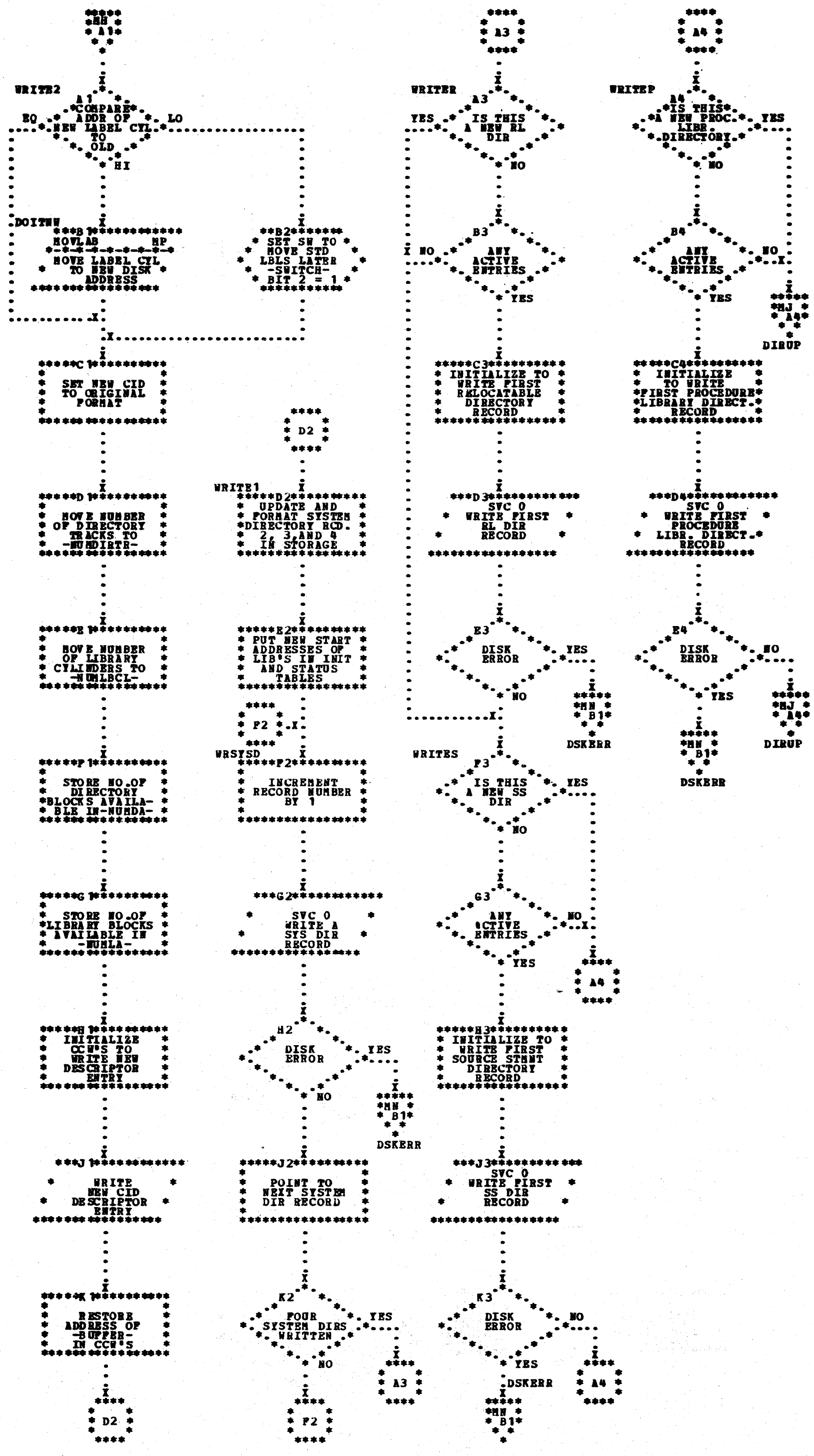
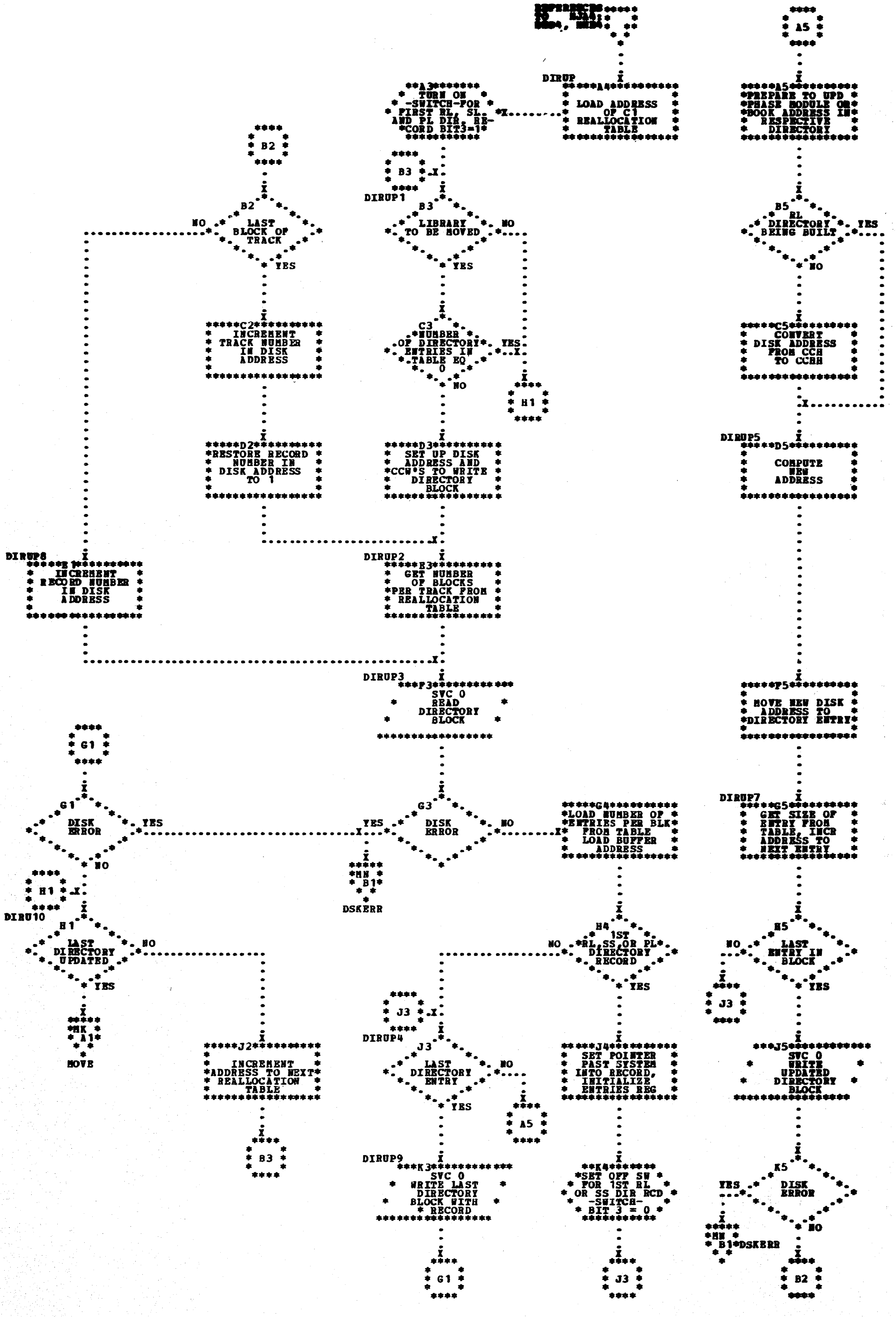

MH A1
WRITE2
A1 COMPARE ADDR OF NEW LABEL CYL. TO OLD
EQ
LO
HI
DOITNW
B1 MOVLAB MP
MOVE LABEL CYL TO NEW DISK ADDRESS
B2 SET SW TO MOVE STD LBLS LATER -SWITCH- BIT 2 = 1
C1 SET NEW CID TO ORIGINAL FORMAT
D1 MOVE NUMBER OF DIRECTORY TRACKS TO -NUMDIRTR-
E1 MOVE NUMBER OF LIBRARY CYLINDERS TO -NUMLBCL-
F1 STORE NO.OF DIRECTORY BLOCKS AVAILA- BLE IN-NUMDA-
G1 STORE NO.OF LIBRARY BLOCKS AVAILABLE IN -NUMLA-
H1 INITIALIZE CCW'S TO WRITE NEW DESCRIPTOR ENTRY
J1 WRITE NEW CID DESCRIPTOR ENTRY
K1 RESTORE ADDRESS OF -BUFFER- IN CCW'S
D2
D2
WRITE1
D2 UPDATE AND FORMAT SYSTEM DIRECTORY RCD. 2, 3,AND 4 IN STORAGE
E2 PUT NEW START ADDRESSES OF LIB'S IN INIT AND STATUS TABLES
F2
WRSYSD
F2 INCREMENT RECORD NUMBER BY 1
G2 SVC 0 WRITE A SYS DIR RECORD
H2 DISK ERROR
YES
NO
MN B1
DSKERR
J2 POINT TO NEXT SYSTEM DIR RECORD
K2 FOUR SYSTEM DIRS WRITTEN
YES
NO
A3
F2
A3
WRITER
A3 IS THIS A NEW RL DIR
YES
NO
B3 ANY ACTIVE ENTRIES
NO
YES
C3 INITIALIZE TO WRITE FIRST RELOCATABLE DIRECTORY RECORD
D3 SVC 0 WRITE FIRST RL DIR RECORD
E3 DISK ERROR
YES
NO
MN B1
DSKERR
WRITES
F3 IS THIS A NEW SS DIR
YES
NO
G3 ANY ACTIVE ENTRIES
NO
YES
A4
H3 INITIALIZE TO WRITE FIRST SOURCE STMNT DIRECTORY RECORD
J3 SVC 0 WRITE FIRST SS DIR RECORD
K3 DISK ERROR
NO
YES
DSKERR
MN B1
A4
A4
WRITEP
A4 IS THIS A NEW PROC. LIBR. DIRECTORY
YES
NO
B4 ANY ACTIVE ENTRIES
NO
YES
MJ A4
DIRUP
C4 INITIALIZE TO WRITE FIRST PROCEDURE LIBRARY DIRECT. RECORD
D4 SVC 0 WRITE FIRST PROCEDURE LIBR. DIRECT. RECORD
E4 DISK ERROR
NO
YES
MJ A4
DIRUP
MN B1
DSKERR

Chart MJ. MAINTA - Update Library Directories
Refer to Chart 40



DIRUP
A3 TURN ON -SWITCH-FOR FIRST RL, SL, AND PL DIR, RECORD BIT3=1
A4 LOAD ADDRESS OF C1 REALLOCATION TABLE
A5
A5 PREPARE TO UPD PHASE MODULE OR BOOK ADDRESS IN RESPECTIVE DIRECTORY
B2
B2 LAST BLOCK OF TRACK
NO
YES
B3
DIRUP1
B3 LIBRARY TO BE MOVED
B5 RL DIRECTORY BEING BUILT
C2 INCREMENT TRACK NUMBER IN DISK ADDRESS
C3 NUMBER OF DIRECTORY ENTRIES IN TABLE EQ 0
H1
C5 CONVERT DISK ADDRESS FROM CCH TO CCHH
D2 RESTORE RECORD NUMBER IN DISK ADDRESS TO 1
D3 SET UP DISK ADDRESS AND CCW'S TO WRITE DIRECTORY BLOCK
DIRUP5
D5 COMPUTE NEW ADDRESS
DIRUP8
E1 INCREMENT RECORD NUMBER IN DISK ADDRESS
DIRUP2
E3 GET NUMBER OF BLOCKS PER TRACK FROM REALLOCATION TABLE
DIRUP3
F3 SVC 0 READ DIRECTORY BLOCK
F5 MOVE NEW DISK ADDRESS TO DIRECTORY ENTRY
G1
G1 DISK ERROR
G3 DISK ERROR
MN B1
DSKERR
G4 LOAD NUMBER OF ENTRIES PER BLK FROM TABLE LOAD BUFFER ADDRESS
DIRUP7
G5 GET SIZE OF ENTRY FROM TABLE, INCR ADDRESS TO NEXT ENTRY
H1
DIRU10
H1 LAST DIRECTORY UPDATED
H4 1ST RL,SS,OR PL DIRECTORY RECORD
H5 LAST ENTRY IN BLOCK
J3
MK A1
MOVE
J2 INCREMENT ADDRESS TO NEXT REALLOCATION TABLE
J3
DIRUP4
J3 LAST DIRECTORY ENTRY
J4 SET POINTER PAST SYSTEM INTO RECORD, INITIALIZE ENTRIES REG
J5 SVC 0 WRITE UPDATED DIRECTORY BLOCK
B3
A5
DIRUP9
K3 SVC 0 WRITE LAST DIRECTORY BLOCK WITH * RECORD
K4 SET OFF SW FOR 1ST RL OR SS DIR RCD -SWITCH- BIT 3 = 0
K5 DISK ERROR
MN B1 DSKERR
G1
J3
B2

Chart MK. MAINTA - Relocate Libraries and Directories
Refer to Chart 40

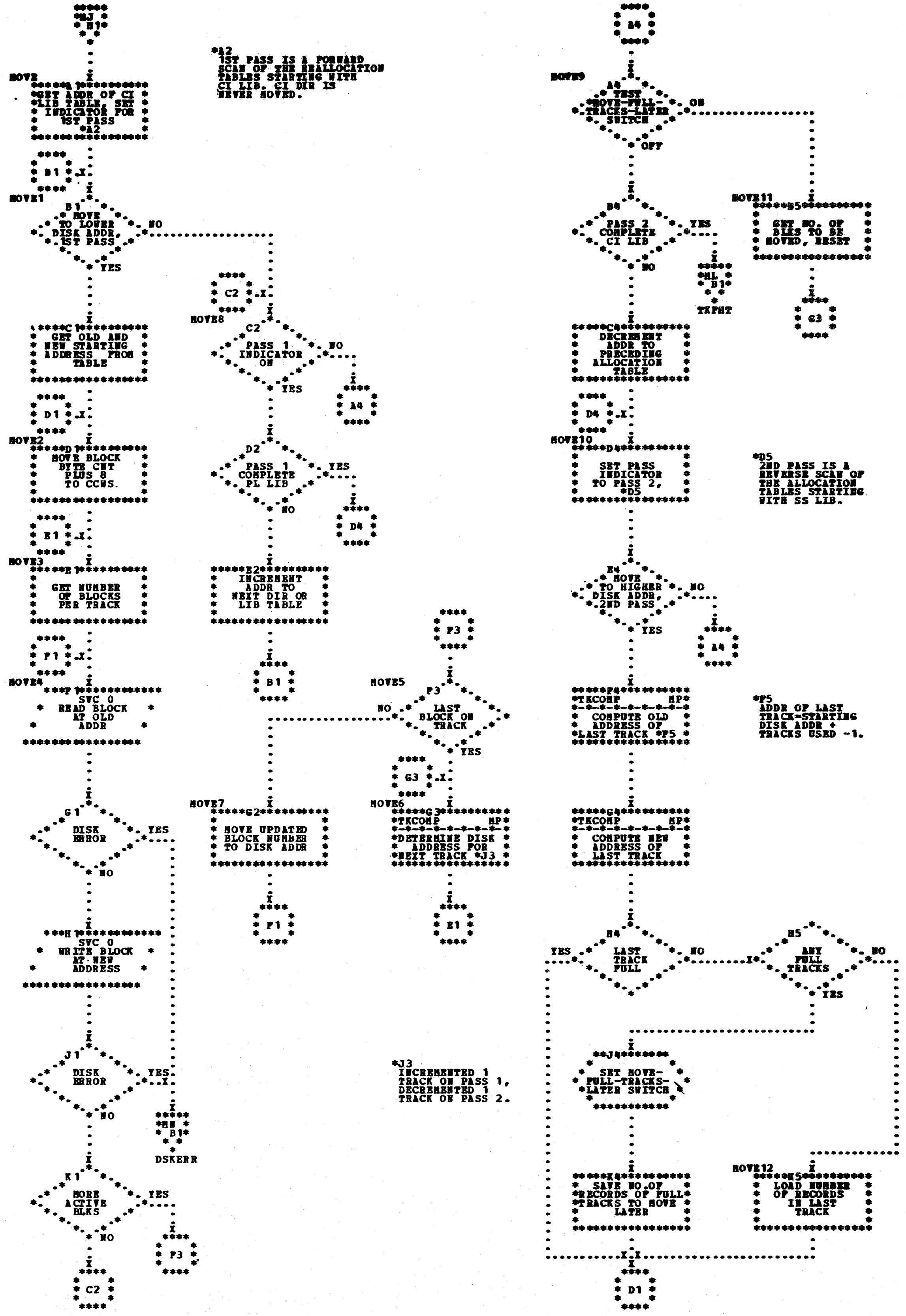

MJ B1
MOVE
A1
GET ADDR OF CI LIB TABLE, SET INDICATOR FOR 1ST PASS *A2
*A2 1ST PASS IS A FORWARD SCAN OF THE REALLOCATION TABLES STARTING WITH CI LIB. CI DIR IS NEVER MOVED.
B1
MOVE1
B1
MOVE TO LOWER DISK ADDR, 1ST PASS
NO
YES
C2
MOVE8
C1
GET OLD AND NEW STARTING ADDRESS FROM TABLE
C2
PASS 1 INDICATOR ON
NO
YES
A4
D1
MOVE2
D1
MOVE BLOCK BYTE CNT PLUS 8 TO CCWS.
D2
PASS 1 COMPLETE PL LIB
YES
NO
D4
E1
MOVE3
E1
GET NUMBER OF BLOCKS PER TRACK
E2
INCREMENT ADDR TO NEXT DIR OR LIB TABLE
F1
MOVE4
F1
SVC 0 READ BLOCK AT OLD ADDR
B1
F3
MOVE5
F3
LAST BLOCK ON TRACK
NO
YES
G3
G1
DISK ERROR
YES
NO
MOVE7
G2
MOVE UPDATED BLOCK NUMBER TO DISK ADDR
MOVE6
G3
TKCOMP MP
DETERMINE DISK ADDRESS FOR NEXT TRACK *J3
F1
E1
H1
SVC 0 WRITE BLOCK AT NEW ADDRESS
J1
DISK ERROR
YES
NO
MN B1
DSKERR
*J3 INCREMENTED 1 TRACK ON PASS 1, DECREMENTED 1 TRACK ON PASS 2.
K1
MORE ACTIVE BLKS
YES
NO
F3
C2
A4
MOVE9
A4
TEST MOVE-FULL-TRACKS-LATER SWITCH
ON
OFF
B4
PASS 2 COMPLETE CI LIB
YES
NO
ML B1
TKFMT
MOVE11
B5
GET NO. OF BLKS TO BE MOVED, RESET
G3
C4
DECREMENT ADDR TO PRECEDING ALLOCATION TABLE
D4
MOVE10
D4
SET PASS INDICATOR TO PASS 2, *D5
*D5 2ND PASS IS A REVERSE SCAN OF THE ALLOCATION TABLES STARTING WITH SS LIB.
E4
MOVE TO HIGHER DISK ADDR, 2ND PASS
NO
YES
A4
F4
TKCOMP MP
COMPUTE OLD ADDRESS OF LAST TRACK *F5
*F5 ADDR OF LAST TRACK=STARTING DISK ADDR + TRACKS USED -1.
G4
TKCOMP MP
COMPUTE NEW ADDRESS OF LAST TRACK
H4
LAST TRACK FULL
YES
NO
H5
ANY FULL TRACKS
NO
YES
J4
SET MOVE-FULL-TRACKS-LATER SWITCH
K4
SAVE NO.OF RECORDS OF FULL TRACKS TO MOVE LATER
MOVE12
K5
LOAD NUMBER OF RECORDS IN LAST TRACK
D1

Chart ML. MAINTA - Format Unused Tracks (Part 1 of 2)
Refer to Chart 40

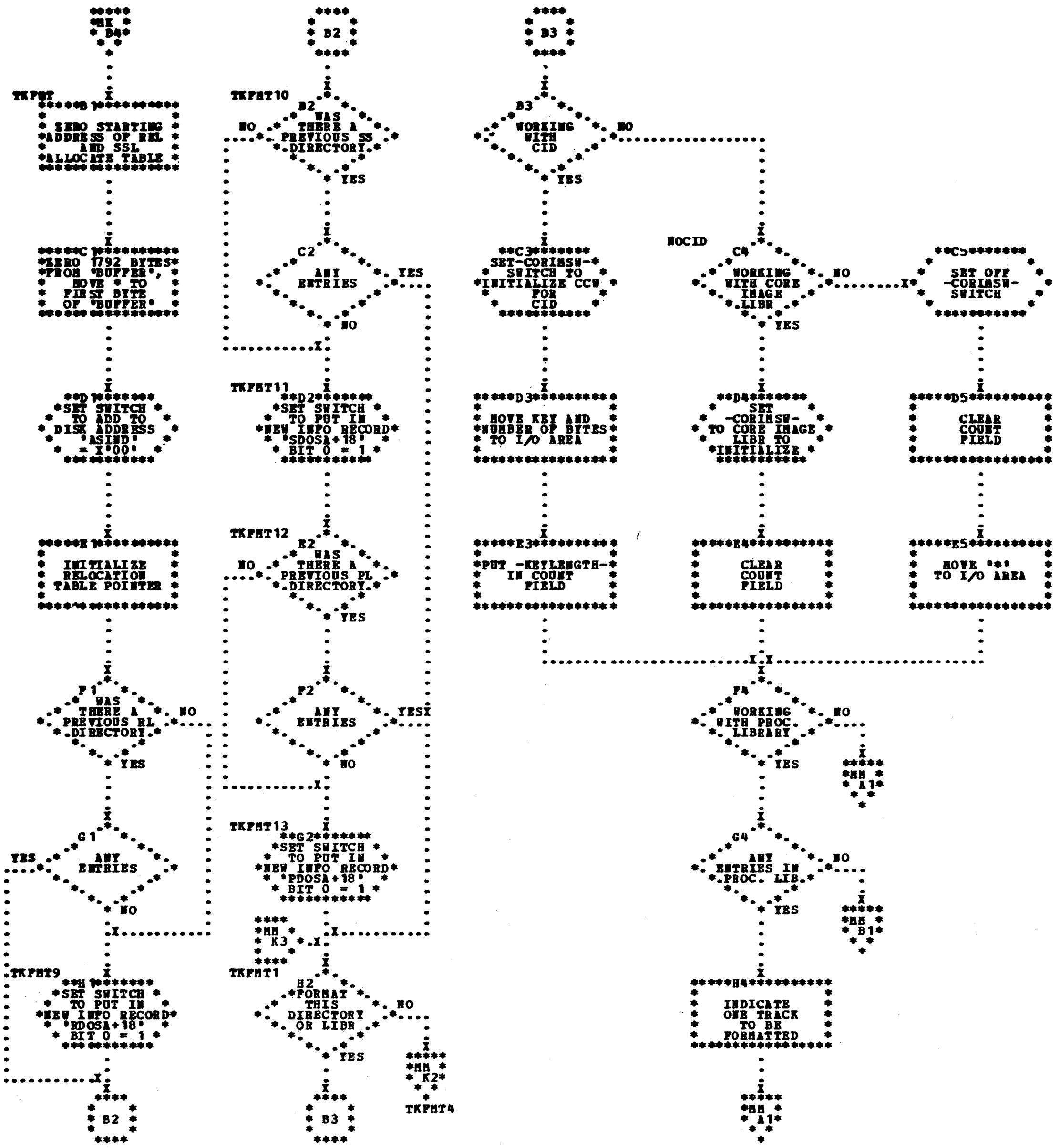

MK B4
TKFMT
B1
ZERO STARTING ADDRESS OF REL AND SSL ALLOCATE TABLE
C1
ZERO 1792 BYTES FROM 'BUFFER', MOVE * TO FIRST BYTE OF 'BUFFER'
D1
SET SWITCH TO ADD TO DISK ADDRESS 'ASIND' = X'00'
E1
INITIALIZE RELOCATION TABLE POINTER
F1
WAS THERE A PREVIOUS RL DIRECTORY
NO
YES
G1
ANY ENTRIES
YES
NO
TKFMT9
H1
SET SWITCH TO PUT IN NEW INFO RECORD 'RDOSA+18' BIT 0 = 1
B2
B2
TKFMT10
B2
WAS THERE A PREVIOUS SS DIRECTORY
NO
YES
C2
ANY ENTRIES
YES
NO
TKFMT11
D2
SET SWITCH TO PUT IN NEW INFO RECORD 'SDOSA+18' BIT 0 = 1
TKFMT12
E2
WAS THERE A PREVIOUS PL DIRECTORY
NO
YES
F2
ANY ENTRIES
YES
NO
TKFMT13
G2
SET SWITCH TO PUT IN NEW INFO RECORD 'PDOSA+18' BIT 0 = 1
MM K3
TKFMT1
H2
FORMAT THIS DIRECTORY OR LIBR
NO
YES
MM K2
TKFMT4
B3
B3
B3
WORKING WITH CID
NO
YES
C3
SET-CORIMSW- SWITCH TO INITIALIZE CCW FOR CID
D3
MOVE KEY AND NUMBER OF BYTES TO I/O AREA
E3
PUT -KEYLENGTH- IN COUNT FIELD
NOCID
C4
WORKING WITH CORE IMAGE LIBR
NO
YES
D4
SET -CORIMSW- TO CORE IMAGE LIBR TO INITIALIZE
E4
CLEAR COUNT FIELD
F4
WORKING WITH PROC. LIBRARY
NO
YES
MM A1
G4
ANY ENTRIES IN PROC. LIB.
NO
YES
MM B1
H4
INDICATE ONE TRACK TO BE FORMATTED
MM A1
C5
SET OFF -CORIMSW- SWITCH
D5
CLEAR COUNT FIELD
E5
MOVE '*' TO I/O AREA

Chart MM. MAINTA - Format Unused Tracks (Part 2 of 2)
Refer to Chart 40

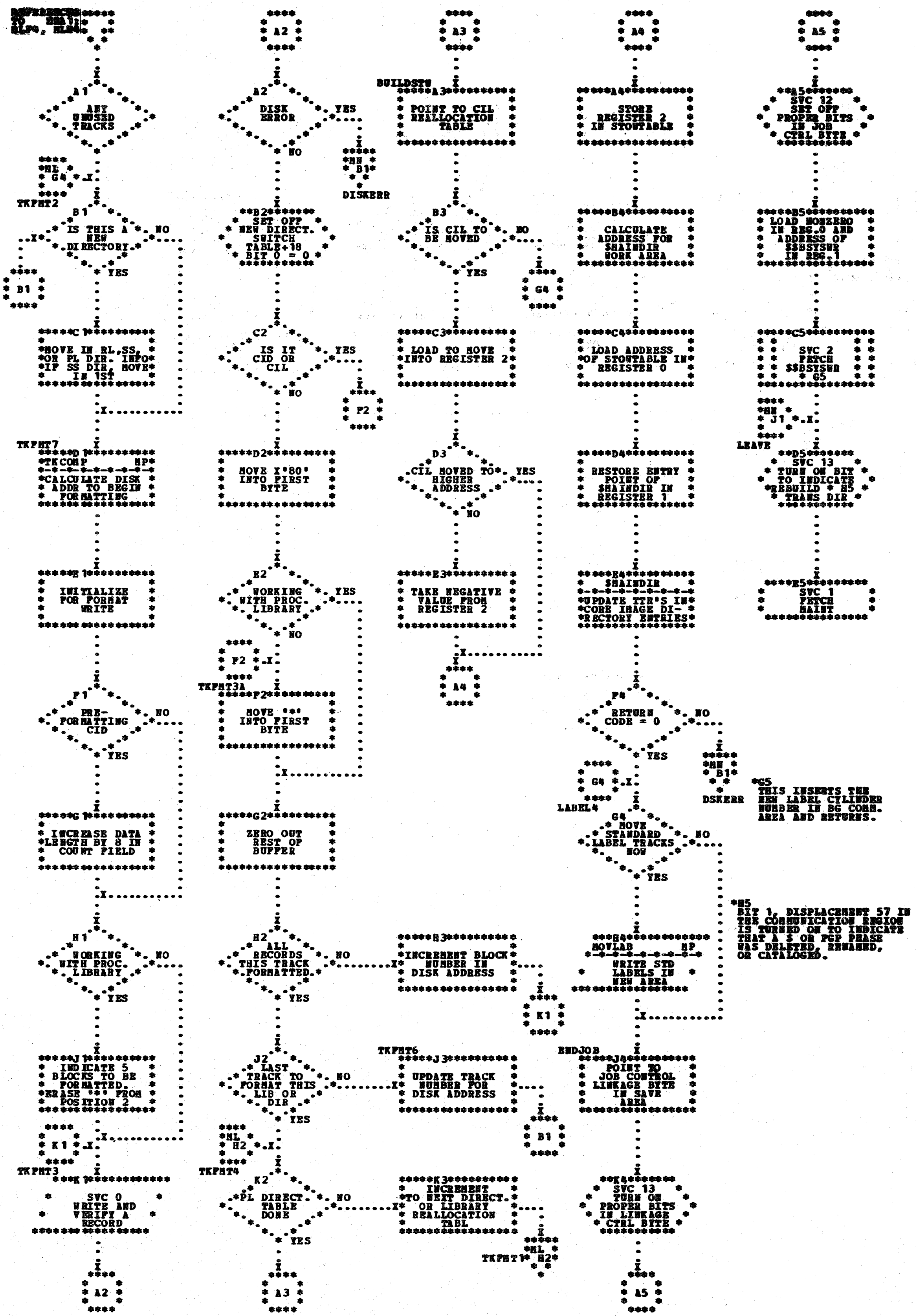

REFERENCES TO MM1; MLP4, ML84
A1 ANY UNUSED TRACKS
ML G4
TKFMT2
B1 IS THIS A NEW DIRECTORY
NO
YES
B1
C1 MOVE IN RL,SS, OR PL DIR. INFO IF SS DIR, MOVE IN 1ST
TKFMT7
D1 TKCOMP MP
CALCULATE DISK ADDR TO BEGIN FORMATTING
E1 INITIALIZE FOR FORMAT WRITE
F1 PRE-FORMATTING CID
NO
YES
G1 INCREASE DATA LENGTH BY 8 IN COUNT FIELD
H1 WORKING WITH PROC. LIBRARY
NO
YES
J1 INDICATE 5 BLOCKS TO BE FORMATTED. ERASE '*' FROM POSITION 2
K1
TKFMT3
K1 SVC 0 WRITE AND VERIFY A RECORD
A2
A2
A2 DISK ERROR
YES
MN B1
DISKERR
NO
B2 SET OFF NEW DIRECT. SWITCH TABLE+18 BIT 0 = 0
C2 IS IT CID OR CIL
YES
F2
NO
D2 MOVE X'80' INTO FIRST BYTE
E2 WORKING WITH PROC. LIBRARY
YES
NO
F2
TKFMT3A
F2 MOVE '*' INTO FIRST BYTE
G2 ZERO OUT REST OF BUFFER
H2 ALL RECORDS THIS TRACK FORMATTED
NO
YES
J2 LAST TRACK TO FORMAT THIS LIB OR DIR
NO
YES
ML H2
TKFMT4
K2 PL DIRECT. TABLE DONE
NO
YES
A3
A3
BUILDSTW
A3 POINT TO CIL REALLOCATION TABLE
B3 IS CIL TO BE MOVED
NO
G4
YES
C3 LOAD TO MOVE INTO REGISTER 2
D3 CIL MOVED TO HIGHER ADDRESS
YES
NO
E3 TAKE NEGATIVE VALUE FROM REGISTER 2
A4
H3 INCREMENT BLOCK NUMBER IN DISK ADDRESS
K1
TKFMT6
J3 UPDATE TRACK NUMBER FOR DISK ADDRESS
B1
K3 INCREMENT TO NEXT DIRECT. OR LIBRARY REALLOCATION TABL
ML H2
TKFMT1
A4
A4 STORE REGISTER 2 IN STOWTABLE
B4 CALCULATE ADDRESS FOR $MAINDIR WORK AREA
C4 LOAD ADDRESS OF STOWTABLE IN REGISTER 0
D4 RESTORE ENTRY POINT OF $MAINDIR IN REGISTER 1
E4 $MAINDIR
UPDATE TTR'S IN CORE IMAGE DI-RECTORY ENTRIES
F4 RETURN CODE = 0
NO
MN B1
DSKERR
YES
G4
LABEL4
G4 MOVE STANDARD LABEL TRACKS NOW
NO
YES
H4 MOVLAB MP
WRITE STD LABELS IN NEW AREA
ENDJOB
J4 POINT TO JOB CONTROL LINKAGE BYTE IN SAVE AREA
K4 SVC 13 TURN ON PROPER BITS IN LINKAGE CTRL BYTE
A5
A5
A5 SVC 12 SET OFF PROPER BITS IN JOB CTRL BYTE
B5 LOAD NONZERO IN REG.0 AND ADDRESS OF $$BSYSWR IN REG.1
C5 SVC 2 FETCH $$BSYSWR * G5
MN J1
LEAVE
D5 SVC 13 TURN ON BIT TO INDICATE REBUILD * H5 TRANS DIR
E5 SVC 1 FETCH MAINT
*G5 THIS INSERTS THE NEW LABEL CYLINDER NUMBER IN BG COMM. AREA AND RETURNS.
*H5 BIT 1, DISPLACEMENT 57 IN THE COMMUNICATION REGION IS TURNED ON TO INDICATE THAT A $ OR FGP PHASE WAS DELETED, RENAMED, OR CATALOGED.

Chart MN. MAINTA - Error Routines
Refer to Chart 40

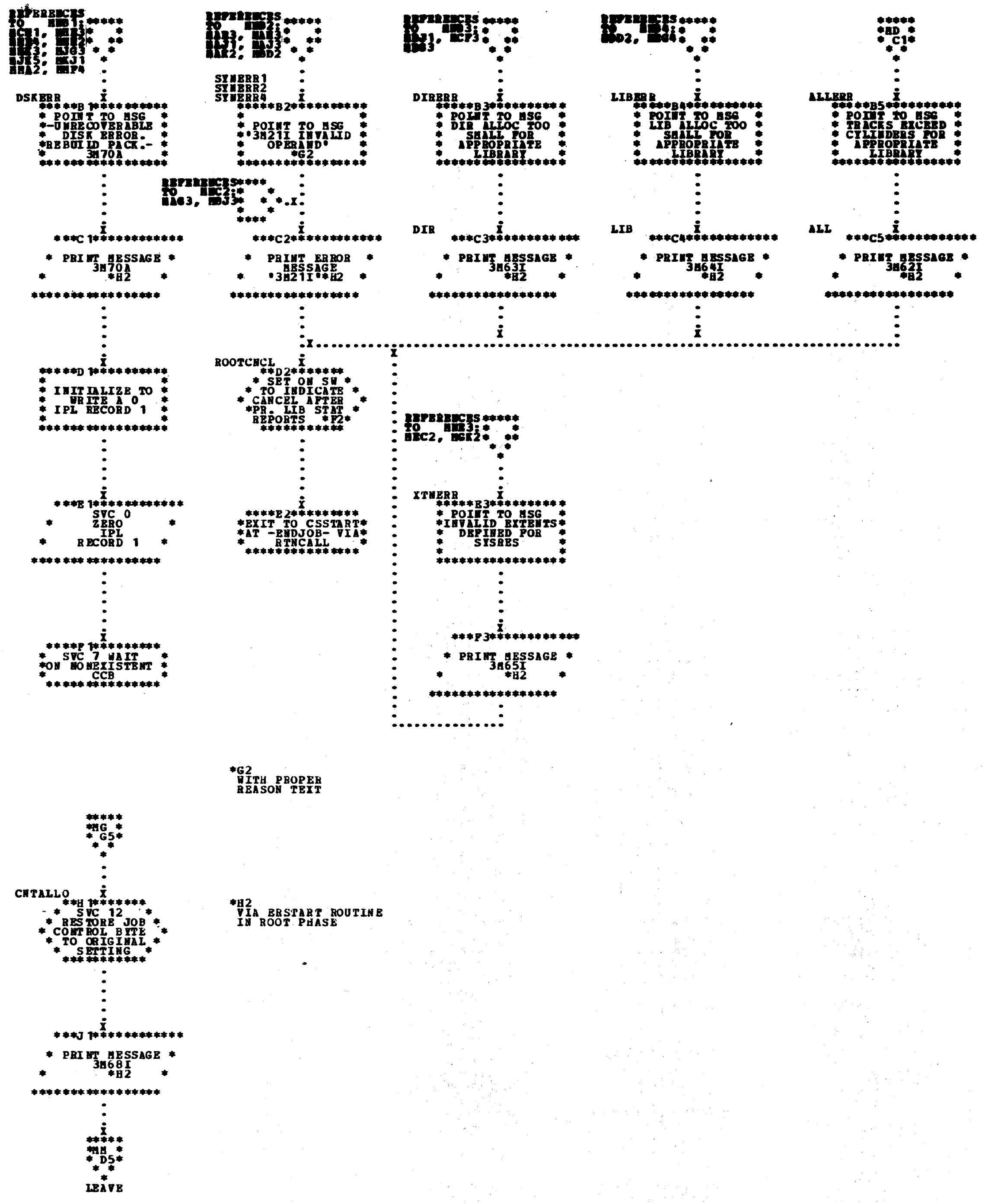

REFERENCES TO MNB1; MCH1, MNB3; MJG3, MJK5, MKJ1, MNA2, MNF4
DSKERR
B1
POINT TO MSG -UNRECOVERABLE DISK ERROR. REBUILD PACK.- 3M70A
C1
PRINT MESSAGE 3M70A *H2
D1
INITIALIZE TO WRITE A 0 IPL RECORD 1
E1
SVC 0 ZERO IPL RECORD 1
F1
SVC 7 WAIT ON NONEXISTENT CCB
REFERENCES TO MND2; MAB3, MAH3, MAJ1, MAJ3, MAK2, MND2
SYNERR1
SYNERR2
SYNERR4
B2
POINT TO MSG '3M21I INVALID OPERAND' *G2
REFERENCES TO MEC2; MAG3, MBJ3
C2
PRINT ERROR MESSAGE '3M21I' *H2
ROOTCNCL
D2
SET ON SW TO INDICATE CANCEL AFTER PR. LIB STAT REPORTS *F2
E2
EXIT TO CSSTART AT -ENDJOB- VIA RTNCALL
REFERENCES TO MND3; MBJ1, MCF3, MND3
DIRERR
B3
POINT TO MSG DIR ALLOC TOO SMALL FOR APPROPRIATE LIBRARY
DIR
C3
PRINT MESSAGE 3M63I *H2
REFERENCES TO MNB3; MEC2, MGK2
XTNERR
E3
POINT TO MSG INVALID EXTENTS DEFINED FOR SYSRES
F3
PRINT MESSAGE 3M65I *H2
REFERENCES TO MND4; MDD2, MDG4
LIBERR
B4
POINT TO MSG LIB ALLOC TOO SMALL FOR APPROPRIATE LIBRARY
LIB
C4
PRINT MESSAGE 3M64I *H2
MD C1
ALLERR
B5
POINT TO MSG TRACKS EXCEED CYLINDERS FOR APPROPRIATE LIBRARY
ALL
C5
PRINT MESSAGE 3M62I *H2
*G2
WITH PROPER REASON TEXT
MG G5
CNTALLO
H1
SVC 12 RESTORE JOB CONTROL BYTE TO ORIGINAL SETTING
*H2
VIA ERSTART ROUTINE IN ROOT PHASE
J1
PRINT MESSAGE 3M68I *H2
MM D5
LEAVE

Chart MP. MAINTA - Subroutines
Refer to Chart 40

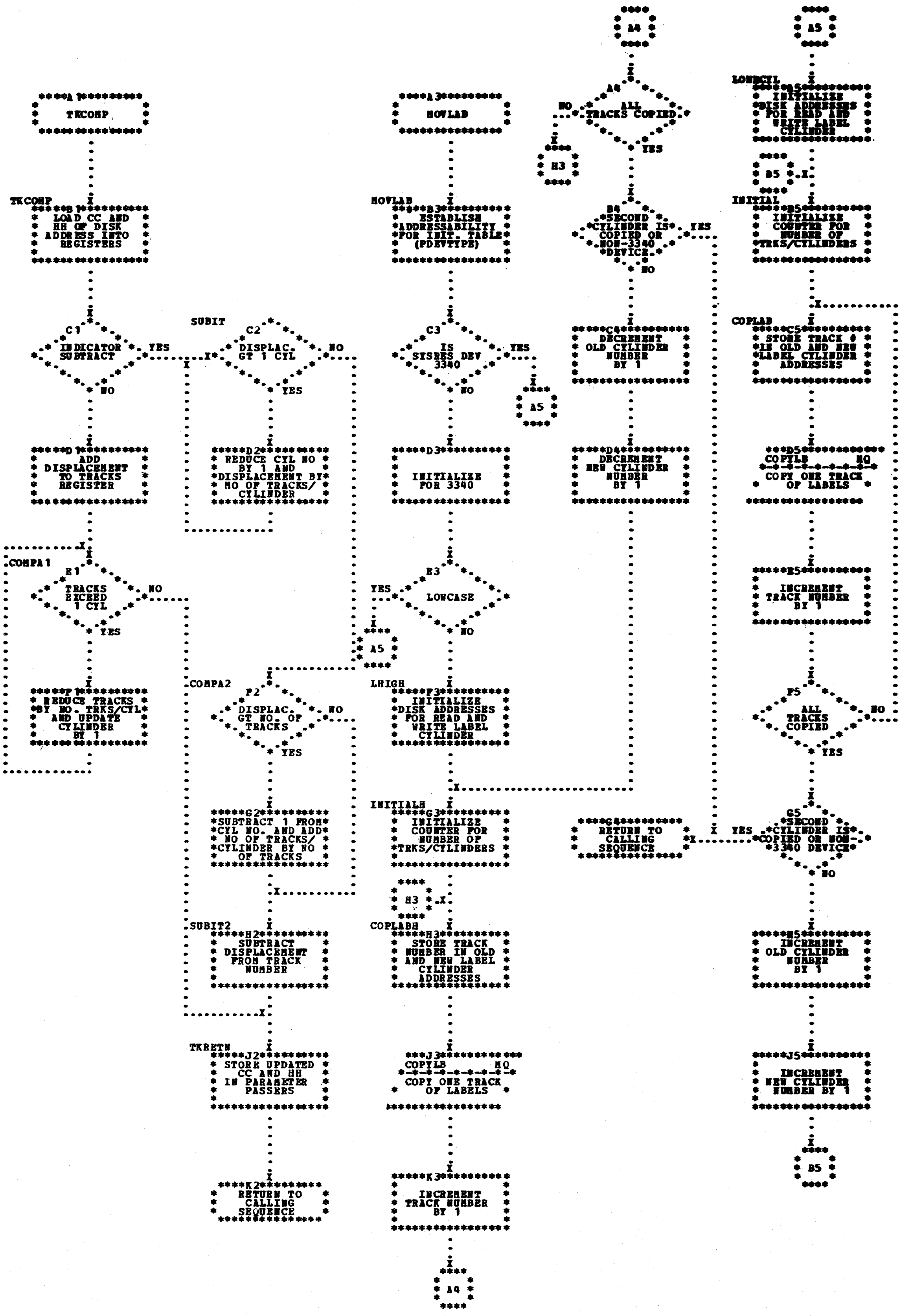

A1
TKCOMP
TKCOMP
B1
LOAD CC AND HH OF DISK ADDRESS INTO REGISTERS
C1
INDICATOR SUBTRACT
YES
NO
D1
ADD DISPLACEMENT TO TRACKS REGISTER
COMPA1
E1
TRACKS EXCEED 1 CYL
NO
YES
F1
REDUCE TRACKS BY NO. TRKS/CYL AND UPDATE CYLINDER BY 1
SUBIT
C2
DISPLAC. GT 1 CYL
NO
YES
D2
REDUCE CYL NO BY 1 AND DISPLACEMENT BY NO OF TRACKS/ CYLINDER
COMPA2
F2
DISPLAC. GT NO. OF TRACKS
NO
YES
G2
SUBTRACT 1 FROM CYL NO. AND ADD NO OF TRACKS/ CYLINDER BY NO OF TRACKS
SUBIT2
H2
SUBTRACT DISPLACEMENT FROM TRACK NUMBER
TKRETN
J2
STORE UPDATED CC AND HH IN PARAMETER PASSERS
K2
RETURN TO CALLING SEQUENCE
A3
MOVLAB
MOVLAB
B3
ESTABLISH ADDRESSABILITY FOR INIT. TABLE (PDEVTYPE)
C3
IS SYSRES DEV 3340
YES
NO
A5
D3
INITIALIZE FOR 3340
E3
LOWCASE
YES
NO
A5
LHIGH
F3
INITIALIZE DISK ADDRESSES FOR READ AND WRITE LABEL CYLINDER
INITIALH
G3
INITIALIZE COUNTER FOR NUMBER OF TRKS/CYLINDERS
H3
COPLABH
H3
STORE TRACK NUMBER IN OLD AND NEW LABEL CYLINDER ADDRESSES
J3
COPYLB
NO
COPY ONE TRACK OF LABELS
K3
INCREMENT TRACK NUMBER BY 1
A4
A4
A4
ALL TRACKS COPIED
NO
YES
H3
B4
SECOND CYLINDER IS COPIED OR NON-3340 DEVICE
YES
NO
C4
DECREMENT OLD CYLINDER NUMBER BY 1
D4
DECREMENT NEW CYLINDER NUMBER BY 1
G4
RETURN TO CALLING SEQUENCE
A5
LOWHCYL
A5
INITIALIZE DISK ADDRESSES FOR READ AND WRITE LABEL CYLINDER
B5
INITIAL
B5
INITIALIZE COUNTER FOR NUMBER OF TRKS/CYLINDERS
COPLAB
C5
STORE TRACK # IN OLD AND NEW LABEL CYLINDER ADDRESSES
D5
COPYLB
NO
COPY ONE TRACK OF LABELS
E5
INCREMENT TRACK NUMBER BY 1
F5
ALL TRACKS COPIED
NO
YES
G5
SECOND CYLINDER IS COPIED OR NON-3340 DEVICE
YES
NO
H5
INCREMENT OLD CYLINDER NUMBER BY 1
J5
INCREMENT NEW CYLINDER NUMBER BY 1
B5

Chart MQ. MAINTA - Subroutines
Refer to Chart 40

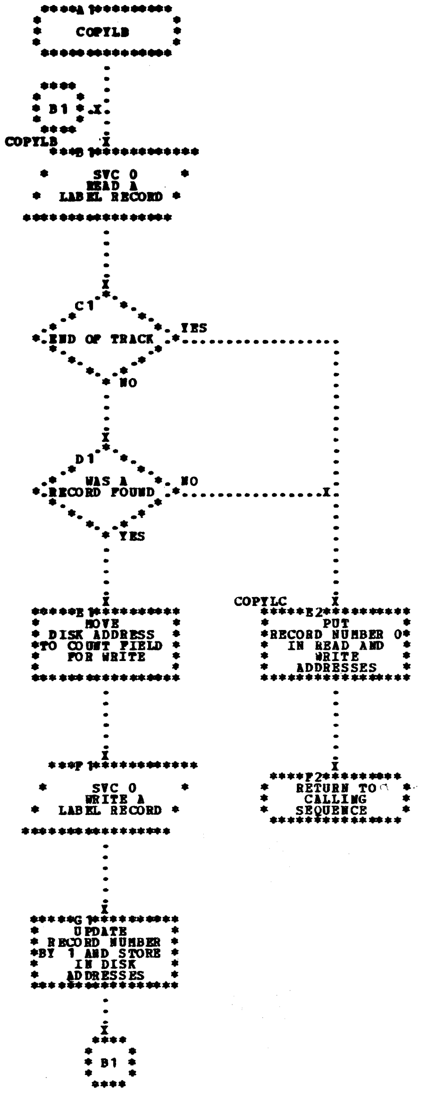


Chart NA. MAINTUP - Initialize I/O Table, Check for Operands (Part 1 of 3)
Refer to Charts 44-46

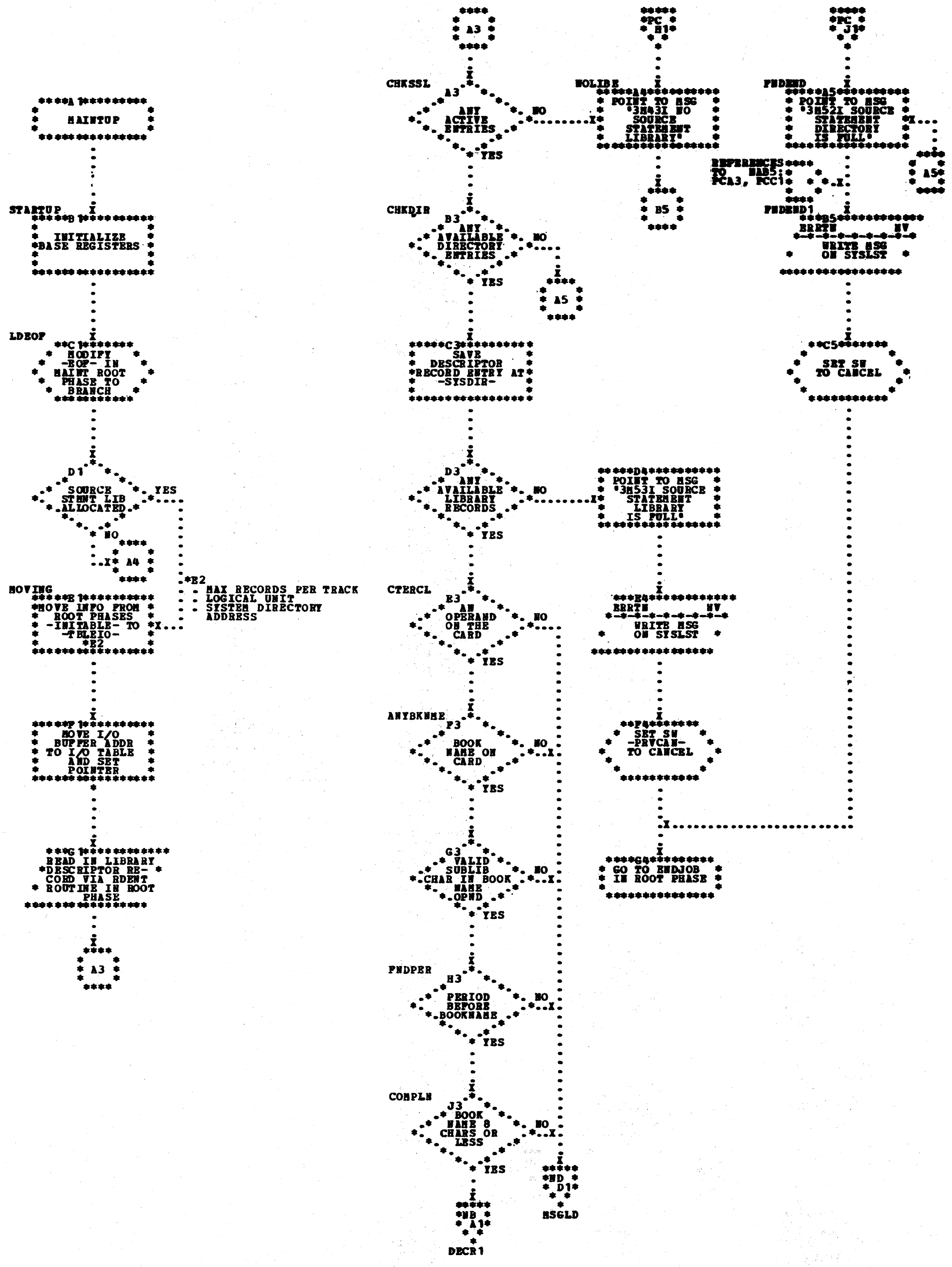

A1 MAINTUP
STARTUP
B1 INITIALIZE BASE REGISTERS
LDEOF
C1 MODIFY -EOF- IN MAINT ROOT PHASE TO BRANCH
D1 SOURCE STMNT LIB ALLOCATED
YES
NO
A4
MOVING
E1 MOVE INFO FROM ROOT PHASES -INITABLE- TO -TBLEIO- *E2
*E2
- MAX RECORDS PER TRACK
- LOGICAL UNIT
- SYSTEM DIRECTORY ADDRESS
F1 MOVE I/O BUFFER ADDR TO I/O TABLE AND SET POINTER
G1 READ IN LIBRARY DESCRIPTOR RE-CORD VIA RDENT ROUTINE IN ROOT PHASE
A3
A3
CHKSSL
A3 ANY ACTIVE ENTRIES
NO
YES
CHKDIR
B3 ANY AVAILABLE DIRECTORY ENTRIES
NO
YES
A5
C3 SAVE DESCRIPTOR RECORD ENTRY AT -SYSDIR-
D3 ANY AVAILABLE LIBRARY RECORDS
NO
YES
CTERCL
E3 AN OPERAND ON THE CARD
NO
YES
ANYBKNME
F3 BOOK NAME ON CARD
NO
YES
G3 VALID SUBLIB CHAR IN BOOK NAME OPND
NO
YES
PNDPER
H3 PERIOD BEFORE BOOKNAME
NO
YES
COMPLN
J3 BOOK NAME 8 CHARS OR LESS
NO
YES
NB A1
DECR1
ND D1
MSGLD
PC H1
NOLIBE
A4 POINT TO MSG '3M43I NO SOURCE STATEMENT LIBRARY'
B5
D4 POINT TO MSG '3M53I SOURCE STATEMENT LIBRARY IS FULL'
E4 ERRTN NV
WRITE MSG ON SYSLST
F4 SET SW -PRVCAN- TO CANCEL
G4 GO TO ENDJOB IN ROOT PHASE
PC J1
PNDEND
A5 POINT TO MSG '3M52I SOURCE STATEMENT DIRECTORY IS FULL'
A5
REFERENCES TO MAB5: PCA3, PCC1
PNDEND1
B5 ERRTN NV
WRITE MSG ON SYSLST
C5 SET SW TO CANCEL

Chart NB. MAINTUP - Initialize I/O Table, Check for Operands (Part 2 of 3)
Refer to Charts 44-46

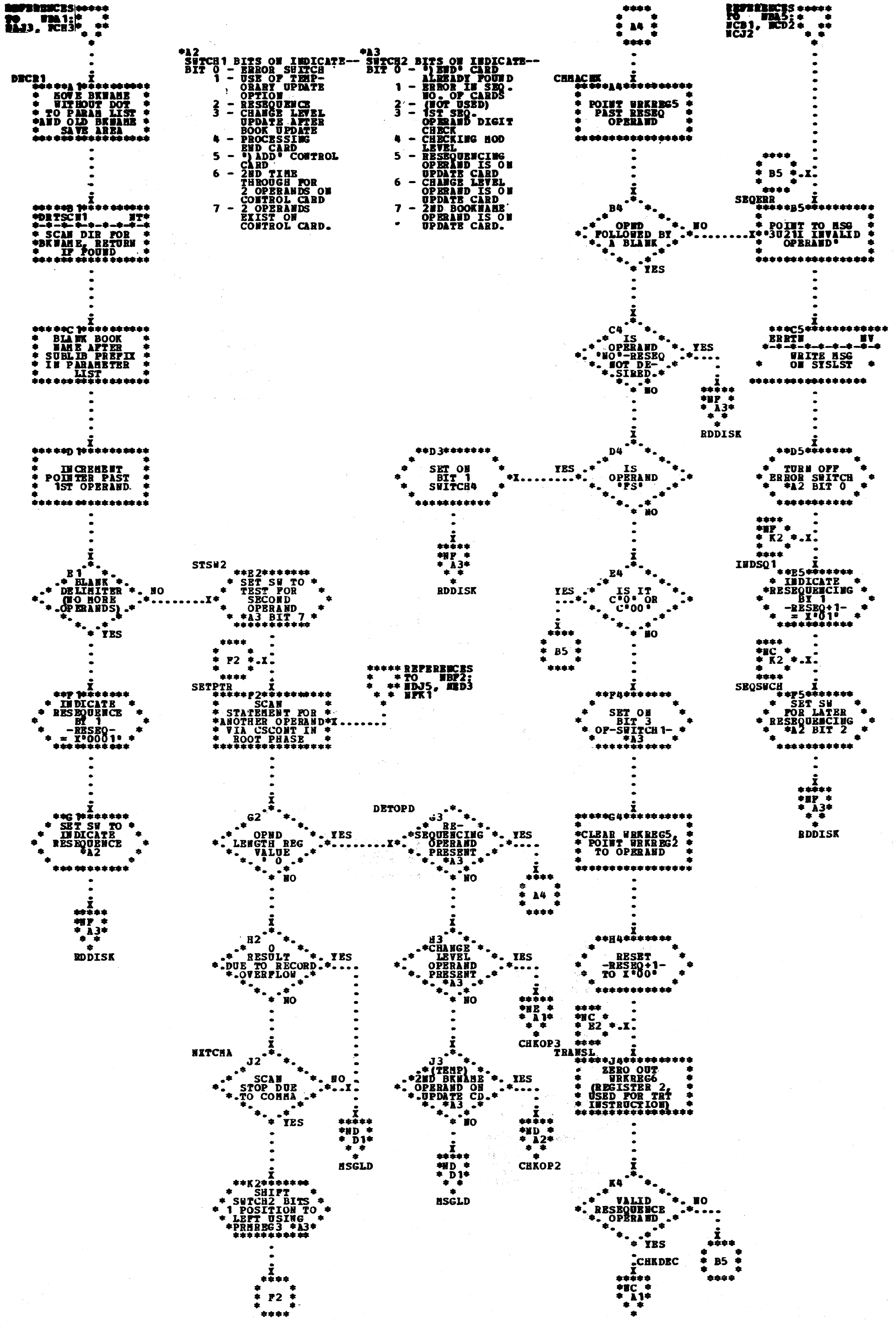
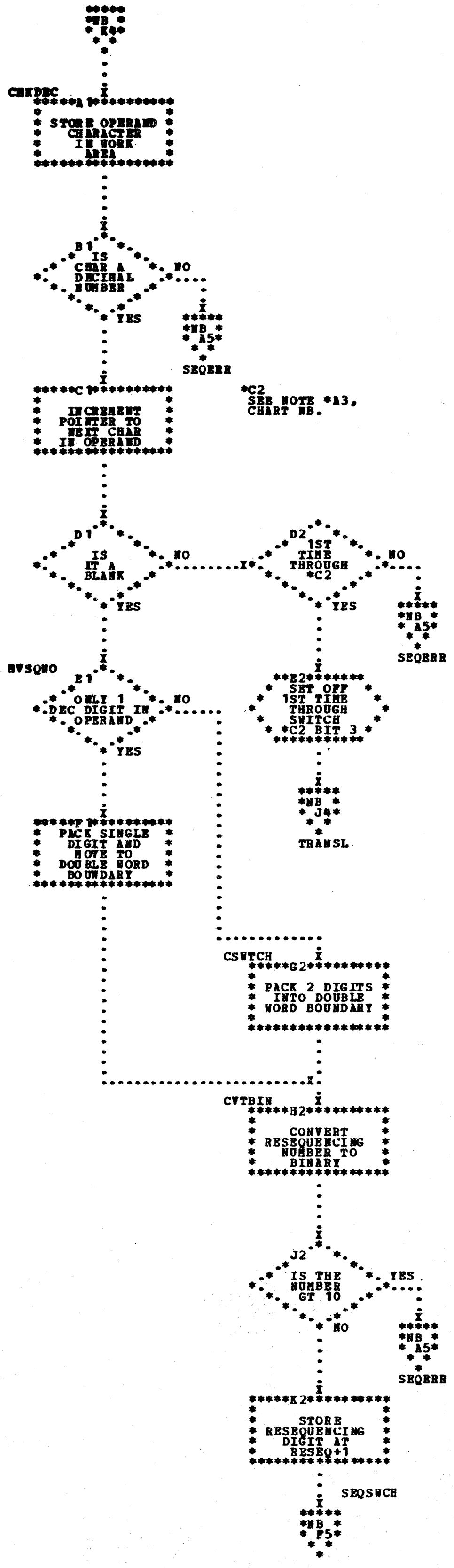
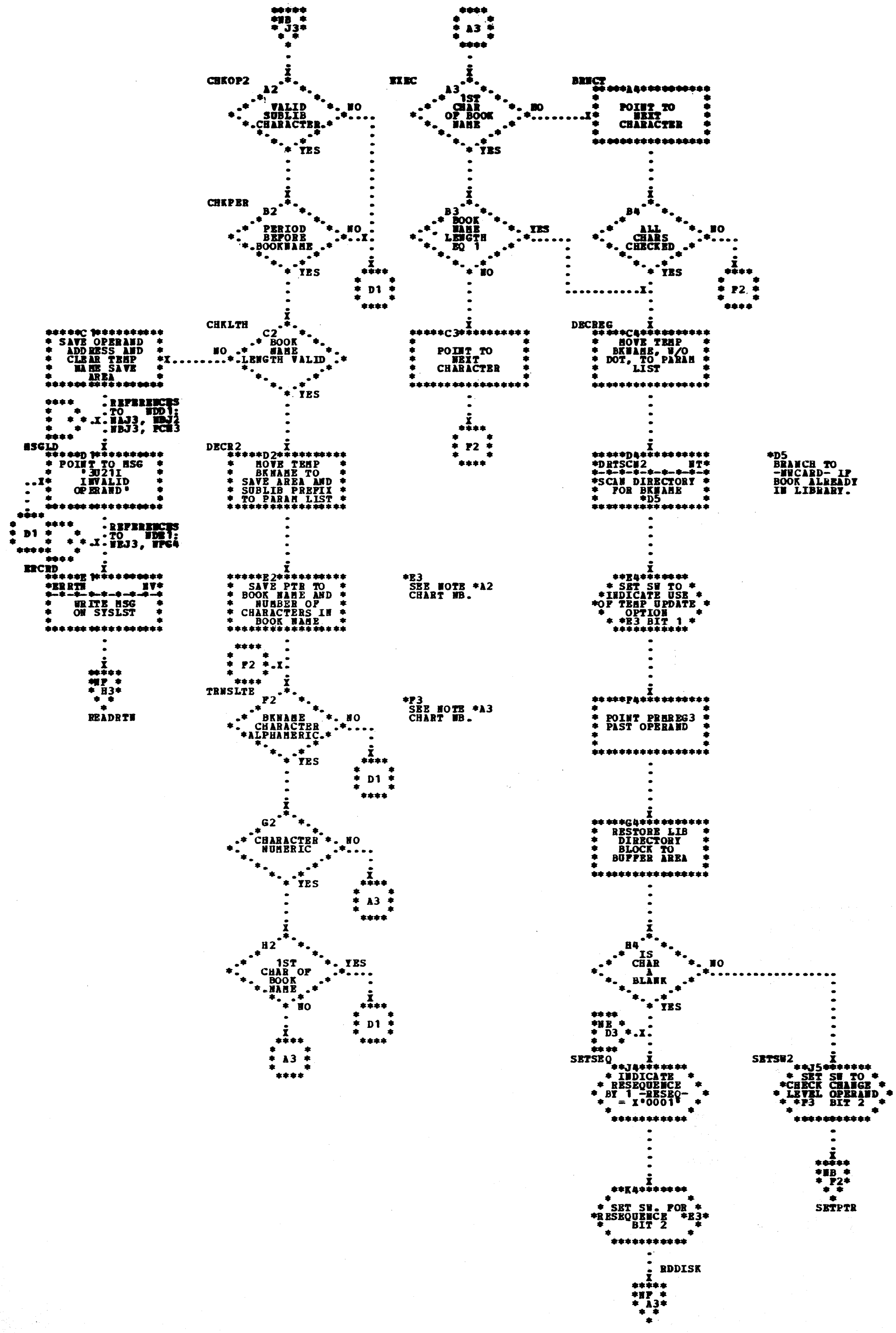

REFERENCES TO NBA1: NAJ3, NCH3
DECR1
A1 MOVE BKNAME WITHOUT DOT TO PARAM LIST AND OLD BKNAME SAVE AREA
B1 DRTSCN1 NT
SCAN DIR FOR BKNAME, RETURN IF FOUND
C1 BLANK BOOK NAME AFTER SUBLIB PREFIX IN PARAMETER LIST
D1 INCREMENT POINTER PAST 1ST OPERAND
E1 BLANK DELIMITER (NO MORE OPERANDS)
NO
YES
F1 INDICATE RESEQUENCE BY 1 -RESEQ- = X'0001'
G1 SET SW TO INDICATE RESEQUENCE *A2
NF A3 RDDISK
*A2
SWTCH1 BITS ON INDICATE--
BIT 0 - ERROR SWITCH
1 - USE OF TEMPORARY UPDATE OPTION
2 - RESEQUENCE
3 - CHANGE LEVEL UPDATE AFTER BOOK UPDATE
4 - PROCESSING END CARD
5 - ') ADD' CONTROL CARD
6 - 2ND TIME THROUGH FOR 2 OPERANDS ON CONTROL CARD
7 - 2 OPERANDS EXIST ON CONTROL CARD.
*A3
SWTCH2 BITS ON INDICATE--
BIT 0 - ') END' CARD ALREADY FOUND
1 - ERROR IN SEQ. NO. OF CARDS
2 - (NOT USED)
3 - 1ST SEQ. OPERAND DIGIT CHECK
4 - CHECKING MOD LEVEL
5 - RESEQUENCING OPERAND IS ON UPDATE CARD
6 - CHANGE LEVEL OPERAND IS ON UPDATE CARD
7 - 2ND BOOKNAME OPERAND IS ON UPDATE CARD.
STSW2
E2 SET SW TO TEST FOR SECOND OPERAND *A3 BIT 7
F2
SETPTR
F2 SCAN STATEMENT FOR ANOTHER OPERAND VIA CSCONT IN ROOT PHASE
REFERENCES TO NBF2: NDJ5, NED3 NFK1
G2 OPND LENGTH REG VALUE ' 0
YES
NO
H2 0 RESULT DUE TO RECORD OVERFLOW
YES
NO
NXTCHA
J2 SCAN STOP DUE TO COMMA
NO
YES
ND D1 MSGLD
K2 SHIFT SWTCH2 BITS 1 POSITION TO LEFT USING PRMREG3 *A3
F2
D3 SET ON BIT 1 SWITCH4
NF A3 RDDISK
DETOPD
G3 RE-SEQUENCING OPERAND PRESENT *A3
YES
NO
A4
H3 CHANGE LEVEL OPERAND PRESENT *A3
YES
NO
NE A1 CHKOP3
J3 (TEMP) 2ND BKNAME OPERAND ON UPDATE CD. *A3
YES
NO
ND A2 CHKOP2
ND D1 MSGLD
A4
CHHACHK
A4 POINT WRKREG5 PAST RESEQ OPERAND
B4 OPND FOLLOWED BY A BLANK
NO
YES
C4 IS OPERAND 'NO'-RESEQ NOT DESIRED
YES
NO
NF A3 RDDISK
D4 IS OPERAND 'FS'
YES
NO
E4 IS IT C'0' OR C'00'
YES
NO
B5
F4 SET ON BIT 3 OP-SWITCH1- *A3
G4 CLEAR WRKREG5, POINT WRKREG2 TO OPERAND
H4 RESET -RESEQ+1- TO X'00'
NC B2
TRANSL
J4 ZERO OUT WRKREG6 (REGISTER 2, USED FOR TRT INSTRUCTION)
K4 VALID RESEQUENCE OPERAND
NO
YES
CHKDEC
NC A1
B5
REFERENCES TO NBA5: NCB1, NCD2 NCJ2
B5
SEQERR
B5 POINT TO MSG '3U21I INVALID OPERAND'
C5 ERRTN NV
WRITE MSG ON SYSLST
D5 TURN OFF ERROR SWITCH *A2 BIT 0
NF K2
INDSQ1
E5 INDICATE RESEQUENCING BY 1 -RESEQ+1- = X'01'
NC K2
SEQSWCH
F5 SET SW FOR LATER RESEQUENCING *A2 BIT 2
NF A3 RDDISK

Chart NC. MAINTUP - Initialize I/O Table, Check for Operands (Part 3 of 3)
Refer to Charts 44-46



NB K4
CHKDEC
A1
STORE OPERAND CHARACTER IN WORK AREA
B1
IS CHAR A DECIMAL NUMBER
NO
YES
NB A5
SEQERR
C1
INCREMENT POINTER TO NEXT CHAR IN OPERAND
*C2
SEE NOTE *A3, CHART NB.
D1
IS IT A BLANK
NO
YES
D2
1ST TIME THROUGH *C2
NO
YES
NB A5
SEQERR
RVSQNO
E1
ONLY 1 DEC DIGIT IN OPERAND
NO
YES
E2
SET OFF 1ST TIME THROUGH SWITCH *C2 BIT 3
NB J4
TRANSL
F1
PACK SINGLE DIGIT AND MOVE TO DOUBLE WORD BOUNDARY
CSWTCH
G2
PACK 2 DIGITS INTO DOUBLE WORD BOUNDARY
CVTBIN
H2
CONVERT RESEQUENCING NUMBER TO BINARY
J2
IS THE NUMBER GT 10
YES
NO
NB A5
SEQERR
K2
STORE RESEQUENCING DIGIT AT RESEQ+1
SEQSWCH
NB F5

Chart ND. MAINTUP - Check Temporary Update Operand
Refer to Charts 44-46

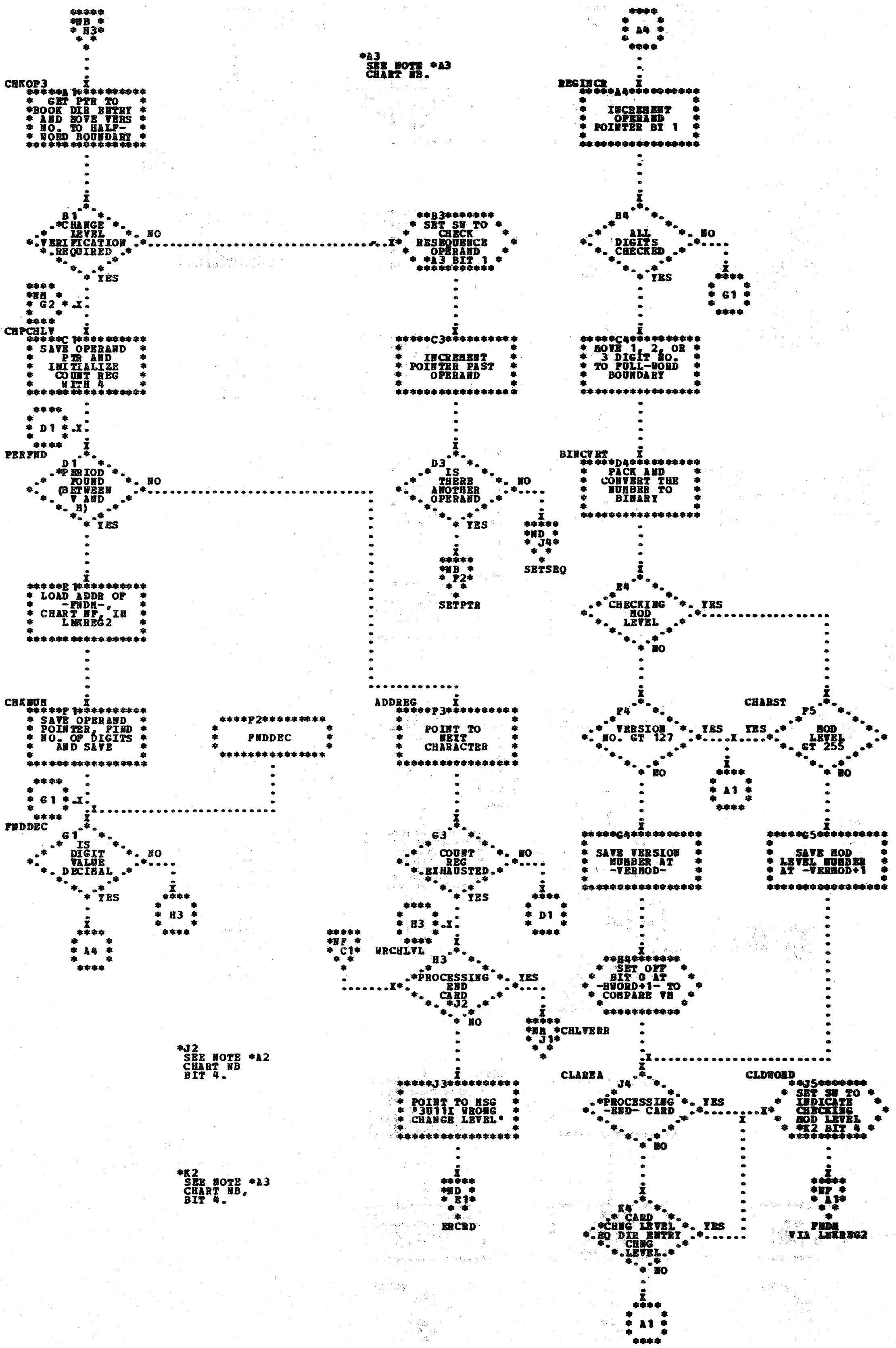

NB J3
CHKOP2
A2
VALID SUBLIB CHARACTER
NO
YES
CHKPER
B2
PERIOD BEFORE BOOKNAME
NO
YES
D1
CHKLTH
C2
BOOK NAME LENGTH VALID
NO
YES
C1
SAVE OPERAND ADDRESS AND CLEAR TEMP NAME SAVE AREA
REFERENCES TO NDD1: NAJ3, NBJ2 NBJ3, PCH3
MSGLD
D1
POINT TO MSG '3U21I INVALID OPERAND'
D1
REFERENCES TO NDE1: NEJ3, NPG4
ERCRD
E1
ERRTN NV
WRITE MSG ON SYSLST
NF H3
READRTN
DECR2
D2
MOVE TEMP BKNAME TO SAVE AREA AND SUBLIB PREFIX TO PARAM LIST
E2
SAVE PTR TO BOOK NAME AND NUMBER OF CHARACTERS IN BOOK NAME
F2
TRNSLTE
F2
BKNAME CHARACTER ALPHAMERIC
NO
YES
D1
G2
CHARACTER NUMERIC
NO
YES
A3
H2
1ST CHAR OF BOOK NAME
YES
NO
D1
A3
A3
NXEC
A3
1ST CHAR OF BOOK NAME
NO
YES
BRNCT
A4
POINT TO NEXT CHARACTER
B3
BOOK NAME LENGTH EQ 1
YES
NO
C3
POINT TO NEXT CHARACTER
F2
B4
ALL CHARS CHECKED
NO
YES
F2
DECREG
C4
MOVE TEMP BKNAME, W/O DOT, TO PARAM LIST
D4
DRTSCN2 NT
SCAN DIRECTORY FOR BKNAME *D5
E4
SET SW TO INDICATE USE OF TEMP UPDATE OPTION *E3 BIT 1
F4
POINT PRMREG3 PAST OPERAND
G4
RESTORE LIB DIRECTORY BLOCK TO BUFFER AREA
H4
IS CHAR A BLANK
NO
YES
NE D3
SETSEQ
J4
INDICATE RESEQUENCE BY 1 -RESEQ- = X'0001'
K4
SET SW. FOR RESEQUENCE *E3* BIT 2
RDDISK
NF A3
SETSW2
J5
SET SW TO CHECK CHANGE LEVEL OPERAND *F3 BIT 2
NB F2
SETPTR

*E3
SEE NOTE *A2
CHART NB.

*F3
SEE NOTE *A3
CHART NB.

*D5
BRANCH TO
-NWCARD- IF
BOOK ALREADY
IN LIBRARY.

Chart NE. MAINTUP - Process V.M Operand (Part 1 of 2)
Refer to Charts 44-46


NB H3
CHKOP3
A1 GET PTR TO BOOK DIR ENTRY AND MOVE VERS NO. TO HALF-WORD BOUNDARY
B1 CHANGE LEVEL VERIFICATION REQUIRED
NO
YES
NM G2
CHPCHLV
C1 SAVE OPERAND PTR AND INITIALIZE COUNT REG WITH 4
D1
PERFND
D1 PERIOD FOUND (BETWEEN V AND M)
NO
YES
E1 LOAD ADDR OF -FNDM-, CHART NF, IN LNKREG2
CHKNUM
F1 SAVE OPERAND POINTER, FIND NO. OF DIGITS AND SAVE
G1
FNDDEC
G1 IS DIGIT VALUE DECIMAL
NO
YES
H3
A4
F2 FNDDEC
B3 SET SW TO CHECK RESEQUENCE OPERAND *A3 BIT 1
C3 INCREMENT POINTER PAST OPERAND
D3 IS THERE ANOTHER OPERAND
NO
YES
ND J4 SETSEQ
NB F2 SETPTR
ADDREG
F3 POINT TO NEXT CHARACTER
G3 COUNT REG EXHAUSTED
NO
YES
D1
H3
NF C1
WRCHLVL
H3 PROCESSING END CARD *J2
YES
NO
NB J1 CHLVERR
J3 POINT TO MSG '3U11I WRONG CHANGE LEVEL'
ND E1 ERCRD
A4
REGINCR
A4 INCREMENT OPERAND POINTER BY 1
B4 ALL DIGITS CHECKED
NO
YES
G1
C4 MOVE 1, 2, OR 3 DIGIT NO. TO FULL-WORD BOUNDARY
BINCVRT
D4 PACK AND CONVERT THE NUMBER TO BINARY
E4 CHECKING MOD LEVEL
YES
NO
CHARST
F4 VERSION NO. GT 127
YES
NO
A1
F5 MOD LEVEL GT 255
YES
NO
G4 SAVE VERSION NUMBER AT -VERMOD-
G5 SAVE MOD LEVEL NUMBER AT -VERMOD+1
H4 SET OFF BIT 0 AT -HWORD+1- TO COMPARE VM
CLAREA
J4 PROCESSING -END- CARD
YES
NO
CLDWORD
J5 SET SW TO INDICATE CHECKING MOD LEVEL *K2 BIT 4
NF A1 FNDM VIA LNKREG2
K4 CARD CHNG LEVEL EQ DIR ENTRY CHNG LEVEL
YES
NO
A1

*A3
SEE NOTE *A3
CHART NB.

*J2
SEE NOTE *A2
CHART NB
BIT 4.

*K2
SEE NOTE *A3
CHART NB,
BIT 4.

Chart NF. MAINTUP - Process V.M Operand (Part 2 of 2)
Refer to Charts 44-46

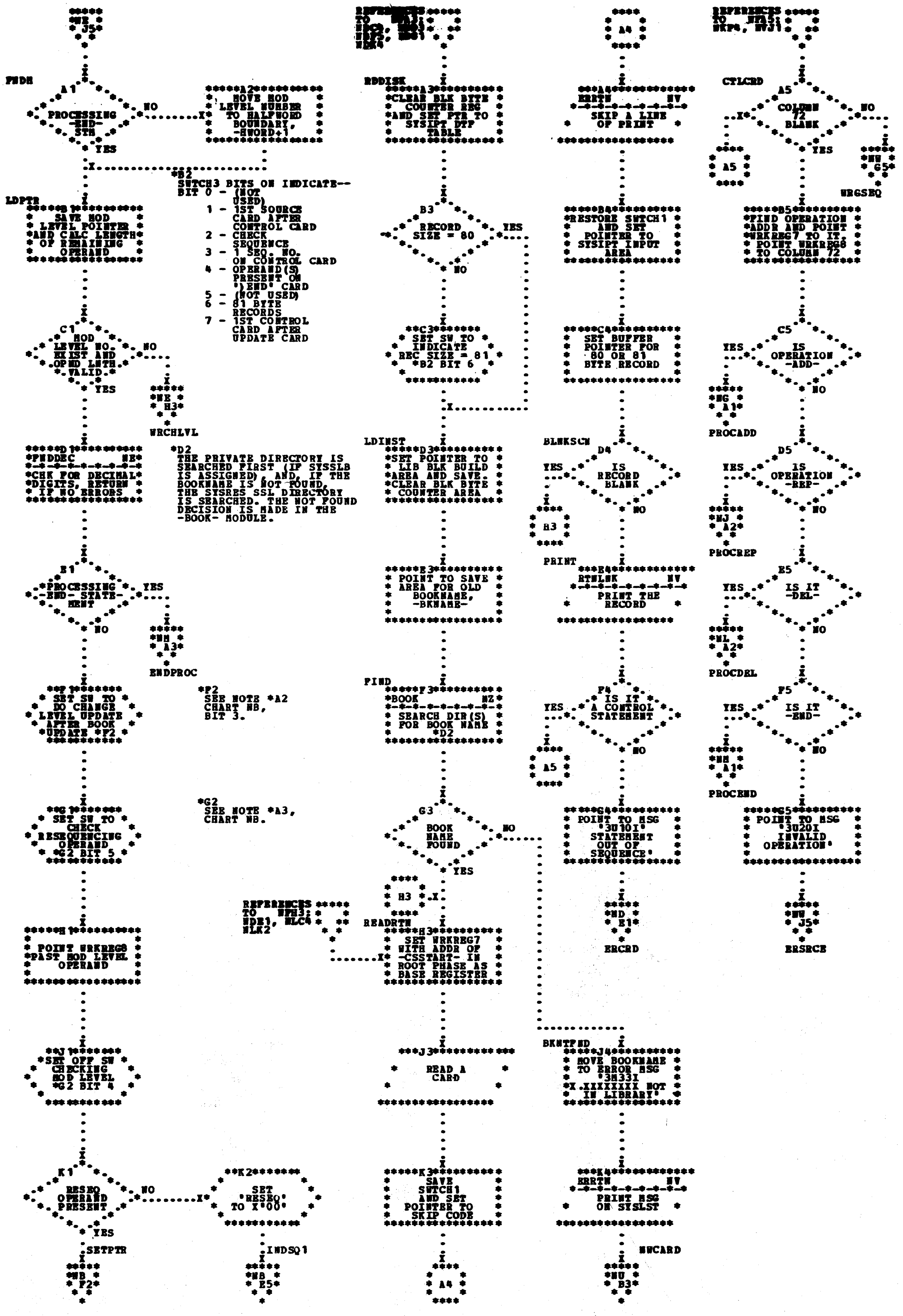
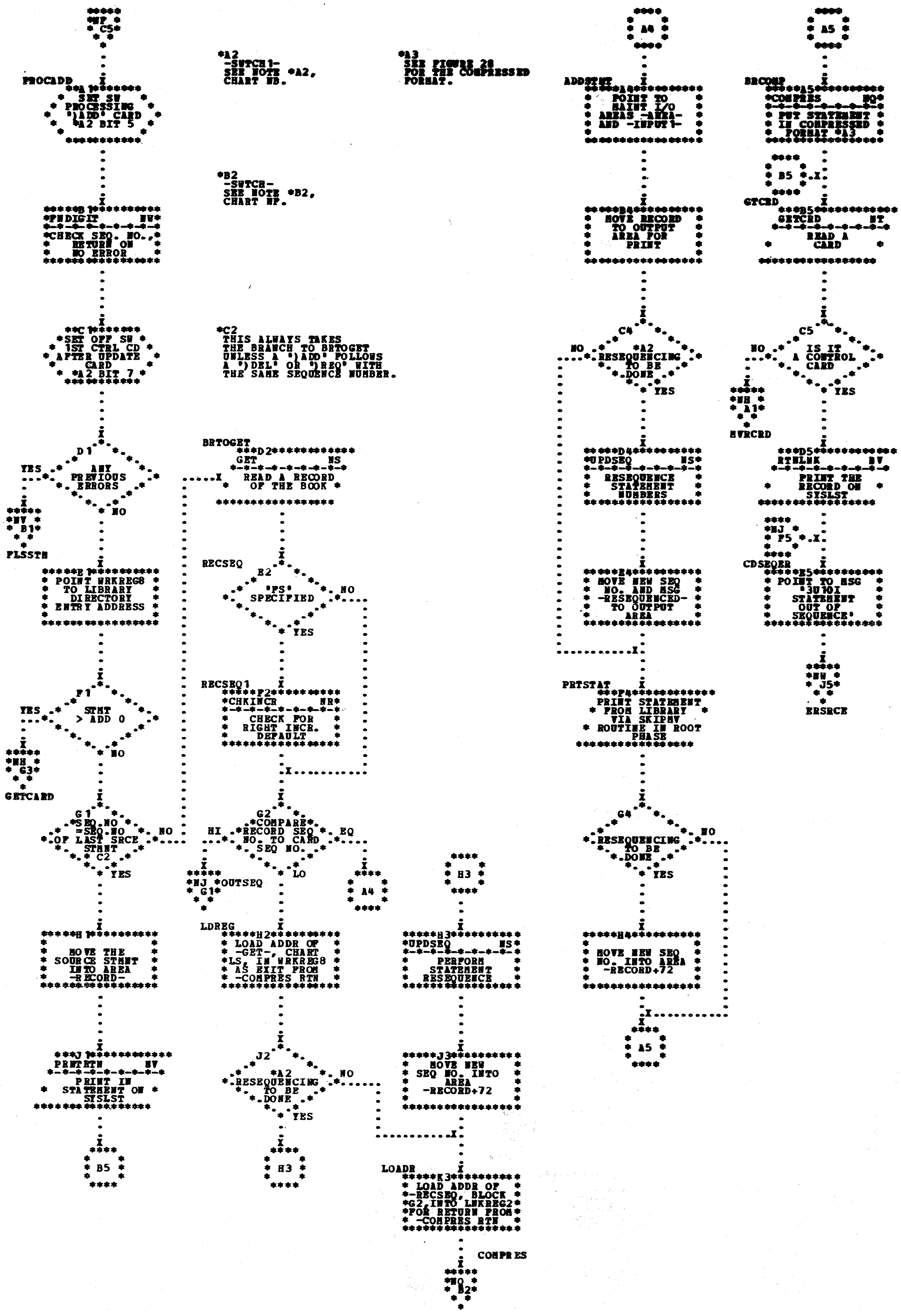

NE J5
FNDM
A1
PROCESSING -END- STM
NO
YES
A2
MOVE MOD LEVEL NUMBER TO HALFWORD BOUNDARY, -HWORD+1
*B2
SWTCH3 BITS ON INDICATE--
BIT 0 - (NOT USED)
1 - 1ST SOURCE CARD AFTER CONTROL CARD
2 - CHECK SEQUENCE
3 - 1 SEQ. NO. ON CONTROL CARD
4 - OPERAND(S) PRESENT ON ')END' CARD
5 - (NOT USED)
6 - 81 BYTE RECORDS
7 - 1ST CONTROL CARD AFTER UPDATE CARD
LDPTR
B1
SAVE MOD LEVEL POINTER AND CALC LENGTH OF REMAINING OPERAND
C1
MOD LEVEL NO. EXIST AND OPND LNTH. VALID.
NO
YES
NE H3
WRCHLVL
D1
FNDDEC NE
CHK FOR DECIMAL DIGITS, RETURN IF NO ERRORS
*D2
THE PRIVATE DIRECTORY IS SEARCHED FIRST (IF SYSSLB IS ASSIGNED), AND, IF THE BOOKNAME IS NOT FOUND, THE SYSRES SSL DIRECTORY IS SEARCHED. THE NOT FOUND DECISION IS MADE IN THE -BOOK- MODULE.
E1
PROCESSING -END- STATEMENT
YES
NO
NM A3
ENDPROC
F1
SET SW TO DO CHANGE LEVEL UPDATE AFTER BOOK UPDATE *F2
*F2
SEE NOTE *A2 CHART NB, BIT 3.
G1
SET SW TO CHECK RESEQUENCING OPERAND *G2 BIT 5
*G2
SEE NOTE *A3, CHART NB.
H1
POINT WRKREG8 PAST MOD LEVEL OPERAND
J1
SET OFF SW CHECKING MOD LEVEL *G2 BIT 4
K1
RESEQ OPERAND PRESENT
NO
YES
K2
SET 'RESEQ' TO X'00'
SETPTR
NB F2
INDSQ1
NB E5
REFERENCES TO NFA3; NDC2, NDD3, NDF5, NDG1 NDK4
RDDISK
A3
CLEAR BLK BYTE COUNTER REG AND SET PTR TO SYSIPT DTF TABLE
B3
RECORD SIZE = 80
YES
NO
C3
SET SW TO INDICATE REC SIZE = 81 *B2 BIT 6
LDINST
D3
SET POINTER TO LIB BLK BUILD AREA AND SAVE. CLEAR BLK BYTE COUNTER AREA
E3
POINT TO SAVE AREA FOR OLD BOOKNAME, -BKNAME-
FIND
F3
BOOK NZ
SEARCH DIR(S) FOR BOOK NAME *D2
G3
BOOK NAME FOUND
NO
YES
H3
REFERENCES TO NFH3; NDE1, NLC4 NLK2
READRTN
H3
SET WRKREG7 WITH ADDR OF -CSSTART- IN ROOT PHASE AS BASE REGISTER
J3
READ A CARD
K3
SAVE SWTCH1 AND SET POINTER TO SKIP CODE
A4
A4
ERRTN NV
SKIP A LINE OF PRINT
B4
RESTORE SWTCH1 AND SET POINTER TO SYSIPT INPUT AREA
C4
SET BUFFER POINTER FOR 80 OR 81 BYTE RECORD
BLNKSCN
D4
IS RECORD BLANK
YES
NO
H3
PRINT
E4
RTNLNK NV
PRINT THE RECORD
F4
IS IT A CONTROL STATEMENT
YES
NO
A5
G4
POINT TO MSG '3U10I' STATEMENT OUT OF SEQUENCE'
ND E1
ERCRD
BKNTFND
J4
MOVE BOOKNAME TO ERROR MSG '3M33I X.XXXXXXXX NOT IN LIBRARY'
K4
ERRTN NV
PRINT MSG ON SYSLST
NWCARD
NU B3
REFERENCES TO NFA5; NKF4, NVJ1
CTLCRD
A5
COLUMN 72 BLANK
NO
YES
A5
NU G5
NRGSEQ
B5
FIND OPERATION ADDR AND POINT WRKREG7 TO IT. POINT WRKREG8 TO COLUMN 72
C5
IS OPERATION -ADD-
YES
NO
NG A1
PROCADD
D5
IS OPERATION -REP-
YES
NO
NJ A2
PROCREP
E5
IS IT -DEL-
YES
NO
NL A2
PROCDEL
F5
IS IT -END-
YES
NO
NM A1
PROCEND
G5
POINT TO MSG '3U20I INVALID OPERATION'
NU J5
ERSRCE

Chart NG. MAINTUP - Process ADD Control Statement (Part 1 of 2)
Refer to Charts 44-46


NP C5
PROCADD
A1
SET SW PROCESSING ')ADD' CARD *A2 BIT 5
B1
PNDIGIT NW
CHECK SEQ. NO., RETURN ON NO ERROR
C1
SET OFF SW 1ST CTRL CD AFTER UPDATE CARD *A2 BIT 7
D1
ANY PREVIOUS ERRORS
YES
NV B1
PLSSTN
NO
E1
POINT WRKREG8 TO LIBRARY DIRECTORY ENTRY ADDRESS
F1
STMT > ADD 0
YES
NH G3
GETCARD
NO
G1
SEQ.NO =SEQ.NO OF LAST SRCE STMT *C2
NO
YES
H1
MOVE THE SOURCE STMNT INTO AREA -RECORD-
J1
PRNTRTN NV
PRINT IN STATEMENT ON SYSLST
B5
BRTOGET
D2
GET NS
READ A RECORD OF THE BOOK
RECSEQ
E2
'PS' SPECIFIED
NO
YES
RECSEQ1
F2
CHKINCR NR
CHECK FOR RIGHT INCR. DEFAULT
G2
COMPARE RECORD SEQ NO. TO CARD SEQ NO.
HI
EQ
NJ G1
OUTSEQ
LO
A4
LDREG
H2
LOAD ADDR OF -GET-, CHART LS, IN WRKREG8 AS EXIT FROM -COMPRES RTN
J2
*A2 RESEQUENCING TO BE DONE
NO
YES
H3
H3
H3
UPDSEQ NS
PERFORM STATEMENT RESEQUENCE
J3
MOVE NEW SEQ NO. INTO AREA -RECORD+72
LOADR
K3
LOAD ADDR OF -RECSEQ, BLOCK G2, INTO LNKREG2 FOR RETURN FROM -COMPRES RTN
COMPRES
NQ B2
A4
ADDSTMT
A4
POINT TO MAINT I/O AREAS -AREA- AND -INPUT1-
B4
MOVE RECORD TO OUTPUT AREA FOR PRINT
C4
*A2 RESEQUENCING TO BE DONE
NO
YES
D4
UPDSEQ NS
RESEQUENCE STATEMENT NUMBERS
E4
MOVE NEW SEQ NO. AND MSG -RESEQUENCED- TO OUTPUT AREA
PRTSTAT
F4
PRINT STATEMENT FROM LIBRARY VIA SKIPNV ROUTINE IN ROOT PHASE
G4
RESEQUENCING TO BE DONE
NO
YES
H4
MOVE NEW SEQ NO. INTO AREA -RECORD+72
A5
A5
BRCOMP
A5
COMPRES NQ
PUT STATEMENT IN COMPRESSED FORMAT *A3
B5
GTCRD
B5
GETCRD NT
READ A CARD
C5
IS IT A CONTROL CARD
NO
NH A1
MVRCRD
YES
D5
RTNLNK NV
PRINT THE RECORD ON SYSLST
NJ F5
CDSEQER
E5
POINT TO MSG '3U10I STATEMENT OUT OF SEQUENCE'
NW J5
ERSRCE

*A2
-SWTCH1-
SEE NOTE *A2,
CHART NB.

*A3
SEE FIGURE 28
FOR THE COMPRESSED
FORMAT.

*B2
-SWTCH-
SEE NOTE *B2,
CHART NP.

*C2
THIS ALWAYS TAKES
THE BRANCH TO BRTOGET
UNLESS A ')ADD' FOLLOWS
A ')DEL' OR ')REQ' WITH
THE SAME SEQUENCE NUMBER.

Chart NH. MAINTUP - Process ADD Control Statement (Part 2 of 2)
Refer to Charts 44-46

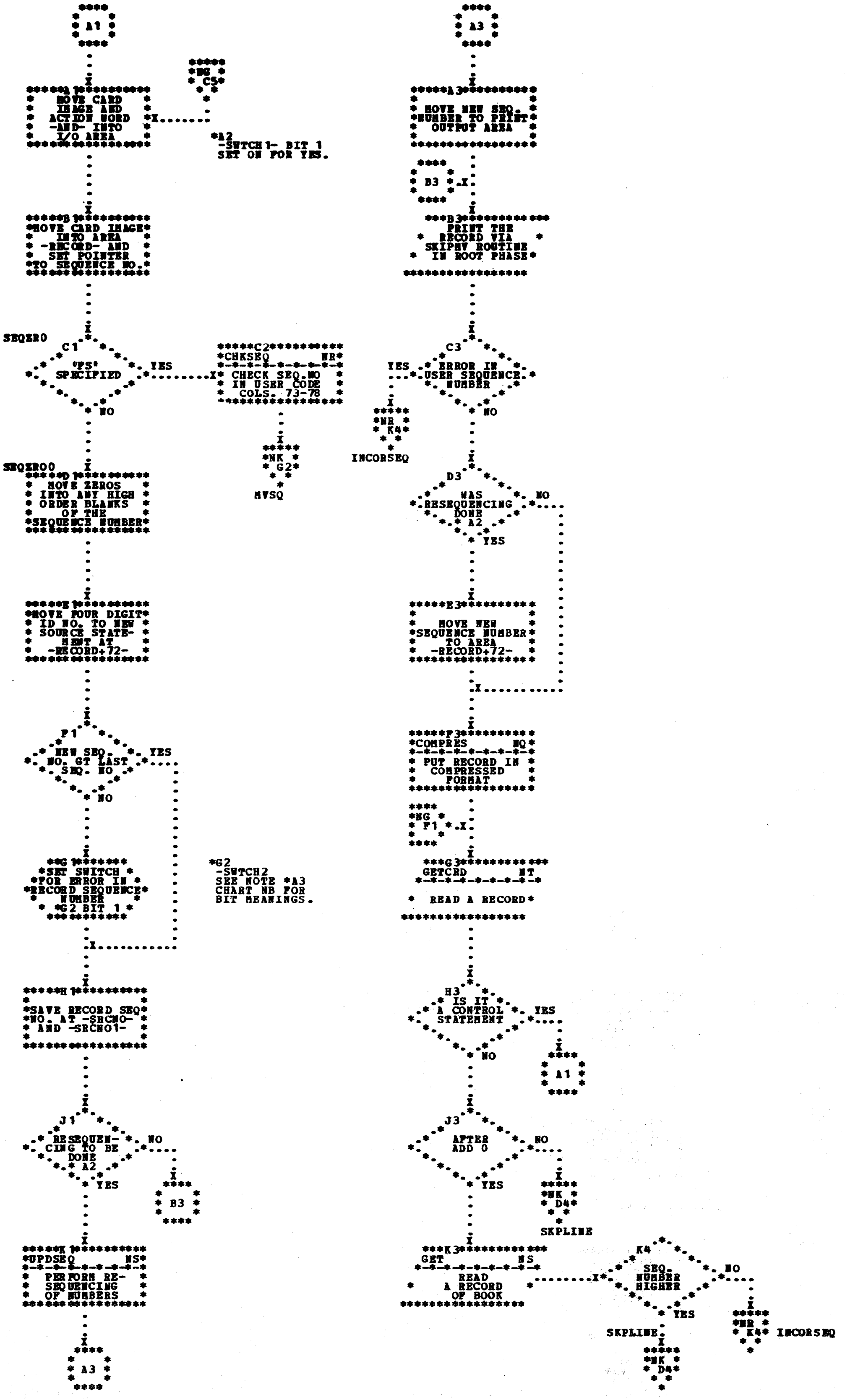

A1
NG C5
A1 MOVE CARD IMAGE AND ACTION WORD -AND- INTO I/O AREA
*A2 -SWTCH1- BIT 1 SET ON FOR YES.
B1 MOVE CARD IMAGE INTO AREA -RECORD- AND SET POINTER TO SEQUENCE NO.
SEQZRO
C1 'PS' SPECIFIED
YES
NO
C2 CHKSEQ NR
CHECK SEQ.NO IN USER CODE COLS. 73-78
NK G2
MVSQ
SEQZROO
D1 MOVE ZEROS INTO ANY HIGH ORDER BLANKS OF THE SEQUENCE NUMBER
E1 MOVE FOUR DIGIT ID NO. TO NEW SOURCE STATE- MENT AT -RECORD+72-
F1 NEW SEQ. NO. GT LAST SEQ. NO
YES
NO
G1 SET SWITCH FOR ERROR IN RECORD SEQUENCE NUMBER *G2 BIT 1
*G2 -SWTCH2 SEE NOTE *A3 CHART NB FOR BIT MEANINGS.
H1 SAVE RECORD SEQ NO. AT -SRCNO- AND -SRCNO1-
J1 RESEQUEN- CING TO BE DONE *A2
NO
YES
B3
K1 UPDSEQ NS
PERFORM RE- SEQUENCING OF NUMBERS
A3
A3
A3 MOVE NEW SEQ. NUMBER TO PRINT OUTPUT AREA
B3
B3 PRINT THE RECORD VIA SKIPNV ROUTINE IN ROOT PHASE
C3 ERROR IN USER SEQUENCE NUMBER
YES
NO
NR K4
INCORSEQ
D3 WAS RESEQUENCING DONE *A2
NO
YES
E3 MOVE NEW SEQUENCE NUMBER TO AREA -RECORD+72-
F3 COMPRES NQ
PUT RECORD IN COMPRESSED FORMAT
NG F1
G3 GETCRD NT
READ A RECORD
H3 IS IT A CONTROL STATEMENT
YES
NO
A1
J3 AFTER ADD 0
NO
YES
NK D4
SKPLINE
K3 GET NS
READ A RECORD OF BOOK
K4 SEQ. NUMBER HIGHER
NO
YES
SKPLINE
NK D4
NR K4 INCORSEQ

Chart NJ. MAINTUP - Proceee REP Control Statement (Part 1 of 2)
Refer to Charts 44-46

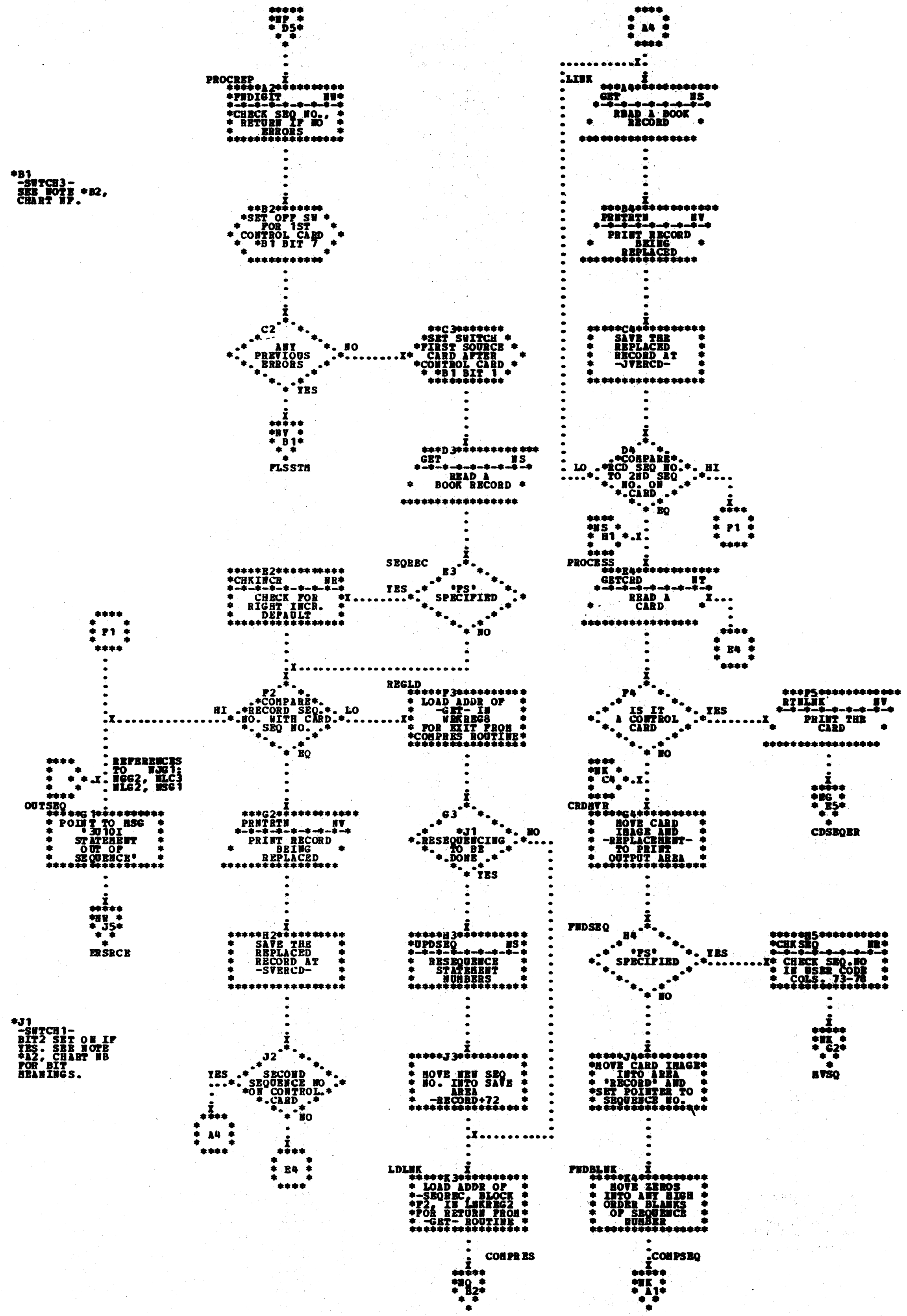
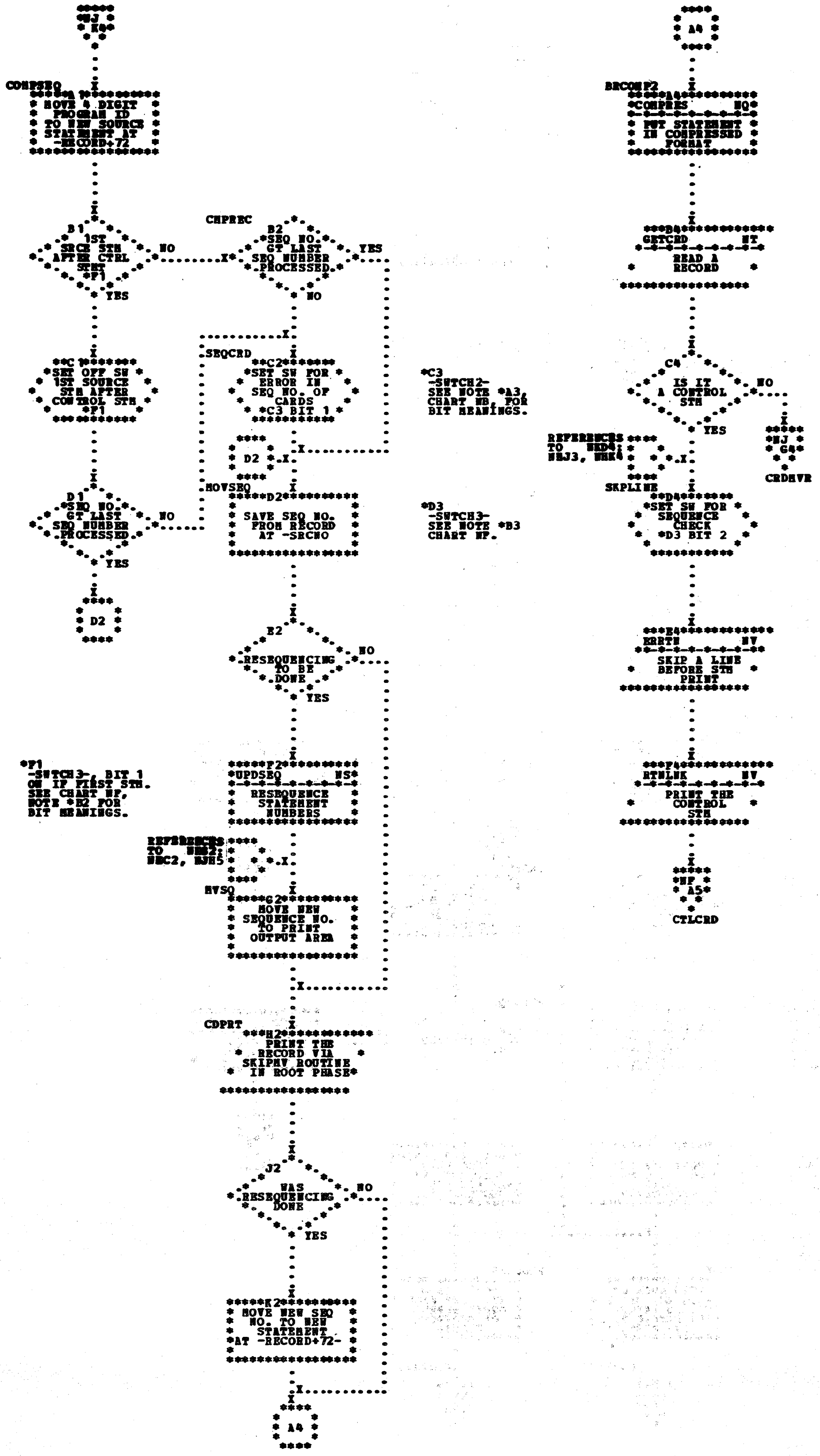

NP D5
PROCREP
A2 PNDIGIT NW
CHECK SEQ NO., RETURN IF NO ERRORS
*B1 -SWTCH3- SEE NOTE *B2, CHART NP.
B2 SET OFF SW FOR 1ST CONTROL CARD *B1 BIT 7
C2 ANY PREVIOUS ERRORS
NO
YES
NV B1
FLSSTM
C3 SET SWITCH FIRST SOURCE CARD AFTER CONTROL CARD *B1 BIT 1
D3 GET NS
READ A BOOK RECORD
SEQREC
E3 'PS' SPECIFIED
YES
NO
E2 CHKINCR NR
CHECK FOR RIGHT INCR. DEFAULT
F1
F2 COMPARE RECORD SEQ. NO. WITH CARD SEQ NO.
HI
LO
EQ
REFERENCES TO NJG1; NGG2, NLC3 NLG2, NSG1
REGLD
F3 LOAD ADDR OF -GET- IN WRKREG8 FOR EXIT FROM COMPRES ROUTINE
OUTSEQ
G1 POINT TO MSG '3U10I STATEMENT OUT OF SEQUENCE'
NW J5
ERSRCE
G2 PRNTRTN NV
PRINT RECORD BEING REPLACED
G3 *J1 RESEQUENCING TO BE DONE
NO
YES
H2 SAVE THE REPLACED RECORD AT -SVERCD-
H3 UPDSEQ NS
RESEQUENCE STATEMENT NUMBERS
*J1 -SWTCH1- BIT2 SET ON IF YES. SEE NOTE *A2, CHART NB FOR BIT MEANINGS.
J2 SECOND SEQUENCE NO ON CONTROL CARD
YES
NO
A4
E4
J3 MOVE NEW SEQ NO. INTO SAVE AREA -RECORD+72
LDLNK
K3 LOAD ADDR OF -SEQREC, BLOCK F2, IN LNKREG2 FOR RETURN FROM -GET- ROUTINE
COMPRES
NQ B2
A4
LINK
A4 GET NS
READ A BOOK RECORD
B4 PRNTRTN NV
PRINT RECORD BEING REPLACED
C4 SAVE THE REPLACED RECORD AT -JVERCD-
D4 COMPARE RCD SEQ NO. TO 2ND SEQ NO. ON CARD
LO
HI
EQ
NS H1
F1
PROCESS
E4 GETCRD NT
READ A CARD
E4
F4 IS IT A CONTROL CARD
YES
NO
F5 RTNLNK NV
PRINT THE CARD
NG E5
CDSEQER
NK C4
CRDMVR
G4 MOVE CARD IMAGE AND -REPLACEMENT- TO PRINT OUTPUT AREA
FNDSEQ
H4 'PS' SPECIFIED
YES
NO
H5 CHKSEQ NR
CHECK SEQ.NO IN USER CODE COLS. 73-78
NK G2
NVSQ
J4 MOVE CARD IMAGE INTO AREA 'RECORD' AND SET POINTER TO SEQUENCE NO.
FNDBLNK
K4 MOVE ZEROS INTO ANY HIGH ORDER BLANKS OF SEQUENCE NUMBER
COMPSEQ
NK A1

Chart NK. MAINTUP - Process REP Control Statement (Part 2 of 2)
Refer to Charts 44-46


NJ
K6
COMPSEQ
A1
MOVE 4 DIGIT PROGRAM ID TO NEW SOURCE STATEMENT AT -RECORD+72
B1
1ST SRCE STM AFTER CTRL STMT *F1
NO
YES
C1
SET OFF SW 1ST SOURCE STM AFTER CONTROL STM *F1
D1
SEQ NO. GT LAST SEQ NUMBER PROCESSED
NO
YES
D2
CHPREC
B2
SEQ NO. GT LAST SEQ NUMBER PROCESSED
YES
NO
SEQCRD
C2
SET SW FOR ERROR IN SEQ NO. OF CARDS *C3 BIT 1
D2
MOVSEQ
D2
SAVE SEQ NO. FROM RECORD AT -SRCNO
E2
RESEQUENCING TO BE DONE
NO
YES
F2
UPDSEQ NS
RESEQUENCE STATEMENT NUMBERS
REFERENCES TO NNG2: NNC2, NJH5
MVSQ
G2
MOVE NEW SEQUENCE NO. TO PRINT OUTPUT AREA
CDPRT
H2
PRINT THE RECORD VIA SKIPMV ROUTINE IN ROOT PHASE
J2
WAS RESEQUENCING DONE
NO
YES
K2
MOVE NEW SEQ NO. TO NEW STATEMENT AT -RECORD+72-
A4
A4
BRCOMP2
A4
COMPRES NQ
PUT STATEMENT IN COMPRESSED FORMAT
B4
GETCRD NT
READ A RECORD
C4
IS IT A CONTROL STM
NO
YES
NJ
G4
CRDMVR
REFERENCES TO NKD4: NEJ3, NHK4
SKPLINE
D4
SET SW FOR SEQUENCE CHECK *D3 BIT 2
E4
ERRTN NV
SKIP A LINE BEFORE STM PRINT
F4
RTNLNK NV
PRINT THE CONTROL STM
NP
A5
CTLCRD

*C3
-SWTCH2-
SEE NOTE *A3, CHART NB, FOR BIT MEANINGS.

*D3
-SWTCH3-
SEE NOTE *B3 CHART NP.

*F1
-SWTCH3-, BIT 1 ON IF FIRST STM. SEE CHART NP, NOTE *E2 FOR BIT MEANINGS.

Chart NL. MAINTUP - Process DEL Control Statement
Refer to Charts 44-46

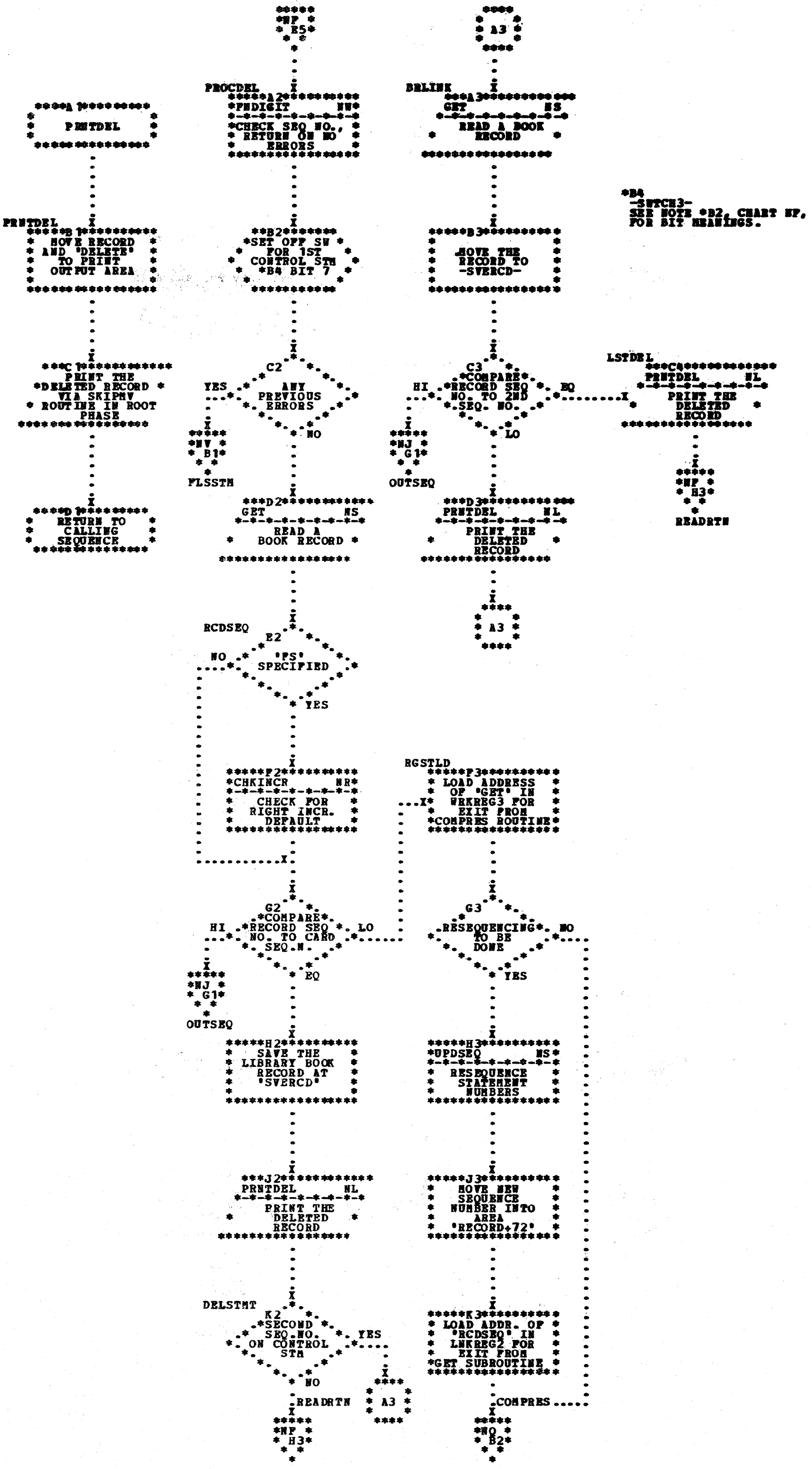

NF B5
A3
PROCDEL
A2
PNDIGIT NW
CHECK SEQ NO., RETURN ON NO ERRORS
BRLINK
A3
GET NS
READ A BOOK RECORD
A1
PRNTDEL
*B4
-SWTCH3-
SEE NOTE *B2, CHART NF, FOR BIT MEANINGS.
PRNTDEL
B1
MOVE RECORD AND 'DELETE' TO PRINT OUTPUT AREA
B2
SET OFF SW FOR 1ST CONTROL STM *B4 BIT 7
B3
MOVE THE RECORD TO -SVERCD-
C1
PRINT THE DELETED RECORD VIA SKIPMV ROUTINE IN ROOT PHASE
C2
ANY PREVIOUS ERRORS
YES
NO
NV B1
FLSSTM
C3
COMPARE RECORD SEQ NO. TO 2ND SEQ. NO.
HI
EQ
LO
NJ G1
OUTSEQ
LSTDEL
C4
PRNTDEL NL
PRINT THE DELETED RECORD
NF H3
READRTN
D1
RETURN TO CALLING SEQUENCE
D2
GET NS
READ A BOOK RECORD
D3
PRNTDEL NL
PRINT THE DELETED RECORD
A3
RCDSEQ
E2
'PS' SPECIFIED
NO
YES
F2
CHKINCR NR
CHECK FOR RIGHT INCR. DEFAULT
RGSTLD
F3
LOAD ADDRESS OF 'GET' IN WRKREG3 FOR EXIT FROM COMPRES ROUTINE
G2
COMPARE RECORD SEQ NO. TO CARD SEQ.N.
HI
LO
EQ
NJ G1
OUTSEQ
G3
RESEQUENCING TO BE DONE
NO
YES
H2
SAVE THE LIBRARY BOOK RECORD AT 'SVERCD'
H3
UPDSEQ NS
RESEQUENCE STATEMENT NUMBERS
J2
PRNTDEL NL
PRINT THE DELETED RECORD
J3
MOVE NEW SEQUENCE NUMBER INTO AREA 'RECORD+72'
DELSTMT
K2
SECOND SEQ.NO. ON CONTROL STM
YES
NO
K3
LOAD ADDR. OF 'RCDSEQ' IN LNKREG2 FOR EXIT FROM GET SUBROUTINE
READRTN
A3
COMPRES
NF H3
NQ B2

Chart NM. MAINTUP - Process END Control Statement (Part 1 of 3)
Refer to Charts 44-46

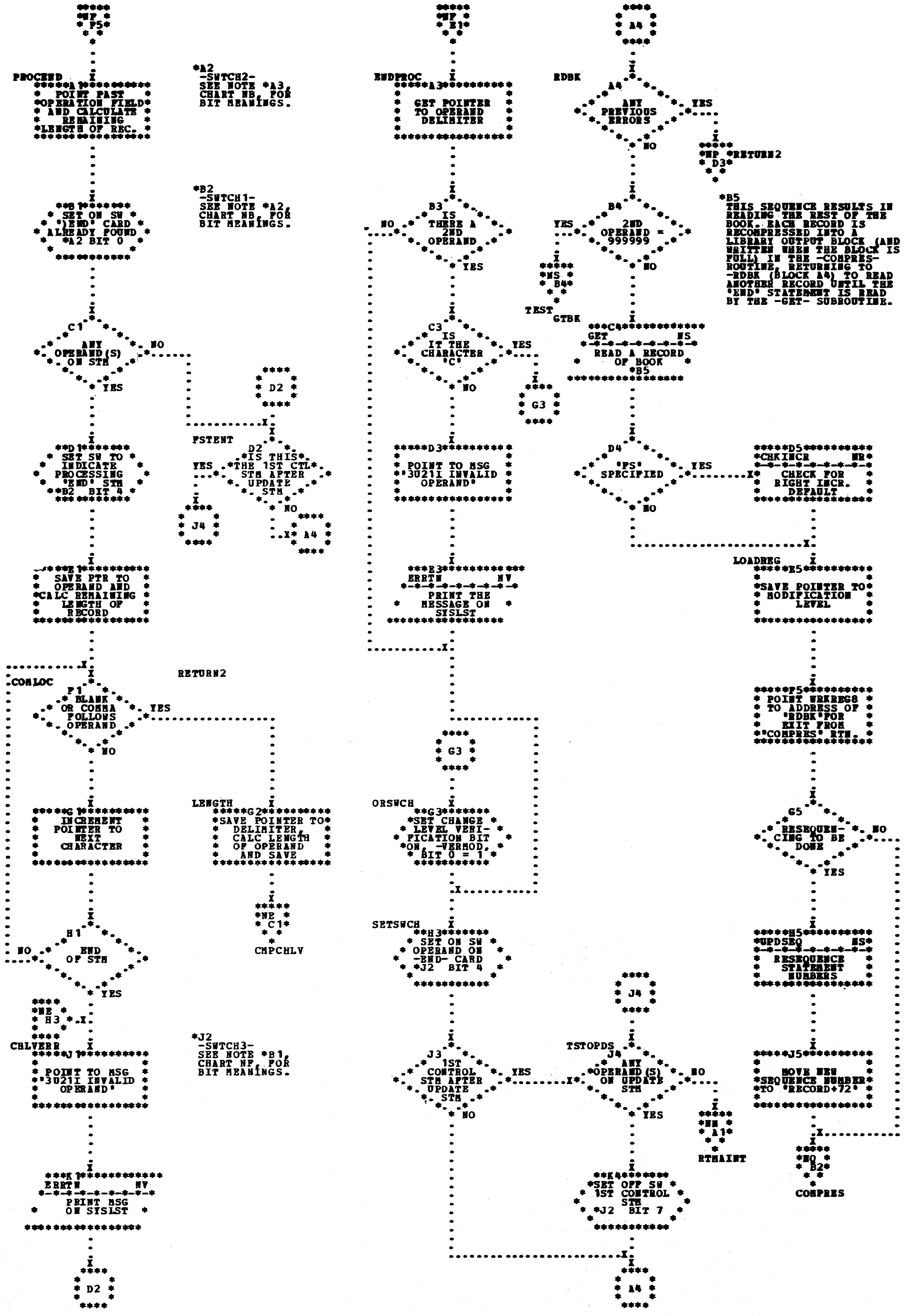

NP F5
PROCEND
A1 POINT PAST OPERATION FIELD AND CALCULATE REMAINING LENGTH OF REC.
*A2 -SWTCH2- SEE NOTE *A3, CHART NB, FOR BIT MEANINGS.
B1 SET ON SW ')END' CARD ALREADY FOUND *A2 BIT 0
*B2 -SWTCH1- SEE NOTE *A2, CHART NB, FOR BIT MEANINGS.
C1 ANY OPERAND(S) ON STM
NO
YES
D2
FSTENT
D2 IS THIS THE 1ST CTL STM AFTER UPDATE STM
YES
NO
J4
A4
D1 SET SW TO INDICATE PROCESSING 'END' STM *B2 BIT 4
E1 SAVE PTR TO OPERAND AND CALC REMAINING LENGTH OF RECORD
COMLOC
RETURN2
F1 BLANK OR COMMA FOLLOWS OPERAND
YES
NO
G1 INCREMENT POINTER TO NEXT CHARACTER
LENGTH
G2 SAVE POINTER TO DELIMITER, CALC LENGTH OF OPERAND AND SAVE
NE C1
CMPCHLV
H1 END OF STM
NO
YES
NE H3
CHLVERR
J1 POINT TO MSG '3U21I INVALID OPERAND'
*J2 -SWTCH3- SEE NOTE *B1, CHART NF, FOR BIT MEANINGS.
K1 ERRTN NV PRINT MSG ON SYSLST
D2
NF B1
ENDPROC
A3 GET POINTER TO OPERAND DELIMITER
B3 IS THERE A 2ND OPERAND
NO
YES
C3 IS IT THE CHARACTER 'C'
YES
NO
G3
D3 POINT TO MSG '3U21I INVALID OPERAND'
E3 ERRTN NV PRINT THE MESSAGE ON SYSLST
G3
ORSWCH
G3 SET CHANGE LEVEL VERIFICATION BIT ON, -VERMOD, BIT 0 = 1
SETSWCH
H3 SET ON SW OPERAND ON -END- CARD *J2 BIT 4
J3 1ST CONTROL STM AFTER UPDATE STM
YES
NO
A4
RDBK
A4 ANY PREVIOUS ERRORS
YES
NO
NP D3 RETURN2
B4 2ND OPERAND = 999999
YES
NO
NS B4
TEST
GTBK
C4 GET NS READ A RECORD OF BOOK *B5
*B5 THIS SEQUENCE RESULTS IN READING THE REST OF THE BOOK. EACH RECORD IS RECOMPRESSED INTO A LIBRARY OUTPUT BLOCK (AND WRITTEN WHEN THE BLOCK IS FULL) IN THE -COMPRES- ROUTINE, RETURNING TO -RDBK (BLOCK A4) TO READ ANOTHER RECORD UNTIL THE 'END' STATEMENT IS READ BY THE -GET- SUBROUTINE.
D4 'PS' SPECIFIED
YES
NO
D5 CHKINCR NR CHECK FOR RIGHT INCR. DEFAULT
LOADREG
E5 SAVE POINTER TO MODIFICATION LEVEL
F5 POINT WRKREG8 TO ADDRESS OF 'RDBK' FOR EXIT FROM 'COMPRES' RTN.
G5 RESEQUENCING TO BE DONE
NO
YES
H5 UPDSEQ NS RESEQUENCE STATEMENT NUMBERS
J4
TSTOPDS
J4 ANY OPERAND(S) ON UPDATE STM
NO
YES
NN A1
RTMAINT
K4 SET OFF SW 1ST CONTROL STM *J2 BIT 7
A4
J5 MOVE NEW SEQUENCE NUMBER TO 'RECORD+72'
NO B2
COMPRES

Chart NN. MAINTUP - Process END Control Statement (Part 2 of 3)
Refer to Charts 44-46

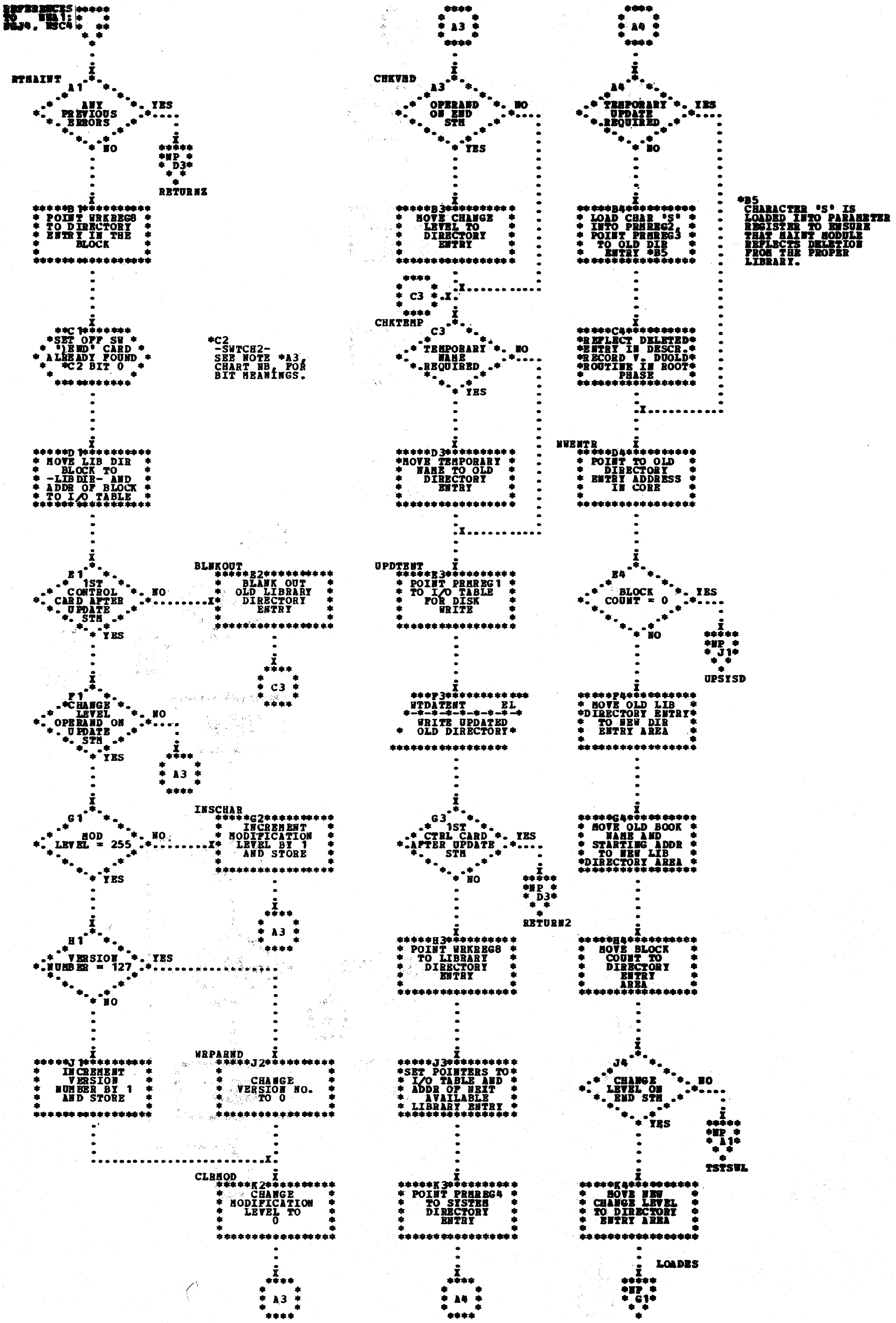
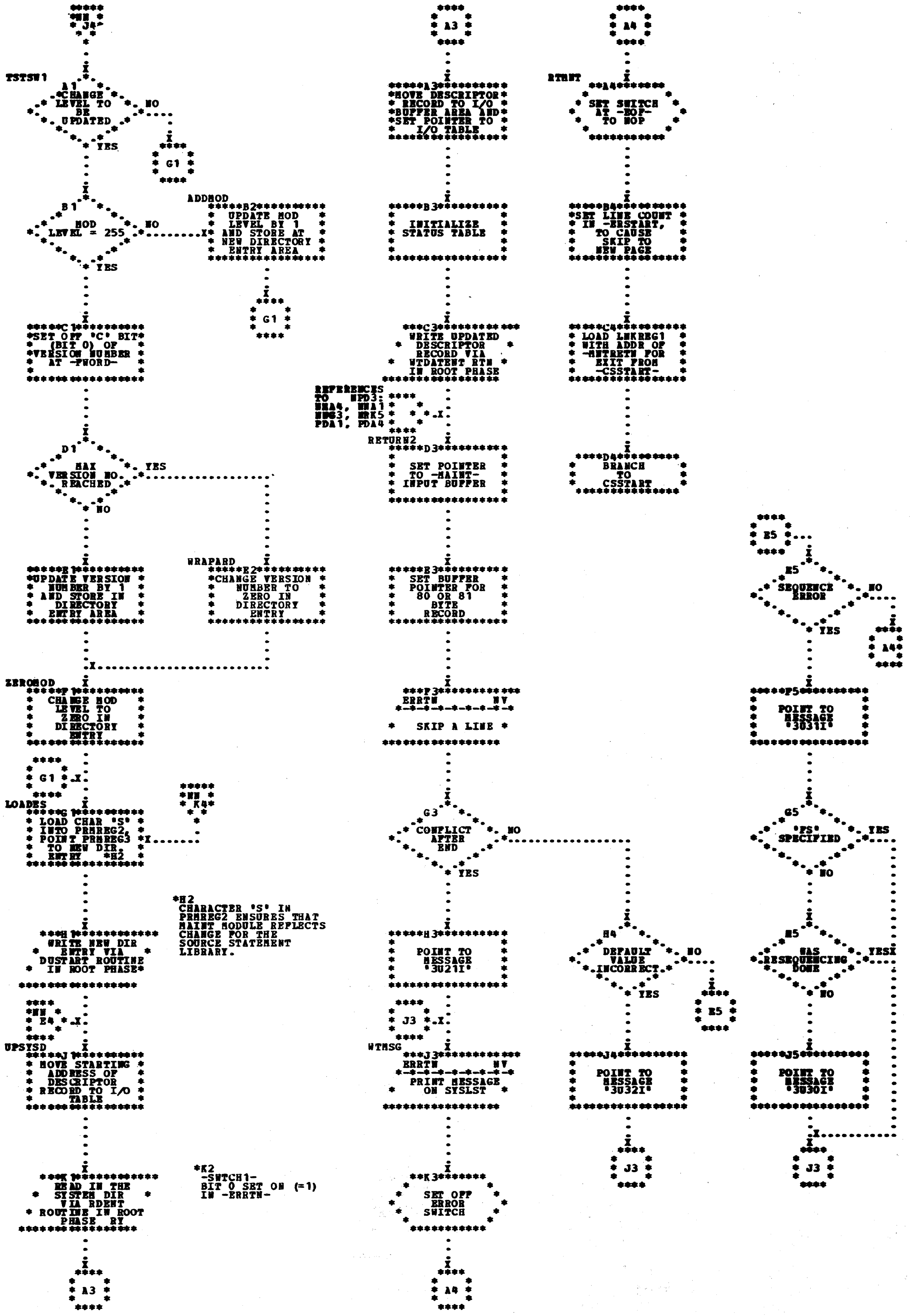

REFERENCES TO NNA1: NNJ4, NNC4
RTNAINT
A1
ANY PREVIOUS ERRORS
YES
NO
NP D3
RETURNZ
B1
POINT WRKREG8 TO DIRECTORY ENTRY IN THE BLOCK
C1
SET OFF SW 'END' CARD ALREADY FOUND C2 BIT 0
*C2 -SWTCH2- SEE NOTE *A3, CHART NB, FOR BIT MEANINGS.
D1
MOVE LIB DIR BLOCK TO -LIBDIR- AND ADDR OF BLOCK TO I/O TABLE
E1
1ST CONTROL CARD AFTER UPDATE STM
NO
YES
BLNKOUT
E2
BLANK OUT OLD LIBRARY DIRECTORY ENTRY
C3
F1
CHANGE LEVEL OPERAND ON UPDATE STM
NO
YES
A3
G1
MOD LEVEL = 255
NO
YES
INSCHAR
G2
INCREMENT MODIFICATION LEVEL BY 1 AND STORE
A3
H1
VERSION NUMBER = 127
YES
NO
J1
INCREMENT VERSION NUMBER BY 1 AND STORE
WRPARND
J2
CHANGE VERSION NO. TO 0
CLRMOD
K2
CHANGE MODIFICATION LEVEL TO 0
A3
A3
CHKVND
A3
OPERAND ON END STM
NO
YES
B3
MOVE CHANGE LEVEL TO DIRECTORY ENTRY
C3
CHKTEMP
C3
TEMPORARY NAME REQUIRED
NO
YES
D3
MOVE TEMPORARY NAME TO OLD DIRECTORY ENTRY
UPDTENT
E3
POINT PRMREG1 TO I/O TABLE FOR DISK WRITE
F3
WTDATENT EL
WRITE UPDATED OLD DIRECTORY
G3
1ST CTRL CARD AFTER UPDATE STM
YES
NO
NP D3
RETURN2
H3
POINT WRKREG8 TO LIBRARY DIRECTORY ENTRY
J3
SET POINTERS TO I/O TABLE AND ADDR OF NEXT AVAILABLE LIBRARY ENTRY
K3
POINT PRMREG4 TO SYSTEM DIRECTORY ENTRY
A4
A4
A4
TEMPORARY UPDATE REQUIRED
YES
NO
B4
LOAD CHAR 'S' INTO PRMREG2, POINT PRMREG3 TO OLD DIR ENTRY *B5
*B5 CHARACTER 'S' IS LOADED INTO PARAMETER REGISTER TO ENSURE THAT MAINT MODULE REFLECTS DELETION FROM THE PROPER LIBRARY.
C4
REFLECT DELETED ENTRY IN DESCR. RECORD V. DUOLD ROUTINE IN ROOT PHASE
NWENTR
D4
POINT TO OLD DIRECTORY ENTRY ADDRESS IN CORE
E4
BLOCK COUNT = 0
YES
NO
NP J1
UPSYSD
F4
MOVE OLD LIB DIRECTORY ENTRY TO NEW DIR ENTRY AREA
G4
MOVE OLD BOOK NAME AND STARTING ADDR TO NEW LIB DIRECTORY AREA
H4
MOVE BLOCK COUNT TO DIRECTORY ENTRY AREA
J4
CHANGE LEVEL ON END STM
NO
YES
NP A1
TSTSUL
K4
MOVE NEW CHANGE LEVEL TO DIRECTORY ENTRY AREA
LOADES
NP G1

Chart NP. MAINTUP - Process END Control Statement (Part 3 of 3)
Refer to Charts 44-46



NN J4
TSTSW1
A1
CHANGE LEVEL TO BE UPDATED
NO
YES
G1
B1
MOD LEVEL = 255
NO
YES
ADDMOD
B2
UPDATE MOD LEVEL BY 1 AND STORE AT NEW DIRECTORY ENTRY AREA
G1
C1
SET OFF 'C' BIT (BIT 0) OF VERSION NUMBER AT -FWORD-
D1
MAX VERSION NO. REACHED
YES
NO
E1
UPDATE VERSION NUMBER BY 1 AND STORE IN DIRECTORY ENTRY AREA
WRAPARD
E2
CHANGE VERSION NUMBER TO ZERO IN DIRECTORY ENTRY
ZEROMOD
F1
CHANGE MOD LEVEL TO ZERO IN DIRECTORY ENTRY
G1
LOADES
G1
LOAD CHAR 'S' INTO PRMREG2, POINT PRMREG3 TO NEW DIR. ENTRY *H2
NN K4
*H2
CHARACTER 'S' IN PRMREG2 ENSURES THAT MAINT MODULE REFLECTS CHANGE FOR THE SOURCE STATEMENT LIBRARY.
H1
WRITE NEW DIR ENTRY VIA DUSTART ROUTINE IN ROOT PHASE
NN E4
UPSYSD
J1
MOVE STARTING ADDRESS OF DESCRIPTOR RECORD TO I/O TABLE
K1
READ IN THE SYSTEM DIR VIA RDENT ROUTINE IN ROOT PHASE RY
*K2
-SWTCH1-
BIT 0 SET ON (=1)
IN -ERRTN-
A3
A3
A3
MOVE DESCRIPTOR RECORD TO I/O BUFFER AREA AND SET POINTER TO I/O TABLE
B3
INITIALIZE STATUS TABLE
C3
WRITE UPDATED DESCRIPTOR RECORD VIA WTDATENT RTN IN ROOT PHASE
REFERENCES TO NPD3: NHA4, NNA1 NNG3, NRK5 PDA1, PDA4
RETURN2
D3
SET POINTER TO -MAINT- INPUT BUFFER
E3
SET BUFFER POINTER FOR 80 OR 81 BYTE RECORD
F3
ERRTN NV
SKIP A LINE
G3
CONFLICT AFTER END
NO
YES
H3
POINT TO MESSAGE '3U21I'
J3
WTMSG
J3
ERRTN NV
PRINT MESSAGE ON SYSLST
K3
SET OFF ERROR SWITCH
A4
A4
RTNNT
A4
SET SWITCH AT -EOP- TO NOP
B4
SET LINE COUNT IN -ERSTART, TO CAUSE SKIP TO NEW PAGE
C4
LOAD LNKREG1 WITH ADDR OF -MNTRETN FOR EXIT FROM -CSSTART-
D4
BRANCH TO CSSTART
H4
DEFAULT VALUE INCORRECT
NO
YES
E5
J4
POINT TO MESSAGE '3U32I'
J3
E5
E5
SEQUENCE ERROR
NO
YES
A4
F5
POINT TO MESSAGE '3U31I'
G5
'FS' SPECIFIED
YES
NO
H5
WAS RESEQUENCING DONE
YES
NO
J5
POINT TO MESSAGE '3U30I'
J3

Chart NQ. MAINTUP - Subroutines (Part 1 of 6)
Refer to Charts 44-46

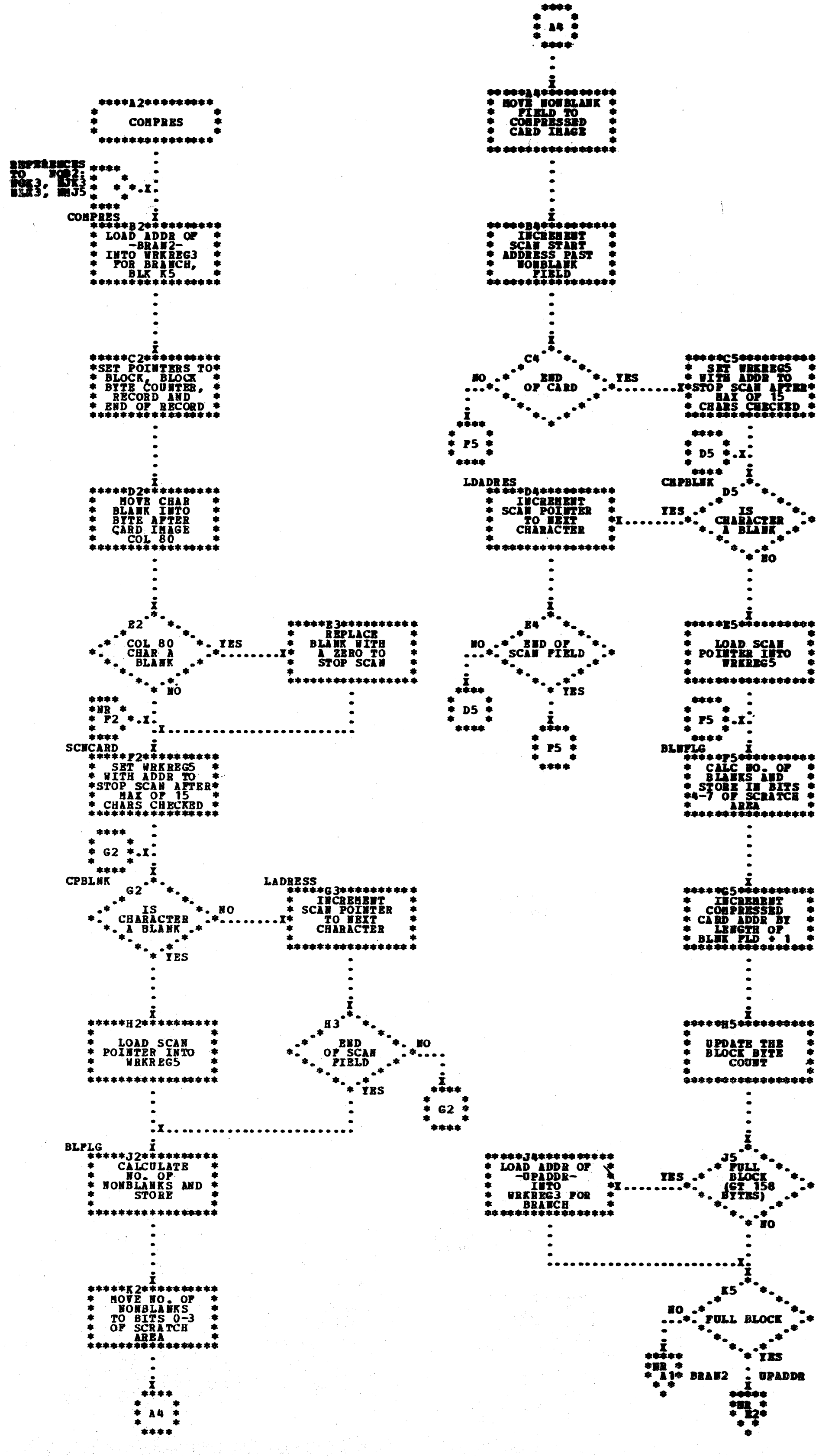

A2
COMPRES
REFERENCES TO NQB2: NQK3, BLK3 BLK3, NMJ5
COMPRES
B2
LOAD ADDR OF -BRAN2- INTO WRKREG3 FOR BRANCH, BLK K5
C2
SET POINTERS TO BLOCK, BLOCK BYTE COUNTER, RECORD AND END OF RECORD
D2
MOVE CHAR BLANK INTO BYTE AFTER CARD IMAGE COL 80
E2
COL 80 CHAR A BLANK
YES
NO
E3
REPLACE BLANK WITH A ZERO TO STOP SCAN
NR F2
SCNCARD
F2
SET WRKREG5 WITH ADDR TO STOP SCAN AFTER MAX OF 15 CHARS CHECKED
G2
CPBLNK
G2
IS CHARACTER A BLANK
NO
YES
LADRESS
G3
INCREMENT SCAN POINTER TO NEXT CHARACTER
H2
LOAD SCAN POINTER INTO WRKREG5
H3
END OF SCAN FIELD
NO
YES
G2
BLFLG
J2
CALCULATE NO. OF NONBLANKS AND STORE
K2
MOVE NO. OF NONBLANKS TO BITS 0-3 OF SCRATCH AREA
A4
A4
A4
MOVE NONBLANK FIELD TO COMPRESSED CARD IMAGE
B4
INCREMENT SCAN START ADDRESS PAST NONBLANK FIELD
C4
END OF CARD
NO
YES
F5
C5
SET WRKREG5 WITH ADDR TO STOP SCAN AFTER MAX OF 15 CHARS CHECKED
D5
CBPBLNK
D5
IS CHARACTER A BLANK
YES
NO
LDADRES
D4
INCREMENT SCAN POINTER TO NEXT CHARACTER
E4
END OF SCAN FIELD
NO
YES
D5
F5
E5
LOAD SCAN POINTER INTO WRKREG5
F5
BLNFLG
F5
CALC NO. OF BLANKS AND STORE IN BITS 4-7 OF SCRATCH AREA
G5
INCREMENT COMPRESSED CARD ADDR BY LENGTH OF BLNK FLD + 1
H5
UPDATE THE BLOCK BYTE COUNT
J5
FULL BLOCK (GT 158 BYTES)
YES
NO
J4
LOAD ADDR OF -UPADDR- INTO WRKREG3 FOR BRANCH
K5
FULL BLOCK
NO
YES
NR A1
BRAN2
UPADDR
NR K2

Chart NR. MAINTUP - Subroutines (Part 2 of 6)
Refer to Charts 44-46

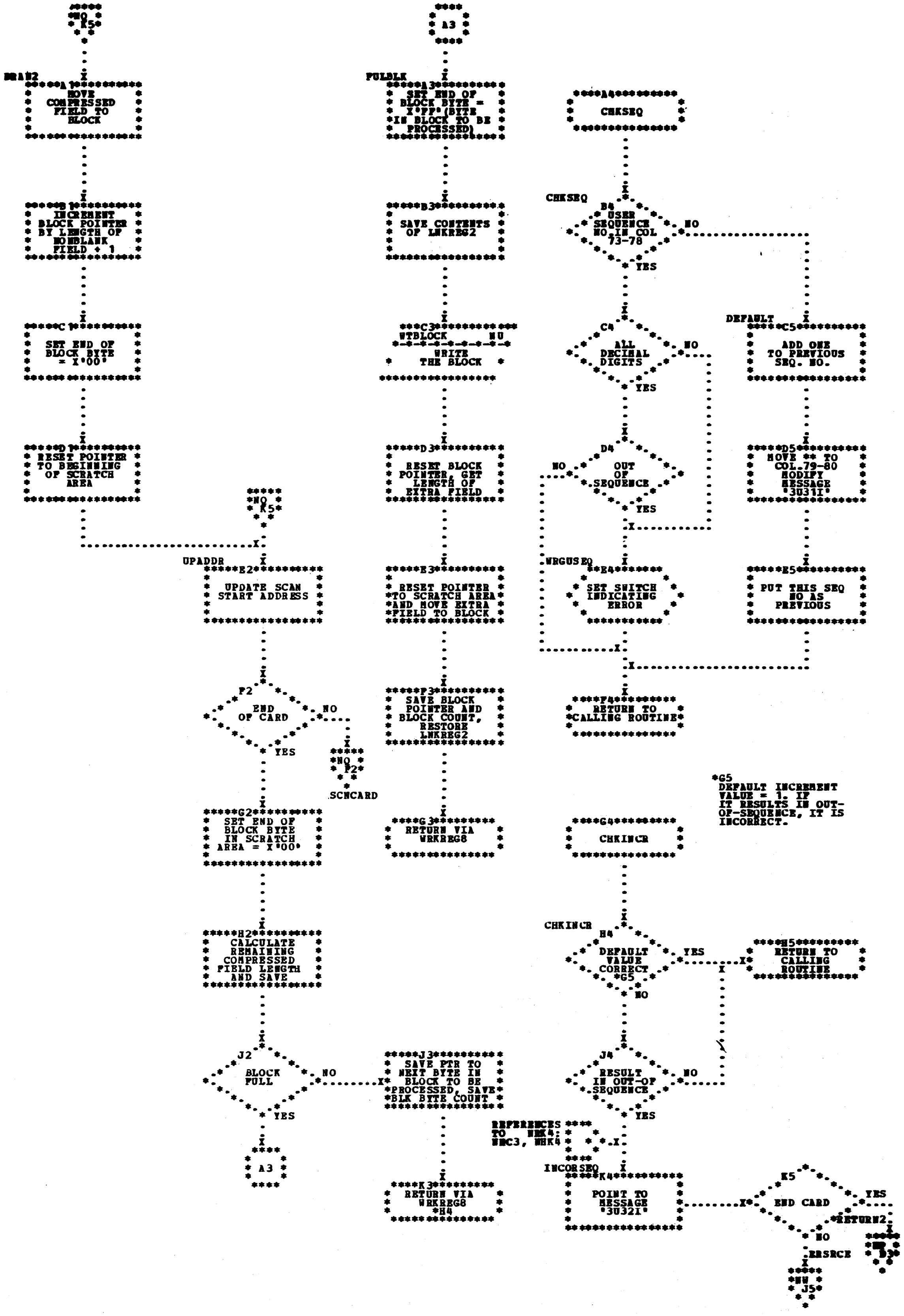

NQ K5
BRAN2
A1 MOVE COMPRESSED FIELD TO BLOCK
B1 INCREMENT BLOCK POINTER BY LENGTH OF NONBLANK FIELD + 1
C1 SET END OF BLOCK BYTE = X'00'
D1 RESET POINTER TO BEGINNING OF SCRATCH AREA
NQ K5
UPADDR
E2 UPDATE SCAN START ADDRESS
F2 END OF CARD
NO
YES
NQ F2
SCNCARD
G2 SET END OF BLOCK BYTE IN SCRATCH AREA = X'00'
H2 CALCULATE REMAINING COMPRESSED FIELD LENGTH AND SAVE
J2 BLOCK FULL
NO
YES
A3
A3
FULBLK
A3 SET END OF BLOCK BYTE = X'FF' (BYTE IN BLOCK TO BE PROCESSED)
B3 SAVE CONTENTS OF LNKREG2
C3 WTBLOCK NU
WRITE THE BLOCK
D3 RESET BLOCK POINTER, GET LENGTH OF EXTRA FIELD
E3 RESET POINTER TO SCRATCH AREA AND MOVE EXTRA FIELD TO BLOCK
F3 SAVE BLOCK POINTER AND BLOCK COUNT, RESTORE LNKREG2
G3 RETURN VIA WRKREG8
J3 SAVE PTR TO NEXT BYTE IN BLOCK TO BE PROCESSED, SAVE BLK BYTE COUNT
K3 RETURN VIA WRKREG8 *H4
A4 CHKSEQ
CHKSEQ
B4 USER SEQUENCE NO.IN COL 73-78
NO
YES
C4 ALL DECIMAL DIGITS
NO
YES
D4 OUT OF SEQUENCE
NO
YES
WRGUSEQ
E4 SET SWITCH INDICATING ERROR
F4 RETURN TO CALLING ROUTINE
DEFAULT
C5 ADD ONE TO PREVIOUS SEQ. NO.
D5 MOVE ** TO COL.79-80 MODIFY MESSAGE '3U31I'
E5 PUT THIS SEQ NO AS PREVIOUS
*G5
DEFAULT INCREMENT VALUE = 1. IF IT RESULTS IN OUT-OF-SEQUENCE, IT IS INCORRECT.
G4 CHKINCR
CHKINCR
H4 DEFAULT VALUE CORRECT *G5
YES
NO
H5 RETURN TO CALLING ROUTINE
J4 RESULT IN OUT-OF SEQUENCE
NO
YES
REFERENCES TO NNK4: NNC3, NHK4
INCORSEQ
K4 POINT TO MESSAGE '3U32I'
K5 END CARD
YES
NO
RETURN2
NQ D3
ERSRCE
NU J5

Chart NS. MAINTUP - Subroutines (Part 3 of 6)
Refer to Charts 44-46

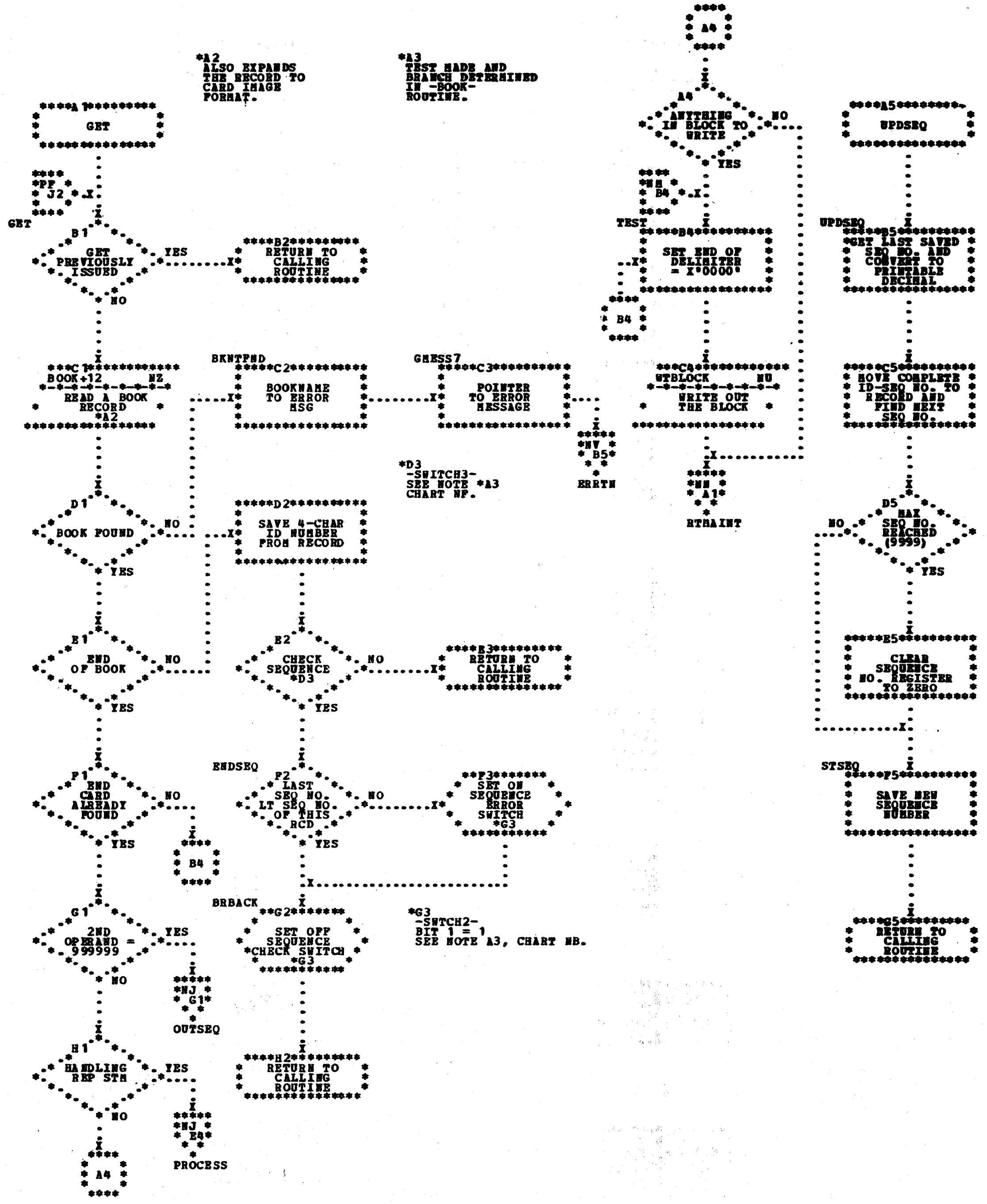


Chart NT. MAINTUP - Subroutines (Part 4 of 6)
Refer to Charts 44-46

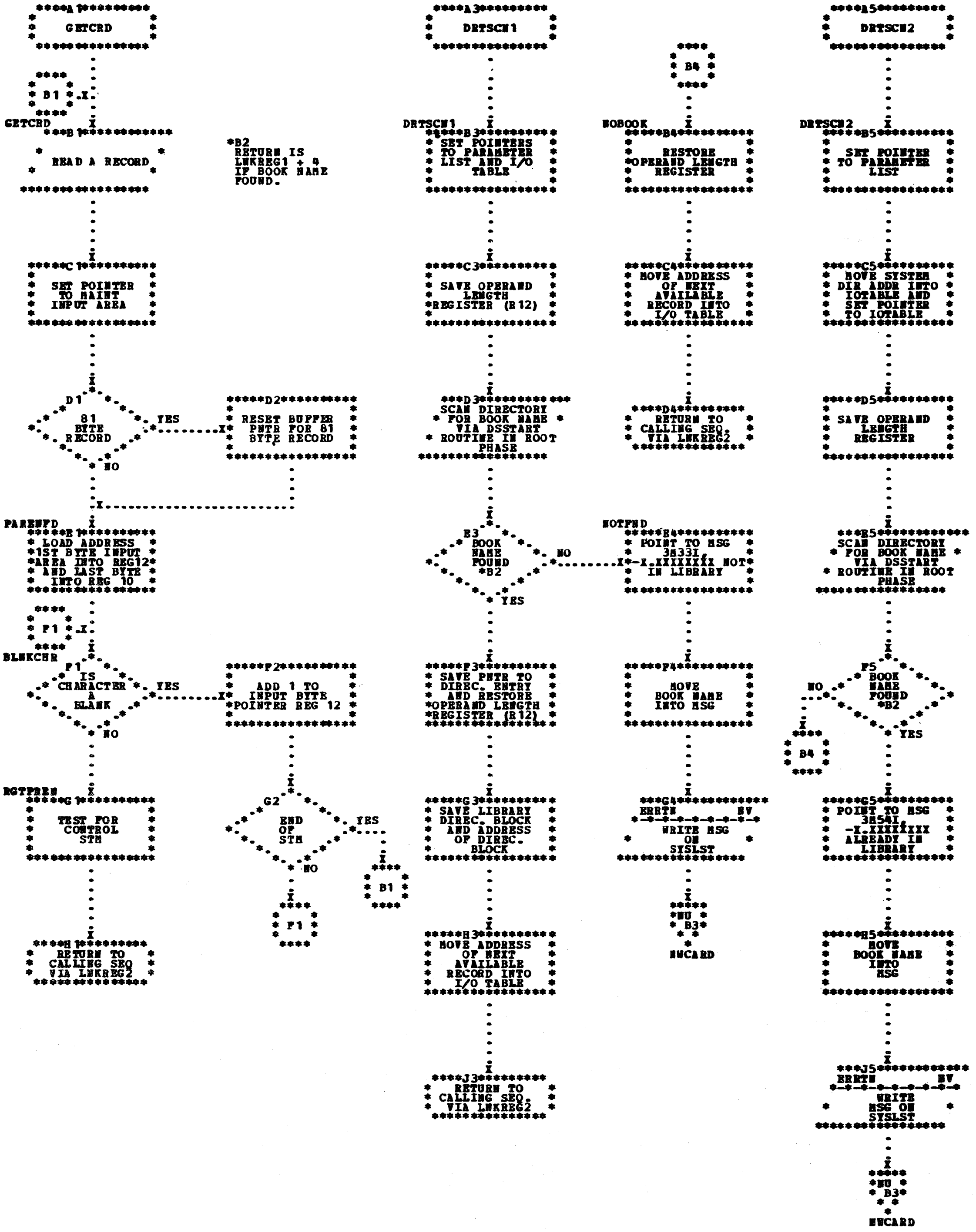


Chart NU. MAINTUP - Subroutines (Part 5 of 6)
Refer to Charts 44-46

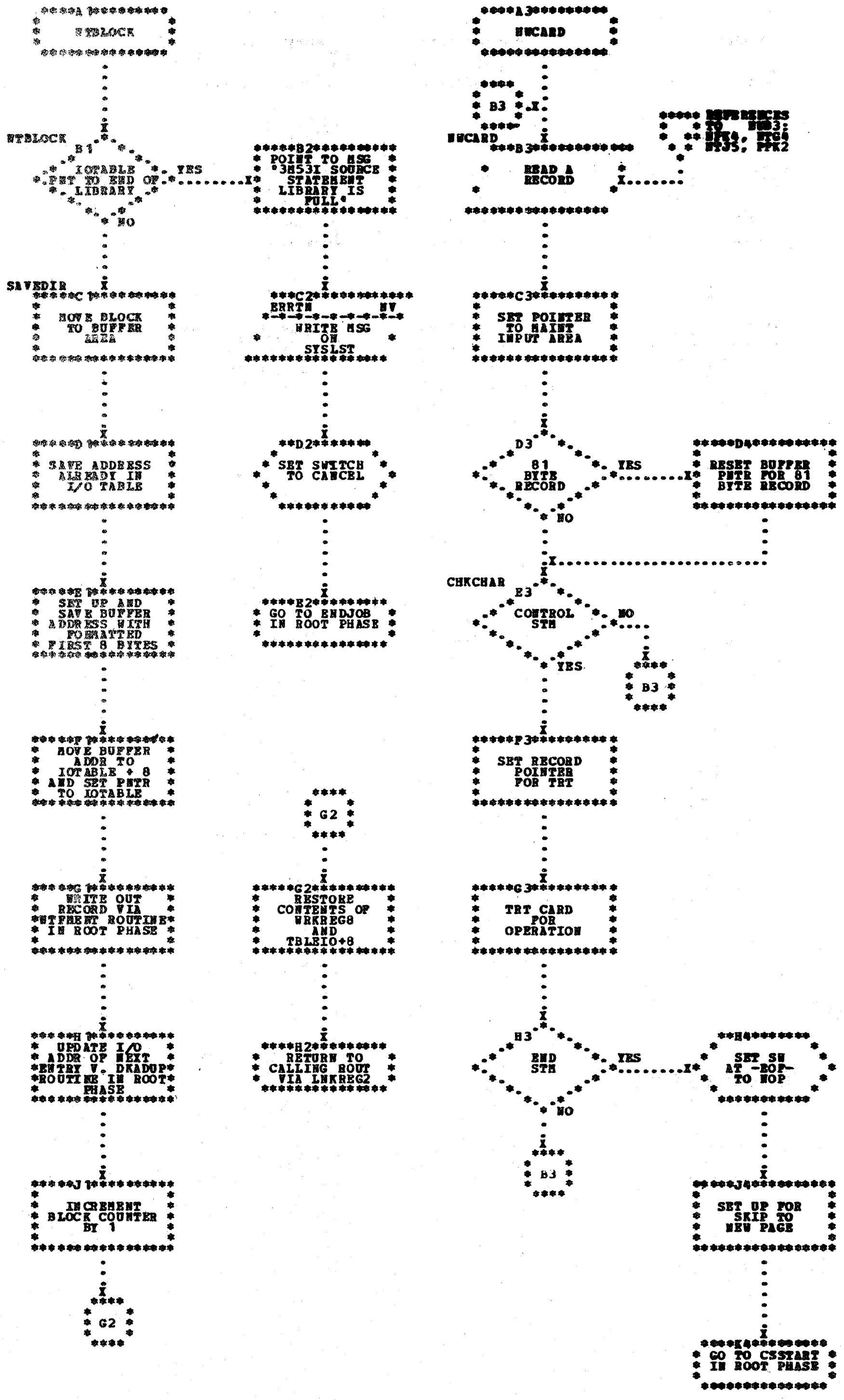
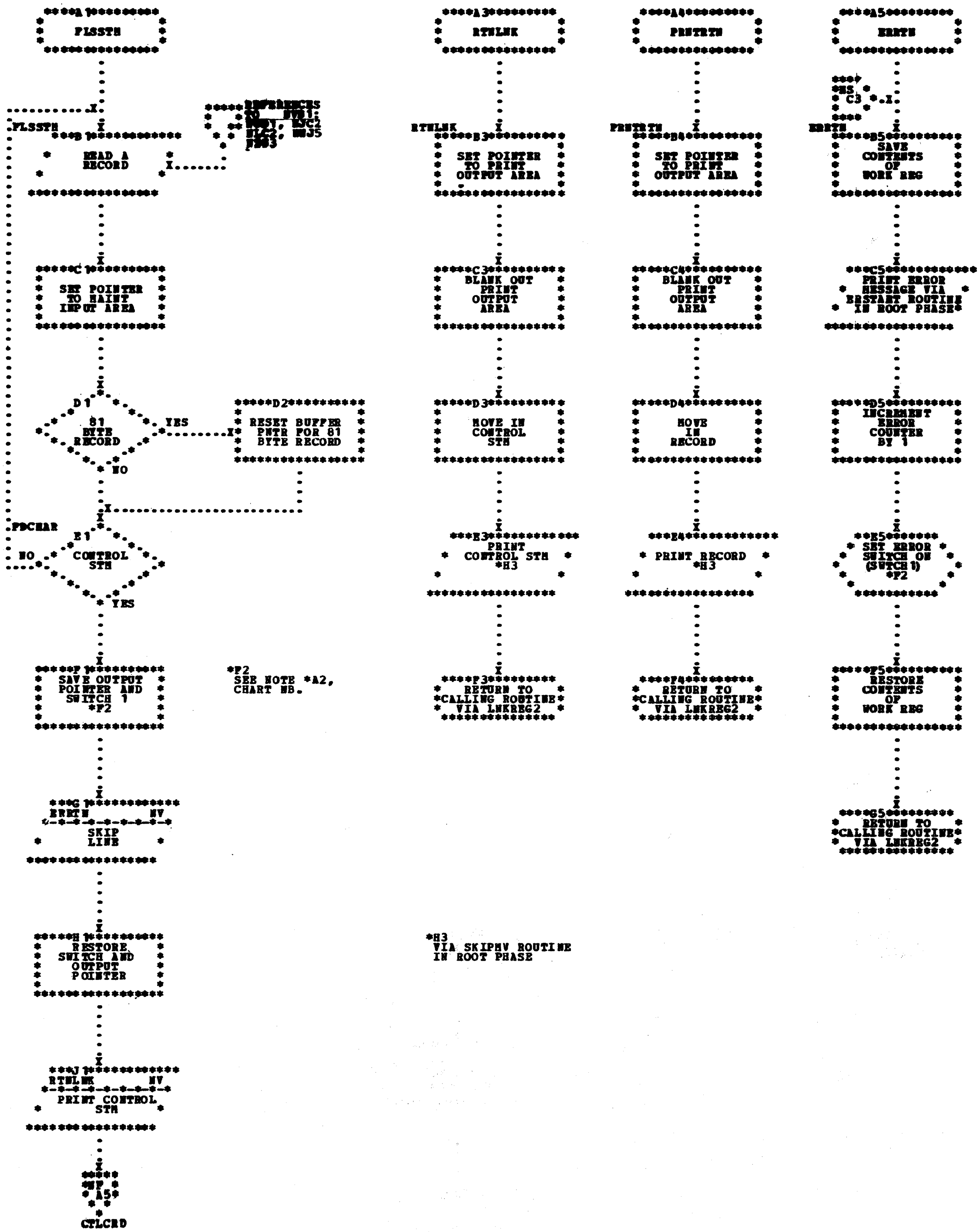

A1
NTBLOCK
NTBLOCK
B1
IOTABLE PNT TO END OF LIBRARY
YES
NO
B2
POINT TO MSG '3M53I SOURCE STATEMENT LIBRARY IS FULL'
SAVEDIR
C1
MOVE BLOCK TO BUFFER AREA
C2
ERRTN
NV
WRITE MSG ON SYSLST
D1
SAVE ADDRESS ALREADY IN I/O TABLE
D2
SET SWITCH TO CANCEL
E1
SET UP AND SAVE BUFFER ADDRESS WITH FORMATTED FIRST 8 BYTES
E2
GO TO ENDJOB IN ROOT PHASE
F1
MOVE BUFFER ADDR TO IOTABLE + 8 AND SET PNTR TO IOTABLE
G2
G1
WRITE OUT RECORD VIA NTPHENT ROUTINE IN ROOT PHASE
G2
RESTORE CONTENTS OF WRKREG8 AND TBLEIO+8
H1
UPDATE I/O ADDR OF NEXT ENTRY V. DKADUP ROUTINE IN ROOT PHASE
H2
RETURN TO CALLING ROUT VIA LNKREG2
J1
INCREMENT BLOCK COUNTER BY 1
G2
A3
NUCARD
B3
NUCARD
B3
READ A RECORD
REFERENCES TO NUB3: NPK4, NTG4 NTJ5, PPK2
C3
SET POINTER TO MAINT INPUT AREA
D3
81 BYTE RECORD
YES
NO
D4
RESET BUFFER PNTR FOR 81 BYTE RECORD
CHKCHAR
E3
CONTROL STM
NO
YES
B3
F3
SET RECORD POINTER FOR TRT
G3
TRT CARD FOR OPERATION
H3
END STM
YES
NO
H4
SET SW AT -EOP- TO NOP
B3
J4
SET UP FOR SKIP TO NEW PAGE
K4
GO TO CSSTART IN ROOT PHASE

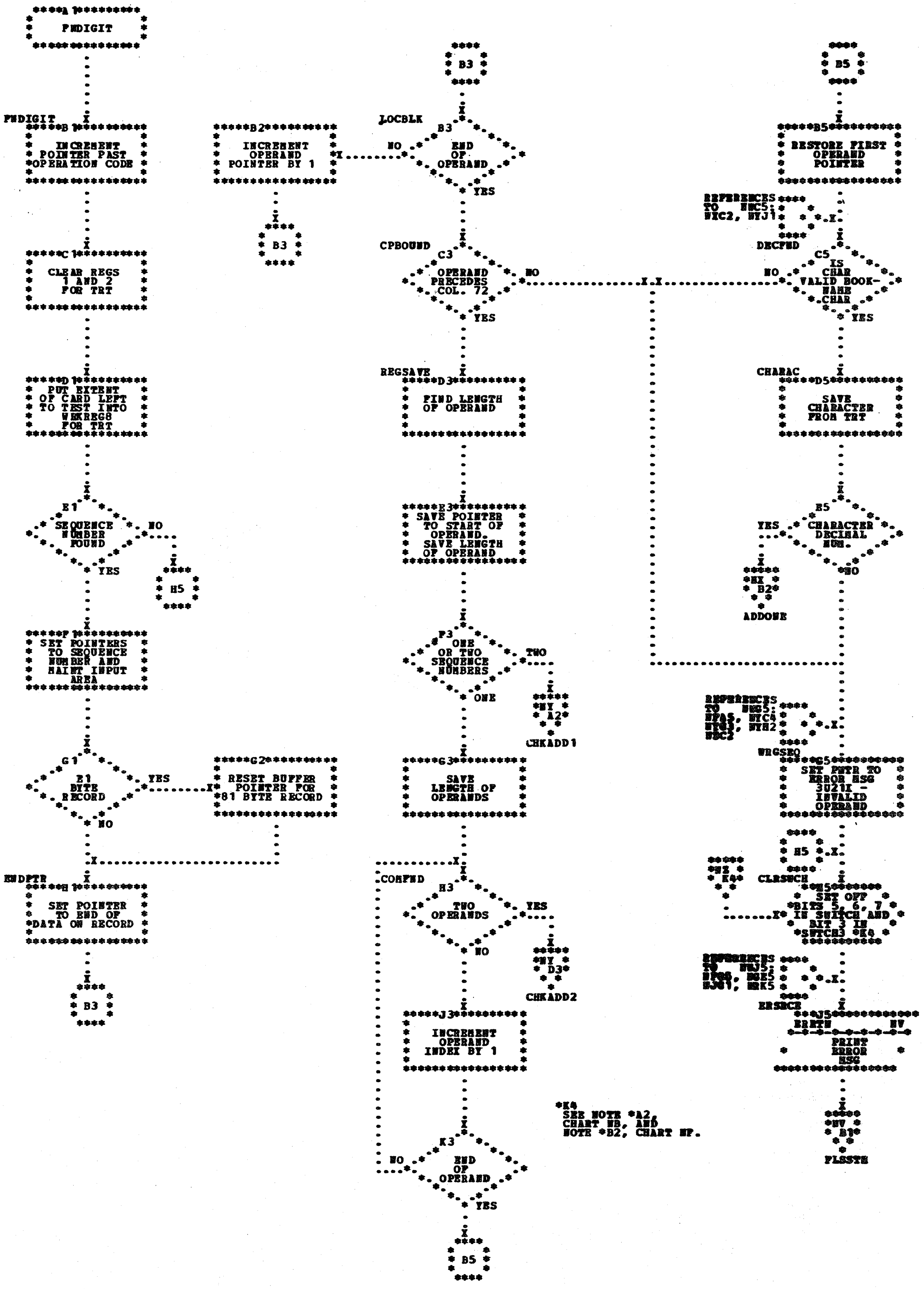
Chart NV. MAINTUP - Subroutines (Part 6 of 6)
Refer to Charts 44-46


A1
FLSSTM
FLSSTM
B1
READ A RECORD
REFERENCES TO NVB1; NWD1, NJC2 NLG2, NMJ5 NDB3
C1
SET POINTER TO MAINT INPUT AREA
D1
81 BYTE RECORD
YES
NO
D2
RESET BUFFER PNTR FOR 81 BYTE RECORD
FDCHAR
E1
CONTROL STM
NO
YES
F1
SAVE OUTPUT POINTER AND SWITCH 1 *F2
G1
ERRTN NV
SKIP LINE
H1
RESTORE SWITCH AND OUTPUT POINTER
J1
RTNLNK NV
PRINT CONTROL STM
NF A5
CTLCRD
A3
RTNLNK
RTNLNK
B3
SET POINTER TO PRINT OUTPUT AREA
C3
BLANK OUT PRINT OUTPUT AREA
D3
MOVE IN CONTROL STM
E3
PRINT CONTROL STM *H3
F3
RETURN TO CALLING ROUTINE VIA LNKREG2
A4
PRNTRTN
PRNTRTN
B4
SET POINTER TO PRINT OUTPUT AREA
C4
BLANK OUT PRINT OUTPUT AREA
D4
MOVE IN RECORD
E4
PRINT RECORD *H3
F4
RETURN TO CALLING ROUTINE VIA LNKREG2
A5
ERRTN
NS C3
ERRTN
B5
SAVE CONTENTS OF WORK REG
C5
PRINT ERROR MESSAGE VIA ERSTART ROUTINE IN ROOT PHASE
D5
INCREMENT ERROR COUNTER BY 1
E5
SET ERROR SWITCH ON (SWTCH 1) *F2
F5
RESTORE CONTENTS OF WORK REG
G5
RETURN TO CALLING ROUTINE VIA LNKREG2

*F2
SEE NOTE *A2, CHART NB.

*H3
VIA SKIPNV ROUTINE IN ROOT PHASE

Chart NW. MAINTUP - Sequence Number Check (Part 1 of 4)
Refer to Charts 44-46


A1
PNDIGIT
PNDIGIT
B1
INCREMENT POINTER PAST OPERATION CODE
C1
CLEAR REGS 1 AND 2 FOR TRT
D1
PUT EXTENT OF CARD LEFT TO TEST INTO WRKREG8 FOR TRT
E1
SEQUENCE NUMBER FOUND
NO
H5
YES
F1
SET POINTERS TO SEQUENCE NUMBER AND MAINT INPUT AREA
G1
81 BYTE RECORD
YES
NO
G2
RESET BUFFER POINTER FOR 81 BYTE RECORD
ENDPTR
H1
SET POINTER TO END OF DATA ON RECORD
B3
B2
INCREMENT OPERAND POINTER BY 1
B3
B3
LOCBLK
B3
END OF OPERAND
NO
YES
CPBOUND
C3
OPERAND PRECEDES COL. 72
NO
YES
REGSAVE
D3
FIND LENGTH OF OPERAND
E3
SAVE POINTER TO START OF OPERAND. SAVE LENGTH OF OPERAND
F3
ONE OR TWO SEQUENCE NUMBERS
TWO
NY
A2
CHKADD1
ONE
G3
SAVE LENGTH OF OPERANDS
COMPND
H3
TWO OPERANDS
YES
NY
D3
CHKADD2
NO
J3
INCREMENT OPERAND INDEX BY 1
K3
END OF OPERAND
NO
YES
B5
B5
B5
RESTORE FIRST OPERAND POINTER
REFERENCES TO NWC5: NWC2, NWJ1
DECFND
C5
IS CHAR VALID BOOK-NAME CHAR
NO
YES
CHARAC
D5
SAVE CHARACTER FROM TRT
E5
CHARACTER DECIMAL NUM.
YES
NX
B2
ADDONE
NO
REFERENCES TO NWG5: NWA5, NWC4 NWG3, NWH2 NWC2
WRGSEQ
G5
SET PNTR TO ERROR MSG 3U21I - INVALID OPERAND
H5
NZ
K4
CLRSWCH
H5
SET OFF BITS 5, 6, 7 IN SWITCH AND BIT 3 IN SWTCH3 *K4
REFERENCES TO NWJ5: NPG8, NGE5 NJG1, NRK5
ERSRCE
J5
ERRTN NV
PRINT ERROR MSG
NV
B1
FLSSTR

*K4
SEE NOTE *A2, CHART NB, AND NOTE *B2, CHART NP.

Chart NX. MAINTUP - Sequence Number Check (Part 2 of 4)
Refer to Charts 44-46

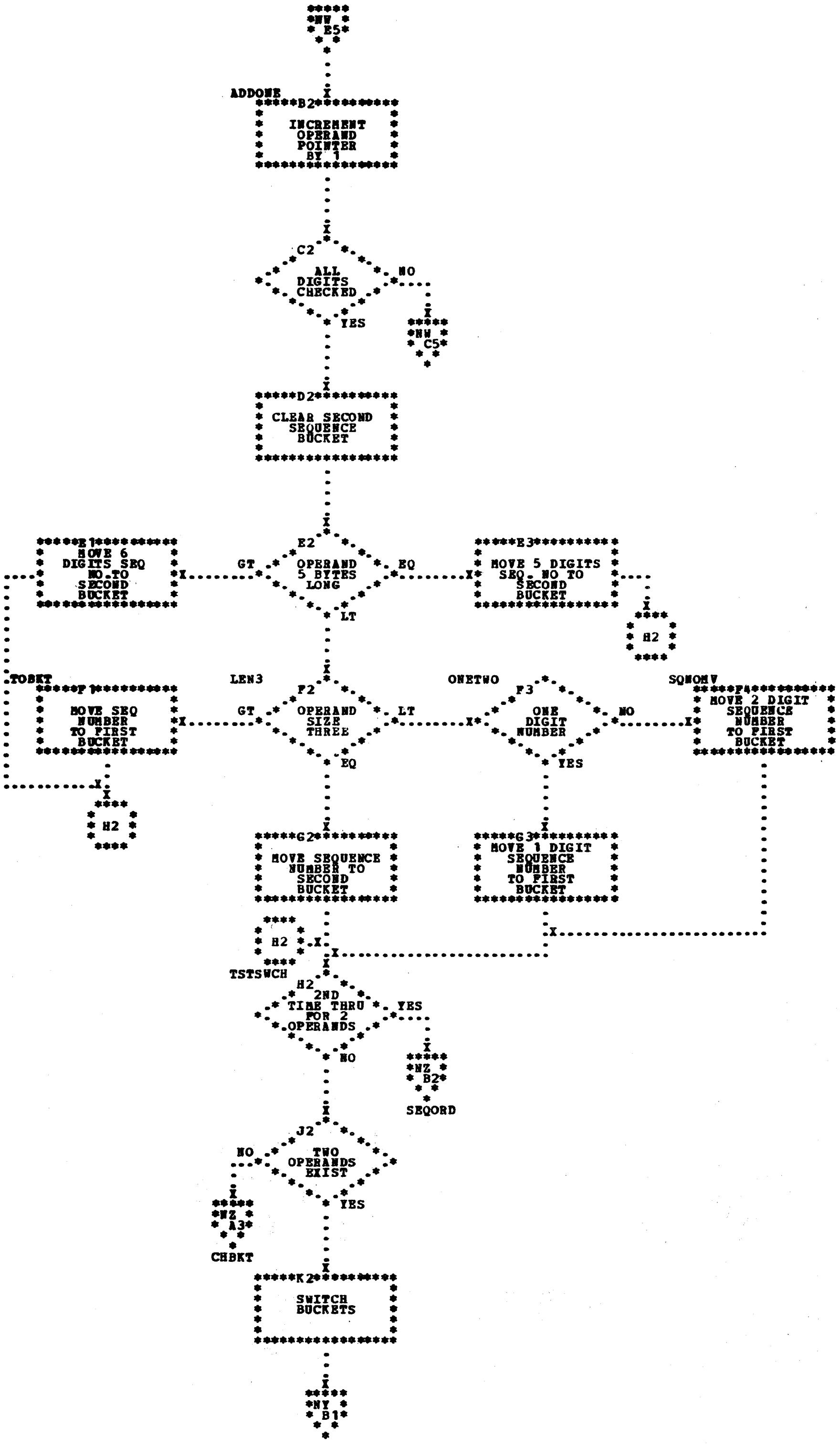

NW E5
ADDONE B2
INCREMENT OPERAND POINTER BY 1
C2 ALL DIGITS CHECKED
NO
NW C5
YES
D2
CLEAR SECOND SEQUENCE BUCKET
E1 MOVE 6 DIGITS SEQ NO.TO SECOND BUCKET
GT
E2 OPERAND 5 BYTES LONG
EQ
E3 MOVE 5 DIGITS SEQ. NO TO SECOND BUCKET
H2
LT
TOBKT F1
MOVE SEQ NUMBER TO FIRST BUCKET
GT
LEN3 F2 OPERAND SIZE THREE
LT
ONETWO F3 ONE DIGIT NUMBER
NO
SQNONV F4
MOVE 2 DIGIT SEQUENCE NUMBER TO FIRST BUCKET
H2
EQ
YES
G2 MOVE SEQUENCE NUMBER TO SECOND BUCKET
G3 MOVE 1 DIGIT SEQUENCE NUMBER TO FIRST BUCKET
H2
TSTSWCH H2 2ND TIME THRU FOR 2 OPERANDS
YES
NZ B2
SEQORD
NO
J2 TWO OPERANDS EXIST
NO
NZ A3
CHBKT
YES
K2
SWITCH BUCKETS
NY B1

Chart NY. MAINTUP - Sequence Number Check (Part 3 of 4)
Refer to Charts 44-46

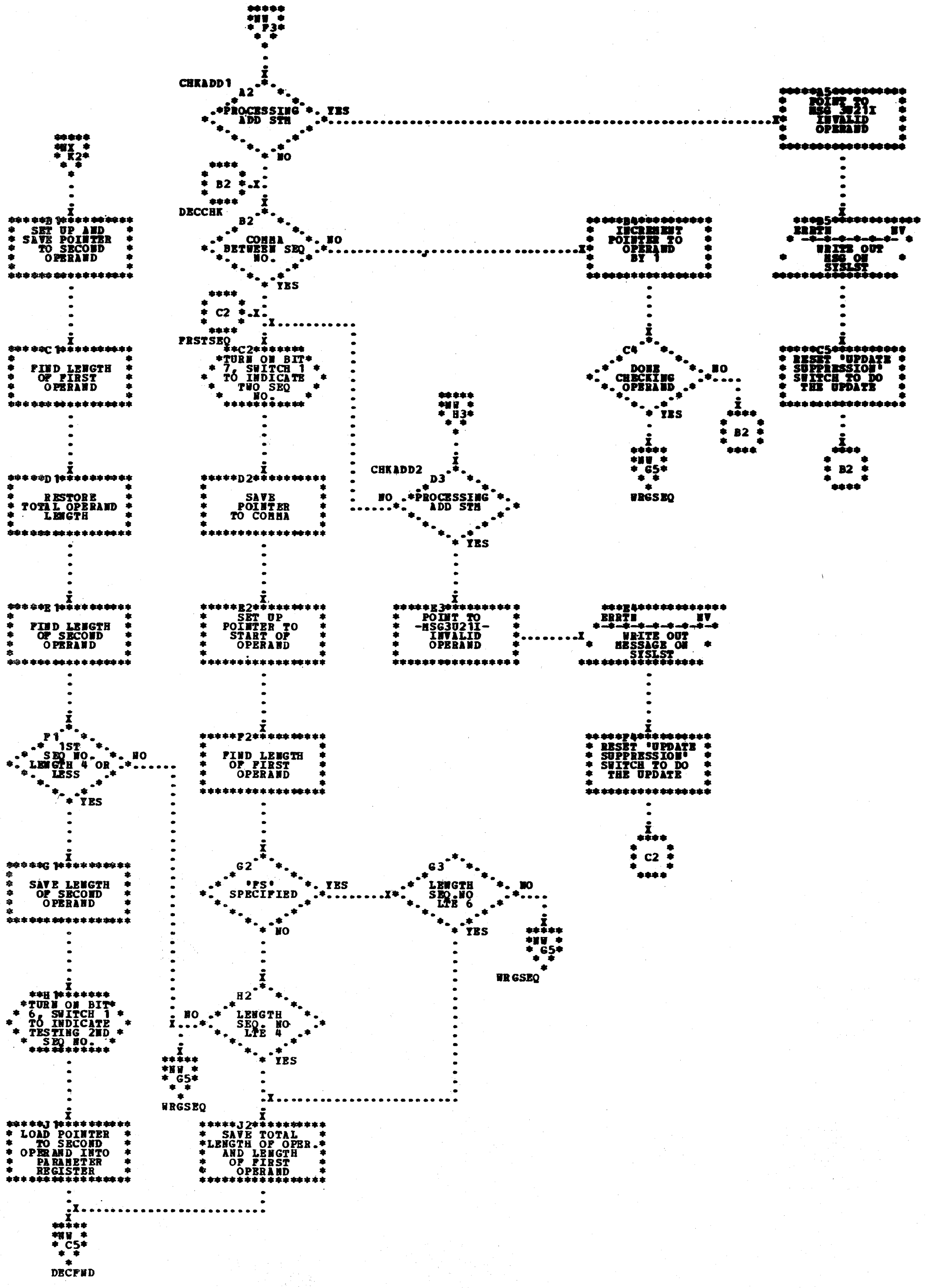

NW F3
CHKADD1
A2
PROCESSING ADD STM
YES
NO
A5
POINT TO MSG 3U21I INVALID OPERAND
NX K2
B1
SET UP AND SAVE POINTER TO SECOND OPERAND
B2
DECCHK
B2
COMMA BETWEEN SEQ NO.
NO
YES
B4
INCREMENT POINTER TO OPERAND BY 1
B5
ERRTN
NV
WRITE OUT MSG ON SYSLST
C1
FIND LENGTH OF FIRST OPERAND
C2
PRSTSEQ
C2
TURN ON BIT 7, SWITCH 1 TO INDICATE TWO SEQ NO.
C4
DONE CHECKING OPERAND
NO
YES
B2
NW G5
WRGSEQ
C5
RESET 'UPDATE SUPPRESSION' SWITCH TO DO THE UPDATE
B2
NW H3
D1
RESTORE TOTAL OPERAND LENGTH
D2
SAVE POINTER TO COMMA
CHKADD2
D3
NO
PROCESSING ADD STM
YES
E1
FIND LENGTH OF SECOND OPERAND
E2
SET UP POINTER TO START OF OPERAND
E3
POINT TO -MSG3U21I- INVALID OPERAND
E4
ERRTN
NV
WRITE OUT MESSAGE ON SYSLST
F1
1ST SEQ NO. LENGTH 4 OR LESS
NO
YES
F2
FIND LENGTH OF FIRST OPERAND
F4
RESET 'UPDATE SUPPRESSION' SWITCH TO DO THE UPDATE
C2
G1
SAVE LENGTH OF SECOND OPERAND
G2
'PS' SPECIFIED
YES
NO
G3
LENGTH SEQ.NO LTE 6
NO
YES
NW G5
WRGSEQ
H1
TURN ON BIT 6, SWITCH 1 TO INDICATE TESTING 2ND SEQ NO.
H2
NO
LENGTH SEQ. NO LTE 4
YES
NW G5
WRGSEQ
J1
LOAD POINTER TO SECOND OPERAND INTO PARAMETER REGISTER
J2
SAVE TOTAL LENGTH OF OPER. AND LENGTH OF FIRST OPERAND
NW C5
DECFND

Chart NZ. MAINTUP - Sequence Number Chart (Part 4 of 4)
Refer to Charts 44-46

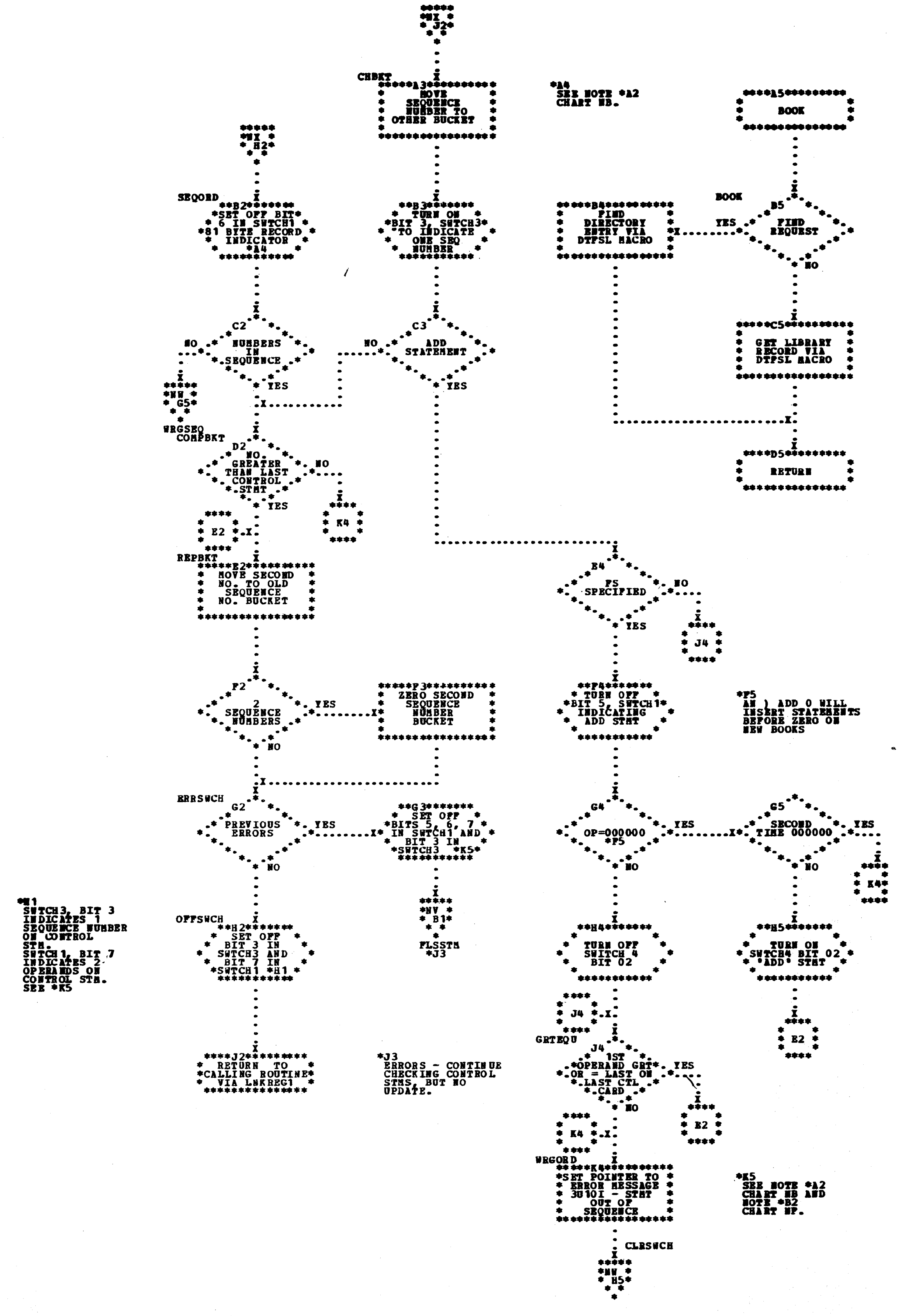

NX J2
CHBKT
A3 MOVE SEQUENCE NUMBER TO OTHER BUCKET
*A4 SEE NOTE *A2 CHART NB.
A5 BOOK
NX H2
SEQORD
B2 SET OFF BIT 6 IN SWTCH1 81 BYTE RECORD INDICATOR *A4
B3 TURN ON BIT 3, SWTCH3 TO INDICATE ONE SEQ NUMBER
B4 FIND DIRECTORY ENTRY VIA DTFSL MACRO
BOOK
B5 FIND REQUEST
YES
NO
C2 NUMBERS IN SEQUENCE
NO
YES
NW G5
C3 ADD STATEMENT
NO
YES
C5 GET LIBRARY RECORD VIA DTFSL MACRO
WRGSEQ
COMPBKT
D2 NO. GREATER THAN LAST CONTROL STMT
NO
YES
K4
E2
D5 RETURN
REPBKT
E2 MOVE SECOND NO. TO OLD SEQUENCE NO. BUCKET
E4 FS SPECIFIED
NO
YES
J4
F2 2 SEQUENCE NUMBERS
YES
NO
F3 ZERO SECOND SEQUENCE NUMBER BUCKET
F4 TURN OFF BIT 5, SWTCH1 INDICATING ADD STMT
*F5 AN) ADD 0 WILL INSERT STATEMENTS BEFORE ZERO ON NEW BOOKS
ERRSWCH
G2 PREVIOUS ERRORS
YES
NO
G3 SET OFF BITS 5, 6, 7 IN SWTCH1 AND BIT 3 IN SWTCH3 *K5
G4 OP=000000 *F5
YES
NO
G5 SECOND TIME 000000
YES
NO
K4
NV B1
FLSSTM *J3
*H1 SWTCH3, BIT 3 INDICATES 1 SEQUENCE NUMBER ON CONTROL STM. SWTCH1, BIT 7 INDICATES 2 OPERANDS ON CONTROL STM. SEE *K5
OFFSWCH
H2 SET OFF BIT 3 IN SWTCH3 AND BIT 7 IN SWTCH1 *H1
H4 TURN OFF SWITCH 4 BIT 02
H5 TURN ON SWTCH4 BIT 02 'ADD' STMT
J4
E2
GRTEQU
J4 1ST OPERAND GRT OR = LAST ON LAST CTL. CARD
YES
NO
E2
K4
J2 RETURN TO CALLING ROUTINE VIA LNKREG1
*J3 ERRORS - CONTINUE CHECKING CONTROL STMS, BUT NO UPDATE.
WRGORD
K4 SET POINTER TO ERROR MESSAGE 3U10I - STMT OUT OF SEQUENCE
*K5 SEE NOTE *A2 CHART NB AND NOTE *B2 CHART NP.
CLRSWCH
NW H5

Chart PC. MAINTUPF - Initialization

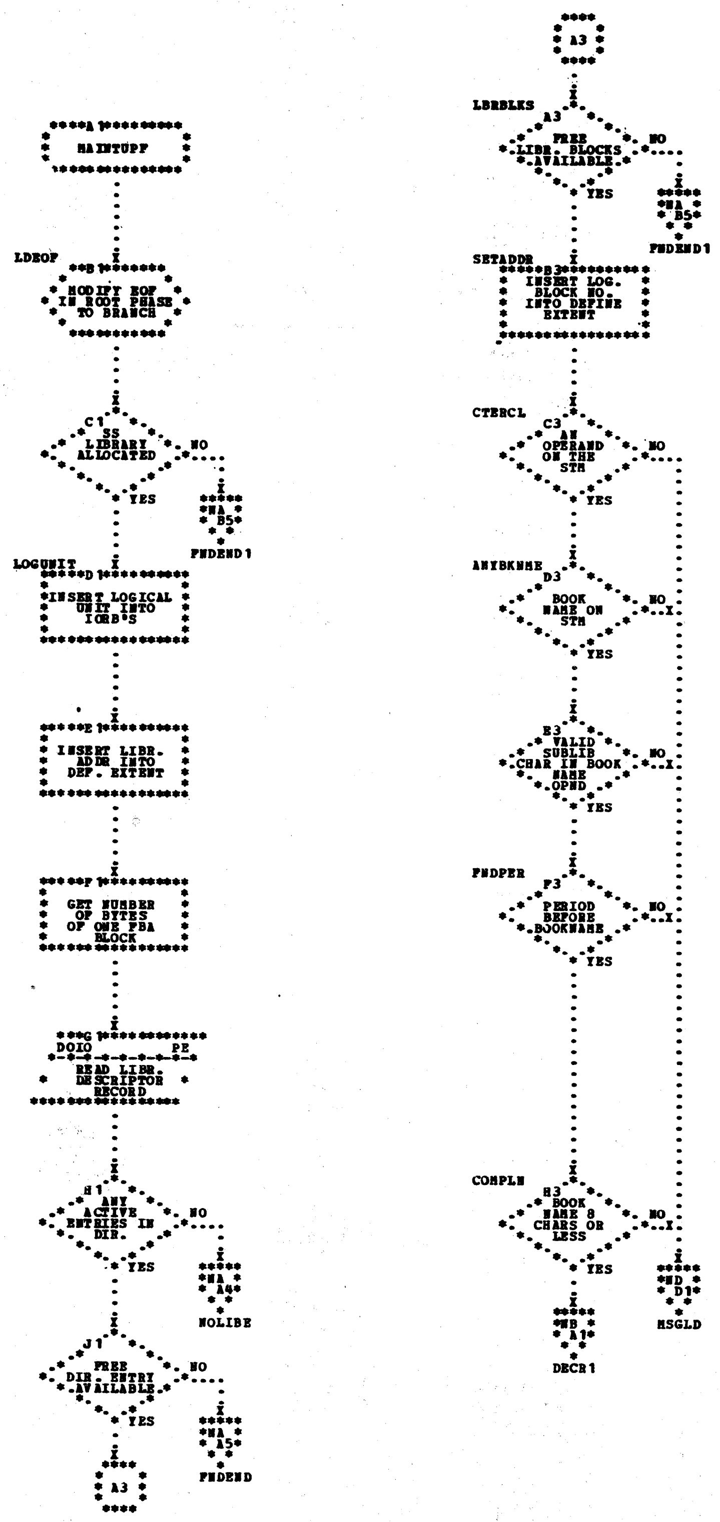

A1
MAINTUPF
LDEOF
B1
MODIFY EOF IN ROOT PHASE TO BRANCH
C1
SS LIBRARY ALLOCATED
NO
YES
NA B5
FNDEND1
LOGUNIT
D1
INSERT LOGICAL UNIT INTO IORB'S
E1
INSERT LIBR. ADDR INTO DEF. EXTENT
F1
GET NUMBER OF BYTES OF ONE FBA BLOCK
G1
DOIO
PE
READ LIBR. DESCRIPTOR RECORD
H1
ANY ACTIVE ENTRIES IN DIR.
NO
YES
NA A4
NOLIBE
J1
FREE DIR. ENTRY AVAILABLE
NO
YES
NA A5
FNDEND
A3
A3
LBRBLKS
A3
FREE LIBR. BLOCKS AVAILABLE
NO
YES
NA B5
FNDEND1
SETADDR
B3
INSERT LOG. BLOCK NO. INTO DEFINE EXTENT
CTERCL
C3
AN OPERAND ON THE STM
NO
YES
ANYBKNME
D3
BOOK NAME ON STM
NO
YES
E3
VALID SUBLIB CHAR IN BOOK NAME OPND
NO
YES
FNDPER
F3
PERIOD BEFORE BOOKNAME
NO
YES
COMPLN
H3
BOOK NAME 8 CHARS OR LESS
NO
YES
NB A1
DECR1
ND D1
MSGLD

Chart PD. MAINTUPF - END Statem. Routine. Update Directory and Library Descriptor

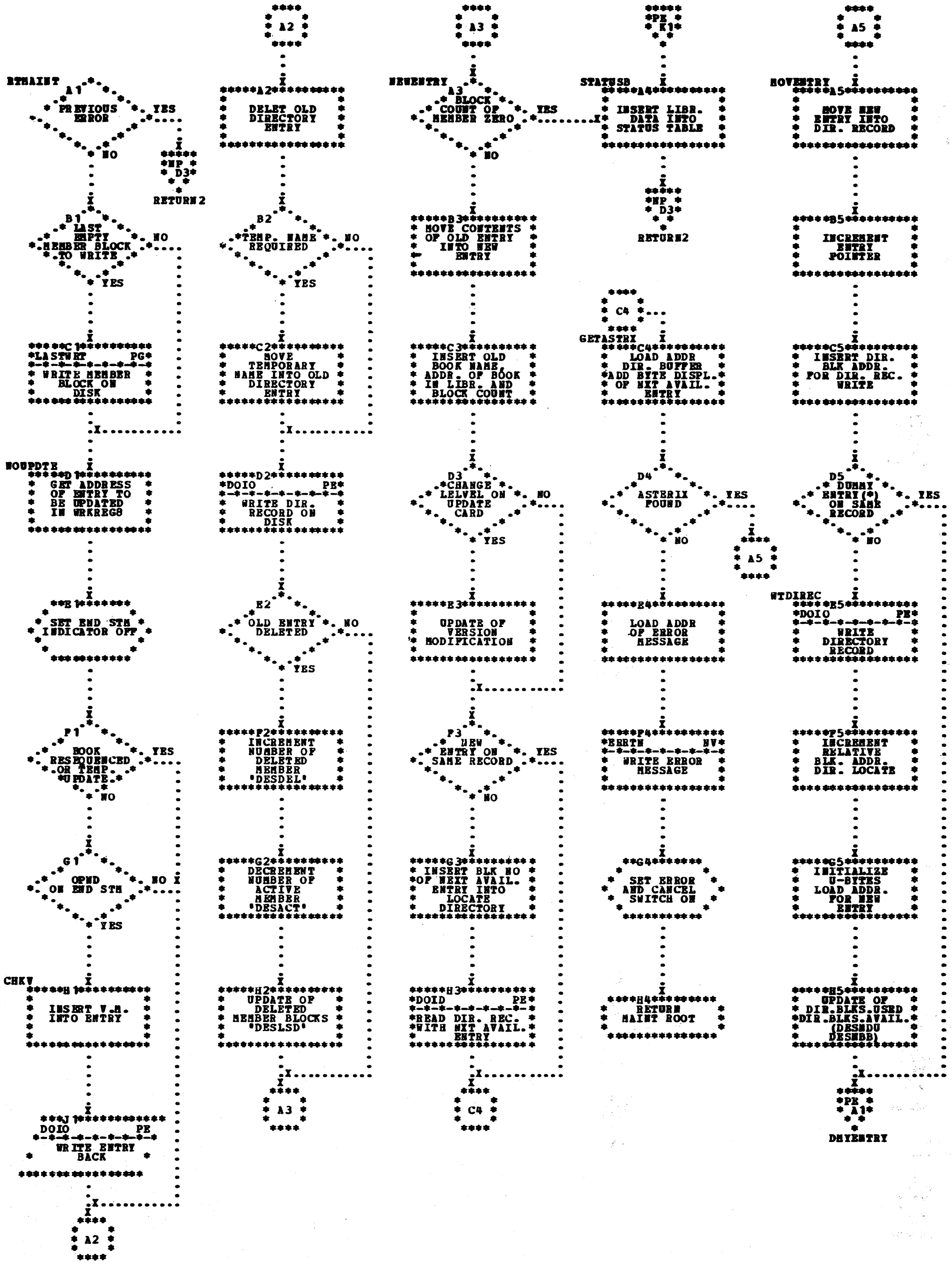
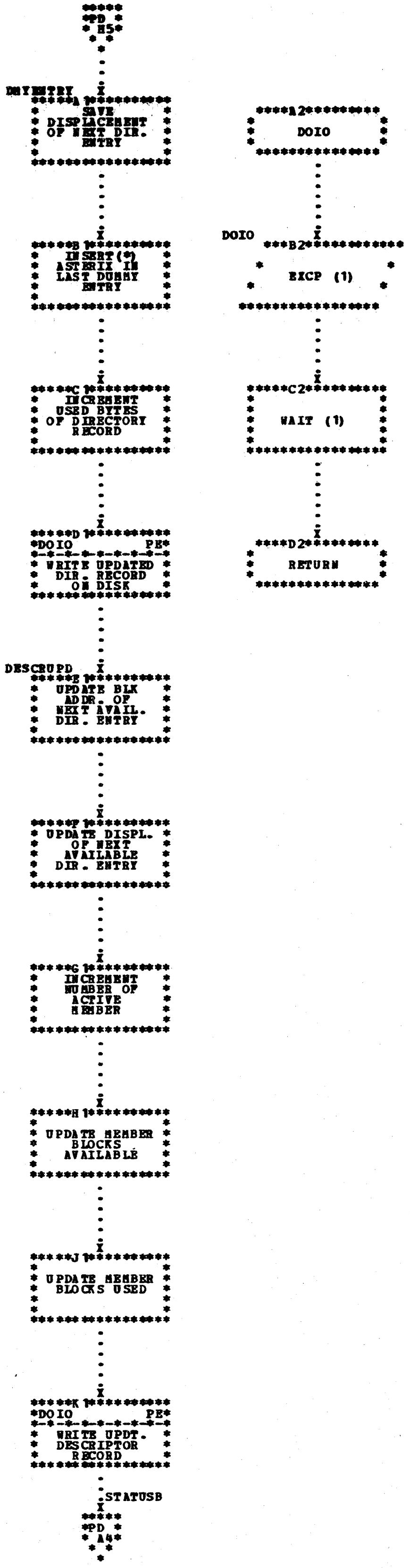

A2
A3
PE K1
A5
RTMAINT
A1
PREVIOUS ERROR
YES
NO
NP D3
RETURN2
A2
DELET OLD DIRECTORY ENTRY
NEWENTRY
A3
BLOCK COUNT OF MEMBER ZERO
YES
NO
STATUSB
A4
INSERT LIBR. DATA INTO STATUS TABLE
NP D3
RETURN2
MOVENTRY
A5
MOVE NEW ENTRY INTO DIR. RECORD
B1
LAST EMPTY MEMBER BLOCK TO WRITE
NO
YES
B2
TEMP. NAME REQUIRED
NO
YES
B3
MOVE CONTENTS OF OLD ENTRY INTO NEW ENTRY
B5
INCREMENT ENTRY POINTER
C4
C1
LASTWRT PG
WRITE MEMBER BLOCK ON DISK
C2
MOVE TEMPORARY NAME INTO OLD DIRECTORY ENTRY
C3
INSERT OLD BOOK NAME, ADDR. OF BOOK IN LIBR. AND BLOCK COUNT
GETASTRX
C4
LOAD ADDR DIR. BUFFER ADD BYTE DISPL. OF NXT AVAIL. ENTRY
C5
INSERT DIR. BLK ADDR. FOR DIR. REC. WRITE
NOUPDTE
D1
GET ADDRESS OF ENTRY TO BE UPDATED IN WRKREG8
D2
DOIO PE
WRITE DIR. RECORD ON DISK
D3
CHANGE LELVEL ON UPDATE CARD
NO
YES
D4
ASTERIX FOUND
YES
NO
A5
D5
DUMMY ENTRY(*) ON SAME RECORD
YES
NO
E1
SET END STM INDICATOR OFF
E2
OLD ENTRY DELETED
NO
YES
E3
UPDATE OF VERSION MODIFICATION
E4
LOAD ADDR OF ERROR MESSAGE
WTDIREC
E5
DOIO PE
WRITE DIRECTORY RECORD
F1
BOOK RESEQUENCED OR TEMP. UPDATE
YES
NO
F2
INCREMENT NUMBER OF DELETED MEMBER 'DESDEL'
F3
NEW ENTRY ON SAME RECORD
YES
NO
F4
ERRTN NV
WRITE ERROR MESSAGE
F5
INCREMENT RELATIVE BLK. ADDR. DIR. LOCATE
G1
OPND ON END STM
NO
YES
G2
DECREMENT NUMBER OF ACTIVE MEMBER 'DESACT'
G3
INSERT BLK NO OF NEXT AVAIL. ENTRY INTO LOCATE DIRECTORY
G4
SET ERROR AND CANCEL SWITCH ON
G5
INITIALIZE U-BYTES LOAD ADDR. FOR NEW ENTRY
CHKV
H1
INSERT V.M. INTO ENTRY
H2
UPDATE OF DELETED MEMBER BLOCKS 'DESLSD'
H3
DOID PE
READ DIR. REC. WITH NXT AVAIL. ENTRY
H4
RETURN MAINT ROOT
H5
UPDATE OF DIR.BLKS.USED DIR.BLKS.AVAIL. (DESMDU DESMBB)
A3
C4
PE A1
DMYENTRY
J1
DOIO PE
WRITE ENTRY BACK
A2

Chart PE. MAINTUPF - Update Directory and Library Descriptor



PD H5
DMYENTRY
A1
SAVE DISPLACEMENT OF NEXT DIR. ENTRY
B1
INSERT(*) ASTERIX IN LAST DUMMY ENTRY
C1
INCREMENT USED BYTES OF DIRECTORY RECORD
D1
DOIO PE
WRITE UPDATED DIR. RECORD ON DISK
DESCRUPD
E1
UPDATE BLK ADDR. OF NEXT AVAIL. DIR. ENTRY
F1
UPDATE DISPL. OF NEXT AVAILABLE DIR. ENTRY
G1
INCREMENT NUMBER OF ACTIVE MEMBER
H1
UPDATE MEMBER BLOCKS AVAILABLE
J1
UPDATE MEMBER BLOCKS USED
K1
DOIO PE
WRITE UPDT. DESCRIPTOR RECORD
STATUSB
PD A4
A2
DOIO
DOIO
B2
EXCP (1)
C2
WAIT (1)
D2
RETURN

Chart PF. MAINTUPF - Prepare Directory Scan

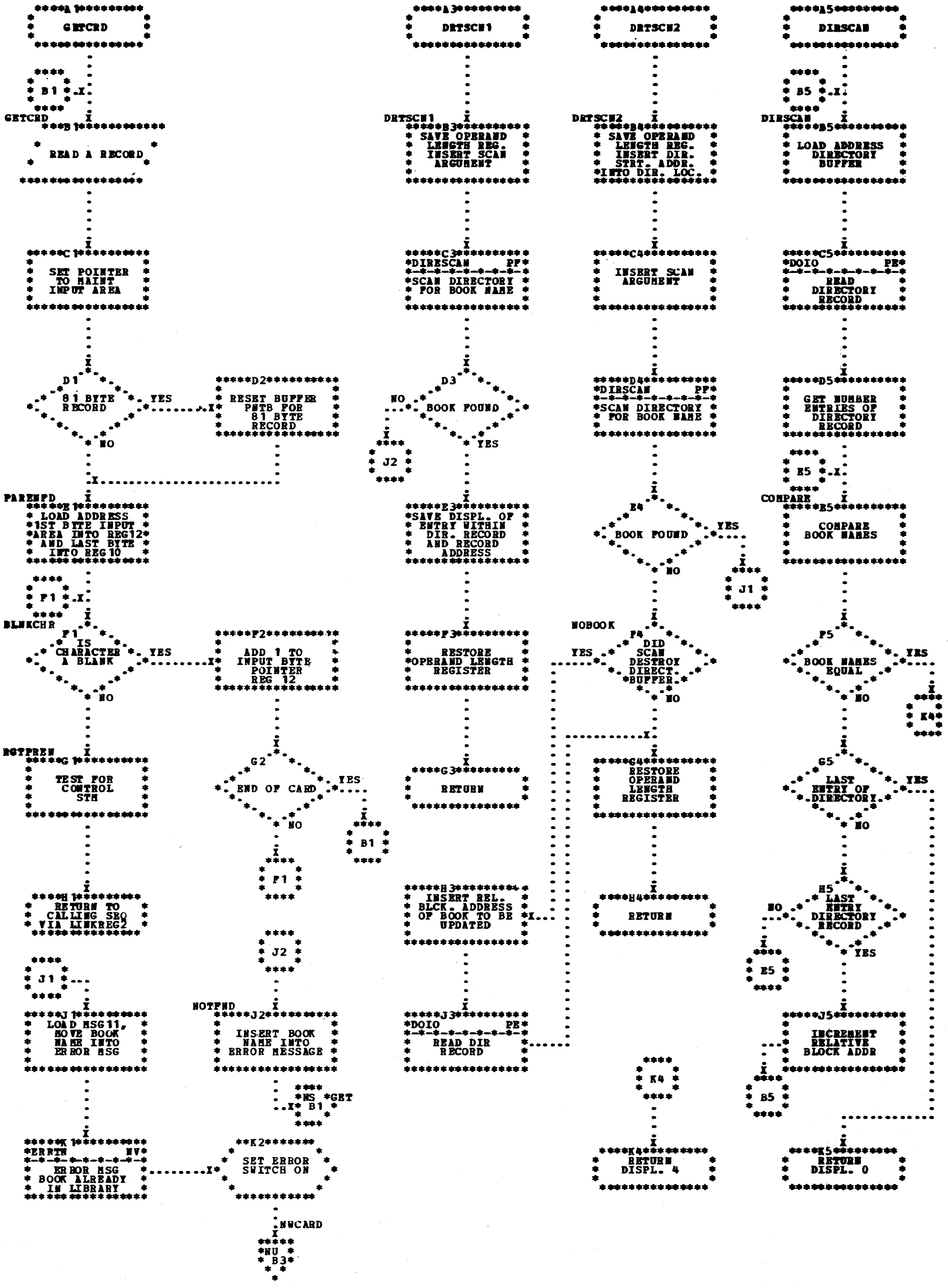

A1
GETCRD
B1
GETCRD
B1
READ A RECORD
C1
SET POINTER TO MAINT INPUT AREA
D1
81 BYTE RECORD
YES
NO
D2
RESET BUFFER PNTB FOR 81 BYTE RECORD
PARENPD
E1
LOAD ADDRESS 1ST BYTE INPUT AREA INTO REG12 AND LAST BYTE INTO REG 10
F1
BLNKCHR
F1
IS CHARACTER A BLANK
YES
NO
F2
ADD 1 TO INPUT BYTE POINTER REG 12
RGTPREN
G1
TEST FOR CONTROL STM
G2
END OF CARD
YES
NO
B1
F1
H1
RETURN TO CALLING SEQ VIA LINKREG2
J2
J1
J1
LOAD MSG11, MOVE BOOK NAME INTO ERROR MSG
NOTFND
J2
INSERT BOOK NAME INTO ERROR MESSAGE
NS GET B1
K1
ERRTN NV
ERROR MSG BOOK ALREADY IN LIBRARY
K2
SET ERROR SWITCH ON
NWCARD
NU B3
A3
DRTSCN1
DRTSCN1
B3
SAVE OPERAND LENGTH REG. INSERT SCAN ARGUMENT
C3
DIRSCAN PF
SCAN DIRECTORY FOR BOOK NAME
D3
BOOK FOUND
NO
YES
J2
E3
SAVE DISPL. OF ENTRY WITHIN DIR. RECORD AND RECORD ADDRESS
F3
RESTORE OPERAND LENGTH REGISTER
G3
RETURN
H3
INSERT REL. BLCK. ADDRESS OF BOOK TO BE UPDATED
J3
DOIO PE
READ DIR RECORD
A4
DRTSCN2
DRTSCN2
B4
SAVE OPERAND LENGTH REG. INSERT DIR. STRT. ADDR. INTO DIR. LOC.
C4
INSERT SCAN ARGUMENT
D4
DIRSCAN PF
SCAN DIRECTORY FOR BOOK NAME
E4
BOOK FOUND
YES
NO
J1
NOBOOK
F4
DID SCAN DESTROY DIRECT. BUFFER
YES
NO
G4
RESTORE OPERAND LENGTH REGISTER
H4
RETURN
K4
K4
RETURN DISPL. 4
A5
DIRSCAN
B5
DIRSCAN
B5
LOAD ADDRESS DIRECTORY BUFFER
C5
DOIO PE
READ DIRECTORY RECORD
D5
GET NUMBER ENTRIES OF DIRECTORY RECORD
E5
COMPARE
E5
COMPARE BOOK NAMES
F5
BOOK NAMES EQUAL
YES
NO
K4
G5
LAST ENTRY OF DIRECTORY
YES
NO
H5
LAST ENTRY DIRECTORY RECORD
NO
YES
E5
J5
INCREMENT RELATIVE BLOCK ADDR
B5
K5
RETURN DISPL. 0

Chart PG. MAINTUPF - Member Block Write Routine

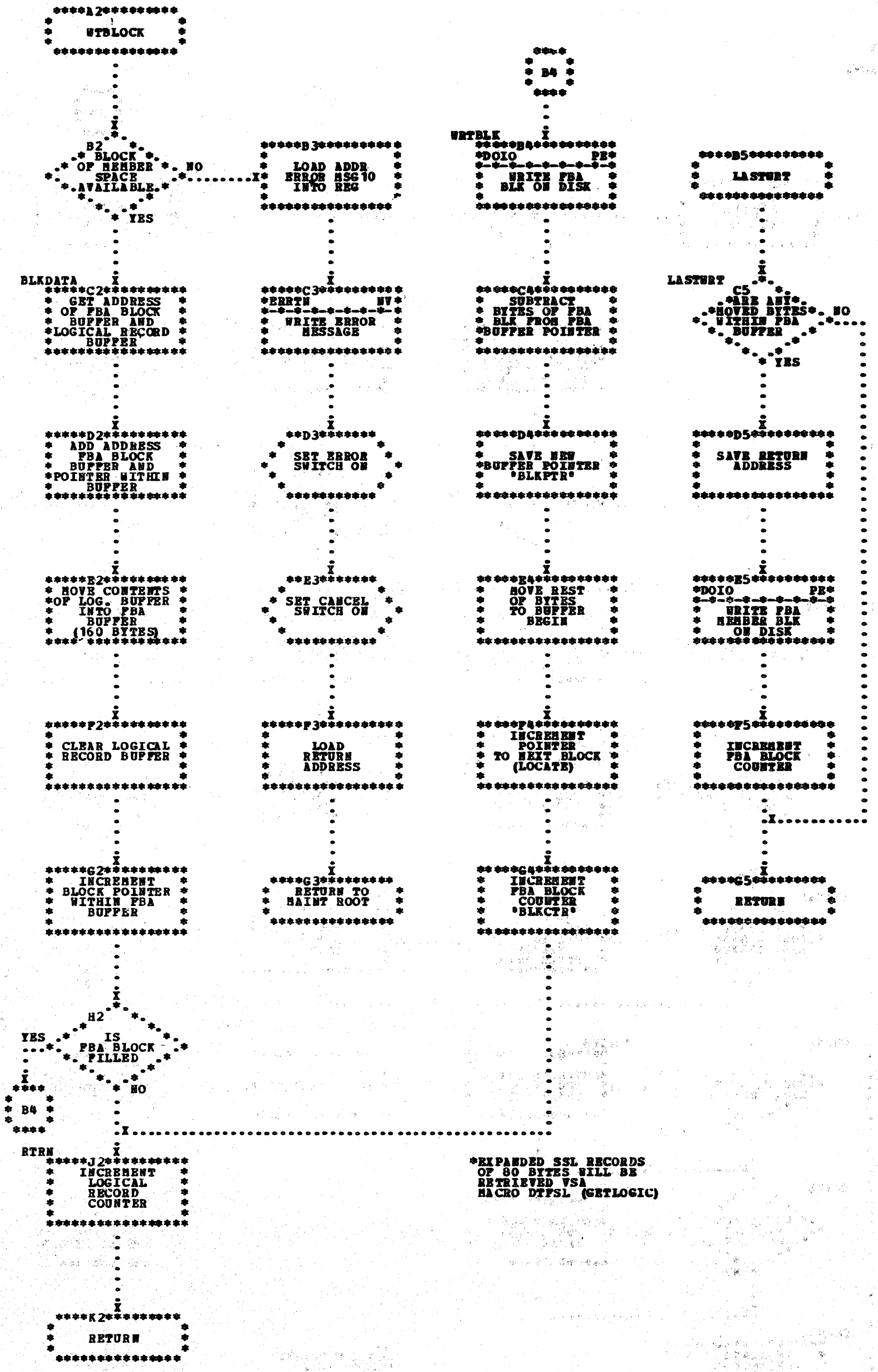

A2
WTBLOCK
B2
BLOCK OF MEMBER SPACE AVAILABLE
NO
YES
B3
LOAD ADDR ERROR MSG10 INTO REG
B4
WRTBLK
B4
DOIO PE
WRITE FBA BLK ON DISK
B5
LASTWRT
BLKDATA
C2
GET ADDRESS OF FBA BLOCK BUFFER AND LOGICAL RECORD BUFFER
C3
ERRTN MV
WRITE ERROR MESSAGE
C4
SUBTRACT BYTES OF FBA BLK FROM FBA BUFFER POINTER
LASTWRT
C5
ARE ANY MOVED BYTES WITHIN FBA BUFFER
NO
YES
D2
ADD ADDRESS FBA BLOCK BUFFER AND POINTER WITHIN BUFFER
D3
SET ERROR SWITCH ON
D4
SAVE NEW BUFFER POINTER 'BLKPTR'
D5
SAVE RETURN ADDRESS
E2
MOVE CONTENTS OF LOG. BUFFER INTO FBA BUFFER (160 BYTES)
E3
SET CANCEL SWITCH ON
E4
MOVE REST OF BYTES TO BUFFER BEGIN
E5
DOIO PE
WRITE FBA MEMBER BLK ON DISK
F2
CLEAR LOGICAL RECORD BUFFER
F3
LOAD RETURN ADDRESS
F4
INCREMENT POINTER TO NEXT BLOCK (LOCATE)
F5
INCREMENT FBA BLOCK COUNTER
G2
INCREMENT BLOCK POINTER WITHIN FBA BUFFER
G3
RETURN TO MAINT ROOT
G4
INCREMENT FBA BLOCK COUNTER 'BLKCTR'
G5
RETURN
H2
IS FBA BLOCK FILLED
YES
NO
B4
RTRN
J2
INCREMENT LOGICAL RECORD COUNTER
K2
RETURN
*EXPANDED SSL RECORDS OF 80 BYTES WILL BE RETRIEVED VSA MACRO DTFSL (GETLOGIC)

Chart RA. DSERV1
Refer to Charts 72-73

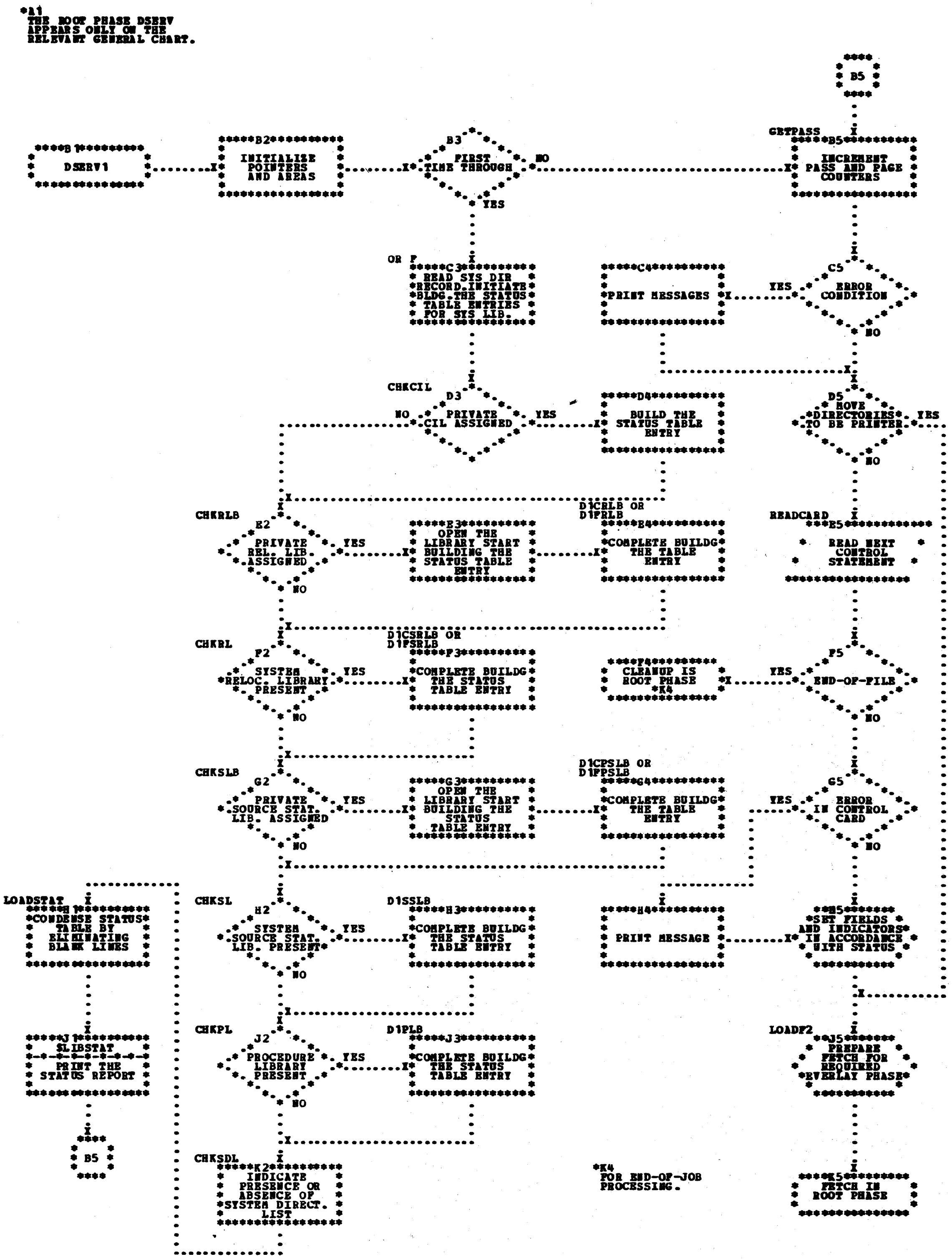

*A1
THE ROOT PHASE DSERV
APPEARS ONLY ON THE
RELEVANT GENERAL CHART.
B1
DSERV1
B2
INITIALIZE POINTERS AND AREAS
B3
FIRST TIME THROUGH
NO
YES
B5
GETPASS
B5
INCREMENT PASS AND PAGE COUNTERS
OR F
C3
READ SYS DIR RECORD.INITIATE BLDG.THE STATUS TABLE ENTRIES FOR SYS LIB.
C4
PRINT MESSAGES
YES
C5
ERROR CONDITION
NO
CHKCIL
D3
NO
PRIVATE CIL ASSIGNED
YES
D4
BUILD THE STATUS TABLE ENTRY
D5
MOVE DIRECTORIES TO BE PRINTER.
YES
NO
CHKRLB
E2
PRIVATE REL. LIB. ASSIGNED
YES
NO
E3
OPEN THE LIBRARY START BUILDING THE STATUS TABLE ENTRY
D1CRLB OR
D1FRLB
E4
COMPLETE BUILDG THE TABLE ENTRY
READCARD
E5
READ NEXT CONTROL STATEMENT
CHKRL
F2
SYSTEM RELOC. LIBRARY PRESENT
YES
NO
D1CSRLB OR
D1FSRLB
F3
COMPLETE BUILDG THE STATUS TABLE ENTRY
F4
CLEANUP IS ROOT PHASE
*K4
YES
F5
END-OF-FILE
NO
CHKSLB
G2
PRIVATE SOURCE STAT. LIB. ASSIGNED
YES
NO
G3
OPEN THE LIBRARY START BUILDING THE STATUS TABLE ENTRY
D1CPSLB OR
D1FPSLB
G4
COMPLETE BUILDG THE TABLE ENTRY
YES
G5
ERROR IN CONTROL CARD
NO
LOADSTAT
H1
CONDENSE STATUS TABLE BY ELIMINATING BLANK LINES
CHKSL
H2
SYSTEM SOURCE STAT. LIB. PRESENT
YES
NO
D1SSLB
H3
COMPLETE BUILDG THE STATUS TABLE ENTRY
H4
PRINT MESSAGE
H5
SET FIELDS AND INDICATORS IN ACCORDANCE WITH STATUS
J1
$LIBSTAT
PRINT THE STATUS REPORT
CHKPL
J2
PROCEDURE LIBRARY PRESENT
YES
NO
D1PLB
J3
COMPLETE BUILDG THE STATUS TABLE ENTRY
LOADF2
J5
PREPARE FETCH FOR REQUIRED OVERLAY PHASE
B5
CHKSDL
K2
INDICATE PRESENCE OR ABSENCE OF SYSTEM DIRECT. LIST
*K4
FOR END-OF-JOB PROCESSING.
K5
FETCH IN ROOT PHASE

Chart RB. DSERV2
Refer to Chart 74

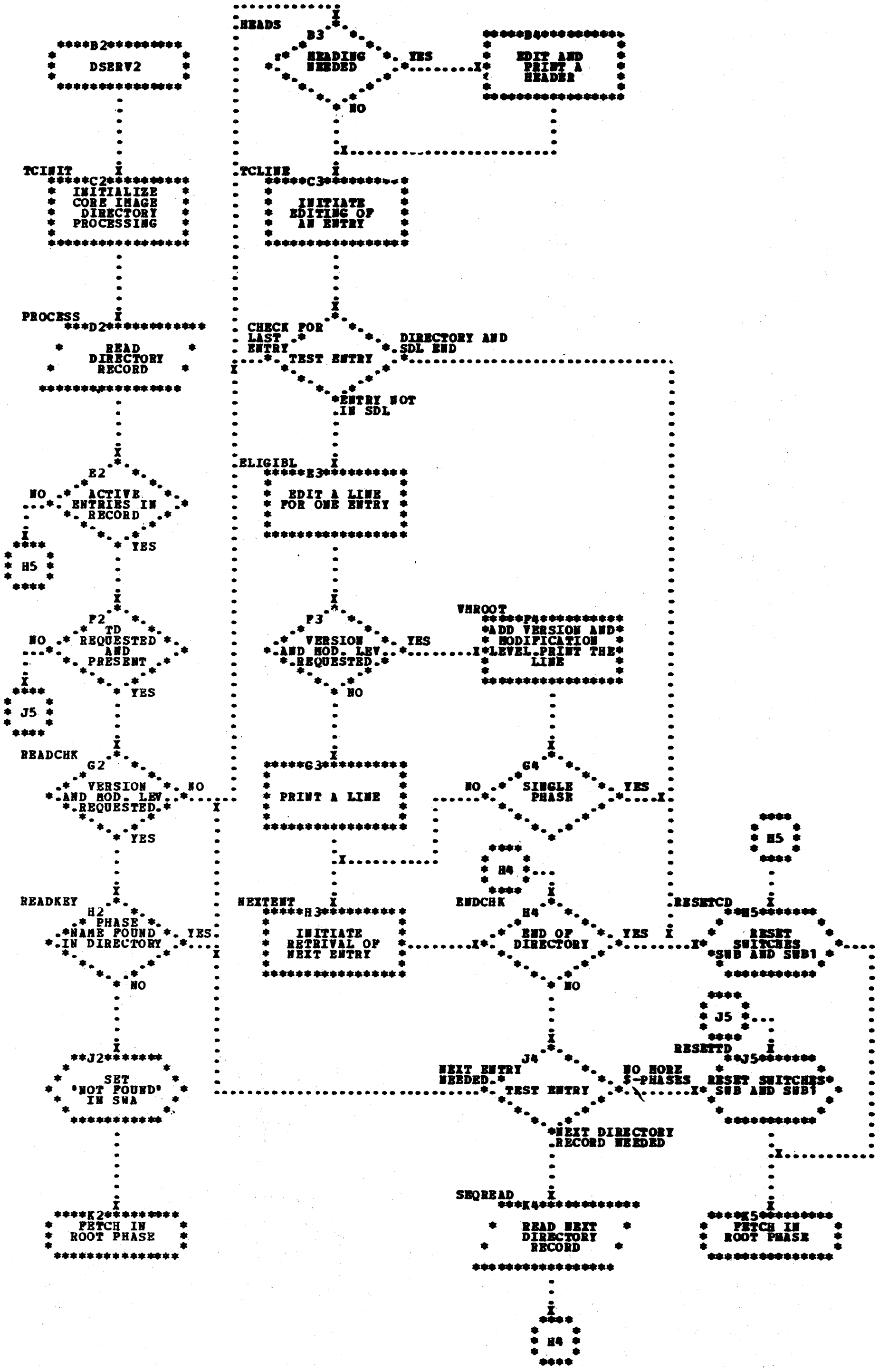


Chart RC. DSERV2F
Refer to Chart 74

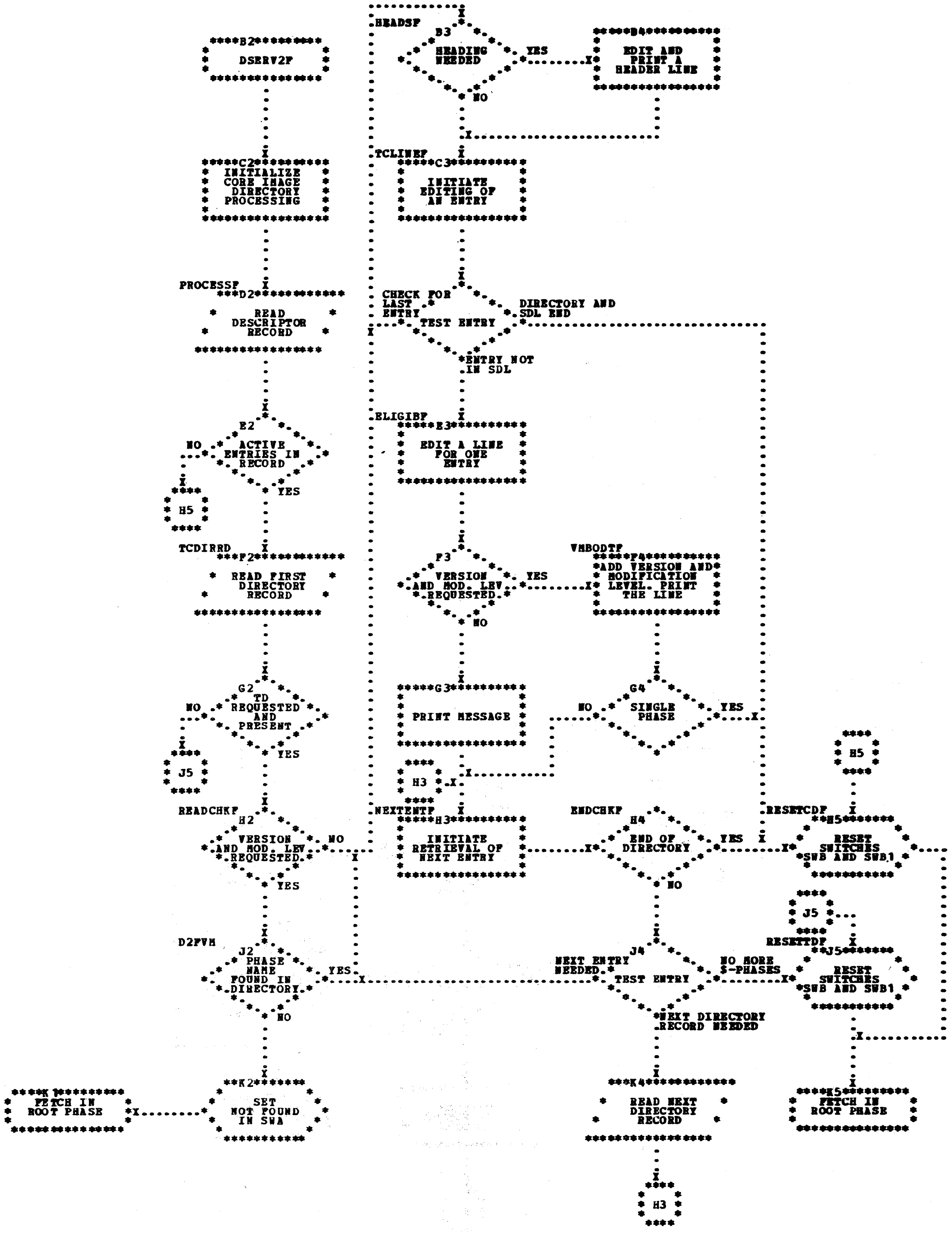


Chart RD. DSERV3
Refer to Chart 75

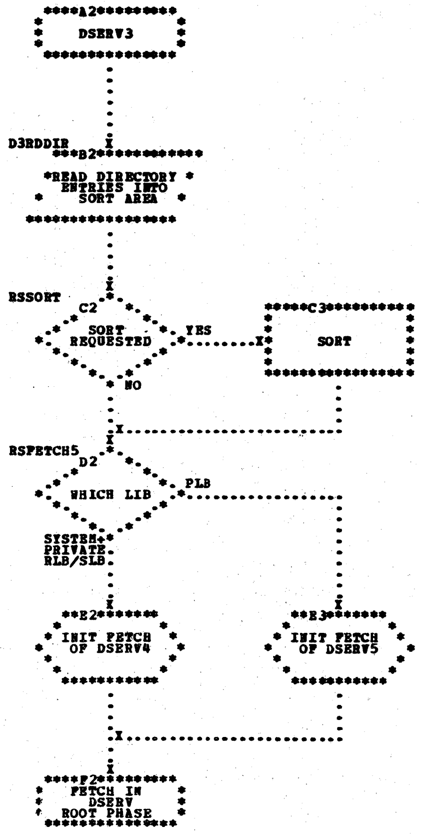


Chart RE. DSERV3F
Refer to Chart 76

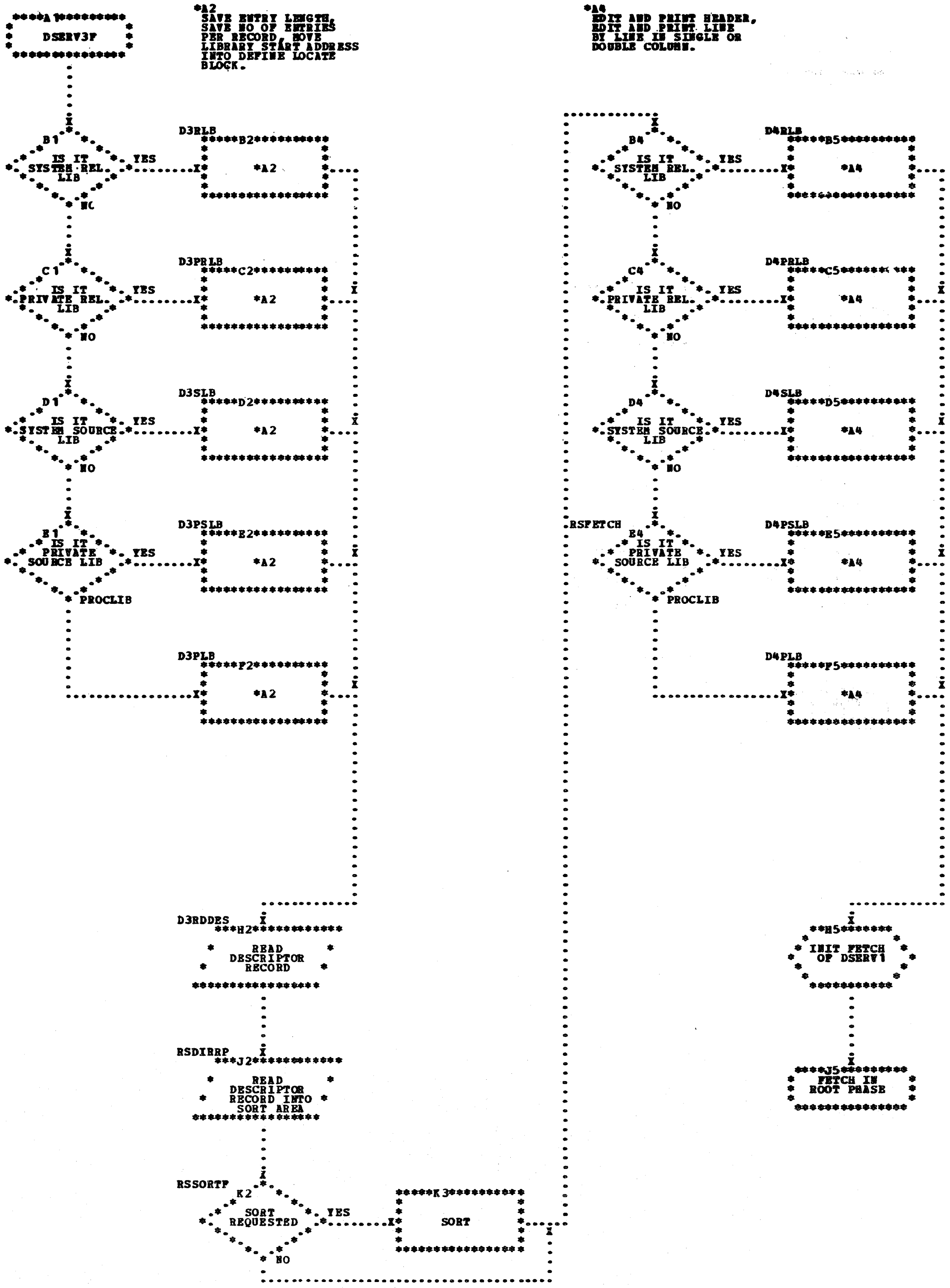


Chart RF. DSERV4 and DSERV5
Refer to Chart 75

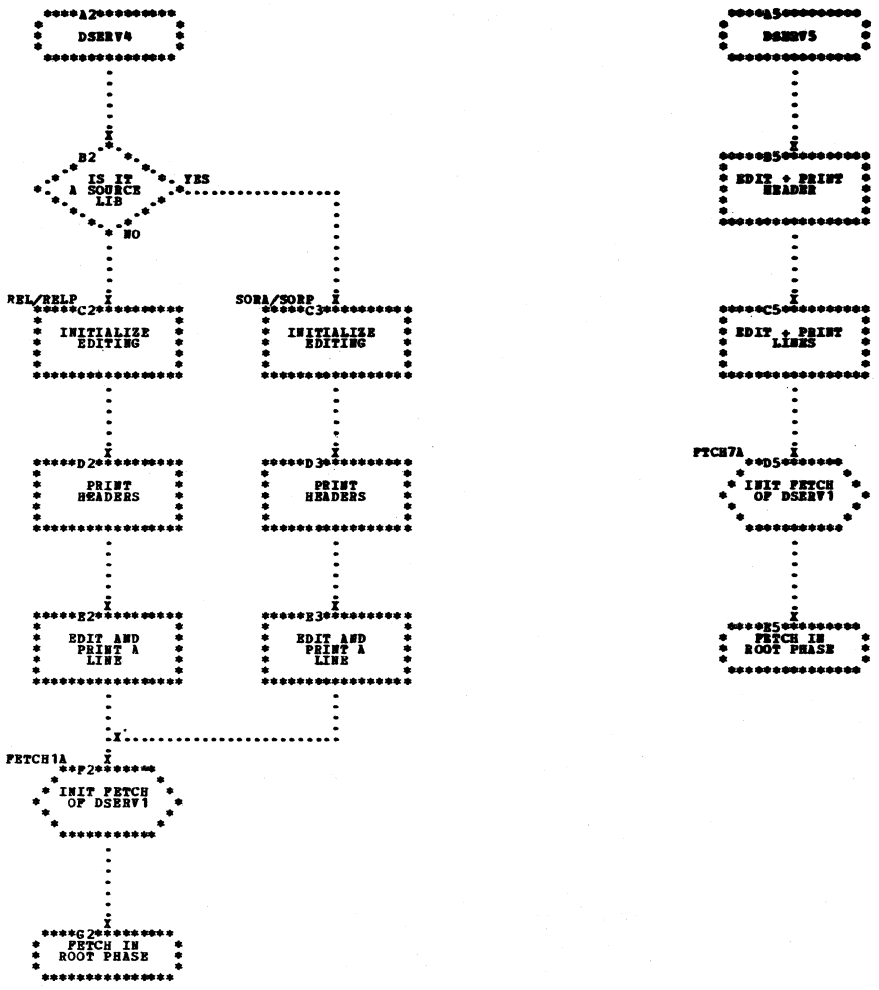
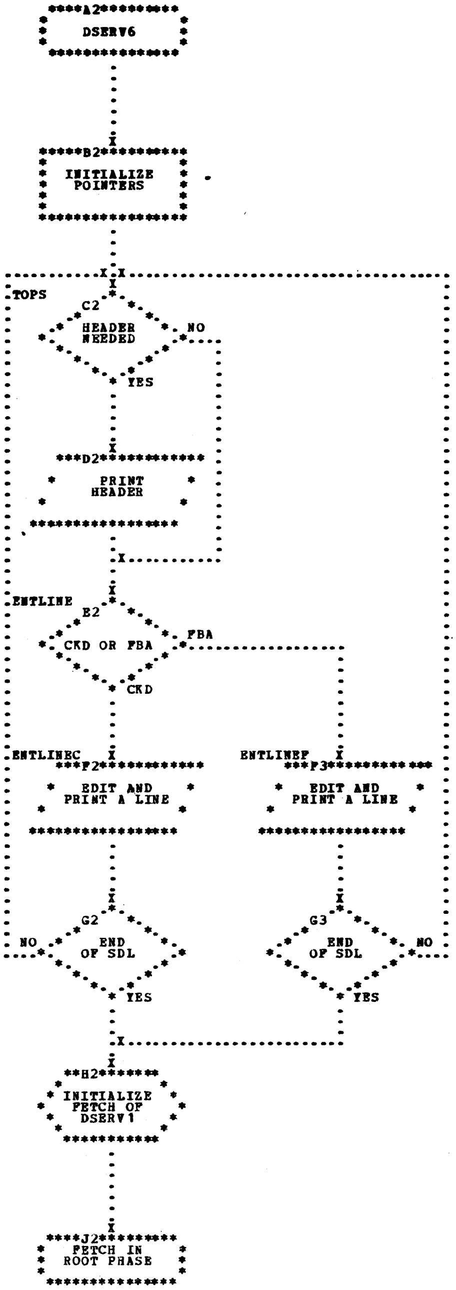


Chart RG. DSERV6
Refer to Chart 76



DATA AREAS

This section shows the formats of data areas used by the librarian programs and of the libraries on SYSRES. It contains:

    INITABLE (MAINT Root Phase)
    Reallocation Tables (MAINTA/F)
    Library Status Table ($LIBSTAT)
    STOWTAB and TABIN Formats (for $MAINDIR/$MAINDIF)
    Switches for various phases
    SYSRES Formats
    Library Data Formats

INITABLE FROM MAINT ROOT PHASE

```
*-----------------------------------------------------------------------
*        CORE IMAGE LIBRARY (CIL) CHARACTERISTICS
*-----------------------------------------------------------------------
CPDEVTYP DS   XL1        DEVICE TYPE CODE AS IN PUB
CODEVTYP DS   XL1        DEVICE TYPE CODE AS IN DTFCP
         DS   0XL5       DEVICE CHARACTERISTICS
CBYBLK   DS   XL2        BYTES IN FBA BLOCK
         ORG  CBYBLK
CTRKS    DS   XL1        NUMBER OF TRACKS PER CYLINDER
CTRKNO   DS   XL1        NUMBER OF TRACKS PER CYLINDER - 1
CIDREC   DS   XL1        RECORDS/TRACK IN CORE IMAGE DIRECTOTY
CILREC   DS   XL1        RECORDS/TRACK IN CORE IMAGE LIBRARY
RPSCLB   DS   XL1        RPS FLAG FOR CIL RPS SUPPORT
CILOGUNT DS   XL1        LOGICAL UNIT FOR CIL (INIT SYSRES)
CILEXTNT DS   XL4        DISK ADDRESS OF CORE IMAGE LIBRARY
*-----------------------------------------------------------------------
*        RELOCATABLE LIBRARY (RL) CHARACTERISTICS
*-----------------------------------------------------------------------
RPDEVTYP DS   XL1        DEVICE TYPE CODE AS IN PUB
RDDEVTYP DS   XL1        DEVICE TYPE CODE AS IN DTFCP
         DS   0XL5       DEVICE CHARACTERISTICS
RBYBLK   DS   XL2        BYTES IN FBA BLOCK
         ORG  RBYBLK
RTRKS    DS   XL1        NUMBER OF TRACKS PER CYLINDER
RTRKNO   DS   XL1        NUMBER OF TRACKS PER CYLINDER - 1
RDREC    DS   XL1        RECORDS/TRACK IN RELOCATABLE DIRECTORY
RLREC    DS   XL1        RECORDS/TRACK IN RELOCATABLE LIBRARY
RPSRL    DS   XL1        RPS FLAG FOR RL RPS SUPPORT
RLLOGUNT DS   XL1        LOGICAL UNIT FOR RL (INIT SYSRES)
RLEXTNT  DS   XL4        DISK ADDRESS OF RELOCATABLE LIBRARY
*-----------------------------------------------------------------------
```

Figure 13. INITABLE from MAINT Root Phase (Part 1 of 2)

```
*----------------------------------------------------------------------
*        SOURCE STATEMENT LIBRARY (SSL) CHARACTERISTICS
*----------------------------------------------------------------------
SPDEVTYP DS   XL1        DEVICE TYPE CODE AS IN PUB
SDDEVTYP DS   XL1        DEVICE TYPE CODE AS IN DTFCP
         DS   0XL5       DEVICE CHARACTERISTICS
SBYBLK   DS   XL2        BYTES IN FBA BLOCK
         ORG  SBYBLK
STRKS    DS   XL1        NUMBER OF TRACKS PER CYLINDER
STRKNO   DS   XL1        NUMBER OF TRACKS PER CYLINDER - 1
SSDREC   DS   XL1        RECORDS/TRACK IN SOURCE STATEMENT DIR.
SSLREC   DS   XL1        RECORDS/TRACK IN SOURCE STATEMENT LIBRARY
RPSSL    DS   XL1        RPS FLAG FOR SSL RPS SUPPORT
SSLOGUNT DS   XL1        LOGICAL UNIT FOR SSL (INIT SYSRES)
SSLEXTNT DS   XL4        DISK ADDRESS OF SOURCE STATEMENT LIBRARY
*----------------------------------------------------------------------
*        PROCEDURE LIBRARY (PL) CHARACTERISTICS
*----------------------------------------------------------------------
RPDEVTYP DS   XL1        DEVICE TYPE CODE AS IN PUB
RDDEVTYP DS   XL1        DEVICE TYPE CODE AS IN DTFCP
         DS   0XL5       DEVICE CHARACTERISTICS
PBYBLK   DS   XL2        BYTES IN FBA BLOCK
         ORG  PBYBLK
PTRKS    DS   XL1        NUMBER OF TRACKS PER CYLINDER
PTRKNO   DS   XL1        NUMBER OF TRACKS PER CYLINDER - 1
PDREC    DS   XL1        RECORDS/TRACK IN RELOCATABLE DIRECTORY
PLREC    DS   XL1        RECORDS/TRACK IN RELOCATABLE LIBRARY
RPSRL    DS   XL1        RPS FLAG FOR RL RPS SUPPORT
PLLOGUNT DS   XL1        LOGICAL UNIT FOR RL (INIT SYSRES)
PLEXTNT  DS   XL4        DISK ADDRESS OF RELOCATABLE LIBRARY
*----------------------------------------------------------------------
*        GENERAL CONSTANTS
*----------------------------------------------------------------------
CNSW     DS   XL1        CONDENSE SWITCH USED IN MAINTCN
SWITCH   DS   XL1        INFORMATION BYTE
*                          X'80'  :  CALL MAINTA AFTER CONDENSE
*                          X'40'  :  CANCEL AFTER GIVING STATUS REPORT
*                          X'20'  :  SYSLST NOT ASSIGNED
*                          X'10'  :  STOWTABLE CREATED IN MAINTDR
*                          X'08'  :  81-BYTE SYSIPT RECORDS
*                          X'04'  :  CONTINUATION CARD READ
*                          X'02'  :  SOME MAINT OPERAT. FAILED
LIBLIM   DS   XL4        MAX CYLINDERS/BLOCKS FOR LIBRARY (CKD)
*                        OR DEVICE CAPACITY IN BLOCKS (FBA)
INITED   EQU  *
```

Figure 13. INITABLE from MAINT Root Phase (Part 2 of 2)

REALLOCATION TABLES FROM MAINTA OR MAINTAF

CDOSA	DISPLACEMENT (DECIMAL)		DIRECTORY TABLE
RDOSA		0	Old starting address (CCHH)
SDOSA		4	New starting address (CCHH)
PDOSA		8	Number of tracks used
		10	Number of tracks allocated
		12	Number of blocks used
		14	Displacement in no. of tracks (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			MEMBER SPACE TABLE
RLOSA	0	28	Old starting address (CCHH)
SLOSA	4	32	New starting address (CCHH)
PDOSA	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	Blocks 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 begins

Note 1: This is the number of tracks that must be added to or subtracted from the old disk address to get the new disk address (the difference between the values at displacements 0 and 4).

Note 2: 0 if the value at displacement 14 is 0.
1 if the value at diaplacement 14 is positive.
2 if the value at displacement 14 is negative.

Figure 14. MAINTA Reallocation Table

	TOTAL TABLE	SLOT	LENGTH	
REALDSCT				
OLDDIRA	0	0	4	OLD DIRECTORY START ADDR.
NEWDIRA	4	4	4	NEW DIRECTORY START ADDR.
DIRBLKU	8	8	4	DIRECTORY REC. USED <IN BLOCKS>
DIRBLKN	12	12	4	DIRECTORY REC. NEW <IN BLOCKS>
DIRMVADR	16	16	4	ADDR. FOR I/O UPERATION
*				INITIALISED DEPENDING ON MOVE
DIRMVUNT	20	20	4	NUMBER OF BLOCKS IN ONE I/O
DIRLEN	24	24	4	LENGTH TO MOVE
DIRDEL	28	28	4	RELOCATION FACTOR FOR DIR. ENT
RTABCLS	32	32	1	SWITCH BYTE IN TABLE
PRESENT				LIBRARY PRESENT ON OLD SYSR.
				(CORE IMAGE LIBR. ALWAYS PRES.
NOACTIV				LIBRARY CONTAINS NO ACTIVES
PASSONE				MOVE LIBRARY ON FIRST PASS
PASSTWO				MOVE LIBRARY ON SECOND PASS
STAYS				DIRECT. OR MEMBER SPACE STAYS
NEW				LIBRARY IS BEING ALLOCATED
RTABCLT	33	33	1	LIBRARY TYPE INDICATOR
RTABCLI				CORE IMAGE
RTABRLI				RELOCATABLE
RTABSLI				SOURCE
RTABPLI				PROCEDURE
RTABDRI				DIRECTORY
OLDMBRA	36	0	4	OLD MEMBER SP. START ADDRESS
NEWMBRA	40	4	4	NEW MEMBER SP. START ADDRESS
MBRBLKU	44	8	4	USED MEMBER RECORDS <IN BLKS>
MBRBLKN	48	12	4	NEW MEMBER BLOCKS
MBRMVADR	52	16	4	CURRENT ADDRESS IN I/O
MBRMVUNT	56	20	4	BLOCKS IN ONE I/O
MBRLEN	60	24	4	LENGTH FOR READ/WRITE CCW
	64	28	4	NOT USED
	68	32	1	SWITCH BYTE AS ABOVE
	69	33	1	LIBRARY TYPE INDICATOR AS ABOVE

Figure 15. MAINTAF Reallocation Table

LIBRARY STATUS TABLE FROM $LIBSTAT

Field Name	Size	Contents
NROFENTR (First entry)	2 bytes	Number of table entries following
Each next entry contains the following items:		
STATCU	2 bytes	Logical unit of library, may be: X'0006' – SYSRES X'0007' – SYSSLB X'0008' – SYSRLB X'000B' – SYSCLB X'0102' – SYS002 X'0103' – SYS003
STATTYPE	1 byte	Type of library, may be: C – Core Image S – Source Statement P – Procedure R – Relocatable
BBCCHH	6 bytes	Disk address of first directory track for CKD
or		
STTBNR	2 bytes 4 bytes	unused block address for FBA
STTARC	1 byte	'C' = CKD 'F' = FBA
STTSW	1 byte	switch byte X'80': print condense limit message if condense limit has been reached

Figure 16. Library Status Table

THE STOW TABLE AND THE TABIN ARRAY (FOR $MAINDIR/$MAINDIF)

The Stow Table consists of a header and as many entries as there are requests to be filled by $MAINDIR/$MAINDIF.

CKD FORMATS IN STOWTAB

For CKD, the fields in the header are:

beginning byte	0	2	8	10	12
field name	LOGUNIT	DIRADR	NROFCALL	NROFENTR	

LOGUNIT: Indicates the logical unit:
X'0006' for system core image library (SCIL)
X'000B' for private core image library (PCIL)

DIRADR: Contains the start address of the directory.

NROFCALL: Number of requests already satisfied by $MAINDIR + 1 (0 when called by IPL or by job control).

The leftmost bit of NROFCALL is named OWPMSHP. It is set to 1, if

- an error occurs during condense,
- the message 3M33I or 3M54I has been issued,
- a return code >8 was given back by $MAINDIR

NROFENTR: number of entries for the current request.

For CKD, the fields in the entry are:

beginning byte	0	8	16	17	18	30
field name	STOWNAME	NEWNAME		STOWTYPE		

STOWNAME: Contains the phase name.

NEWNAME: Contains the new phase name for a rename request. Otherwise, the field is split into two subfields as follows:
Field STOWTTR (bytes 8 through 10). Contains the disk address of the phase.
Field STOWN (byte 11). Contains the number of halfwords of new directory entry.

STOWTYPE: Indicates the requested service:
as for FBA, see below

FIXED BLOCK FORMATS IN STOWTAB

For FBA, the fields in the header are:

beginning byte	0	2	4	8	10	12
field name	OWPLGUN		OWPLBEG	OWPNOCL	OWPNOEN	

OWPLGUN: Indicates the logical unit:
X'0006' for system core image library (SCIL).
X'000B' for private core image library (PCIL).
X'0102' for new SCIL.
X'0103' for new PCIL.

OWPLBEG: Contains the library start address.

OWPNOCL: Number of requests already filled. It is zero, when the phase is called by IPL or job control.

The leftmost bit of OWPNOCL is named OWPMSHP.
It is set to 1, if

- an error occurs during condense
- the message 3M33I or 3M54I has been issued,
- a return code >8 was given back by $MAINDIF.

OWPNOEN: Number of entries.

<u>For FBA, the fields in the entry are</u>:

```
beginning
byte          0         8          16       17          18        26         27      30
             ┌──────────────────────────────────────────────────────────────────────────┐
field name   | OWPNAME | OWPNEWN |         | OWPTYPE |           | OWPNRB |              |
             └──────────────────────────────────────────────────────────────────────────┘
```

OWPNAME: Contains the phase name.

OWPNEWN: Contains the new phase name for a rename request. Otherwise, the field is split into two subfields as follows:
Field OWPADR (bytes 8 through 11). Contains the address of the first text block.
Field OWPNTX (bytes 12 and 13). Contains the number of text blocks.

OWPTYPE: Indicates the requested service:
C'B' = load (build) the phase in the SVA.
C'C' = catalog the phase and delete the previous entry if one exists.
C'D' = delete the entry with the indicated phase name.
C'E' = use the entry in building a second level directory.
C'F' = the last call for a condense operation to update the library descriptor record.
C'I' = load the phase in the SVA (request comes from IPL).
C'L' = catalog the phase temporarily in the link area.
C'M' = load the self-relocating phase in the SVA.
C'N' = include the CIL entry in the SDL.
C'R' = rename the phase.
C'U' = update to reflect a library condense operation.
C'X' = update to reflect a library reallocation operation.

In addition to the above listed characters, OWPTYPE may contain characters set for internal control purpose as follows:

C'A' instead of C'B'
C'K' instead of C'M'
C'S' instead of C'R'

OWPNRB: Indicates the number of additional RLD blocks.

Note: The overall format of a Stow Table entry is similar to that of a core image directory entry.

THE TABIN ARRAY

For each entry in the Stow Table there is a rearranged entry in the TABIN array for easier processing. The format of the TABIN entry is:

beginning byte	0	8	9	12	14
field name	1	2	3	4	

1. TABNAME: Phase name as extracted from the corresponding Stow Table entry. Subfields may be accessed as follows:
   TNAMSHRT -- bytes 0-3
   TNAMLAST -- bytes 4-7

2. TABTYPE: Copy of byte OWPTYPE in corresponding Stow Table entry.

3. TABADR: Contains a pointer to the corresponding Stow Table entry.

4. TABMSG: Used as message indicator:

Byte	Bits	Meaning when 1
12	0	Message 3M33I is to be generated.
	1-7	Reserved.
13	0-7	Reserved.

SWITCHES FOR VARIOUS PHASES

SWITCHES FOR $MAINDIR

$MAINDIR uses the bits of three consecutive bytes in the data area as switches. The bits of these bytes which are declared as SWITCHES, are accessed symbolically for the purposes indicated below:

Byte	Bit	Name	Explanation
0	0	SEQUENCE	Set by EXTR to OK=0 If the entries in the Stow Table are in the correct sequence. INERROR=1 If these entries are out of sequence.
	1	FOUND	Set by LOOKUP to YES=1 If a matching phase name was found. NO=0 If a phase name higher than the search name was found.
	2	SWOPEN	Set by MSG to YES=1 To indicate that SYSLST has been opened.
	3	SWLOG	Set by MSG to YES=1 To indicate that the SYSLOG CCB and CCW have been built.
	4	SWASGN	Set by INITSYSL to YES=1 To indicate that SYSLST is assigned.

Byte	Bit	Name	Explanation
	5	SWFIRST	Used by GETENTRY: if YES=1, the directory input will be initialized.
	6	SWINPTR	Used by PUTENTRY: if YES=1, INPTR --> DIRENTRY is used as input
	7	SWREALLC	Used by GETENTRY: YES=1, indicates a reallocate run.
1	0	SWDRYRUN	Used by PUTENTRY: YES=1 indicates that the directory should not be modified (dry run)
	1	SWEQUAL	Set by CHKEQL to YES=1 If a matching phase name was found in the CIL directory
	2	SWSDLEQL	Set by CHKEQL to YES=1 If a matching phase name was found in the SDL.
	3	SWDIST	Set by KEEPDIST to YES=1 To indicate to GETINP that CCW string, disk address, IOREG and INPTR must not be modified.
	4	SWSVAFUL	Set by LOAD to YES=1 To indicate that no more phases can be loaded into the SVA.
	5	SWRASEQL	Set by RASEQL to YES=1 To indicate that the phase is a RAS transient.
	6	SWM91I	Set by UPDSLD to ON=1 To indicate that message 3M91I has to be displayed.
	7	SWM51I	Set by PUTOUTP to ON=1 To indicate that message 3M51I has to be displayed.
2	0	SWASSGN2	Set by INITSYSL and FINISH to ON=1 To indicate that message 3M92I has to be displayed on SYSLST.
	1	SWALL	Set by BLOWUP to YES=1 To indicate a DELETC ALL request.
	2	SWALLC	Set by BLOWUP to ON=1 To indicate a DELETC ALL request.
	3	SWMSHP	Set by UPDATE,RENM,SVAMSG,GENMSG and FINISH to ON=1 To indicate that OWPMSHP in the Stow Table has to be switched on.
	4-7	reserved	

SWITCHES FOR $MAINDIF

$MAINDIF uses the bits of two consecutive bytes in the data area as switches. The bits of these bytes, which are declared as SWITCHES, are accessed symbolically for the purposes indicated below:

Byte	Bit	Name	Explanation
0	0	SEQUENCE	Set by EXTR to OK=0 If the entries in the Stow Table are in the correct sequence. INERROR=1 if these entries are out of sequence.
	1	SWEQUAL	Set by CHKEQL to YES=1 If a matching phase name was found in the CIL directory.
	2	SWSDLEQL	Set by CHKEQL to YES=1 If a matching phase name was found in the SDL.
	3	SWRASEQL	Set by CHKEQL to YES=1 If a matching phase name was found in the RAS load list.
	4	SWASGN	Set by INITSYSL to YES=1 To indicate that SYSLST is assigned.
	5	SWOPEN	Set by MSG to YES=1 To indicate that SYSLST has been opened.
	6	SWLOG	Set by MSG to YES=1 To indicate that the SYSLOG CCB and CCW have been built.
	7	SWSVAFUL	Set by LOAD to ON=1 When the SVA becomes full.
1	0	FOUND	Set by LOOKUP to YES=1 If a matching phase name was found. NO=0 If a phase name higher than the search name was found.
	1	SWMSHP	Set by UPDAT,RENM,SVAMSG,GEWMSG and FINISH to YES=1 To indicate that OWPMSHP in the Stow Table has to be switched on.
	2-7	reserved	

SWITCHES FOR THE DSERV PROGRAM

SWITCHES TO CONTROL THE PROGRAM FLOW

All the switch bytes "....DVTP" come from the PUB device type byte and are used to set the LOADSW to CKDPH and/or FBAPH for loading the twin phase(s). In addition "SRESDVTP" is used in DSERV1 to select the twin phase subroutines for SYSRES/SRLB/SSLB/PLB whether on CKD or FBA.

```
SRESDVTP DC    X'00'            SYSRES PUB DEV TYPE
PCILDVTP DC    X'00'            PCIL   PUB DEV TYPE
PRLBDVTP DC    X'00'            PRLB   PUB DEV TYPE
PSLBDVTP DC    X'00'            PSLB   PUB DEV TYPE
FBAPUB   EQU   X'90'            FBA    PUB DEV TYPE CODE
```

The switch bytes "..ARC" are set by the root phase testing the PUB device type byte and get the value C'C' for CKD and C'F' for FBA.

```
RESARC   DC    X'00'
   Used in DSERVC/F to prepare status table entry for
   SCIL/SRLB/SSLB/PLB.
   Used in DSERV1 to initialize FETCH of overlay phases on
   FBA or CKD for SRLB/SSLB/PLB.
   Used in DSERV6 to select header printing for FBA or CKD.
CILARC   DC    X'00'
   Used in DSERV1 to prepare status table entry for PCIL.
   Used in DSERV1 to initialize FETCH of overlay phase
   DSERV2 or DSERV2F.
LBARC    DC    X'00'
   Used in DSERV1 to prepare status table entry for PRLB.
   Used in DSERV1 to initialize FETCH of overlay phase
   DSERV3 or DSERV3F.
SLBARC   DC    X'00'
   Used in DSERV1 to prepare status table entry for PSLB.
   Used in DSERV1 to initialize FETCH of overlay phase
   DSERV3 or DSERV3F.
```

SWA AND SWA1   NOTES THE SPECIFIED DSPLY OPTION

```
SWA      DC    X'0'
         ENTRY SWA
VMIND    EQU   X'80'                 VER AND MOD LEVEL IND
HEADING  EQU   X'40'                 HEADER NEEDED IND
NONAME   EQU   X'20'                 PHASE NAME NOT FOUND IND
ALLIND   EQU   X'1E'                 DISPLAY ALL INDICATORS
PDIND    EQU   X'10'                 DISPLAY PROCEDURE DIRECTORY
SDIND    EQU   X'08'                 DISPLAY SOURCE STMNT DIRECTORY
RDIND    EQU   X'04'                 DISPLAY RELOCATABLE DIRECTORY
CDIND    EQU   X'02'                 DISPLAY CORE IMAGE DIRECTORY
TDIND    EQU   X'01'                 DISPLAY TRANSIENT DIRECTORY

SWA1     DC    X'0'                  DISPLAY SWITCH A1

SDLIND   EQU   X'80'                 DISPLAY SYS. DIR. LIST
```

SWB AND SWB1   CONTROLS THE CALL OF OVERLAY PHASES

```
SWB       DC     X'0'                    SWITCH BYTE B

SYSTD     EQU    X'80'                   DISPLAY TRANSIENT DIRECTORY
SYSCL     EQU    X'40'                   DISPLAY SYSTEM CORE IMAGE DIRECTORY
SYSRL     EQU    X'20'                   DISPLAY SYSTEM REL DIRECTORY
SYSSL     EQU    X'10'                   DISPLAY SYSTEM SOURCE DIRECTORY
SYSPL     EQU    X'08'                   DISPLAY SYSTEM PROC. DIRECTORY
SYSSDL    EQU    X'04'                   DISPLAY SYSTEM DIRECTORY LIST
ANYMORE   EQU    X'FC'                   ANY DIR. DISPLAY MASK

SWB1      DC     X'0'                    DISPLAY SWITCH B1

PTD       EQU    X'80'                   PRIVATE TRANSIENT DIR INDICATOR
PCLB      EQU    X'40'                   DISPLAY PRIVATE CORE IMAGE DIRECTORY
PRLB      EQU    X'20'                   DISPLAY PRIVATE REL DIRECTORY
PSLB      EQU    X'10'                   DISPLAY PRIVATE SOURCE DIRECTORY
RESERVE   EQU    X'08'                   RESERVED
FIRST     EQU    X'01'                   FIRST TIME INDICATOR
```

STATUS AND ERROR INDICATOR SWITCHES

```
SWC       DC     X'0'                    SWITCH BYTE C

FULLTBL   EQU    X'80'                   FULL TABLE INDICATOR
RELOOP    EQU    X'40'                   GO THROUGH SORT LOOP AGAIN
ONEIND    EQU    X'10'                   DISPLAY SINGLE PHASE
LEVELNO   EQU    X'08'                   NEED LEVEL NO. FROM NEXT RECORD
SKIPNAME  EQU    X'04'                   DO NOT SCAN PHASE NAME INDICATOR
DUMYCNT   EQU    X'02'                   DUMY LOOP-COUNT RECDS LEFT INDICATOR
DISPLACE  EQU    X'01'                   DISPLACEMENT SEPECIFIED INDICATOR

SWD       DS     X'0'                    SWITCH BYTE D

SORT      EQU    X'80'                   ALPHANUMERICAL DISPLAY
SVADIR    EQU    X'40'                   SVA PRESENT INDICATOR
PCST      EQU    X'20'                   PRIVATE CORE IMAGE STATUS INDICATOR
PRST      EQU    X'10'                   PRIVATE REL STATUS INDICATOR
PSST      EQU    X'08'                   PRIVATE SOURCE STATUS INDICATOR
SECOND    EQU    X'04'                   SECOND TIME INDICATOR
DIREND    EQU    X'02'                   END OF DIRECTORY REACHED

SWE       DC     X'00'                   SWITCH BYTE E

ERR3      EQU    X'80'                   NO SYSTEM REL ACTIVE ENTRIES 4-0
ERR4      EQU    X'40'                   NO SYSTEM SOR ACTIVE ENTRIES
ERR5      EQU    X'20'                   NO PRI ACTIVE ENTRIES
ERR6      EQU    X'10'                   NO PRI REL ACTIVE ENTRIES
ERR7      EQU    X'08'                   NO PRI SOR ACTIVE ENTRIES
ERR8      EQU    X'04'                   NO PRIV TD ACTIVE ENTRIES
ERR9      EQU    X'02'                   NO SYSTEM PROC LIBRARY

SWE1      DC     X'0'                    SWITCH BYTE E1

ERR11     EQU    X'80'                   NO SDL PRESENT
```

```
IPTSW    DC     X'00'                       INPUT SWITCH BYTE
IPT81BYT EQU    X'80'                       81 BYTE SYSIPT INDICATOR


LOADSW   DC     X'00'                       CONTROL OF LOADING FOR
*                                             DSERVC AND/OR DSERVF
FBAPH    EQU    X'80'                       DSERVF NECESSARY
CKDPH    EQU    X'40'                       DSERVC NECESSARY

SLIBSW   DC     X'00'                       ARE THERE SYSTEM LIB'S

SRLB     EQU    X'20'                       SYSTEM RLB PRESENT
SSLB     EQU    X'10'                       SYSTEM SLB PRESENT
SPLB     EQU    X'08'                              PLB PRESENT
```

SYSRES FORMATS

This section will first given an overview of the different parts of a SYSRES file and their distribution on a disk of CKD or FBA type and then describe in detail the areas important for the librarian programs, that is, the system directory and the various library types.

SYSRES OVERVIEW

Figures 17 and 18 present the organization of a disk resident system as shipped by IBM. The SYSRES file may be on an

IBM 2314/19	(20 tracks per cylinder),
3330/3333	(19 tracks per cylinder),
3340	(12 tracks per cylinder),
3350	(30 tracks per cylinder, or a
3370	(558000 blocks)
3310	(126016 blocks).

SYSRES is contained in a continuous area at the beginning of the disk pack. This disk pack is, by extension, also sometimes called SYSRES. Certain areas are predefined.

Figure 17 shows the layout of SYSRES on CKD.

Component		Starting Disk Address			Number of Tracks (Alloc.)	R=Required O=Optional
		CC	HH	R		
IPL Record	(Phase $$A$IPL1)	00	00	1	1	R
IPL Record		00	00	2		R
System Volume Label		00	00	3		R
User Volume Label		00	00	4		O
System Directory	Record 1	00	01	1	1	R
	Record 2	00	01	2		R
	Record 3	00	01	3		R
	Record 4	00	01	4		R
IPL Records (Phase $$A$PLBK)		00	01	5		R
Core Image Directory	Cataloged Phases Linked Phase	00	02		*	R
Core Image Library Member Space		X	Y+1	1	*	R
Relocatable Directory		Z+1	00	1	*	O
Relocatable Library Member Space		X	Y+1	1	*	O
Source Statement Directory		Z+1	00	1	*	O
Source Statement Library Member Space		X	Y+1	1	*	O
Procedure Directory		Z+1	00	1	*	O
Procedure Library Member Space		X	Y+1	1	*	O
Label Information Area		Z+1	00	1	device dependent **	R

* Allocation Dependent on User Requirements
X = Ending CC of the Preceding Directory
Y = Ending HH of the Preceding Directory
Z = Ending CC of the Preceding Library

** Allocation Dependent on Number of Tracks per Cylinder:

2314/2319:	20 Tracks	(one Cylinder)
3333/3330:	19 Tracks	(one Cylinder)
3340:	24 Tracks	(two Cylinders)
3350:	30 Tracks	(one Cylinder)

Figure 17. Layout of SYSRES on a CKD Device

Figure 18 shows the layout of SYSRES on FBA.

Component	Starting Disk Address BLock Number	Number of Blocks	R=Required O=Optional
IPL Records (Phase $$A$IPLO)	0	1	R
System Volume Label[1]	1	1	R
System Directory	2	1	R
IPL Retrieval Program (Phase $$A$PLBF)	3	7	R
Core Image Directory	10	*	R
Core Image Library Member Space	X+1	*	R
Relocatable Directory	Y+1	*	O
Relocatable Library Member Space	X+1	*	O
Source Statement Directory	Y+1	*	O
Source Statement Library Member Space	X+1	*	O
Procedure Directory	Y+1	*	O
Procedure Library Member Space	X+1	*	O
Label Information Area	Y+1	200[2]	R

* = Allocation dependent on user requirements
X = Last block of preceding directory
Y = Last block of preceding library member space

[1] Optional user volume labels if written will be in the same block following the system volume label.

[2] Using the Restore program you may allocate a label information area different than the default size of 200 blocks.

Figure 18. Layout of SYSRES on a Fixed Block Device

Figures 17 and 18 show the correspondence in the layout of SYSRES on different types on disks.

IPL Records 1 and 2 (CKD) or block 1 (FBA)

This area contains the initial program load bootstrap program which causes the IPL retrieval program to be read from SYSRES and loaded into real storage.

Volume Labels

This area contains the address of the volume table of contents (VTOC) established when the disk pack was initialized. The VTOC can be located on any cylinder outside the SYSRES File.

System Directory

This area contains records that show the start addresses of the library directories in the system, the number of partitions of the supervisor last IPLed, and the start address of the label information area.

The Libraries

For CKD, the directory and the member space of each library starts on a new track and the library uses all of the last allocated cylinder. For FBA, the library and the member space start on block boundary and the library uses all of the blocks allocated.

The core image library contains for example the following programs in load format:

system control programs
linkage editor and librarian
problem determination and system debugging aids
Programming Languages (Assembler, PL/I, and so on)
User Programs
IBM Program Products

The relocatable library contains programs in relocatable format (language translator output). All programs supplied in the core image library (except the transients) are also contained in this area. In addition, this area can contain other programs in relocatable format.

The source statement library contains blocks in source language format. The books supplied by IBM are macro definitions in the assembler sublibrary.

The procedure library contains procedures in card image format.

The Label Information Area

This area is reserved to contain standard, partition standard, and user labels for background and foreground partitions.

THE SYSTEM DIRECTORY

The start address of the system directory is:

for CKD	cylinder 0 track 01 record 1
for FBA	block 0002

For CKD, the system directory consists of four records of 80 bytes each, as shown in Figure 19. For FBA, it consists of one record of 66 bytes. The rest of the block is empty.

System Directory Records:

1 Record One

field	bytes	
1	0-6	Start Address of the Core Image Library in the format BBCCHHR.
2	7-75	Reserved.
3	76-77	Number of Label Cylinders
4	78-79	Address of the Label Area

2 Record Two

field	bytes	
1	0-6	Start Address of the Relocatable Library in the format BBCCHHR.
2	7-79	Reserved.

3 Record Three

field	bytes	
1	0-6	Start Address of the Source Statement Library in the format BBCCHHR.
2	7-79	Reserved.

4 Record Four

field	bytes	
1	0-6	Start Address of the Procedure Library in the format BBCCHHR.
2	7-59	Reserved.
3	60-65	Number of Partitions of Supervisor last IPLed.
4	65-79	Reserved.

Figure 19. System Directory on a CKD Device

Figure 20 shows the System Directory Format on an FBA device. The System Directory occupies the first 66 bytes of block 2 of the device. The library addresses are 4 digit block numbers.

bytes	
0-3	Start address of the core image library.
4-11	Zeros.
12-15	Start address of the relocatable library.
16-23	Zeros.
24-27	Start address of the source statement library.
28-35	Zeros.
36-39	Start address of the procedure library.
40-47	Zeros.
48-51	Start address of the label information area.
52-55	Zeros.
56-59	Address of the last block of the label information area relative to the beginning of that area.
60-65	Number of partitions of the supervisor last IPLed.

Figure 20. System Directory on a Fixed Block Device

LIBRARIES ON CKD DEVICES

CORE IMAGE LIBRARY

The directory of a core image library has 2 or more tracks. Since the size of a track is device dependent, the number of 256-byte records per track is

16 for a 3340
17 for a 2314/19
28 for a 3330/3333
36 for a 3350

Each record is preceded by a key containing the phase name from the last entry in this record. A directory record has the following format:

```
 0                                      256
┌──────────────────────────────────────────┐
| LL | entry 1 | entry 2 | ... entry n | U |
└──────────────────────────────────────────┘
```

LL   = the number of bytes used including LL
entry = describes one phase in the library
U   = unused space

The first entry of the directory, the library descriptor, has 58 bytes, the last entry has 12 bytes. All other directory entries have 18 to 30 bytes.

The first entry of the first record is called the library descriptor:

```
beginning byte:  0   8   11  12  14  16  18  20  22  24  26  28  32  36  40  42  50  58
                ┌─────────────────────────────────────────────────────────────────────┐
field number:   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10| 11| 12| 13| 14| 15| 16| 17|
                └─────────────────────────────────────────────────────────────────────┘
```

1 Contains zeros or, after incomplete condense, X'00' ||C'DSTROYD'.
2 Address of the first deleted library record, otherwise X'FFFFFF'.
3 Number of halfword containing user data after this byte
4 Number of tracks per cylinder
5 Number of directory tracks
6 Number of library cylinders
7 Number of active entries in the directory
8 Number of directory records per track
9 Number of directory records used
10 Number of directory records available
11 Number of member records per track
12 Number of member records used
13 Number of member records deleted
14 Number of member records available
15 Number of member records for automatic condense
16 Date and time when the core image library has been updated. (Set by store clock instruction), set to zero if the clock is defective.
17 Reserved.

The directory entry describing a phase of the library has these fields:

beginning byte:	0	8	11	12	14	16	17	18	21	24	26	27	30
field number:	1	2	3	4	5	6	7	8	9	10	11	12	13

1 Phase name

2 Track address and record number of phase, relative to the beginning of the directory.

3 Number of halfwords containing user data after this byte.

4 Number of text records.

5 Number of text bytes in the last text record.

6 Switch indicating type of phase.
The settings X'80', X'40', and X'20' exist on disk and resident entries.
The settings X'10', X'08', X'04', X'02', and X'01' are reserved for storage resident entries, e.g. SDL resident or partition resident.
X'80': selfrelocating phase
X'40': relocatable phase
X'20': SVA eligible
X'10': phase has been placed in the SVA
X'08': phase has been found in a PCIL
X'04': phase has not been found
X'02': entry has been filled in by $MAINDIR for SDL
entry has been filled in by FETCH for GENL
X'01': used by $MAINDIR only, indicates that message 3M90I 'PHASE IS NOT SVA ELIGIBLE' should be issued when SVA is built.

7 STOW type byte, in CIL always X'00'

The fields 8 and 9 are not present if both are 0 and the phase is not relocatable.

8 Load point at linkage edit time

9 Entry point at linkage edit time.

The fields 10 to 12 are only present for relocatable phases.

10 Number of RLD items.

11 Number of additional RLD blocks.

12 Partition starting address at linkage edit time.

13 Entry point of phase in SVA, only used for entries in the SDL.

The last entry in the directory is a twelve byte entry with a dummy phase name containing 8X'FF', a dummy phase address of 3 bytes containing zeros, and a dummy field 3 of 1 byte also containinq zeros.

Following the directory are the records of the member space in the core image library. Figure 21 illustrates the structure of tracks, records, and phases where each record has 1024 bytes and each phase starts on record boundary.

First Track

Record 1	Record 2	Record 3	...	Record N
Phase A	Phase A	Phase A	...	Phase A

Second Track

Record 1	Record 2	Record 3	...	Record N
Phase A |	Phase B	Phase B	...	Phase B |

nth Track

Record 1	Record 2	Record 3	...	Record N
Phase X |	empty	empty	...	empty

Last Track

Record 1	Record 2	Record 3	...	Record N
empty	empty	empty	...	empty

The last record of each phase can contain less than 1024 bytes.

Figure 21. Core Image Library Member Space on a CKD Device

The remainder of the last record of a phase is used for relocation information for each individual address constant in the phase. Thus, the last TXT record of a phase has the format:

Field name	TXT	AL	M	PPP	M	PPP	...
Field length	0-max.	0-3	1	3			
Field no.	1	2	3	4			

1 The rest of the code. Overflow to the next record occurs either after the AL field or after each MPPP item.

2 Used to align the MPPP on a fullword boundary.

3 Bit string indicating the object length of the address constant (bits 3 and 4) and whether the relocation factor is to be added or subtracted (bit 7).

4 Address of the address constant when the phase was link-edited.

RELOCATABLE LIBRARY

The directory of a relocatable library has one or more tracks. The number of 320 byte records per track varies with the track size of the device:

```
    for 2314/2319 and 3340: 17
    for 3330.               28
    for 3350:               38
```

Each record takes 20 entries of 16 bytes each. A directory entry describing one module (the output of a complete language translator run) has the following format:

```
     0          8          10     14            16
    ┌───────────────────────────────────────────────┐
    | mod. name | rec. no. | CCHR | change level    |
    |───────────+──────────+──────+─────────────────|
    |     1     |    2     |  3   |        4        |
    └───────────────────────────────────────────────┘
```

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 Records — Total number of text records required to contain this module.

3. Disk Address — Start disk address of the first text record of this module in the relocatable library. The cylinder address is stored here in reverse form (C_2C_1). The field is then expanded to a $C_1C_2H_1H_2R$ seek address.

4. Change Level — Module identification.

The first five entries of the directory constitute the library descriptor and have together the following contents:

```
beginning byte: 0   7  15  23  30  37  44  48  52  56   60   64   66   68   70   80
               ┌──────────────────────────────────────────────────────────────────┐
field number:  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
               └──────────────────────────────────────────────────────────────────┘
```

1. Start address of the directory.
2. Address of the next availabel entry in the directory.
3. End address of the directory including last entry.
4. Start address of the member space.
5. Address of next available record in the member space.
6. End address of the library.
7. Number of active entries in the directory.
8. Number of records allocated for the member space.
9. Number of active records in the member space.
10. Number of deleted records in the member space.
11. Number of records available for additions.
12. Automatic condense limit for the library.
13. Total number of cylinders for the member space.
14. Number of tracks for the directory.
15. Reserved.

Following the directory are the 322-byte records of the member space in the relocatable library. Figure 22 illustrates the structure of tracks, records, and modules of the relocatable library for the various disk devices.

```
                             First Track
┌──────────────────────────────────────────────────────────────────┐
│ Record │     │                                              │     │
│    1   │  2  │  3     .    .    .    .                      │  s  │
│──────────────────────────────────────────────────────────────────│
│    Module A        │   Module B    │   Module C    │   Module D   │
└──────────────────────────────────────────────────────────────────┘

                             Second Track
┌──────────────────────────────────────────────────────────────────┐
│ Record │                                                    │     │
│    1   │  2     .    .    .    .    .                       │  s  │
│──────────────────────────────────────────────────────────────────│
│    Module D        │                                              │
└──────────────────────────────────────────────────────────────────┘

                             Last Track
┌──────────────────────────────────────────────────────────────────┐
│ Record │     │                                              │     │
│    1   │  2  │  3     .    .    .    .                      │  s  │
│──────────────────────────────────────────────────────────────────│
│                                                                   │
└──────────────────────────────────────────────────────────────────┘
```

s = the number of records per track
s = 16 for 2314/19
    17 for 3340
    28 for 3330/3333
    37 for 3350

Figure 22. Relocatable Library Member Space on a CKD Device

As figure 22 shows, the members of the relocatable library, called modules, always consist of several records. They always start on record boundary but can span over track boundary. The different types of contents of those records which make up a module are illustrated in figure 23.

```
  ┌─────── 1st byte = Number of logical records (1 in Linkage
  │                   Editor control cards 1 or 2 in 12-2-9 cards)
  │   ┌─── 2nd byte = Record length (160 in all records)
  V   V
┌─────────────────────────────────────────┐ ──┐
│ 1 │ A0 │ INCLUDE │        Unused         │   │ Linkage Editor
│───┼────┼─────────│───────────────────────│   > Control Cards
│ 1 │ A0 │ PHASE   │        Unused         │   │
│───┼────┼───────────────────────────────────│ ──┘
│ 2 │ A0 │      ESD       │      ESD         │ ──┐
│───┼────┼────────────────┼──────────────────│   │
│ 2 │ A0 │      ESD       │      TXT         │   │
│───┼────┼────────────────┼──────────────────│   │
│ 2 │ A0 │      TXT       │      TXT         │   │ 12-2-9 Cards
│───┼────┼────────────────┼──────────────────│   │
│ 2 │ A0 │  REP   │ Unused │      RLD         │   │
│───┼────┼────────────────┼──────────────────│   │
│ 2 │ A0 │      RLD       │      RLD         │   │
│───┼────┼────────────────┼──────────────────│   │
│ 2 │ A0 │      RLD       │  END   │ Unused  │   │
└─────────────────────────────────────────┘ ──┘
└───V───┘ └────────V────────┘ └───────V───────┘
 2 Bytes     160 Bytes           160 Bytes
└────────────────────V───────────────────────┘
                  322 Bytes
```

Figure 23. Module in the Relocatable Library

The formats of ESD, TXT, and RLD records are shown in Figures 24-26.

EXTERNAL SYMBOL DICTIONARY

SYMBOL	TYPE	ID	ADDR	LENGTH	LD ID
I JBLNK 10	SD	01	001900	000928	
I JBLNK	LD		001900		01
I JBLOV	LD		002008		01
I JBINL 10	SD	02	002228	000650	
I JBINL	LD		002228		02
I JJCPD3	ER	03			
I JJCPD1	ER	04			
I JBESD10	SD	05	002878	000458	
ETC					

Example of 8 ESD Items from Assembler Output Listing.

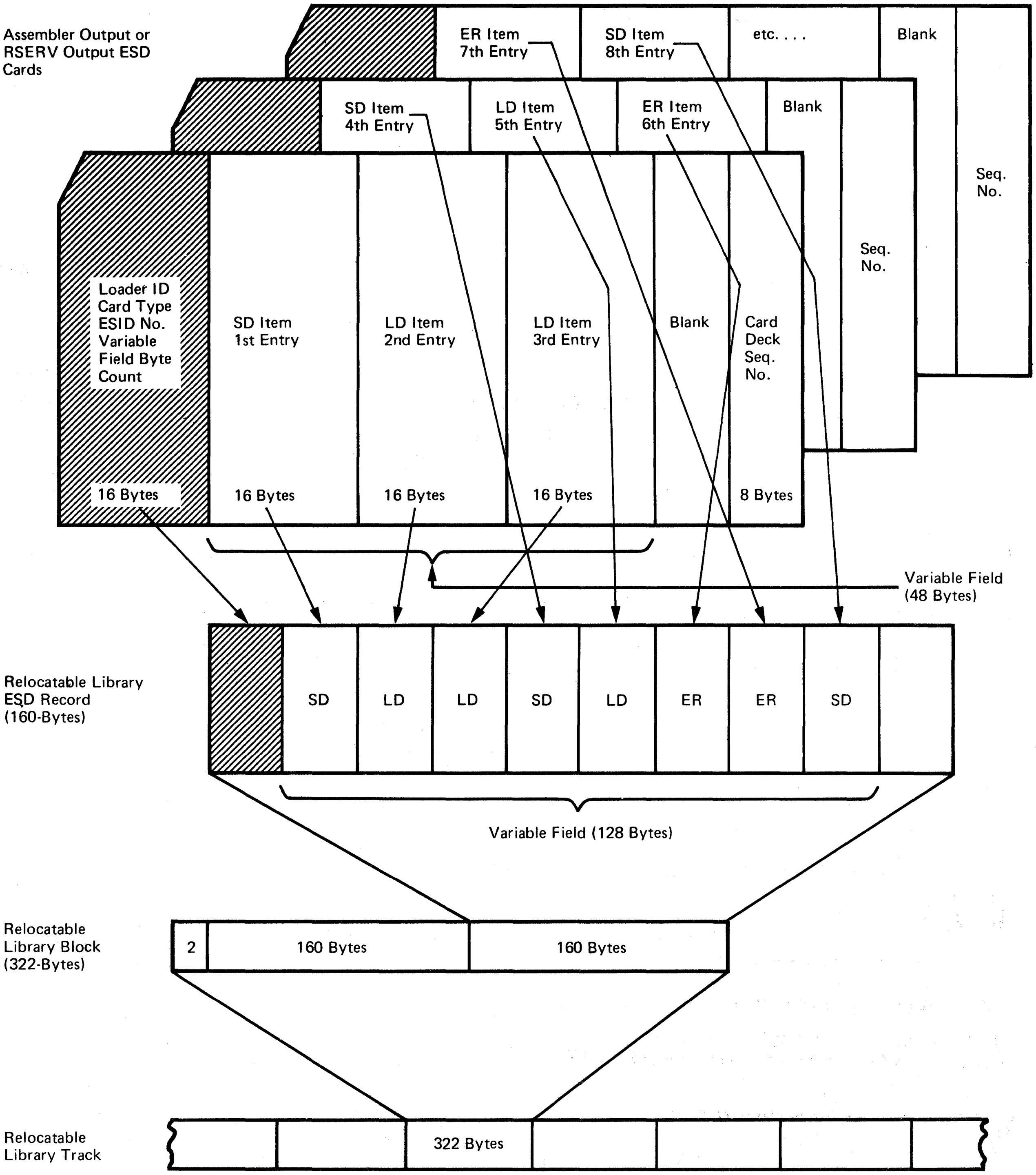
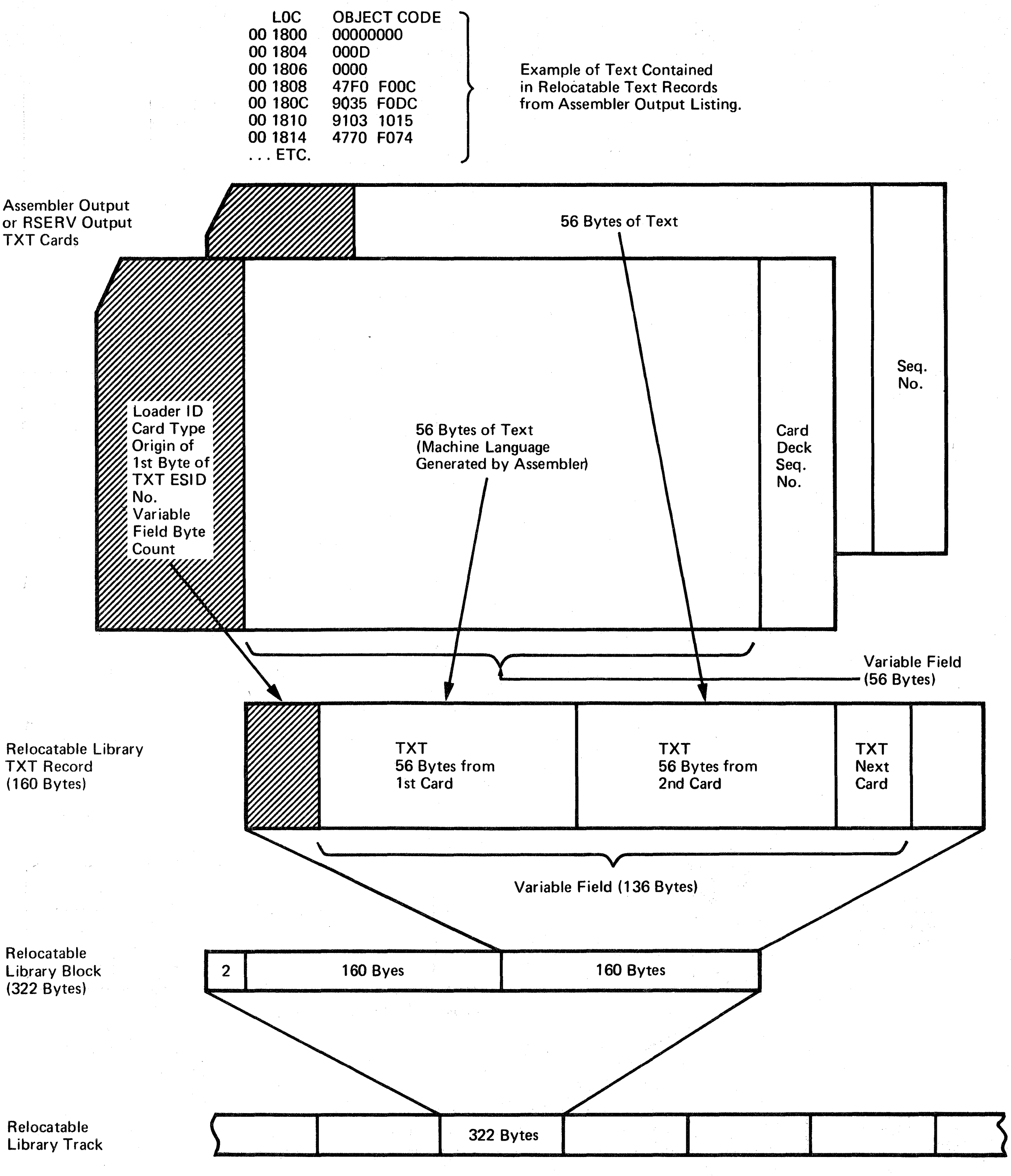


Figure 24. Format of ESD Records



```
  LOC       OBJECT CODE
00 1800    00000000
00 1804    000D
00 1806    0000
00 1808    47F0  F00C
00 180C    9035  F0DC
00 1810    9103  1015
00 1814    4770  F074
. . . ETC.
```

Example of Text Contained in Relocatable Text Records from Assembler Output Listing.
Assembler Output or RSERV Output TXT Cards
56 Bytes of Text
Seq. No.
Loader ID Card Type Origin of 1st Byte of TXT ESID No. Variable Field Byte Count
56 Bytes of Text (Machine Language Generated by Assembler)
Card Deck Seq. No.
Variable Field (56 Bytes)
Relocatable Library TXT Record (160 Bytes)
TXT 56 Bytes from 1st Card
TXT 56 Bytes from 2nd Card
TXT Next Card
Variable Field (136 Bytes)
Relocatable Library Block (322 Bytes)
2
160 Byes
160 Bytes
Relocatable Library Track
322 Bytes

Figure 25. 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

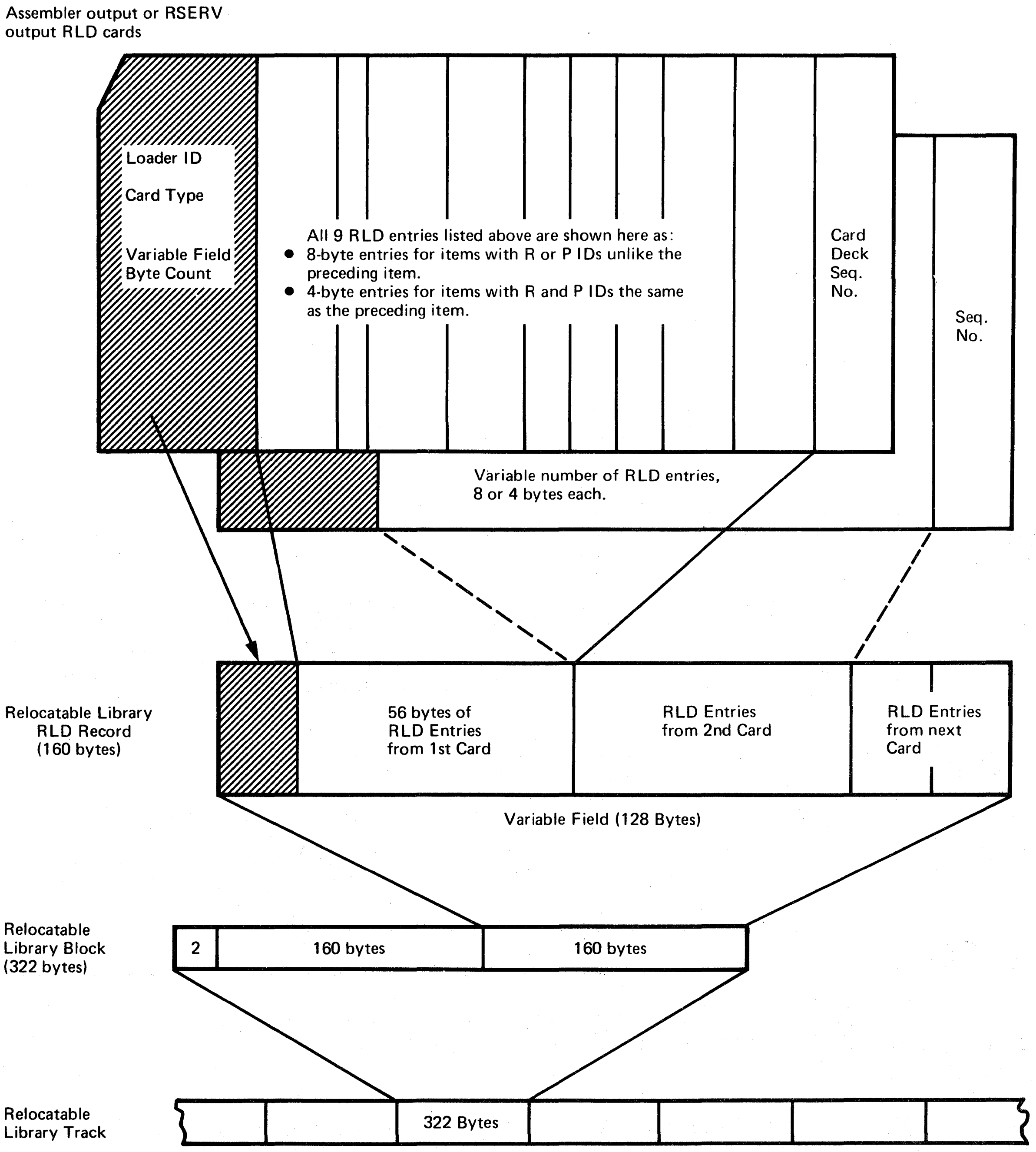

Assembler output or RSERV output RLD cards
Loader ID
Card Type
Variable Field Byte Count
All 9 RLD entries listed above are shown here as:
● 8-byte entries for items with R or P IDs unlike the preceding item.
● 4-byte entries for items with R and P IDs the same as the preceding item.
Card Deck Seq. No.
Seq. No.
Variable number of RLD entries, 8 or 4 bytes each.
Relocatable Library RLD Record (160 bytes)
56 bytes of RLD Entries from 1st Card
RLD Entries from 2nd Card
RLD Entries from next Card
Variable Field (128 Bytes)
Relocatable Library Block (322 bytes)
2
160 bytes
160 bytes
Relocatable Library Track
322 Bytes

Figure 26. Format of RLD Records

SOURCE STATEMENT LIBRARY

The directory of a source statement library has one or more tracks. The number of 160-byte records per track varies with the track size of the device:

for 2314/2319:	27
for 3340:	26
for 3330:	44
for 3350:	55

Each record takes 10 entries of 160 bytes each.

A directory entry describes one member of the source statement library, called a book.

A directory entry has the following format and contents:

```
beg. byte   0     1               9     12          14        16
           ┌──────────────────────────────────────────────────────┐
Field name │ Pre- │ Record name  │ CHR │ rec. no. │ change │
           │ fix  │              │     │          │ level  │
           ├──────┼──────────────┼─────┼──────────┼────────┤
Field no   │  1   │      2       │  3  │    4     │   5    │
           └──────────────────────────────────────────────────────┘
```

1. Sublibrary prefix   Any alphameric character, $, ┘, or @, "A", "C" and "E" are reserved for Assembler and COBOL. An * in this field indicates the logical end of the directory.

2. Record name   8 characters from the "CATALS" control statement.

3. Disk address   Start disk address of the first record of this book.
The two high order bits of the H-field in this 3-byte address are used as the two low order bits of C1 in the 5-byte seek address C1C2H1H2R to which the 3 byte address is expanded.

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

5. Change level   Book identification.

The library descriptor occupying the first five entries of the directory has the same format as in the relocatable library. See above.

Following the directory are the 160-byte records of the member space in the source statement library. Figure 27 illustrates the structure of tracks, records, and books in the source statement library for the various devices:

```
                               First Track
         ┌──────────────────────────────────────────────────────────────┐
records  | 1 | 2 | 3 | .. |         |         |         |         | s  |
         |──────────────────────────────────────────────────────────────|
contents |    Book A      | Book B  | Book C  | Book D | Book E         |
         └──────────────────────────────────────────────────────────────┘

                            Second or nth Track
         ┌──────────────────────────────────────────────────────────────┐
records  | 1 | 2 | 3 | .. |         |                             | s  |
         |──────────────────────────────────────────────────────────────|
contents |    Book E      | Book N  |           empty                  |
         └──────────────────────────────────────────────────────────────┘

                           Last Track of Last Cylinder
         ┌──────────────────────────────────────────────────────────────┐
records  | 1 | 2 | 3 | . . .                                       | s  |
         |──────────────────────────────────────────────────────────────|
contents |                         empty                                |
         └──────────────────────────────────────────────────────────────┘
```

s = the number of records per track
  = 26 for 3340
    27 for 2314/19
    44 for 3330/3333
    55 for 3350

Figure 27. Source Statement Library Member Space on a CKD Device

As Figure 27 shows the members of the source statement library, called books, consist of one or more records. They always start on record boundary.

A book is a sequence of source language statements (either macro definitions or source deck books) in compressed format. The compressed format for input to the source statement library is as follows:

Format: xynn ... nnxynn ... nnxynn ...

```
where: x  indicates the number of nonblank characters (maximum 15)
       y  indicates the number of blanks following
      nn  is any nonblank character
```

Example: Tne statement:

```
    LABEL1bbMVCbbbHERE,THEREb ... bCOMMENT1bCOMMENT2 ...
```

will be in compressed format:

```
    6 2 LABEL1 3 3 M V C A F H E R E , T H E R E 0 5 8 1 C O M M E N T 1 ...
```

PROCEDURE LIBRARY

The directory of the procedure library has the same structure as that of the source statement library with 160-byte records and ten entries per record. Also the library descriptor is identical to those of the relocatable and source statement libraries, occupying five entries or 80 bytes at the beginning of the directory.

However, the directory entry has a different format and contents as follows:

beginning byte	0	8	10	13	14	16
field no.	1	2	3	4	5	

1. Procedure name: One to eight alphameric characters, the first of which must be alphabetic. First character blank indicates that this entry is deleted, * indicates logical end of directory.
2. Number of records: Total number of records required to contain the procedure.
3. Disk address: Start address of the first record of this procedure. The two high bits of the H-field in this 3-byte address are used as the two low order bits of the C1C2H1H2R seek address to which the 3-byte address is expanded.
4. Program switches: If this byte contains X'80', the procedure may contain SYSIPT data.
5. Change level: Procedure identification.

Following the directory are the 80-byte records of the member space in the procedure library. The number of records per track has been determined as:

34 for 3340
40 for 2314/19
61 for 3330/3333
72 for 3350.

Each record contains one statement in card image format. The unused member space is unformatted.

LIBRARIES ON FIXED BLOCK DEVICES

FBA library sizes are defined by the user via the ALLOC/NEWVOL functions of the MAINT/CORGZ programs or via BACKUP/RESTORE. He has to specify the sizes of total library space and directory space.

Example: CL = 8000 (201)

This means 8000 blocks for the total library and 201 for the directory space.

FBA libraries have the following structure:

Library
- directory space allocated
  - library descriptor record (see Figure 28)
  - directory records used (see Figure 28)
  - directory records available
- member space allocated
  - member records
  - available blocks

All addresses given in a directory are relative to the beginning of the library.

Parts of blocks which are not occupied by records and also available records can contain arbitrary data.

DIRECTORY SPACE ON A FIXED BLOCK DEVICE

The Library Descriptor Record

The library descriptor record is the beginning of each library directory, has 86 bytes (CL 106 bytes), and is of the same format for all libraries as described in Figure 28. The rest of the 512 byte block is not used.

The library descriptor record contains the following variables:

N, N1 : NUMBER OF DIRECTORY RECORDS (DERIVED FROM USER SPECIFICATIONS)
M : NUMBER OF DIRECTORY RECORDS (1 TO N-1) USED, DELETED, OR AVAILABLE
P, P1 : NUMBER OF BLOCKS ALLOCATED TO MEMBER SPACE (DERIVED FROM USER SPECIFICATION)
Q : NUMBER OF MEMBER BLOCKS (0 TO P) USED, DELETED, OR AVAILABLE
Z : NUMBER OF USED BYTES IN DIRECTORY RECORD.

All addresses in this figure are relative to the beginning of the library.

Field Name	Displa-cement	Length <BYTE>	Description	Set to ... by ...	Changed to ... by ...
DESR	0	8	Reserved This field is used during MAINT CONDS or ALLOC to indicate inconsistent library status.	XL8'00' by RESTORE, NEWVOL, ALLOC	X'00'.C'DSTROYD' by CONDS at begin of CL CONDS XL8'00' by $MAINDIF at end of CL CONDS
DESLD	8	2	Library Descriptor Record Length <in bytes>	CL : X'0064' RL,SL,PL : X'0050' by RESTORE, NEWVOL, ALLOC	---
DEST	A	1	Library Type The first halfbyte gives the library type, the second halfbyte is reserved.	CL : X'80' RL : X'40' SL : X'20' PL : X'10' by RESTORE, NEWVOL, ALLOC	---
DESLDE	B	1	Length of Directory Entry <in bytes> If directory entries are of fixed length, the length is given in this filed. If directory entries can be of variable length, a length field is provided there.	CL : X'1E' RL,SL,PL : X'00' by RESTORE, NEWVOL, ALLOC	---
DESBAD	C	4	Address of Start Block of the First Directory Record <in blocks>	1 by RESTORE, NEWVOL, ALLOC	---
DESEAD	10	4	Address of the Last Block in Directory Space <in blocks> It points to the last block allocated to the directory, (i.e. for CL to the last block of the link record).	CL : 4N minim. 8 RL,:) 2N minim. 2 SL,) PL) by RESTORE, NEWVOL, ALLOC	CL : 4N; minim. 8 RL,:) 2N; minim. 2 SL,) PL) by ALLOC (MAINT)
DESFFD	14	6	First Available Entry in Directory		

Figure 28. Library Descriptor for Libraries on a Fixed Block Device (Part 1 of 4)

Field Name	Displacement	Length <BYTE>	Description	Set to ... by ...	Changed to ... by ...
DESFFDB	14	4	Block Number of Directory Record Containing First Available Entry <in blocks>	CL : 1+4M RL,SL, PL : 1+2M empty CL, RL, SL, PL:	CL : 1+4M RL, SL, PL : 1+2M
DESFFDD	18	2	Displacement to First Available Entry <in bytes> The first available entry is always the directory end indicator (* or XL8'FF'). A new entry overwrites this indicator.	CL : X'02' <Z <X'0800' Rl,) SL,) X'02' PL :) <Z <X'0400' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL : X'02' <Z <X'0800' RL,) SL,) X'02' PL:) <Z <X'0400' BY LNKEDT($MAINDIF), CONDS, CATALx, COPYx, UPDATE
	1A	6	Reserved for Condense (CONDS)		
DESBAL	20	4	Address of First Block <in blocks>	CL : 1+4N; minim.9 RL,) SL,)1+2N; minim.3 PL :) by RESTORE, NEWVOL, ALLOC	CL : 1+4N1; minim. 9 RL,) SL,)1+2N1; minim. 3 PL :) by ALLOC (MAINT)
DESEAL	24	4	Address of Last Block of Member Space <in blocks> This block points to the last block allocated to a library, i.e. it includes any block(s) which may not be useable to store members for member record > FBA block size.	CL : 4N+P RL, SL, PL : 2N+P by RESTORE, NEWVOL, ALLOC	CL : 4N1+P1 RL, SL, PL : 4N1+P1 by ALLOC(MAINT)
DESFFL	28	4	Address of First Available Block in Member Space <in blocks> The next member catalogued starts at this address. (For CL, this holds also for linked members.)	CL : 1+4N+2Q RL,) SL,) 1+2N+Q PL :) empty CL,RL,SL, PL : DESBAL by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL : 1+4N+2Q RL,SL,PL : 1+2N+Q by CATALx, COPYx, CONDS, UPDATE, LNKEDT($MAINDIF) CL : 1+4N1+2Q RL,SL,PL : 1+2N1+Q by ALLOC(MAINT)
DESFDL	2C	4	Address of First Deleted Member Record <in blocks>	CL,RL,SL,PL : F'0' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL : 1+4N+2Q RL,SL,PL : Not Used by DELETC($MAINDIF) CL : F'0' by CONDS($MAINDIF)

Figure 28. Library Descriptor for Libraries on a Fixed Block Device (Part 2 of 4)

Field Name	Displacement	Length <BYTE>	Description	Set to ... by ...	Changed to ... by ...
DESACT	30	4	Number of Active Members <in blocks> Number of members which are accessible in a library. For CL, linked members are however not reflected.	CL,RL,SL,PL : Z empty CL,RL,SL, PL : F'0' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL, RL, SL, PL : Z by CATALx, DELETx, COPYx, $MAINDIF
DESDEL	34	4	Number of Deleted Members Number of members which are marked 'deleted' and are thereforce not accessible. Linked members (CL) are not marked deleted after a link step.	CL,RL,SL,PL : F'0' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL, RL, SL, PL : Z by DELETx($MAINDIF), UPDATE, CATALx, LNKEDT($MAINDIF) CL, RL, SL, PL: F'0' by CONDS($MAINDIF)
DESNDB	38	4	Number of Directory Blocks Available <in blocks> Number of blocks available to contain directory records. For CL, the directory record for linked phases is not reflected in this number.	CL : 4M SL, RL, PL : 2M empty CL : 4N-4 empty RL, SL, PL : 2N-2	CL : 4M SL, RL, PL : 2M by CATALx, CONDS, COPYx, UPDATE, LNKEDT($MAINDIF) CL : 4M1 SL, RL, PL : 2M1 by ALLOC(MAINT)
DESNDU	3C	4	Number of Directory Blocks Used Number of blocks which contain directory records with at least one entry (end indicator is considered an entry). The Directory record for linked members (CL) is not reflected in this number.	empty CL : 4 empty RL,SL,PL : 2 by RESTORE, NEWVOL, ALLOC, DELETx ALL	by CATALx, LNKEDT ($MAINDIF), COPYx, UPDATE, CONDS CL : 4M1 Rl, SL, PL : 2M1 by ALLOC(MAINT)

Figure 28. Library Descriptor for Libraries on a Fixed Block Device (Part 3 of 4)

Field Name	Displa- cement	Length <BYTE>	Description	Set to ... by ...	Changed to ... by ...
DESNLB	40	4	Member Blocks Available This number excludes the last block of the CL if its member space has a odd number of blocks, since one record in it takes 2 blocks.	CL : 2Q RL, SL, PL : Q empty CL,RL, SL,PL: P by RESTORE, NEWVOL, ALLOC, DELETx ALL Note : An empty CL must contain a multiple of 2 as available blocks. P may be adjusted.	CL : 2Q RL, SL, PL : Q by CATALx, LNKEDT ($MAINDIF), COPYx, CONDS, ALLOC(MAINT), UPDATE
DESNLU	44	4	Member Blocks Used	CL : 2Q RL, SL, PL : Q empty CL : F'0' empty RL,SL, PL : F'0' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL : 2Q RL, SL, PL : Q by CATALx, LNKEDT ($MAINDIF) COPYx, UPDATE, CONDS
DESLSD	48	4	Member Blocks Deleted	CL : 2Q RL, SL, PL : Q empty CL : F'0' empty RL,SL, PL: F'0' by RESTORE, NEWVOL, ALLOC, DELETx ALL	CL : 2Q RL, SL, PL : Q by CATALx, DELETx, COPYx, LNKEDT ($MAINDIF), UPDATE CL,RL,SL,PL : F'0' by CONDS
DESCL	4C	4	Available Member Blocks for CONDS. Note : This number when reached in an operation will result in a message.	CL,RL,SL,PL : F'0' by RESTORE, NEWVOL, ALLOC,	CL : 2Q RL, SL, P : Q by CONDL
The following items are valid only for CL					
DESDTE	50	8	Date and Time of Last Update	CL : F'1' by RESTORE	Date and Time LNKEDT, COPYC, CONDS CL via $MAINDIF; NEWVOL, ALLOC
	58	12	Not Used		

Figure 28. Library Descriptor for Libraries on a Fixed Block Device (Part 4 of 4)

Directory Records on a Fixed Block Device

The directory records on fixed block libraries have a length of 1024 bytes for RL, SL, and PL and of 2048 bytes for CL. They have the following format:

```
beginning byte   0      2                                            1 or 2K
                 ┌──────────────────────────────────────────────────┐
field name       | U  | entry 1 || entry 2 | ... entry n |   F   |
                 └──────────────────────────────────────────────────┘
```

U = number of used bytes in the record

```
entry = CL     - 30 bytes
        RL     - 18 bytes
        SL+PL  - 19 bytes
```

```
maximum number of entries: CL     - 68
                           RL     - 56
                           SL+PL  - 53
```

F = the free space in the record

The last record in the CL directory is reserved for the link directory. Following are the list of contents of directory entries for each library type.

Core Image Directory Entry:

```
CDYBEG   EQU   *         BEGIN OF DIRECTORY ENTRY DEFINITION
CDYNME   DS    XL8       NAME OF PHASE.
CDYADR   DS    XL4       ADDRESS OF PHASE IN LIBRARY <BL>.
CDYNTX   DS    XL2       NUMBER OF TEXT RECORDS <1024BY>.
CDYNTB   DS    XL2       NUMBER OF TEXT BYTES IN LAST TEXT
*                        RECORD <BY>.
CDYSW    DS    XL1       SWITCH BYTE:
*                  ON    |     X'80':  SELFRELOCATING PHASE,
*                  DISK  |     X'40':  RELOCATING PHASE,
*                  ONLY  |     X'20':  SVA ELIGIBLE PHASE,
*
*                        :     X'10':  PHASE HAS BEEN PLACED
*                  FOR   :             IN THE SVA.
*                  IN    :     X'08':  PHASE HAS BEEN FOUND IN
*                  CORE  :             PRIVATE CORE IMAGE LIB.
*                  DIR.  :     X'04':  PHASE HAS NOT BEEN FOUND.
*                        :     X'02':  ENTRY HAS BEEN FILLED IN
*                        :             BY 'IJBLBIOF'.
*
*                              X'01':  FOR 'IJBLBIOF' ONLY:
*                                      TURNED ON BY 'CATALOG'
*                                      TO INDICATE THAT MESSAGE
*                                      3M90I SHOULD BE ISSUED
*                                      WHEN SVA IS BUILT.
CDYTBY   DS    XL1       TYPE BYTE FOR STOW TABLE.
CDYLPL   DS    XL3       LOAD POINT AT LINKAGE EDIT TIME.
CDYEPL   DS    XL3       ENTRY POINT AT LINKAGE EDIT TIME.
CDYNRD   DS    XL2       NUMBER OF RLD ITEMS IN LAST TEXT REC.
CDYNRB   DS    XL1       NUMBER OF ADD.RLD RECORDS <1024BY>.
CDYPST   DS    XL3       PARTITION START ADDRESS AT LINKAGE
*                        EDIT TIME.
CDYEND   EQU   *         END OF DIRECTORY ENTRY DEFINITION
```

Relocatable Directory Entry:

```
RDYBEG    EQU   *          BEGIN OF DIRECTORY ENTRY DEFINITION.
RDYNME    DS    XL8        NAME OF MODULE.
RDYADR    DS    XL4        ADDRESS OF MODULE IN LIBRARY <BL>.
RDYLE     DS    XL1        LENGTH OF ENTRY <BY>.
RDLMD     DS    XL3        LENGTH OF MODULE <322-BYTE RECORDS>.
RDYCHD    DS    0XL2       CHANGE IDENTIFICATION:
RDYCHDV   DS    XL1          VERSION,
RDYCHDM   DS    XL1          MODIFICATION.
RDYEND    EQU   *          END OF DIRECTORY ENTRY DEFINITION.
```

Source Statement Directory Entry:

```
SDYBEG    EQU   *          BEGIN OF DIRECTORY ENTRY DEFINITION.
SDYNME    DS    0XL9       NAME OF BOOK:
SDYSUBLB  DS    XL1          SUBLIBRARY QUALIFIER,
SDYBKNME  DS    XL8          BOOK NAME.
SDYADR    DS    XL4        ADDRESS OF BOOK IN LIBRARY.
SDYLE     DS    XL1        LENGTH OF ENTRY.
SDLMD     DS    XL3        LENGTH OF BOOK =160-BYTE RECORDS>.
SDYCHD    DS    0XL2       CHANGE IDENTIFICATION:
SDYCHDV   DS    XL1          VERSION,
SDYCHDM   DS    XL1          MODIFICATION.
SDYEND    EQU   *          END OF DIRECTORY ENTRY DEFINITION.
SDYNDER   EQU   53         NO OF DIRECTORY ENTRIES IN A DIRECTORY
```

Procedure Directory Entry:

```
PDYBEG    EQU   *          BEGIN OF DIRECTORY ENTRY DEFINITION.
PDYNME    DS    XL8        PROCEDURE NAME.
PDYADR    DS    XL4        ADDRESS OF PROCEDURE IN LIBRARY <BL>.
PDYLE     DS    XL1        LENGTH OF ENTRY <BY>.
PDLMD     DS    XL3        LENGTH OF PROCEDURE <80-BYTE RECORDS>.
PDYCHD    DS    0XL2       CHANGE IDENTIFICATION:
PDYCHDV   DS    XL1          VERSION,
PDYCHDM   DS    XL1          MODIFICATION.
PDYSW     DS    XL1        SWITCH BYTE:
*                            X'80':  PROCEDURE WITH SYSIPT DATA,
*                            X'40':  UNUSED,
*                            X'20':  UNUSED,
*                            X'10':  UNUSED,
*                            X'08':  UNUSED,
*                            X'04':  UNUSED,
*                            X'02':  UNUSED,
*                            X'01':  UNUSED.
PDYEND    EQU   *          END OF DIRECTORY ENTRY DEFINITION.
```

MEMBER SPACE ON A FIXED BLOCK DEVICE

The records of a member are stored contiguously in fixed blocks, except in the procedure library where each block contains up to six job control statements, unused space, and control information. Each member starts on block boundary. Therefore the members can be addressed by block number. Their formats are like in CKD:

CL - TXT and RLD records - 1024 bytes
RL - ESD, TST, RLD records - 322 bytes
SL - source statements compressed - 160 bytes
PL - job control statements - 80 bytes

PRIVATE LIBRARIES

Three types of private libraries are supported:

core image,
relocatable, and
source statement libraries.

The private libraries may be on the same disk pack as the SYSRES file. Otherwise they have to be on the same type of disk unit as the SYSRES pack. An exception are the core image libraries which may be on any disk device (but not for CORGZ COPY).

Several private libraries may be on the same disk pack, but in such cases they must have different file identifications. For example, two private source statement libraries have the same file name IJSYSSL. Therefore, their file identification must be different; for example, ONEPRSL and TWOPRSL.

Each private library has an organization identical to the corresponding system library.

Each private library contains only one directory. The directory of each library starts at the lower limit of the file and consists of the number of tracks specified in the NEWVOL control statement for the CORGZ program. The member space of each library starts on the track following the last track used by its directory and uses the rest of the cylinder(s) specified in the NEWVOL control statement. The private libraries thus have the same format as the system libraries on SYSRES.

The contents and organization of the private libraries are the same as for the system libraries.

Private libraries are created by the NEWVOL function of the CORGZ program. All librarian functions may be performed on private libraries as well.

References can be made to a private library only if SYSCLB, SYSRLB, or SYSSLB are assigned. When any of these assignements are made the corresponding system library cannot be changed.

GENERAL OVERVIEW OF LIBRARY RECORD SIZES

Descriptor:

CKD:	CL :	58 bytes = 1 entry	on 256 byte record
	RL :	80 bytes = 5 entries	on 320 byte record
	SL :	80 bytes = 5 entries	on 160 byte record
	PL :	like SL	
FBA :	CL :	100 bytes)	
	all)	)	on 512 byte block
	others):	80 bytes)	

Directory Entries:

CKD :	CL :	18-30	bytes	on 256 byte record
	RL :	16	bytes	on 320 byte record
	SL+PL :	16	bytes	on 160 byte record
FBA :	CL :	30	bytes	on 2048 byte record
	RL :	18	bytes)	on 1024 byte record
	SL+PL :	19	bytes)	

Member Records:

FBA and CKD :	CL :	1024 bytes
	RL :	322 bytes
	SL :	160 bytes
	PL :	80 bytes

Figure 29. Library Record Size Overview

LABEL LIST FOR CHARTS 01-93

CORYSERV:	
CHECKTYP	1
COPYSERV	1
CORGZ:	
BEGIN0	2
END	3
EXIT	3
FROMSIX	3
INITCK	2
ITSALLOC	3
ITSCOPY	3
ITSMERGE	3
ITSNEWVL	3
RTN2	3
STARTR	2
STMTSCAN	3
CORGZ3:	
ALLOCMB	4
CALSPACE	4
CHKTOD	5,6
CONTALL	4
CONTALLN	6
CONTRDCD	5
COPYAL1	6
COPYDIR	4
COPYEND	5
COPYI	4
COPYMEM	4
COPYMEM1	4
COPYPL	4
COPYRL	4
COPYSL	4
COPYXALL	6
COPYXNEW	6
DBALLOC	4
DONCHKT	5
INITCPY	4
JOINT	4
JOINT1	4
NOTCALL	4
NXTCOMP	4
STNBEL	4
TSTALL	6
TSTCONT	5
TSTMORE	5
TSTNEW	6
WRTLAST	5
CORGZ6:	
ALLOCMB	8
BEGIN6	8
CHKALLOC	8
CONTALL	9
CONTCNEW	9
COPYCALL	9
COPYCNEW	9
COPYMEM1	8
COPYWHAT	8
COPY$$	8
DBALLOC	8
GETCARD	9
JOINT	8
JOINT1	8
NXTCOMP	8
NXTCOMPF	9
NXTCOMPN	9
PROCSTT	7
CORGZ7:	
ALLOC	10
ALLOCA	10
ALLOC1	10
BEGIN7	10
CKOPER	10
CONT	11
CONTALLC	11
IPRVRL	10
IPCIL	10
NEWVOL	10
ONEUP	10
RESPL	10
RESRL	10
RESSL	10
SYSLABEL	11
UPDATE	10
UPONE	10
UPPTR	10
CORGZ3F:	
BEGIN3F	13
CHKDUPNM	13
CHKTOD	14
COPY	13
COPYDIR	13
COPYPL	13
COPYRL	13
COPYSL	13
COPYXALL	14
COPYXAL1	14
COPYXNEW	14
COPY1	13
ENDCOPY	14
ENDCPYM	14
ENDCPYM1	14
ENDNMCMP	13
FNDCPY	13
IPLMERGE	13
JOINT	13
JOINT1	13
NEXTCARD	14
NOTCALL	13
NXTALL	14
NXTNEW	14
CORGZ6F:	
BEGIN6F	15
CHKDUPNM	15
CHKNEW	15
COMPARE	15
CONTFDIR	15
COPYCALL	15
COPYWHAF	15
DOMEMBER	17
DOPHASE	16
DOPHASE2	16
ENDCDIR	16
ENDFMCMP	15
GETCARD	16
INSERT	16
JOINT	15
JOINT1	15
MAINDIF	17
NXTINTRY	16
NTTOCMP	16
PRINTNAM	17
PROCNEW	16
STOWNXT	17
STOWPROC	17
CORGZ7F:	
ALLOC	18
ALLOC1	18
BEGIN7F	18
CKOPER	18
CONTALLC	19
COPYIPL	19
IPCIL	18
LPRVRL	18
ONEUP	18
RESPL	18
RESRL	18
RESSL	18
THEN	19
UPDATE	18
UPONE	18
UPPTR	18
MAINT:	
AALLOC	21
ACATAL	21
ACONDL	22
ACONDS	22
ADELREN	21
AUPDATE	22
BEGINN	20
BEGINN1	20

CSCONT 20
CSSTART 20
DKADUP 23
DSCONT 23
DSSTART 23
DUNEW 23
DUOLD 23
DUSTART 23
ENDJOB 20
EOF 20
FETCHP2 21
FETCHR2 21
FETCHS2 21
ILLGOPA1 22
INSTART 20
IOSTART 23
LDMNTPH1 23
MAINTRTN 23
MNTLP 22
MNTRETN 21
NWREAD 20
NXTCONDS 22
WTDATEND 23
WTFMENT 23

MAINTCL:
CLREQ 24
MAINLINE 24
PLREQ 24
RLRREQ 24
SLREQ 24

MAINTCN:
CICONRTN 25
COMPDIA 25
EOJRTN 25
IOEXEC 25
MAINCIL 25
PLCONRTN 25
RLCONRTN 25
SSCONRTN 25
TYPCONCK 25

MAINTCNF:
CICONRTN 26
CILINIT 26
DESREAD 28
EOJRTN 29
INITST 26
LIBCONDS 29
LOADINIT 29
MAINCIL 29
NEXTLIB 29
PLCONRTN 27
PREPARE 28
RLCONRTN 27
RLSLPL 28
SSCONRTN 27
SYSCIL 26
TYPCONCK 26

MAINTR2:
CATALR 30
COMCAT 30
LEAVE 30
RCESD 30
RCRLD 30
RCTXT 30

MAINTR2F:
CATALR 31
COMCAT 31
LEAVE 31
RCESD 31
RCRLD 31
RCTXT 31

MAINTS2:
LCDPR1 32
READCD 32

MAINTS2F:
LCDPR1 33
READSD 33

MAINTP2:
CATALP 34

MAINTP2F:
CATALP 35

MAINTDR:
CLIB 36
DRSTART 38
MAINRTN 37
PLL 36
RENAMRTN 37
RLL 36
SLL 36

MAINTDRF:
CLIB 38
DRSTARTF 38
MAINRTN 39
PLL 38
RENAMRTN 39
RLL 38
SLL 38

MAINTA:
DIRUP 40
GOON1 40
MAINTA 40
MOVE 40
TKFMT 40
WRITE2 40

MAINTAF:
ALLERR 41
BLDREALT 41
BLDTABL 41
DOPASS1 42
DOPASS2 42
DSPLN 42
ENVERR 41
INITLISE 41
INTL1 41
MAINTAF 41
PROCRDO 41
REALLOC 42
SYNERR 41
TYPEL 41
UPDTE1 42
UPSTAT 42

MAINTUP/F:
CHKOP2 44
CHKOP3 44
CTLCRD 44
ENDPROC 46
GTBK 46
PROCADD 45
PROCDEL 45
PROCEND 45
PROCREP 45
RDDISK 44
READRTN 44
RETURN2 46
RTMAINT 46
SETPTR 44
STARTUP 44

$LIBSTAT:
CKDRUN 47
FBARUN 47
IJBLBI1 47

$MAINDIR:
BLOWUP 52
CATALOG 54
CHKEQL 53
CLOSE 56
DELET 55
DOIO 57
EXTR 55
FINI 49
FINISH 54
GENMSG 54
GETENTRY 56
GETINP 56
INITIALZ 49
INITMAIN 50
INITSLDL 50
INITSYSL 50
KEEPDIST 57
LCCALL 51
LOAD 54
LOOKUP 55
MAINT 52
MERGESDL 51
MSG91I 57
OPTLINK 51
PROCESS 53
PUTENTRY 56
PUTOUTP 56
RASCONV1 57
RASCONV2 57
RASEQL 53
REALLOC 51
RENM 52
RUN 53
SCANSDL 54
SDLSVA 50
SORT 56
SORTSDL 51
SPACETST 53
SVAMSG 54
UPDAT 54
UDDSLD 57

$MAINDIF:
CATALOG 64
CHKEAL 64
DELET 66

DELETALL 61
EXTR 66
FINISH 64
GENMSG 64
GETENTRY 67
INITIALZ 59
INITFBA 59
INITLIBD 59
INITMAIN 59
INITSLDL 59
INITSYSL 59
INTERROR 69
LCCALL 61
LOAD 65
LOOKUP 67
MAINT 62
MERGESDL 60
MSG 66
OPTLINK 61
PROCESS 64
PUTENTRY 68
RASCONV 69
RDDIRREC 68
RDLIBDES 68
REALLOC 61
RENM 63
RUN 63
SCANSDL 65
SCLVLDR 65
DSLSVA 60
SORT 63
SORTSDL 60
SPACETST 63
SVAMSG 65
UPDAT 64
WRDIRREC 69
WRLIBDES 68

DSERV:
D1CPLB 71
D1CPRLB 71
D1CPSLB 71
D1CRES 71
D1CSRLB 71
D1CSSLB 71
READDIR 71

DSERVF:
D1FPLB 71
D1FPRLB 71
D1FPSLB 71
D1FRES 71
D1FSRLB 71
D1FSSLB 71
READDIRF 71

DSERV1:
GETPASS 73
LOADF2 73
LOADSTAT 73
READCARD 73

CSERV:
CMPRBL 77
ENDRTN 77
ERRMSGS 77
PCHRTN 77
RDCARD 77
START 77
TRANSLT 77
TRANTST 77
WRITE 77

CSERVC, CSERVF:
AFTERBUF 78
EXECUTE(F) 78
IADDR(F) 78
NEXTLN 78
SEQREAD 78
SERVICE(F) 78

RSERV:
ENDRTN 79
ERILOP 79
EXTRCT 79
IGNORE 79
ILOPRD 79
NOLIB 79
RDCD 79
SOME 79
START 79

RSERVC, RSERVF:
CNVORG 80
ESDPCH 80
ESDPRT 80
PRTHDR 80
RDLIBC(F) 81
RDRD1C(F) 81
RDSEQU 81
REPPRT 80
RLABAS(F) 81
RLDPCH 80
RLDPRT 80
SYMPRT 80
TXTPCH 80
TXTPRT 80
UPDTC 81

SSERV:
ALOERR 83
BKNDOU 83
BLNENT 82
CALLCS 82
CPRPCH 83
CPRSC1 83
CSERR0 82
CSERR1 82
CSERR2 82
CSERR3 83
EXPOUT 83
EXPTIN 82
FBAINIT 82
FNDBK 83
GETALL 83
NBLENT 82
NFERR 83
NOFND 83
PRTCCD 82
SLASHO 82
START 82

PSERV:
ALOERR 85
BKNDOU 85
CALLCS 84
CSERR1 84
CSERR2 84
CSERR3 85
EXPCD1 85
FNDBK 85
GETBK1 84
NFERR 85
OPSCAN 84
PRTMSG 84
RDRDR 84
READMEMB 85
START 84

$$BSYSWR:
DOCKDDEV 86
DOFBADEV 86
SYSWR 86
TESTTYPE 86

$$BOPNLB:
PRIVLIB 87
PRIVL01 87
PRIVLIB1 87
SYSTLIB 87
SYSTLIB1 87

DTFSL:
DIRSCAN 88
DTFSLCHK 88
DTFSLGET 90
DTFSLFND 90
DTFSLFN1 90
DTFSLPNT 90
DTFSNTE 90
FNDBK 88
FNDL 88
FNDLPR 88
FNDLSL 88
GFTL 89
GOPEN 88
NOTEL 89
PTL1 89
RDBLK 89

$IJBLBSL:
BOOKFND 91
DECOMPR 92
DIRSCAN 91
DOFINDLB 91
DOSFQNTL 91
ENDFIELD 92
FINDRQST 91
GETRQST 92
IJBLBSL0 91
INITPRV0 93
INITPRV3 93
INITSYSL 93
INITSYS01 93
NOTERQST 92
OPENLIBS 93
PNTRQST 93
READRQST 93
SELSQNTL 91
SINGLBOK 91

LABEL LIST FOR CHARTS AA-RG

Label	Phase	Location
ADDMOD	MAINTUP	NPB2
ADDONE	MAINTUP	NXB2
ADDREG	MAINTUP	NEF3
ADDRUPD	MAINTCN	FMC1
ADDRUPDD	MAINTCN	FMB1
ADDRUPDL	MAINTCN	FMB2
ADDSTMT	MAINTUP	NGA4
ADREX	MAINTCN	FLB4
ADRINS	MAINTCN	FLB5
ADRUPDLP	MAINTCN	FGF2
ALL	MAINTA	MNC5
ALLERR	MAINTA	MNB5
ANYBKNME	MAINTUP	NAF3
ANYBKNME	MAINTUPF	PCD3
ARND1	MAINTCN	FLE1
ASTCHK	MAINTCN	FFC5
ASTCHK1	MAINTCN	FFD5
BINCVRT	MAINTUP	NED4
BKNTFND	MAINTUP	NFJ4
BKNTFND	MAINTUP	NSC2
BLDTB	MAINTA	MEA3
BLFLG	MAINTUP	NQJ2
BLKDATA	MAINTUPF	PGC2
BLKEX	MAINTCN	FLB3
BLNFLG	MAINTUP	NQF5
BLNKCHR	MAINTUP	NTF1
BLNKCHR	MAINTUPF	PFF1
BLNKOUT	MAINTUP	NNE2
BLNKSCN	MAINTUP	NFD4
BOOK	MAINTUP	NZB5
BRAN2	MAINTUP	NRA1
BRBACK	MAINTUP	NSG2
BRCOMP	MAINTUP	NGA5
BRCOMP2	MAINTUP	NKA4
BRKCHN	MAINTCN	FHC4
BRKCHNRB	MAINTCN	FHA5
BRLINK	MAINTUP	NLA3
BRNCT	MAINTUP	NDA4
BRTOGET	MAINTUP	NGD2
BUILDSTW	MAINTA	MMA3
BUMPCCW	MAINTCN	FKD4
BUMPENT	MAINTCN	FFG4
CANCEL	MAINTCL	EYH2
CCWBUMP	MAINTCN	FHJ2
CCWCHK	MAINTCN	FGB3
CDPRT	MAINTUP	NKH2
CDSEQER	MAINTUP	NGE5
CHARAC	MAINTUP	NWD5
CHARST	MAINTUP	NEF5
CHBKT	MAINTUP	NZA3
CHKADD1	MAINTUP	NYA2
CHKADD2	MAINTUP	NYD3
CHKCHAR	MAINTUP	NUE3
CHKCIL	DSERV1	RAD3
CHKDEC	MAINTUP	NCA1
CHKDIR	MAINTUP	NAB3
CHKINCR	MAINTUP	NRH4
CHKLTH	MAINTUP	NDC2

Label	Phase	Location
CHKNUM	MAINTUP	NEF1
CHKOP2	MAINTUP	NDA2
CHKOP3	MAINTUP	NEA1
CHKPER	MAINTUP	NDB2
CHKPL	DSERV1	RAJ2
CHKRL	DSERV1	RAF2
CHKRLB	DSERV1	RAE2
CHKSDL	DSERV1	RAK2
CHKSEQ	MAINTUP	NRB4
CHKSL	DSERV1	RAH2
CHKSLB	DSERV1	RAG2
CHKSSL	MAINTUP	NAA3
CHKTEMP	MAINTUP	NNC3
CHKV	MAINTUPF	PDH1
CHKVMD	MAINTUP	NNA3
CHLVERR	MAINTUP	NMJ1
CICONRTN	MAINTCN	FDG1
CIKSMOVE	MAINTCN	FEC2
CKMULTI	MAINTCN	FDB3
CLAREA	MAINTUP	NEJ4
CLDWORD	MAINTUP	NEJ5
CLREQ	MAINTCL	EXG2
CLRMOD	MAINTUP	NNK2
CLRSWCH	MAINTUP	NWH5
CLSWT	MAINTA	MAG3
CMMACHK	MAINTUP	NBA4
CMPBLNK	MAINTUP	NQD5
CMPCHLV	MAINTUP	NEC1
CMPREC	MAINTUP	NKB2
CNDSETUP	MAINTCN	FGG2
CNTALLO	MAINTA	MNH1
CNVRTK	MAINTA	MBE1
COMFND	MAINTUP	NWH3
COMLOC	MAINTUP	NMF1
COMP	MAINTCL	EXE1
COMPARE	MAINTUPF	PFE5
COMPA1	MAINTA	MPE1
COMPA2	MAINTA	MPF2
COMPBKT	MAINTUP	NZD2
COMPLN	MAINTUP	NAJ3
COMPLN	MAINTUPF	PCH3
COMPRES	MAINTUP	NQB2
COMPSEQ	MAINTUP	NKA1
CONDENS	MAINTCN	FFE4
CONVRT	MAINTA	MBB4
COPENTRY	COPYSERV	AAB2
COPLAB	MAINTA	MPC5
COPLABH	MAINTA	MPH3
COPYLB	MAINTA	MQB1
COPYLC	MAINTA	MQE2
COPY020	COPYSERV	AAC2
CPBLNK	MAINTUP	NQG2
CPBOUND	MAINTUP	NWC3
CRDMVR	MAINTUP	NJG4
CSWTCH	MAINTUP	NCG2
CTERCL	MAINTUP	NAE3
CTERCL	MAINTUPF	PCC3
CTLCRD	MAINTUP	NFA5
CVTBIN	MAINTUP	NCH2
CYL	MAINTA	MAA5

Label	Phase	Location
DECCHK	MAINTUP	NYB2
DECFND	MAINTUP	NWC5
DECREG	MAINTUP	NDC4
DECR1	MAINTUP	NBA1
DECR2	MAINTUP	NDD2
DEFAULT	MAINTUP	NRC5
DELSTMT	MAINTUP	NLK2
DESCRUPD	MAINTUPF	PEE1
DETOPD	MAINTUP	NBG3
DIR	MAINTA	MNC3
DIRENTBP	MAINTCN	FMB4
DIRERR	MAINTA	MNB3
DIRINBMP	MAINTCN	FGD1
DIROUTBP	MAINTCN	FGD5
DIRRDRTN	MAINTCN	FKB1
DIRSCAN	MAINTUPF	PFB5
DIRUP	MAINTA	MJA4
DIRUP1	MAINTA	MJB3
DIRUP2	MAINTA	MJE3
DIRUP3	MAINTA	MJF3
DIRUP4	MAINTA	MJJ3
DIRUP5	MAINTA	MJD5
DIRUP7	MAINTA	MJG5
DIRUP8	MAINTA	MJE1
DIRUP9	MAINTA	MJK3
DIRU10	MAINTA	MJH1
DIRWTRTN	MAINTCN	FKB2
DISASTER	MAINTCN	FLH2
DMYENTRY	MAINTUPF	PEA1
DNTCK	MAINTA	MGA5
DOIO	MAINTCN	FKF2
DOIO	MAINTUPF	PEB2
DOITNW	MAINTA	MHB1
DRTSCN1	MAINTUP	NTB3
DRTSCN1	MAINTUPF	PFB3
DRTSCN2	MAINTUP	NTB5
DRTSCN2	MAINTUPF	PFB4
DSKERR	MAINTA	MNB1
DSPLN	MAINTA	MFB3
DSPL1	MAINTA	MFD3
DSPL2	MAINTA	MFF2
DSPL3	MAINTA	MFJ3
DSPL4	MAINTA	MFF4
D1FPSLB	DSERV1	RAG4
D1FRLB	DSERV1	RAE4
D1FSRLB	DSERV1	RAF3
D1PLB	DSERV1	RAJ3
D1SSLB	DSERV1	RAH3
D2FVM	DSERV2F	RCJ2
D3PLB	DSERV3F	REF2
D3PRLB	DSERV3F	REC2
D3PSLB	DSERV3F	REE2
D3RDDES	DSERV3F	REH2
D3RDDIR	DSERV3	RDB2
D3RLB	DSERV3F	REB2
D3SLB	DSERV3F	RED2
D4PLB	DSERV3F	REF5
D4PRLB	DSERV3F	REC5
D4PSLB	DSERV3F	REE5
D4RLB	DSERV3F	REB5
D4SLB	DSERV3F	RED5
ELIGIBF	DSERV2F	RCE3
ELIGIBL	DSERV2	RBE3
ENDCHK	DSERV2	RBH4
ENDCHKF	DSERV2F	RCH4

Label	Phase	Location
ENDJOB	MAINTA	MMJ4
ENDPROC	MAINTUP	NMA3
ENDPTR	MAINTUP	NWH1
ENDSEQ	MAINTUP	NSF2
ENTLINE	DSERV6	RGE2
ENTLINEC	DSERV6	RGF2
ENTLINEF	DSERV6	RGF3
ENVERRO	MAINTA	MAJ5
EOJRTN	MAINTCN	FJB1
ERCONT	MAINTCL	EYH3
ERCRD	MAINTUP	NDE1
ERINOP	MAINTCL	EYG1
ERRDSP	MAINTCL	FAB2
ERRSWCH	MAINTUP	NZG2
ERRTN	MAINTUP	NVB5
ERSRCE	MAINTUP	NWJ5
EXDIRIO	MAINTCN	FKD2
EXEC	MAINTUP	NDA3
EXECIO	MAINTCN	FKE2
EXITOK	MAINTCN	FKH3
FBACIL	MAINTCL	EXH4
FBAPROC	MAINTCL	EYH5
FBAREL	MAINTCL	EYA4
FBASSL	MAINTCL	EXD5
FBAWRITE	MAINTCL	FAG1
FDCHAR	MAINTUP	NVE1
FETCH1A	DSERV4	RFF2
FIND	MAINTUP	NFF3
FLSSTM	MAINTUP	NVB1
FNDBLNK	MAINTUP	NJK4
FNDDEC	MAINTUP	NEG1
FNDEND	MAINTUP	NAA5
FNDEND1	MAINTUP	NAB5
FNDIGIT	MAINTUP	NWB1
FNDM	MAINTUP	NFA1
FNDPER	MAINTUP	NAH3
FNDPER	MAINTUPF	PCF3
FNDSEQ	MAINTUP	NJH4
FRSTSEQ	MAINTUP	NYC2
FSTDIRRD	MAINTCN	FFA5
FSTENT	MAINTUP	NMD2
FTCH7A	DSERV4	RFD5
FULBLK	MAINTUP	NRA3
GET	MAINTUP	NSB1
GETAL	MAINTA	MAA3
GETASTRX	MAINTUPF	PDC4
GETCRD	MAINTUP	NTB1
GETCRD	MAINTUPF	PFB1
GETPASS	DSERV1	RAB5
GMESS7	MAINTUP	NSC3
GOON1	MAINTA	MCA1
GOTIT	MAINTCN	FKG5
GOTIT2	MAINTCN	FKE5
GRES	COPYSERV	AAD3
GRTEQU	MAINTUP	NZJ4
GTBK	MAINTUP	NMC4
GTCRD	MAINTUP	NGB5
HEADS	DSERV2	RBB3
HEADSF	DSERV2F	RCB3
INCORSEQ	MAINTUP	NRK4
INDSQ1	MAINTUP	NBE5
INITIAL	MAINTA	MPB5

Label	Phase	Location
INITIALH	MAINTA	MPG3
INSCHAR	MAINTUP	NNG2
INTL1	MAINTA	MEE3
INTL2	MAINTA	MEF3
INTL3	MAINTA	MEG3
IOEXEC	MAINTCN	FHB1
IPLDSTRY	MAINTCN	FLE2
LABEL4	MAINTA	MMG4
LADRESS	MAINTUP	NQG3
LASTWRT	MAINTUPF	PGC5
LBRBLKS	MAINTUPF	PCA3
LDADRES	MAINTUP	NQD4
LDEOF	MAINTUP	NAC1
LDEOF	MAINTUPF	PCB1
LDINST	MAINTUP	NFD3
LDLNK	MAINTUP	NJK3
LDPTR	MAINTUP	NFB1
LDREG	MAINTUP	NGH2
LEAVE	MAINTA	MMD5
LENGTH	MAINTUP	NMG2
LEN3	MAINTUP	NXF2
LHIGH	MAINTA	MPF3
LIB	MAINTA	MNC4
LIBERR	MAINTA	MNB4
LINK	MAINTUP	NJA4
LINKCH	MAINTCN	FHA2
LMAINDIR	MAINTCN	FDF4
LOADES	MAINTUP	NPG1
LOADF2	DSERV1	RAJ5
LOADINIT	MAINTCN	FJB4
LOADR	MAINTUP	NGK3
LOADREG	MAINTUP	NME5
LOADSTAT	DSERV1	RAH1
LOCBLK	MAINTUP	NWB3
LOGDONE	MAINTCN	FLC1
LOGGER	MAINTCN	FEC5
LOGUNIT	MAINTUPF	PCD1
LONECYL	MAINTA	MPA5
LSTDEL	MAINTUP	NLC4
LSTWRT	MAINTCN	FHF4
MAINCIL	MAINTCN	FFJ1
MAINLINE	MAINTCL	FAA1
MAINTA	MAINTA	MAB1
MLLSTENT	MAINTCN	FHA3
MNLINST	MAINTCN	FGB1
MNLNLP	MAINTCN	FGG1
MOROP	MAINTCL	FAH3
MOV	MAINTA	MAB4
MOVE	MAINTA	MKA1
MOVENTRY	MAINTUPF	PDA5
MOVE1	MAINTA	MKB1
MOVE10	MAINTA	MKD4
MOVE11	MAINTA	MKB5
MOVE12	MAINTA	MKK5
MOVE2	MAINTA	MKD1
MOVE3	MAINTA	MKE1
MOVE4	MAINTA	MKF1
MOVE5	MAINTA	MKF3
MOVE6	MAINTA	MKG3
MOVE7	MAINTA	MKG2
MOVE8	MAINTA	MKC2
MOVE9	MAINTA	MKA4
MOVING	MAINTUP	NAE1
MOVLAB	MAINTA	MPB3

Label	Phase	Location
MOVSEQ	MAINTUP	NKD2
MOV1	MAINTA	MAG5
MOV2	MAINTA	MAA4
MSGLD	MAINTUP	NDD1
MVALDC	MAINTA	MBF4
MVSQ	MAINTUP	NKG2
MVSQNO	MAINTUP	NCE1
NEWCARD	MAINTCL	FAH4
NEWENTRY	MAINTUPF	PDA3
NEWSDWRT	MAINTCN	FHF5
NEXTA	MAINTA	MCH5
NEXTAI	MAINTA	MDD1
NEXTAL	MAINTA	MCJ5
NEXTENT	DSERV2	RBH3
NEXTENTF	DSERV2F	RCH3
NEXTENTR	MAINTCN	FFC2
NEXTLIB	MAINTCN	FHJ5
NOBOOK	MAINTUP	NTB4
NOBOOK	MAINTUPF	PFF4
NOCID	MAINTA	MLC4
NOLIBE	MAINTUP	NAA4
NOPOST	MAINTCN	FHH1
NORECMSG	MAINTCN	FKB4
NOTFND	MAINTUP	NTE4
NOTFND	MAINTUPF	PFJ2
NOTH1	MAINTCL	FAG3
NOUPDTE	MAINTUPF	PDD1
NRECMSG	MAINTCN	FKH5
NRFERR	MAINTCN	FKK2
NWCARD	MAINTUP	NUB3
NWENTR	MAINTUP	NND4
NXTCMA	MAINTUP	NBJ2
NXTL	MAINTA	MBD5
OFFSWCH	MAINTUP	NZH2
OFLOCHK	MAINTCN	FGC3
ONETWO	MAINTUP	NXF3
OR F	DSERV1	RAC3
ORSWCH	MAINTUP	NMG3
OUTSEQ	MAINTUP	NJG1
PARENFD	MAINTUP	NTE1
PARENFD	MAINTUPF	PFE1
PCLIB	MAINTA	MBK5
PENULT	MAINTCN	FKH1
PERFND	MAINTUP	NED1
PLCONRTN	MAINTCN	FEA1
PLOK	MAINTCL	EYF4
PLPERR	MAINTCL	EYG2
PLREQ	MAINTCL	EYF1
PLUGCH	MAINTCN	FHD2
PRCLIB	COPYSERV	AAH3
PRINT	MAINTUP	NFE4
PRNTDEL	MAINTUP	NLB1
PRNTRTN	MAINTUP	NVB4
PROCADD	MAINTUP	NGA1
PROCDEL	MAINTUP	NLA2
PROCEND	MAINTUP	NMA1
PROCESS	DSERV2	RBD2
PROCESS	MAINTUP	NJE4
PROCESSF	DSERV2F	RCD2
PROCRDO	MAINTA	MAE1
PROCREP	MAINTUP	NJA2
PRTSTAT	MAINTUP	NGF4
PRVSET	MAINTCN	FDB4

Label	Phase	Location
PTRINIT	MAINTCN	FGH3
RCDSEQ	MAINTUP	NLE2
RDBK	MAINTUP	NMA4
RDDISK	MAINTUP	NFA3
RDSYSD	MAINTA	MCF1
RDSYS1	MAINTA	MCD4
RDSYS2	MAINTA	MCA5
RDSYS3	MAINTA	MCB5
RDSYS4	MAINTA	MCK4
RDSYS5	MAINTA	MCG5
RDSYS6	MAINTA	MDA3
RDSYS7	MAINTA	MDB4
READCARD	DSERV1	RAE5
READCHK	DSERV2	RBG2
READCHKF	DSERV2F	RCH2
READCKD	MAINTCL	FAD2
READDIR	MAINTCN	FFA2
READKEY	DSERV2	RBH2
READRTN	MAINTUP	NFH3
RECCHK	MAINTCN	FGA3
RECSEQ	MAINTUP	NGE2
RECSEQ1	MAINTUP	NGF2
REGINCR	MAINTUP	NEA4
REGLD	MAINTUP	NJF3
REGSAVE	MAINTUP	NWD3
REL/RELP	DSERV4	RFC2
RELIB	COPYSERV	AAE3
REPBKT	MAINTUP	NZE2
RESENTCT	MAINTCN	FMC5
RESETCD	DSERV2	RBH5
RESETCDF	DSERV2F	RCH5
RESETTD	DSERV2	RBJ5
RESETTDF	DSERV2F	RCJ5
RESTREC	MAINTCN	FMC2
RESTTRK	MAINTCN	FMD3
RETURN2	MAINTUP	NPD3
RGSTLD	MAINTUP	NLF3
RGTPREN	MAINTUP	NTG1
RGTPREN	MAINTUPF	PFG1
RLCONRTN	MAINTCN	FEA3
RLRREQ	MAINTCL	EYA1
RLSWT	MAINTA	MAH3
ROOTCNCL	MAINTA	MND2
RPSRTN	MAINTCN	FJH1
RSDIRRP	DSERV3F	REJ2
RSFETCH	DSERV3F	REE4
RSFETCH5	DSERV3	RDD2
RSSORT	DSERV3	RDC2
RSSORTF	DSERV3F	REK2
RTMAINT	MAINTUP	NNA1
RTMAINT	MAINTUPF	PDA1
RTMNT	MAINTUP	NPA4
RTNLNK	MAINTUP	NVB3
RTRN	MAINTUPF	PGJ2
SAVEDIR	MAINTUP	NUC1
SCNCARD	MAINTUP	NQF2
SCNDMSG	MAINTCN	FLB1
SDUPDATE	MAINTCN	FHB3
SDUPDATE	MAINTCN	FHC3
SEQCRD	MAINTUP	NKC2
SEQERR	MAINTUP	NBB5
SEQORD	MAINTUP	NZB2
SEQREAD	DSERV2	RBK4
SEQREC	MAINTUP	NJE3

Label	Phase	Location
SEQSWCH	MAINTUP	NBF5
SEQZRO0	MAINTUP	NHD1
SEQZRO	MAINTUP	NHC1
SETADDR	MAINTUPF	PCB3
SETCCBS	MAINTCN	FFA1
SETPTR	MAINTUP	NBF2
SETSEQ	MAINTUP	NDJ4
SETSWCH	MAINTUP	NMH3
SETSW2	MAINTUP	NDJ5
SKPLINE	MAINTUP	NKD4
SKSETUP	MAINTCN	FHF1
SLIB	MAINTA	MBG5
SORA/SORP	DSERV4	RFC3
SQNOMV	MAINTUP	NXF4
SRCLIB	COPYSERV	AAG2
SSCONRTN	MAINTCN	FEA4
START	MAINTCL	EXC1
STARTUP	MAINTUP	NAB1
STATUSB	MAINTUPF	PDA4
STRTM	MAINTA	MGH5
STSEQ	MAINTUP	NSF5
STSW2	MAINTUP	NBE2
SUBIT	MAINTA	MPC2
SUBIT2	MAINTA	MPH2
SYNERR4	MAINTA	MNB2
SYSCIL	MAINTCN	FDB5
SYSDIRUP	MAINTCN	FHG3
SYSDIRWT	MAINTCN	FKC2
TCDIRRD	DSERV2F	RCF2
TCINIT	DSERV2	RBC2
TCLINE	DSERV2	RBC3
TCLINEF	DSERV2F	RCC3
TEST	MAINTUP	NSB4
TKCOMP	MAINTA	MPB1
TKFMT	MAINTA	MLB1
TKFMT1	MAINTA	MLH2
TKFMT10	MAINTA	MLB2
TKFMT11	MAINTA	MLD2
TKFMT12	MAINTA	MLE2
TKFMT13	MAINTA	MLG2
TKFMT2	MAINTA	MMB1
TKFMT3	MAINTA	MMK1
TKFMT3A	MAINTA	MMF2
TKFMT4	MAINTA	MMK2
TKFMT6	MAINTA	MMJ3
TKFMT7	MAINTA	MMD1
TKFMT9	MAINTA	MLH1
TKRETN	MAINTA	MPJ2
TOBKT	MAINTUP	NXF1
TOPS	DSERV6	RGC2
TRANSL	MAINTUP	NBJ4
TRCKS	MAINTA	MBB1
TRNSLTE	MAINTUP	NDF2
TSTB	MAINTA	MBC2
TSTSWCH	MAINTUP	NXH2
TSTSW1	MAINTUP	NPA1
TYPCONCK	MAINTCN	FDC1
TYPEL	MAINTA	MAG2
UPADDR	MAINTUP	NRE2
UPDATE	MAINTA	MBH3
UPDSEQ	MAINTUP	NSB5
UPDTENT	MAINTUP	NNE3
UPD1	MAINTA	MAC3
UPD2	MAINTA	MAD5

Label	Phase	Location
UPSYSD	MAINTUP	NPJ1
UPSYSN	MAINTA	MFK4
UPSYS1	MAINTA	MGB3
UPSYS2	MAINTA	MGD4
UPSYS3	MAINTA	MGD2
VMBODTF	DSERV2F	RCF4
VMROOT	DSERV2	RBF4
WRAPARD	MAINTUP	NPE2
WRCHLVL	MAINTUP	NEH3
WRGORD	MAINTUP	NZK4
WRGSEQ	MAINTUP	NWG5
WRGUSEQ	MAINTUP	NRE4
WRITEP	MAINTA	MHA4
WRITER	MAINTA	MHA3
WRITES	MAINTA	MHF3
WRITE1	MAINTA	MHD2
WRITE2	MAINTA	MHA1
WRPARND	MAINTUP	NNJ2
WRSYSD	MAINTA	MHF2
WRTBLK	MAINTUPF	PGB4
WTBLOCK	MAINTUP	NUB1
WTDIREC	MAINTUPF	PDE5
WTMSG	MAINTUP	NPJ3
XTNERR	MAINTA	MNE3
ZEROMOD	MAINTUP	NPF1

PHASE TO MODULE CROSS REFERENCE

Phase	Module
$$BOPNLB	$$BOPNLB
$IJBLBSL	IJBLBSL
$MAINDIF	IJBLBI2
$MAINDIR	IJBLBIO
COPYSERV	IJBSMERG
CORGZ	IJBLBJ
CORGZ3	IJBLBT
CORGZ3F	IJBLBTF
CORGZ6	IJBLBW
CORGZ6F	IJBLBWF
CORGZ7	IJBLBX
CORGZ7F	IJBLBXF
CSERV	IJBLBP
CSERVC	IJBLBP
CSERVF	IJBLBP
DSERV	IJBSL1
DSERVC	IJBSL1
DSERVF	IJBSL1
DSERV1	IJBSL1
DSERV2	IJBSL1
DSERV2F	IJBSL1
DSERV3	IJBSL1
DSERV3F	IJBSL1
DSERV4	IJBSL1
DSERV5	IJBSL1
DSERV6	IJBSL1
MAINT	IJBLBA
MAINT	IJBLBC
MAINT	IJBMCS
MAINT	IJBMDS
MAINT	IJBMDU
MAINT	IJBMIN
MAINT	IJBMIO
MAINT	IJBMUP
MAINTA	IJBLBL
MAINTAF	IJBLBLF
MAINTCL	IJBLBM
MAINTCN/F	IJBLBG/F
MAINTDR/F	IJBLBD/F
MAINTP2/F	IJBLBN/F
MAINTR2/F	IJBLBE/F
MAINTS2/F	IJBLBF/F
MAINTUP	IJBLBQ
MAINTUPF	IJBLBQF
PSERV	IJBSL6
RSERV	IJBSL3
RSERVC	IJBSL3
RSERVF	IJBSL3

Internal Macros: (only the first two are explicitly documented in this manual)

DTFPL
DTFSL
FNDPL
FNDSL
GETSL
IJBCDT
IJBDISP
IJBLBCDY
IJBLBIOT
IJBLBOWA
IJBLBPDY
IJBLBRDY
IJBLBSDB
IJBLBSDY
IJBLBSTA
IJBLBSTT
IJBLBTAB
NTSL
READPL
READSL
REGISTERS
RTNCALL

Basic Macros: (not documented in this manual)

IJBLBDES
IVPCAPC
IVPCAPS

Link-books:

Phase	Module
**MAINT	IJBSL2
**SSERV	IJBSL4
**CORGZ	IJBSL5

MESSAGES CROSS REFERENCE

3A01	COPYSERV
3A02	COPYSERV
3A03	COPYSERV
3A04	COPYSERV
3C30	CORGZ
3C31	CORGZ
3C35	CORGZ3, CORGZ6(F)
3C66	CORGZ, CORGZ7(F)
3C67	CORGZ, CORGZ7
3M00	PSERV, MAINTCN(F)
3M09	SSERV, PSERV
3M10	DSERV, CORGZ, MAINT, MAINTDR(F), PSERV, RSERV, CSERV, SSERV
3M11	MAINTR2(F)
3M12	MAINTCN(F)
3M15	DSERV, MAINTA(F), MAINTDR(F), MAINTR2F, MAINTS2F, MAINTP2F, MAINT, CORGZ
3M16	MAINTAF, MAINTCNF, MAINTUP(F)
3M17	CORGZ, MAINTAF, MAINT
3M18	MAINTDRF, MAINTR2F, MAINTP2F, MAINTS2F, MAINTAF, MAINTCNF
3M20	PSERV
3M21	DSERV, MAINTCL, MAINTA(F), SSERV, PSERV, MAINTR2(F), MAINTDR(F), CSERV, MAINT, MAINTCN(F)
3M23	MAINTS2(F)
3M24	"
3M25	"
3M26	"
3M27	MAINTP2(F), MAINTR2(F), MAINTS2(F)
3M28	MAINTS2(F)
3M29	MAINTP2(F)
3M30	"
3M31	MAINT, MAINTP2(F)
3M32	MAINTP2(F)
3M33	$MAINDIR, CORGZ3(F), CORGZ6(F), MAINTUP(F), PSERV, CSERV, RSERV, SSERV
3M34	MAINT
3M35	DSERV
3M37	MAINTCN(F), MAINTCL
3M38	MAINTP2(F)
3M43	DSERV, MAINTCL, MAINTCN(F), SSERV, PSERV, MAINDIR(F), MAINTR2(F), MAINTS2(F), MAINTP2(F), CORGZ3(F), MAINTUP(F)
3M44	CORGZ, MAINTCN(F), MAINTDR(F)
3M45	DSERV
3M51	$MAINTDIR
3M52	MAINTR2(F), MAINTS2(F), MAINTP2(F), MAINTUP(F), CORGZ3(F),
3M53	CORGZ6(F)
3M54	$MAINDIR, MAINRDR(F), MAINTUP(F)
3M55	MAINTR2(F)
3M56	$MAINDIR
3M61	CORGZ7(F), MAINTAF
3M62	MAINTA
3M63	CORGZ7(F), MAINTA
3M64	MAINTA
3M65	CORGZ, CORGZ7(F)
3M66	CORGZ7(F)
3M68	MAINTA(F), MAINTCN(F)
3M70	MAINTCN(F), MAINT, MAINTA
3M75	MAINTCN(F)
3M78	$LIBSTAT
3M80	MAINTA(F), MAINTCN(F)
3M81	MAINTCN
3M90	$MAINDIR
3M91	$MAINDIR/DIF
3M92	$MAINDIR
3U10	$MAINTUP(F)
3U11	$MAINTUP(F)
3U20	$MAINTUP(F)
3U21	$MAINTUP(F)
3U30	$MAINTUP(F)
3U31	$MAINTUP(F)
3U32	$MAINTUP(F)

INTERNAL LIBRARIAN ERROR AND RETURN CODES

RETURN CODES WITH MESSAGE 3M17I:

If a librarian program during execution discovers an invalid program status or invalid program data the message 3M17I INTERNAL LIBRARIAN ERROR XXXXX is issued and an Error Code is placed in register 15. Following is a list of these error codes:

```
CORGZ  : X'600' GETVCE macro received non-zero return code for SYSRES.

CORGZ3F: X'615' Contents of 'FROM' library have been changed during
                processing so that a member formerly found is no
                longer there.

CORGZ6F: X'630' End of 'FROM' directory was not found.

MAINTAF: X'110' Invalid table status or data found. Reg 14 then contains
                the address where the error was detected.

DSERV  : X'901' Neither on CKD nor on FBA a library is specified.

DSERV3 : X'906' The phase is called, but no related library on CKD (RL,
                SL, PL, or private RL or SL) is specified (Initialize).

DSERV3 : X'907' The phase is called, but no related library on CKD (RL,
                SL, PL, or private RL or SL) is specified (Sort).

DSERV3 : X'908' The phase is called, but no related library on CKD, (RL,
                SL, PL, or private RL or SL) is specified (Fetch).

DSERV4 : X'909' The phase is called, but no RL or SL or private RL or
                SL is specified.

DSERV3F: X'902' The phase is called, but no related library on FBA (RL,
                SL, PL, or private RL or SL) is specified (Initialize
                print).

DSERV3F: X'904' The phase is called, but no related library on FBA (RL,
                SL, PL, or private RL or SL) is specified (Sort).

       ) X'101' Nesting depth of NOTE exceeded (user error)
       )
SSERV  ) X'102' NOTE, POINT or READ given, but no expansion for it (user)
       )
DTFSL  ) X'103' Too many POINT given (user error)
       )
ASSEMB.) X'104' Invalid request or library OPEN failure (user/system)
       ) X'105' GET, NOTE, READ, or POINT given without FIND (user)
```

RETURN CODES FOR $MAINDIR/$MAINDIF

At the end of execution, the phase inserts one of the following codes in register 15:

X'00'	successful completion
X'04'	status report is to be displayed
X'08'	directory is full
X'10'	irrecoverable I/O error

$

$$BOPNLB transient
   chart 130
   text 38
$$BSYSWR transient
   chart 129
   text 38
$IJBLBSL
   charts 134
   text 42
$LIBSTAT
   chart 90
   text 28
$MAINDIF
   charts 101
   text 29
$MAINDIR
   charts 91
   text 29

A

allocation of library space 25

B

book in source library 229

C

CKD libraries 219
compressed format for SSL 229
control statements (see DOS/VS System Control Statements manual)
)ADD 28
)DEL 28
)END 28
)REP 28
   ALLOC (CORGZ) 19
   (MAINT) 23, 25, 30
   CATALP 24
   CATALR 23
   CATALS 24
   CONDL 22
   CONDS 23, 30
   COPYC 19, 30
   COPYx 18
   DELETC 24, 30
   DELETx 24
   DSPCH 32
   DSPLY 32
   MERGE 19
   NEWVOL 19
   PUNCH 32
   RENAMC 24, 30
   RENAMx 24
   UPDATE 28
conventions
   flow charts 137
   general charts 43
COPYSERV program
   charts 44, 138
   text 13
core image library
   CKD 219
   FBA 236, 238, 239
CORGZ program
   charts 45
   text 15
CSERV program
   charts 120
   text 36

D

data areas 202
descriptor record, see directory CKD and FBA
directory, CKD
   core image 219
   procedure 230
   relocatable 222
   source statement 228
directory, FBA
   records 236
   space 231
directory, system 217
DSERV program
   charts 113, 195
   text 33
DTFSL macro
   charts 131
   text 39

E

error codes 250
error messages cross reference 249
ESD record format 225

F

FBA libraries 231
flowchart symbols 137
flowchart label list 243

G

general charts conventions 43

I

INITABLE 202
I/O access to librarian files 11

I/O areas in $MAINDIR /$MAINDIF 31

L

label information area 215, 217
label lists 240
librarian programs
   COPYSERV 13
   CORGZ 15
   CSERV 36
   DSERV 33
   MAINT 19
   PSERV 37
   RSERV 36
   SSERV 37
libraries on CKD
   core image 219
   procedure 230
   relocatable 222
   source (statement) 228
libraries on FBA 231
libraries private 238
library descriptor
   CKD core image library 219
      procedure library 230
      relocatable library 222
      source library 228
   FBA 231
library record sizes 239
library status table 206

M

macros 39
MAINT program 19
member space on CKD
   core image library 220
   procedure library 230
   relocatable library 222
   source library 228
member space on FBA 238
messages cross reference 249

N

note word table 40

P

phases, see Contents
phase to module cross reference 248
private libraries 238
procedure library
   CKD 230
   FBA 231, 237
programs, see librarian programs
PSERV program
   charts 127
   text 37

R

reallocation of SYSRES 26
reallocation tables 204
record formats 239
relocatable library
   CKD 222
   FBA 231, 237
return codes 250
RLD record format 227
RPS support 9
RSERV program
   charts 122
   text 36

S

SDL, build 29
source statement library
   CKD 228
   FBA 231, 237
SSERV program
   charts 125
   text 37
stow table (STOWTAB) 32, 206
SVASTAT 29
switches 209
SYSRES
   layout, CKD 215
           FBA 216
   reallocation 26
system directory 217

T

TABIN array 209
twin phases 9
TXT record format 226

SY33-8557-4

DOS/VSE Librarian Logic Printed in U.S.A. SY33-8557-4

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

DOS/VSE Librarian Logic

READER'S COMMENT FORM

SY33-8557-4

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Name

Address

What is your occupation?

How did you use this manual?

As a reference source

As a classroom text

As a self-study text

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text
Organization of the text

Accuracy
Cross-references

Index
Tables

Illustrations
Examples

Appearance
Printing

Paper
Binding

SY33-8557-4

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for system analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

CUT ALONG THIS LINE

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 BP

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601