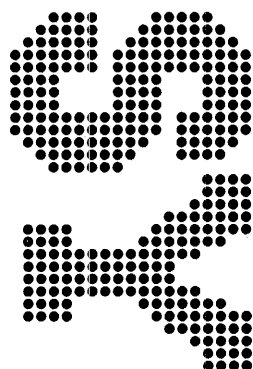


IBM System/3 Disk Systems System Control Program Logic Manual

**Program Numbers 5702-SC1
5703-SC1**

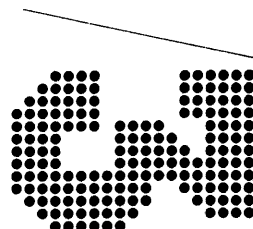
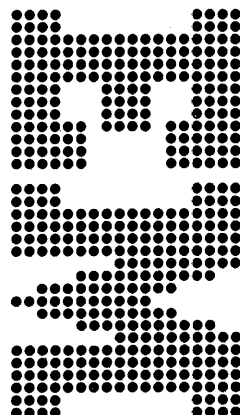


This logic manual is intended for IBM support personnel responsible for maintaining the system control programs for IBM System/3 Disk System and IBM System/3 Model 6. The system control programs include: Initial Program Load Bootstrap Program, Nucleus Initialization Program, Supervisors, Scheduler, Linkage Editor, and System Generation.

Maintenance programs, data areas, and diagnostic aids are also discussed.



This manual is intended to be a recall mechanism and a guide to program listings. It is not intended to be used as a coding or operating guide for the programs described.



Third Edition (March 1971)

This is a major revision of, and obsoletes, SY21-0502-1. The manual is now designed to satisfy requirements of support personnel responsible for maintenance of the IBM System/3 Disk System or the IBM System/3 Model 6.

This edition applies to version 03, modification 00 of IBM System/3 Disk System, Program Number 5702-SC1, and IBM System/3 Model 6, Program Number 5703-SC1, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM Systems, consult the latest IBM System/3 Newsletter, Order Number GN20-2228 for the editions that are applicable and current. (Although this is the first release of IBM System/3 Model 6, it is identified as version 03, modification 00 for purposes of compatibility with the Model 10 Disk System.)

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

© Copyright International Business Machines Corporation 1970, 1971

This Program Logic Manual is designed to satisfy the documentation requirements of the support personnel responsible for the maintenance of the IBM System/3 System Programs.

RELATED PUBLICATIONS

This publication references the following publications:

- *IBM System/3 Disk System Operation Control Language and Disk Utilities Reference Manual*, GC21-7512.
- *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual*, GC21-7516.
- *IBM System/3 Disk Systems Data Management and Input/Output Supervision Logic Manual*, SY21-0512.
- *IBM System/3 Disk System Halt Procedure Guide*, GC21-7540.
- *IBM System/3 Model 6 Halt Procedure Guide*, GC21-7541.

Part 1. Initial Program Load Bootstrap Program and Nucleus Initialization Program	1
Part 2. Disk System Dedicated Supervisor	2
Part 3. Disk System DPF Supervisor	3
Part 4. Model 6 Supervisor	4
Part 5. Scheduler	5
Part 6. Transient and Scheduler Support.	6
Part 7. Linkage Editor	7
Part 8. System Generation	8
Part 9. Maintenance Programs	9
Appendix A. Data Area Formats	A
Appendix B. Diagnostic Aids	B
Appendix C. Abbreviations and Terminology	C
Appendix D. Flowcharting Techniques	D

Contents

How This Manual is Organized	ix	PART 4. MODEL 6 SUPERVISOR	4-1
How the IBM System/3 Disk Systems are Organized	xi	SECTION 1. INTRODUCTION	4-3
PART 1. INITIAL PROGRAM LOAD BOOTSTRAP PROGRAM AND NUCLEUS INITIALIZATION PROGRAM		System Requirements	4-3
	1-1	SECTION 2. METHOD OF OPERATION	4-5
SECTION 1. INTRODUCTION	1-3	Load or Fetch With Find	4-5
System Requirements	1-3	Load or Fetch Only	4-5
		Transient Requests	4-6
SECTION 2. PROGRAM ORGANIZATION	1-5	SECTION 3. PROGRAM ORGANIZATION	4-7
Initial Program Load (IPL) Bootstrap Program—Disk System	1-5	General Entry/Exit and Transient Area Scheduler (NENTRY)	4-7
Initial Program Load (IPL) Bootstrap Program—Model 6	1-5	Resident Loader and Relocation Program (NLOADR)	4-8
Nucleus Initialization Program (IPLNIP)—Disk System	1-5	Local Storage Register Display Routine (LSR)	4-8
Nucleus Initialization Program (IPLNIP)—Model 6	1-6	Resident Dump Linkage (CDUMPD)	4-8
		System Disk Read Routine (NREAD)	4-8
SECTION 4. DIRECTORY	1-9	SECTION 4. DIRECTORY	4-9
PART 2. DISK DEDICATED SUPERVISOR	2-1	PART 5. SCHEDULER	5-1
SECTION 1. INTRODUCTION	2-3	SECTION 1. INTRODUCTION	5-3
System Requirements	2-3	System Requirements	5-3
SECTION 2. METHOD OF OPERATION	2-5	SECTION 2. METHOD OF OPERATION	5-5
Load or Fetch With Find	2-5	Terminator	5-6
Load or Fetch Only	2-5	Reader/Interpreter	5-6
Transient Requests	2-6	Conversational Reader/Interpreter	5-6
		Disk System Reader/Interpreter Mainline (\$\$RDML)	5-6
SECTION 3. PROGRAM ORGANIZATION	2-9	Initiator	5-7
General Entry/Exit and Transient Area Scheduler (NENTRY)	2-9	SECTION 3. PROGRAM ORGANIZATION	5-11
Resident Loader and Relocation Program (NLOADR)	2-10	1. Terminator	5-12
Local Storage Register Display Routine (LSR)	2-10	Terminator Initial Program Load (IPL) (\$\$TMIP)	5-12
Resident Dump Linkage (CDUMPD)	2-10	Terminator End of Job—Phase One (\$\$TMEO)	5-13
System Disk Read Routine (NREAD)	2-10	Terminator End of Job—Phase Two (\$\$TMEJ)	5-13
SECTION 4. DIRECTORY	2-17	Terminator Re-Initialization (\$\$TMRI)	5-14
PART 3. DISK DPF SUPERVISOR	3-1	Terminator Standby Routine (\$\$TMSB)	5-15
SECTION 1. INTRODUCTION	3-3	Terminator Sysin Initialization (\$\$TMSI)	5-16
System Requirements	3-3	Terminator Key Sort (\$\$TMSK)	5-17
SECTION 2. METHOD OF OPERATION	3-5	Terminator Duplicate Key Scan (\$\$TMDS)	5-17
SECTION 3. PROGRAM ORGANIZATION	3-7	2. Reader/Interpreter for Model 6 (Conversational Mode)	5-18
General Entry/Exit and Transient Area Scheduler (NENTRY)	3-7	Conversational Scheduler Initializer (\$\$RBIP)	5-18
Resident Loader and Relocation Program (NLOADR)	3-7	Scheduler Control (\$\$RBCO)	5-19
Resident Halt Display Routine (HPLNUC)	3-8	Build-Chained Cycle (\$\$RBBC)	5-20
Terminator Interrupt Handler (INTLO)	3-8	Call Control (\$\$RBCC)	5-21
SECTION 4. DIRECTORY	3-17	Call Cycle (\$\$RBCL)	5-22
		Positional Statement Processor (\$\$RBPC)	5-23
		Data Type Statement Processor (\$\$RBCD)	5-24
		Keyword Type Statement Processor (\$\$RBCK)	5-25
		Modify (\$\$RBMY)	5-26
		Procedure Library Scheduler Interface (\$\$RBM1)	5-27
		Utility OCL Processor (\$\$RBM2)	5-27
		Run Processor (\$\$RBRN)	5-28
		Interaction Routine (\$\$RBIN)	5-29
		Date/Switch (\$\$RBDS)	5-29
		Switch Control Statement Processor (\$\$RBSW)	5-30

Date Scan (\$\$RBDT)	5-30	Halt/Syslog—Halt Code Conversion Routine (\$\$STOK)	6-8
Log (\$\$RBLG)	5-30	Model 6 Halt/Syslog Halt Routine—Keyboard (\$\$STOS)	6-10
FORMS (\$\$RBFM)	5-31	Model 6 Halt/Syslog Routine—Matrix Printer (\$\$STOM)	6-10
HIKEY (\$\$RBHI)	5-32	Model 6 Halt/Syslog Routine—Output Only, Matrix Printer (\$\$STON)	6-10
3. Reader/Interpreter for Disk System	5-33	Model 6 Halt/Syslog Halt Routine—CRT (\$\$STOT)	6-10
Reader/Interpreter Mainline (\$\$RDML)	5-33	Model 6 Halt/Syslog Routine—CRT On and Off Transient (\$\$STOX)	6-10
// LOAD Control Card Processor (\$\$RDLD)	5-33	Model 6 Halt/Syslog Scheduler Error Message Routine (\$\$SLSE, \$\$SLS1)	6-10
// RUN Control Card Processor (\$\$RDRN)	5-34	Scheduler Work Area—Get (\$\$SSGT)	6-11
// FILE Control Card Processor (\$\$RDFL)	5-35	Scheduler Work Area—Put (\$\$SSPT)	6-12
// DATE Control Card Processor (\$\$RDDT)	5-36	Scheduler Work Area—Read/Write (\$\$SSSC)	6-13
// SWITCH Control Card Processor (\$\$RDSW)	5-37	Volume Table of Contents—Read/Write (\$\$SSVT)	6-14
// IMAGE Control Card Processor (\$\$RDIM)	5-37	Rollin—Phase One (\$\$STRI)	6-15
// COMPILE Control Card Processor (\$\$RDCM)	5-38	Rollin—Phase Two (\$\$STRN)	6-15
// FORMS Control Card Processor (\$\$RDFM)	5-38	Rollout—Phase One (\$\$STRO)	6-15
// LOG Control Card Processor (\$\$RDLG)	5-39	Rollout—Phase Two (\$\$STRU)	6-15
// PAUSE Control Card Processor (\$\$RDPS)	5-40	Rollout—Phase Three (\$\$STRT)	6-15
// HALT, // NOHALT Control Card Processor (\$\$RDHN)	5-40	Disk System System List—Print (\$\$SYP1)	6-16
// READER Control Card Processor (\$\$RDRR)	5-41	Model 6 System List—Print (\$\$SYP2)	6-16
// PARTITION Control Card Processor (\$\$RDPN)	5-42	Disk System System List—Print/Punch (\$\$SPC1)	6-17
// CALL Control Card Processor (\$\$RDCL)	5-43	Model 6 System List—Punch Routine (\$\$SPC2)	6-18
Positional Parameter Syntax Scan Routine (\$\$RDS3)	5-44	Source Library Get (\$\$SYSG)	6-19
FILE Diagnostics and Merge Keyword Parameters (\$\$RDMK)	5-45	Find Transient (\$\$SPFN)	6-19
4. Initiator	5-46	Allocate Initiator (\$\$STAI)	6-20
Initiator Allocate—Phase One Routine (\$\$INA1)	5-46	LOAD * Mainline (\$\$SYLA)	6-21
Initiator Allocate—Phase Two (\$\$INA2)	5-47	End of Job Transient (\$\$SPEJ)	6-21
Initiator Allocate—Phase Three (\$\$INA3)	5-48	End of Job—Disk Status (\$\$TMST)	6-22
Initiator Allocate—Phase Four (\$\$INA4)	5-49	Copy Main Storage to Source Library (\$\$SYSP)	6-22
Initiator Disk File Processor—Phase One (\$\$INPD)	5-50	Resource Allocation—Phase One (\$\$STR1)	6-23
Initiator Disk File Processor—Phase Two (\$\$INDF)	5-50	Resource Allocation—Phase Two (\$\$STR2)	6-23
Initiator Disk File Processor—Phase Three (\$\$INDC)	5-51	Transient Resolver (\$\$OXRF)	6-23
Initiator Disk File Processor—Phase Four (\$\$INDS)	5-51	Allocate Terminator (\$\$INAT)	6-24
Initiator Program Setup—Phase One (\$\$INPS)	5-52	System Queue Routine (\$\$STNQ)	6-24
Initiator Program Setup—Phase Two (\$\$INP2)	5-52	Halt/Syslog Parameter Build Routine (\$\$STHB)	6-25
Initiator Allocate—Support Routines (\$\$INMS)	5-53	Keyword Syntax Scan Routine (\$\$RDS1)	6-25
Convert Records to Tracks	5-53	Transient Resolver No-Op Routine (\$\$ODNP)	6-26
Determine Size of Index	5-54	HIKEY Transient (\$\$STF7)	6-26
Track Apportioning	5-54	HIKEY Parameter Scan Routine (\$\$RDHK)	6-27
Track Conversion	5-54	Model 6 Logical Put Routine (\$\$RBLLP)	6-27
Set Bits Routine	5-54	Model 6 Syntax Checker (\$\$RBSX)	6-28
Possible Allocation Fit	5-54	Model 6 Syntax Checker Interface (\$\$RBSI)	6-29
SECTION 4. DIRECTORY	5-183	Model 6 Library Interaction Routine (\$\$RBLI)	6-29
PART 6. TRANSIENT AND SCHEDULER SUPPORT	6-1	SECTION 3. DIRECTORY	6-111
SECTION 1. INTRODUCTION	6-3	PART 7. LINKAGE EDITOR	7-1
SECTION 2. PROGRAM ORGANIZATION	6-5	SECTION 1. INTRODUCTION	7-3
System Input Device—Console (\$\$STIC)	6-5	Control Records—Input to the Linkage Editor	7-3
System Input Device—MFCU/1442 (\$\$STIM)	6-5	PHASE Control Records	7-3
Model 6 System Input Device—Keyboard/Printer (\$\$STIP, \$\$STIS)	6-6	INCLD Control Records	7-4
Model 6 System Input Device—Data Recorder (\$\$STID)	6-6	ENTRY Control Record	7-4
Model 6 System Input Device—CRT (\$\$STIT, \$\$STIX)	6-7	OPTNS Control Record	7-4
System Output Printer Error Recovery (\$\$STEP)	6-7	Source Records—Input to the Linkage Editor	7-5
Halt/Syslog—Console Printer (\$\$STOC)	6-7	External Symbol List (ESL) Input Records	7-5
Halt/Syslog—Printer (\$\$STOP)	6-7	TEXT-RLD Input Records	7-5
System Halt Transient (\$\$STOH)	6-8	END Input Record	7-5
Model 6 Printer Syslog Error Recovery Procedures Routine (\$\$STP6)	6-8	Output From the Linkage Editor	7-6
		System Requirements	7-6

SECTION 2. METHOD OF OPERATION	7-7	SECTION 3. PROGRAM ORGANIZATION	8-13
PASS1 Root Phase (\$LINKB)	7-8	System Generation	8-13
INCLD and ENTRY Control Record and AUTOLINK Processor (\$LINKC)	7-8	Keyboard Selection Routine (\$SGKBD)—Model 6 System Routine Only	8-13
TEXT-RLD and END Control Records Processor (\$LINKD)	7-10	Co-Resident System Analyzer (\$SGBSV)—Model 6 System Routine Only	8-13
ESL and OPTNS Control Record Processor (\$LINKE)	7-11	System Generation—Phase One (\$SGEN for the Disk System, \$SGENB for the Model 6 System)	8-14
PHASE Control Record Processor (\$LINKF)	7-11	System Generation—Phase Two (\$SGEN1)	8-15
External Symbol List (ESL) Table—Process Phase (\$LINKM)	7-12	System Generation Error Note Processor (\$SGNOT)	8-15
PASS2 Root Phase (\$LINKG)	7-12	Pre-Processor Routines	8-16
Error and Overlay Fetch Table—Print Phase (\$LINKK) Punch Processor (\$LINKN)	7-12	Pre-Processor Mainline—Phase One (\$SGROC)	8-16
		Pre-Processor Mainline—Phase Two (\$SGMN1)	8-16
		Pre-Processor Mainline—Phase Three (\$SGMN2)	8-17
		Pre-Processor Mainline—Phase Four (\$SGMN3)	8-18
		Get Variable Symbol Routine (\$SGGSY)	8-18
SECTION 3. PROGRAM ORGANIZATION	7-13	Variable Symbol Table—Find/Build Routine (\$SGVST)	8-18
PASS1—Root and Initialization Phase (\$LINKB)	7-13	Variable Symbol Table—Value Build Routine (\$SGGVA)	8-19
PASS1—INCLD and ENTRY Control Record and AUTOLINK Processor (\$LINKC)	7-14	Number—Check/Build Routine (\$SGNCB)	8-19
PASS1—TEXT-RLD and END Record Processor (\$LINKD)	7-14	Character String—Check/Build Routine (\$SGCSB)	8-20
PASS1—ESL and OPTNS Record Processor (\$LINKE)	7-15	Read Pre-Processor Statement Routine (\$SGRED)	8-20
PASS1—PHASE Control Record Processor (\$LINKF)	7-15	Link Statement Processor (\$SGSTM)	8-20
PASS2—Root Phase (\$LINKG)	7-16	Prototype Statement Processor (\$SGROT)	8-20
PASS2—TEXT-RLD Conversion Phase (\$LINKH)	7-17	Keyword Operand Processor (\$SGOPR)	8-21
PASS2—Directory Build Phase (\$LINKJ)	7-17	AIF Statement Processor (\$SGAIF)	8-21
Error and Overlay Fetch Table—Print Phase (\$LINKK). External Symbol List (ESL) Table—Process Phase (\$LINKM)	7-18	Model Statement Build Routine (\$SGMST)	8-21
Punch Output Processor (\$LINKN)	7-18	Subfield Analyzer (\$SGSUB)	8-22
		Model Statement Output Processor (\$SGOUT)	8-22
		Table Statement Processor (\$SGTBL)	8-22
		Table Definition Processor (\$SGDEF)	8-22
SECTION 4. DIRECTORY	7-55	AGO Statement Processor (\$SGAGO)	8-23
		MNOTE Statement Processor (\$SGNTT)	8-23
PART 8. SYSTEM GENERATION	8-1	SETB Statement Processor (\$SGSET)	8-24
		Global and Local Statement Processor (\$SGGBL)	8-24
SECTION 1. INTRODUCTION	8-3	IPL Bootstrap Modify (\$SGPRI)—Model 6 Only	8-24
Disk System	8-3	System Verification Message Routine (\$SGIVP)	8-24
Model 6 System	8-3		
Control Records Used by System Generation	8-4	SECTION 4. DIRECTORY	8-59
LINK—Definition Header Record	8-4		
MEND—LINK-Definition End Record	8-4	PART 9. MAINTENANCE PROGRAMS	9-1
Keyword Prototype Record	8-4		
System Configuration Statement	8-5	SECTION 1. INTRODUCTION	9-3
TABLE Record	8-6		
TABLE Definition Record	8-6	SECTION 2. PROGRAM ORGANIZATION	9-5
TEXT Record	8-6	Field Engineering Maintenance Program (\$SGPTF)	9-5
AGO Record—Unconditional Branch	8-6	PTF Logging Program (\$SGPTG)	9-6
ANOP Record—Assembly No Operation	8-6	System Maintenance (\$SGMNT)	9-6
AIF Record—Conditional Branch	8-6		
GBLB or LCLB Set Records	8-8	APPENDIX A. DATA AREAS	A-1
SETB Record—Set Binary	8-8		
Comment Records	8-8	DISK SYSTEM DEDICATED SUPERVISOR DATA AREA FORMATS	A-3
MEXIT Record—Link-Definition Exit	8-8	Supervisor Entry Points	A-3
MNOTE Record	8-8	Transient Area	A-3
Model Records	8-8	Chain Image Area	A-3
System Residence Disk Pack Organization	8-9	System Communication Region—NCPL1	A-3
System Requirements	8-10	Program Level Communication Region—N1COMN	A-3
SECTION 2. METHOD OF OPERATION	8-11		

MODEL 6 SUPERVISOR DATA AREA FORMATS	A-9	APPENDIX B. DIAGNOSTIC AIDS	B-1
Supervisor Entry Points	A-9	Current Phase Identification Procedure	B-2
Transient Area	A-9	Main Storage Dump (Phase Identification)	B-2
Keyboard Translate Table	A-9	Console Address/Data Switches (Phase Identification)	B-2
System Communication Region—NCPL1	A-9	Halt/Syslog Procedure	B-3
Program Level Communication Region—N1COMN	A-11	Setting Up Halt/Syslog Device for Logging	B-3
SCHEDULER DATA AREA FORMATS	A-15	Calling Halt/Syslog Routine	B-3
Configuration Record	A-15	Halt/Syslog Parameter Formats	B-4
Volume Label	A-19	Format of the Output from Halt/Syslog	B-7
Scheduler Work Area (SWA)	A-19	Local Storage Register Display Procedure	B-8
Volume Table of Contents—VTOC	A-26	Tracing the Transient Queue	B-8
System Communication Region	A-28	Storage Dump Selection Procedure	B-10
Program Level Communication Region	A-28	Disk System	B-10
Reader/Interpreter Work Area (RDIWA)	A-28	Model 6	B-10
Initial Halt/Syslog Parameter Table	A-31	Linkage for the Storage Dump Routines	B-11
HALT/SYSLOG Parameter Table Three (\$\$STH2)	A-34	Disk System	B-11
HALT/SYSLOG Parameter Table Four (\$\$STH3)	A-35	Model 6	B-11
HALT/SYSLOG Parameter Table Five (\$\$STH4)	A-36	Dump Selection Routine (DMPFND)	B-11
Call Common Area	A-37	Main Storage Dump Routine (DUMPPG) (Option 0)	B-11
Scheduler Control Table	A-38	DTF Dump Routine (\$\$SPDF on the Disk System, \$\$SPD3 on Model 6) (Option 1)	B-11
LINKAGE EDITOR DATA AREA FORMATS	A-41	Disk Storage Dump Routine (\$\$SPD4 on the Disk System, \$\$SPD2, Model 6) (Option 2)	B-12
PASS1 Communication Region (COMMON)	A-41	Dump Format Routine (\$\$SPD1, Disk System, \$\$SPD0, Model 6)	B-13
PASS2 Communication Region (COMMON)	A-42	Request Indicator Byte (RIB)	B-14
Input/Output Buffer (BUFFER)	A-44	The RIB Fetch/Trace Procedure	B-16
Error Table (ERRTAB)	A-44	RIB Fetch/Trace Routine (\$\$FTRC)	B-17
Print Buffer (PRTBUF)	A-44	Program Temporary Patch Program (\$\$SGFIX)	B-17
External Symbol List (ESL) Table (ESLTAB)	A-45	Program Temporary Patch (\$\$SGFIX) Description and Procedures	B-26
Text Work Area (TXTWRK)	A-45	HDR O program name,unit	B-26
Relocation Directory (RLD) Work Area (RLDWRK)	A-45	PTF Module Identification	B-26
Language Translator Output Work Area (\$WORK)	A-45	DATA Disp,chkbyte,hh,hhhh,hhhhR	B-26
Object Library Directory	A-46	END	B-26
RPG Overlay Fetch Table	A-47	/*	B-26
SYSTEM GENERATION DATA AREA FORMATS	A-49	Procedures for Running \$\$SGFIX	B-27
Pre-Processor Communication Region (RDPARM)	A-49	Maintenance Program Procedure	B-27
Pre-Processor Statement Input Buffer (BUFFR1)	A-51	Disk System	B-27
Statement Save Area (SVPROT)	A-51	Model 6	B-27
System Generation Communication Region (COMMON)	A-51	System Generation '1 Halt Procedure	B-28
Pre-Processor Switch Area (SWADR)	A-52	GM'1 Halt	B-28
Input Work Area (INBUF)	A-52	GG'1 Halt	B-30
Output Work Area (OUTBUF)	A-53	GS'1 Halt	B-31
Variable Symbol Table	A-53	System Halts and the Modules that can Issue Them	B-31
Source Library	A-53	Error Messages	B-42
Translate Table 1—Domestic (English)	A-55	Table of Light Identifiers	B-45
Translate Table 2—Austria/Germany	A-56	Disk System	B-45
Translate Table 3—Belgium/France	A-57	Model 6	B-45
Translate Table 4—Denmark	A-58	Reader/Interpreter CR96 Error Procedure	B-46
Translate Table 5—Norway	A-59	APPENDIX C. ABBREVIATIONS AND TERMINOLOGY	C-1
Translate Table 6—Finland/Sweden	A-60	APPENDIX D. FLOWCHARTING TECHNIQUES	D-1
Translate Table 7—Spain	A-61		
Translate Table 8—Brazil/Portugal	A-62		
Translate Table 9—United Kingdom	A-63		



HOW THIS MANUAL IS ORGANIZED



This publication is comprised of nine parts and four appendices. Except as noted under *Deviation*, each of the parts is divided into four sections:

- *Section 1 – Introduction* contains general information about the functions and characteristics of the program.
- *Section 2 – Method of Operation* describes the data and functional flow of the program in general terms, emphasizing the use of the data areas.
- *Section 3 – Program Organization* describes the organization of each routine using narrative, flowcharts and diagrams. Flowcharts are designed to provide convenient entry into the program listings.
- *Section 4 – Directory* is included as a quick-reference guide to the information contained in the logic manual. Information in this section includes the following:

Module/Phase Name


Chart ID

Descriptive Name

Entry Point

Function

Deviations


- Part 1, *Initial Program Load Bootstrap Program and Nucleus Initialization Program*, does not contain a *Method of Operation* section.
 - Part 6, *Transient and Scheduler Support*, does not contain a *Method of Operation* section.
 - Part 9, *Maintenance Programs*, does not contain a *Method of Operation* section or a *Directory*.
- 

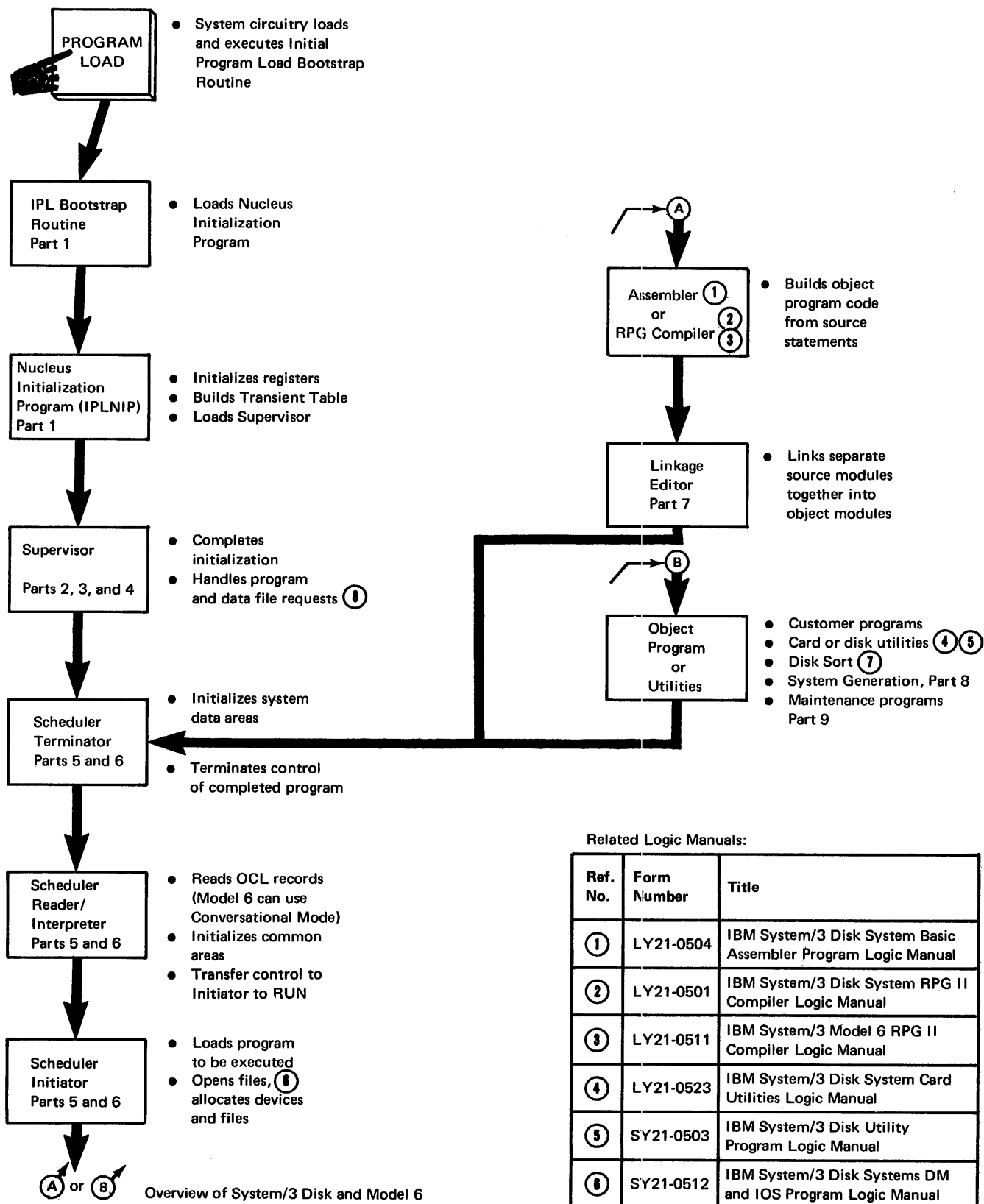


HOW THE IBM SYSTEM/3 DISK SYSTEMS ARE ORGANIZED



The following chart illustrates:

- The sequence in which the major system programs are called.
 - How the system programs are related to the User's programs.
 - The IBM System/3 publications associated with the functions being described.
- 



Related Logic Manuals:

Ref. No.	Form Number	Title
①	LY21-0504	IBM System/3 Disk System Basic Assembler Program Logic Manual
②	LY21-0501	IBM System/3 Disk System RPG II Compiler Logic Manual
③	LY21-0511	IBM System/3 Model 6 RPG II Compiler Logic Manual
④	LY21-0523	IBM System/3 Disk System Card Utilities Logic Manual
⑤	SY21-0503	IBM System/3 Disk Utility Program Logic Manual
⑥	SY21-0512	IBM System/3 Disk Systems DM and IOS Program Logic Manual
⑦	LY21-0517	IBM System/3 Disk Sort Program Logic Manual

PART 1.

**INITIAL PROGRAM LOAD BOOTSTRAP PROGRAM
AND NUCLEUS INITIALIZATION PROGRAM**

The Initial Program Load (IPL) process loads the user's programming system into main storage. The user selects the disk (F1 or R1) from which his system is to be loaded. When the PROGRAM LOAD key on the Disk System is pressed, or the PROGRAM LOAD switch on Model 6 is set, system hardware loads the IPL Bootstrap Program from disk, placing it at location X'0000'. The IPL Bootstrap Program then loads the Nucleus Initialization Program (IPLNIP) and passes control to it. IPLNIP then loads and passes control to the Supervisor program at X'0380'.

SYSTEM REQUIREMENTS

The system requirements for the IPL process are:

Disk System

- One 5203 Line Printer
- One 5410 Processing Unit (Model A 13)
- One 5444 Disk Storage Drive (Model 2)

Model 6

- One 5213 Matrix Printer
- One 5406 Processing Unit with Keyboard
- One 5444 Disk Storage Drive (Model 2)

► **Initial Program Load (IPL) Bootstrap Program—Disk System**

CHART: None

ENTRY POINT: IPLSRT

FUNCTIONS:

- This routine is automatically read into main storage at location X'0000' after the user presses PROGRAM LOAD.
- IPL Bootstrap then relocates itself to location X'1200'.
- This routine then loads the Nucleus Initialization Program (IPLNIP) at location X'1800'.

INPUT: None

OUTPUT: System Q code passed to IPLNIP

EXITS: Control is passed to location X'1800'.

► **Initial Program Load (IPL) Bootstrap Program—Model 6**

CHART: None

ENTRY POINT: IPLSRT

FUNCTIONS:

- This routine is automatically read into main storage at location X'0000' after the user presses PROGRAM LOAD.
- IPL Bootstrap then relocates itself to location X'1200'.
- IPL Bootstrap determines whether DSM or BASIC system is being used.
- IPL Bootstrap loads DSM IPLNIP at location X'1800'; loads BASIC IPLNIP at location X'0000'.

INPUT: None

OUTPUT: System Q code passed to IPLNIP

EXITS:

- Normal: To X'1800' for DSM system
To X'0000' for BASIC system
- Error: Halt '1245' if a permanent disk error occurs

► **Nucleus Initialization Program (IPLNIP)—Disk System**

CHART: AA

ENTRY POINT: IPLNIP

FUNCTIONS:

- Obtains the Q code from the IPL bootstrap program.
- Searches the system directory for the disk locations of the transient routines.
- Loads the correct parity into all registers.
- Builds a table of the cylinder and sector addresses for each system transient.
- Locates and loads the requested supervisor (DPF or dedicated).
- Loads the supervisor communication regions with transient addresses.
- Reads in the configuration record.
- Determines status of disk writer switch; if on, a halt of '0' is displayed on the console stick lights.
- Writes out the dump program (DUMPPG) from X'100' to cylinder 0, sector D0 on F1.
- Branches to X'0380' to clear main storage.

INPUT:

- Q code from IPL bootstrap program
- System configuration record
- System directory

OUTPUT:

- System Q code located at NCSYSQ within the system communication region
- Table of cylinder and sector addresses of the transient routines, loaded into the supervisor communication region
- Core size is passed to the supervisor

EXITS:

- Normal: To supervisor at X'0380'
- Error: Halt of '0' is displayed on the console stick lights

► **Nucleus Initialization Program (IPLNIP)—Model 6**

CHART: AB

ENTRY POINT: IPLNIP

FUNCTIONS:

- Obtains the Q code from the IPL bootstrap program.
- Searches the system directory for the disk locations of the transient routines.
- Loads the correct parity into all registers.
- Builds a table of the cylinder and sector addresses for each system transient.
- Locates and loads the supervisor.
- Loads the supervisor with transient addresses.
- Reads in the configuration record.
- Determines status of disk writer switch; if on, a halt of '1245' is displayed on the halt display lights.
- Writes out the dump program (DUMPPG) from X'100' to cylinder 0, sector D0 on F1.
- Branches to X'0380' to clear main storage.

INPUT:

- Q code from IPL bootstrap program
- System configuration record
- System directory

OUTPUT:

- System Q code located at NCSYSQ within the system communication region
- Table of cylinder and sector addresses of the transient routines, loaded into the supervisor communication region
- Core size is passed to the supervisor

EXITS:

- Normal: To supervisor at X'0380'
- Error: Halt of '1245' is displayed on the halt display lights

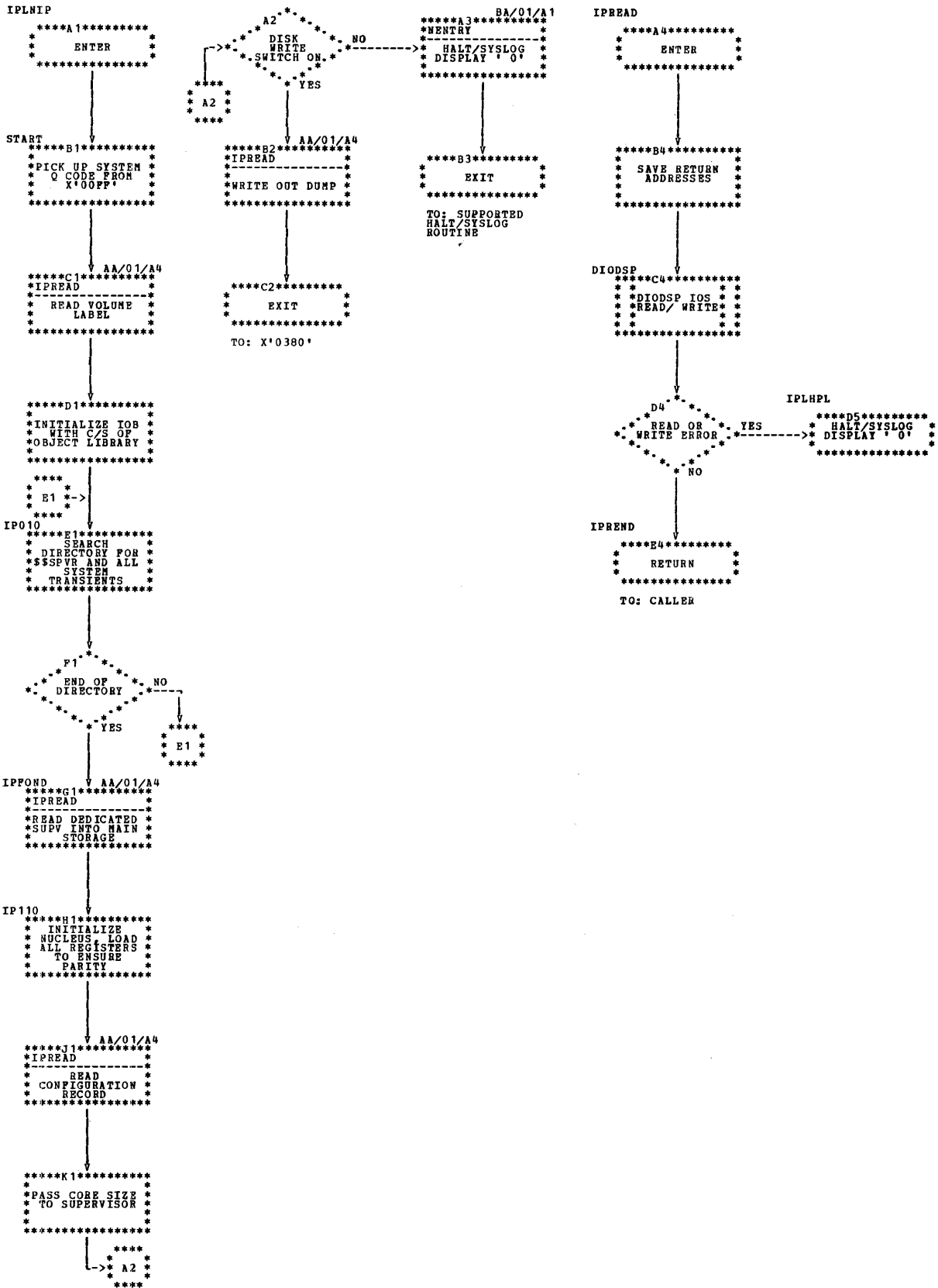


Chart AA. Nucleus Initialization Program (IPLNIP)–Disk System

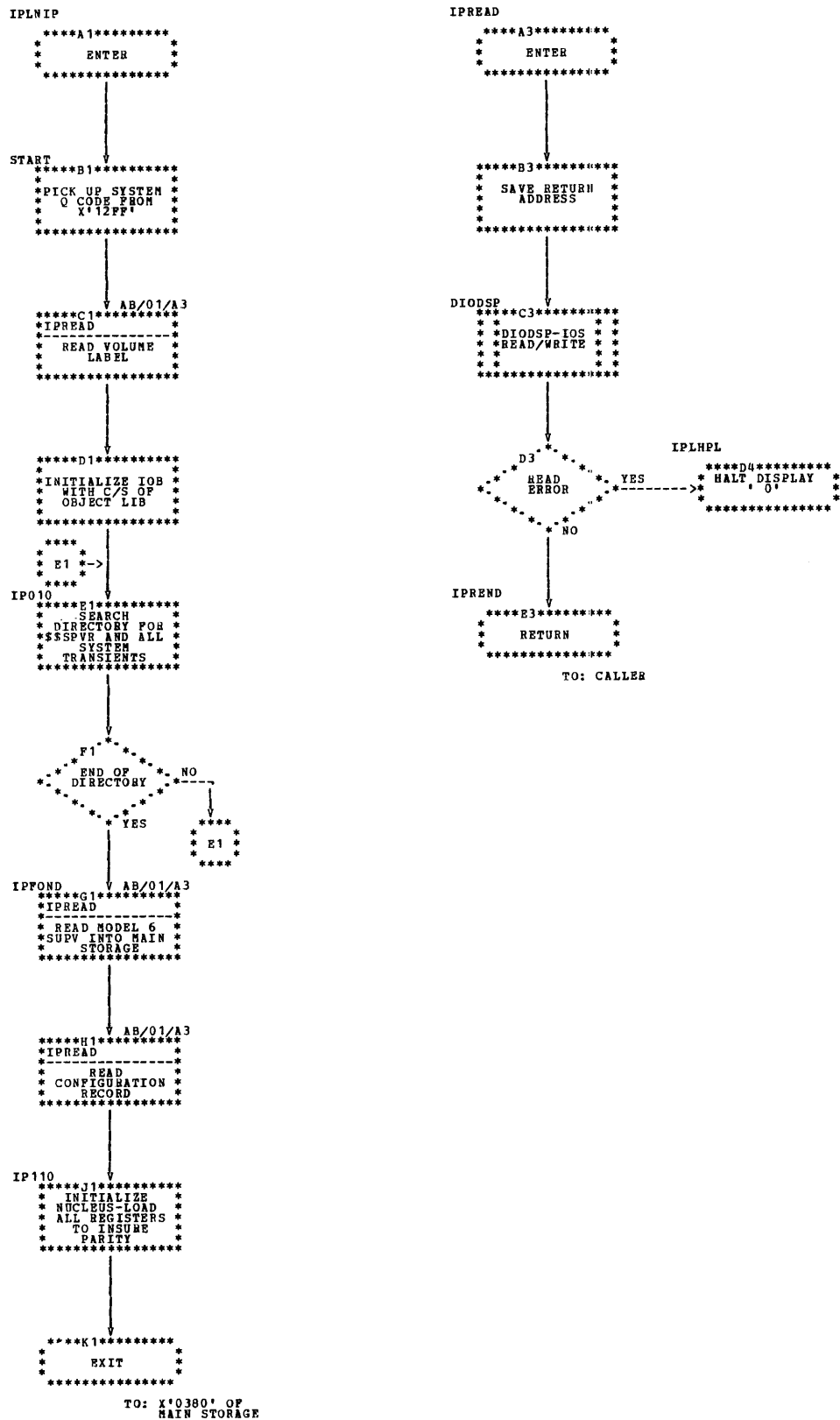


Chart AB. Nucleus Initialization Program (IPLNIP)—Model 6

Figure 1-1 contains the directory entries for IPL Bootstrap and IPLNIP.



Module Name	Chart ID	Descriptive Name	Entry Point	Function
IPLBOT	none	Initial Program Load Bootstrap Program	IPLSRT	Reads in IPLNIP
IPLNIP	AA for Disk system AB for Model 6	Nucleus Initialization Program	IPLNIP	<ul style="list-style-type: none">● Loads supervisor● Determines transient addresses● Loads registers with correct parity

Figure 1-1. IPLBOOT and IPLNIP Directory

PART 2.

DISK DEDICATED SUPERVISOR

The Dedicated Supervisor provides six functions:

- Locates, loads and passes control to modules
- Contains and coordinates the use of the transient area
- Provides system and program level communication regions
- Contains resident dump linkages
- Provides a general entry/exit routine
- Contains Disk IOS linkages. (This is discussed in the *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0502.*)

SYSTEM REQUIREMENTS

The system requirements for the Dedicated Supervisor are:

- One 5203 Line Printer
- One 5410 Processing Unit (Model A 13)
- One 5410 MFCU
- One 5444 Disk Storage Drive (Model 2)

After the Dedicated Supervisor has been loaded into main storage, it will accept any of four types of user requests.

- Request for general entry
- Request for dump program
- Request for IOS program
- Request for IOS Wait program

These requests go through a LOAD IAR at the following fixed locations:

X'00' — Dump linkage
X'04' — General entry
X'08' — IOS
X'0C' — IOS Wait

Most user requests will go to general entry (NENTRY), X'04' (Figure 2-1). General Entry saves the system linkages and register 1 along with the caller's return address. The user's request is then analyzed. The request can be one of seven types:

- Load with find
- Load only
- Fetch with find
- Fetch only
- Transient request
- Fetch to address
- Fetch to address with find

LOAD OR FETCH WITH FIND

Upon receiving this request, a request indicator byte (RIB) is set for a find condition. The requested module is located and control is passed to the Loader and Relocation program (NLOADR). NLOADR moves the specified module into main storage and, if required, adds a relocation factor to the module's address. Control is then passed to one of two locations:

1. If the module was moved using a fetch request, control is passed to the General Exit routine. The General Exit routine (1) restores register 1, (2) restores system linkages, and (3) passes control to the routine that was fetched into main storage.
2. If the module was moved using a load request, control is passed to the General Exit routine located within NENTRY. The General Exit routine restores register 1, restores system linkages, and returns control to the calling routine.

LOAD OR FETCH ONLY

If a load or fetch only request is received, control is passed directly to the Loader and Relocation program (NLOADR). NLOADR moves the requested module into main storage and adds a relocation factor to the module's addresses if required. Control is then passed to one of two locations:

1. If the module was moved using a fetch only instruction, control is passed via the General Exit routine to the routine that was fetched into main storage.
2. If the request was a load only, control is passed to caller.

TRANSIENT REQUESTS

Upon receiving a request for a transient, the transient area is indicated as busy. The requested transient address (cylinder/sector number) is placed in the active transient ID, and the requested transient is then moved into the transient area (located in main storage at location X'100'). Control is then passed to the loaded transient. Any active transient can issue three types of requests depending upon the stage of completion:

1. **Embedded transient request:** The transient is not through processing and would like to bring another transient into main storage. Control is passed to the Transient Area Scheduler (NENTRY) where the address of the calling transient is placed on a transient queue (NXQUE). The requested transient is then loaded into main storage; and control is passed to the transient, then back to the Transient Area Scheduler (see arrows 1 and 3 of Figure 2-1).
2. **Load next transient:** The transient is through processing and would like to pass control to another transient. Control is passed back to the Transient Area Scheduler (NENTRY) and the requested transient is loaded into main storage at X'100'. Control is then passed to it.
3. **Exit from transient:** The transient has completed its processing and would like to return to its caller. Control is passed to the Transient Area Scheduler located within NENTRY. If the transient queue is empty, control is passed to the General Exit routine, located within NENTRY; otherwise control is passed to the transient located in level number one, of the queue.

The following example of the transient queue illustrates the four queue levels. This queue is located at the label NXQUE.

	1	1	1	2	2	Total of 7 bytes per level
Level Number 4 on Q	C	S	#	XR1		Next sequential instruction
Level Number 3 on Q	C	S	#	XR1		Next sequential instruction
Level Number 2 on Q	C	S	#	XR1		Next sequential instruction
Level Number 1 on Q	C	S	#	XR1		Next sequential instruction

The transients are loaded into and removed from the transient queue via level one. See *Appendix B. Diagnostic Aids* for the procedure for tracing the transient queue.

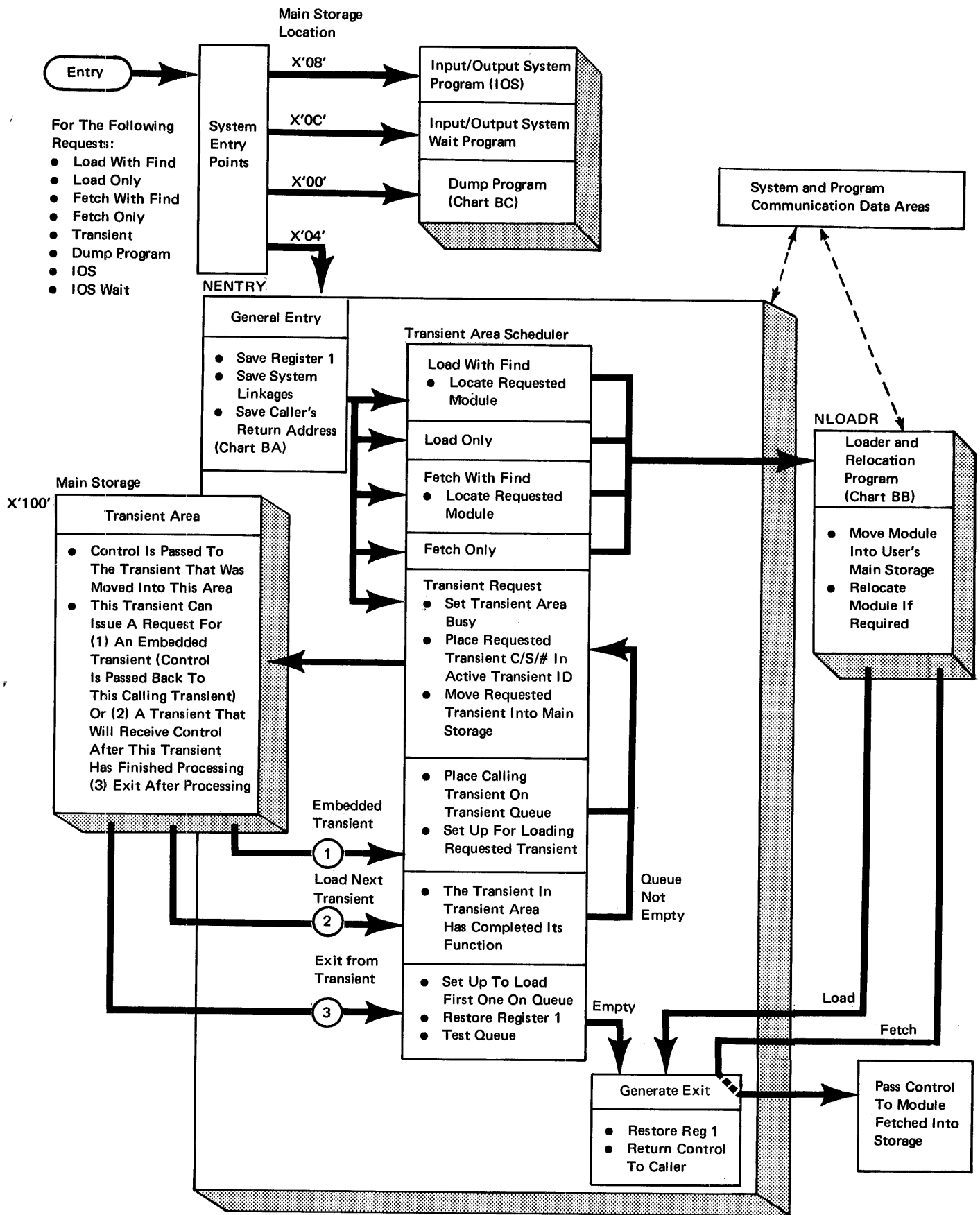


Figure 2-1. Functional Flow of the Dedicated Supervisor

The following section describes in detail the organization of the Dedicated Supervisor routines shown in Figure 2-1. Included in the following descriptions are:

- Explanation of routine containing: name of routine, input, output, function, exits from this routine.
- Main storage map showing the routines that might be in main storage at the same time as the routine being described.
- Flowchart showing the functional flow of the routine being described.

► **General Entry/Exit and Transient Area Scheduler (NENTRY)**

CHART: BA
 FIGURE: 2-2
 ENTRY POINT: NENTRY
 FUNCTION:

- General Entry functions are to save:
 1. System registers
 2. System linkages
 3. Caller's return address
- Transient Area Scheduler performs the following:
 1. Analyzes requests for transients.
 2. Locates the corresponding transient routine.
 3. Retrieves the specified routine from disk storage.
 4. Passes command as requested.
 5. Tests to find if the Loader and Relocation program (NLOADR) was requested:
 - a. If requested, control is passed to NLOADR.
 - b. If not requested, control is passed to the General Exit.
- General Exit restores system registers and linkages and passes control to the calling routine.

EXIT:

- Normal: Calling program or passed to the program called.
- Error: A halt 'Ø1' is issued if there are too many (five or more) transients on queue.

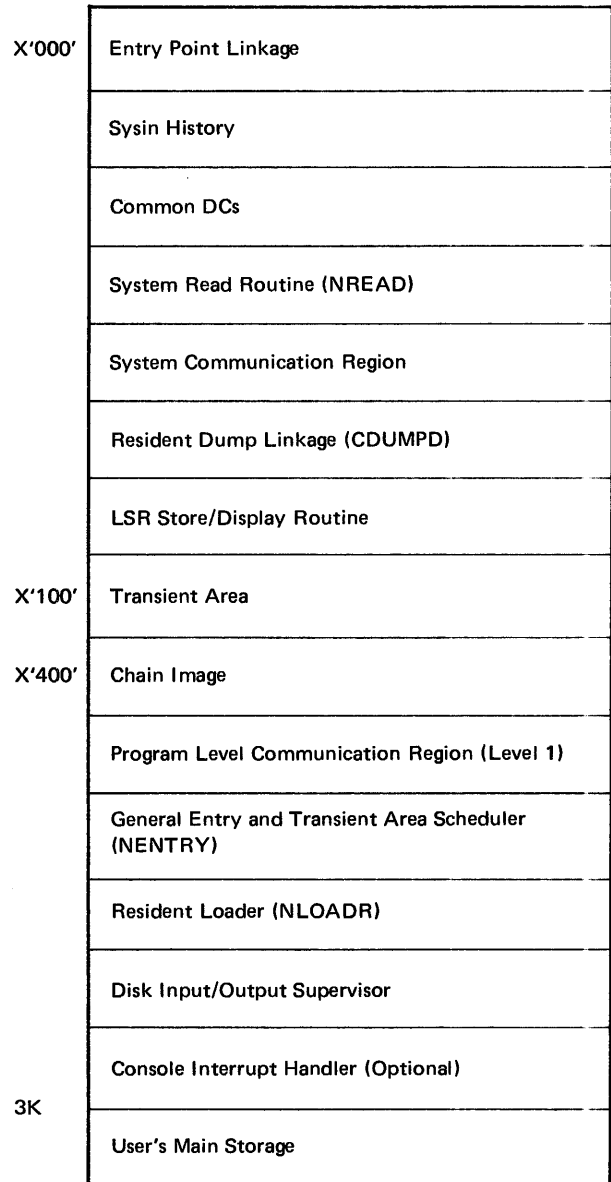


Figure 2-2. Main Storage Map for General Entry/Exit and Transient Area Scheduler (NENTRY) for Dedicated Supervisor

► Resident Loader and Relocation Program (NLOADR)

CHART: BB

FIGURE: 2-2

ENTRY POINT: NLOADR

FUNCTION:

- The Resident Loader loads the requested routines by using a fetch or a load RIB request.
- The Relocation program adds a relocation factor to the loaded program and passes control to the General Exit routine

INPUT: Register 1 contains the address of the program level communication region.

OUTPUT: The requested module is loaded as specified by the caller's request indicator byte (RIB). See *Appendix B. Diagnostic Aids* for RIB procedure.

EXIT:

- Normal:
 1. After issuing a load request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the caller.
 2. After issuing a fetch request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the program that was fetched into main storage.
- Error: A halt of 'b1' is issued if the requested load is outside of the partition boundaries.

► Local Storage Register Display Routine (LSR)

CHART: None

FIGURE: 2-2

ENTRY POINT: HLT1, a manual branch via the CE console switches (see the LSR display procedures in *Appendix B. Diagnostic Aids*).

FUNCTION: The Local Storage Register Display routine displays registers without destroying the contents of the displayed register by issuing two halts:

1. The Q-byte of the first halt is the high-order byte of the register being displayed.
2. The Q-byte of the second halt is the low-order byte of the register being displayed.

INPUT: Console Address/Data switches

OUTPUT: The console stick lights indicate the contents of the register being displayed.

EXIT: None

► Resident Dump Linkage (CDUMPD)

CHART: BC

FIGURE: 2-2

ENTRY POINT: CDUMPD

FUNCTION:

- Seeks to cylinder zero on disk pack R1
- Writes the three sectors, which start at X'0100', on R1, cylinder 0, sectors B0-B8.
- Loads the Dump Selection program (DMPFND) from cylinder zero, sector D0 of the system disk pack into the transient area. (See *Appendix B. Diagnostic Aids*.)
- Branches to the Dump Selection program (DMPFND) at location X'0100'.

INPUT: None

OUTPUT: None

EXIT: Dump Selection program (DMPFND) at location X'0100'.

► System Disk Read Routine (NREAD)

CHART: None

FIGURE: 2-2

ENTRY POINT: NREAD

FUNCTION:

- Reads located modules from disk into main storage.
- Checks for successful read from disk storage.

INPUT: System IOB

OUTPUT: None

EXIT:

- Normal: Caller
- Error: A halt of 'b1' is issued.

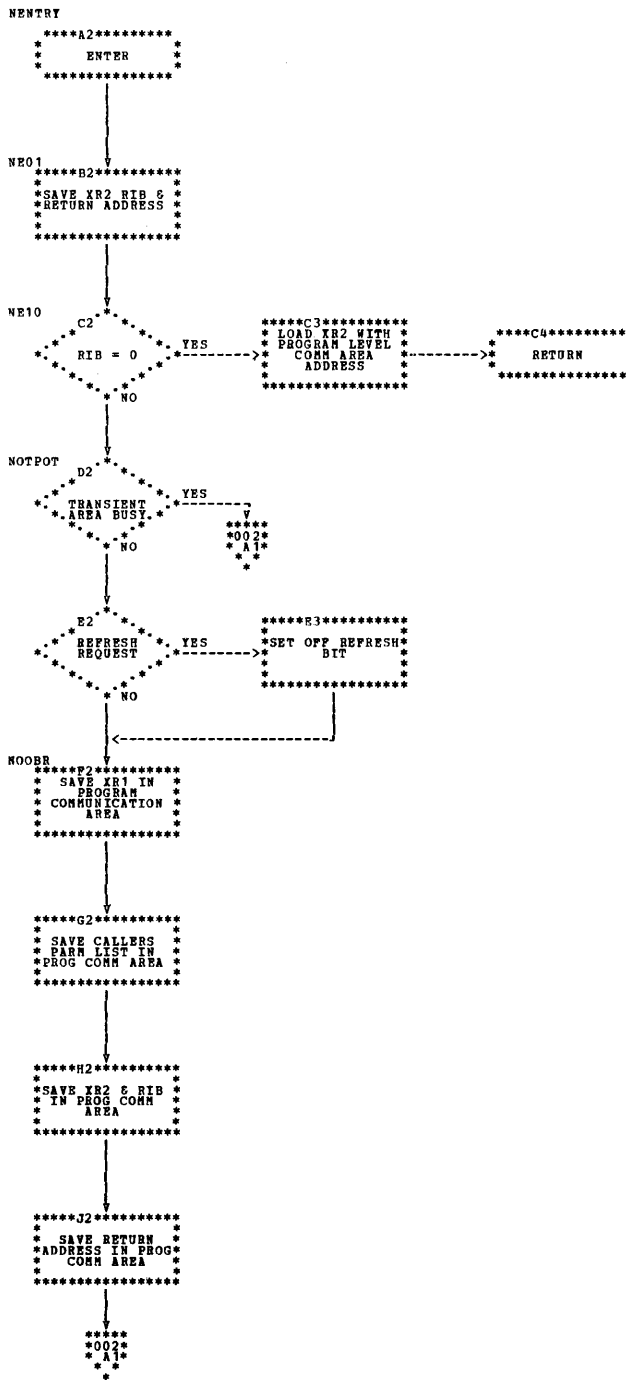


Chart BA (Part 1 of 2). Dedicated Supervisor General Entry/Exit and Transient Area Scheduler (NENTRY)

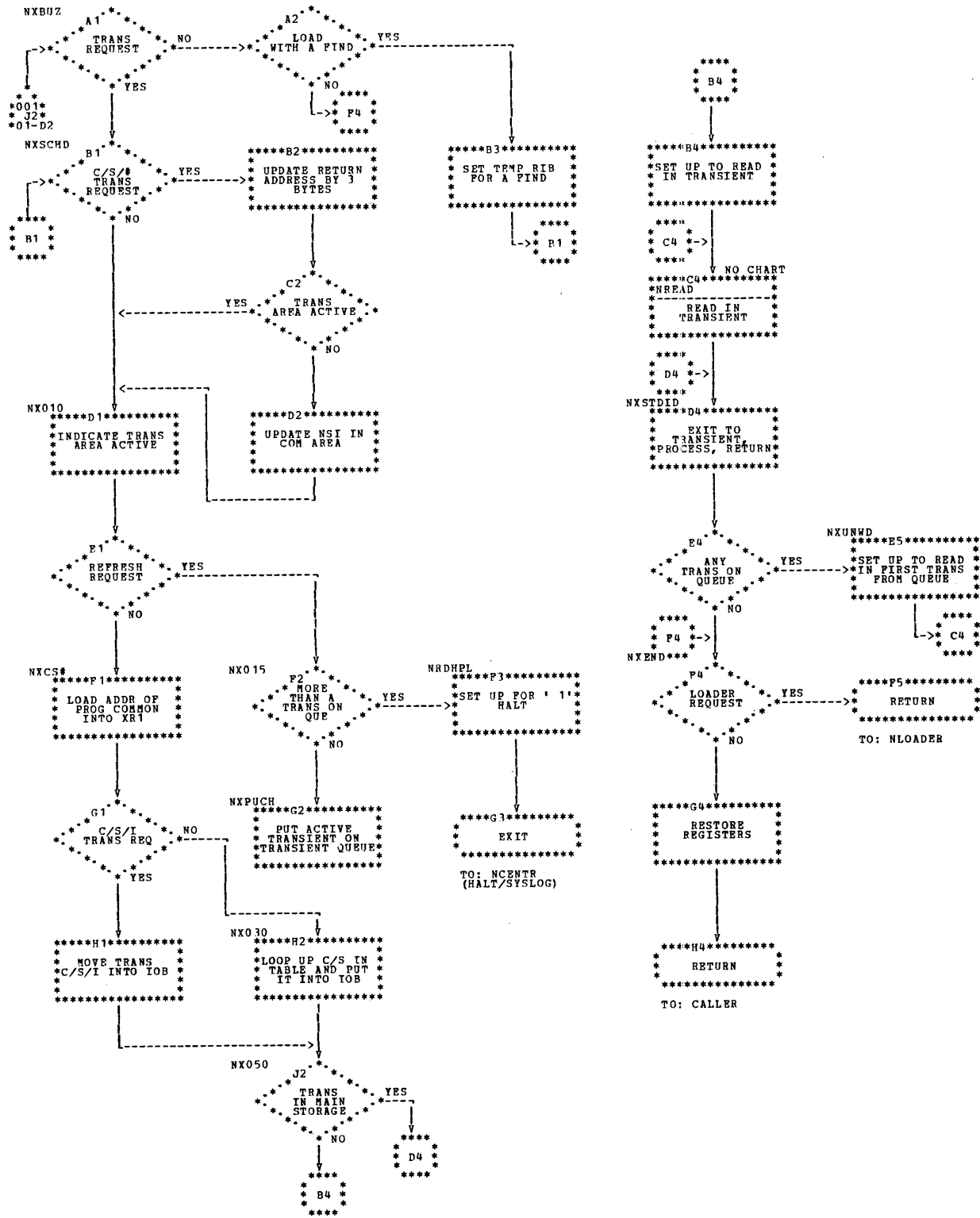


Chart BA (Part 2 of 2). Dedicated Supervisor General Entry/Exit and Transient Area Scheduler (NENTRY)

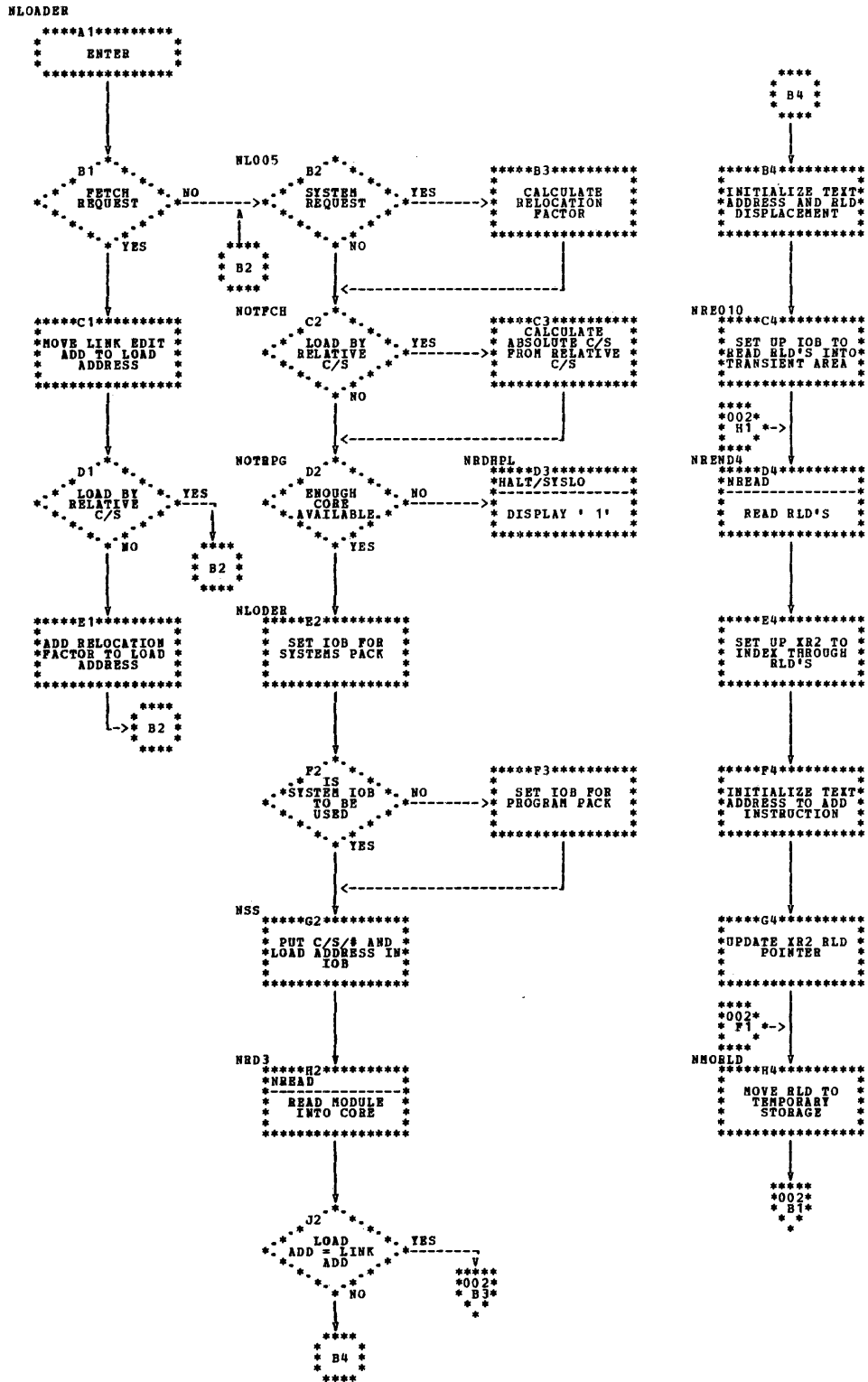


Chart BB (Part 1 of 2). Resident Loader and Relocation Program (NLOADR)

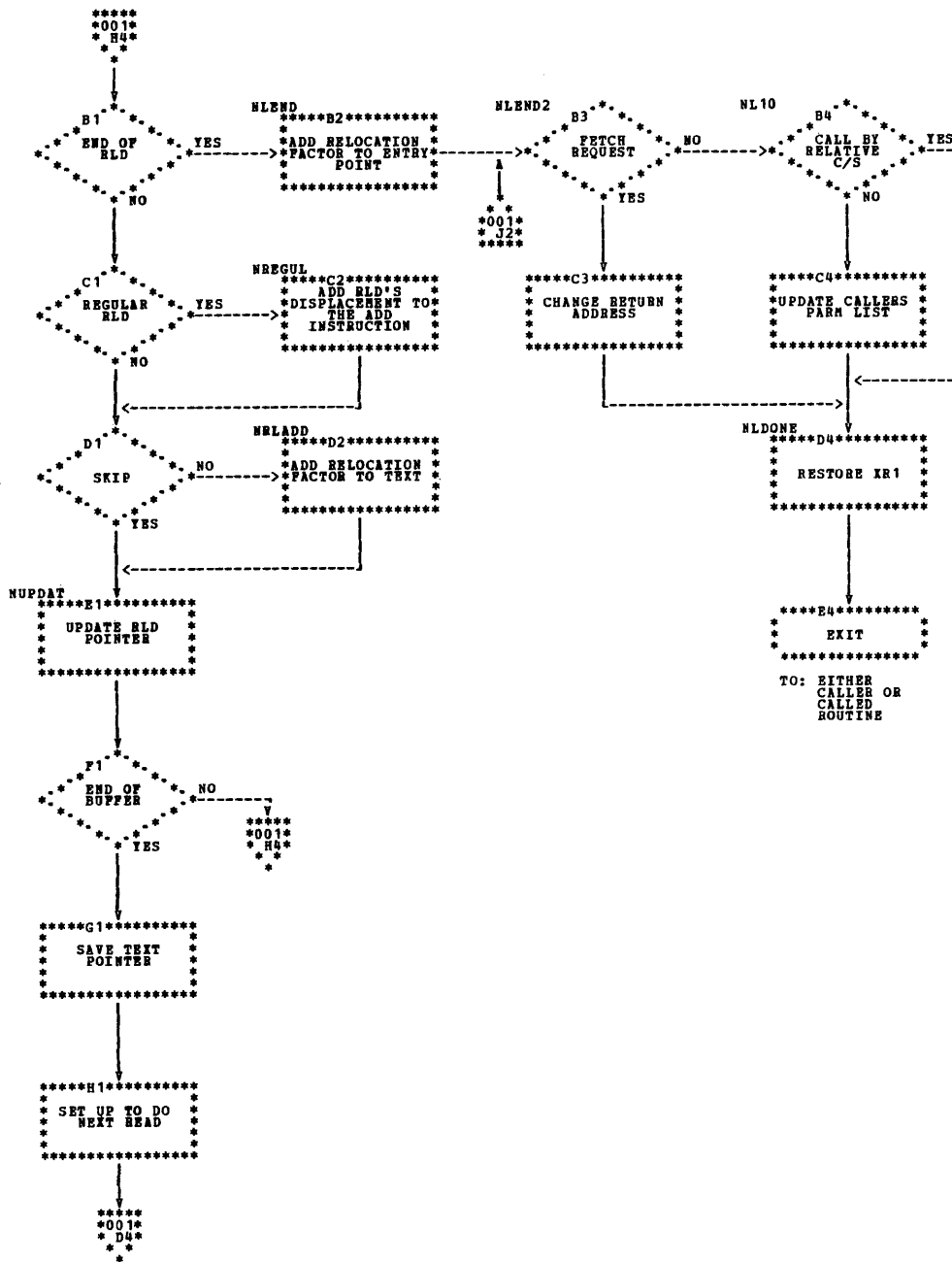


Chart BB (Part 2 of 2). Resident Loader and Relocation Program (NLOADR)

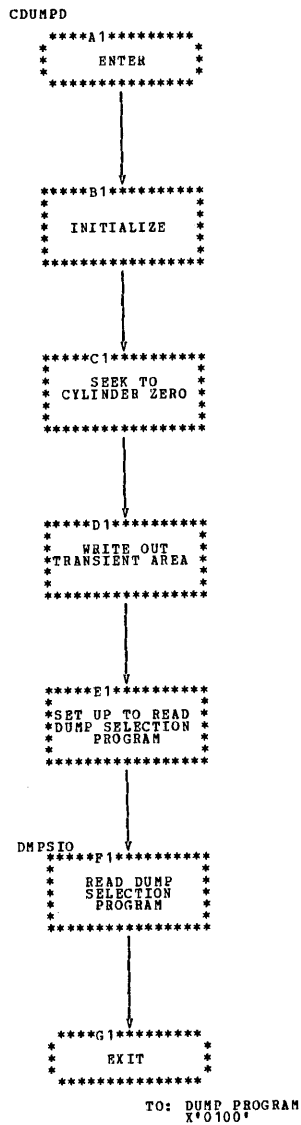


Chart BC. Supervisor Resident Dump Linkage Routine (CDUMPD)

Figure 2-3 contains the directory entries for the Dedicated Supervisor.

Module/Phase Name	Chart ID	Descriptive Name	Entry Point	Function
\$@SPV1	BA	General Entry/Exit and Transient Area Scheduler	NENTRY	<ul style="list-style-type: none"> ● Saves all system registers and linkages ● Analyzes user and system requests ● Locates corresponding transient routines ● Passes control to transient ● Passes control to Loader or General Exit ● Restores system registers and linkages
	BB	Resident Loader and Relocation Program	NLOADR	<ul style="list-style-type: none"> ● Loads the requested routine ● Passes control to General Exit routine, which in turn passes control to the caller or . . . ● Passes control to the routine that was fetched via General Exit routine ● Passes control to General Exit routine upon completion
	none	Local Storage Register Display Routine	HLT1	<ul style="list-style-type: none"> ● Displays registers without destroying the contents
	none	System Read Routine for Disk	NREAD	<ul style="list-style-type: none"> ● Reads located modules from disk into main storage and checks for successful read
	BC	Resident Dump Linkage	CDUMPD	<ul style="list-style-type: none"> ● Seeks to cylinder zero on R1 ● Writes out the transient area ● Loads in the Dump Selection Program from cylinder zero of the system pack ● Branches to the Dump Selection Program at location X'0100'

Figure 2-3. Dedicated Supervisor Directory

PART 3.

DISK DPF SUPERVISOR

This section identifies the differences between the Dedicated Supervisor and the DPF Supervisor. The Dedicated Supervisor is discussed in Part 2 of this document.

The basic difference between the Dedicated and DPF Systems is that the DPF System supports two user programs in main storage at one time, and the Dedicated System supports one. The two systems are identical with the following exceptions:

1. The DPF Supervisor uses two identical program level communication regions, the Dedicated Supervisor uses one.
2. The DPF Supervisor contains a Resident Interrupt Handler (INTLO) and a Resident Halt Routine (HPLNUC); the Dedicated Supervisor does not contain either.
3. The DPF General Entry and Transient Area Scheduler (NENTRY) routine must determine which program level (one or two) is requesting a transient or the Loader routine (NLOADR).

The shaded areas on Figure 3-1 indicate the routines that are unique to, or modified for, the DPF Supervisor.

SYSTEM REQUIREMENTS

The system requirements for the DPF Supervisor are:

- One 5203 Line Printer
- One 5410 MFCU
- One 5410 Processing Unit (Model A13)
- One 5444 Disk Storage Drive (Model 2)

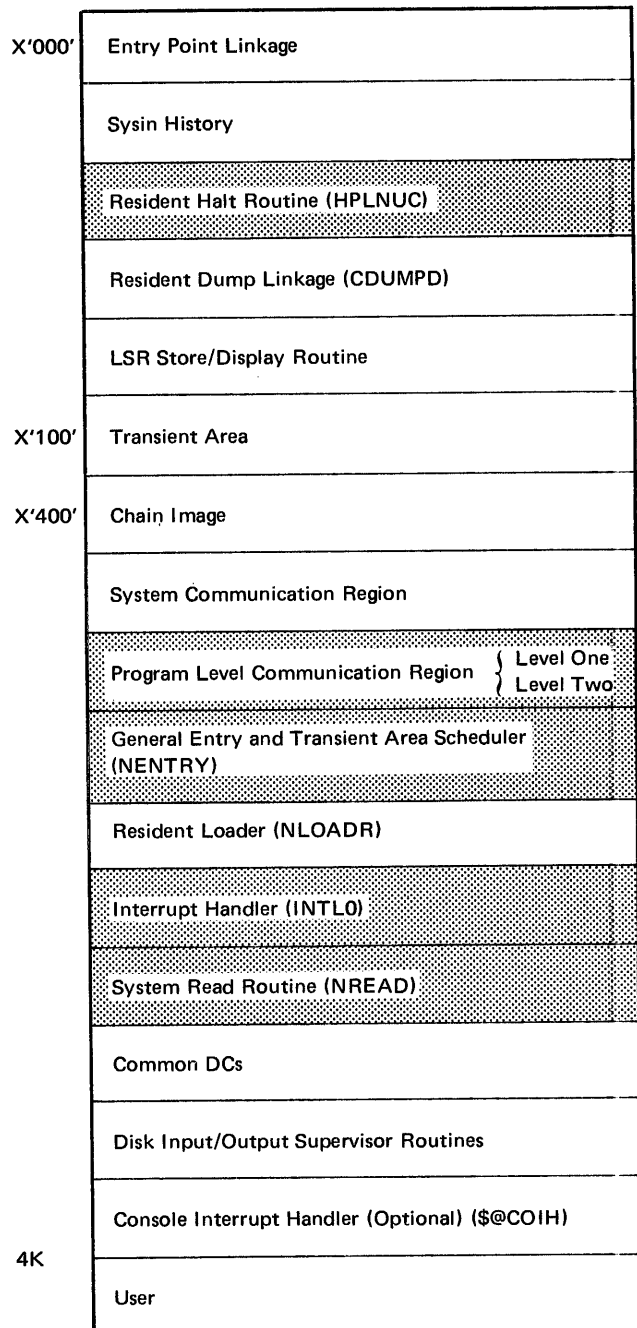


Figure 3-1. Main Storage Map for DPF Supervisor

Section 2. Method of Operation

The Method of Operation for the DPF System is identical to the Dedicated System with the exception that the General Entry/Exit and Transient Area Scheduler (NENTRY) in the DPF Supervisor determines which level is evoking the request. See Figure 2-1 in *Part 2. Dedicated Supervisor*.

This section describes only those routines unique to the DPF Supervisor. For the other DPF Supervisor routines, see the routine descriptions in *Section 3. Program Organization of Part 2. Dedicated Supervisor.*

► **General Entry/Exit and Transient Area Scheduler (NENTRY)**

CHART: CA

FIGURE: 3-1

ENTRY POINT: NENTRY

FUNCTION:

- Determines the correct program level of the caller.
- Determines the caller's request and takes appropriate action to handle the request.
- Passes control back to transients that are queued after they call another transient.

INPUT: The request indicator byte (RIB) and either:

1. A 3-byte C/S/# address of a desired transient; or
2. A parameter list (seven or ten bytes in length) for the Loader. See *Appendix B. Diagnostic Aids* for RIB byte.

OUTPUT: On a load request, the parameter list (*Appendix B. Diagnostic Aids*) is updated to reflect the directory entry for the module loaded.

EXIT:

- Normal: Caller, if transient request.
Loader (NLOADR), if loader request.
- Error: A halt of 'ψ1' is displayed on the console stick lights.

► **Resident Loader and Relocation Program (NLOADR)**

CHART: CB

FIGURE: 3-1

ENTRY POINT: NLOADR

FUNCTION:

- The resident Loader loads the requested routines by using a fetch or a load RIB request.
- The Relocation program adds a relocation factor to the loaded program and passes control to the General Exit routine.
- Determines level being used.

INPUT: Register 1 contains the address of the program level communication region.

OUTPUT: The requested module is loaded as specified by the caller's request indicator byte (RIB). See *Appendix B. Diagnostic Aids* for RIB procedure.

EXIT:

- Normal:
 1. After issuing a load request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the caller.
 2. After issuing a fetch request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the program that was fetched into main storage.
- Error: A halt of 'ψ1' is issued if the requested load is outside of the partition boundaries.

► Resident Halt Display Routine (HPLNUC)

CHART: CC

FIGURE: 3-1

ENTRY POINT: HPLNUC

FUNCTIONS:

- Saves the transient area queue.
- Displays the halt code passed to it by the caller.
- Checks for a valid response to displayed code.
- Passes control to the End-of-Job transient (\$\$SPEJ) if an immediate cancel was requested and is a valid response.

INPUT:

- The halt code to be issued. (See the *Halt/Syslog Procedure* in *Appendix B. Diagnostic Aids.*)
- The transient area queue.

OUTPUT:

- The halt code is displayed.
- The user's reply is returned to the caller.

EXIT:

- Normal:
 1. Syslog Halt Processor (\$\$STOH)
 2. End-of-Job transient (\$\$SPEJ) if an immediate cancel was requested.
- Error: A halt '00' is displayed on the console stick lights if the response is invalid.

► Terminator Interrupt Handler (INTLO)

CHART: CD

FIGURE: 3-1

ENTRY POINT: INTLO

FUNCTION:

Note: The interrupt handler is present in a DPF system only.

- Senses the Reader/Select switch to determine sysin device or the level being cancelled.
- Stores the sensed information in the current program level communication region.
- Displays the JU halt to allow the user to select whether the job is to be cancelled or processing is to continue after the job has been interrupted.

INPUT: None

OUTPUT: The sensed information from the Reader/Select switch is stored in the current level communication region.

EXIT:

- Normal: Terminator Standby Routine (\$\$TMSB) when a device has been selected
- Error: End of job transient if the job is cancelled (\$\$SPEJ)

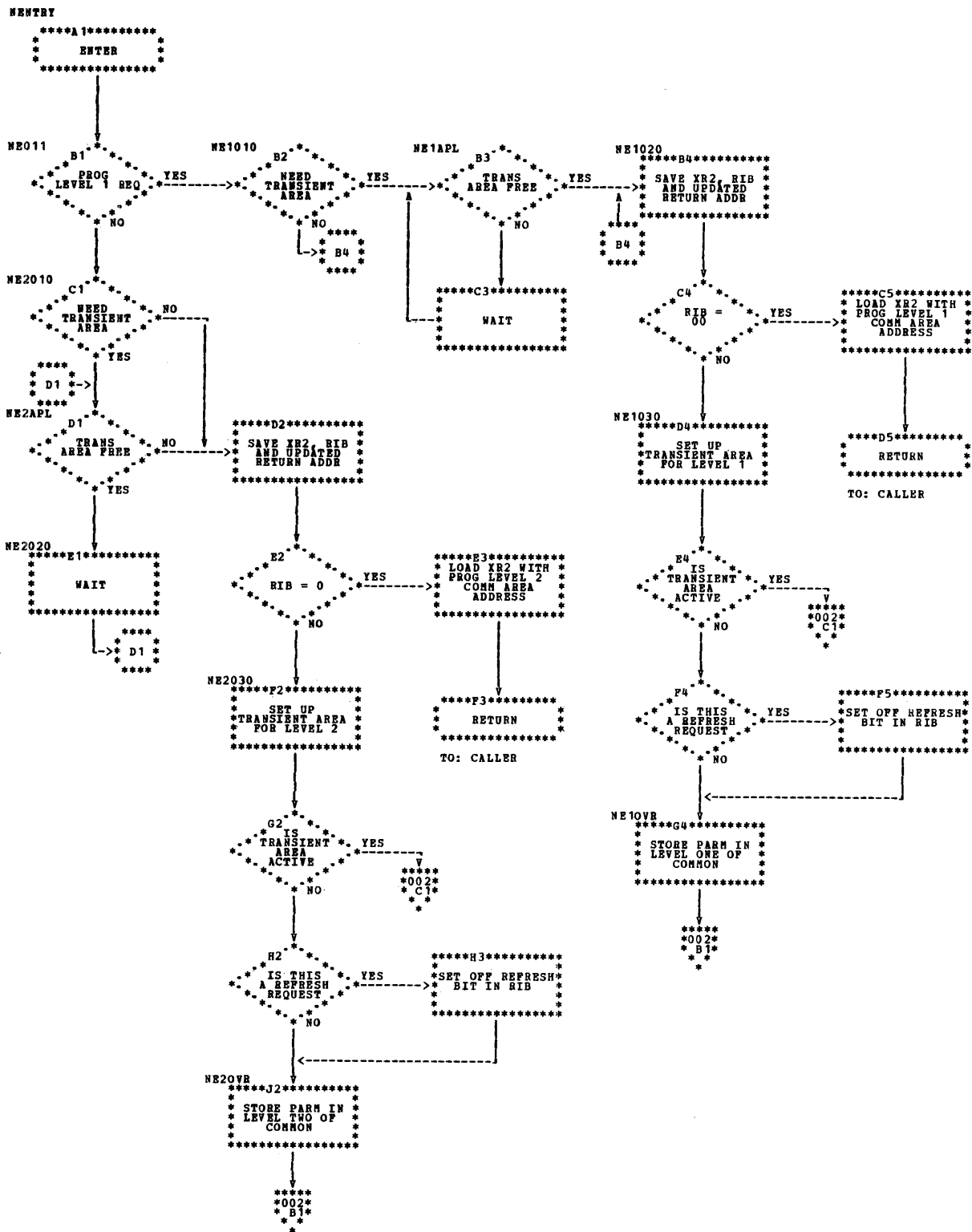


Chart CA (Part 1 of 2). General Entry/Exit and Transient Area Scheduler (NENTRY)

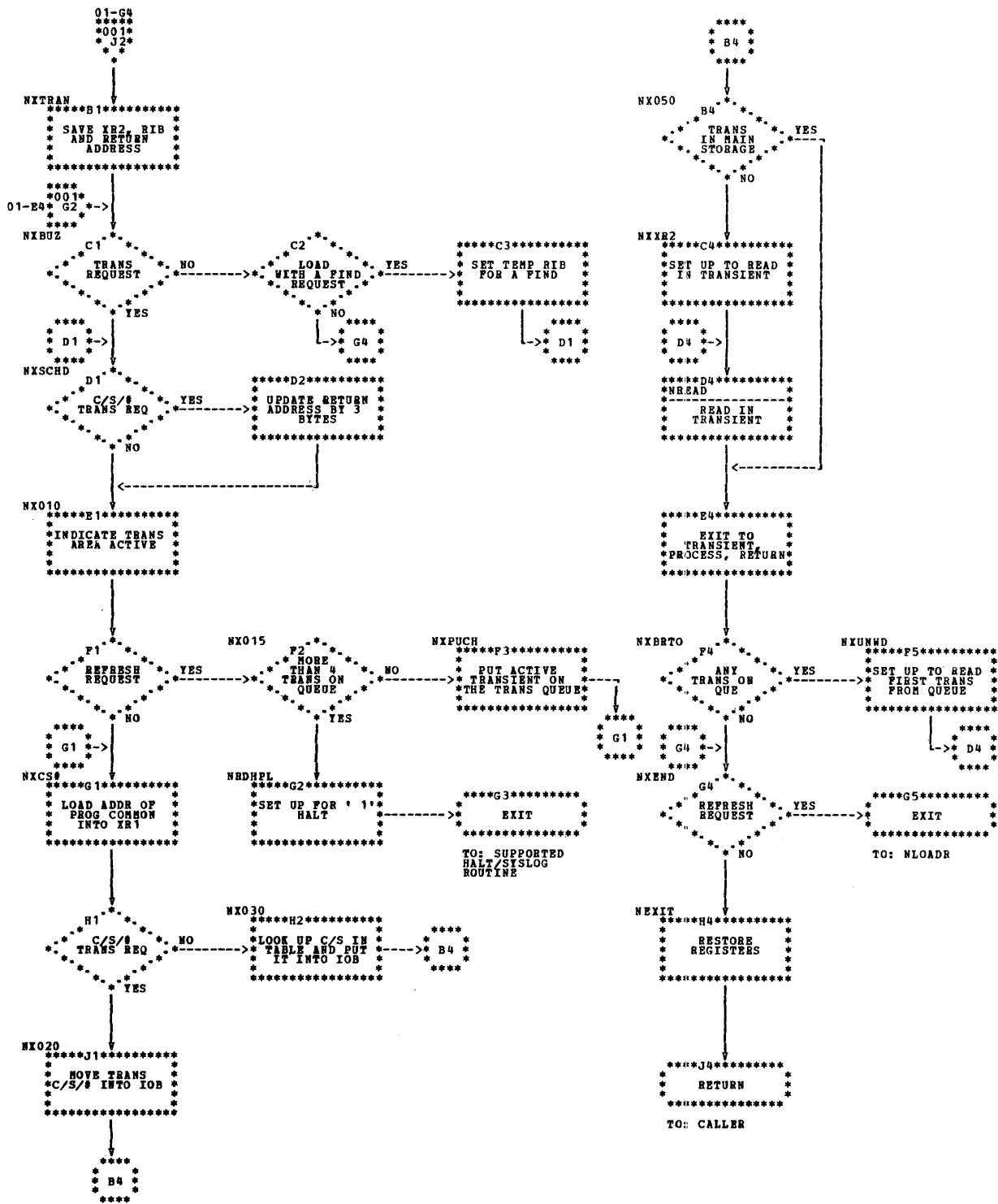


Chart CA (Part 2 of 2). General Entry/Exit and Transient Area Scheduler (NENTRY)

NLOADR

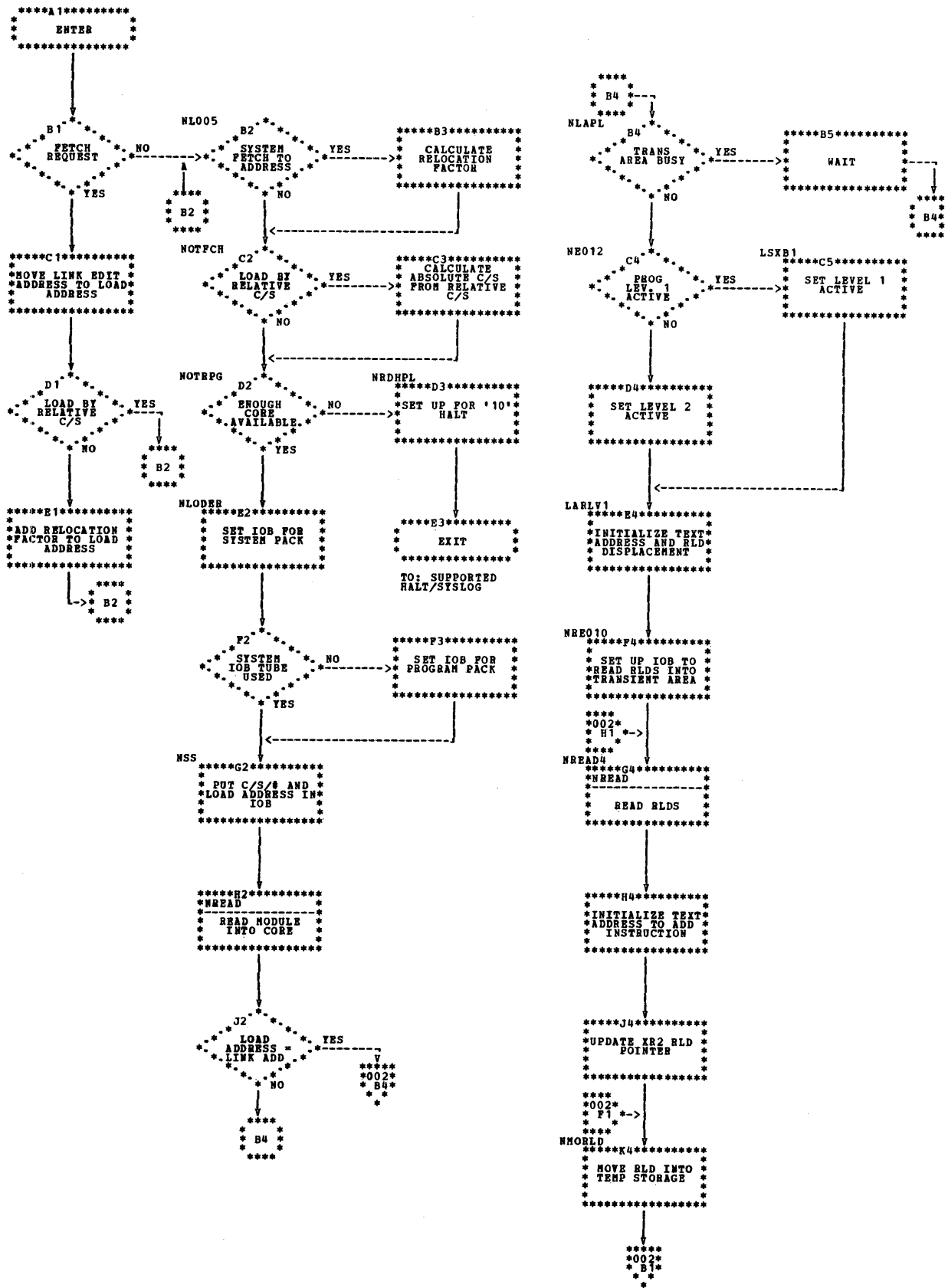


Chart CB (Part 1 of 2). Resident Loader and Relocation Program (NLOADR)

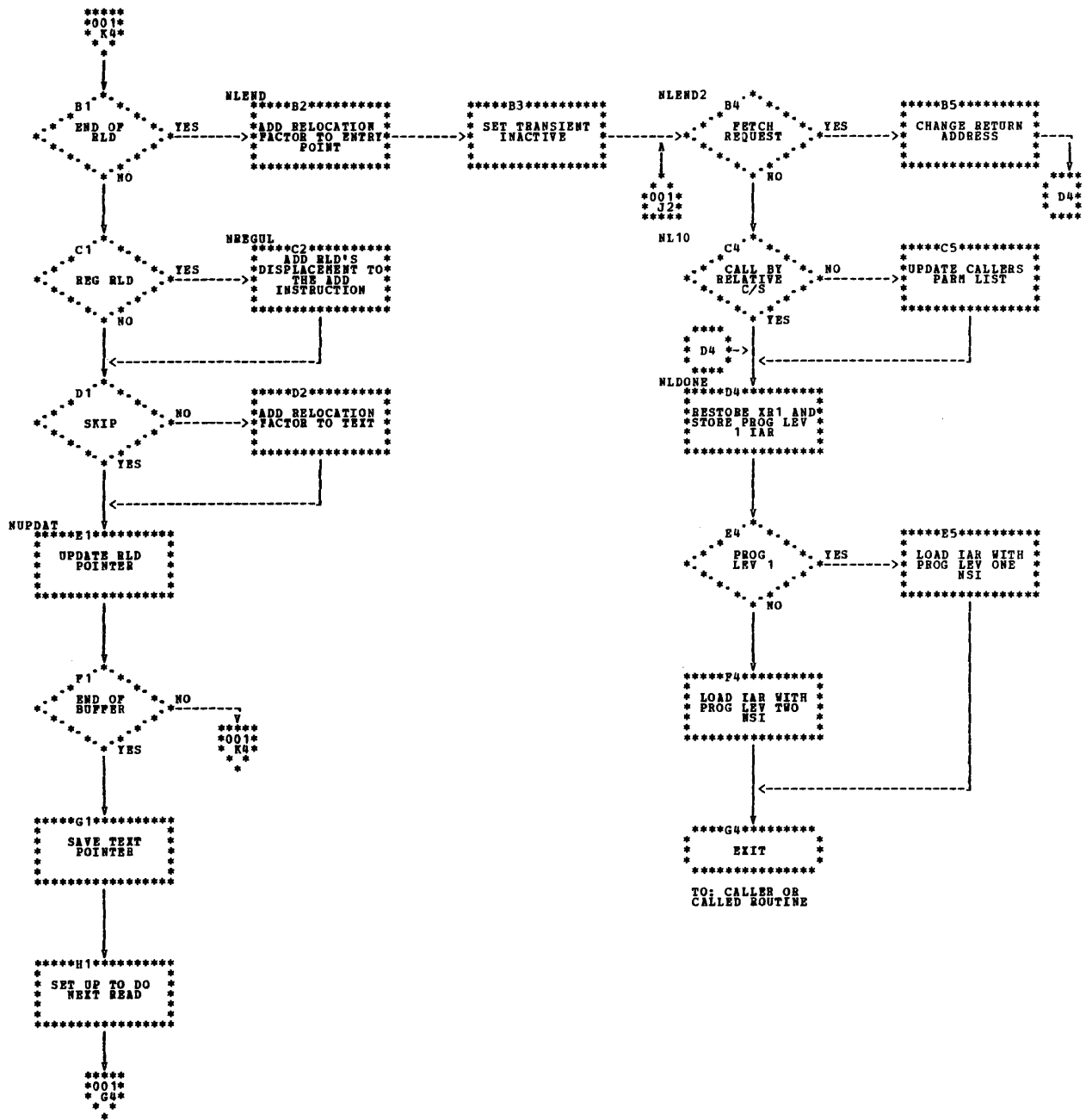


Chart CB (Part 2 of 2). Resident Loader and Relocation Program (NLOADR)

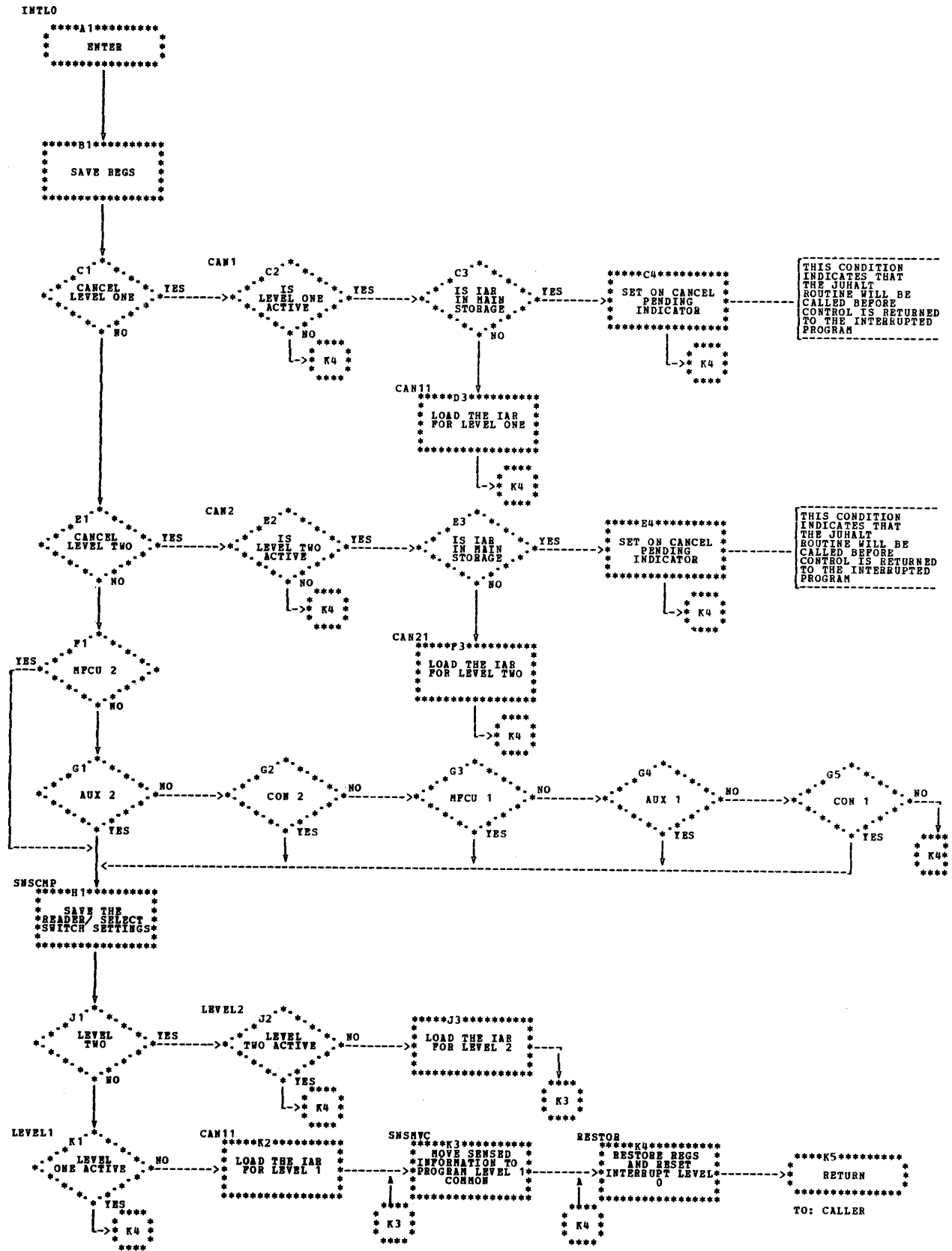


Chart CD (Part 1 of 2). Terminator Interrupt Handler

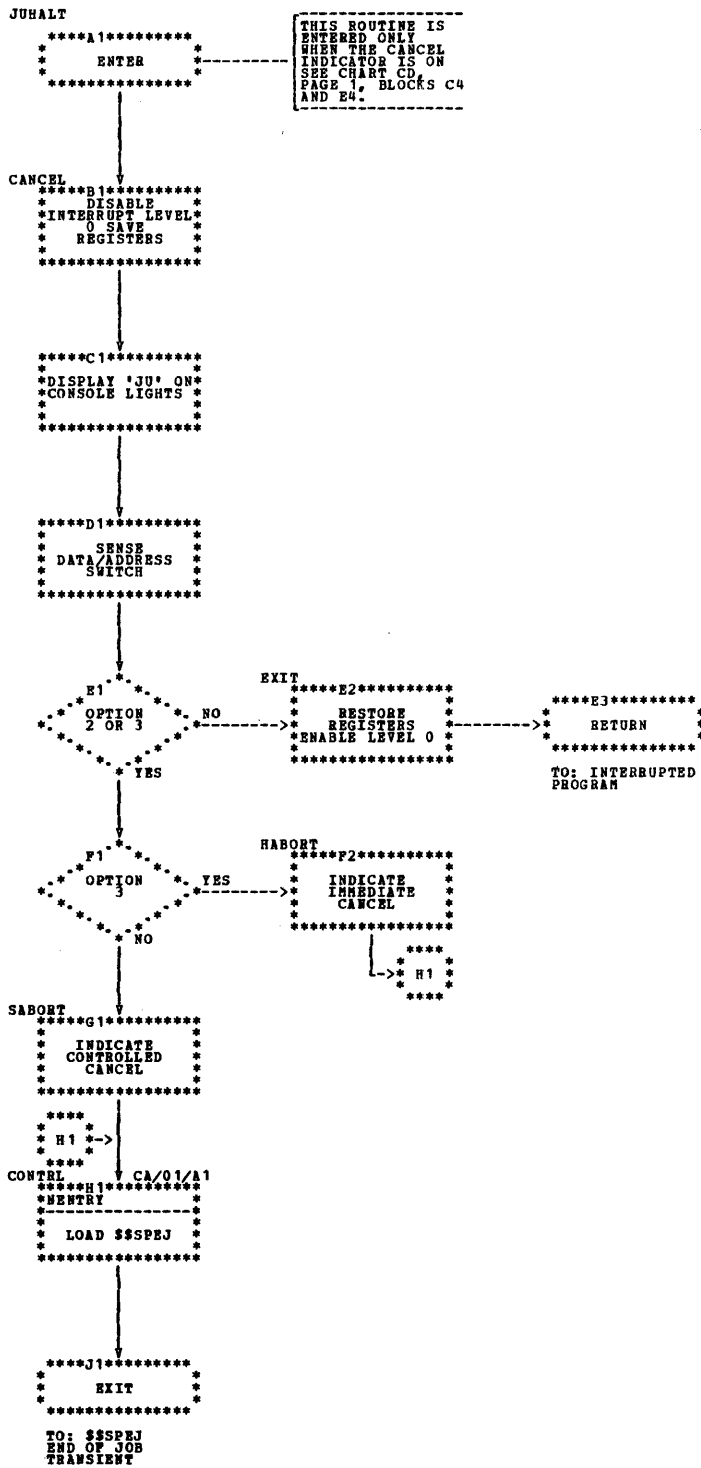


Chart CD (Part 2 of 2). Terminator Interrupt Handler (INTL0)

Figure 3-2 contains the directory entries for the DPF Supervisor.

Module Name	Chart ID	Descriptive Name	Entry Point	Function
\$@SPV2	CA	General Entry and Transient Area Scheduler	NENTRY	<ul style="list-style-type: none"> ● Determines program level of caller ● Determines the caller's request ● Passes control to the appropriate handler routine
	CB	Resident Loader and Relocation Program	NLOADR	<ul style="list-style-type: none"> ● Loads the requested routine ● Passes control to General Exit routine, which in turn passes control to the caller; or, ● Passes control to the routine that was fetched via General Exit routine ● Passes control to General Exit routine upon completion
	none	Local Storage Register Display Routine	HLT1	<ul style="list-style-type: none"> ● Displays registers without destroying the contents
	none	System Read Routine for Disk	NREAD	<ul style="list-style-type: none"> ● Reads located modules from disk into main storage and checks for successful read
	BC	Resident Dump Linkage	CDUMPD	<ul style="list-style-type: none"> ● Seeks to cylinder zero on R1 ● Writes out the transient area ● Loads in the Dump Selection Program from cylinder zero of the system pack ● Branches to the Dump Selection Program at location X'27C'
	CC	Resident Halt Display Routine	HPLNUC	<ul style="list-style-type: none"> ● Saves the transient area queue ● Displays the halt code ● Checks for valid reply from operator ● Passes control to the End of Job Transient (\$\$SPEJ) if an immediate cancel was requested ● Restores transient queue and returns control to caller if immediate cancel was not selected
	CD	Terminator Interrupt Handler	INTLO	<ul style="list-style-type: none"> ● Determines SYSIN device or level being cancelled ● Stores this information in the program level communication region ● Gives control to JU halt display routine if a job is being cancelled
	CD	JU Halt	JUHALT	<ul style="list-style-type: none"> ● Displays JU halt and determine option selected

Figure 3-2. DPF Supervisor Directory

PART 4.

MODEL 6 SUPERVISOR

The Model 6 Supervisor provides six functions:

- Locates, loads and passes control to modules
- Contains and coordinates the use of the transient area
- Provides system and program level communication regions
- Contains resident dump linkages
- Provides a general entry/exit routine
- Contains Disk IOS linkages. (This is discussed in the *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0512.*)

SYSTEM REQUIREMENTS

The system requirements for the Model 6 Supervisor are:

- One 5213 Matrix Printer
- One 5406 Processing Unit with Keyboard
- One 5444 Disk Storage Drive (Model 2)

After the Model 6 Supervisor has been loaded into main storage, it will accept any of four types of user requests.

- Request for General Entry
- Request for dump program
- Request for IOS program
- Request for IOS Wait program

These requests go through a LOAD IAR at the following fixed locations:

X'00' — Dump linkage
X'04' — General Entry
X'08' — IOS
X'0C' — IOS Wait

Most user requests will go to general entry (NENTRY), X'04' (Part 2, Figure 2-1). General Entry saves the system linkages and register 1 along with the caller's return address. The user's request is then analyzed. The request can be one of eight types:

- Load with find
- Load only
- Fetch with find
- Fetch only
- Transient request from system pack
- Transient request from program pack
- Fetch to address
- Fetch to address with find

LOAD OR FETCH WITH FIND

Upon receiving this request, a request indicator byte (RIB) is set for a find condition. The requested module is located and control is passed to the Loader and Relocation program (NLOADR). NLOADR moves the specified module into main storage and, if required, adds a relocation factor to the module's address. Control is then passed to one of two locations:

1. If the module was moved using a fetch request, control is passed to the General Exit routine. The General Exit routine (1) restores register 1, (2) restores system linkages, and (3) passes control to the routine that was fetched into main storage.
2. If the module was moved using a load request, control is passed to the General Exit routine located within NENTRY. The General Exit routine restores register 1, restores system linkages, and returns control to the calling routine.

LOAD OR FETCH ONLY

If a load or fetch only request is received, control is passed directly to the Loader and Relocation program (NLOADR). NLOADR moves the requested module into main storage and adds a relocation factor to the module's addresses if required. Control is then passed to one of two locations:

1. If the module was moved using a fetch only instruction, control is passed via the General Exit routine to the routine that was fetched into main storage.
2. If the request was a load only, control is passed to caller.

TRANSIENT REQUESTS

Upon receiving a request for a transient, the transient area is indicated as busy. The requested transient address (Q and R bytes, cylinder/sector/number) is placed in the active transient ID, and the requested transient is then moved into the transient area (located in main storage at location X'100'). Control is then passed to the loaded transient. Any active transient can issue three types of requests depending upon the stage of completion:

1. The transient is not through processing and would like to bring another transient into main storage. Control is passed to the Transient Area Scheduler (NENTRY) where the address of the calling transient is placed on a transient queue, NXQUE, located in the Supervisor. The requested transient is then loaded into main storage; and control is passed to the transient, then back to the Transient Area Scheduler (see arrows 1 and 3 of Figure 2-1).
2. The transient is through processing and would like to pass control to another transient. Control is passed back to the Transient Area Scheduler (NENTRY) and the requested transient is loaded into main storage at X'100'. Control is then passed to it.
3. The transient has completed its processing and would like to return to its caller. Control is passed to the Transient Area Scheduler (NENTRY). If the transient queue is empty, control is passed to the General Exit routine; otherwise control is passed to the transient located in level number one of the queue.

Transients can be loaded from either the system pack or the program pack. If the transient is to be loaded from the program pack, the user must specify the program pack.

Instructions for finding a transient are:

```
LA PARM, XR2
B NCENTR
DC XL1'81'
```

PARM is a 12-byte parameter list which contains the information for finding the transient:

```
Byte 1 - X'00'
Byte 2-7 - name of the transient
Byte 8 - P for program pack or S for system pack
Byte 9-12 - Reserved for supervisor
```

The C/S/N for the transient is returned in the first three bytes of PARM. These must be moved to the 3-byte DC following the next supervisor call.

The instruction for loading a transient is:

```
B NCENTR
DC XL1'80' (the RIB to load by)
DC XL3'CSN' (disk location of the transient)
```

The high order bit of N is '1111xxxx' if the transient is to be loaded from the program pack.

The following example of the transient queue illustrates the four queue levels. This queue is located at the label NXQUE within the Supervisor.

	1	1	1	1	1	2	2	Total of 9 bytes per level
Level Number 4 on Q	Q	R	C	S	#	XR1		Next sequential instruction
Level Number 3 on Q	Q	R	C	S	#	XR1		Next sequential instruction
Level Number 2 on Q	Q	R	C	S	#	XR1		Next sequential instruction
Level Number 1 on Q	Q	R	C	S	#	XR1		Next sequential instruction

The transients are loaded into and removed from the transient queue via level one. See *Appendix B. Diagnostic Aids* for the procedure for tracing the transient queue.

The following section describes in detail the organization of the Model 6 Supervisor routines. Included in the following descriptions are:

- Description containing: name of routine, input, output, function, exits.
- Main storage map showing the routines that might be in main storage at the same time as the routine being described.
- Flowchart showing the functional flow of the routine being described. (Flowcharts are shown in Part 2 of this manual.)

► **General Entry/Exit and Transient Area Scheduler (NENTRY)**

CHART: BA
 FIGURE: 4-1
 ENTRY POINT: NENTRY
 FUNCTION:

- General Entry functions are to save:
 1. System registers
 2. System linkages
 3. Caller's return address
- Transient Area Scheduler performs the following:
 1. Analyzes requests for transients
 2. Locates the corresponding transient routine
 3. Retrieves the specified routine from disk storage
 4. Passes control as requested
 5. Tests to find if the Loader and Relocation program (NLOADR) was requested:
 - a. If requested, control is passed to NLOADR.
 - b. If not requested, control is passed to the General Exit.
- General Exit restores system registers and linkages and passes control to the calling routine.

EXIT:

- Normal: Calling program or passed to the program called.
- Error: A halt 135 is issued if there are too many (five or more) nested transients, or if a transient is called when no more can be nested.

X'000'	Entry Point Linkage
	Keyboard Save Area
	Common DCs
	System Read Routine (NREAD)
	System Communication Region
	Program Level Communication Region
	LSR Store/Display Routine
X'100'	Transient Area
X'400'	Keyboard Translate Table
	General Entry and Transient Area Scheduler (NENTRY)
	Resident Loader (NLOADR)
	Keyboard Interrupt Handler
	Resident Dump Linkage (CDUMPD)
	Disk Input/Output Supervisor
	User's Main Storage

Figure 4-1. Main Storage Map for General Entry/Exit and Transient Area Scheduler (NENTRY) for Supervisor

► Resident Loader and Relocation Program (NLOADR)

CHART: BB

FIGURE: 4-1

ENTRY POINT: NLOADR

FUNCTION:

- The Resident Loader loads the requested routines by using a fetch or a load RIB request
- The Relocation program adds a relocation factor to the loaded program and passes control to the General Exit routine

INPUT: Register 1 contains the address of the program level communication region.

OUTPUT: The requested module is loaded as specified by the caller's request indicator byte (RIB). See *Appendix B. Diagnostic Aids* for RIB procedure.

EXIT:

- Normal:
 1. After issuing a load request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the caller.
 2. After issuing a fetch request, control is passed to the General Entry/Exit routine (NENTRY) which in turn passes control to the program that was fetched into main storage.
- Error: A halt of 135 is issued if the requested load is below the beginning address of the program level.

► Local Storage Register Display Routine (LSR)

CHART: None

FIGURE: 4-1

ENTRY POINT: A manual branch via the CE console switches (see the LSR display procedures in *Appendix B. Diagnostic Aids*).

FUNCTION: The Local Storage Register Display routine displays registers without destroying the contents of the displayed register by issuing two halts:

1. The Q-byte of the first halt is the high-order byte of the register being displayed.
2. The Q-byte of the second halt is the low-order byte of the register being displayed.

INPUT: Console Address/Data switches

OUTPUT: The roller lights indicate the contents of the register being displayed.

EXIT: None

► Resident Dump Linkage (CDUMPD)

CHART: BC

FIGURE: 4-1

ENTRY POINT: CDUMPD

FUNCTION:

- Seeks to cylinder zero on disk pack R1
- Writes the three sectors, beginning at X'100', on R1, cylinder 0, sectors B4-B8
- Loads in the Dump Selection program (DMPFND) from cylinder zero of the system disk pack into the transient area. (See *Appendix B. Diagnostic Aids*.)
- Branches to the Dump Selection program (DMPFND) at location X'0100'.

INPUT: None

OUTPUT: None

EXIT: Control is passed to the Dump Selection program (DMPFND).

► System Disk Read Routine (NREAD)

CHART: None

FIGURE: 4-1

ENTRY POINT: NREAD

FUNCTION:

- Reads located modules from disk into main storage.
- Checks for successful read from disk storage.

INPUT: System IOB

OUTPUT: None

EXIT:

- Normal: Control is returned to the caller.
- Error: A halt of 135 is displayed on the halt display lights.

Figure 4-2 contains the directory entries for the Model 6 Supervisor.

Module/Phase Name	Chart ID	Descriptive Name	Entry Point	Function
NENTRY—N/A	BA	General Entry/Exit and Transient Area Scheduler	NENTRY	<ul style="list-style-type: none"> ● Saves all system registers and linkages ● Analyzes user and system requests ● Locates corresponding transient routines ● Passes control to transient ● Passes control to Loader or General Exit ● Restores system registers and linkages
NLOADR—N/A	BB	Resident Loader and Relocation Program	NLOADR	<ul style="list-style-type: none"> ● Loads the requested routine ● Passes control to General Exit routine, which in turn passes control to the caller; or, ● Passes control to the routine that was fetched via General Exit routine ● Passes control to General Exit routine upon completion
HLT1	none	Local Storage Register Display Routine	HLT1	<ul style="list-style-type: none"> ● Displays registers without destroying the contents
NREAD	none	System Read Routine for Disk	NREAD	<ul style="list-style-type: none"> ● Reads located modules from disk into main storage and checks for successful read
CDUMPD	BC	Resident Dump Linkage	CDUMPD	<ul style="list-style-type: none"> ● Seeks to cylinder zero on R1 ● Writes out the transient area ● Loads in the Dump Selection Program from cylinder zero of the system pack ● Branches to the Dump Selection Program at location X'100'

Figure 4-2. Model 6 Supervisor Directory

PART 5.
SCHEDULER

The Scheduler, through the use of control statements, provides the linkage between the user and the system. The Scheduler provides three major functions:

1. Performs IPL and end of job functions
2. Reads, diagnoses, and processes control cards
3. Performs required initialization for each new program

In order to accomplish the above functions, the Scheduler is divided into three phases.

1. The Terminator
2. The Reader/Interpreter
3. The Initiator

SYSTEM REQUIREMENTS

Disk System

The Scheduler requires the following system configuration:

- One 5203 Line Printer (Model A1)
- One 5410 Processing Unit (Model A13)
- One 5424 Multi-Function Card Unit (Model A)
- One 5444 Disk Storage Drive (Model 2)

Model 6

- One 5213 Matrix Printer (Model 1)
- One 5406 Processing Unit with Keyboard
- One 5444 Disk Storage Drive (Model 2)

The Scheduler will also support the 5496 Data Recorder (Model 1).

The chart in Figure 5-1 gives an overview of the Scheduler. The routines that are shown are discussed in greater detail in *Section 3—Program Organization* in this part (Part 5) or *Part 6*.

The data areas that are shown in the following chart are discussed in *Appendix A. Data Area Formats*.

Note: All Scheduler routines use either the system communication region or the program level communication region.

TERMINATOR

The terminator has three functions:

- IPL initialization
- EOJ deallocation
- Re-initialization

Following the user's Initial Program Load (IPL), control is passed to the Terminator IPL routine (\$\$TMIP) which in turn performs all necessary initialization of the scheduler work area (SWA) and system and program level communication areas needed by the Scheduler by means of the volume label and configuration record. Control is then passed to the Reader/Interpreter Mainline (\$\$RDML) on the Disk System, or to the Conversational Scheduler Initializer (\$\$RBIP) on Model 6.

After this initial entry, the Scheduler is always re-entered through the Terminator End of Job routine (\$\$TMEO). This routine frees the disk files used by the last program and allows them to be re-used by the next program. Control is then passed to the Terminator Re-initialization program (\$\$TMRI) which frees devices, SWA, and initializes all disk and main storage areas that are needed to run the next program. Control is passed from the Terminator to the Reader/Interpreter Mainline (\$\$RDML) on the Disk System, or to the Conversational Scheduler Initializer (\$\$RBIP) on Model 6.

READER/INTERPRETER

Conversational Reader/Interpreter

IPL Time

When the Conversational Scheduler Initializer (\$\$RBIP) receives control from the Terminator at IPL time, \$\$RBIP determines whether or not the data recorder is a supported sysin device. If the data recorder is supported in the system configuration, \$\$RBIP prompts for the date, then prompts **READER**. If **DATA96** is the response, \$\$RBIP calls the End of Job transient (\$\$SPEJ), after which the Terminator Re-Initialization Routine (\$\$TMRI) passes control to the Reader/Interpreter Mainline (\$\$RDML). If the data recorder is not supported at IPL time, or if the response to **READER** is not **DATA96**, the Conversational Scheduler Initializer (\$\$RBIP) retains control and continues as it does during other than IPL time.

Other Than IPL Time

When the Conversational Scheduler Initializer (\$\$RBIP) receives control from the Terminator during other than IPL time, \$\$RBIP prompts **READY**. **READER** (if the data recorder is supported), **LOG**, **LOAD**, **BUILD**, **BUILD**, **BUILD**, **CALL** are valid responses at this time. If the response is **READER** and the data recorder is supported, \$\$RBIP prompts **READER**. A reply of **DATA96** causes control to pass to the Reader/Interpreter Mainline (\$\$RDML) by way of End of Job (\$\$SPEJ).

Log routine (\$\$RBLG) initiates logging immediately if the response to **READY** is **LOG**. For each of the other four valid responses to **READY** (**LOAD**, **BUILD**, **BUILD**, **CALL**), \$\$RBIP passes control to Conversational Scheduler Control (\$\$RBCO), which then passes control to Build-Chained Cycle (\$\$RBBC) if the response to **READY** was **BUILD**, or to Call Cycle (\$\$RBCL) by way of Call Control (\$\$RBCC) if the response to **READY** was **CALL**. \$\$RBCO retains control if the response to **READY** was **LOAD** or **BUILD**.

After all required OCL statements have been keyed under the supervision of \$\$RBCO, \$\$RBBC, or \$\$RBCL, control passes to Modify (\$\$RBM) until the command **RUN**. If a **LOAD** or **CALL** is to be executed, Run (\$\$RBRN) receives control and passes control to the Initiator (\$\$INPS) after the **LOAD** or **CALL** has been executed. If a **BUILD** or **BUILD** is to be executed, \$\$RBM passes control to the Procedure Library Interface (\$\$RBM1) and Utility OCL Processor (\$\$RBM2) to build the card images in the procedure library. The End of Job transient (\$\$SPEJ) then receives control.

Disk System Reader/Interpreter Mainline (\$\$RDML)

After receiving control from the Terminator, the Reader/Interpreter Mainline (\$\$RDML) reads from either the supported Sysin device (console, MFCU, data recorder) or the source library, if a procedure has been called. It then determines if the record contains valid operation control language (OCL). Once a valid control record is located, the record is encoded, and control is passed to the proper control card processor. After the record has been processed, the control card processor returns control to the Reader/Interpreter Mainline (\$\$RDML) which in turn checks for any procedure overrides. If an override exists, control is returned to the control card processor to process the override. If an override does not exist, control is returned to the control card processor to perform end of record diagnostics. The Reader/Interpreter processes each control card in the same manner until the // **RUN** control card is read. The // **RUN** control card processor passes control to the Initiator.

INITIATOR

The RUN routine passes control from the Reader/Interpreter to the Initiator Program Setup—Phase One (\$\$INPS). This routine (\$\$INPS) allocates the resources necessary for the user program to execute. Control is passed from \$\$INPS to the first of the Initiator Disk File Processor phases: Phase One (\$\$INPD), Phase Two (\$\$INDF), Phase Three (\$\$INDC), Phase Four (\$\$INDS). The Initiator Disk File Processor decodes the pre-format one images and builds them in the scheduler work area for the requested disk files. The format one images are then read from the scheduler work area and checked for being unique and valid format one images. The format one images are then sorted into the order required by the allocate and IOS Open and Close routines. Control is passed from Phase Four (\$\$INDS) to the Initiator Program Setup—Phase Two (\$\$INP2). This routine uses the attributes of the program to be loaded for this job to set up the necessary conditions before giving control to the supervisor to load the program. For example, RPG II Compiler programs require \$WORK and \$SOURCE file allocation. \$\$INP2 calls \$\$STAI to allocate the files. Compiler programs also require the source to be copied into the \$SOURCE file. Some programs require SYSIN for the entire job. These and other conditions are set up as necessary by \$\$INP2.

If the operator presses the Printer-Keyboard REQ key on the Disk System, or sets the INQUIRY REQUEST key on the Model 6, control is passed to the first phase of the Roll-

out phases: Phase One (\$\$STRO), Phase Two (\$\$STRU), Phase Three (\$\$STRT). The Rollout phases move the interrupted user program into disk storage. Control is then passed from Rollout—Phase Three (\$\$STRT) to the Terminator Re-initialization program (\$\$TMRI). The interrupting program is loaded into main storage by \$\$TMRI and processed. Only at the completion of the processing of the interrupting program will the Terminator Re-initialization program (\$\$TMRI) pass control to Rollin—Phase One (\$\$STRI). The Rollin phases restore the interrupted program's data areas and pass control to the instruction that was to be executed next. Upon end of job, the RPG user program passes control back to the Terminator End of Job routine (\$\$TMEO) by way of the End of Job transient (\$\$SPEJ).

The Allocation Initiator (\$\$STA1) is entered from either Initiator Program Setup—Phase Two (\$\$INP2) or from the user's RPG program. The Allocation Initiator loads the Initiator Allocate routine (\$\$INAI) and moves the last 4K (4096) bytes of the user's main storage into the scheduler work area. Control is then passed to either Resource Allocation Phase One (\$\$STRI) followed by Phase Two (\$\$STR2) for allocating non-disk devices or to the Initiator Allocate Phases: Phase One (\$\$INA1), Phase Two (\$\$INA2), Phase Three (\$\$INA3), Phase Four (\$\$INA4). Phase Four (\$\$INA4) passes control to the Initiator Allocate Terminator (\$\$INAT) which in turn restores, to main storage, the user's last 4K bytes that was moved to disk storage. Control is returned from \$\$INAT to the calling routine.

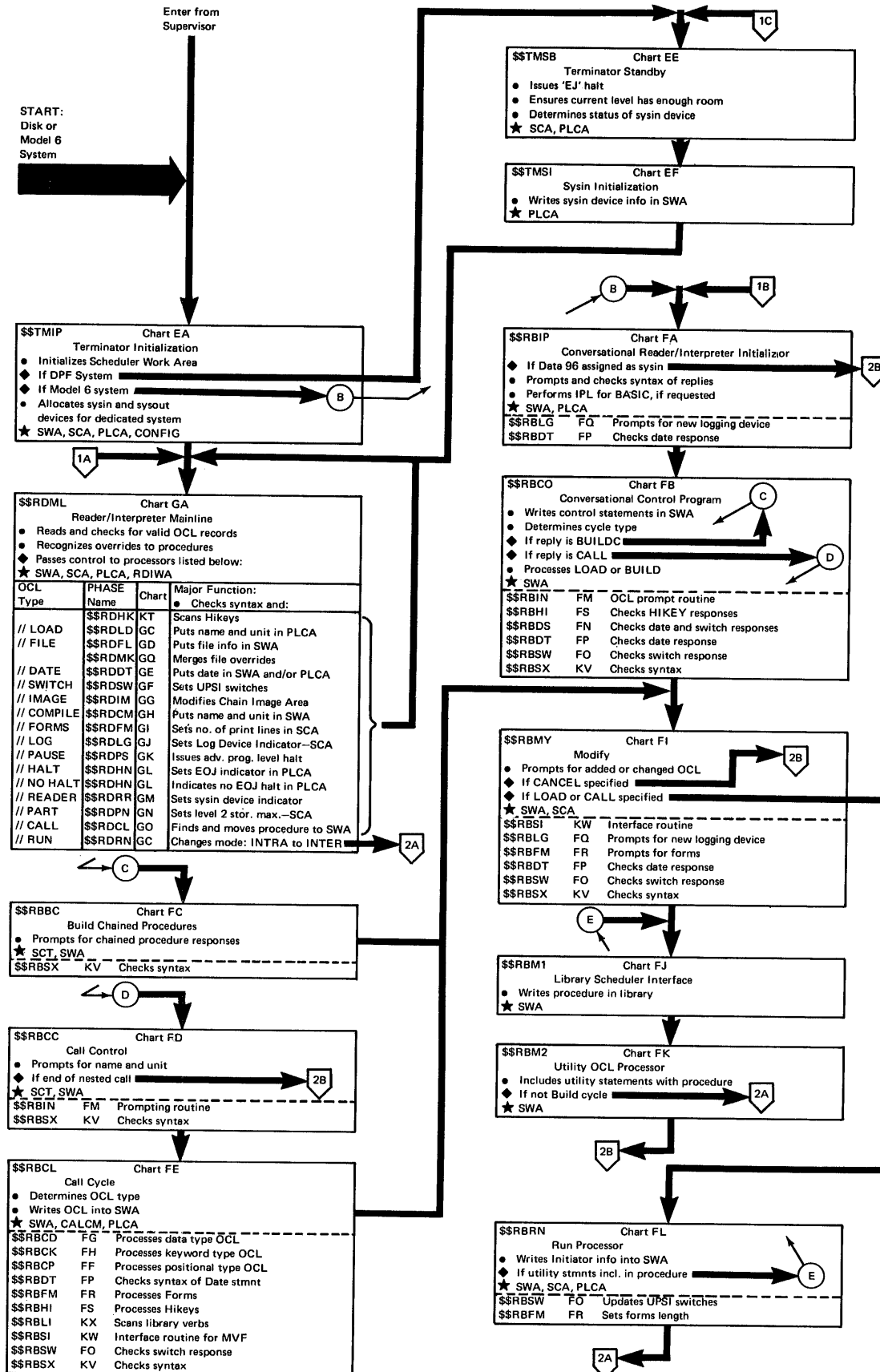


Figure 5-1. Main Logic of Scheduler for Disk and Model 6 Systems (Part 1 of 2)

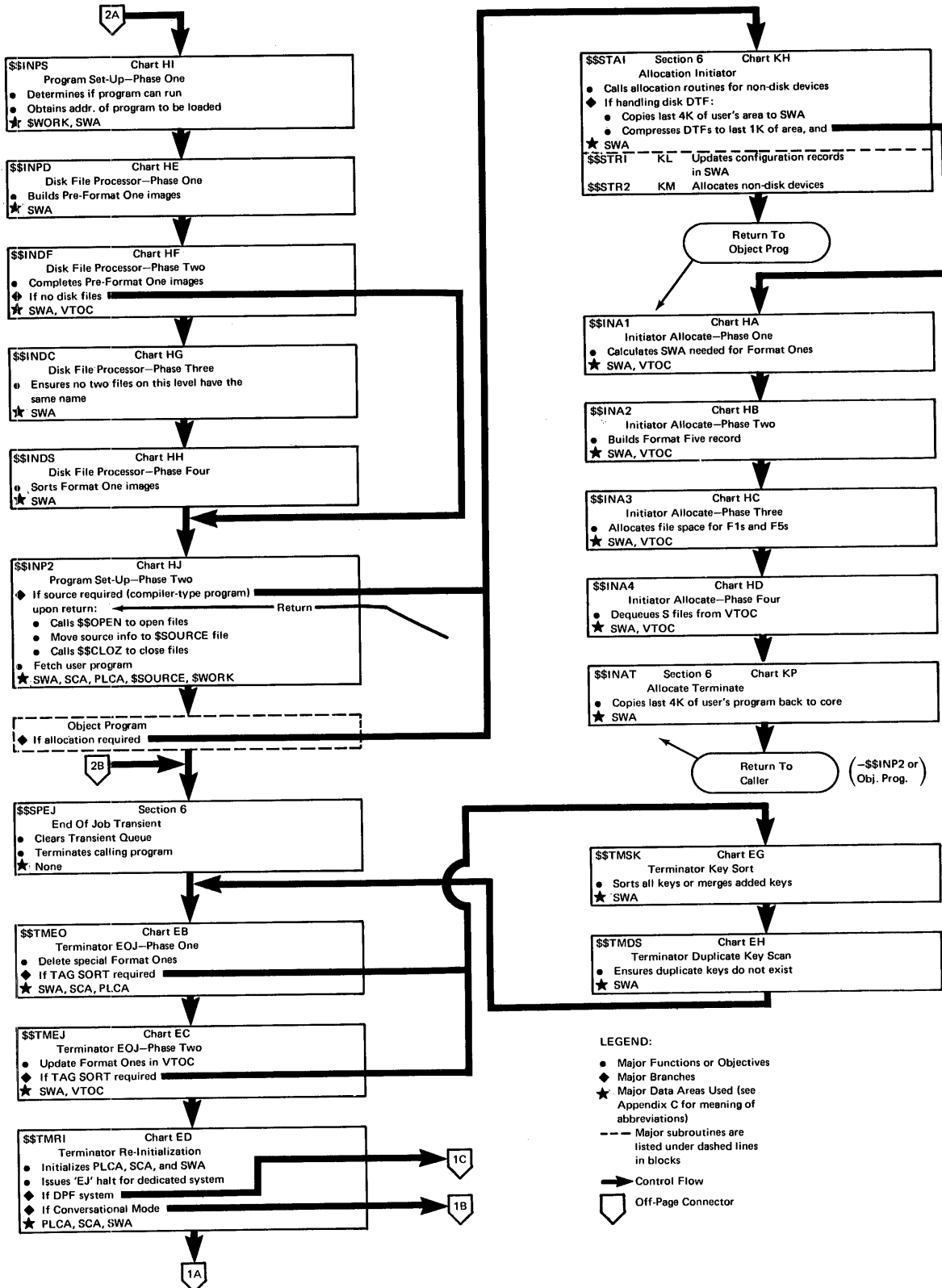


Figure 5-1. Main Logic of Scheduler for Disk and Model 6 Systems (Part 2 of 2)

This section describes, in detail, the organization of the Scheduler routines shown in Figure 5-1. Included in the following discussion are:

- Descriptions containing: name of the routine, input, output, function, and exits (both normal and error) from this routine.
- Main storage maps showing the routines that are found in main storage at the same time as the routine that is being described.
- Flowcharts showing the functional flow of the routine being described.

This Program Organization section is divided into four parts:

1. Terminator
2. Reader/Interpreter for Model 6 System
3. Reader/Interpreter for Disk System
4. Initiator

1. TERMINATOR

► Terminator Initial Program Load (IPL) (\$\$TMIP)

CHART: EA

FIGURE: 5-2

ENTRY POINT: \$\$TMIP

FUNCTION:

- Initializes the following scheduler areas of the system communication region: NCSYSL, NCSLOG, NCSWRK, NCSYSQ, NCSCH1, NCSMV2, NCSCH, NCSCH2, NCSTOR.
- Allocates the system input (Sysin) and output (Sysout) devices in a dedicated system.
- Determines whether or not this is a Model 6 System.

INPUT:

- Configuration record
- Volume label of system pack

OUTPUT:

- The system communication region reflects C/S device information for the system logging device module.
- The scheduler area of the system communication region is initialized.
- Print image address register is initialized
- Model 6 bit is set (NCMBVS is X'80')
- Conversational bit mode is set (NCSCH is X'20').
- Address of end of core (NPEND) and C/S device information for the system input device module in the program level communication region are set.

EXIT:

- Disk Dedicated: Reader/Interpreter Mainline (\$\$RDML).
- Disk DPF: Terminator Standby Routine (\$\$TMSB) for both levels
- Model 6: Conversational Reader/Interpreter (\$\$RBIP)

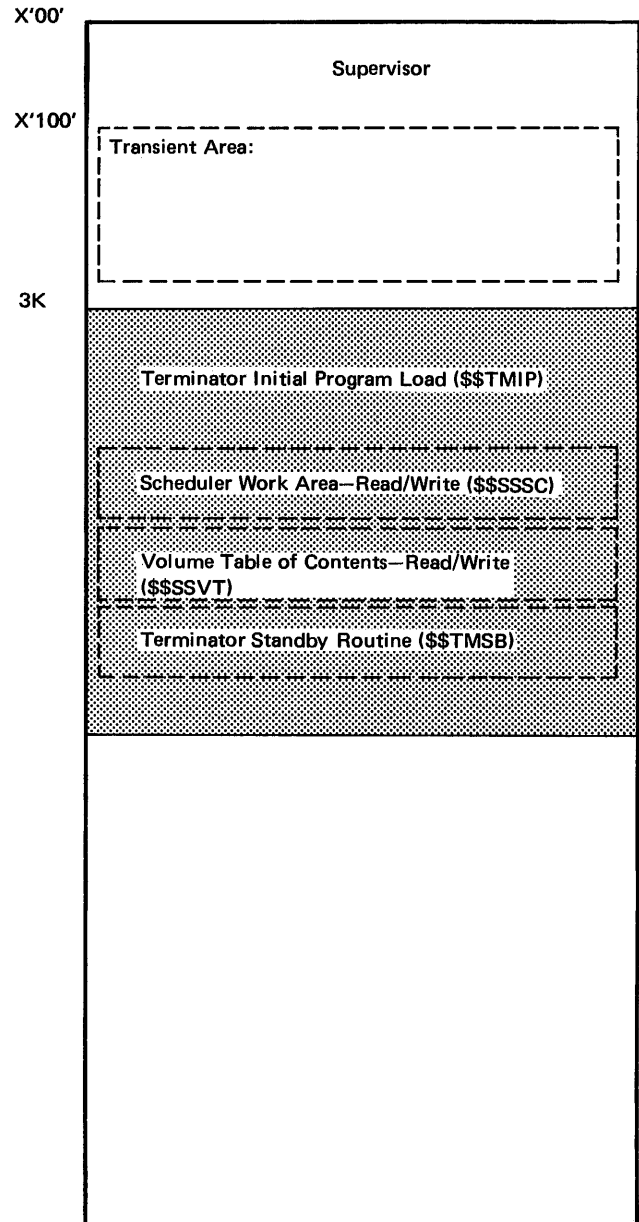


Figure 5-2. Main Storage Map for Terminator Initial Program Load (IPL) Routine (\$\$TMIP)

► Terminator End of Job—Phase One (\$\$TMEO)

CHART: EB
 FIGURE: 5-3
 ENTRY POINT: \$\$TMEO
 FUNCTION:

- This routine performs the initial setup for the terminator End of Job—Phase Two (\$\$TMEJ), and also the successive setups when Tag Sort (\$\$TMSK) returns to the end of job routines.
- The following items are involved in the setup:
 1. The program level is determined
 2. The system date is formatted
 3. The DPF indicator is set
 4. All disk packs used by this job are determined
 5. The four special F1s are deleted from the SWA if they exist

INPUT:

- Format one images from the scheduler work area (SWA)
- System and program level communication regions

OUTPUT: The SWA is set up for the second phase (\$\$TMEJ).

EXIT:

- Second phase of the End of Job (\$\$TMEJ) to release files.
- \$\$TMSK if a Tag Sort is required.
- Terminator End of Job—Disk Status routine (\$\$TMST) for SIO logging if no FILE statements exist.

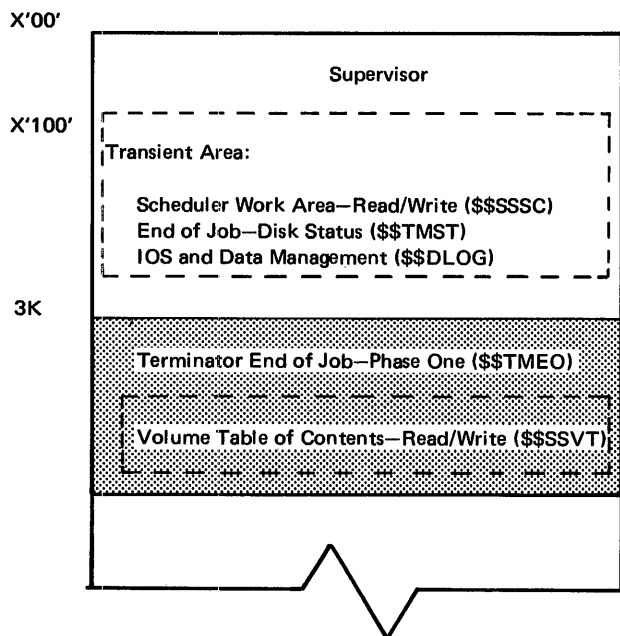


Figure 5-3. Main Storage Map for Terminator End of Job—Phase One Routine (\$\$TMEO)

► Terminator End of Job—Phase Two (\$\$TMEJ)

CHART: EC
 FIGURE: 5-4
 ENTRY POINT: \$\$TMEJ
 FUNCTION:

- Creates an entry in the volume table of contents (VTOC) for every new file that was used during the past job.
- Updates existing entries in the VTOC for each old file that was used during the past job.
- Deletes existing entries in the VTOC for the remove function of the VTOC delete utility program.

INPUT: Data contained in the format one work area located at the label FORMT1 within the SWA. See *Appendix A. Data Area Formats* for the contents of the format one work area.

OUTPUT:

- Updated VTOC index
- Updated volume label

EXIT:

- Terminator Re-initialization Phase (\$\$TMRI)
- Tag Sort (\$\$TMSK) if sorting is required

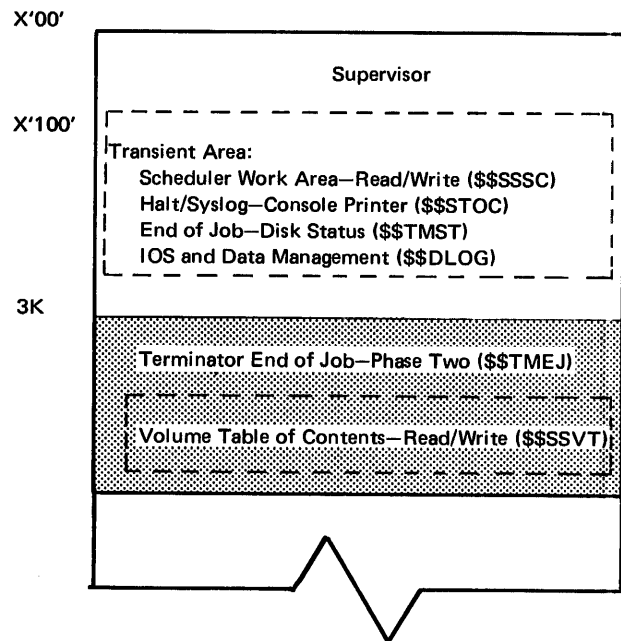


Figure 5-4. Main Storage Map for Terminator End of Job—Phase Two Routine (\$\$TMEJ)

► Terminator Re-Initialization (\$\$TMRI)

CHART: ED

FIGURE: 5-5

ENTRY POINT: \$\$TMRI

FUNCTION:

- Re-initializes the following data areas:
 1. SWA.
 2. Program level communication region.
 3. System communication region.
- Determines if system is Model 6.

INPUT:

- SWA
- Program level communication region

OUTPUT: The above data areas are re-initialized.

EXIT:

- Terminator Standby routine (\$\$TMSB) if using in a DPF system
- Reader/Interpreter Mainline (\$\$RDML)
- Conversational Scheduler (\$\$RBIP) for Model 6

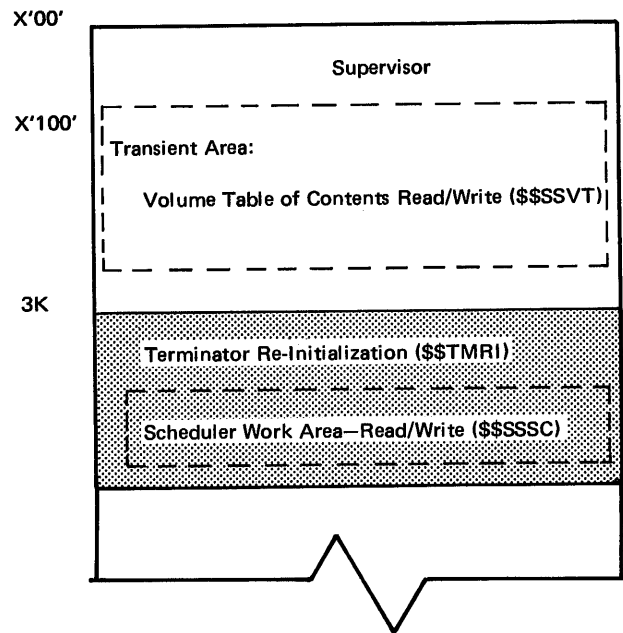


Figure 5-5. Main Storage Map for Terminator Re-Initialization Routine (\$\$TMRI)

► Terminator Standby Routine (\$\$TMSB)

CHART: EE

FIGURE: 5-6, 5-7

ENTRY POINT: \$\$TMSB

FUNCTION:

- Issues end of job (EJ) halts.
- Ensures that current program level has enough main storage before proceeding.

- Expands to 4K if the current program level is level two.
- Determines the availability of the Sysin device desired for this level by checking NPBSD in PLCA.
- If Sysin device is available, the device is indicated at label NPSCH5 in PLCA.

INPUT: Reader/Select switch setting

OUTPUT: Device indicated at NPSCH5 in PLCA

EXIT: Terminator Sysin Initialization routine (\$\$TMSI)

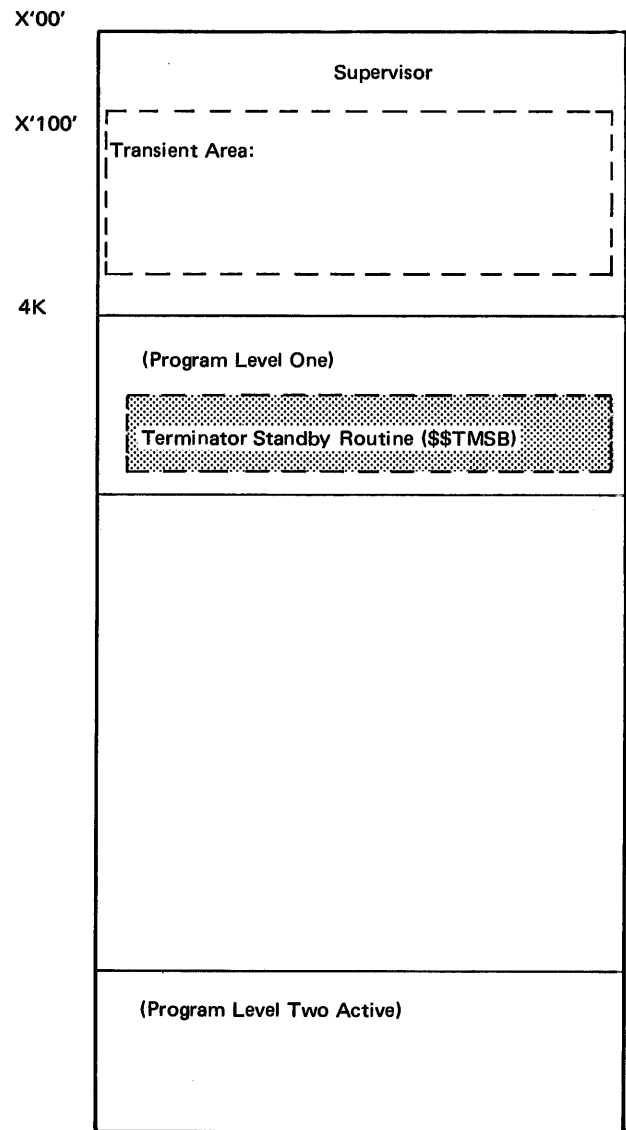
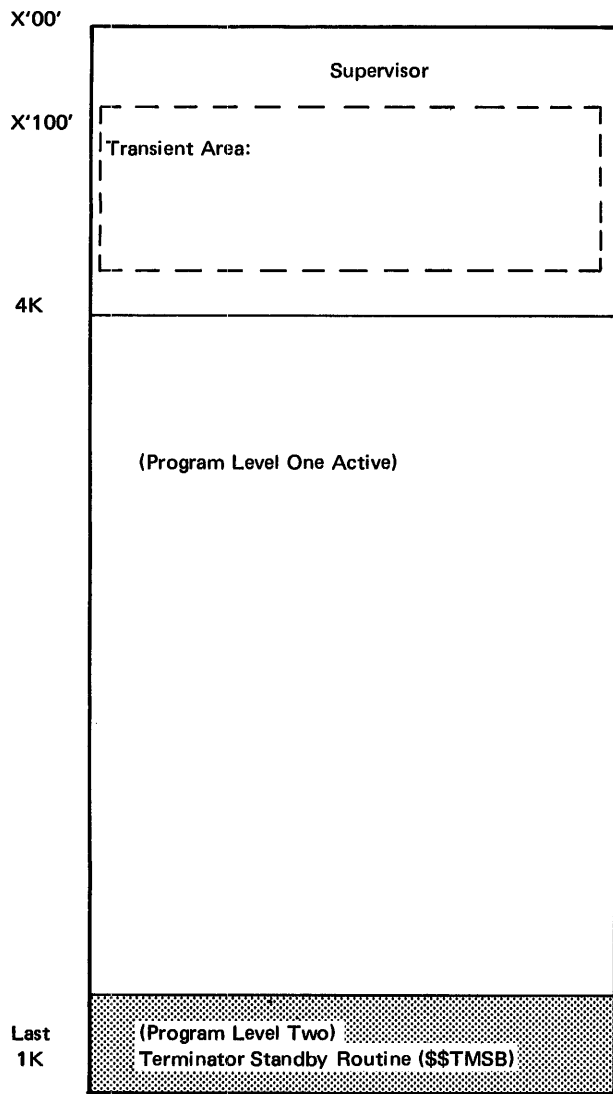


Figure 5-6. Main Storage Map for Terminator Standby Routine (\$\$TMSB)--Level One Active

Figure 5-7. Main Storage Map for Terminator Standby Routine (\$\$TMSB)--Level Two Active

► Terminator Sysin Initialization (\$\$TMSI)

CHART: EF

FIGURE: 5-8, 5-9

ENTRY POINT: \$\$TMSI

FUNCTION:

- Reads information from NPSCH5.
- Determines the location of the routine needed to support the selected input device.
- Places the information in NPSYSI.
- Indicates, at NPBPSD, that the selected input device is being used.

Note: This routine is used only by a DPF system.

INPUT: One byte (NPSCH5) of the program level communication region

OUTPUT: The program level communication region is updated to reflect the new system input device at labels NPSYSI and NPBPSD.

EXIT:

- Normal: Reader/Interpreter Mainline (\$\$RDML)
- Error: Supported Halt/Syslog routine to display the halt condition before control is passed to the End of Job-Phase One routine (\$\$TME0)

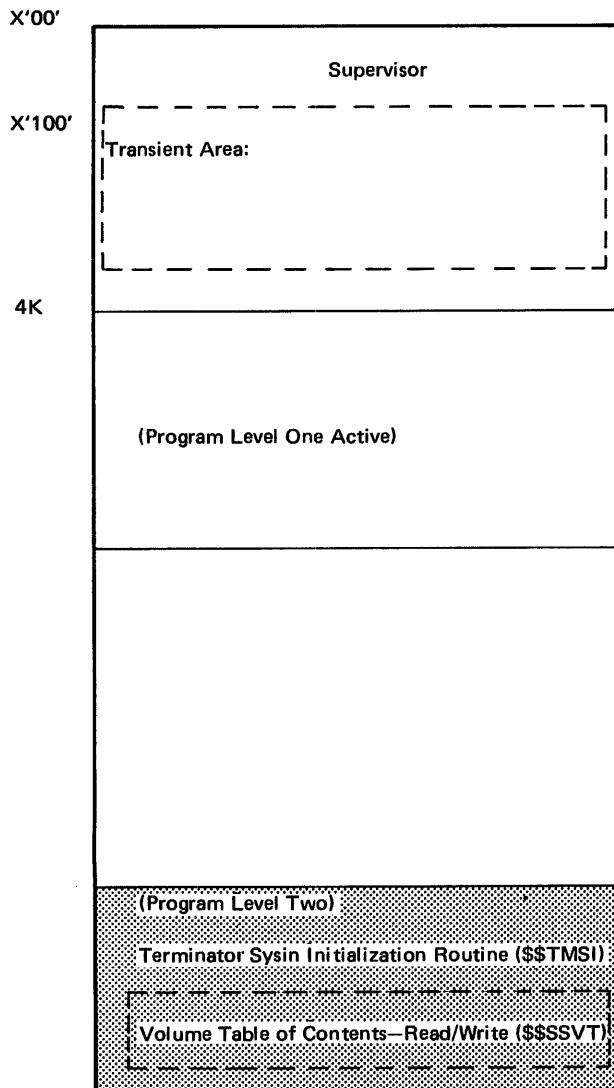


Figure 5-8. Main Storage Map for Terminator Sysin Initialization Routine (\$\$TMSI)-Level One Active

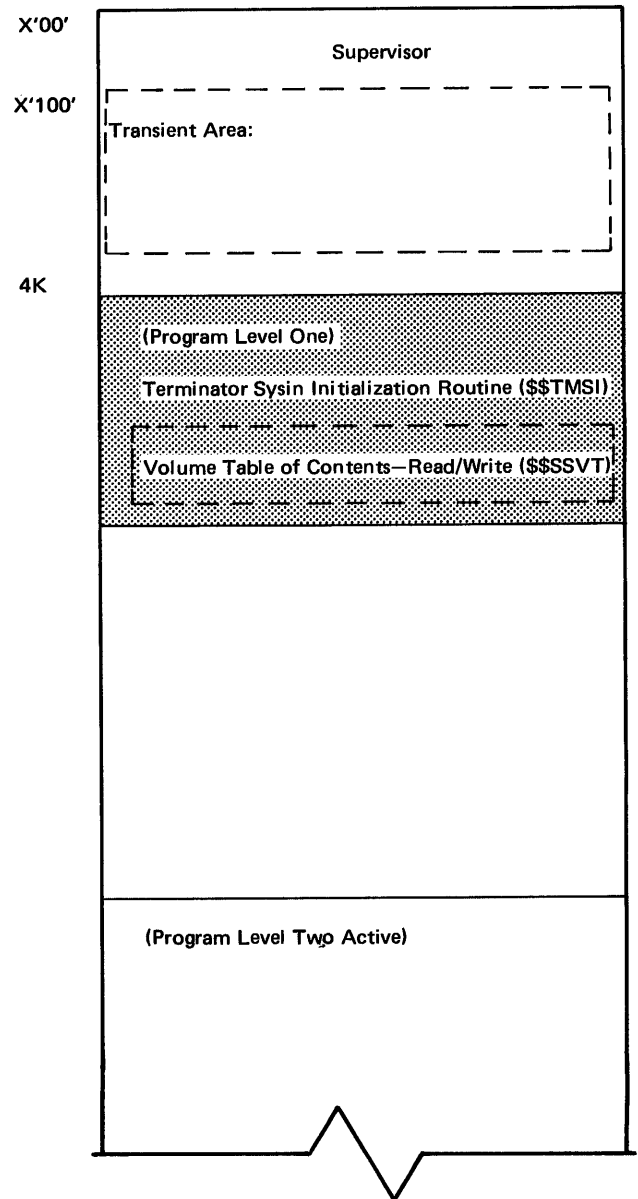


Figure 5-9. Main Storage Map for Terminator Sysin Initialization Routine (\$\$TMSI)-Level Two Active

► Terminator Key Sort (\$\$TMSK)

CHART: EG

FIGURE: 5-10

ENTRY POINT: DMTMSK

FUNCTION: Determines if the format one labels (located in sectors 3-15) are from:

1. An unordered load.
2. An index random add.
3. A sequential add.

INPUT: Format one labels located at label FORMT1 within the scheduler work area.

OUTPUT: An ordered disk file index located in the VTOC.

EXIT:

- Normal: Terminator Duplicate Key Scan Routine (\$\$TMDS)
- Error: Supported Halt/Syslog routine

► Terminator Duplicate Key Scan (\$\$TMDS)

CHART: EH

FIGURE: 5-11

ENTRY POINT: \$\$TMDS

FUNCTION:

- Reads the keys of all indexed load or add files.
- Scans the keys for duplicate entries.

INPUT: Format one label located at label FORMT1 within the scheduler work area

OUTPUT: The supported Halt/Syslog routine will list the duplicate entries.

EXIT:

- Normal: Terminator End of Job-Phase One (\$\$TMEO)
- Error: Supported Halt/Syslog routine receives control to print the duplicate key entry.

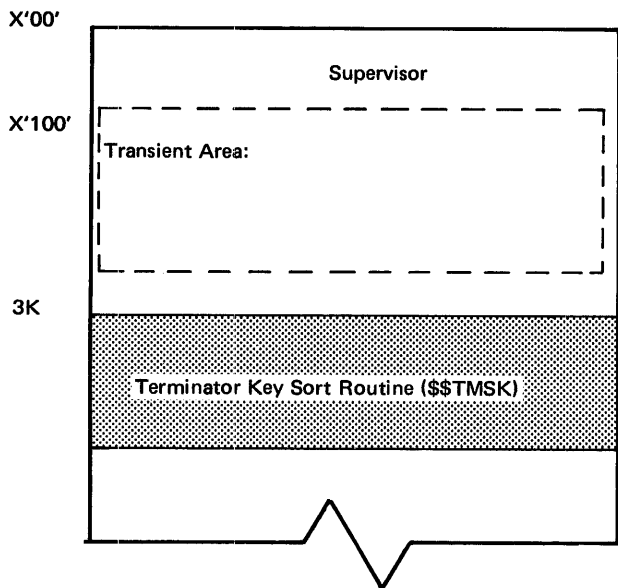


Figure 5-10. Main Storage Map for Terminator Key Sort Routine (\$\$TMSK)

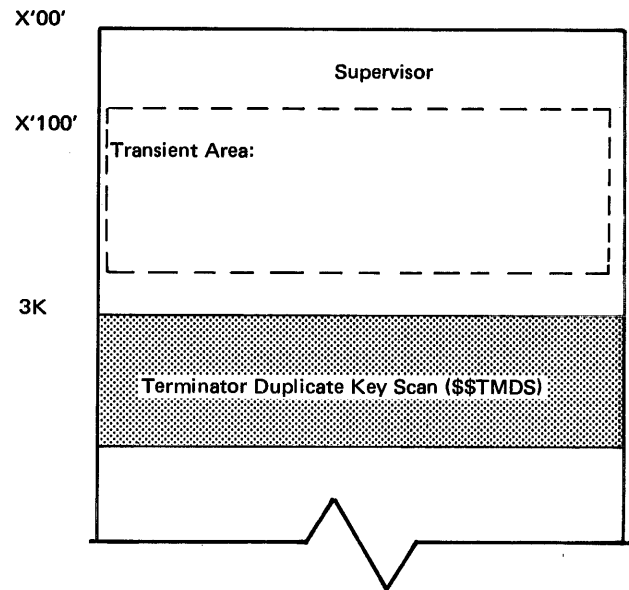


Figure 5-11. Main Storage Map for Terminator Duplicate Key Scan Routine (\$\$TMDS)

2. READER/INTERPRETER FOR MODEL 6 (CONVERSATIONAL MODE)

► Conversational Scheduler Initializer (\$\$RBIP)

CHART: FA

FIGURE: 5-12

ENTRY POINT: \$\$RBIP

FUNCTION:

- Initializes print element location.
- Prompts for and syntax checks the response to system date, if in IPL mode.
- Allows change of logging device if it is attached.
- Determines mode (conversational or not).
- Passes control to Reader/Interpreter Mainline, non-conversational mode (\$\$RDML), if it is required.
- Prompts for and syntax checks the type of job cycle requested if in conversational mode.
- Passes control to Scheduler Control (\$\$RBCO) if in conversational mode.
- Performs IPL procedure for System/3 BASIC, if required.

INPUT: Keyboard responses

OUTPUT: System date in program level communication region and SWA common sector

EXIT: Control is passed to the routine required by the response from the operator.

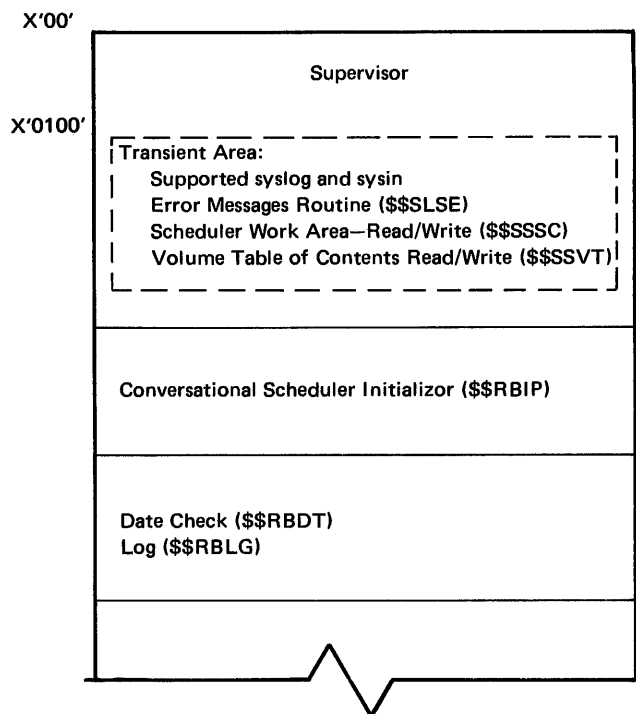


Figure 5-12. Main Storage Map for Conversational Scheduler Initializer (\$\$RBIP)

► Scheduler Control (\$\$RBCO)

CHART: FB

FIGURE: 5-13

ENTRY POINT: \$\$RBCO

FUNCTION:

- Loads the control table.
- Finds all phases needed for conversational mode.
- Writes encoded control statements into the SWA.
- Interfaces with supervisor for all scheduler phase loads.
- Determines cycle type (build, build-call, call, or load).

INPUT: XR1 contains a bit configuration specifying the job type:

- X'80' - Build
- X'40' - Load
- X'20' - Call
- X'10' - Build-call

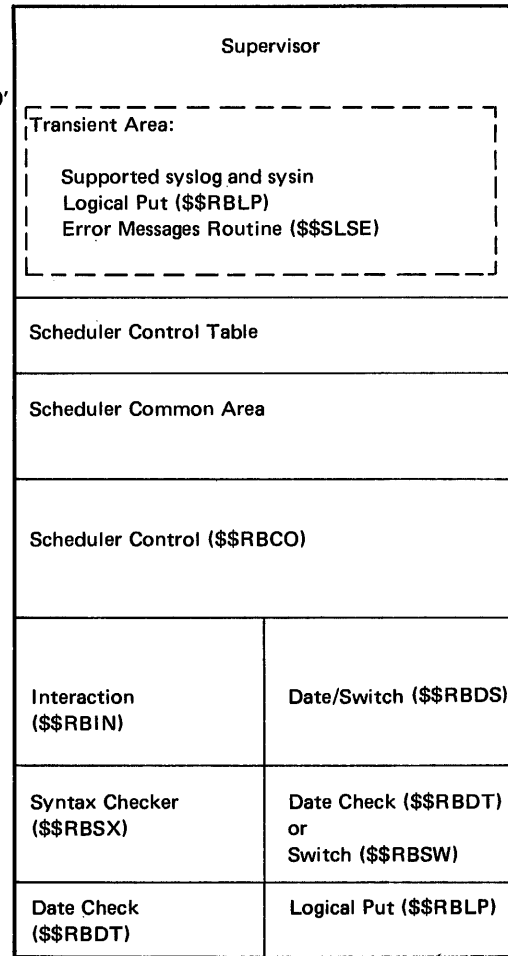
OUTPUT: Encoded keywords, appended sequence number, and operator responses are written into the SWA starting at the second sector of the initiator table.

EXITS:

- Normal:
 1. To Modify (\$\$RBMV) at the end of OCL prompting sequence during load or build.
 2. To Call Control (\$\$RBCC) if the job type is a Call cycle.
 3. To Build Chained (\$\$RBBC) if the job type is a Build Chain.
- Error: Cancel with /* response or End of Job (\$\$TMEO).

X'00'

X'0100'



First Prompt or
Prompt for File
Statements

Prompt for DATE
and SWITCH

Figure 5-13. Main Storage Map for Scheduler Control (\$\$RBCO)

► **Build-Chained Cycle (\$\$RBBC)**

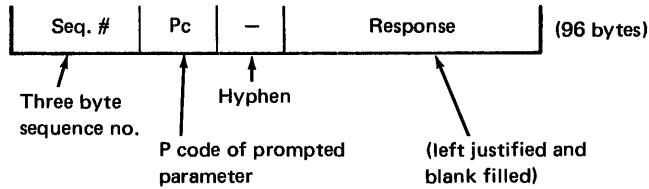
CHART: FC

FIGURE: 5-14

ENTRY POINT: \$\$RBBC

FUNCTION:

- Prompts operator for responses needed to build chained procedures.
- Syntax checks the responses.
- Checks for duplicate procedures.
- Writes correct responses in the SWA in the following format:



- Terminates when response to CALL NAME or UNIT is the ENTER MINUS (-) key.

INPUT:

- Keyboard responses
- Scheduler Control Table

OUTPUT: Responses written to the Scheduler Work Area

EXIT:

- Normal: Modify (\$\$RBMV)
- Error or Operator Cancel (/): End of Job (\$\$SPEJ)

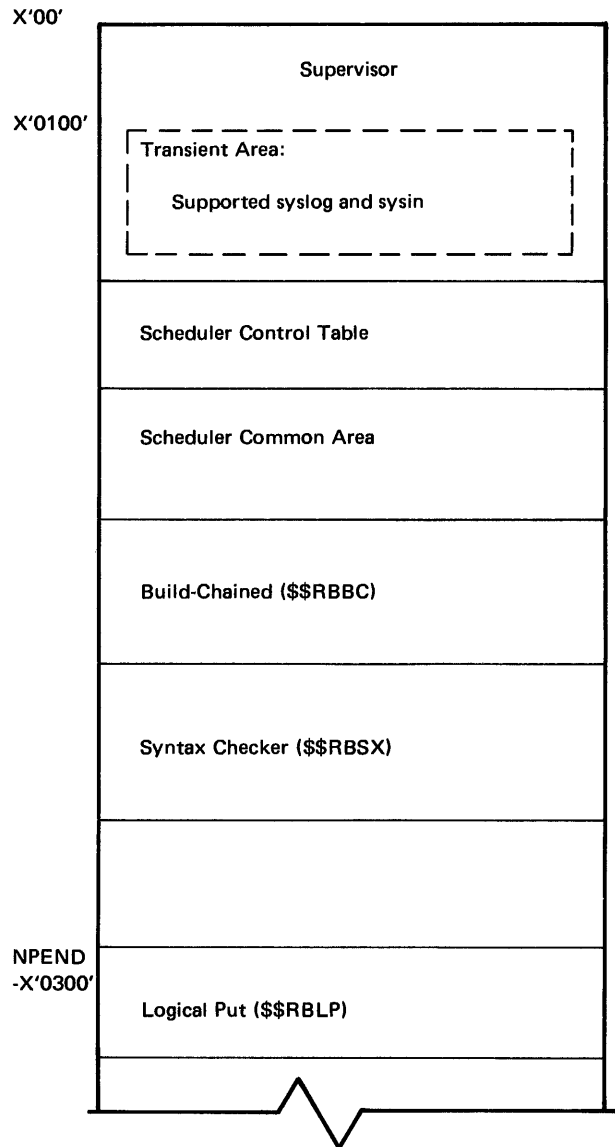


Figure 5-14. Main Storage Map for Build-Chained (\$\$RBBC)

► Call Control (\$\$RBCC)

CHART: FD

FIGURE: 5-15

ENTRY POINT: \$\$RBCC

FUNCTION:

- Prompts the operator for the name and unit of the procedure during a CALL.
- Prints the name and unit of the procedures, if the procedure contains chained calls of other procedures. Runs all the jobs at one time.
- Runs the job, if the procedure does not contain chained calls.

INPUT:

- XR1 points to Scheduler Common Area
- Scheduler Control Table
- Procedure called from source library

OUTPUT:

- For chained calls, the name and unit are printed
- First statement or procedure is passed to \$\$RBCL

EXIT:

- Normal: \$\$RBCL
- Error: Supported Halt/Syslog

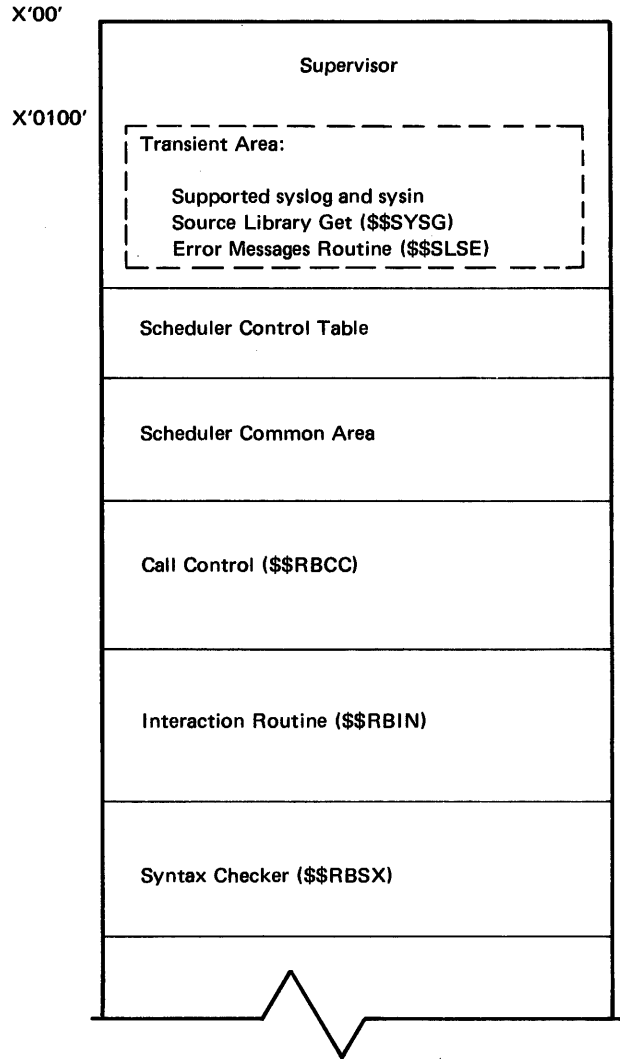


Figure 5-15. Main Storage Map for Call Control (\$\$RBCC)

► Call Cycle (\$\$RBCL)

CHART: FE

FIGURE: 5-16

ENTRY POINT: \$\$RBCL

FUNCTION:

- Determines type of OCL statements in procedure and loads proper processing modules.
- Writes OCL into SWA.

INPUT:

- Procedure library
- Return code from library interface

OUTPUT: Statements written in SWA

EXIT:

- Normal:
 1. Modify (\$\$RBMY), when run precedes end of records indicator and the no-go bit is not set on
 2. End of Job transient (\$\$SPEJ), after all records in a procedure have been processed
- Error: Error Logging Routine (\$\$SLSE)

X'00'

X'0100'

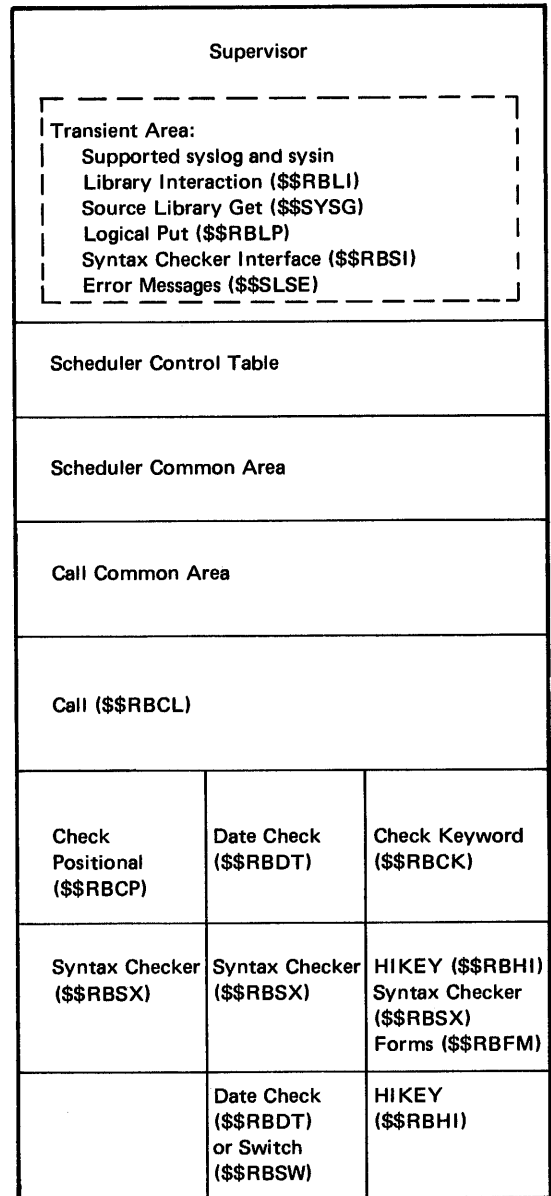


Figure 5-16. Main Storage Map for Call (\$\$RBCL)

► **Positional Statement Processor (\$\$RBCP)**

CHART: FF

FIGURE: 5-17

ENTRY POINT: \$\$RBCP

FUNCTION:

- Processes positional statements from procedure library (LOAD).
- Prompts for and syntax checks delayed statements.
- Writes OCL into the SWA.
- Waits for and syntax checks responses to delayed parameters.

INPUT:

- Procedure library record located in the scheduler common area
- Scheduler Control Table

OUTPUT: Statements listed on logging device

EXIT:

- Normal: Call Cycle (\$\$RBCL)
- Error: End of Job Transient (\$\$SPEJ)
Error Logging Routine (\$\$SLSE)

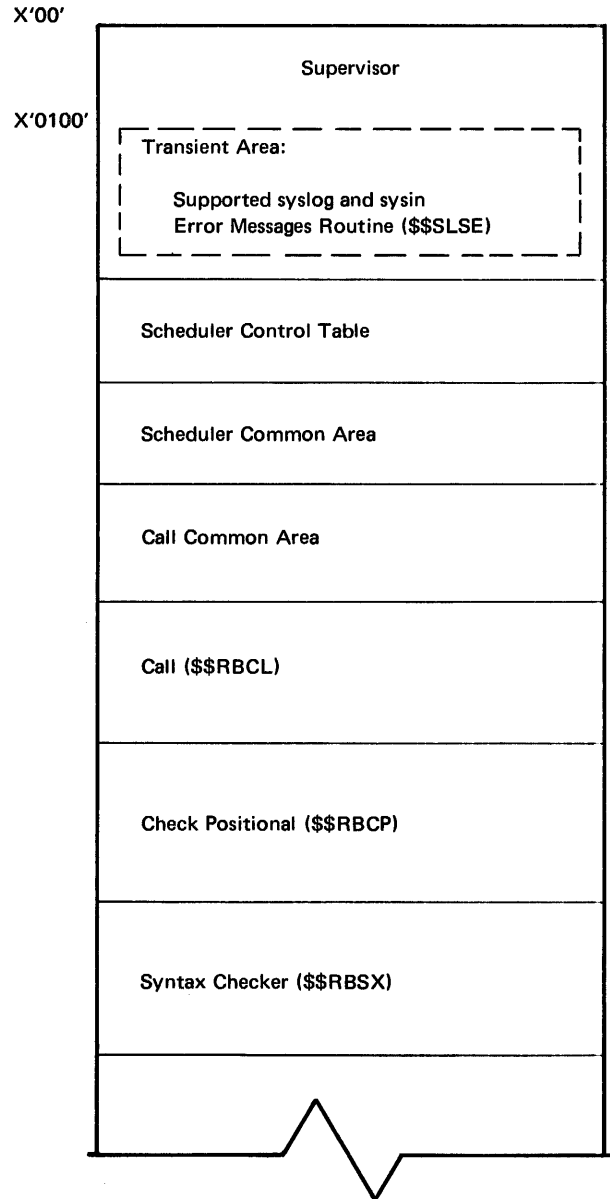


Figure 5-17. Main Storage Map for Check Positional (\$\$RBCP)

► **Data Type Statement Processor (\$\$RBCD)**

CHART: FG

FIGURE: 5-18

ENTRY POINT: \$\$RBCD

FUNCTION:

- Processes date type statements.
- Prompts for and syntax checks response to a delayed parameter.

INPUT:

- Procedure library
- Response from operator for a delayed parameter

OUTPUT: No-go bit is set if error occurs

EXIT:

- Normal: Call Cycle (\$\$RBCL)
- Error: End of Job Transient (\$\$SPEJ)
Error Logging Routine (\$\$SLSE)

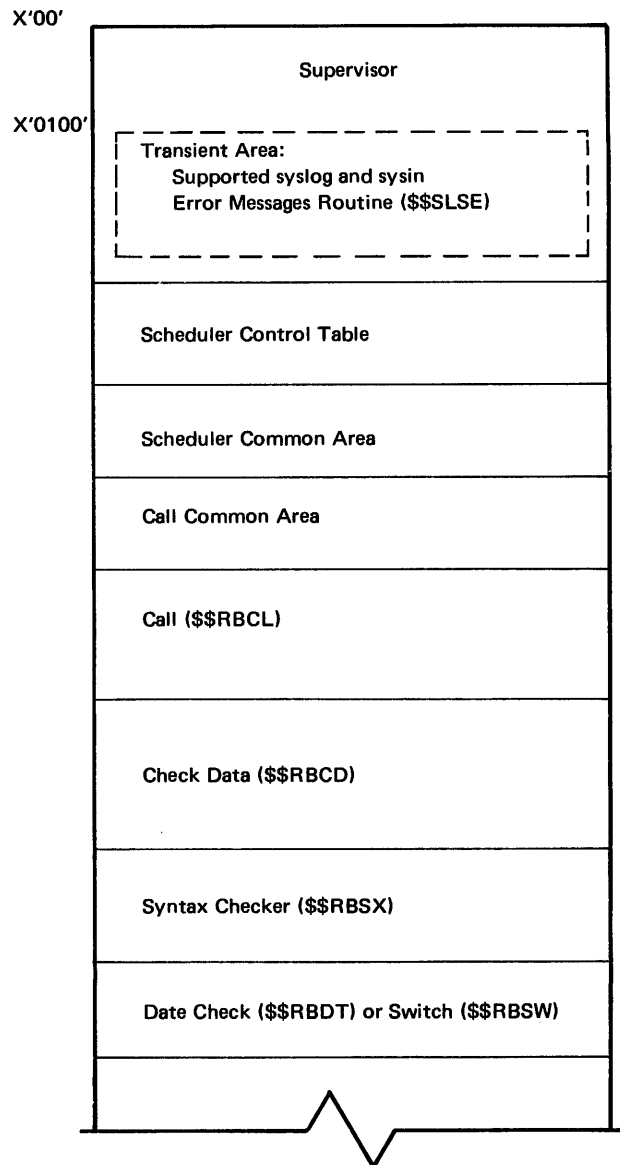


Figure 5-18. Main Storage Map for Data Type Statement Processor (\$\$RBCD)

► **Keyword Type Statement Processor (\$\$RBCK)**

CHART: FH
 FIGURE: 5-19
 ENTRY POINT: \$\$RBCK
 FUNCTION:

- Processes keyword type parameters during a call cycle.
- Prompts for and syntax checks response to a delayed parameter.

INPUT:

- Record from the procedure being called, located in the scheduler common area
- Scheduler Control Table

OUTPUT: No-go bit is set if error occurs

EXIT:

- Normal:
 1. Call Cycle (\$\$RBCL)
 2. Hikey (\$\$RBHI) if required
- Error: End of Job Transient (\$\$SPEJ)
 Error Logging Routine (\$\$SLSE)

X'00'

X'0100'

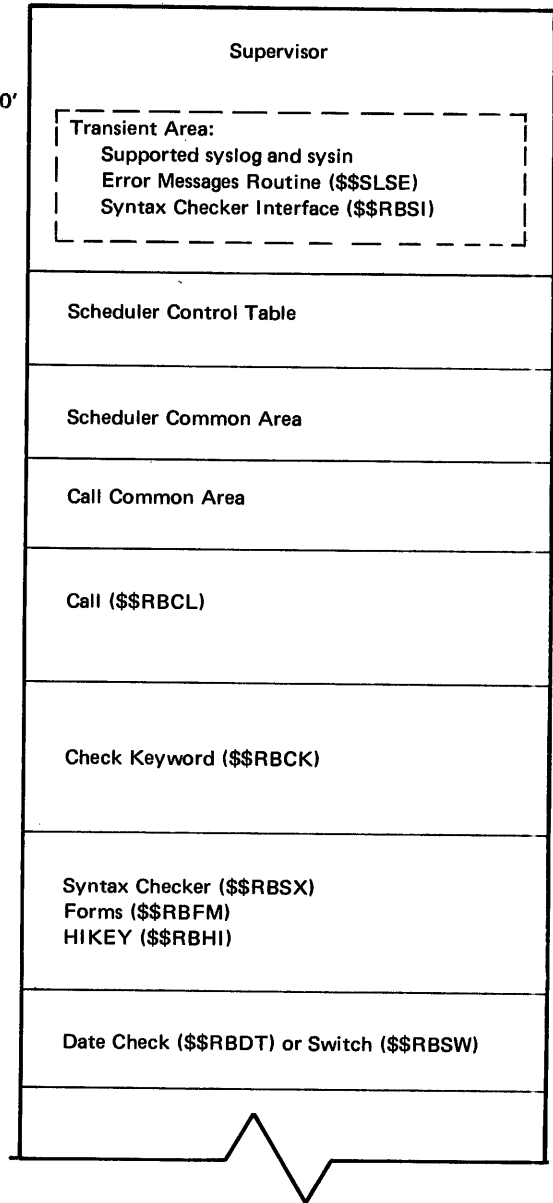


Figure 5-19. Main Storage Map for Keyword Type Statement Processor (\$\$RBCK)

► **Modify (\$\$RBM)**

CHART: FI

FIGURE: 5-20

ENTRY POINT: \$\$RBM

FUNCTION: Allows operator to change and/or add OCL previously given in load, build, or call cycle.

INPUT: Table of responses in the Scheduler Work Area

OUTPUT: Updated responses if requested

EXIT:

- Normal:
 1. For a LOAD, Run (\$\$RBRN) is called
 2. For a BUILD, Library Interface (\$\$RBM1) is called
 3. For a CALL, Run (\$\$RBRN) is called
- Error: End of Job Transient (\$\$SPEJ)

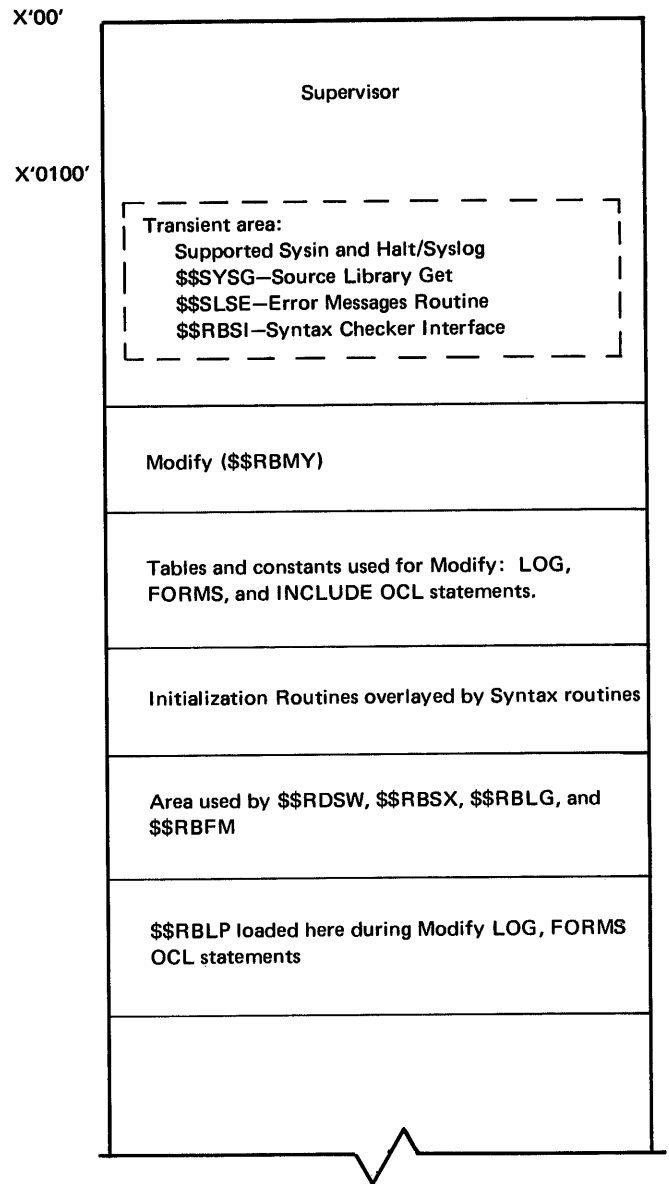


Figure 5-20. Main Storage Map for Modify (\$\$RBM)

► Procedure Library Scheduler Interface (\$\$RBM1)

CHART: FJ

FIGURE: 5-21

ENTRY POINT: \$\$RBM1

FUNCTION: Reads responses from SWA and writes them into the procedure library.

INPUT: Responses in the SWA

OUTPUT: User's created procedure library entry

EXIT:

- Normal: \$\$RBM2
- Error: End of Job Transient (\$\$SPEJ)

► Utility OCL Processor (\$\$RBM2)

CHART: FK

FIGURE: 5-21

ENTRY POINT: \$\$RBM2

FUNCTION:

- During a build cycle, includes and modifies utility control statements and places them with the rest of the procedure in the source library
- During a call cycle, displays utility control statements in the source library and allows them to be modified before the called program is loaded and executed

INPUT:

- Utility control statements read from the source library during a call cycle
- Modifications entered by the operator during a build cycle

OUTPUT:

- Utility control statements are placed in the SWA during a call cycle
- Utility control statements are placed in the source library during a build cycle

EXIT:

- Normal: Initiator Program Setup (\$\$INPS)
- Error: The End of Job Transient (\$\$SPEJ) is called if an error occurs or during a Build cycle

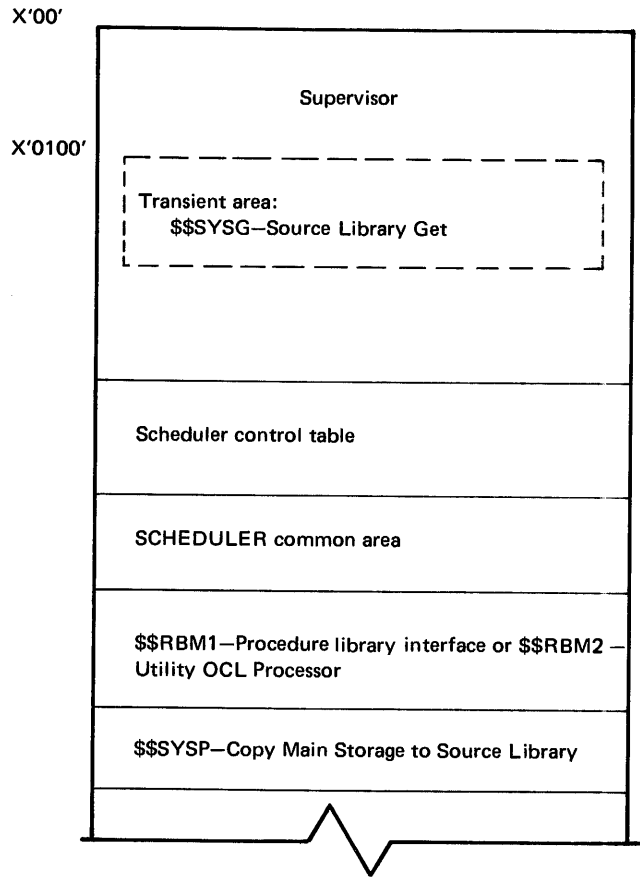


Figure 5-21. Main Storage Map for Procedure Library Scheduler Interface (\$\$RBM1) and Utility OCL Processor (\$\$RBM2)

► Run Processor (\$\$RBRN)

CHART: FL

FIGURE: 5-22

ENTRY POINT: \$\$RBRN

FUNCTION: Writes file information to SWA in the following format:

70	CC	L	Response	L	Response	L	etc.	00
----	----	---	----------	---	----------	---	------	----

or, for multivolume files,

CC	L	Response	L	Response	etc.	FE
----	---	----------	---	----------	------	----

- 70 — denotes the beginning of a files statement
- CB — control bytes for files information
- L — length of the response
- 00 — denotes the end of a files statement
- FE — denotes the end of a multivolume file statement

Disk File Keywords

Keyword	Control Bytes
NAME	11
UNIT*	31
PACK*	41
LABEL	21
RECORDS*	71
TRACKS*	61
LOCATION*	81
RETAIN	51
DATE	91
HIKEY	A2

*If the response is multi-volume, the numeric portion of the code is 2 instead of 1.

INPUT: Responses read from the SWA

OUTPUT: Encoded information written to the SWA

EXIT:

- Normal:
 1. Initiator Program Setup (\$\$INPS)
 2. Procedure Library Scheduler Interface (\$\$RBM1) if procedure called contains utility control statements
- Error: End of Job Transient (\$\$SPEJ)

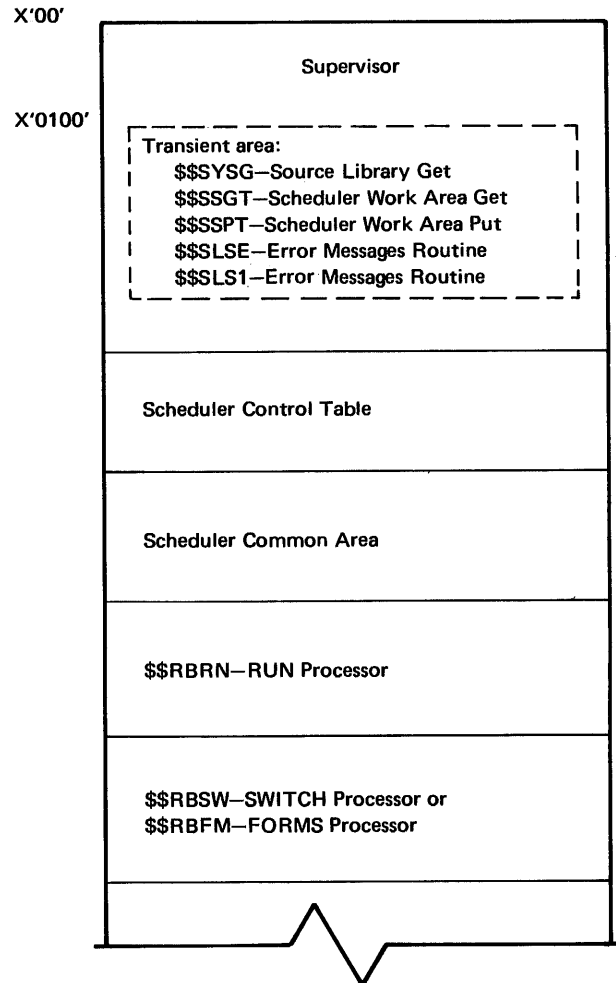


Figure 5-22. Main Storage Map for Run Processor (\$\$RBRN)

► Interaction Routine (\$\$RBIN)

CHART: FM

FIGURE: 5-23

ENTRY POINT: \$\$RBIN

FUNCTION: Completes building of the prompt buffer.

INPUT:

- Scheduler Control Table
- Operator responses

OUTPUT: Prompts and error messages

EXIT:

- Normal: Next sequential instruction of the calling program, or next sequential instruction + 3 when the operator signals completion of the OCL sequence
- Error: Next sequential instruction + 6 of the calling program, or End of Job Transient (\$\$SPEJ) if operator response is /* or /&

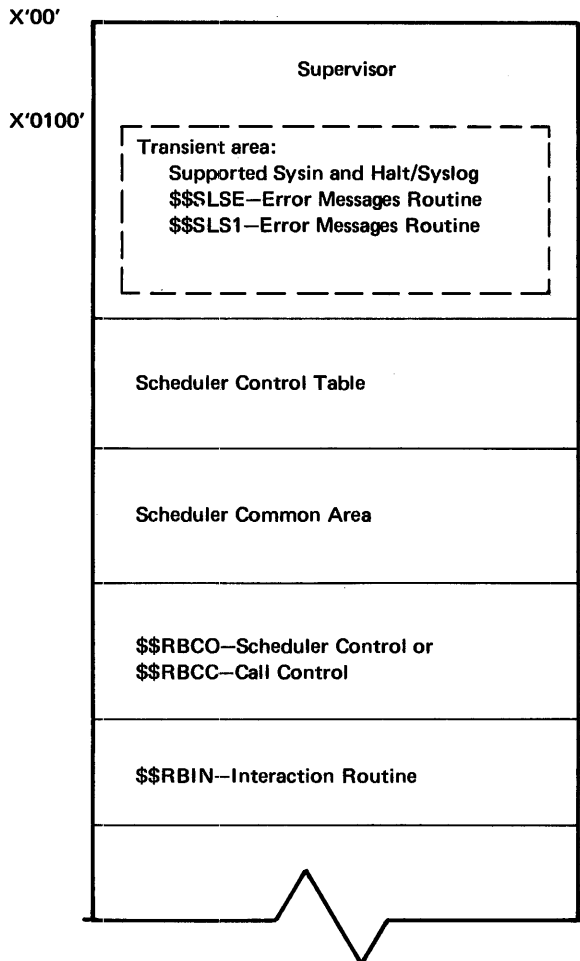


Figure 5-23. Main Storage Map for Interaction Routine (\$\$RBIN)

► Date/Switch (\$\$RBDS)

CHART: FN

FIGURE: 5-24

ENTRY POINT: \$\$RBDS

FUNCTION:

- Prompts for and syntax checks the Date and Switch statements for the conversational scheduler during load and build cycles.
- Places correct statements in the SWA.

INPUT: Operator responses

OUTPUT: Statements placed in the SWA

EXIT:

- Normal: Return to Scheduler Control (\$\$RBCO)
- Error: End of Job Transient (\$\$SPEJ) if operator responds with /* or /&

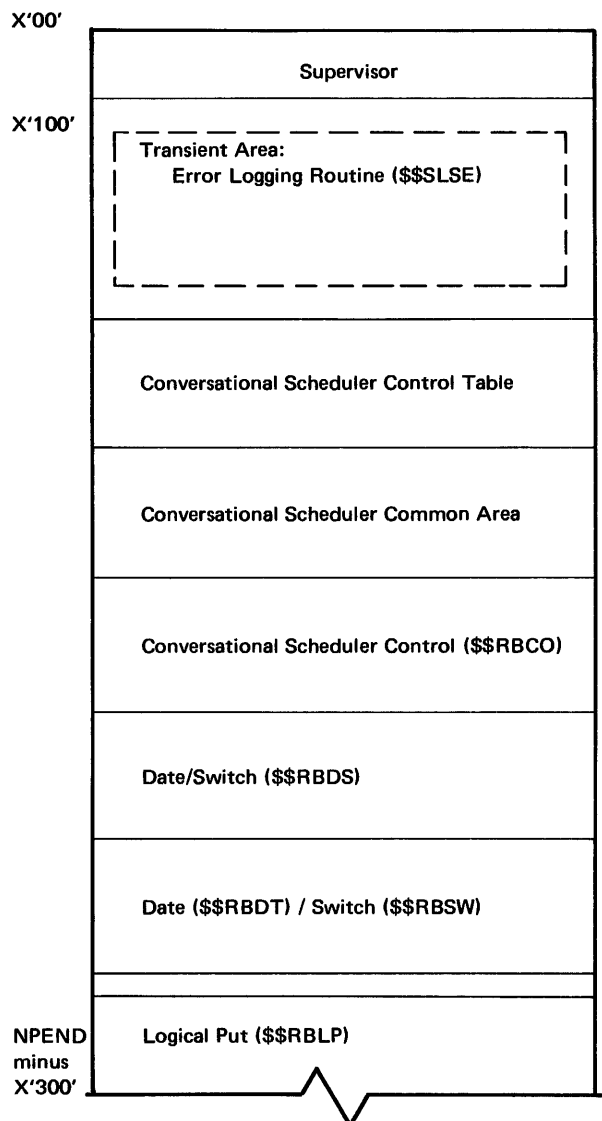


Figure 5-24. Main Storage Map for Date/Switch (\$\$RBDS)

► **Switch Control Statement Processor (\$\$RBSW)**

CHART: FO

FIGURE: 5-25

ENTRY POINT: \$\$RBSW

FUNCTION: Sets UPSI switches

Note: The UPSI switches are located within the program level communication region.

INPUT: The reader/interpreter work area contains the beginning and end addresses of the record to be scanned.

OUTPUT: The UPSI switches are set to reflect the SWITCH control statement.

EXIT:

- Normal: Calling routine
- Error: Calling routine (NSI+3)

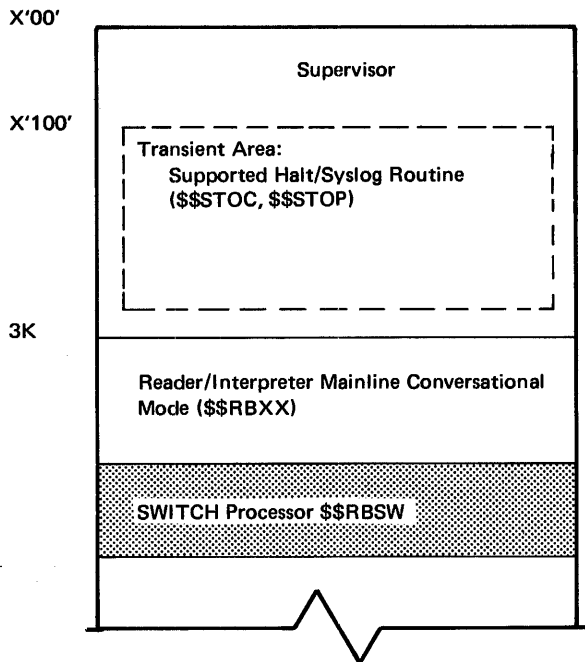


Figure 5-25. Main Storage Map for SWITCH Control Statement Processor (\$\$RBSW)

► **Date Scan (\$\$RBDT)**

CHART: FP

FIGURE: 5-26

ENTRY POINT: \$\$RBDT

FUNCTION: Ensures that date parameters are syntactically correct.

INPUT: XR2 points to a 7-byte parameter list containing in the following order:

1. A flag byte to receive an error code
2. Two bytes containing the starting address of the input buffer
3. Two bytes containing the ending address of the input buffer
4. Two bytes containing the address of the 6-byte area in which to place a good date

OUTPUT: A syntactically correct date is placed in the designated 6 bytes, or an error code is returned. The error codes are:

- X'40' = good date
- X'81' = no date found
- X'82' = date too long/short
- X'83' = misplaced/double delimiter
- X'84' = invalid character

EXIT:

- Normal: Next sequential instruction of the calling program
- Error: Next sequential instruction + 3 of the calling program

► **Log (\$\$RBLG)**

CHART: FQ

FIGURE: 5-26

ENTRY POINT: \$\$RBLG

FUNCTION:

- Prompts for new logging device.
- Allocates supported device.
- Finds required halt/syslog module and writes C/S of the module in the communications area.
- Finds required sysin module and writes C/S of the module in the communications area.

INPUT:

- Device response
- Configuration record

OUTPUT: The C/S of the new halt/syslog and sysin modules are written in the communications area.

EXIT:

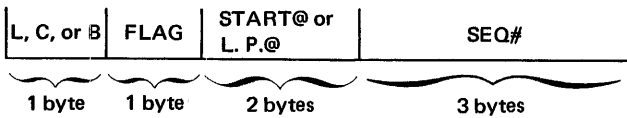
- Normal: Next sequential instruction of the calling program
- Error: End of Job Transient (\$\$SPEJ) is called if an error occurs or if the operator responds with /* or /&.

► FORMS (\$\$RBFM)

CHART: FR
 FIGURE: 5-26
 ENTRY POINT: \$\$RBFM
 FUNCTION:

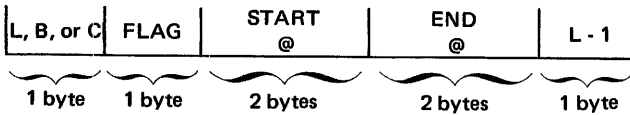
- Prompts for the device and the number of lines per page.
- Writes the new device and number of lines per page in the system communication region.

INPUT: XR2 points to the following parameter list:



- L, B, or C—load, build, or call
- FLAG—indicates function to be performed
 - X'80' — update communications area
 - X'40' — prompt forms
 - X'20' — syntax check device
 - X'10' — syntax check lines
- START@—starting address of buffer containing response. Required for updating and syntax check.
- L.P.@—SCA address in core of logical put routine. Required only during prompt function.
- SEQ#—Current sequence number in use. Required only for prompt function.

OUTPUT: The same parameter list pointed to by XR2 is returned to the caller containing:



- FLAG—00 indicates successful
 - X1 indicates error found
- START@—Start @ of buffer
- END@—Address of rightmost byte of response
- L-1—Length minus one of response

EXIT:

- Normal: Return to next sequential instruction of caller.
- Error: End of job is called if error in sysin.

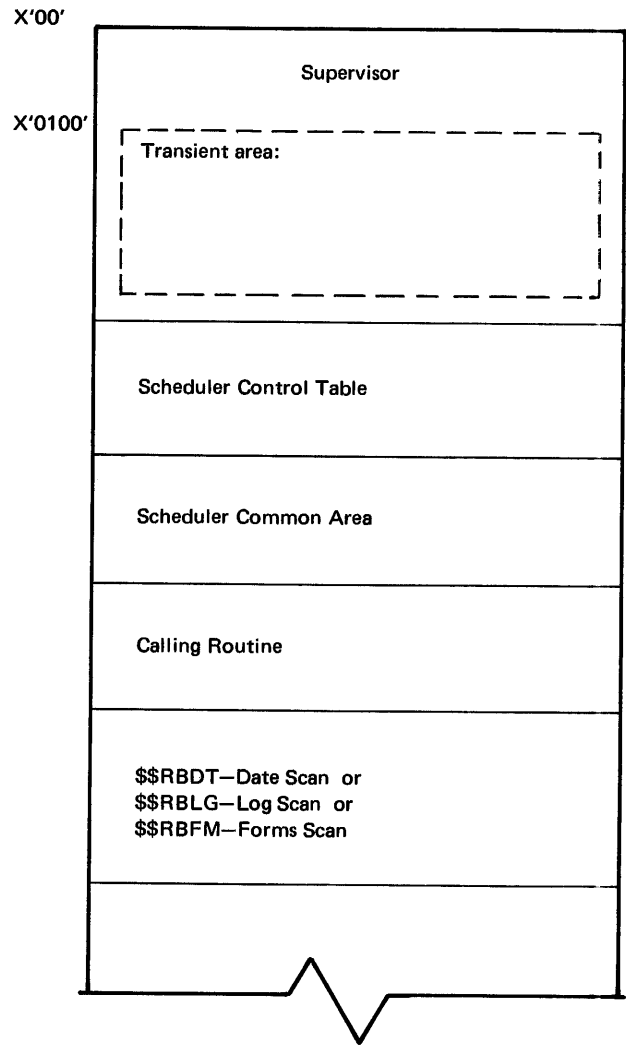


Figure 5-26. Main Storage Map for Date Scan (\$\$RBDT), Log (\$\$RBLG), Forms (\$\$RBFM)

► **HIKEY (\$\$RBHI)**

CHART: FS

FIGURE: 5-27

ENTRY POINT: \$\$RBHI

FUNCTION: Prompts and processes responses for HIKEYs during LOAD, BUILD, or CALL.

INPUT: OCL images or keyboard responses

OUTPUT: Correct responses are written to the SWA

EXIT:

- Normal: Next sequential instruction of calling program
- Error: End of Job Transient (\$\$SPEJ)

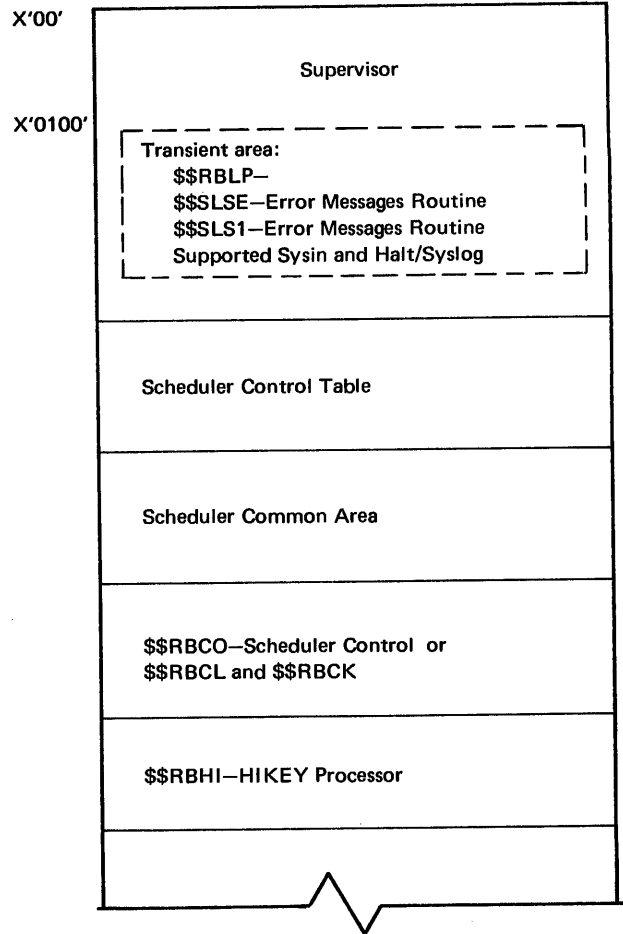


Figure 5-27. Main Storage Map for HIKEY Processor (\$\$RBHI)

3. READER/INTERPRETER FOR DISK SYSTEM

► Reader/Interpreter Mainline (\$\$RDML)

CHART: GA
 FIGURE: 5-28
 ENTRY POINT: \$\$RDML
 FUNCTION:

- Reads an OCL statement.
- Performs a syntax check for recognizable control statements.
- Performs a syntax check for valid comments.
- Performs a syntax check for valid keyword.
- Determines whether a statement is a procedure.
- Encodes keyword parameters on OCL statements.
- Passes control to the appropriate control card processor.

INPUT: Control statements from the supported Sysin device

OUTPUT: Parameters for the control card processing routines are placed in the reader/interpreter work area (RDIWA).

EXIT: None

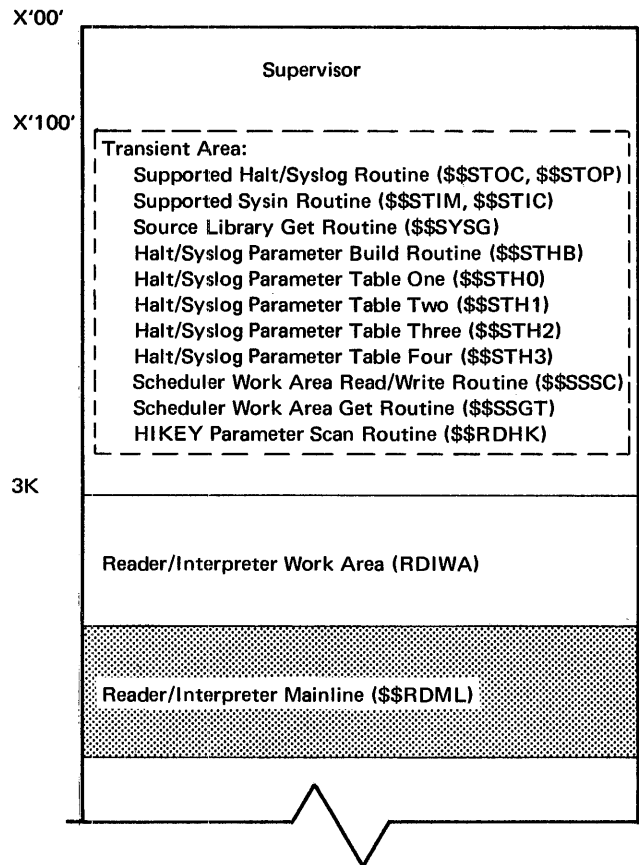


Figure 5-28. Main Storage Map for the Reader/Interpreter Mainline Routine (\$\$RDML)

► // LOAD Control Card Processor (\$\$RDLD)

CHART: GB
 FIGURE: 5-29
 ENTRY POINT: \$\$RDLD
 FUNCTION:

- Ensures syntactical correctness of // LOAD control card by passing control to \$\$RDS3.
- Processes any // LOAD override parameters.

INPUT: Register 2 contains the address parameters of the area to be scanned.

OUTPUT: The program name and its resident unit ID are loaded into the program level communication area at label NPCC

EXIT: Calling routine (\$\$RDML), which then reads another card.

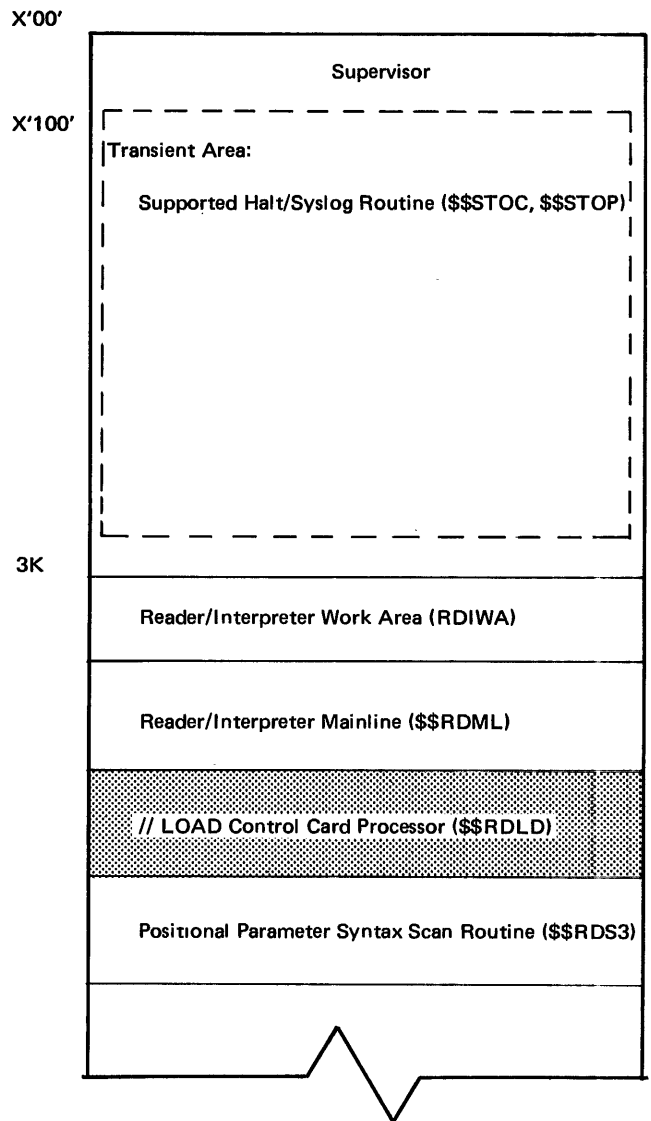


Figure 5-29. Main Storage Map for the // LOAD Control Card Processor (\$\$RDLD)

► // RUN Control Card Processor (\$\$RDRN)

CHART: GC

FIGURE: 5-30

ENTRY POINT: \$\$RDRN

FUNCTION:

- Closes the initiator table containing the encoded // FILE records.
- Changes the system mode from INTRA to INTER.
- Alters the input source for OCL statements from procedure to procedure additions (contained in the scheduler work area).
- Dequeues a nested procedure during INTER mode.

INPUT: None

OUTPUT:

- The system mode, indicated in the program level communication region at label NPSCH2, is changed from INTRA to INTER and the initiator table is closed.
- A procedure is dequeued if this routine was called because of nested procedures during INTER mode.

EXIT:

- Normal: Initiator Program Setup—Phase One (\$\$INPS) unless \$\$RDRN was called during INTER mode to dequeue a nested procedure. In this case, control is returned to the Reader/Interpreter Mainline (\$\$RDML).
- Error: End of Job transient (\$\$SPEJ)

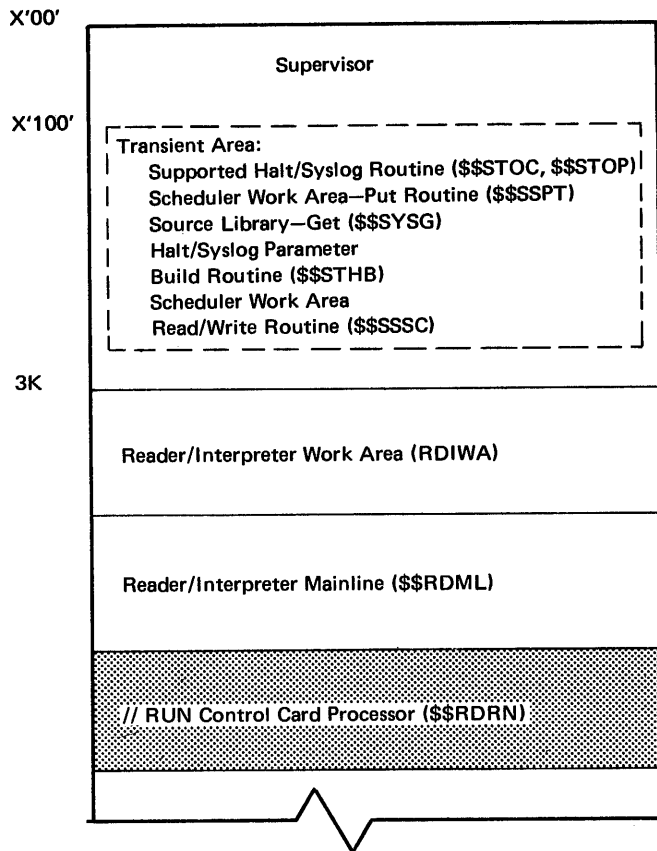


Figure 5-30. Main Storage Map for the // RUN Control Card Processor (\$\$RDRN)

► // FILE Control Card Processor (\$\$RDFL)

CHART: GD

FIGURE: 5-31

ENTRY POINT: \$\$RDFL

FUNCTION:

- Insure the correctness of the // FILE control card key-word.
- Loads the information found on the // FILE control card into the scheduler work area.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning address of the encoded // FILE record.

OUTPUT: The encoded // FILE statement is loaded into the initiator table (INITAB) located within the SWA.

EXIT: \$\$RDMK is fetched.

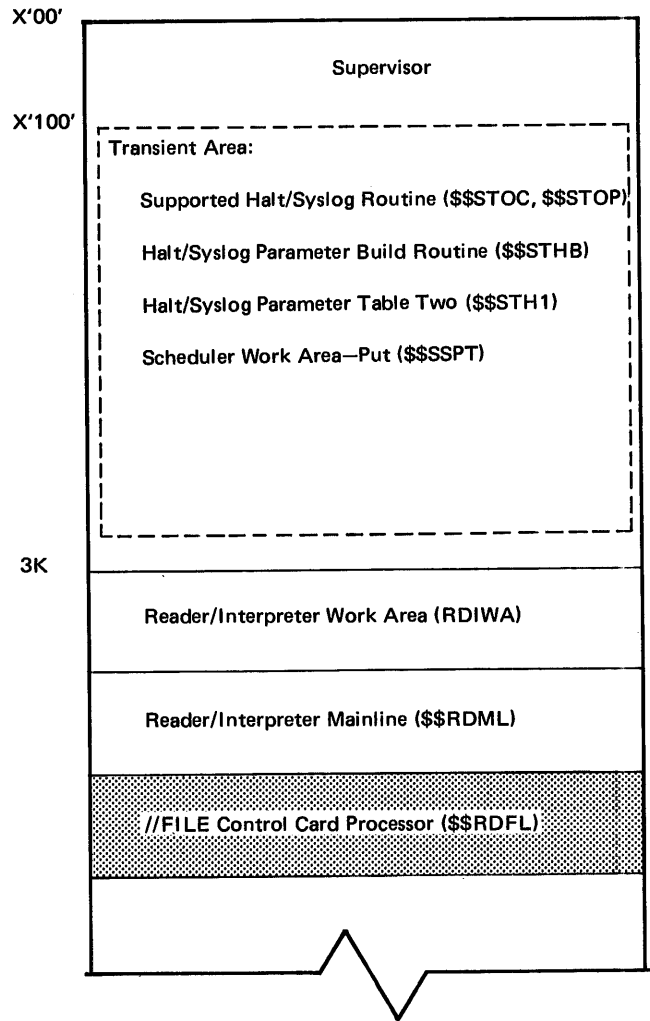


Figure 5-31. Main Storage Map for the // FILE Control Card Processor (\$\$RDFL)

► // DATE Control Card Processor (\$\$RDDT)

CHART: GE

FIGURE: 5-32

ENTRY POINT: \$\$RDDT

FUNCTION:

- Insures the syntactical correctness of the // DATE control card.
- Processes the date parameter into either one of two 6-byte formats:
 - MM DD YY (Month day year) Domestic.
 - DD MM YY (Day month year) European.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the record to be scanned.

OUTPUT:

- Normal: The 6-byte formatted area is moved to either (1) the program level communication region (INTRA mode) or (2) the scheduler work area and the program level communication region (IPL mode).
- Error: The statement is ignored, the controlled cancel indicator is turned on, and control is returned to the calling routine.

EXIT: Calling routine

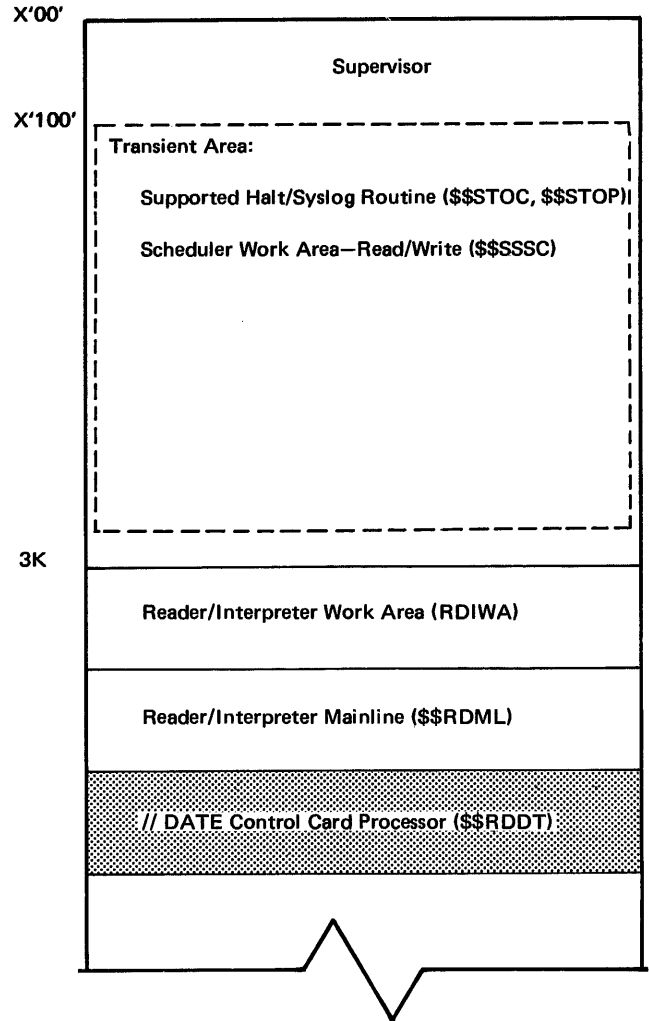


Figure 5-32. Main Storage Map for the // DATE Control Card Processor (\$\$RDDT)

► // SWITCH Control Card Processor (\$\$RDSW)

CHART: GF
 FIGURE: 5-33
 ENTRY POINT: \$\$RDSW
 FUNCTION:

- Checks the syntactical correctness of the // SWITCH control card.
 - If the control card is valid, the corresponding UPSI switches are set.
- Note:* The UPSI switches are located within the program level communication region.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the record to be scanned.

OUTPUT: The UPSI switches are set to reflect the // SWITCH control card.

- EXIT:
- Normal: Reader/Interpreter Mainline (\$\$RDML)
 - Error: Supported Halt/Syslog routine

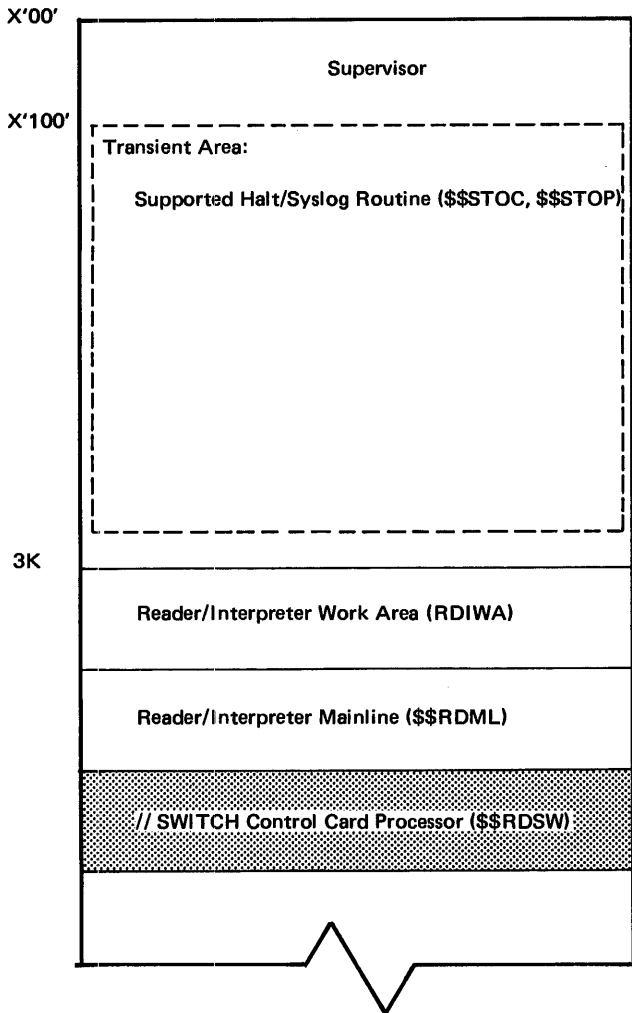


Figure 5-33. Main Storage Map for the // SWITCH Control Card Processor (\$\$RDSW)

► // IMAGE Control Card Processor (\$\$RDIM)

CHART: GG
 FIGURE: 5-34
 ENTRY POINT: \$\$RDIM
 FUNCTION:

- Ensures the syntactical correctness of the // IMAGE record.
- Modifies the chain image area to reflect different print chain configurations.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end address of the area to be scanned by this processor.

OUTPUT: The modified chain image is loaded into the nucleus chain image area (location X'400').

EXIT: Calling routine

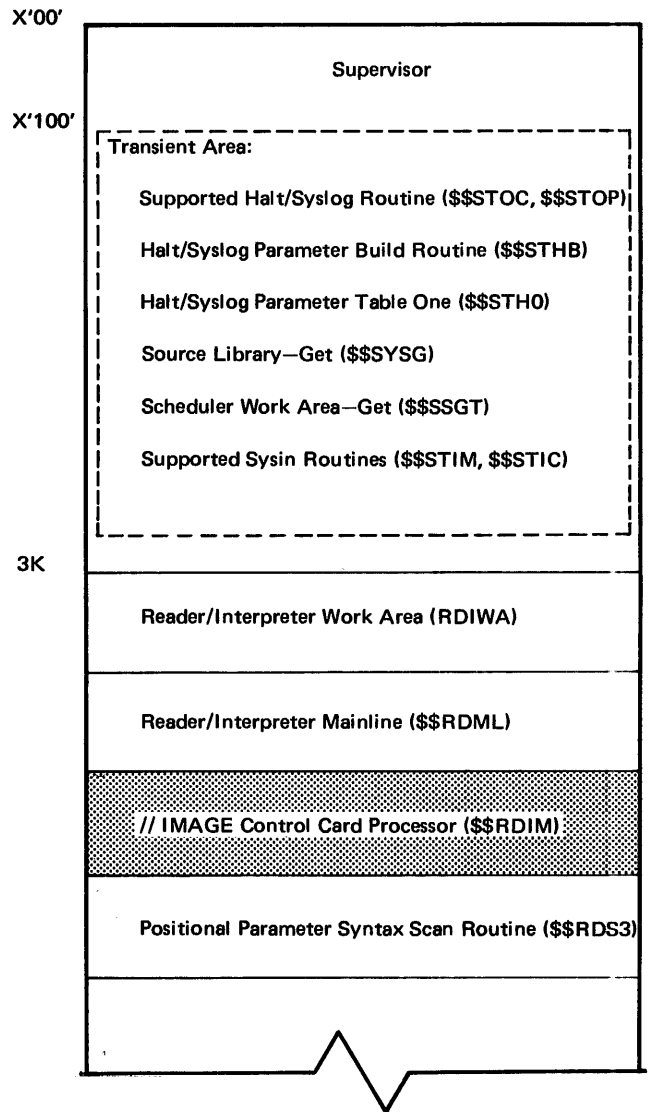


Figure 5-34. Main Storage Map for the // IMAGE Control Card Processor (\$\$RDIM)

► // COMPILE Control Card Processor (\$\$RDCM)

CHART: GH
 FIGURE: 5-35
 ENTRY POINT: \$\$RDCM
 FUNCTION:

- Syntactically checks the encoded parameters.
- Loads the syntactically correct source name and unit into the index sector of the SWA.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the encoded parameters.

OUTPUT: The source name and unit are placed in the index sector of the SWA.

EXIT: Calling routine

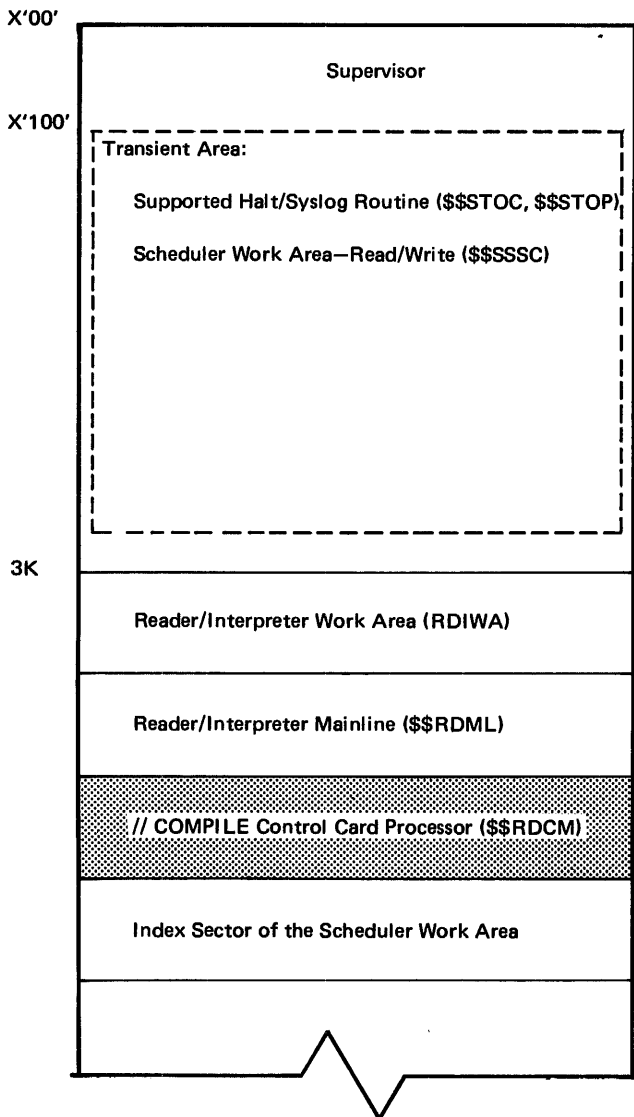


Figure 5-35. Main Storage Map for the // COMPILE Control Card Processor (\$\$RDCM)

► // FORMS Control Card Processor (\$\$RDFM)

CHART: GI
 FIGURE: 5-36
 ENTRY POINT: \$\$RDFM
 FUNCTION:

- Ensures the syntactical correctness of the // FORMS record.
- Alters the number of printable lines per page on the system printer as specified on the // FORMS record.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning address of the encoded parameter.

OUTPUT: A 1-byte number defining the number of lines to be printed per page. This number is located in either the right tractor (NCRPSZ) or the left tractor (NCLPSZ) byte of the system communication region.

EXIT: Calling program

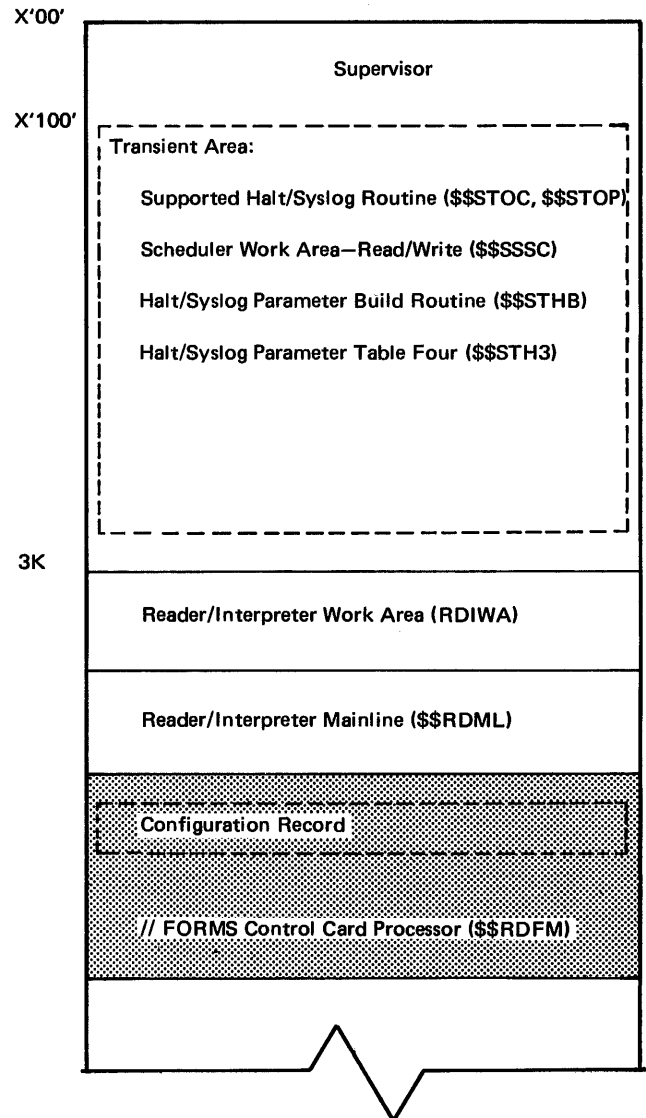


Figure 5-36. Main Storage Map for the // FORMS Control Card Processor (\$\$RDFM)

► // LOG Control Card Processor (\$\$RDLG)

CHART: GJ

FIGURE: 5-37

ENTRY POINT: \$\$RDLG

FUNCTION:

- Ensures the syntactical correctness of the // LOG control card.
- Indicates the system's logging device at label NCSLOG within the system communication region.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the record to be scanned.

OUTPUT:

- The C/S address of the Halt/Syslog module that supports the requested device is loaded into the system communication region.
- The device is allocated to the requested program level.

EXIT: Reader/Interpreter Mainline (\$\$RDML)

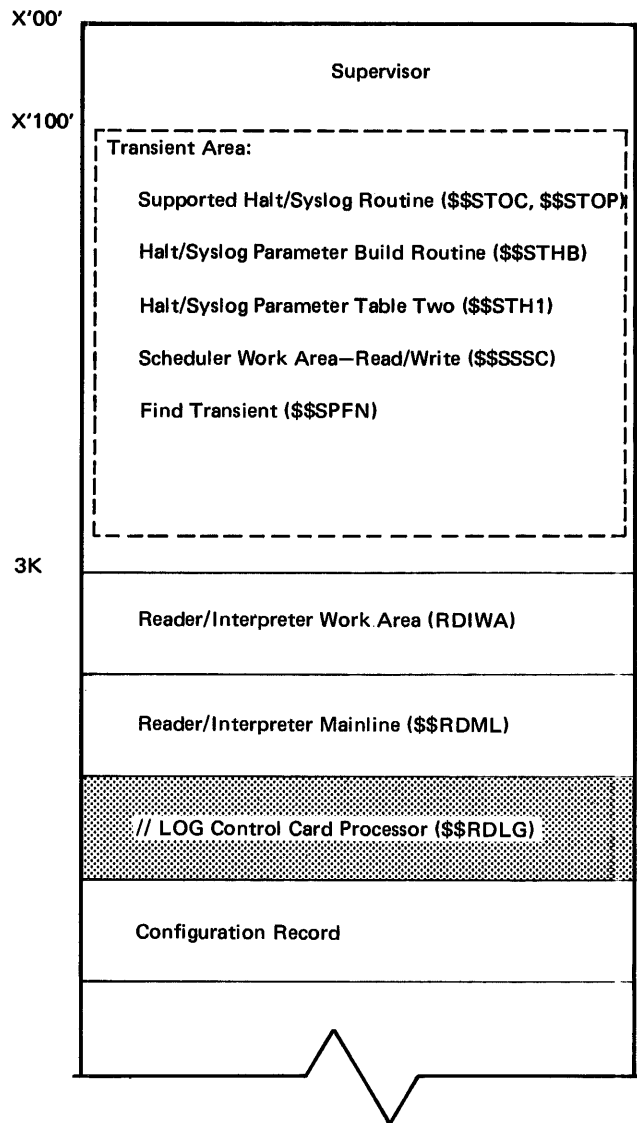


Figure 5-37. Main Storage Map for the // LOG Control Card Processor (\$\$RDLG)

► // PAUSE Control Card Processor (\$\$RDPS)

CHART: GK
 FIGURE: 5-38
 ENTRY POINT: \$\$RDPS
 FUNCTION: The // PAUSE control card issues an advance program level (APL) HALT.
 INPUT: None
 OUTPUT: None
 EXIT: Reader/Interpreter Mainline (\$\$RDML)

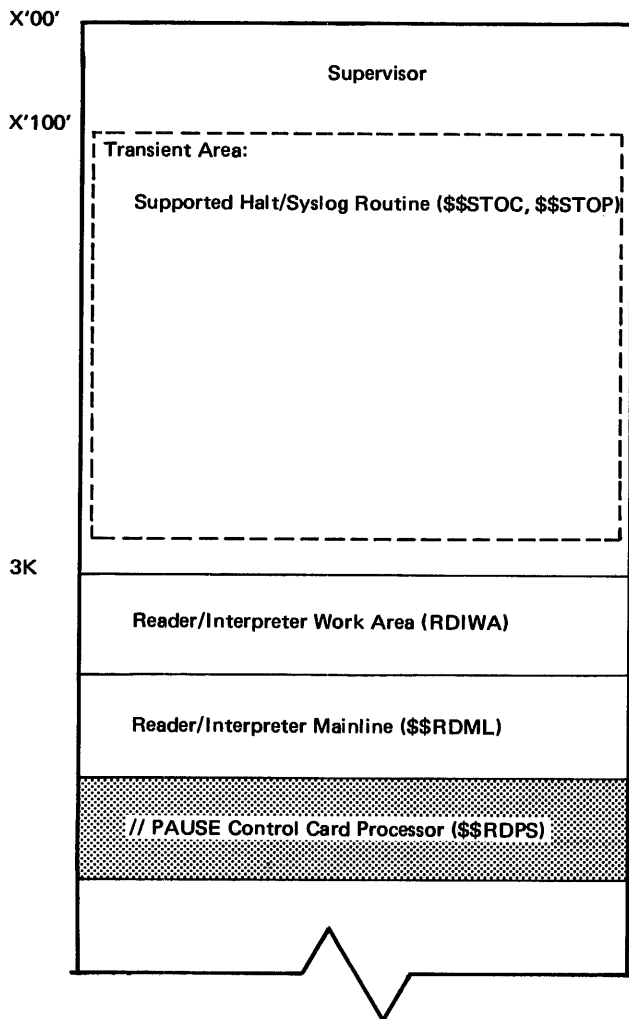


Figure 5-38. Main Storage Map for the // PAUSE Control Card Processor (\$\$RDPS)

► // HALT, // NOHALT Control Card Processor (\$\$RDHN)

CHART: GL
 FIGURE: 5-39
 ENTRY POINT: \$\$RDHN
 FUNCTION:
 - The // HALT control card indicates that normal program end of job should be performed.
 - The // NOHALT control card indicates that normal end of job should *not* be performed.
 INPUT: None
 OUTPUT: The end of job indicator, located at label NPSCH3 within the program level communication region, is set to indicate the type of halt control card.
 EXIT: Reader/Interpreter Mainline (\$\$RDML)

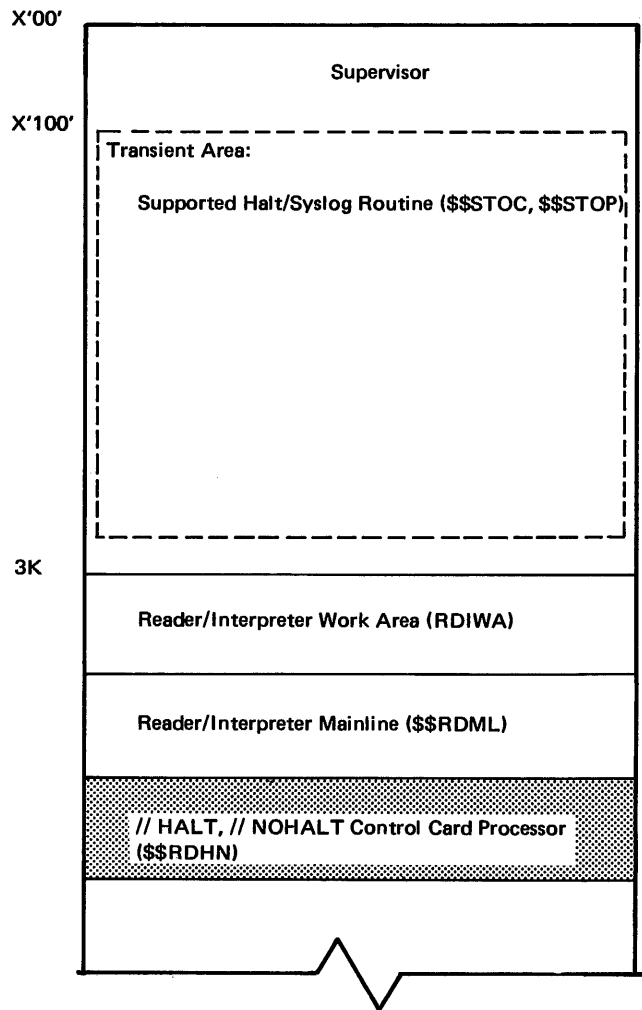


Figure 5-39. Main Storage Map for the // HALT, //NOHALT Control Card Processor (\$\$RDHN)

► // READER Control Card Processor (\$\$RDRR)

CHART: GM

FIGURE: 5-40

ENTRY POINT: \$\$RDRR

FUNCTION:

- Ensures that the specified device is valid.
- Indicates the specified device as the system input device.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the area to be scanned.

OUTPUT:

- The system input device is indicated in the program level communication region at label NPSYSI.
- Moves the disk address of the supporting input module into the program level communication region.

EXIT: Reader/Interpreter Mainline (\$\$RDML)

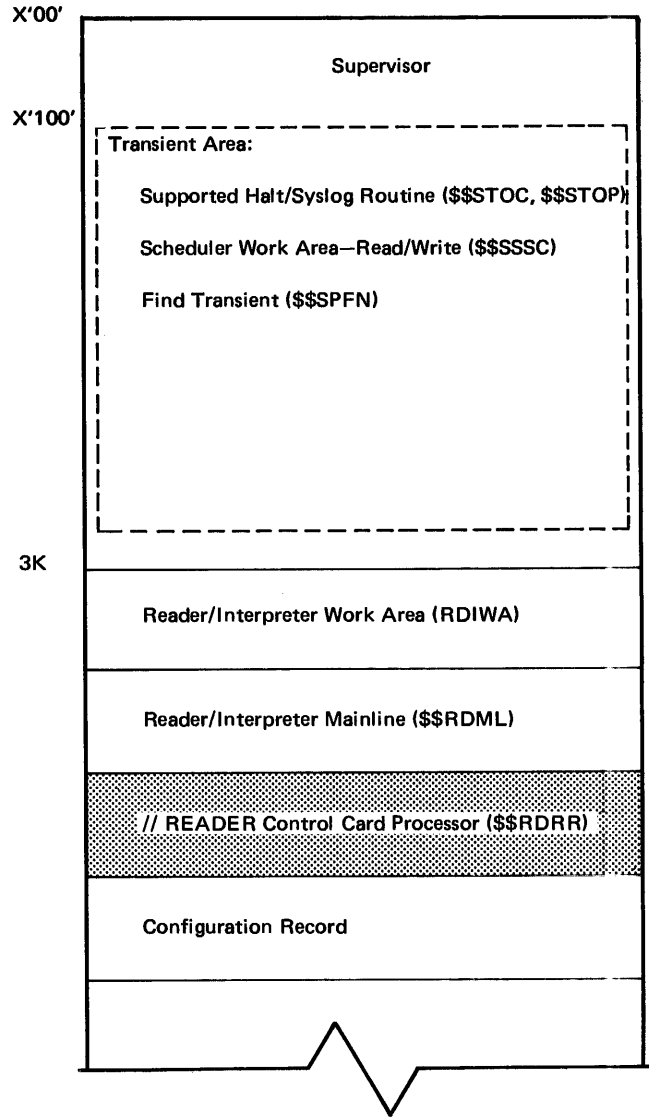


Figure 5-40. Main Storage Map for the // READER Control Card Processor (\$\$RDRR)

► // PARTITION Control Card Processor (\$\$RDPN)

CHART: GN

FIGURE: 5-41

ENTRY POINT: \$\$RDPN

FUNCTION:

- Ensures the syntactical correctness of the specified parameter.
- Determines if the requested partition change can be supported.
- Indicates the maximum amount of main storage allocated to the second program level.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end addresses of the area to be scanned.

OUTPUT: The indicator NCSCH2, located within the system communication region, is updated to reflect the number of sectors allocated to program level two.

EXIT: Reader/Interpreter Mainline (\$\$RDML)

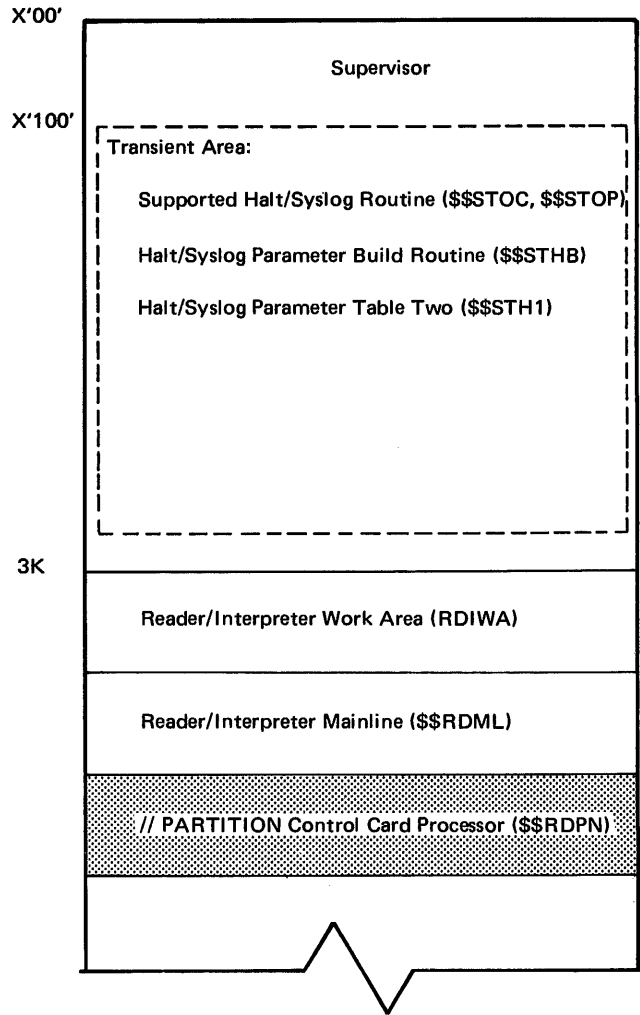


Figure 5-41. Main Storage Map for the // PARTITION Control Card Processor (\$\$RDPN)

► // CALL Control Card Processor (\$\$RDCL)

CHART: GO

FIGURE: 5-42

ENTRY POINT: \$\$RDCL

FUNCTION:

- Ensures the syntactical correctness of the // CALL control card.
- Locates the specified procedure.
- Reads procedure additions and/or override statements into the SWA.
- Builds the \$\$\$SYSG parameter list so \$\$RDML can read from the procedure.
- Queues any previously called procedure in the index sector of the SWA.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end address of the area to be scanned for parameters.

OUTPUT:

- The parameter list used to read a procedure is placed in the reader/interpreter work area (RDIWA).
- The procedure indicator located at label NPSCH3 within the program level communication region is turned on.
- Any previously called procedure is queued in the index sector of the SWA.

EXIT: Reader/Interpreter Mainline (\$\$RDML)

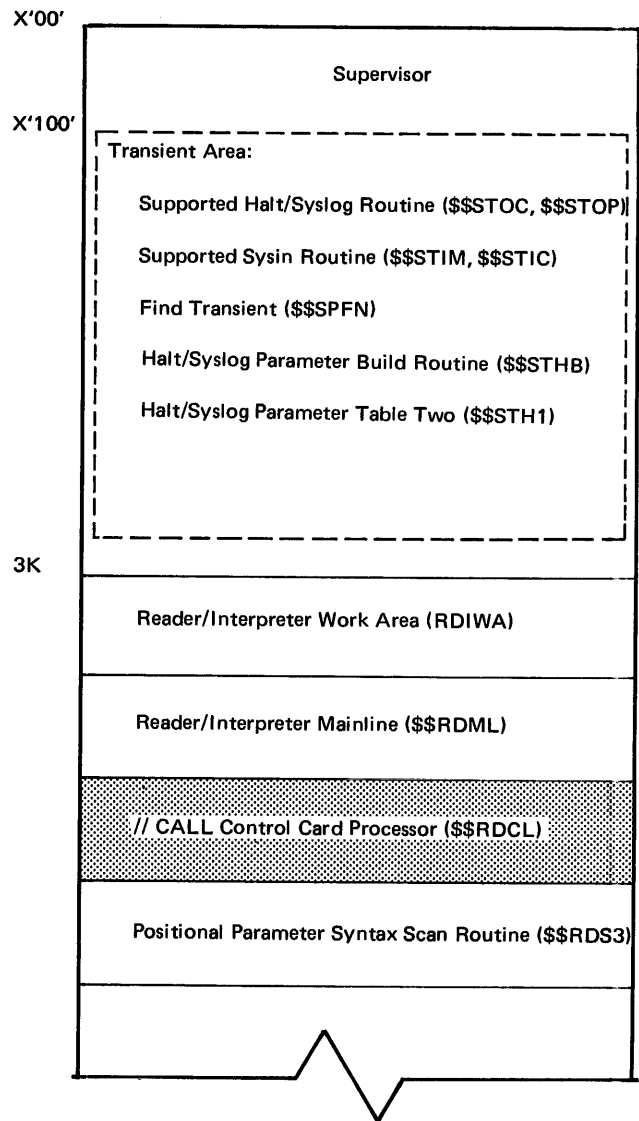


Figure 5-42. Main Storage Map for the // CALL Control Card Processor (\$\$RDCL)

► Positional Parameter Syntax Scan Routine (\$\$RDS3)

CHART: GP

FIGURE: 5-43

ENTRY POINT: \$\$RDS3

FUNCTION: This routine scans the specified area for the program name and unit. This information is used to build a 2-byte table (NPCS). If either the program name or the unit is not specified, the corresponding table entry is left blank.

INPUT: The reader/interpreter work area (RDIWA) contains the beginning and end address of the area to be scanned.

OUTPUT: A return code indicating:

- An invalid program name
- An invalid unit
- Invalid parameters
- Successful completion

EXIT: Calling routine

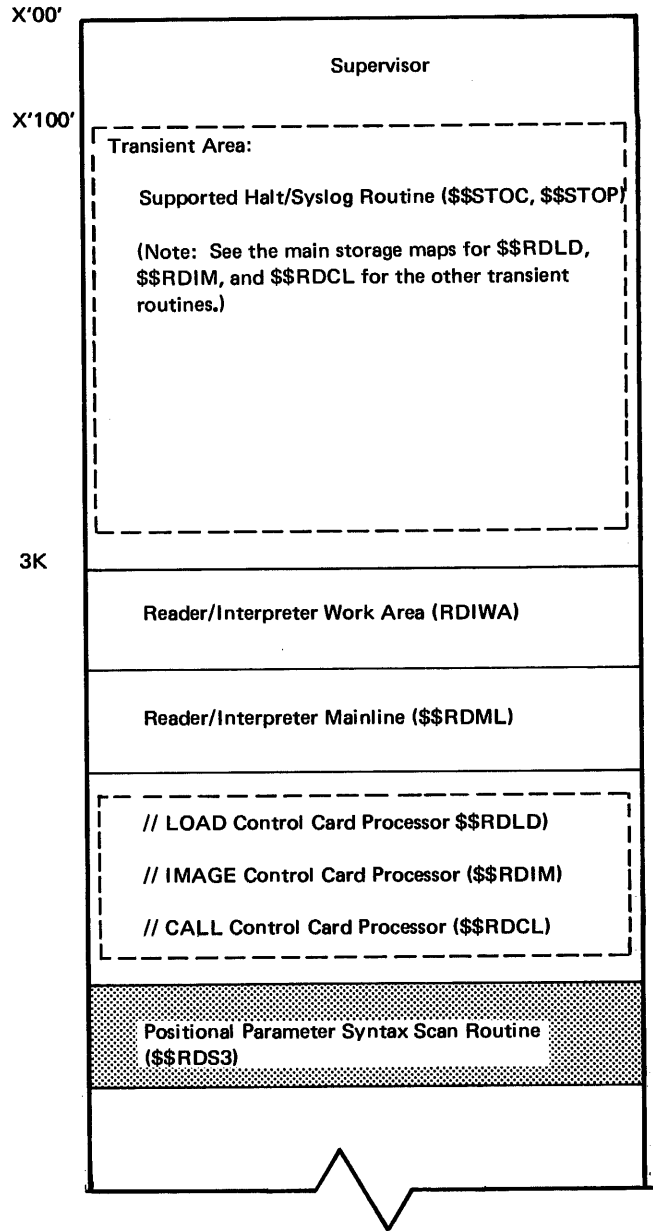


Figure 5-43. Main Storage Map for the Positional Parameter Syntax Scan Routine (\$\$RDS3)

► **FILE Diagnostics and Merge Keyword Parameters (\$\$RDMK)**

CHART: GQ

FIGURE: 5-44

ENTRY POINT: \$\$RDMK

FUNCTION:

- Performs the end of record diagnostics for the // FILE record.
- Merges the procedure and the override // FILE records into the initiator table.

INPUT:

- The reader/interpreter work area (RDIWA) contains all information needed to ensure that the correct // FILE record was provided.
- The initiator table contains either:
 1. The encoded // FILE record, if OCL records are being read from the system input device.
 2. The encoded override // FILE record, while a procedure is being merged.
- The format one area of the SWA contains the encoded // FILE record that was read from the procedure.

OUTPUT:

- Error messages if any errors are encountered.
- The merged // FILE record is loaded into the initiator table of the SWA. (For example: procedure // FILE + override // FILE is loaded into the initiator table.)

EXIT: Reader/Interpreter Mainline (\$\$RDML)

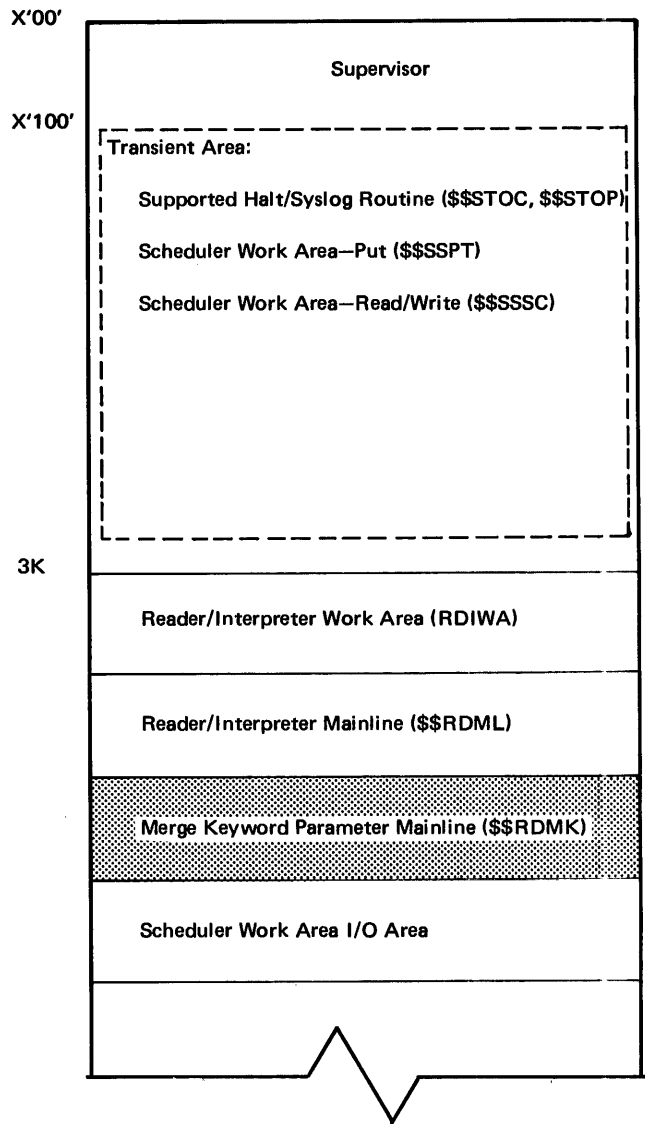


Figure 5-44. Main Storage Map for File Diagnostics and Merge Keyword Parameters Routine (\$\$RDMK)

4. INITIATOR

► Initiator Allocate—Phase One Routine (\$\$INA1)

CHART: HA

FIGURE: 5-45

ENTRY POINT: \$\$INA1

FUNCTION: Calculates the required space needed in the SWA for format one images that have DTFs defined.

INPUT: Compressed DTFs with the following format:

DISPLACEMENT	LENGTH	CONTENTS
0	1	Device Q byte
1	1	UPSI indicators
3	2	Attributes of the file
5	2	Record length
13	8	File name
14	1	Key length of index files
15	1	End of DTF compression byte indicated by X'FF'

OUTPUT: The space requirement fields in the format one images updated and placed in the SWA.

EXIT:

- Normal: Terminator Allocate—Phase Two (\$\$INA2). If no new files are being allocated, control is returned to \$\$INAT.
- Error: Supported Halt/Syslog routine

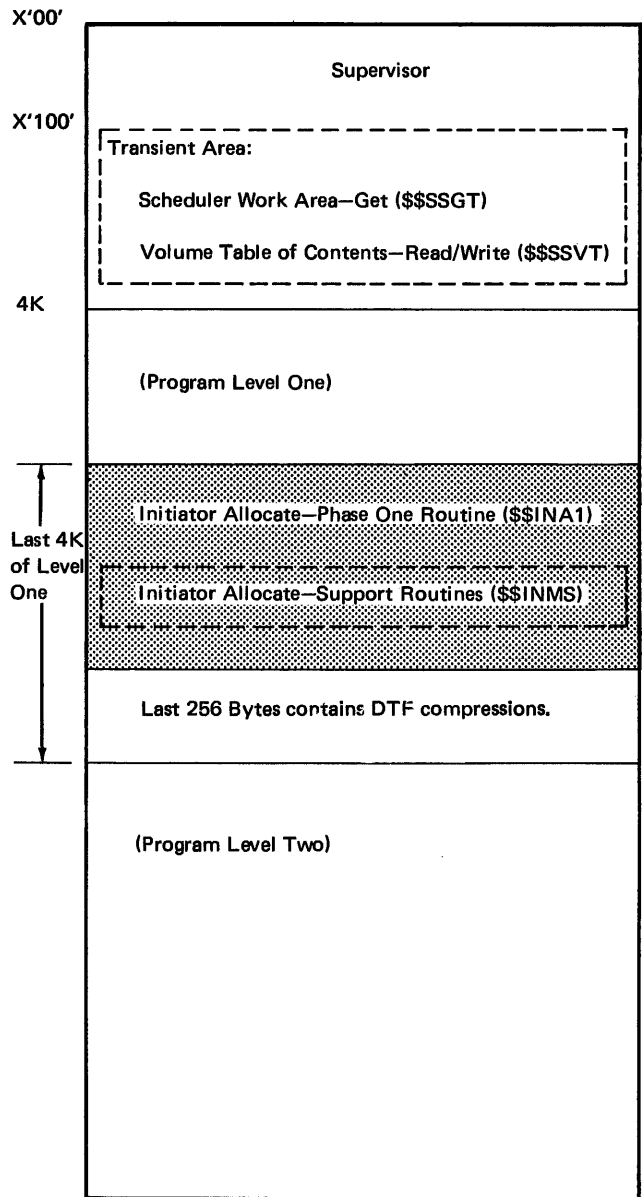


Figure 5-45. Main Storage Map for the Initiator Allocate—Phase One (\$\$INA1)

► Initiator Allocate—Phase Two (\$\$INA2)

CHART: HB

FIGURE: 5-46

ENTRY POINT: \$\$INA2

FUNCTION: Builds a composite image of all tracks, allocated on all units, needed for file allocating. The composite consists of volume labels and scheduler work areas for both program levels.

INPUT:

- Volume labels
- SWA

OUTPUT:

- Composite of the format fives for the possible four disk units
- The format five image from the current program level

EXIT: Initiator Allocate—Phase Three (\$\$INA3)

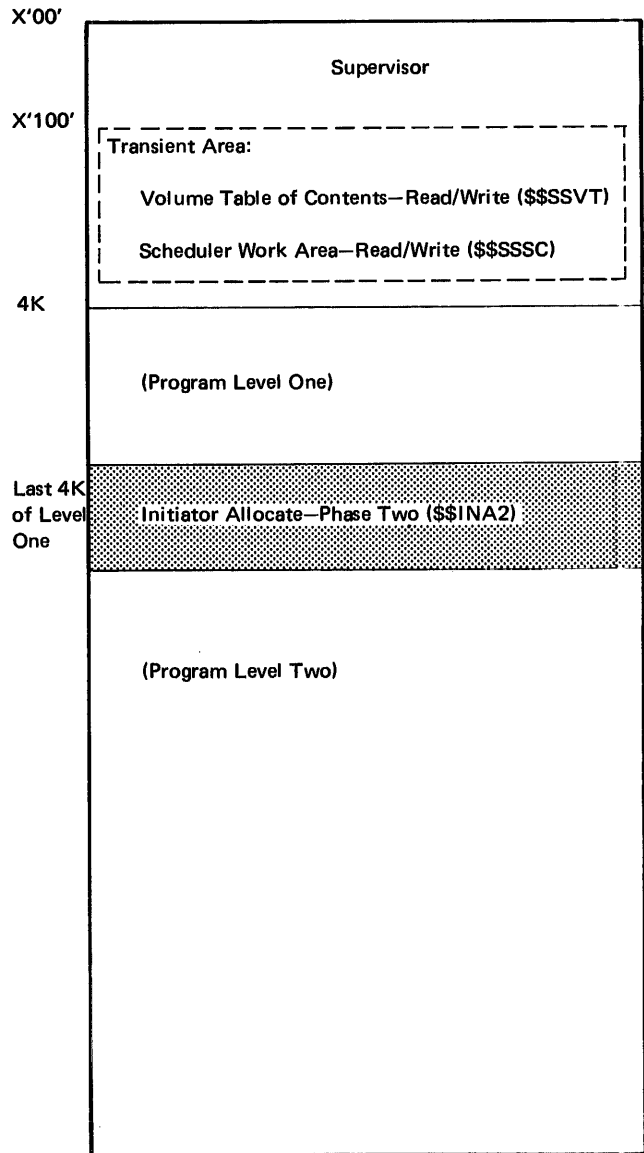


Figure 5-46. Main Storage Map for the Initiator Allocate—Phase Two Routine (\$\$INA2)

► Initiator Allocate—Phase Three (\$\$INA3)

CHART: HC

FIGURE: 5-47

ENTRY POINT: \$\$INA3

FUNCTION: Allocates file space by using the format one and format five labels.

INPUT:

- Format five composites
- Format one images

OUTPUT: The format five buffer in the SWA is altered to reflect new allocated space.

EXIT:

- Normal: Initiator Allocate—Phase Four (\$\$INA4)
- Error: Supported Halt/Syslog routine, or when allocating temporary disk space for special requests and space is not available, control is passed to \$\$INAT.

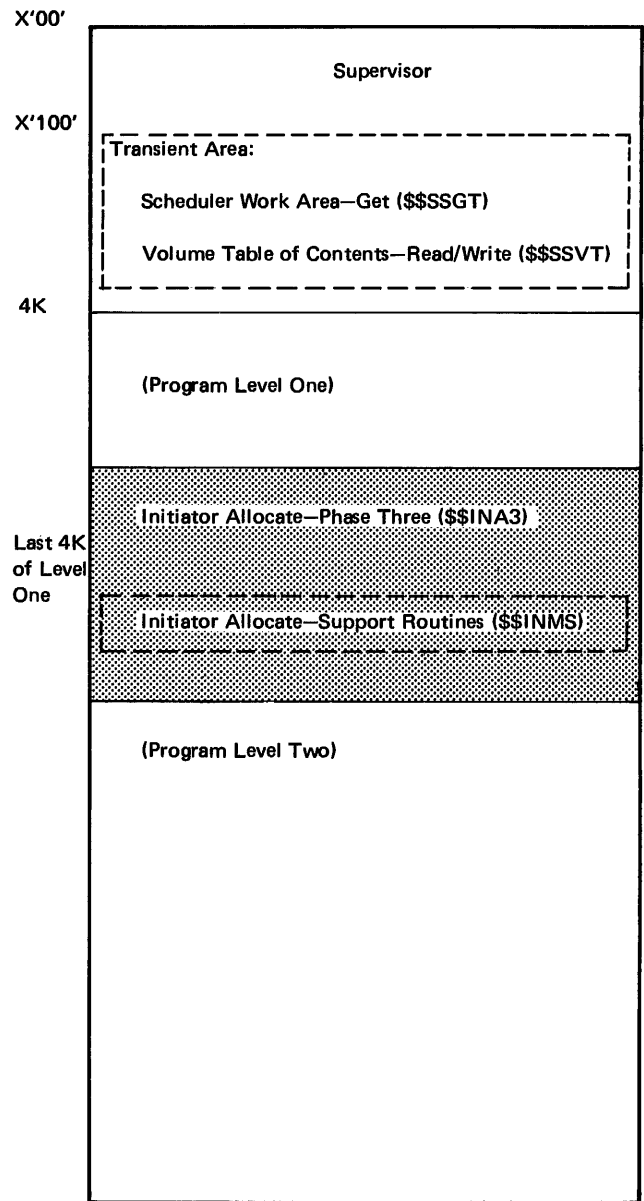


Figure 5-47. Main Storage Map for the Initiator Allocate—Phase Three Routine (\$\$INA3)

► Initiator Allocate—Phase Four (\$\$INA4)

CHART: HD

FIGURE: 5-48

ENTRY POINT: \$\$INA4

FUNCTION:

- Dequeues all S files whose space was just allocated elsewhere.
- Calls the Allocate Terminator (\$\$INAT).

INPUT:

- The format ones from the VTOCs
- The sector that contains indicators for all new files

OUTPUT:

- The format five sector is written back into the SWA.
- The conflicting S files are removed from the volume table of contents (VTOC).

EXIT: Allocate Terminator (\$\$INAT)

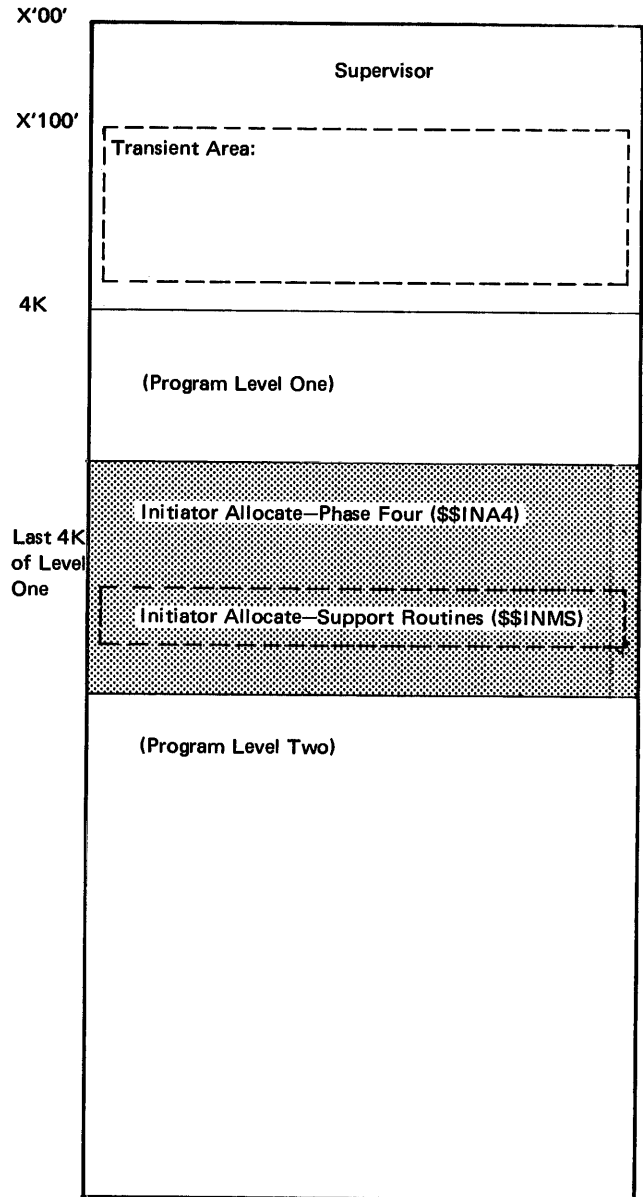


Figure 5-48. Main Storage Map for the Initiator Allocate—Phase Four (\$\$INA4)

► Initiator Disk File Processor—Phase One (\$\$INPD)

CHART: HE

FIGURE: 5-49

ENTRY POINT: \$\$INPD—Mainline

INTRA-PHASE ENTRY POINTS:

- PDHIKE – HIKEY Processor
- PDPROC – Name, Label, Retain, and Pack Processor
- PDTRKS – Location, Tracks, and Records Processor
- PDUNIT – Unit Processor
- PDATE – Date Processor
- PDBLF1 – Format One Build Routine

FUNCTION:

- Decodes the // FILE control cards located in the initiator table.
- Transfers the information found on the // FILE control cards to the format one image format located in the SWA.

INPUT: Initiator table located in the SWA

OUTPUT: Format one pre-images located in the SWA

EXIT:

- Normal: Initiator Disk File Processor—Phase Two (\$\$INDF)
- Error: End of Job transient (\$\$SPEJ)

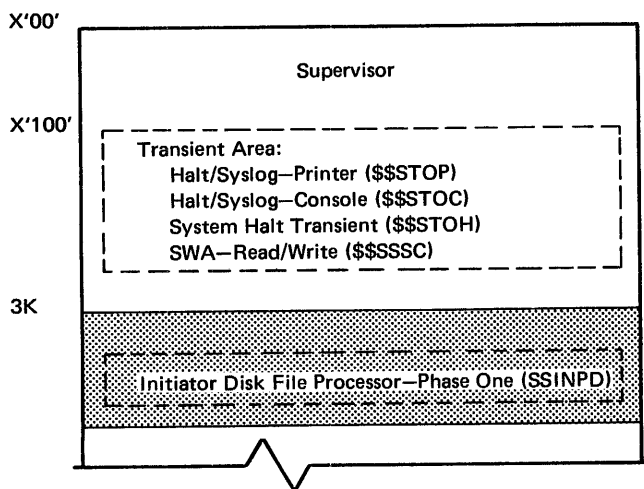


Figure 5-49. Main Storage Map for the Initiator Disk File Processor—Phase One (\$\$INPD)

► Initiator Disk File Processor—Phase Two (\$\$INDF)

CHART: HF

FIGURE: 5-50

ENTRY POINT: \$\$INDF—Mainline

FUNCTION:

- Performs all pre-processing of disk files by creating format one images in the SWA.
- Ensures that the MVF label for each pack is unique.

INPUT: Format one pre-images supplied by \$\$INPD

OUTPUT:

- Diagnostics of format one images
- Format one images

EXIT:

- Normal:
 1. Initiator Program Setup—Phase Two routine (\$\$INP2) if no // FILE Control cards were given.
 2. Initiator Disk File Processor—Phase Three (\$\$INDC) if // FILE Control Cards were given.
- Error: End of Job Transient (\$\$SPEJ)

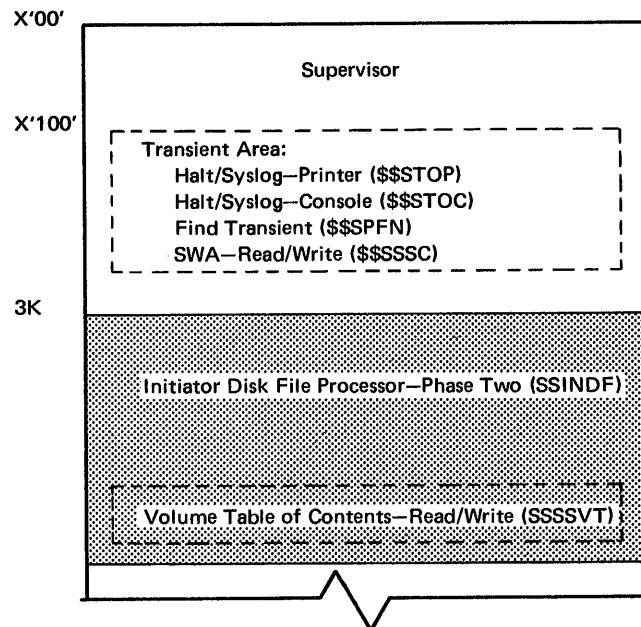


Figure 5-50. Main Storage Map for the Initiator Disk File Processor—Phase Two (\$\$INDF)

► Initiator Disk File Processor—Phase Three (\$\$INDC)

CHART: HG
 FIGURE: 5-51
 ENTRY POINT: \$\$INDC
 FUNCTION:

- Determines the current program level.
- Ensures that within this level, no two files have the same name.
- Ensures that file labels and dates located on the same disk pack are unique.

INPUT: Format one images found in the SWA (sectors 3-15 on the system disk pack)

OUTPUT: A halt is issued if a duplicate format one image name is located or if an attempt is being made to create a file with the same label and date as an existing file on a given pack.

EXIT:

- Normal: Initiator Disk File Processor—Phase Four (\$\$INDS)
- Error: Supported Halt/Syslog routine which performs an immediate cancel

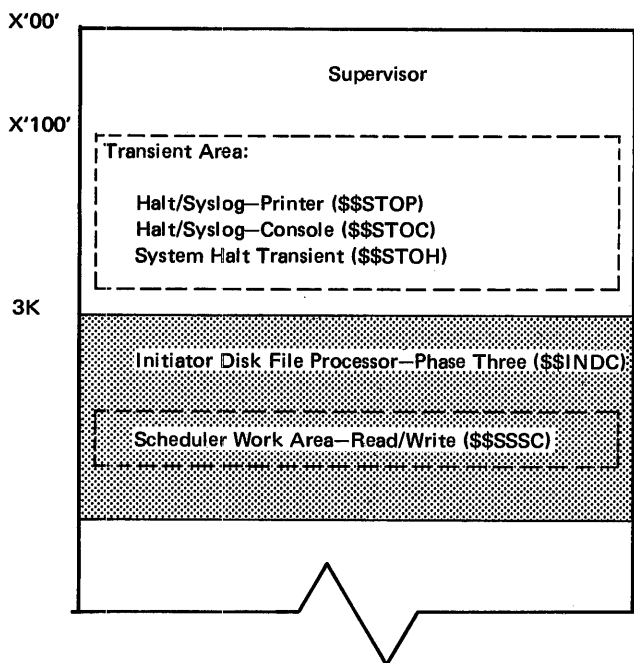


Figure 5-51. Main Storage Map for the Initiator Disk File Processor—Phase Three (\$\$INDC)

► Initiator Disk File Processor—Phase Four (\$\$INDS)

CHART: HH
 FIGURE: 5-52
 ENTRY POINT: \$\$INDS
 FUNCTION:

Sorts the format one images (located in sectors 3-15 of the SWA) into the order required by the Allocate or Open routines.

INPUT: The format one images located in the SWA

OUTPUT: Ordered format one images are reloaded into the SWA

EXIT: Initiator Program Setup—Phase Two (\$\$INP2)

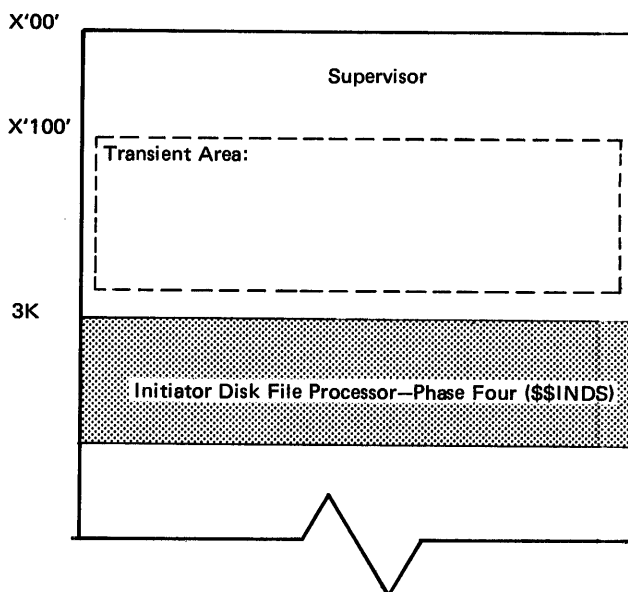


Figure 5-52. Main Storage Map for the Initiator Disk File Processor—Phase Four (\$\$INDS)

► Initiator Program Setup—Phase One (\$\$INPS)

CHART: HI

FIGURE: 5-53

ENTRY POINT: \$\$INPS

FUNCTION:

- Performs tests to ensure the program can be loaded.
- Obtains the address of the program to be executed for the loader.

INPUT:

- Initiator table located in the SWA
- Temporary configuration record located in the SWA

OUTPUT: Program attributes placed in the SWA

EXIT:

- Normal: Initiator Disk File Processor—Phase One (\$\$INPD)
- Error: Supported Halt/Syslog routine

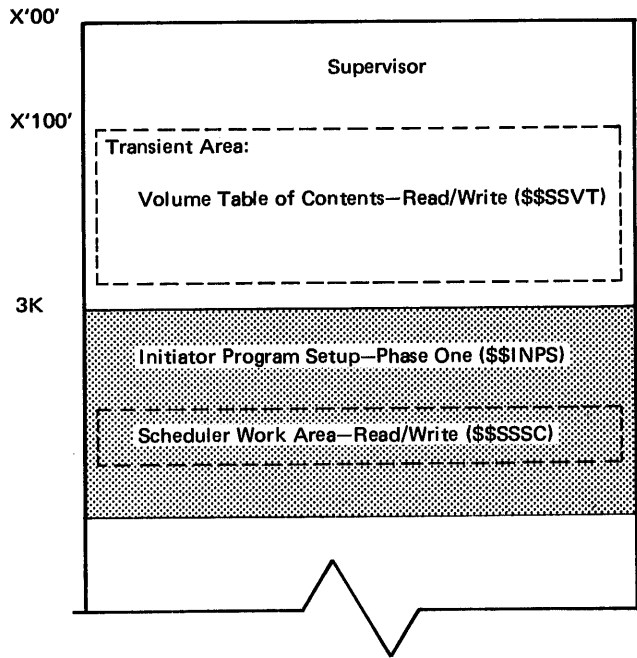


Figure 5-53. Main Storage Map for the Initiator Program Setup—Phase One (\$\$INPS)

► Initiator Program Setup—Phase Two (\$\$INP2)

CHART: HJ

FIGURE: 5-54

ENTRY POINT: \$\$INP2

FUNCTION:

- Moves source information into a source file.
- Allocates disk storage for the source and work files if space does not already exist by calling \$\$STAI.

INPUT: Source from Sysin device or from the source library

OUTPUT: Source file containing information from the Sysin device or the source library

EXIT:

- Normal: Initiated program that was fetched into main storage
- Error: Supported Halt/Syslog routine

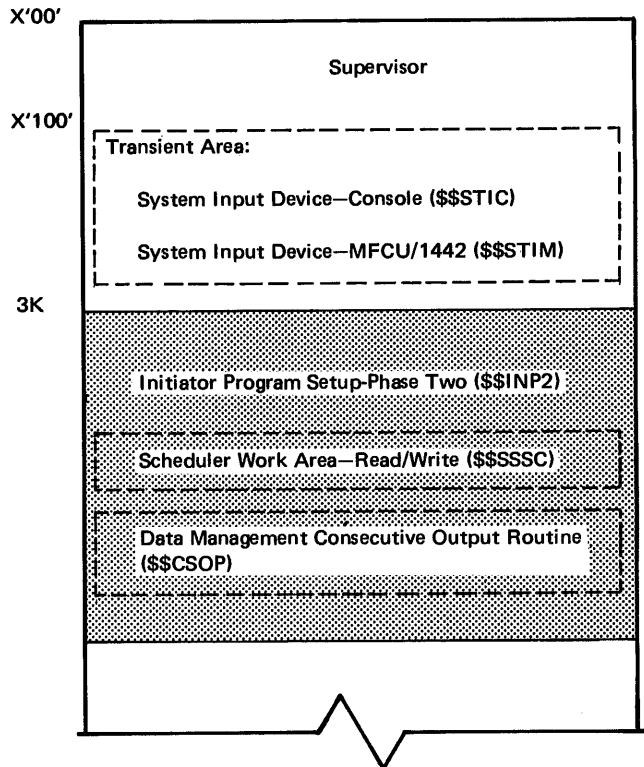


Figure 5-54. Main Storage Map for the Initiator Program Setup—Phase Two (\$\$INP2)

► Initiator Allocate—Support Routines (\$\$INMS)

CHART: HL-HQ

FIGURE: 5-55

ENTRY POINT:

- PAITRK—Convert Records to Tracks
- PAINDX—Determine size of index
- PARTIN—Track Apportioning
- PAGTRK—Track Conversion
- DFSBN—Set Bits Routine
- DCLPAF—Possible Allocation Fit

FUNCTION: The entry points are to individual routines that are linkage edited together under the module name \$\$INMS. These routines provide computation support for the Initiator Disk File Allocation routines (\$\$INA1, \$\$INA2, \$\$INA3, \$\$INA4). Each routine, referenced by the entry points listed, is described by its own flowchart and description.

INPUT: See the individual descriptions.

OUTPUT: See the individual descriptions.

EXIT: See the individual descriptions.

► Convert Records to Tracks

CHART: HL

FIGURE: 5-55

ENTRY POINT: PAITRK

FUNCTION: Determines the number of tracks required by a file, when only the number of records are given.

INPUT:

- Register 2 contains the address of the format one file.
- The format one file contains the number of records and their lengths.

OUTPUT: Register 2 contains the address of the updated format one file.

EXIT: Calling routine

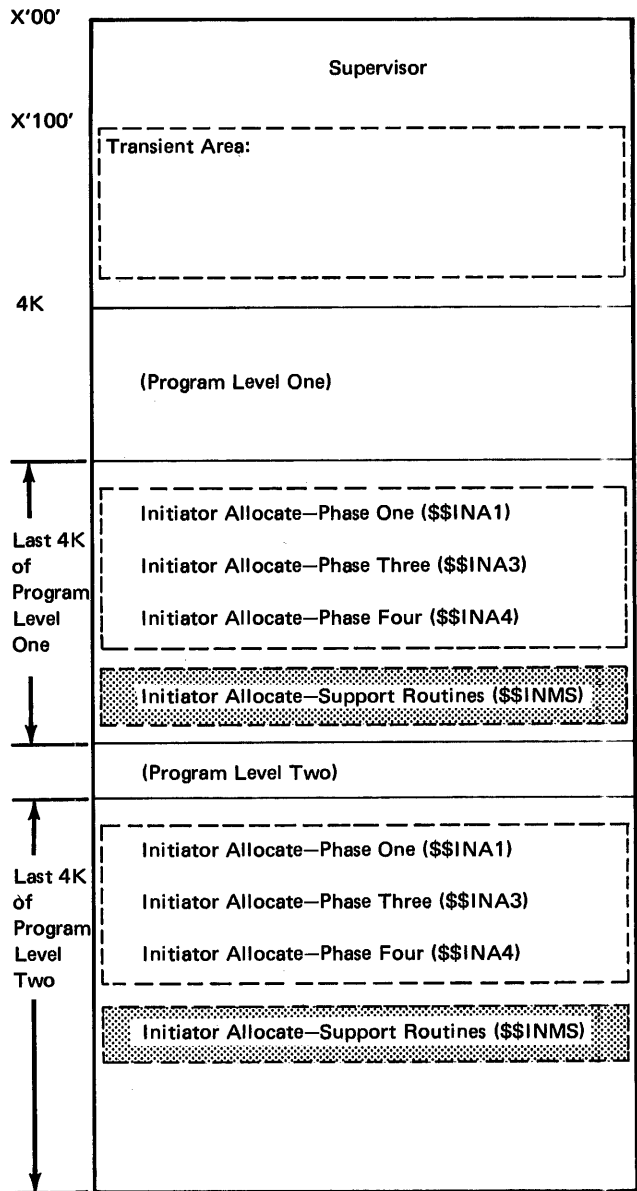


Figure 5-55. Main Storage Map for the Initiator Allocate—Support Routines (\$\$INMS)

► **Determine Size of Index**

CHART: HM

FIGURE: 5-55

ENTRY POINT: PAINDX

FUNCTION: Determines the number of tracks that are required for the index when only the number of records are given for the index file.

INPUT: Register 2 contains the address of the format one area. This area contains the number and size of the records to be contained in the index file.

OUTPUT: Register 2 contains the address of the format one area. This area will contain an updated maximum and minimum address field along with the value for the number of tracks required for the index file.

EXIT: Next sequential instruction (NSI) of the caller

► **Track Apportioning**

CHART: HN

FIGURE: 5-55

ENTRY POINT: PARTIN

FUNCTION: Apportions the tracks between the data and index

INPUT: Register 2 contains the address of the format one area. This area contains the number of tracks that are available for apportioning, the record length, and the key length.

OUTPUT: Register 2 contains the address of the format one area. This area will contain the minimum and maximum addresses and the number of tracks that have been apportioned.

EXIT: Calling routine

► **Track Conversion**

CHART: HO

FIGURE: 5-55

ENTRY POINT: PAGTRK

FUNCTION: Determines the displacement and the proper Q code for the referencing of a track indicator within the format five image.

INPUT: Register 2 contains the address of a 2-byte area that contains the track number.

OUTPUT: Register 2 contains the mask used to reference the 2-byte data area located within the format five image.

EXIT: Calling routine

► **Set Bits Routine**

CHART: HP

FIGURE: 5-55

ENTRY POINT: DFSBN

FUNCTION: Sets on the appropriate track bits in the format five file indicating the tracks that are used in the source format one file.

INPUT: Format one files

OUTPUT: The format five file is updated to reflect the tracks used by the format one file.

EXIT: Calling routine

► **Possible Allocation Fit**

CHART: HQ

FIGURE: 5-55

ENTRY POINT: DCLPAF

FUNCTION: This routine will locate the smallest amount of contiguous disk space that will contain the specified new disk file.

INPUT:

- Register 1 contains the address of the format five image to be used.
- Register 2 contains the address of the format one for the new disk file.

OUTPUT:

- Registers 1 and 2 contain the same addresses.
- A return code is passed to the caller indicating the degree of success of locating the needed space.

EXIT: Calling routine

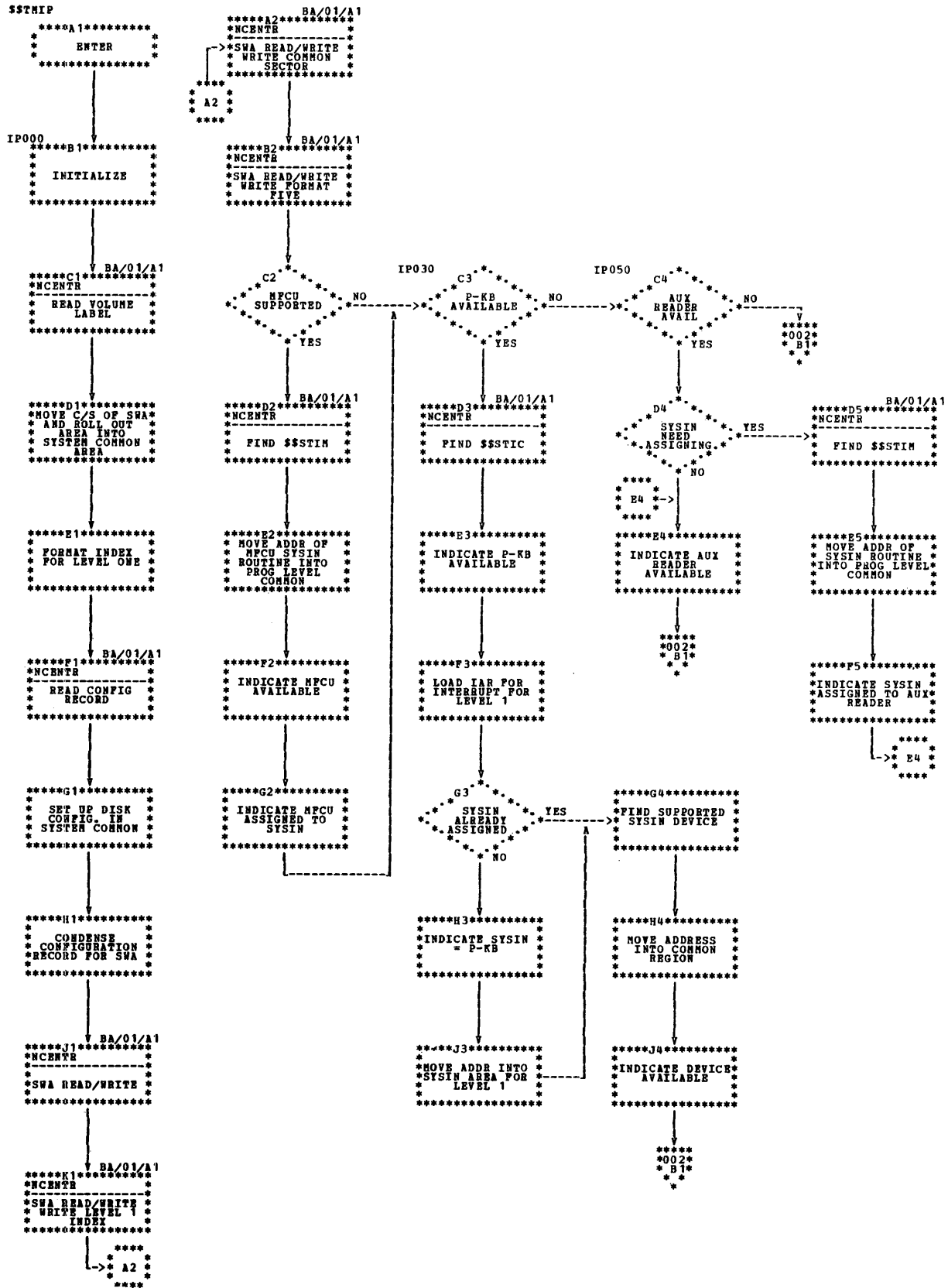


Chart EA (Part 1 of 2). Terminator Initial Program Load Routine (\$\$TMIP)

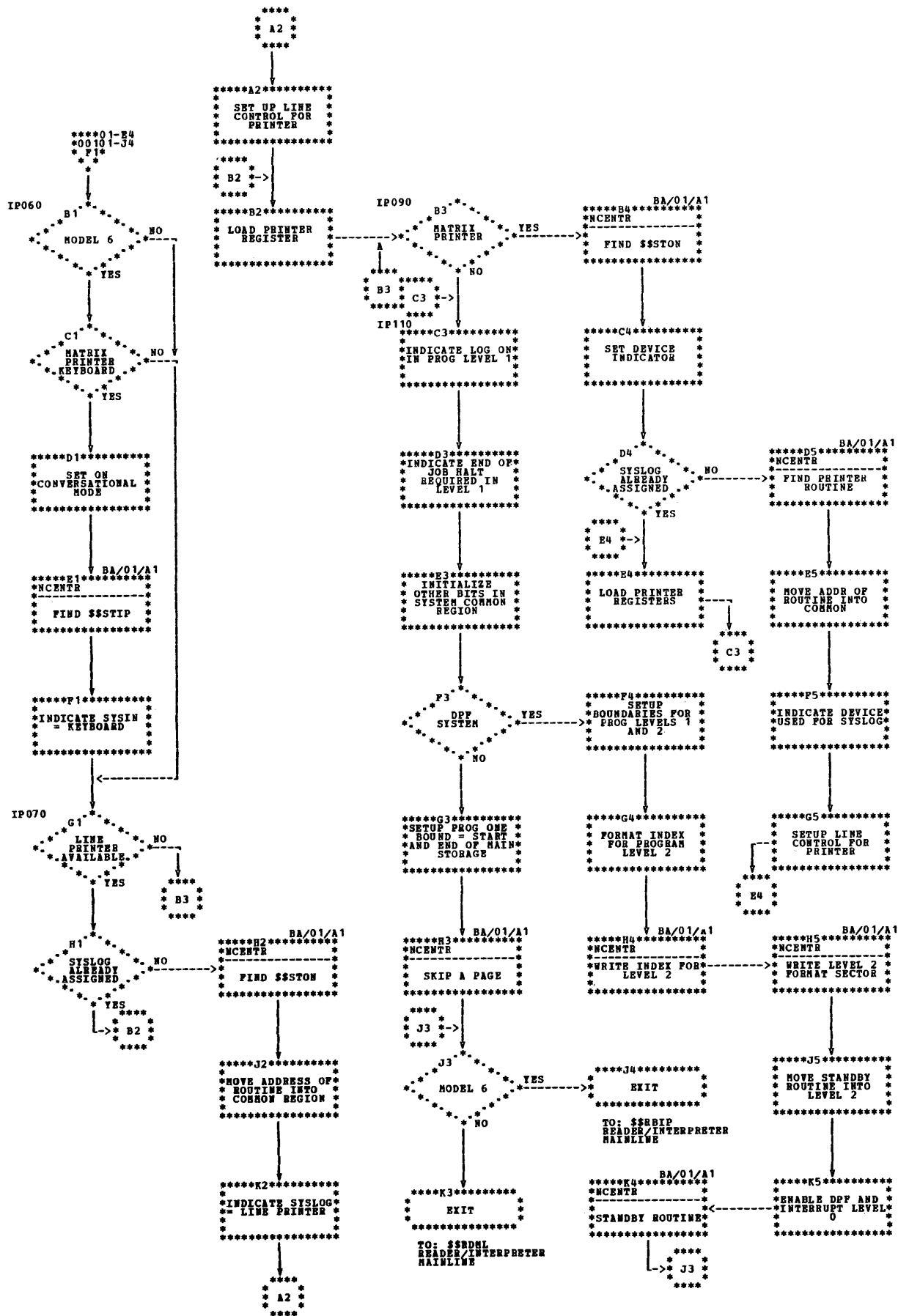


Chart EA (Part 2 of 2). Terminator Initial Program Load Routine (\$\$TMP)

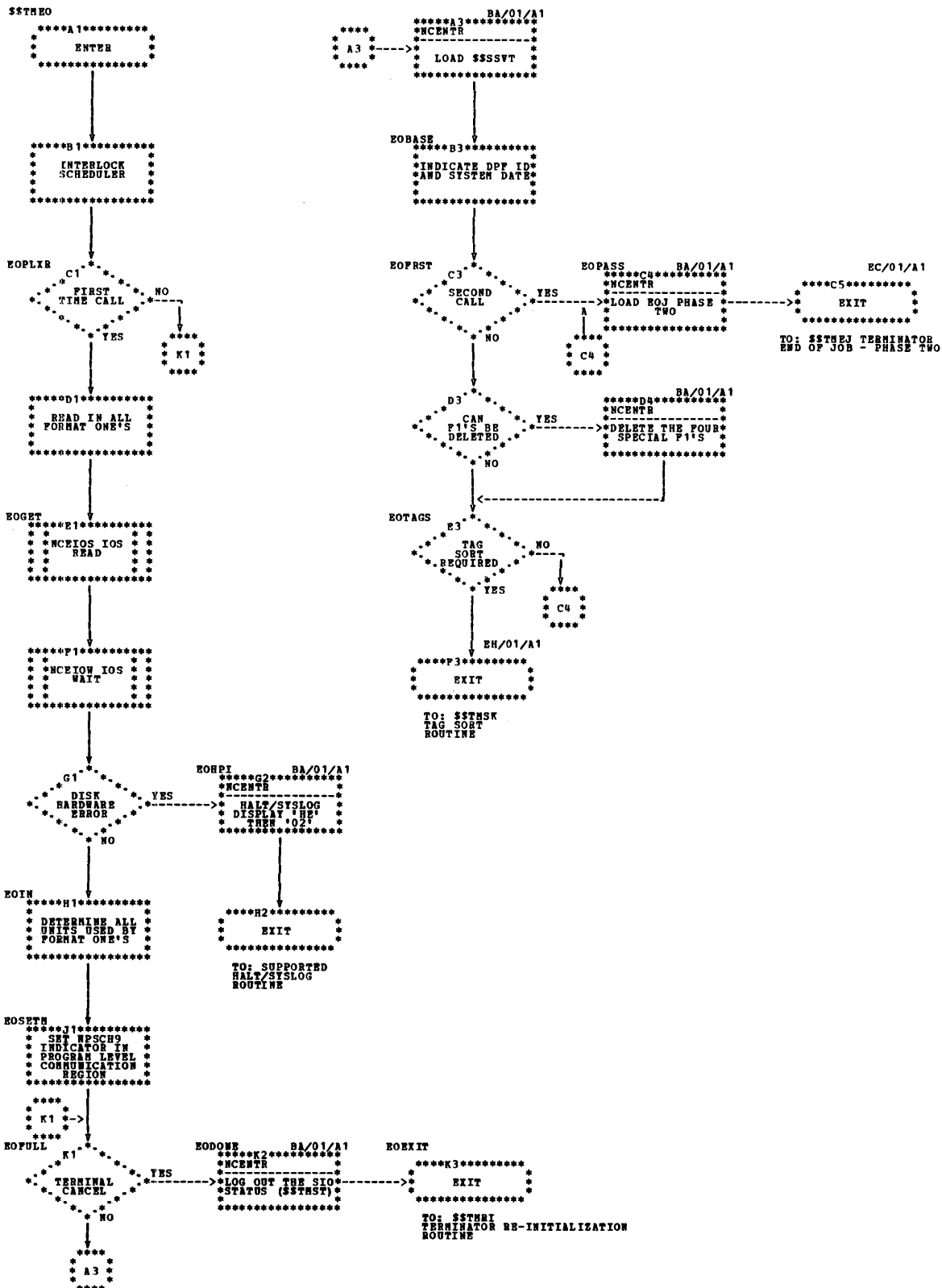


Chart EB. Terminator End of Job-Phase One (\$\$TMEO)

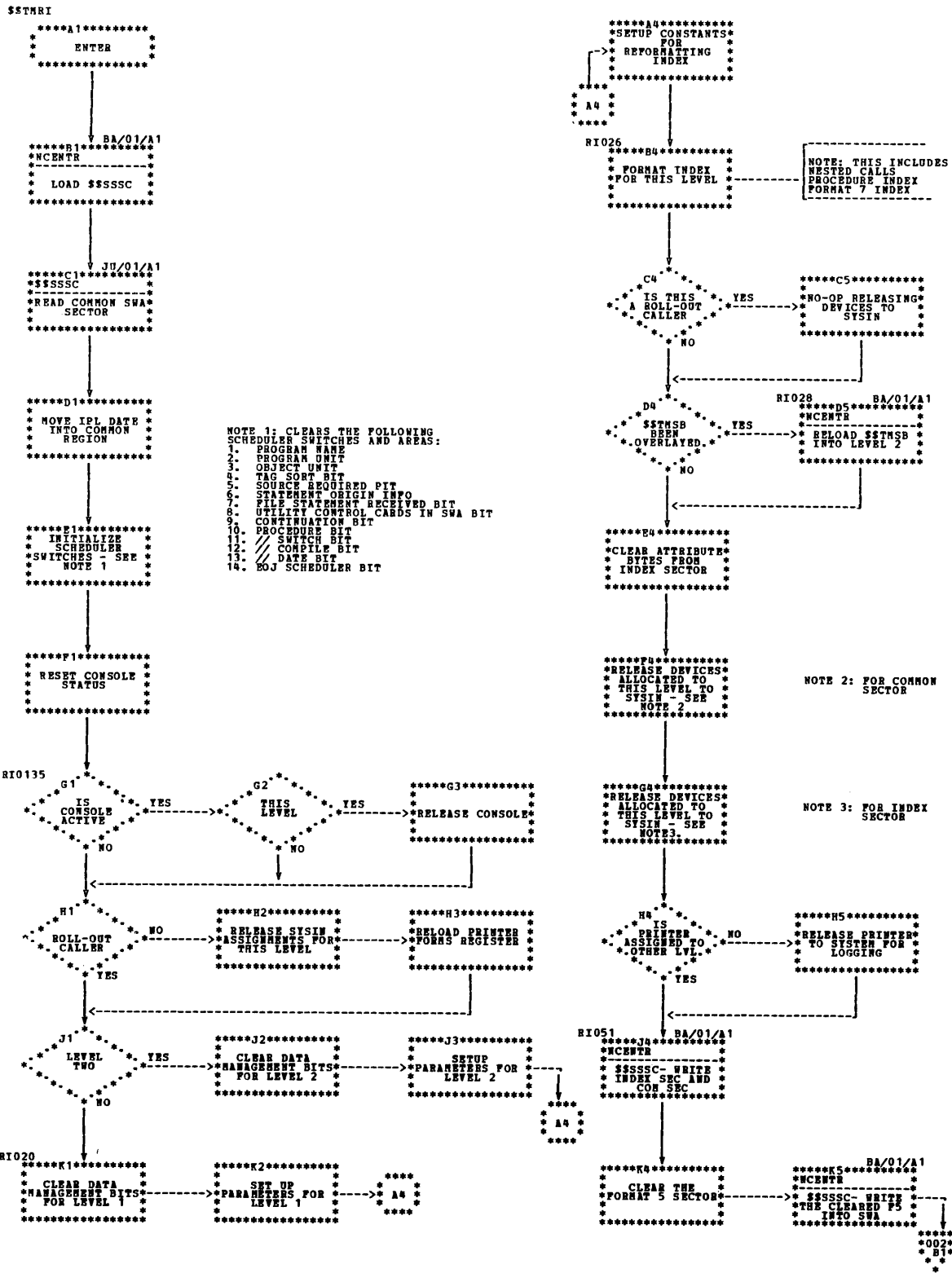


Chart ED (Part 1 of 3). Terminator Re-Initialization (\$\$TMRI)

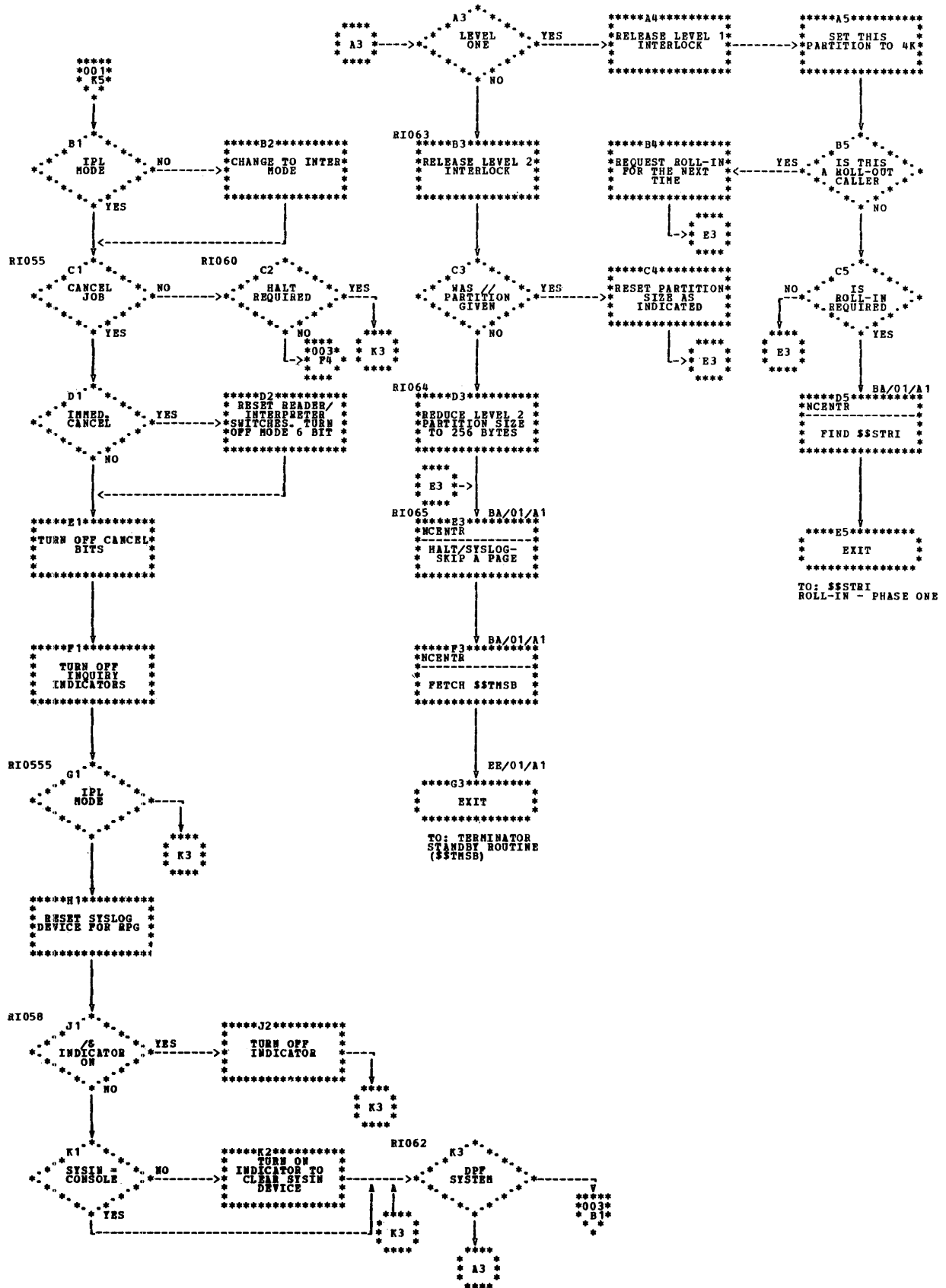


Chart ED (Part 2 of 3). Terminator Re-Initialization (\$\$TMRI)

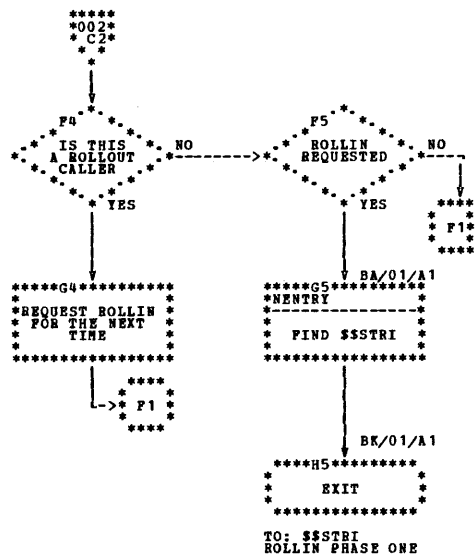
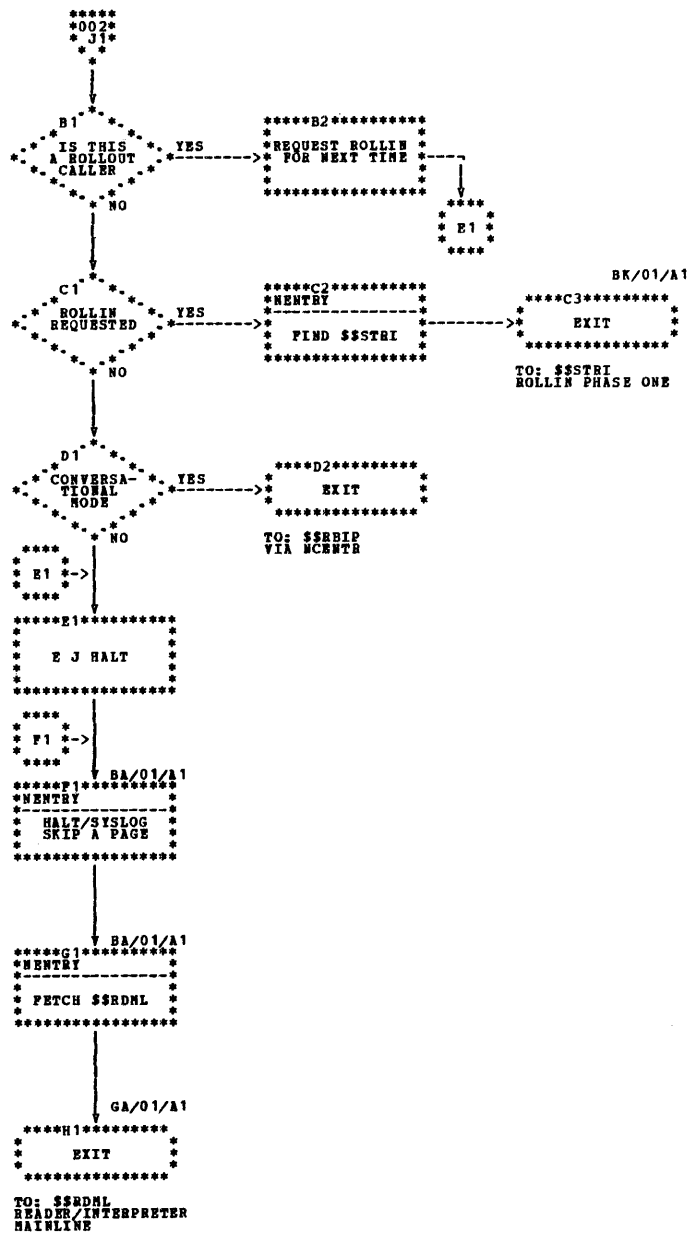


Chart ED (Part 3 of 3). Terminator Re-Initialization (\$\$TMRI)

```

$$$TMSB
*****A1*****
*   ENTER   *
*****
STBY
*****E1*****
*OBTAIN POINTER *
*TO PROG LEVEL *
*COMMON REGION *
*****
*   C1 *-> *
*****
*   SET HALT TO *
*   'BJ' *
*****
HPLNSI
*****D1*****
*STORE NSI OF *
*HPL IN PROG *
*LEVEL COMMON *
*REGION *
*****
*   E1 *-> *
*****
CLEAR
*****E1*****
*CLEAR SENSED *
*SWITCH *
*****
INOPV
*****F1*****
*INDICATE THIS *
*LEVEL *
*IN-OPERATIVE *
*****
HALT
*****C1*****
*HALT THIS *
*PROGRAM LEVEL *
*****
HPLNSI
*****H1*****
*INDICATE THIS *
*LEVEL OPERATIVE *
*****
*   J1 *
*IS *
*THERE 4K *
*AVAILABLE *
*   YES *-> *
*   A3 *
*   NO *
*****
*****K1*****
*SET HALT TO *
*'JL' *
*****
*-> *
*   E1 *
*****

```

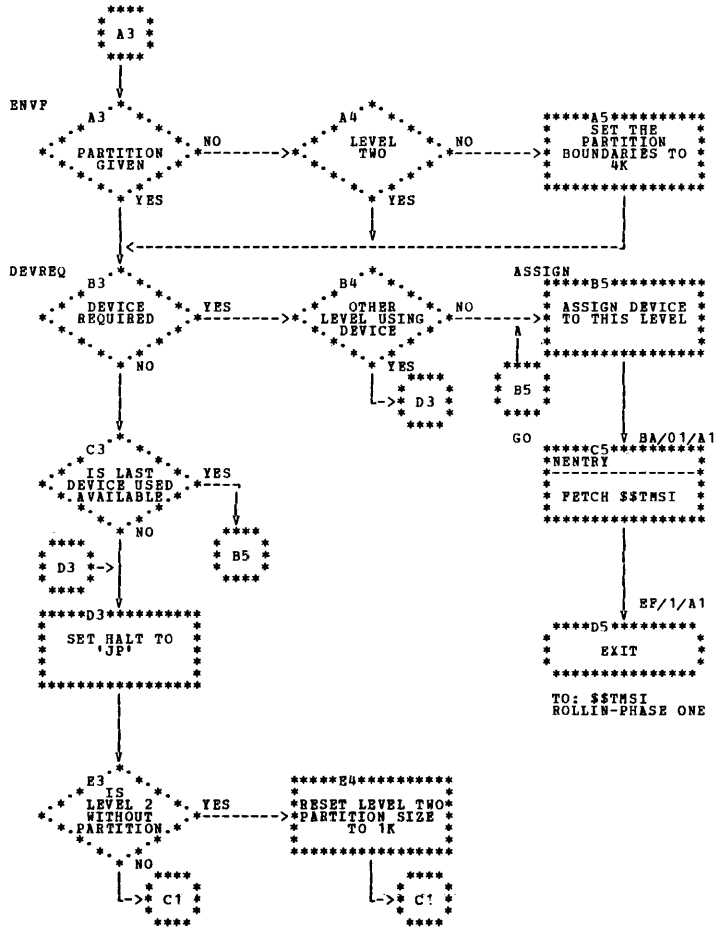


Chart EE. Terminator Standby Routine (\$\$TMSB)

\$\$\$TMSI

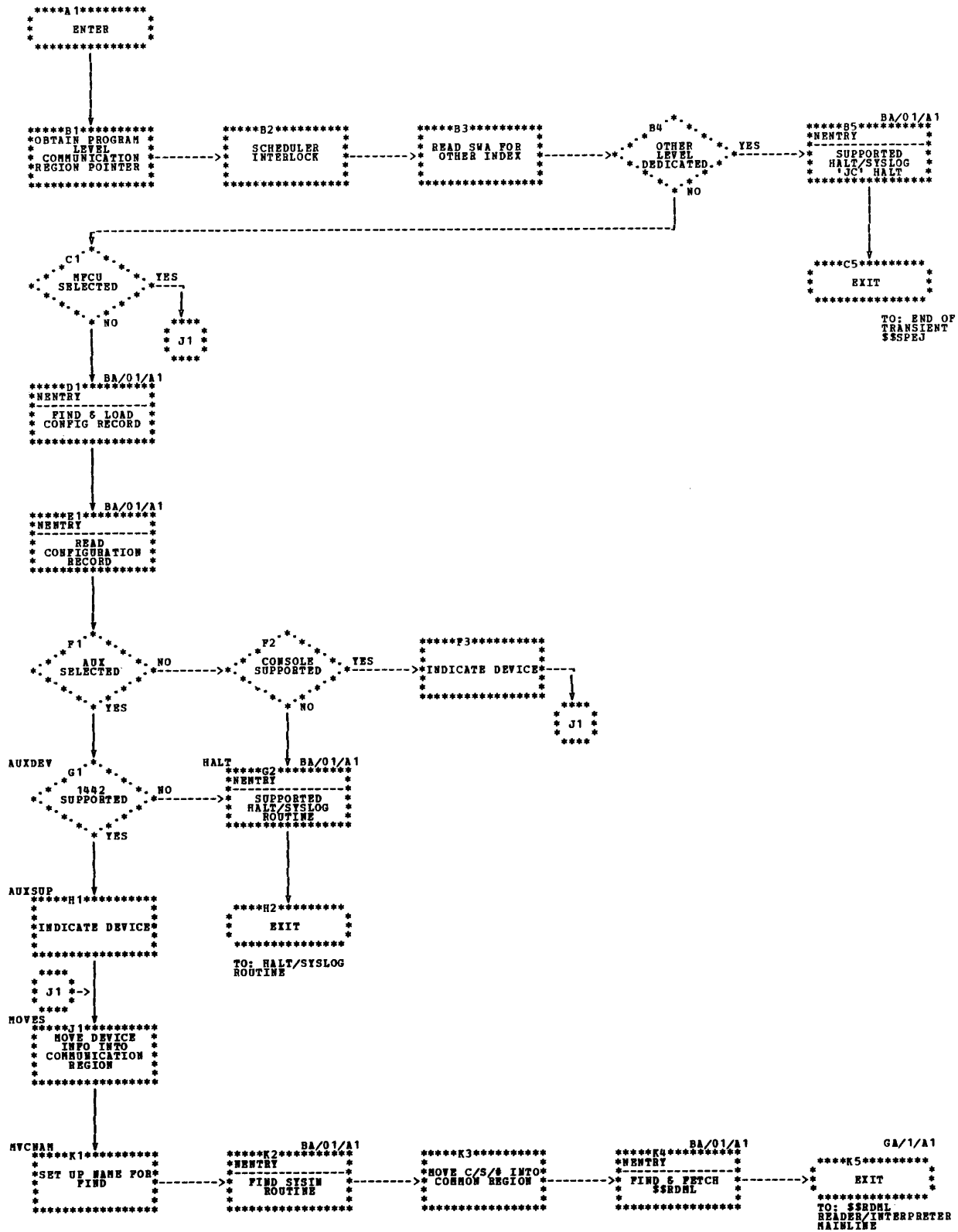
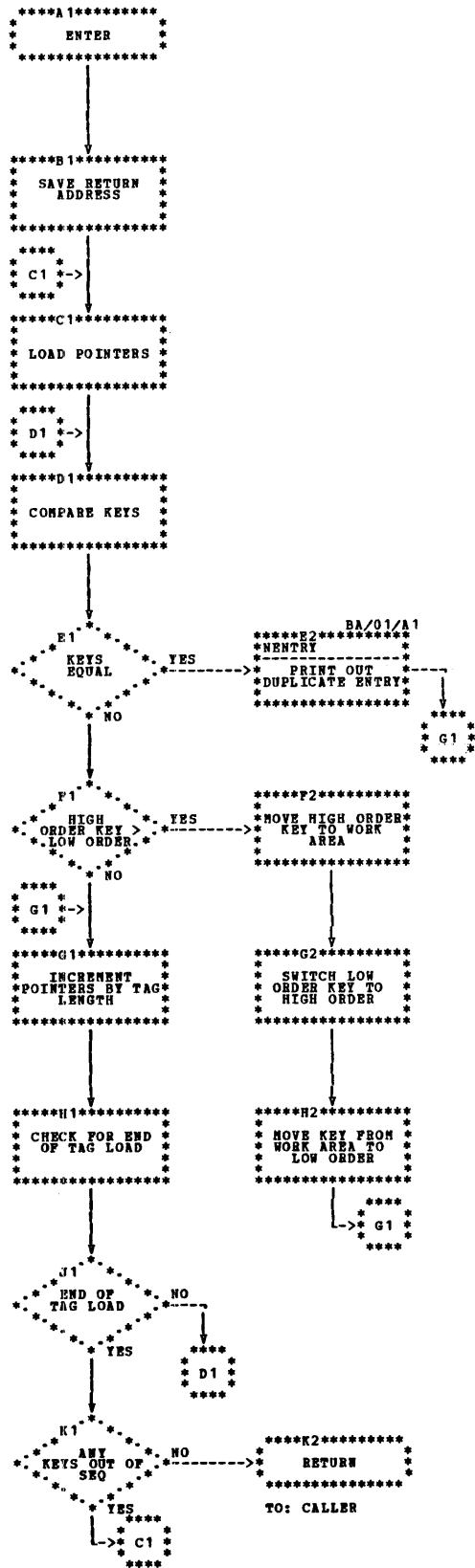


Chart EF. Terminator Sysin Initialization Routine (\$\$TMSI)

SISORT



SIRTGA

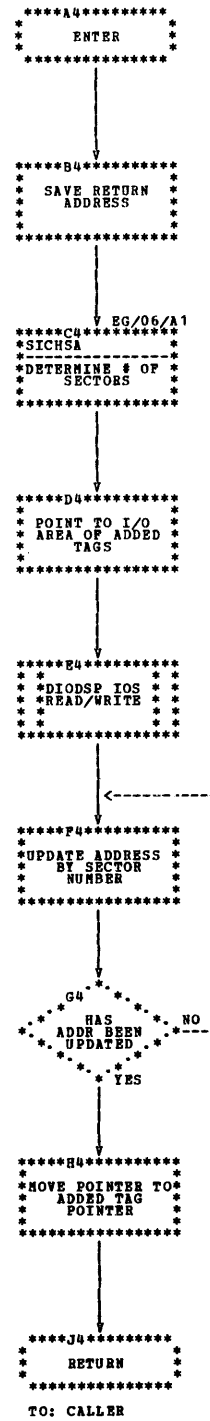


Chart EG (Part 3 of 6). Terminator Key Sort Routine (\$\$TMSK)

SIUPDT

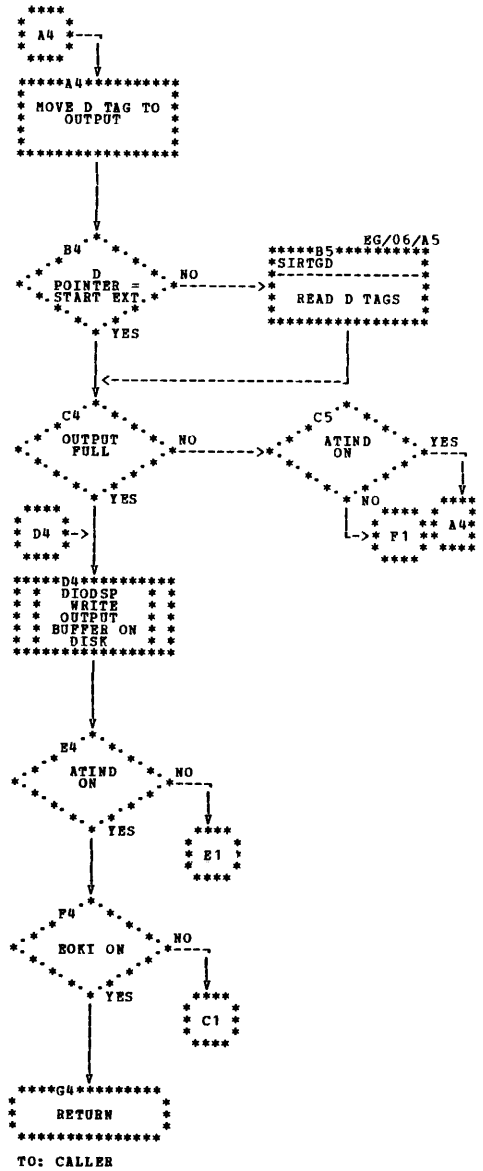
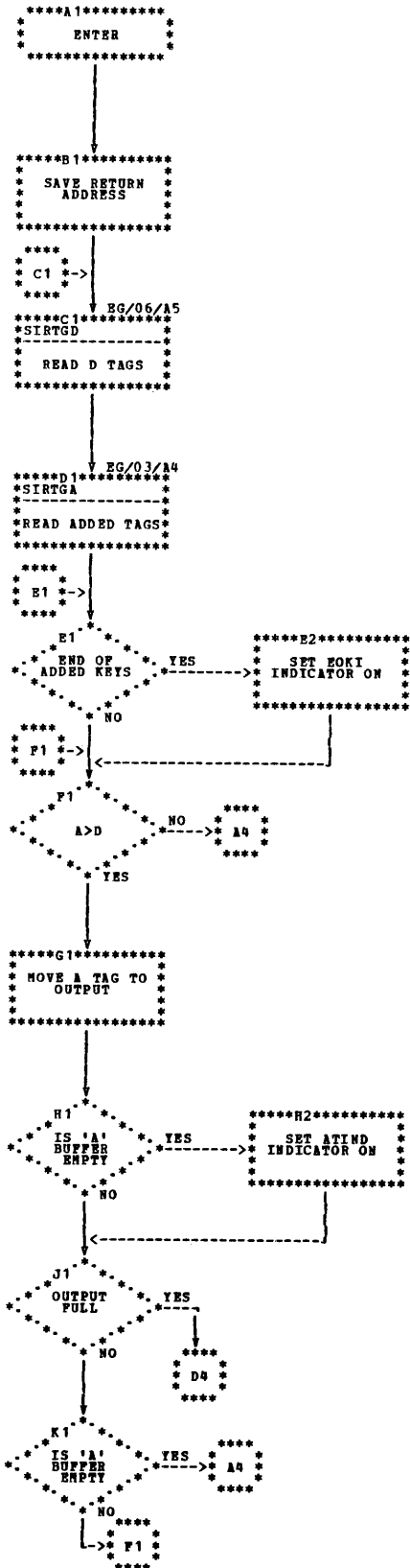


Chart EG (Part 4 of 6). Terminator Key Sort Routine (\$TMSK)

SIORDR

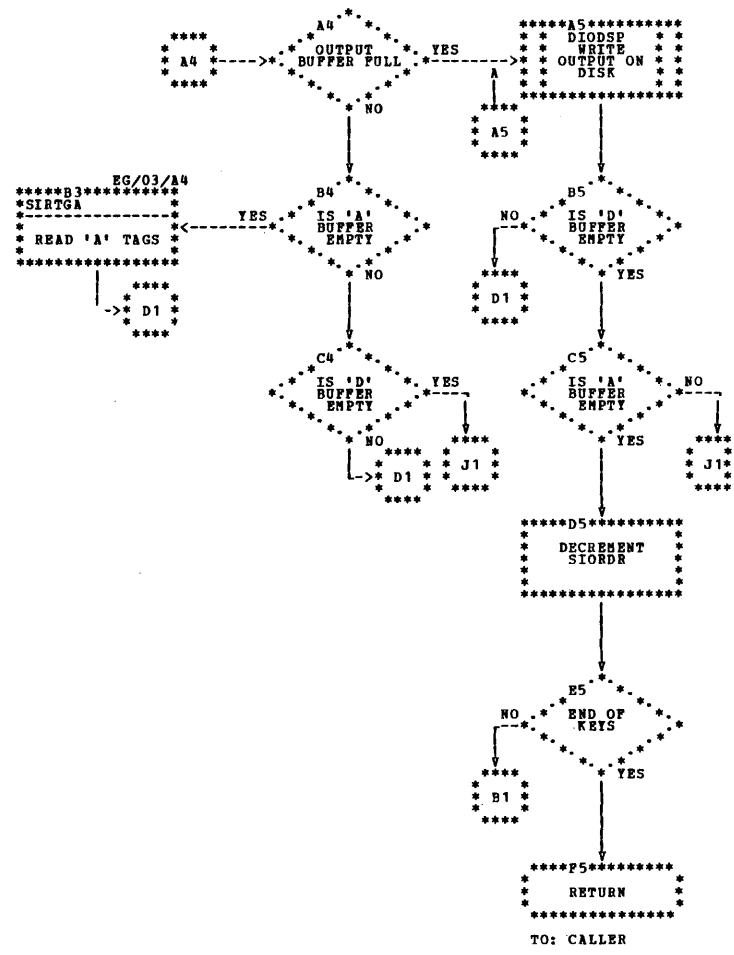
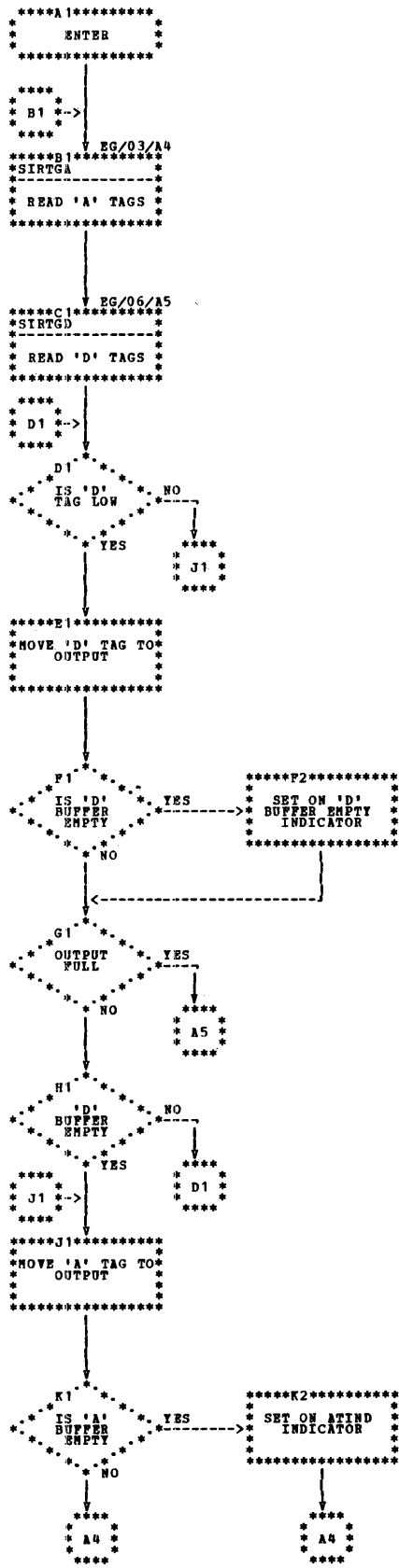


Chart EG (Part 5 of 6). Terminator Key Sort Routine (\$\$TMSK)

DMCDDS

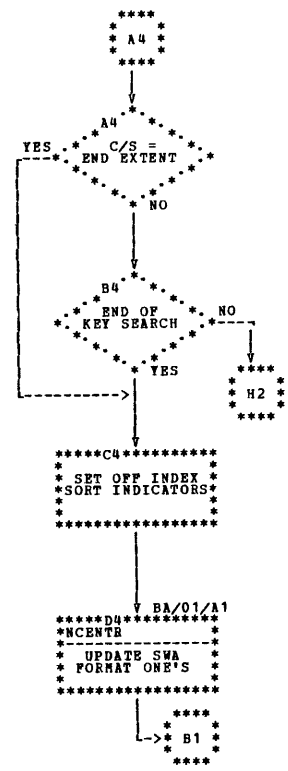
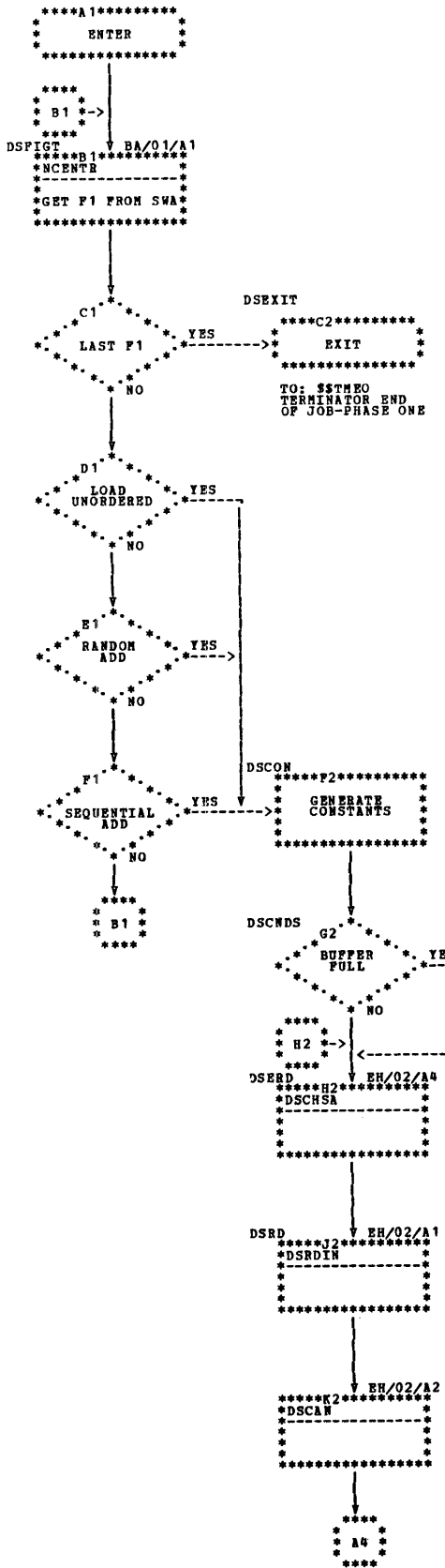


Chart EH (Part 1 of 2). Terminator Duplicate Key Scan Routine (\$\$TMDS)

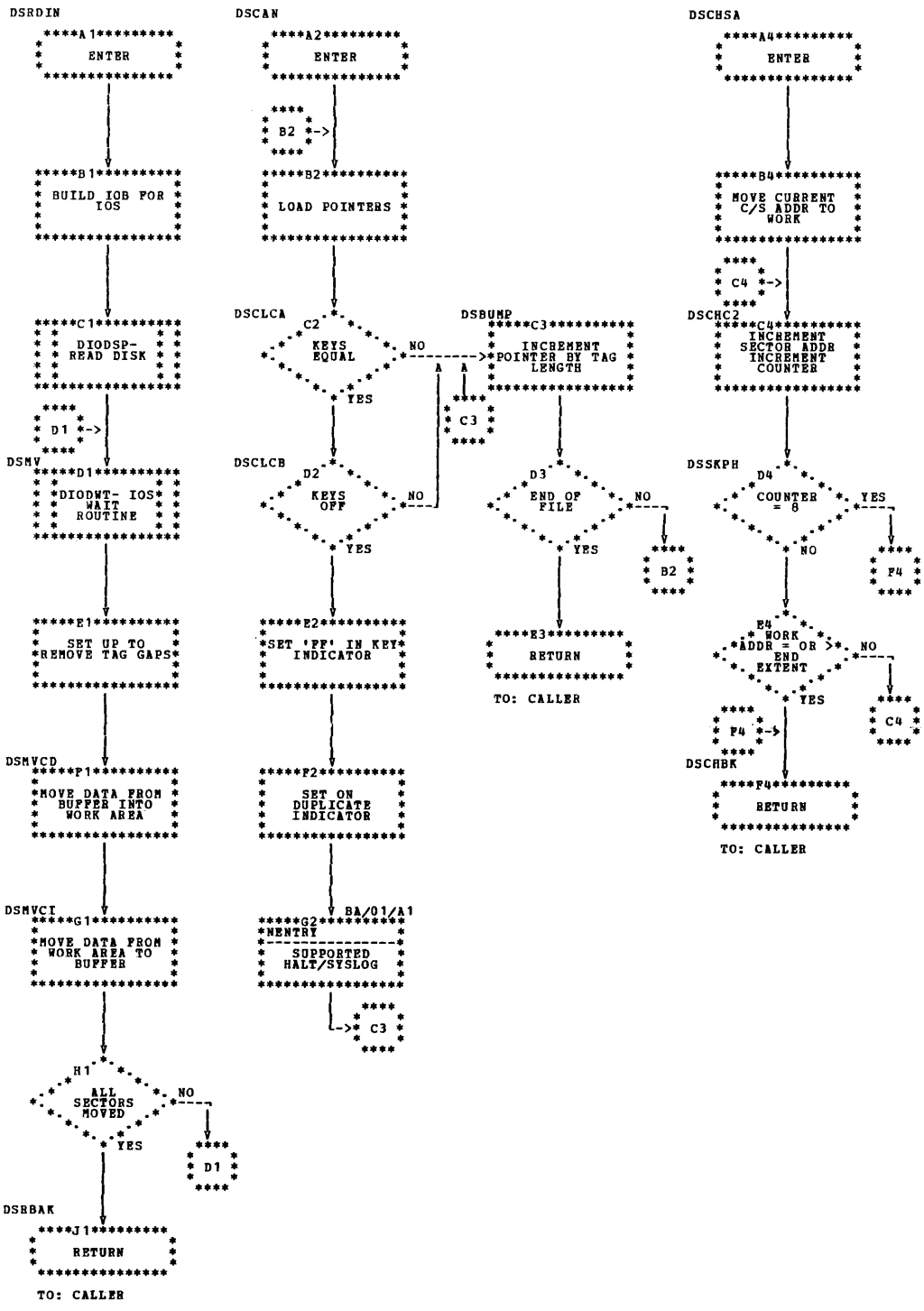
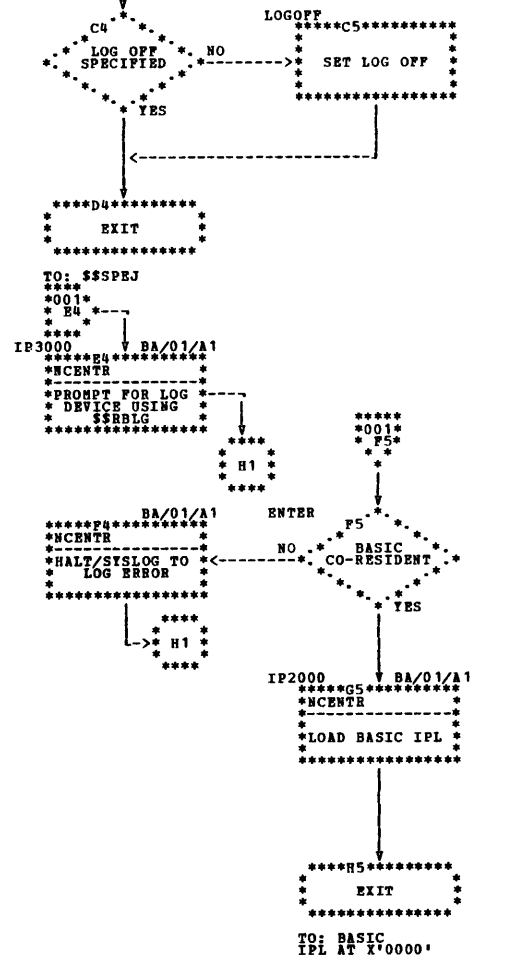
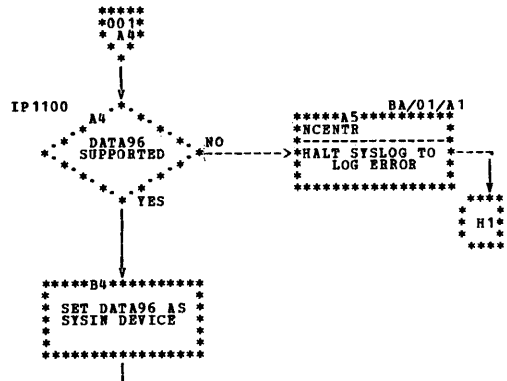
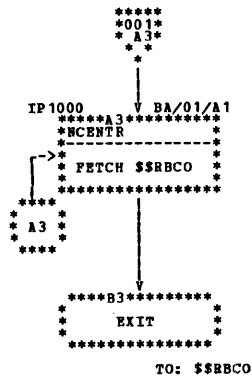
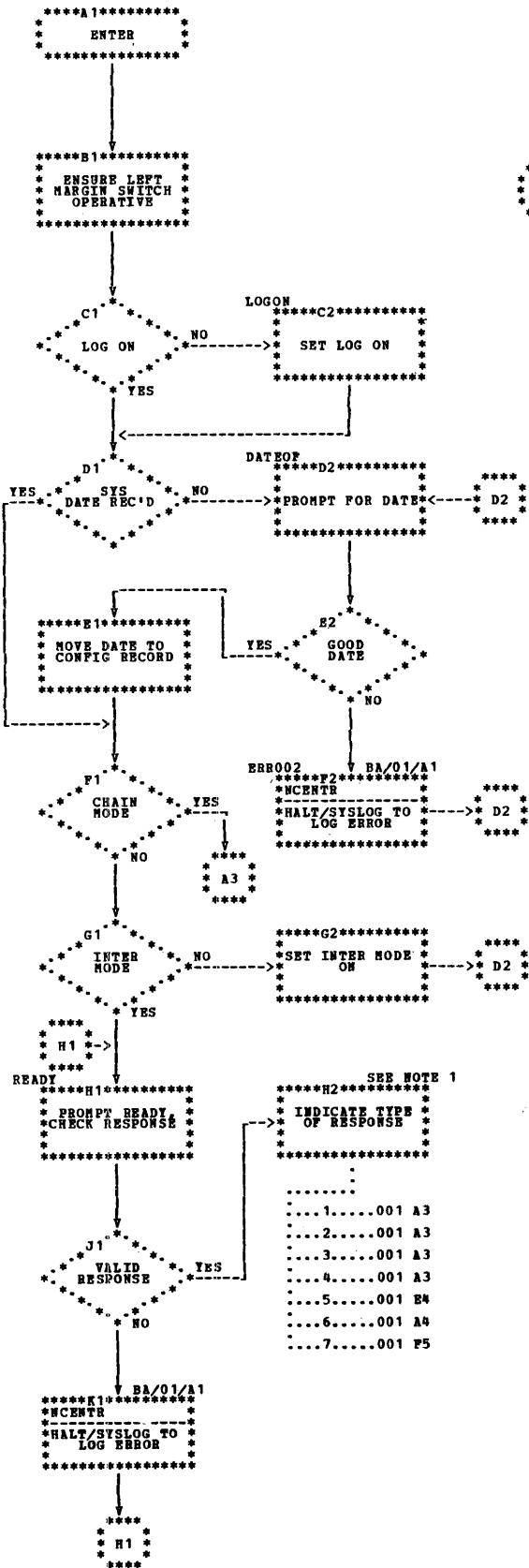


Chart EH (Part 2 of 2). Terminator Duplicate Key Scan Routine (\$\$TMDS)

JSRBIP



NOTE 1
 1..RESPONSE WAS CALL
 2..RESPONSE WAS LOAD
 3..RESPONSE WAS BUILD
 4..RESPONSE WAS BUILDC
 5..RESPONSE WAS LOG
 6..RESPONSE WAS READER
 7..RESPONSE WAS ENTER BAS

Chart FA Conversational Scheduler Initializer (\$\$RBIP)

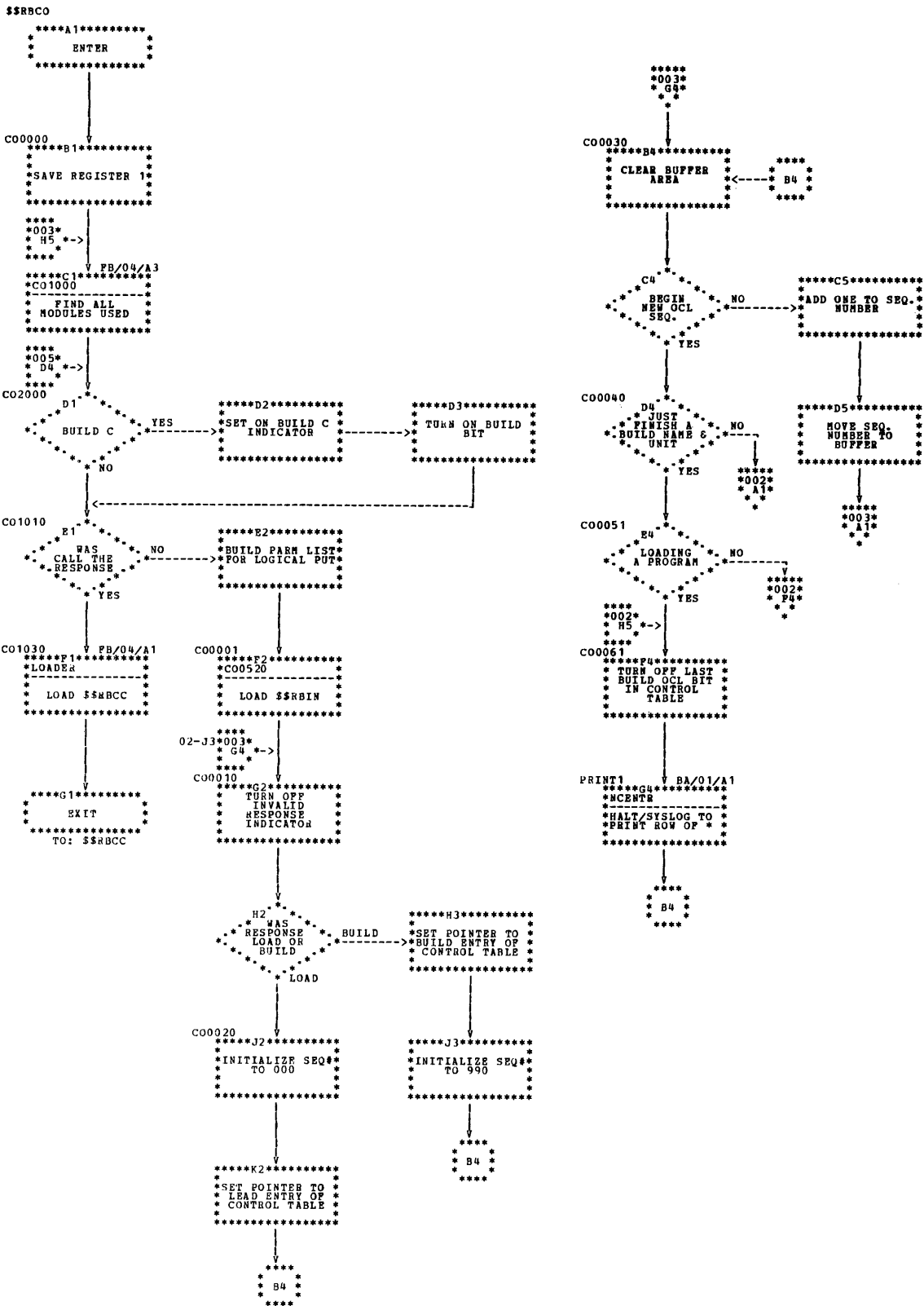


Chart FB (Part 1 of 5). Scheduler Control Routine (\$\$RBCO)

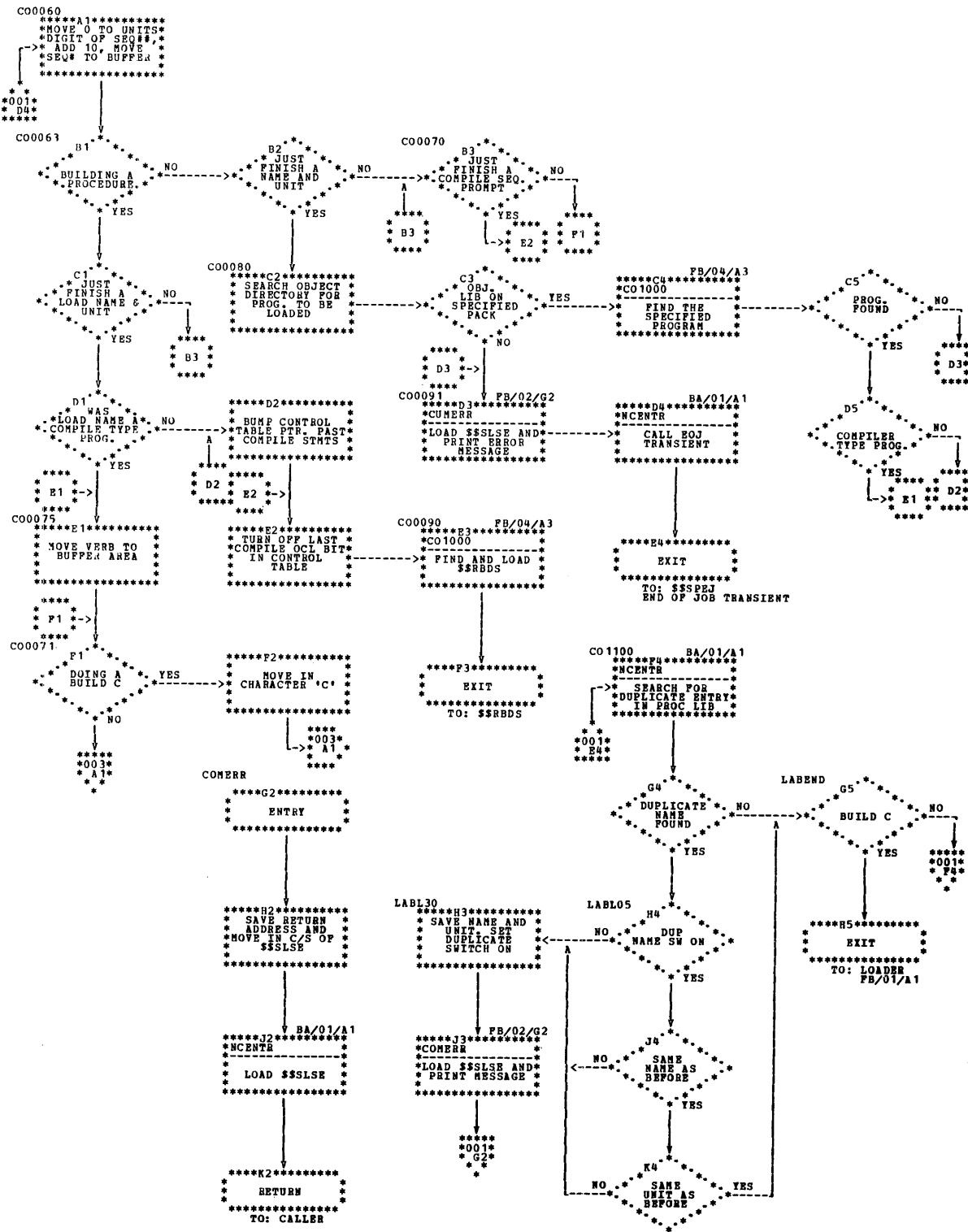


Chart FB (Part 2 of 5). Scheduler Control Routine (\$\$RBCO)

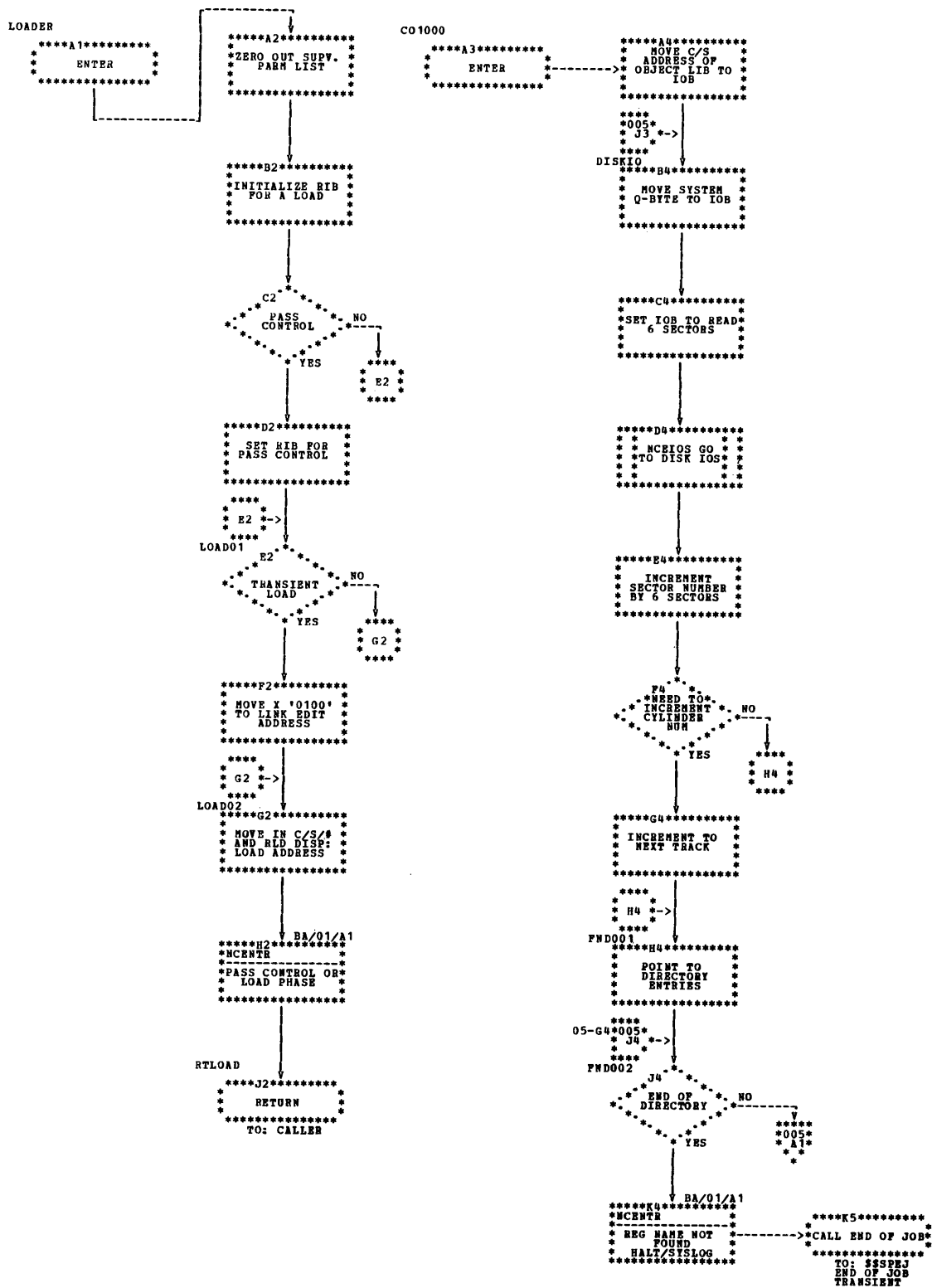


Chart FB (Part 4 of 5). Scheduler Control Routine (\$\$RBCO)

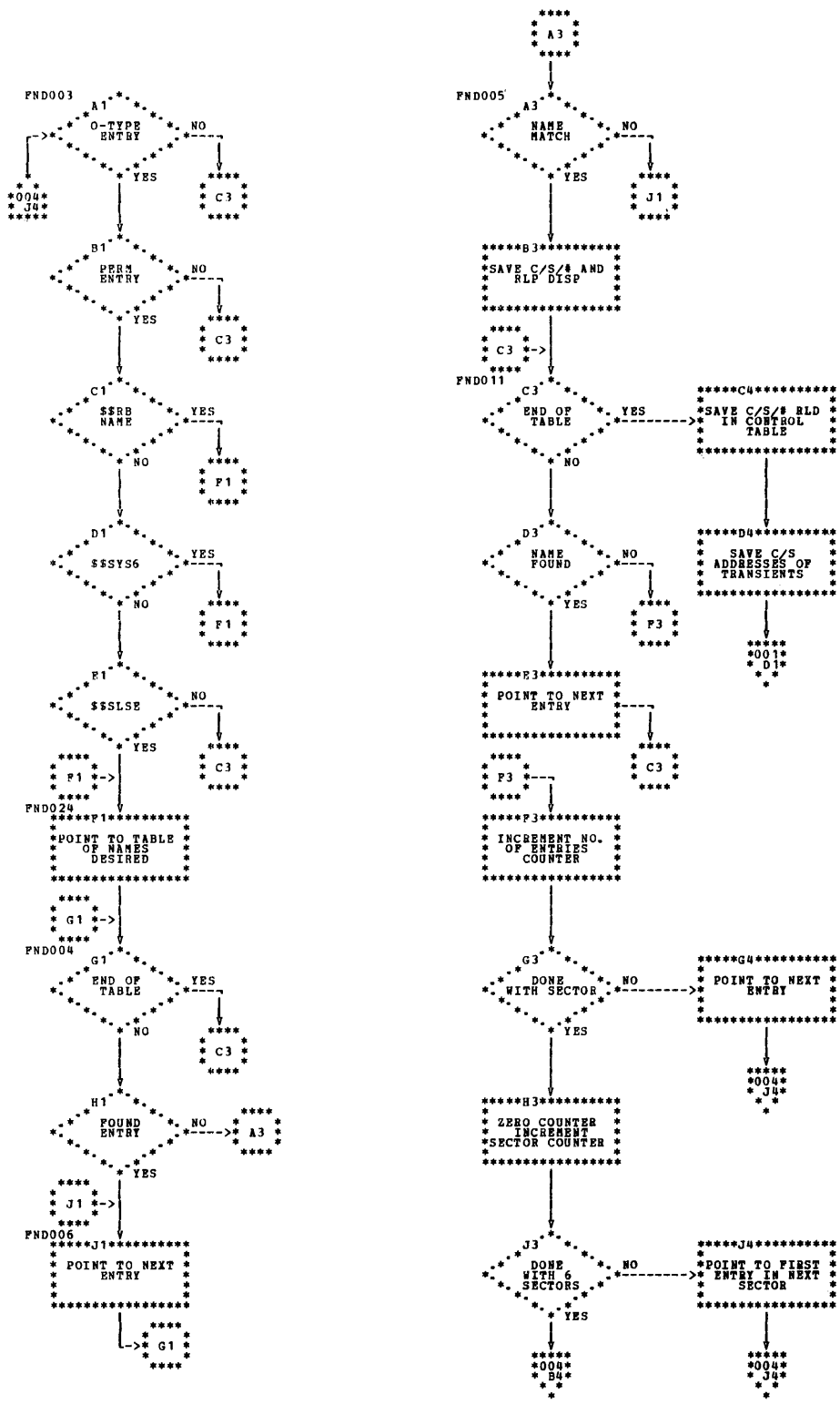


Chart FB (Part 5 of 5). Scheduler Control Routine (\$\$RBCO)

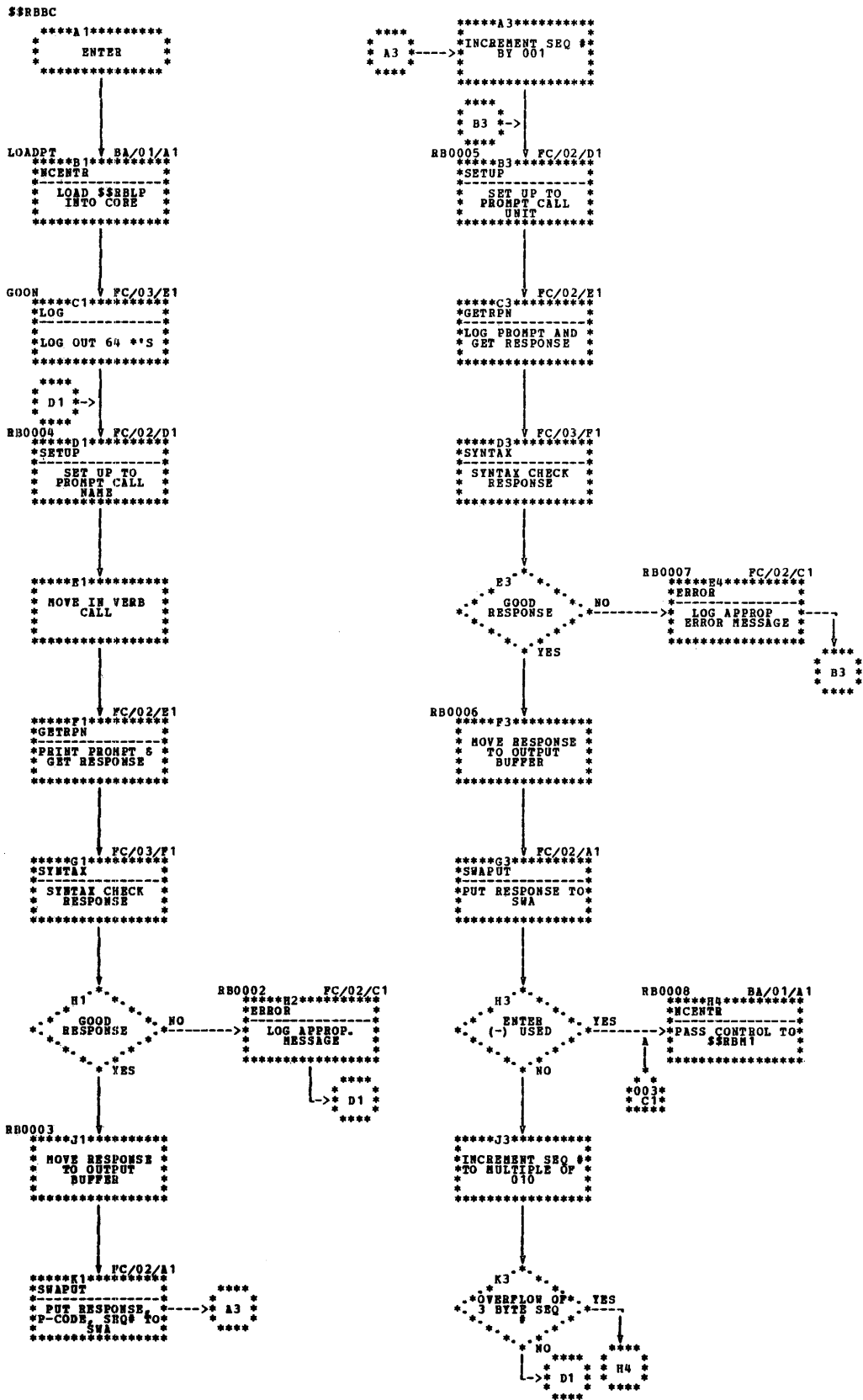


Chart FC (Part 1 of 3). Build-Chaind Cycle Routine (\$\$RBC)

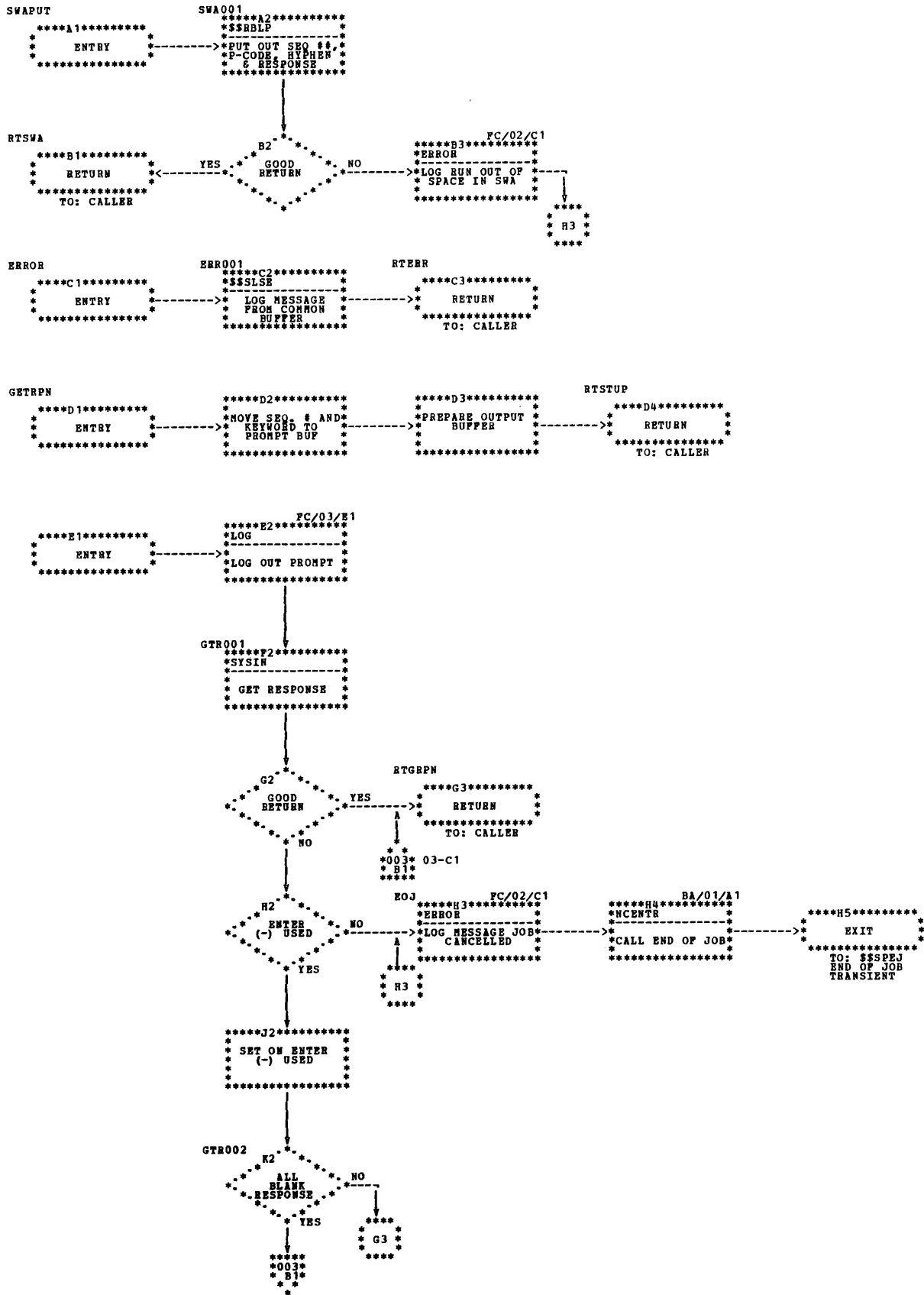


Chart FC (Part 2 of 3). Build-Chaind Cycle Routine (\$\$RBBC)

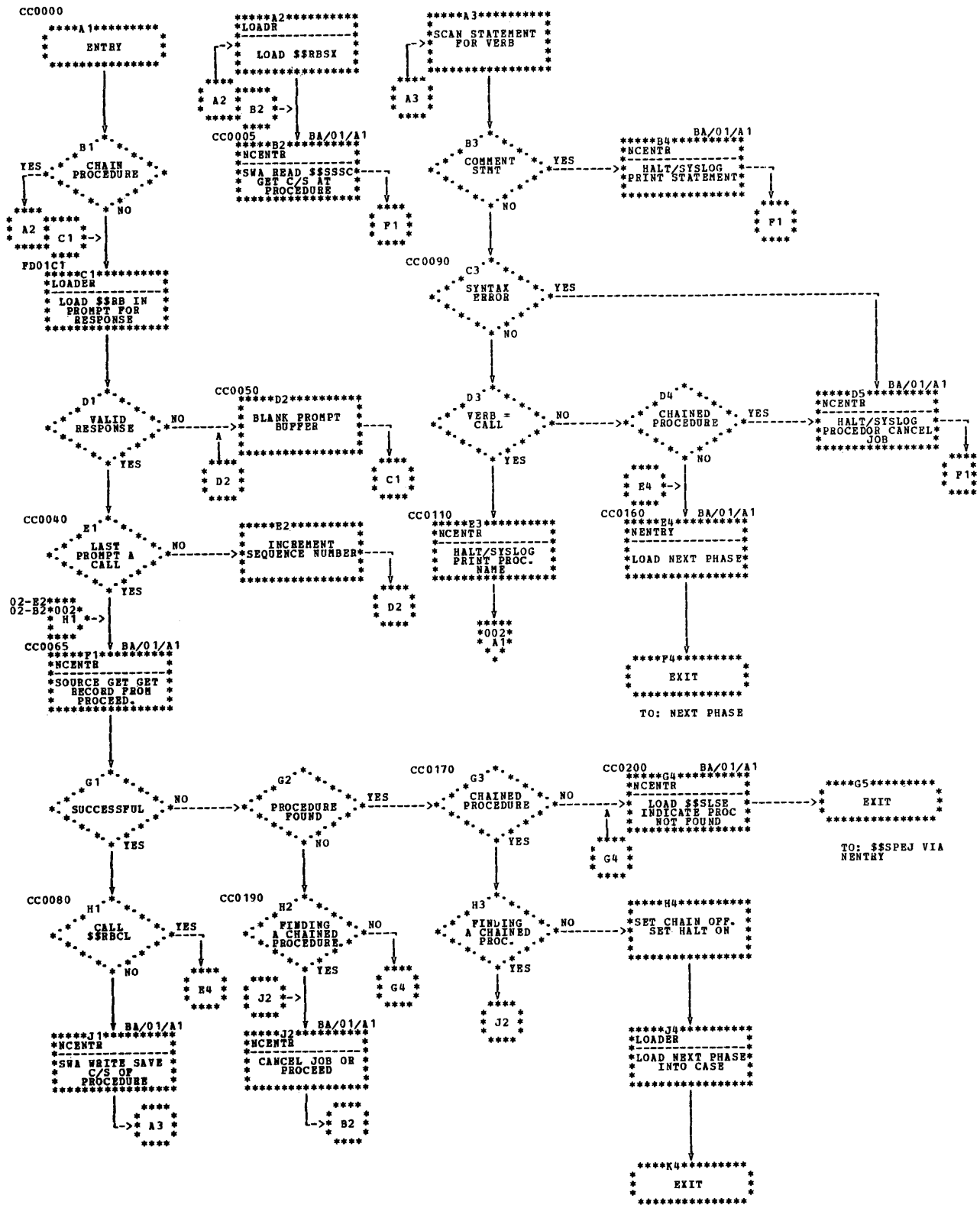


Chart FD (Part 1 of 2). Call Control Routine (\$\$RBCC)

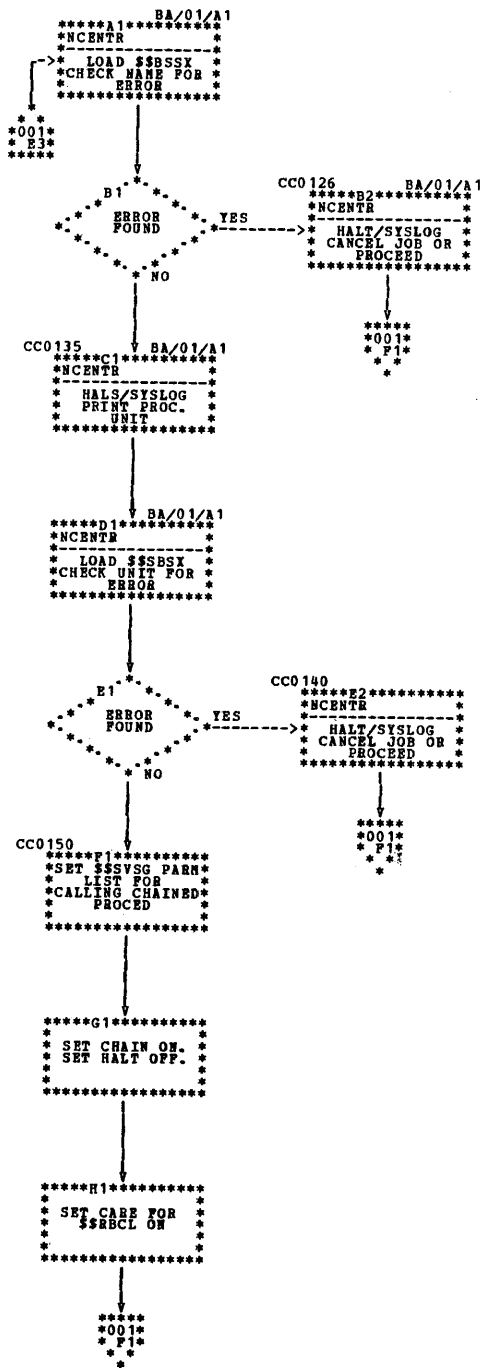


Chart FD (Part 2 of 2). Call Control Routine (\$\$RBCC)

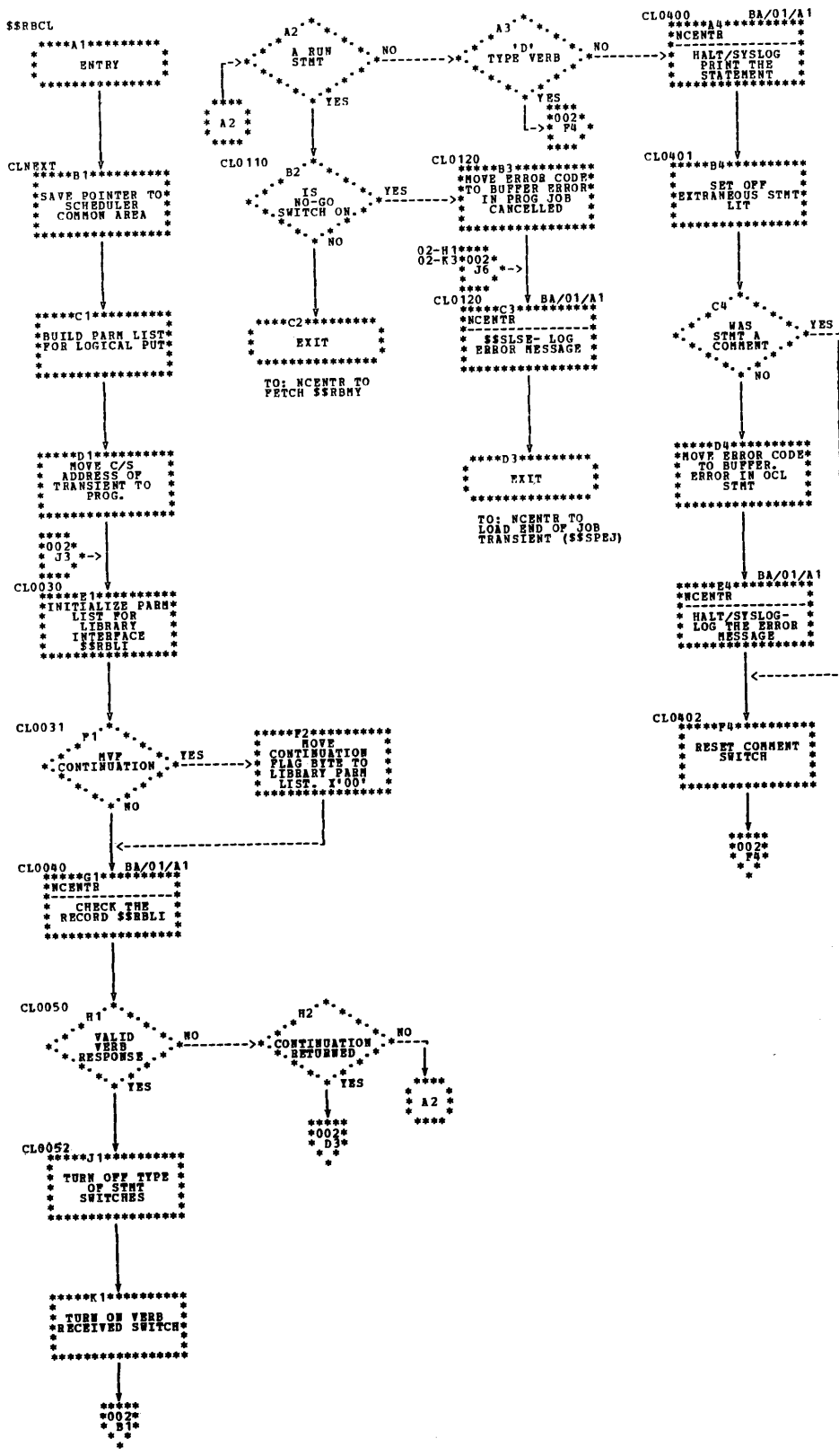


Chart FE (Part 1 of 2). Call Cycle Routine (\$\$RBCL)

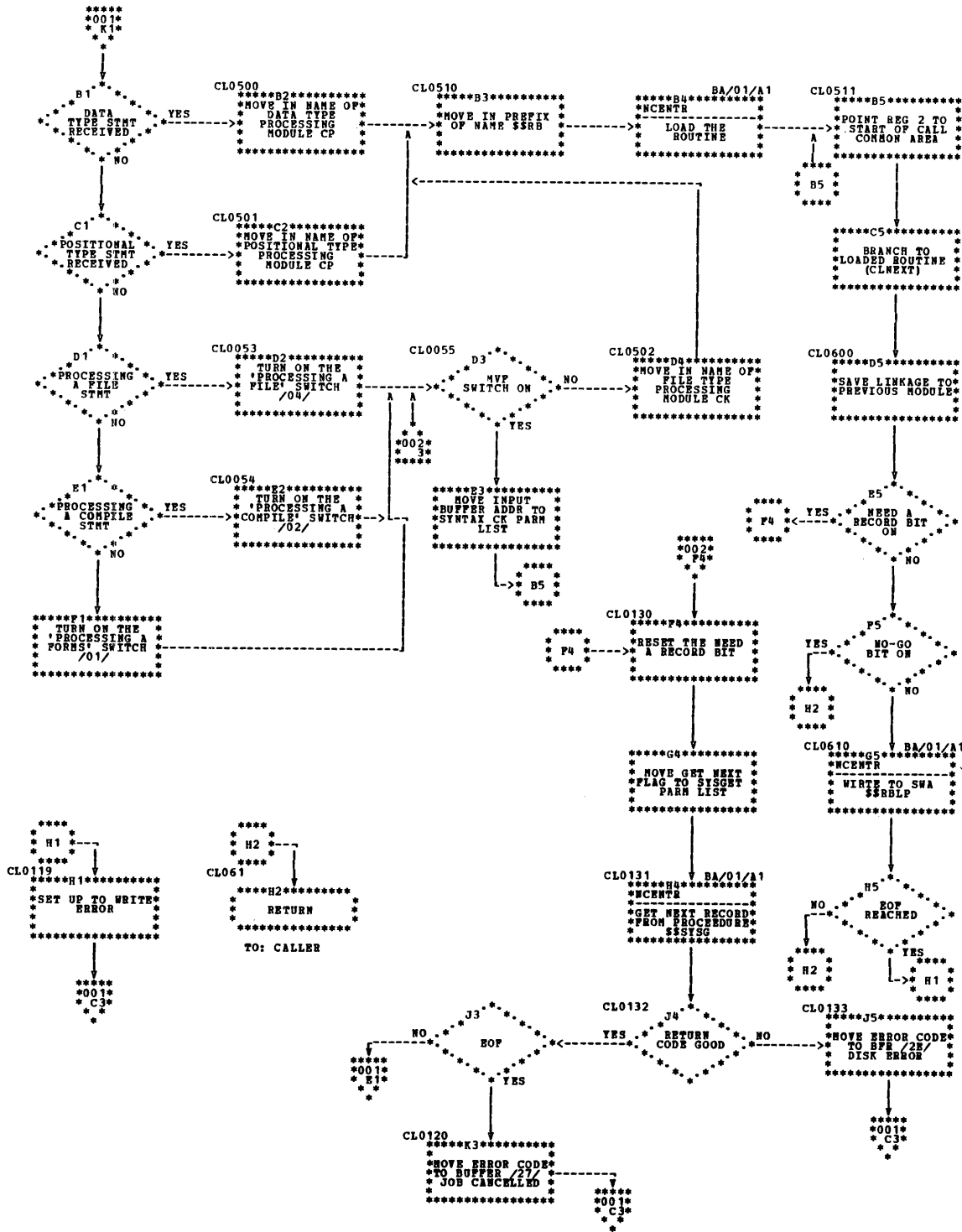


Chart FE (Part 2 of 2). Call Cycle Routine (\$\$RBCL)

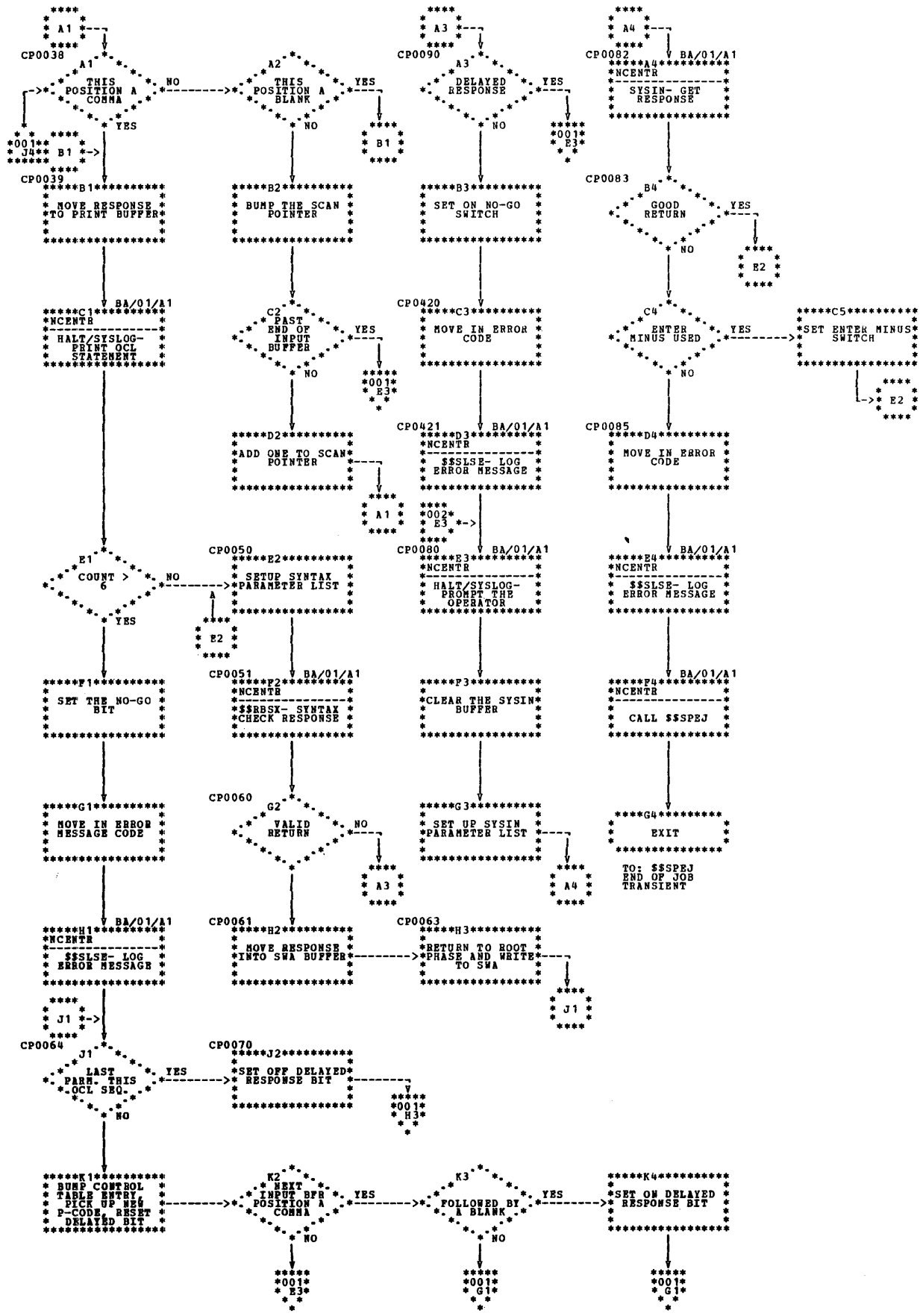


Chart FF (Part 2 of 2). Positional Statement Processor (\$\$RBCP)

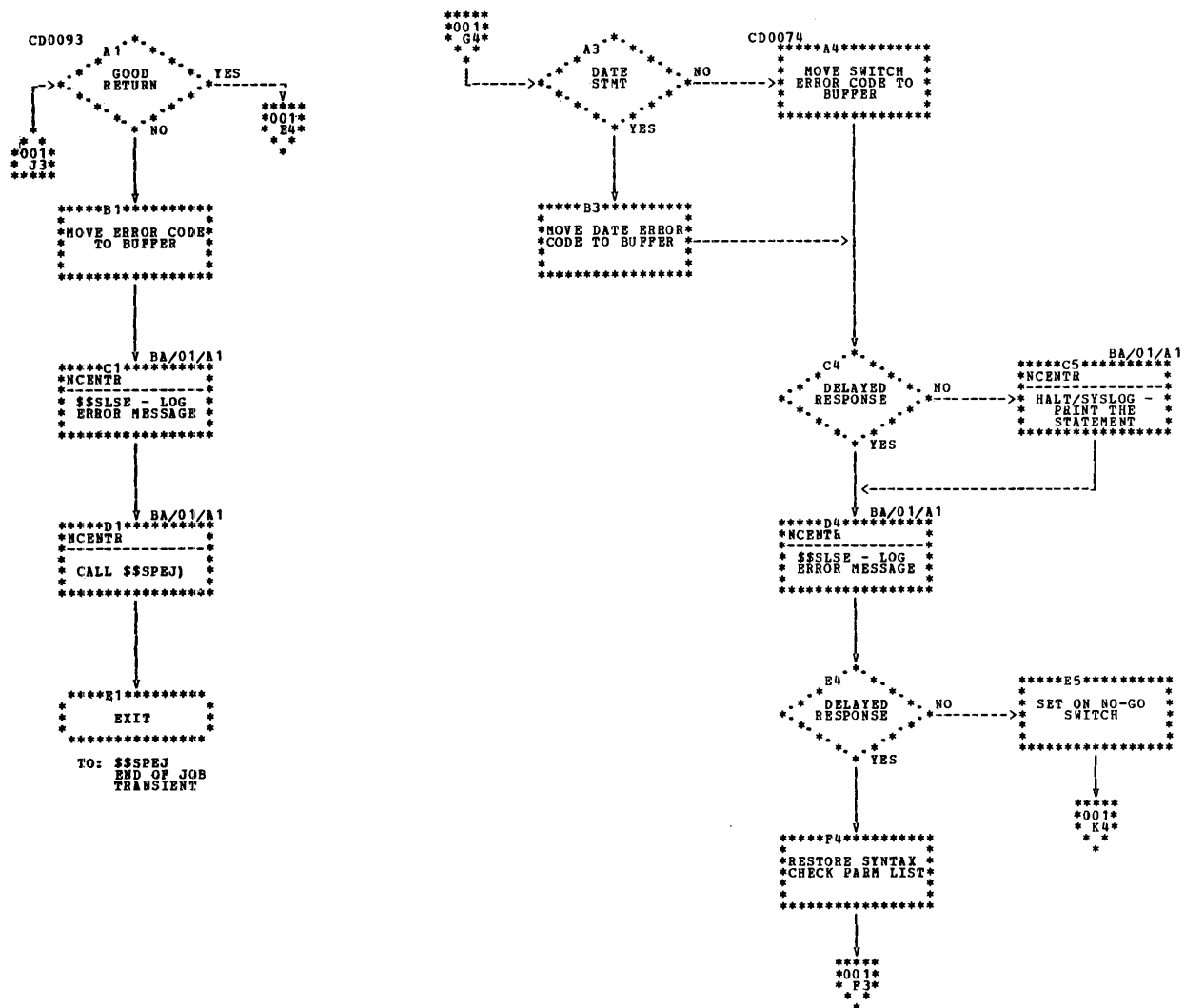


Chart FG (Part 2 of 2). Data Type Statement Processor (\$\$RBCD)

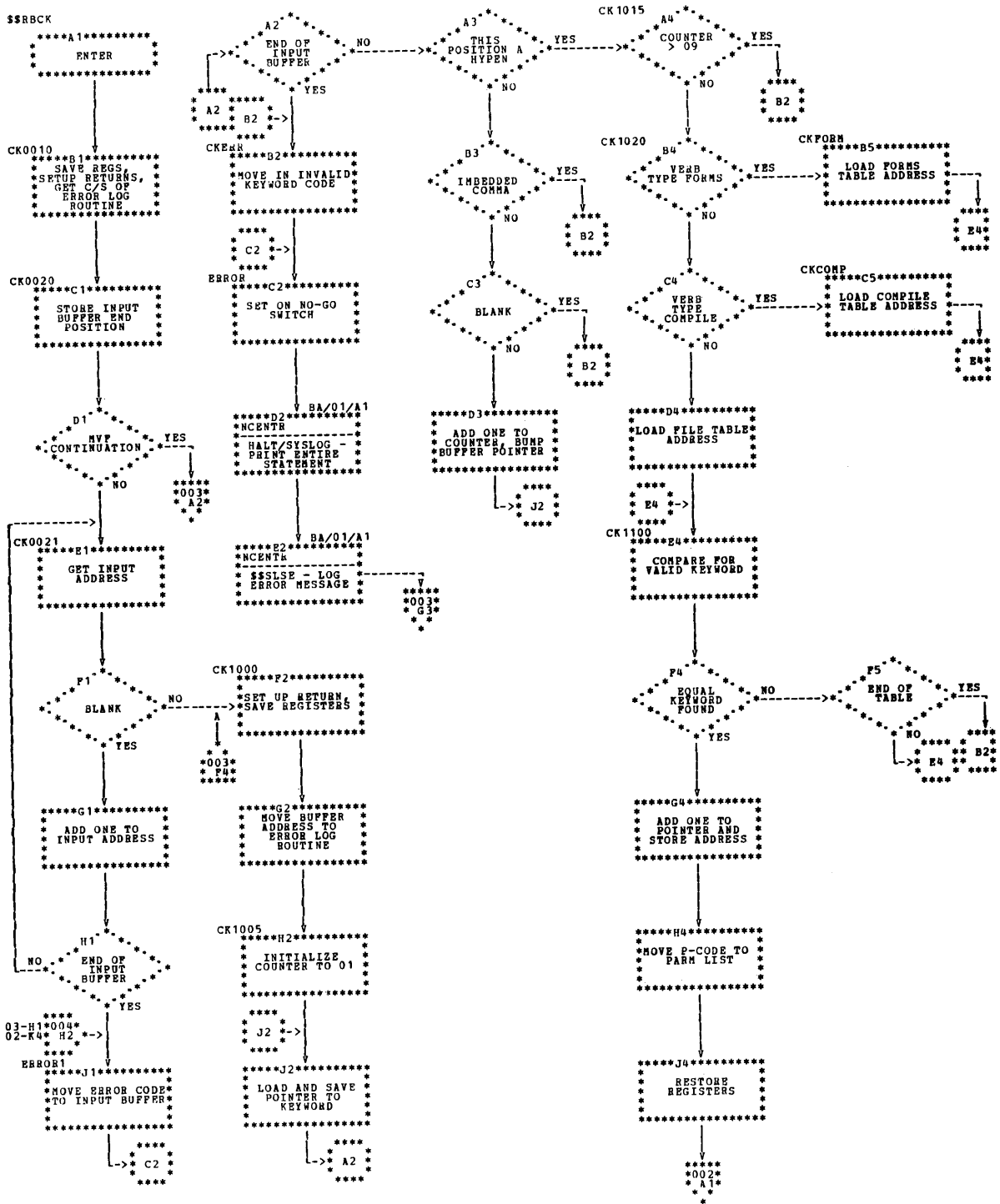


Chart FH (Part 1 of 4). Keyword Type Statement Processor (\$\$RBCK)

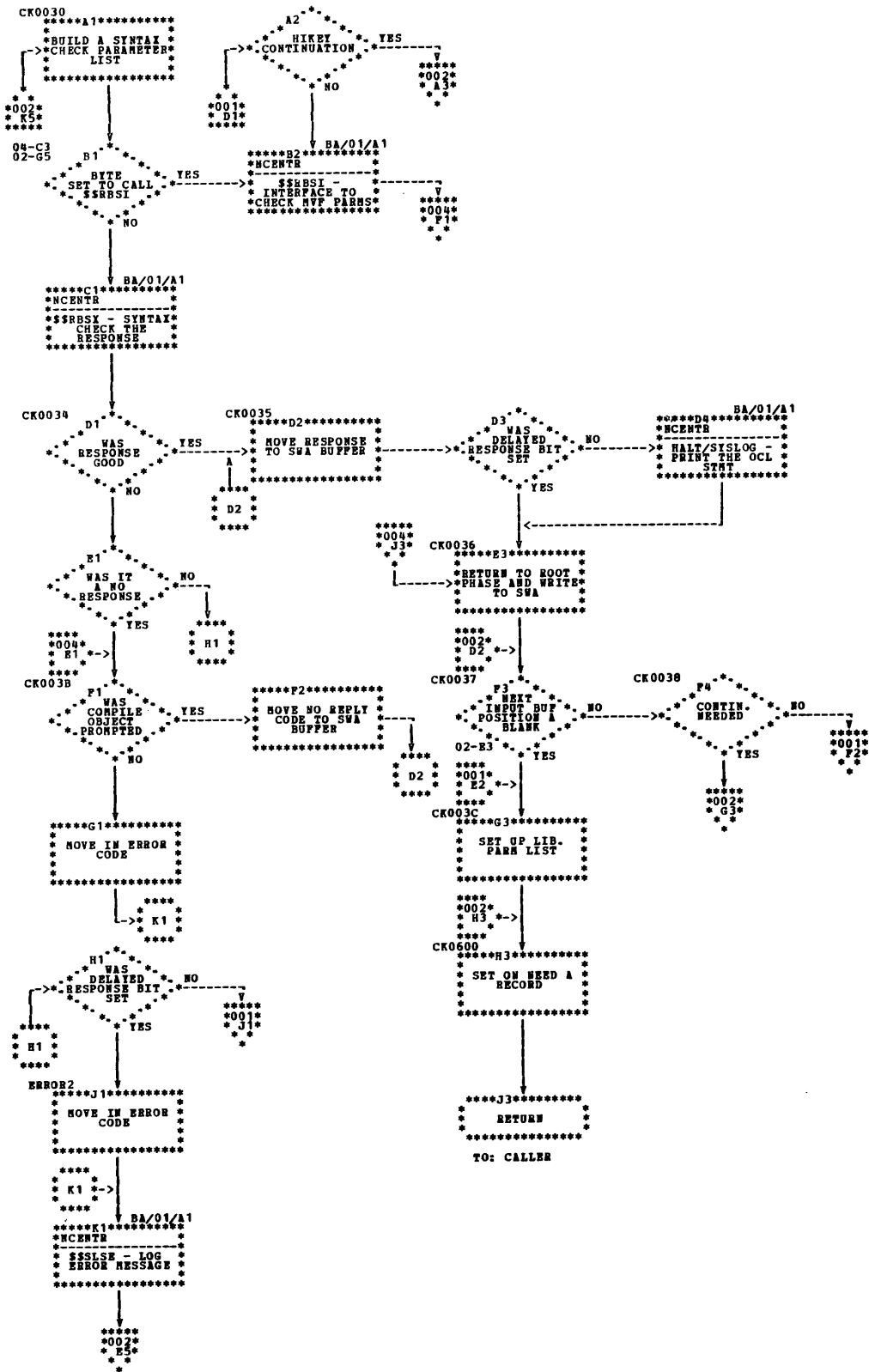


Chart FH (Part 3 of 4). Keyword Type Statement Processor (\$\$RBCK)

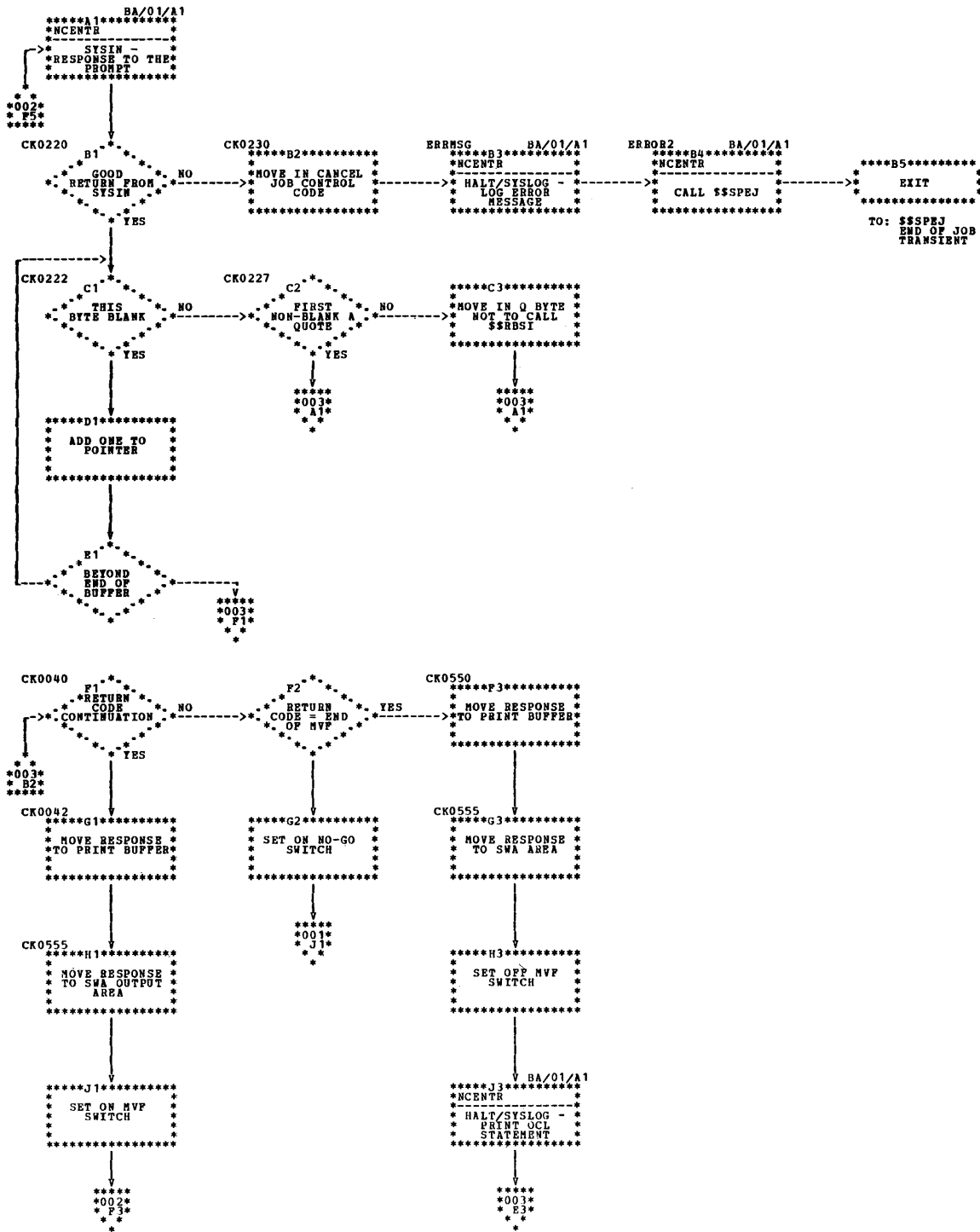


Chart FH (Part 4 of 4). Keyword Type Statement Processor (\$\$RBCK)

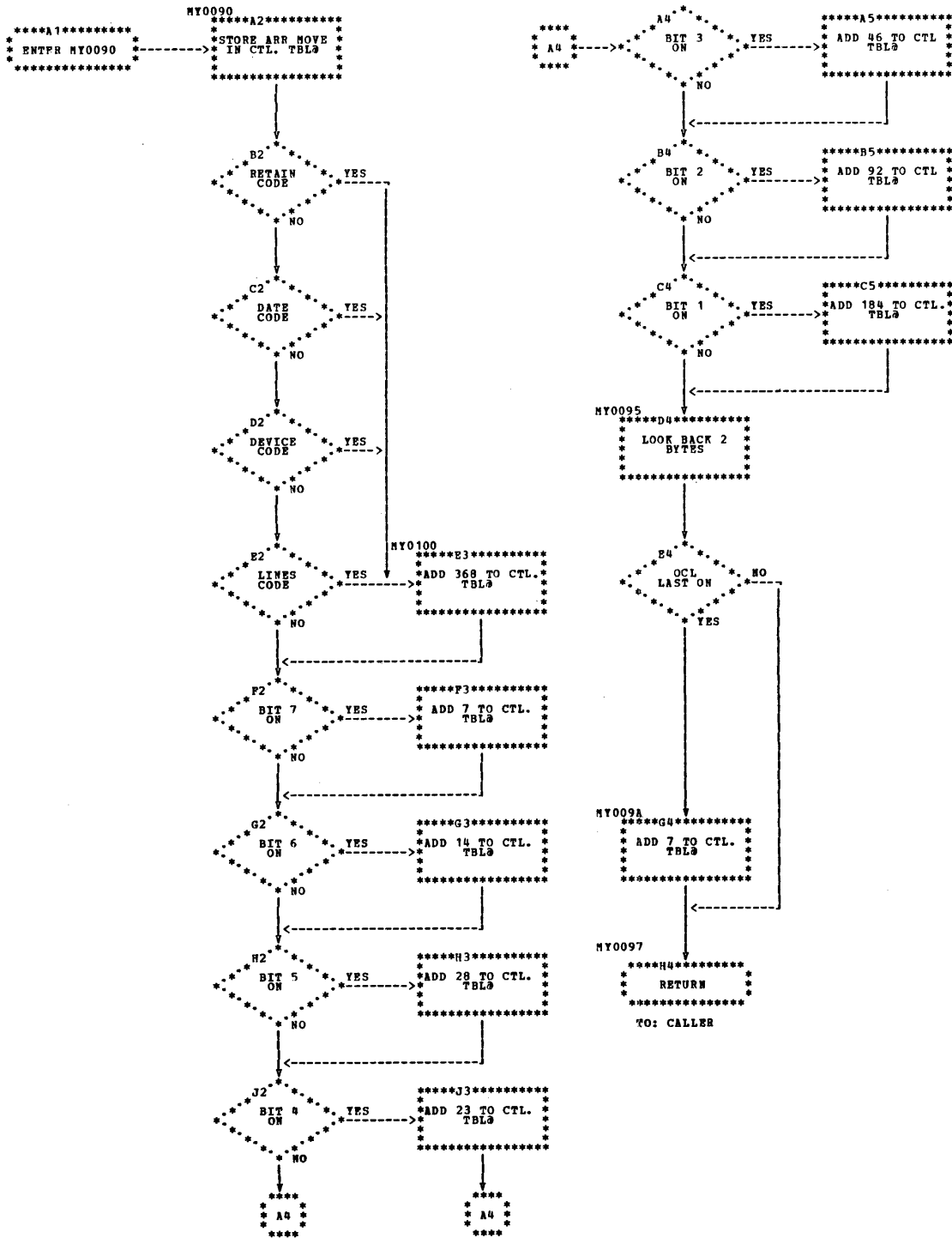


Chart FI (Part 2 of 4). Modify Routine (\$\$RBMV)

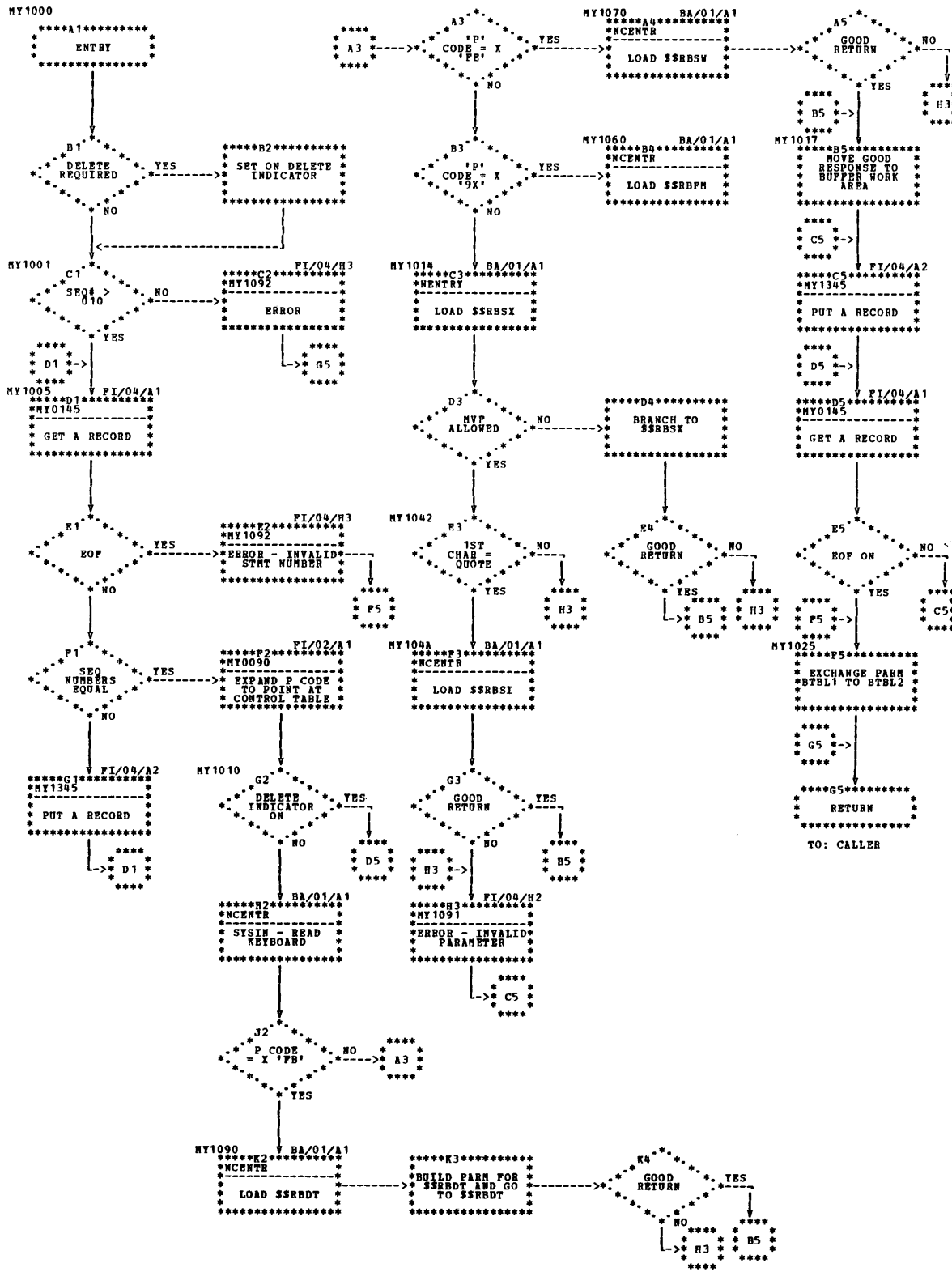


Chart FI (Part 3 of 4). Modify Routine (\$\$RBM)

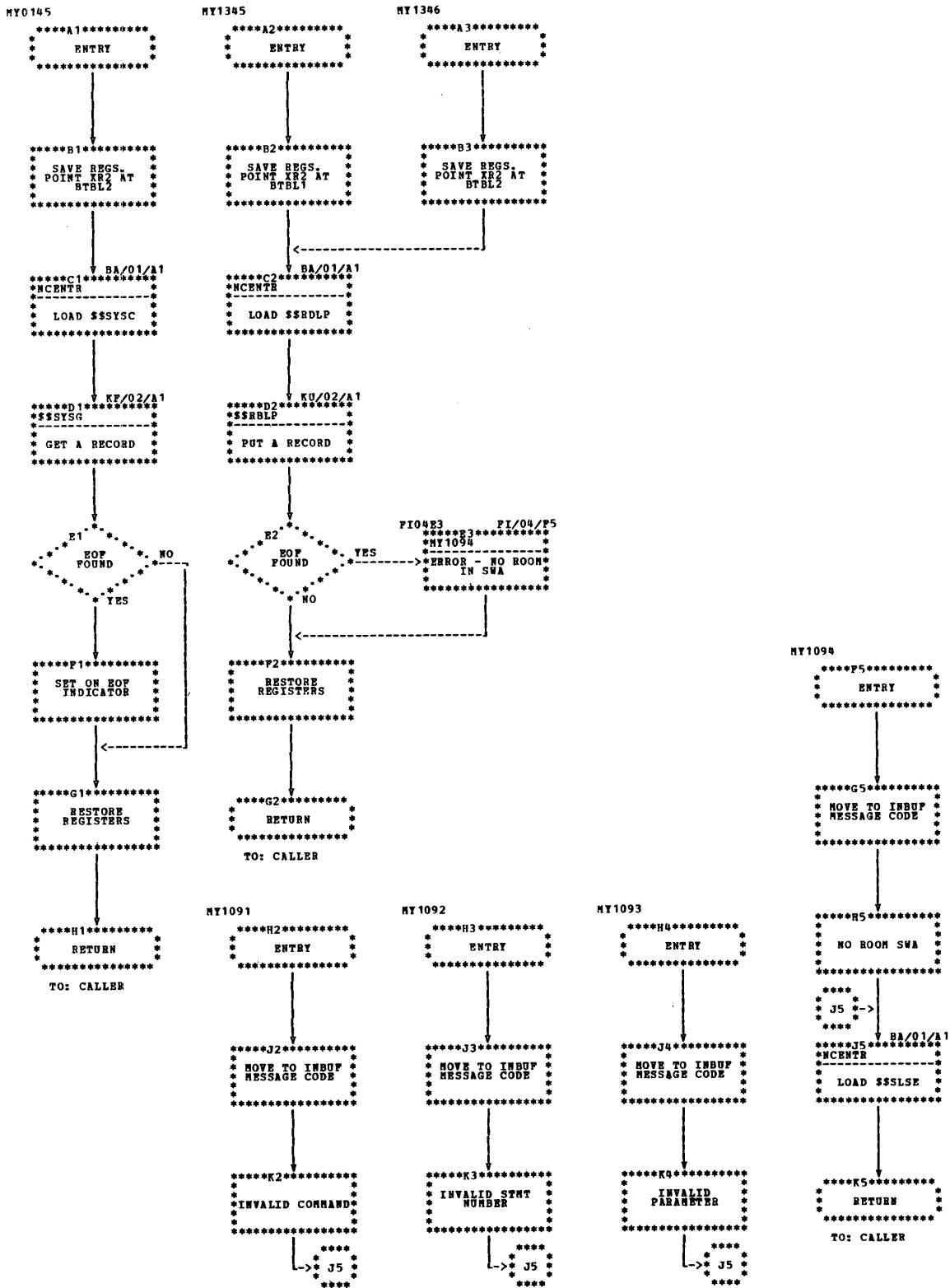


Chart FI (Part 4 of 4). Modify Routine (\$\$RBMY)

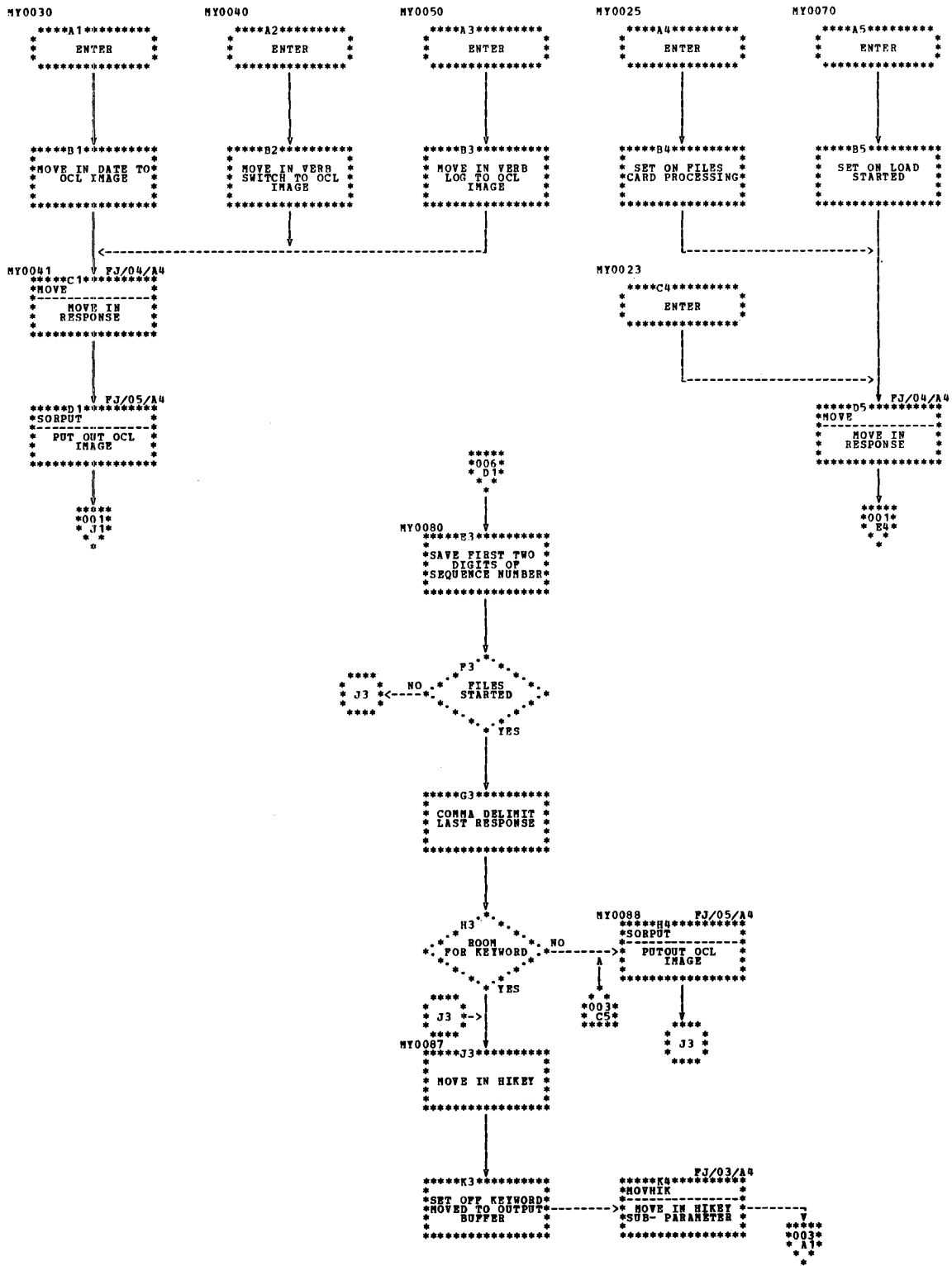


Chart FJ (Part 2 of 6). Procedure Library Scheduler Interface Routine (\$\$RBM1)

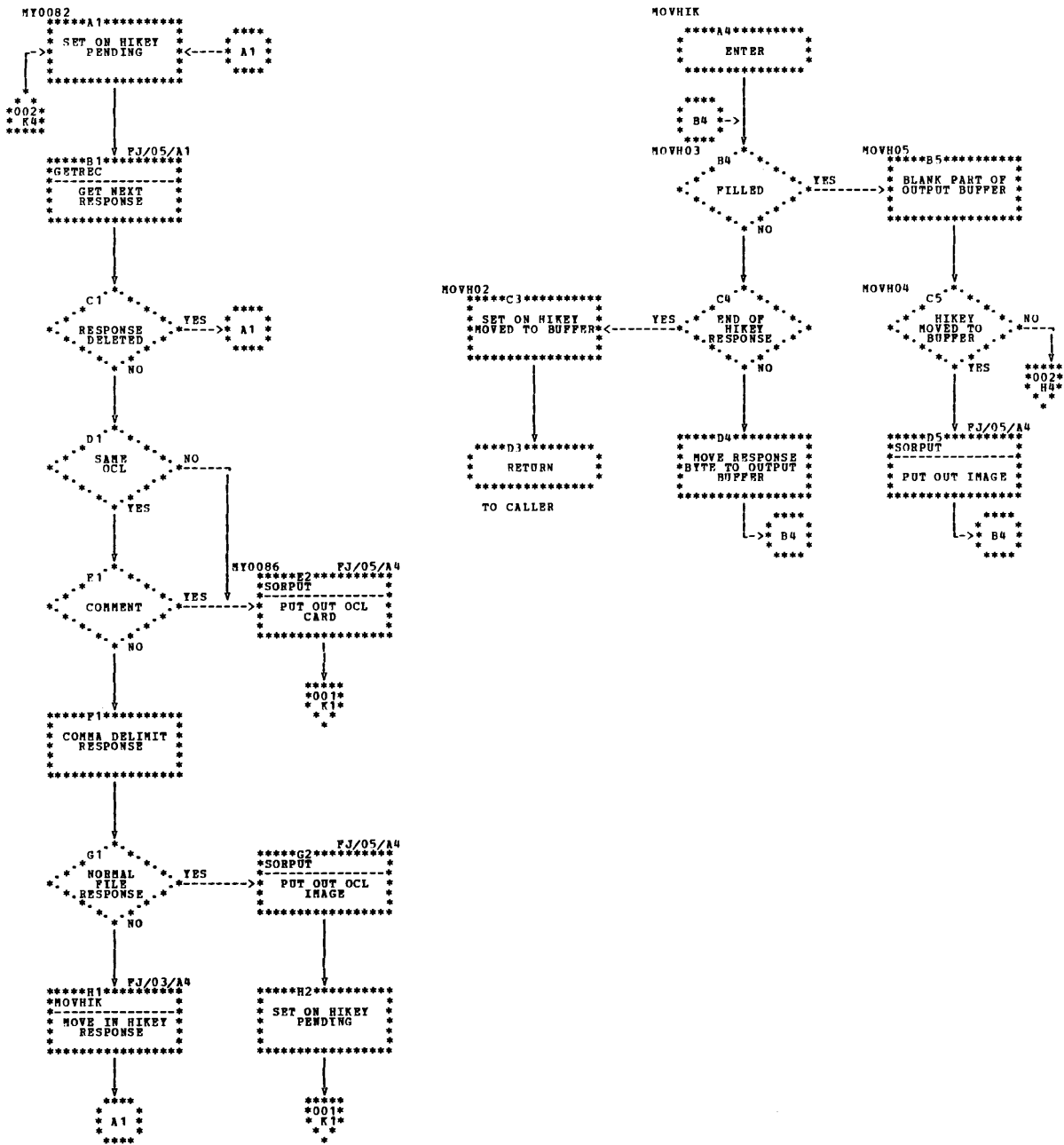


Chart FJ (Part 3 of 6). Procedure Library Scheduler Interface Routine (\$\$RBM1)

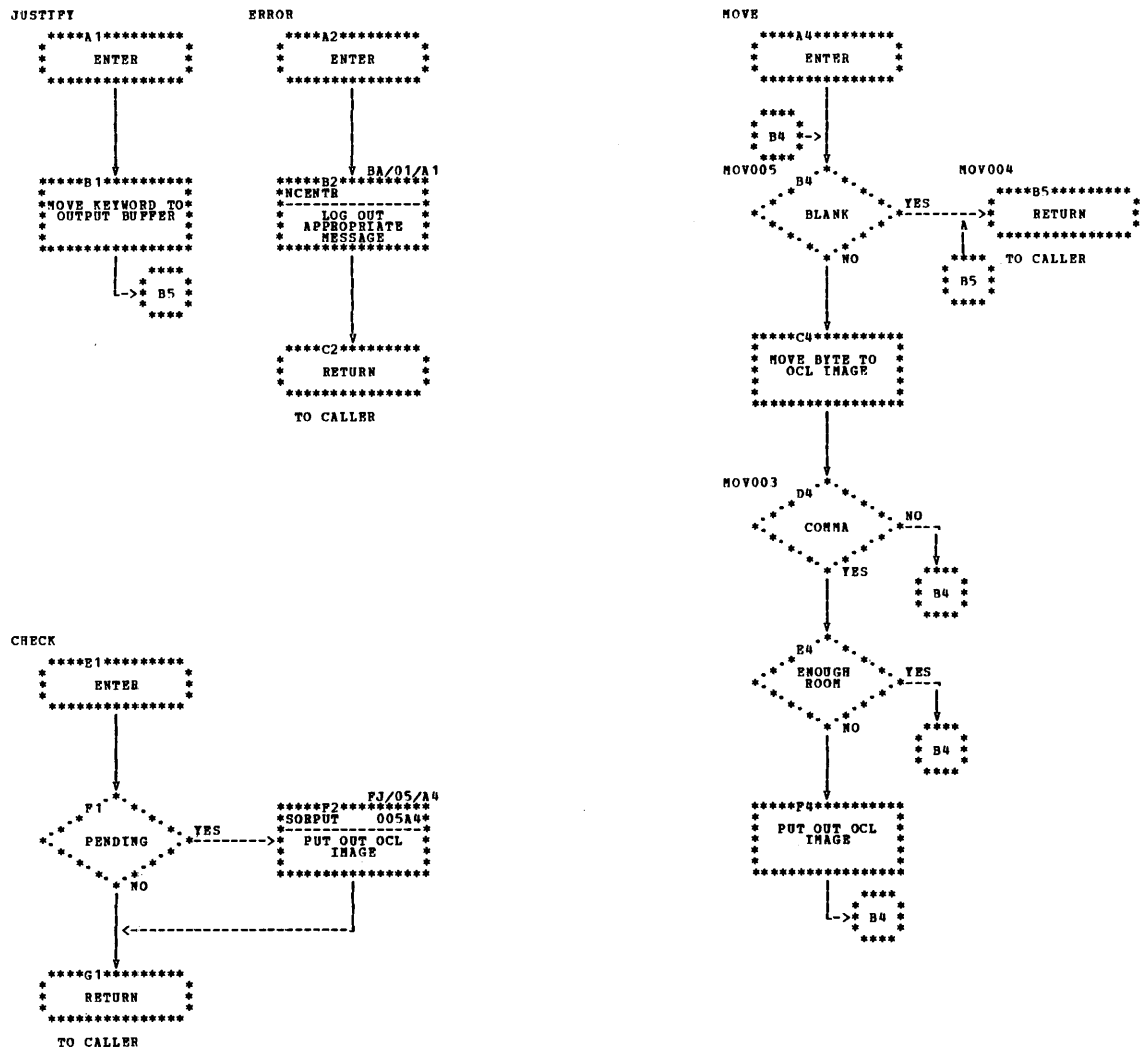


Chart FJ (Part 4 of 6). Procedure Library Scheduler Interface Routine (\$\$RBM1)

GETREC

*****A1*****
* ENTER *

V BA/01/A1
*****B1*****
* NCENTR *
* READ RECORD *
* FROM SWA *

C1
GOOD
RETURN

*****D1*****
* RETURN *
* TO CALLER *

NYEND FJ/05/E1
*****C2*****
* CHECK *
* CHECK FOR OCL *
* PENDING *

D2
BUILD
DC

V FJ/05/A4
*****E2*****
* SORPUT *
* PUT OUT //ROW *

NYE003 F2
INCLUDE
STATEMENTS

FJ/05/A4
*****G2*****
* SORPUT *
* CLOSE OUT ENTRY *

BA/01/A1
*****H2*****
* NCENTR *
* CALL END OF JOB *

*****J2*****
* EXIT *
* TO: \$\$\$PEJ *

NYE001 F3
INITIALIZE FOR
\$\$\$RB2

BA/01/A1
*****G3*****
* NCENTR *
* CALL \$\$\$RB2 *

*****H3*****
* EXIT *
* TO: \$\$\$RB2 *

SORPUT

*****A4*****
* ENTER *

V KK/01/A1
*****B4*****
* \$\$\$YSP *
* COPY TO SOURCE *
* LIBRARY *

SOR003 C4
GOOD
RETURN

FJ/04/A2
*****D5*****
* ERROR *
* LOG RUN OUT OF *
* SWA *

-> H2

D4
DUPLICATE
ENTRY

V FJ/04/A2
*****E4*****
* ERROR *
* LOG DUPLICATE *
* ENTRY DELETED *

SOR001 F4
BLANK OUTPUT
BUFFER AND MOVE
IN //

G4
SET OFF
LOAD

H4
RETURN
TO CALLER

Chart FJ (Part 5 of 6). Procedure Library Scheduler Interface Routine (\$\$RBM1)

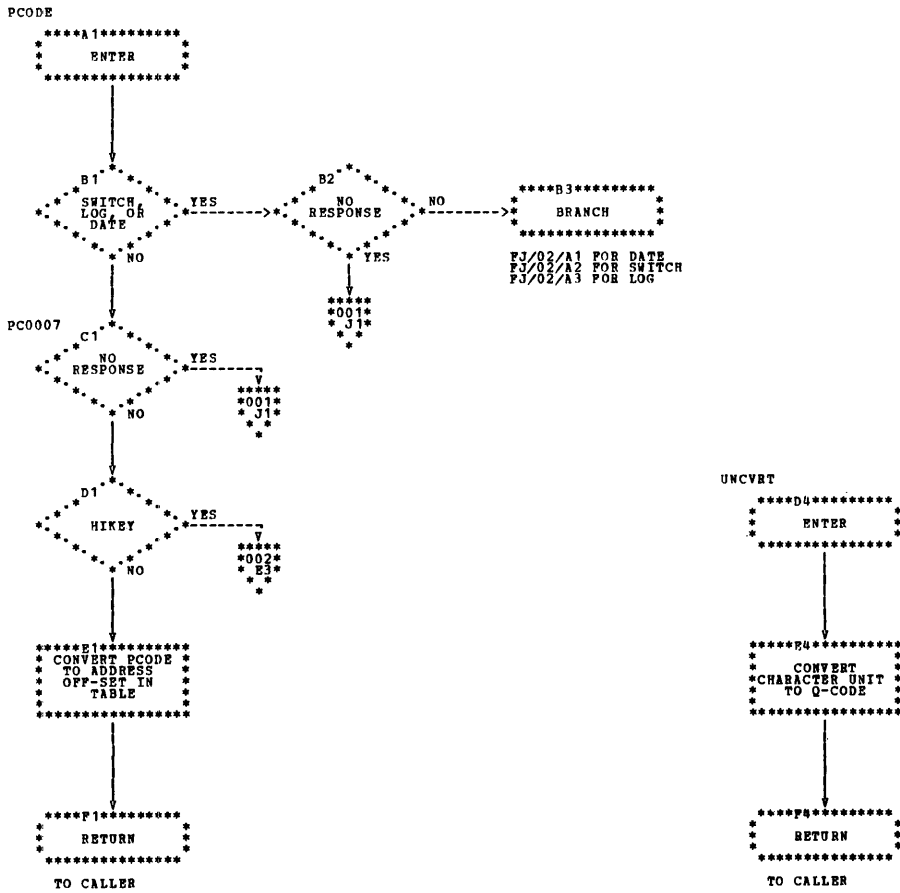


Chart FJ (Part 6 of 6). Procedure Library Scheduler Interface Routine (\$\$RBM1)

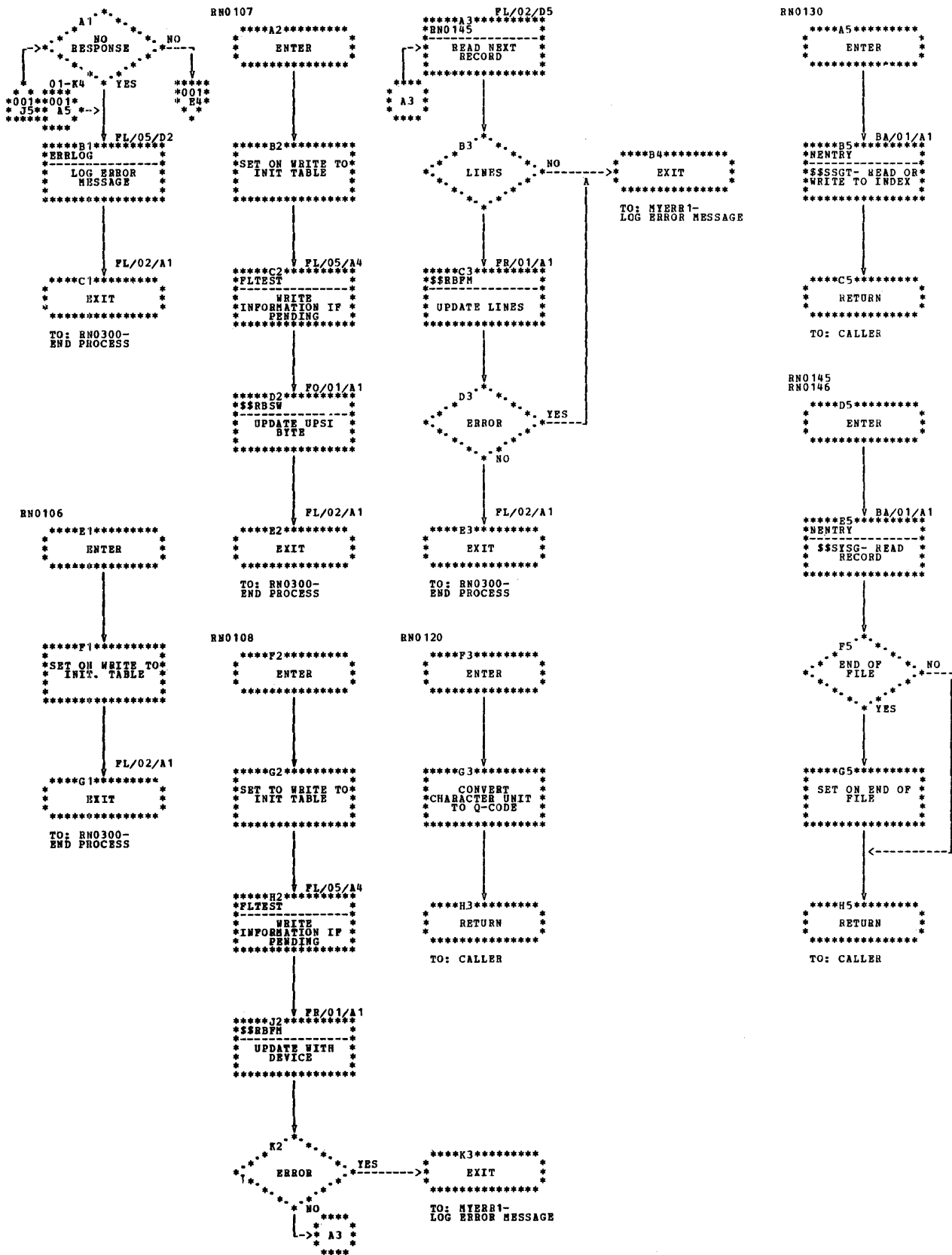
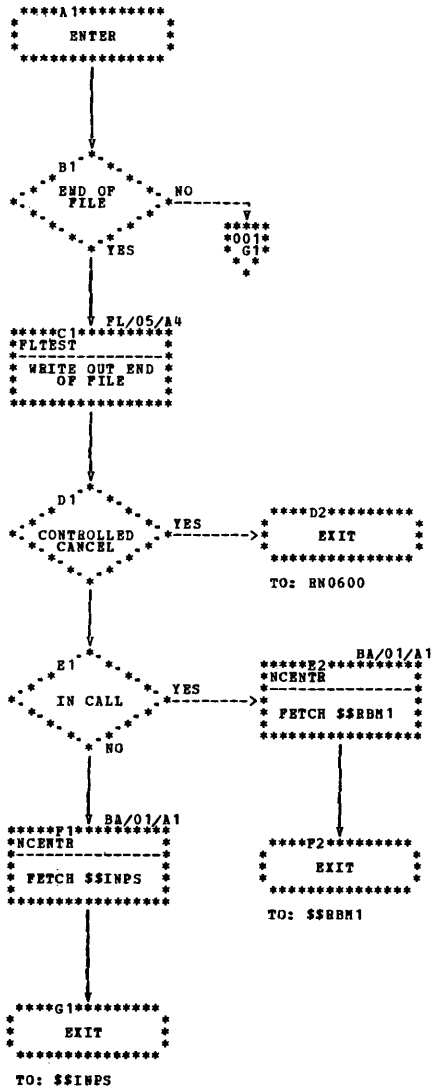


Chart FL (Part 2 of 5). Run Processor (\$RBRN)

RN0300



BBF001

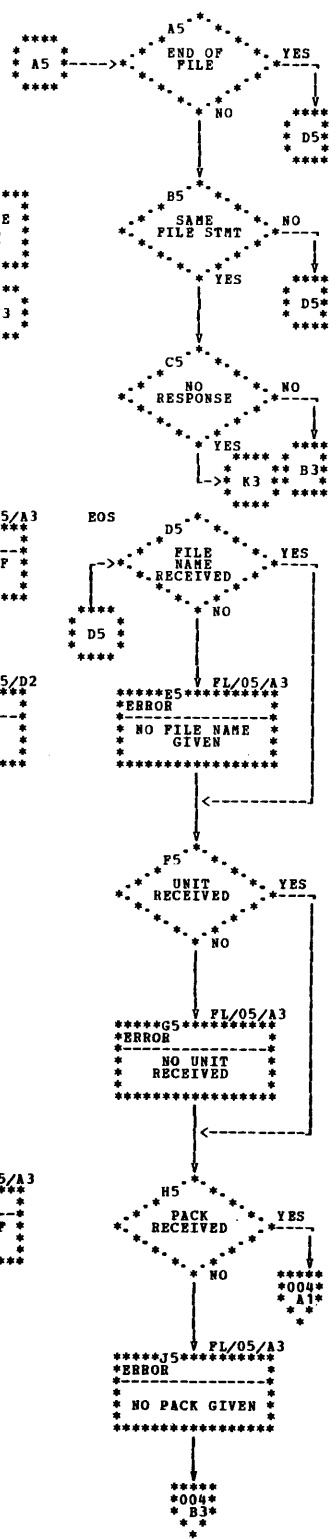
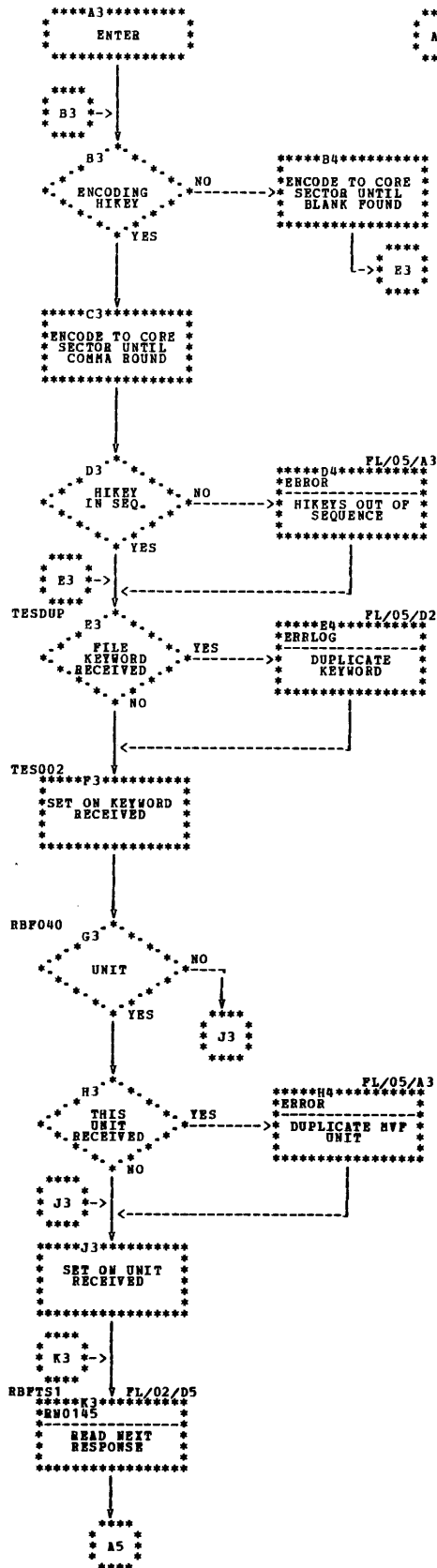


Chart FL (Part 3 of 5). Run Processor (\$\$RBRN)

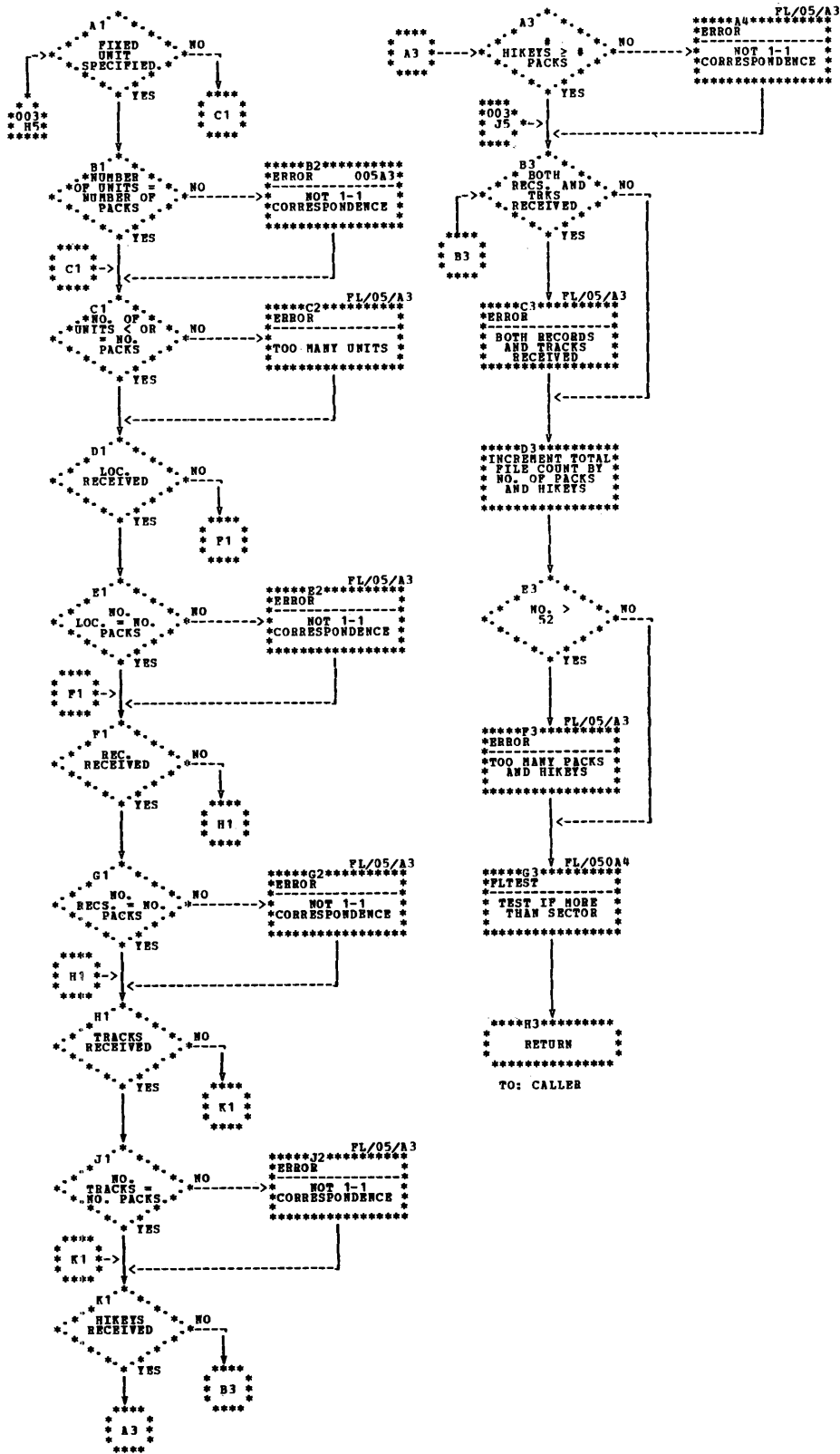


Chart FL (Part 4 of 5). Run Processor (\$RBRN)

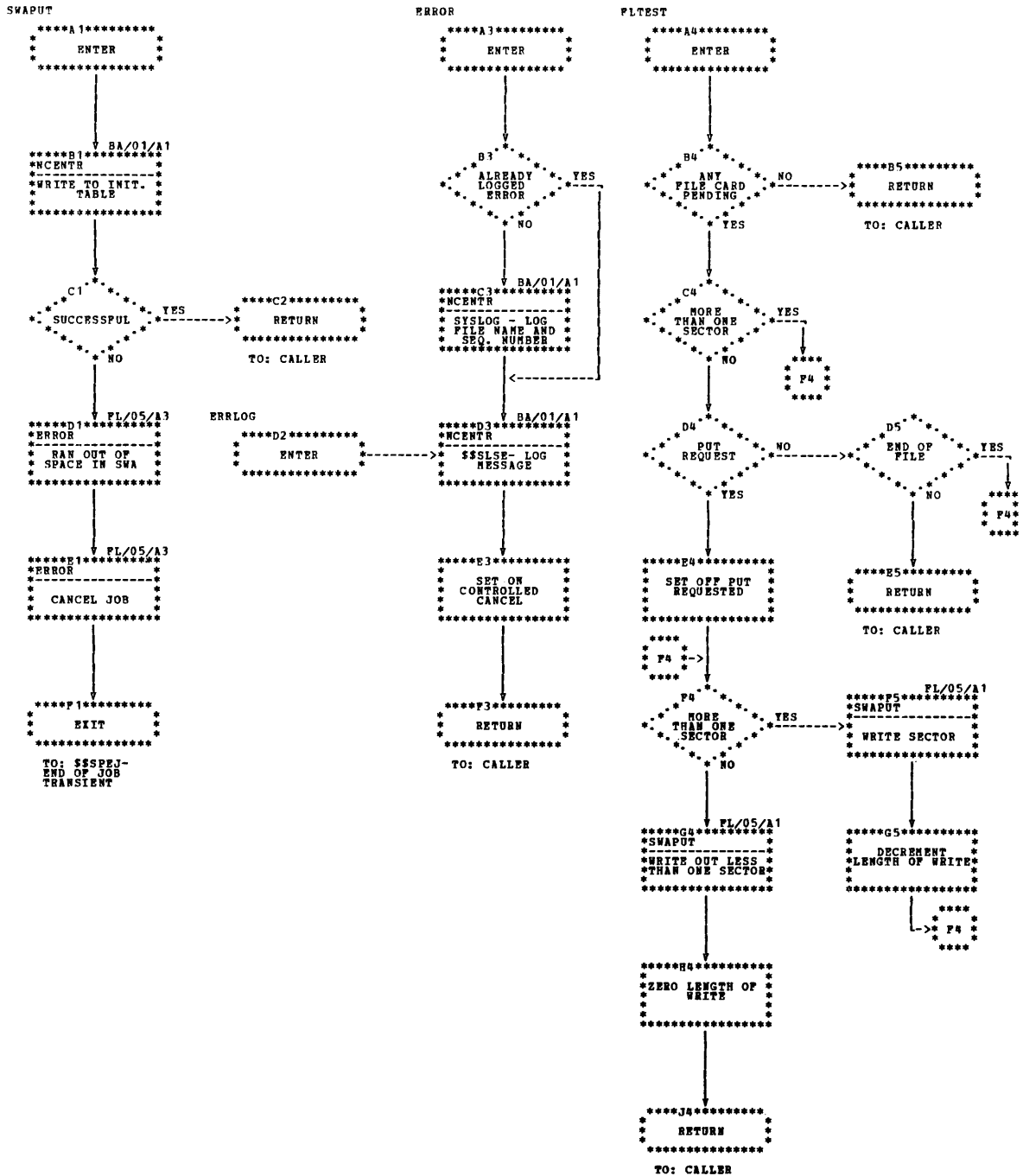


Chart FL (Part 5 of 5). Run Processor (\$\$RBRN)

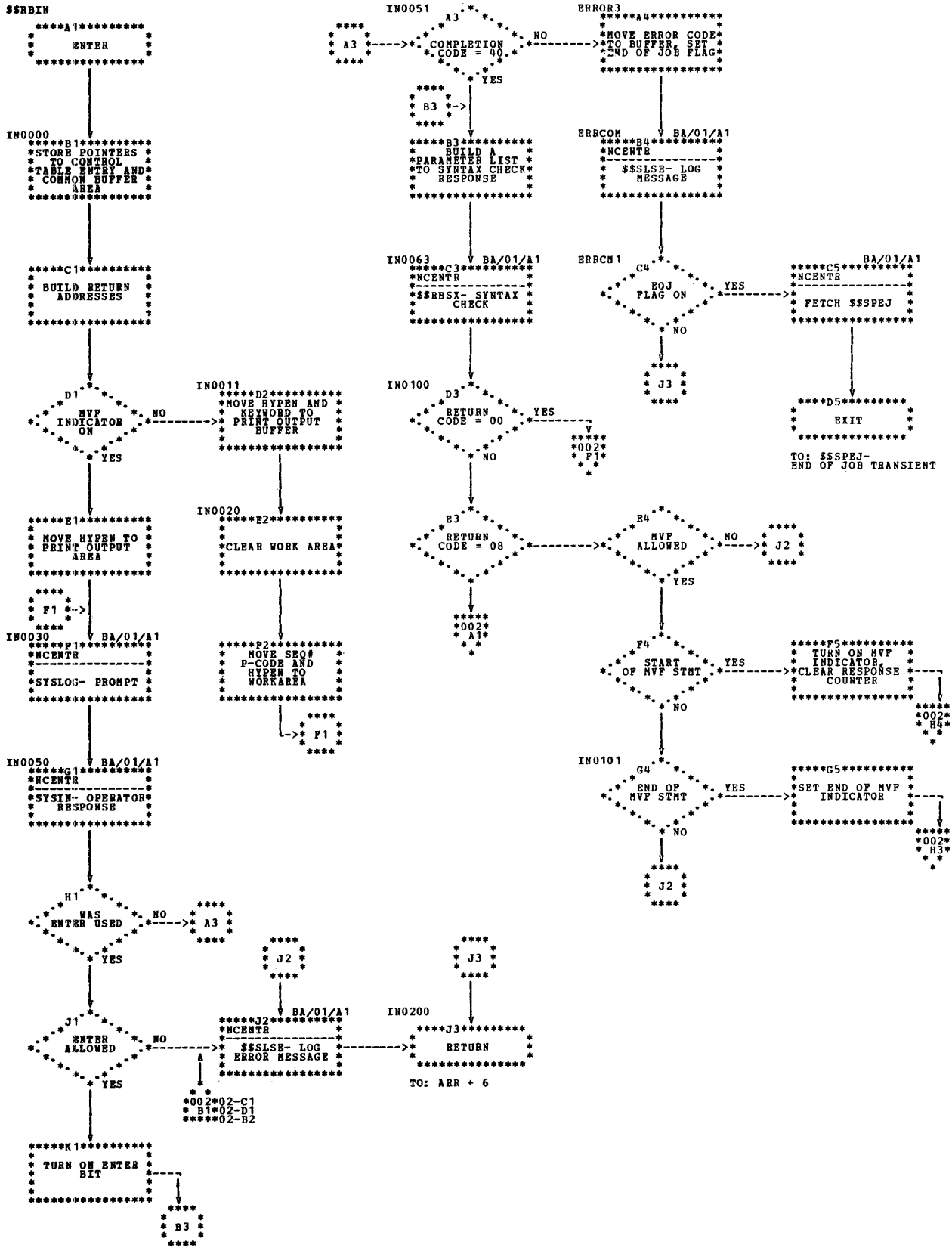


Chart FM (Part 1 of 2). Interaction Routine (\$\$RBIN)

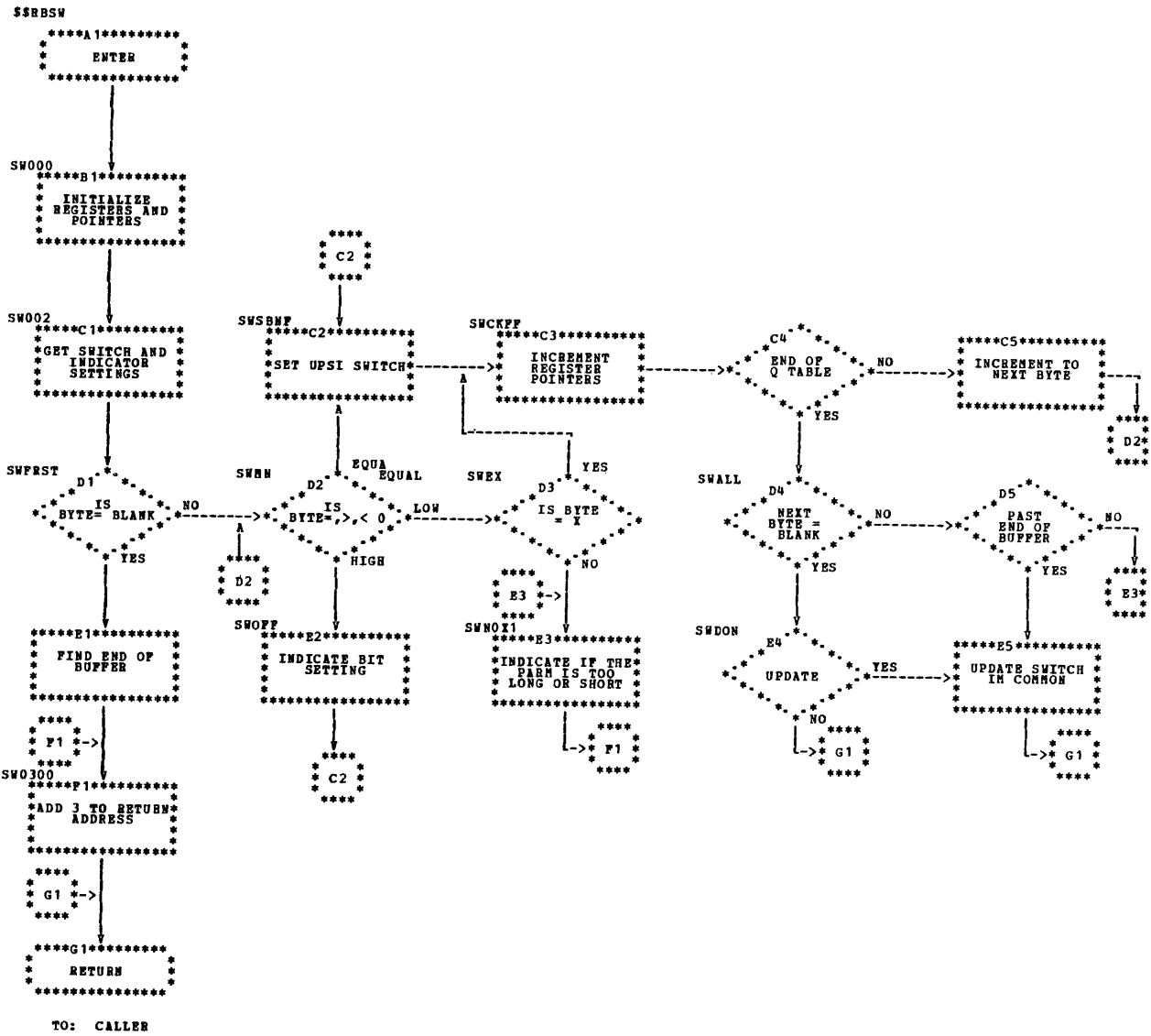


Chart FO. Switch Control Statement Processor (\$RBSW)

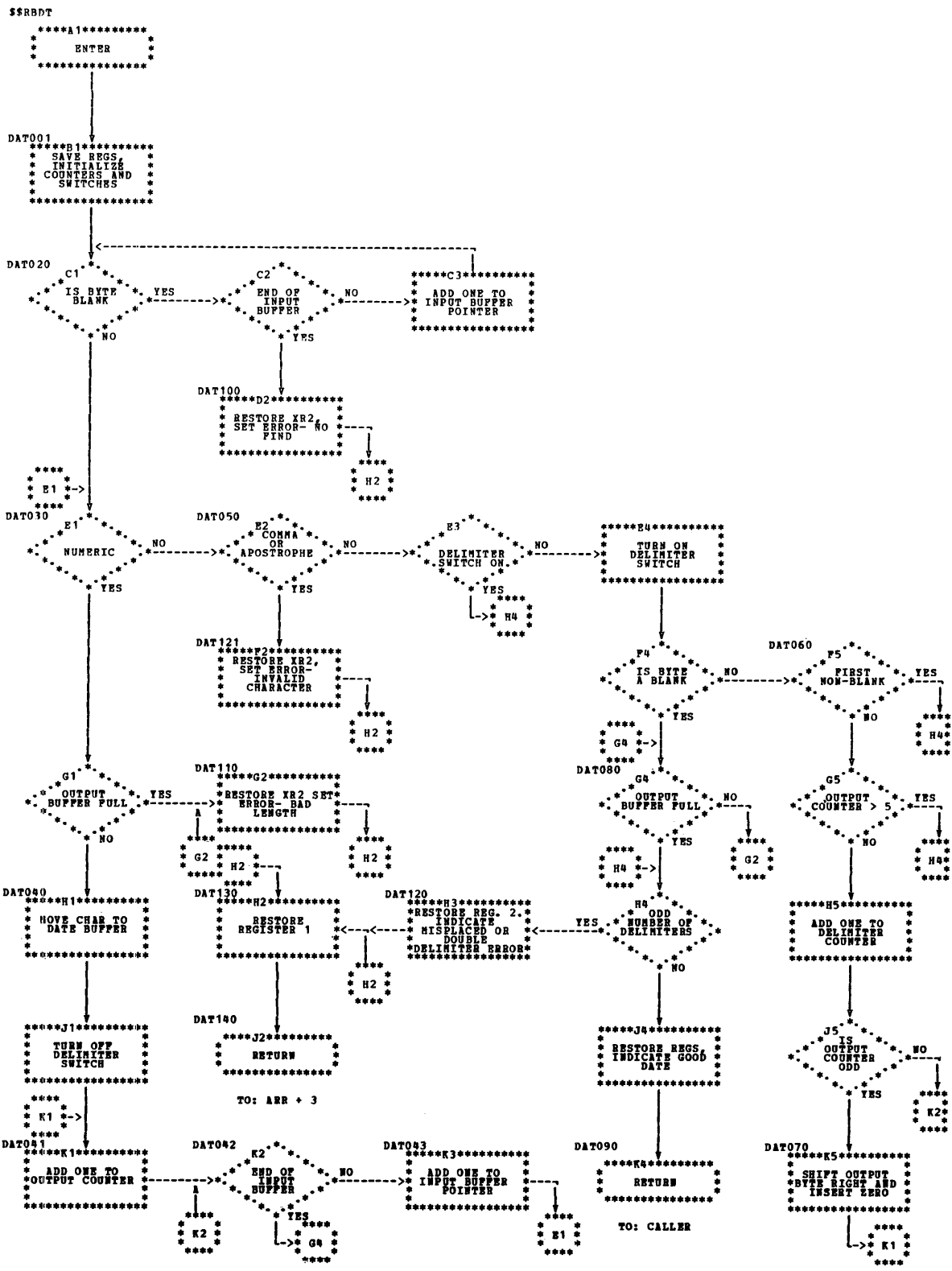


Chart FP. Date Scan Routine (\$\$RBDT)

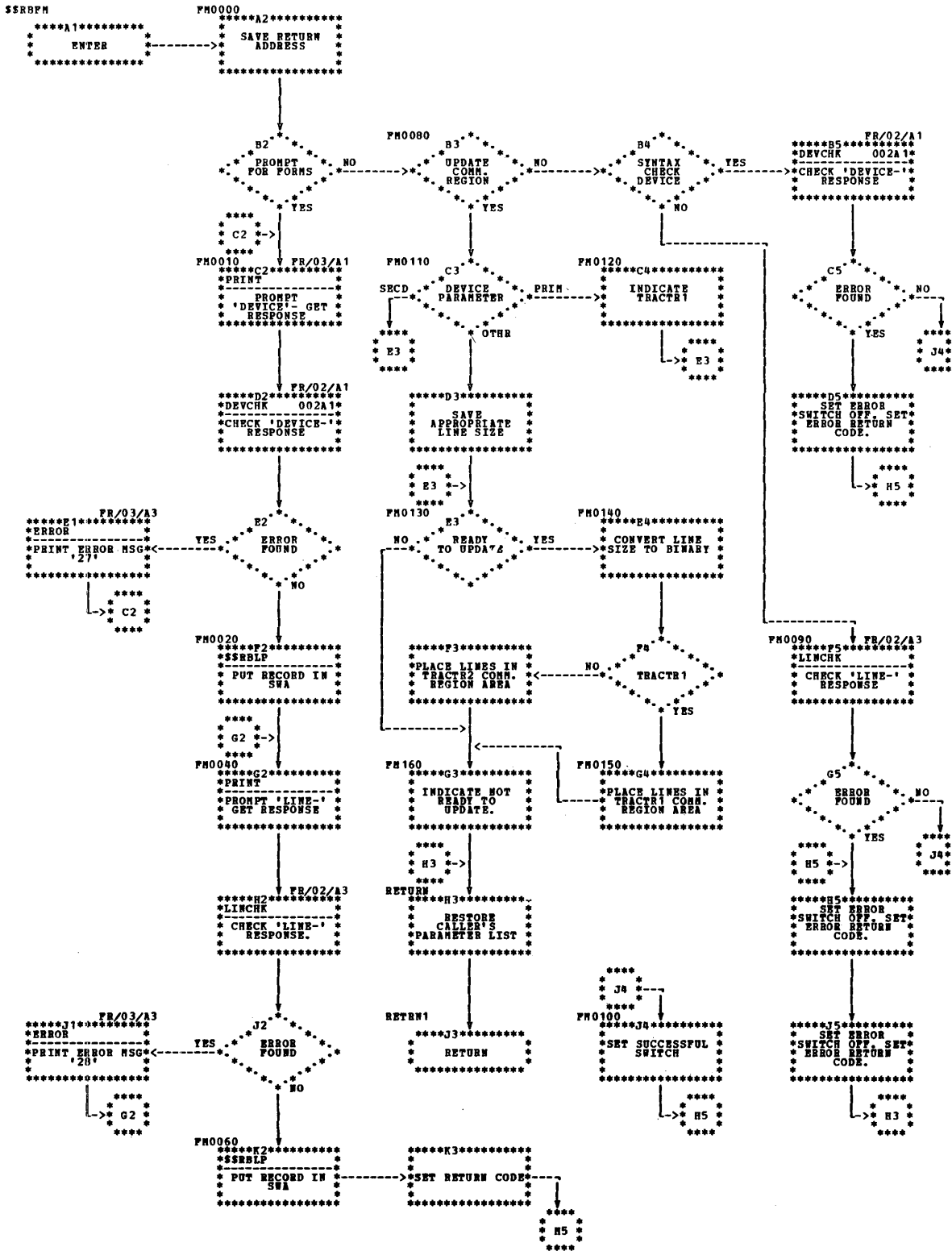


Chart FR (Part 1 of 3). Forms Statement Processor (\$\$RBFM)

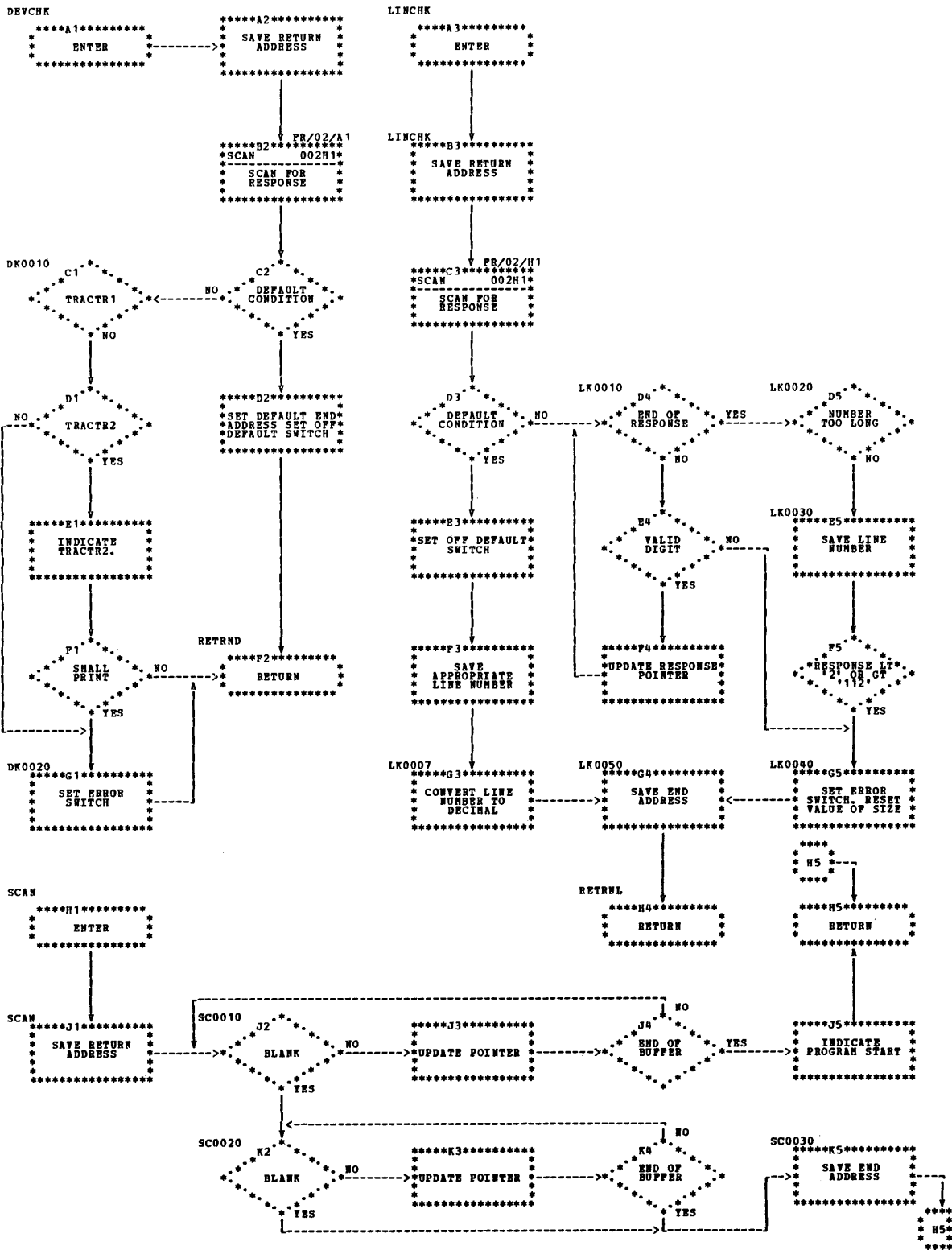


Chart FR (Part 2 of 3). Forms Statement Processor (\$\$RBFM)

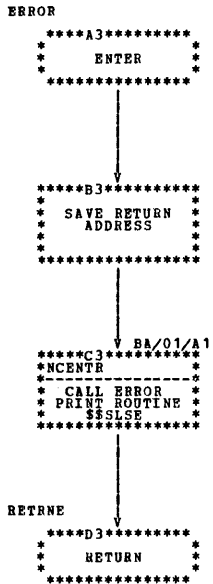
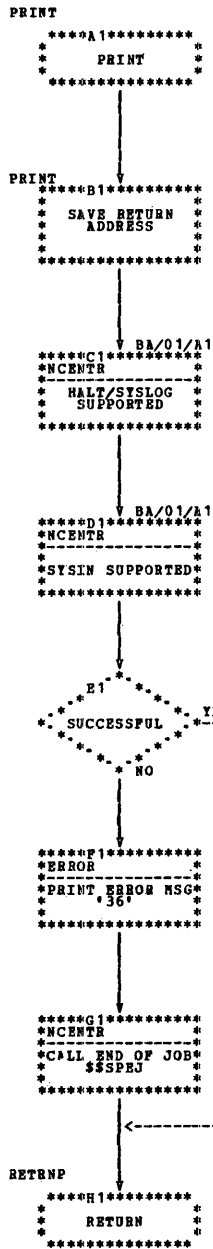


Chart FR (Part 3 of 3). Forms Statement Processor (\$\$RBFM)

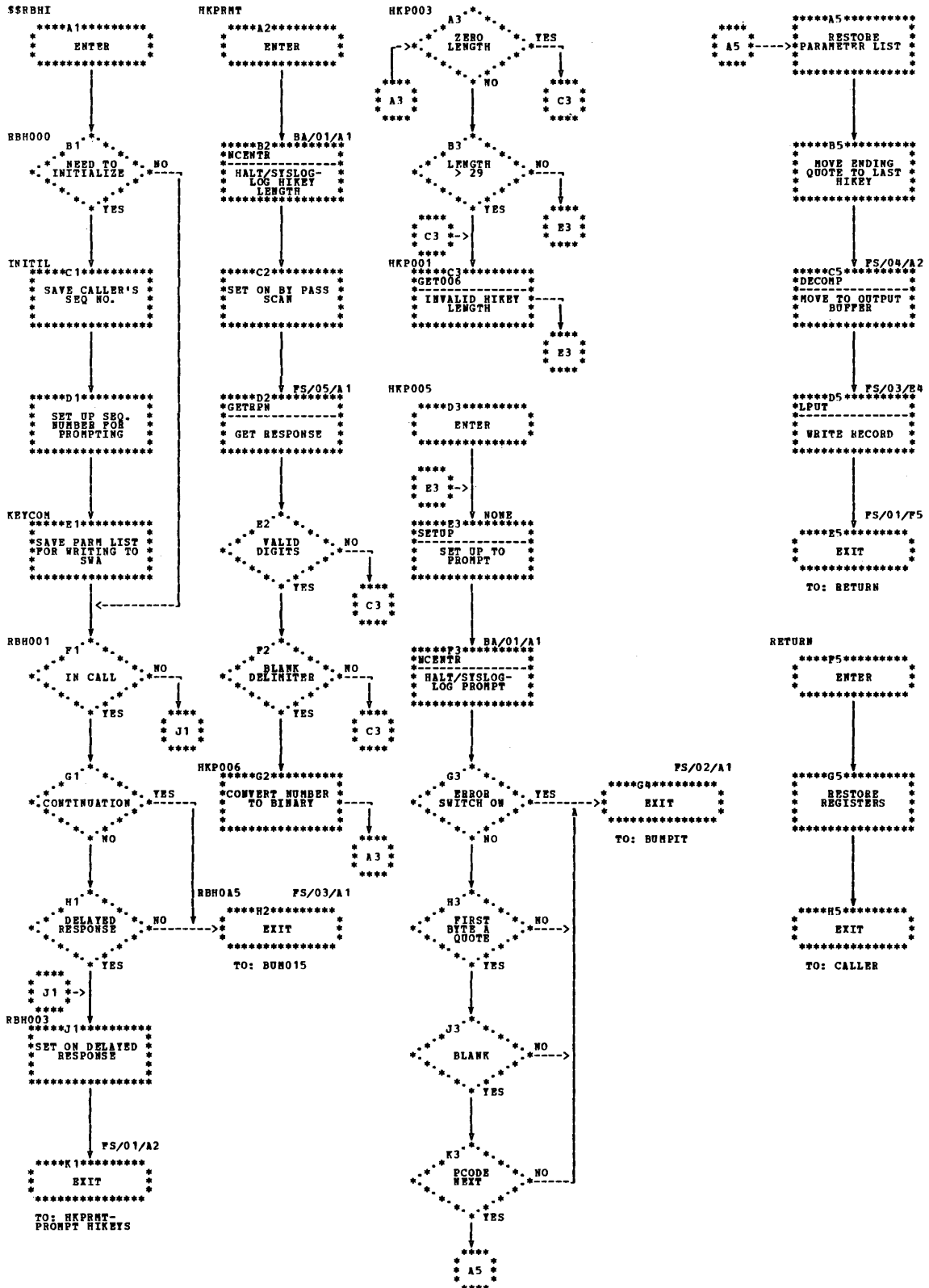


Chart FS (Part 1 of 5). HIKEY Processor (\$\$RBHI)

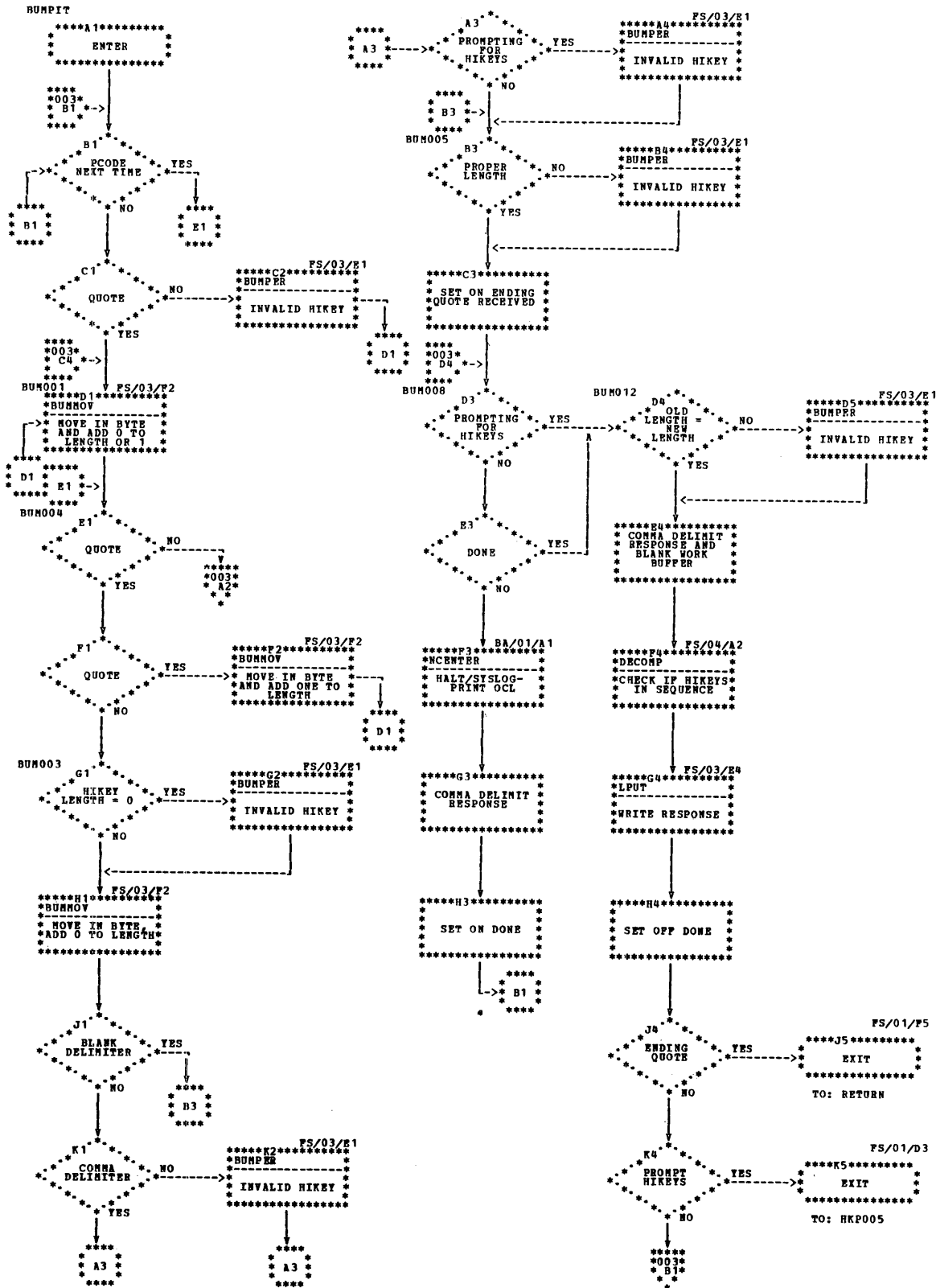


Chart FS (Part 2 of 5). HIKEY Processor (\$RBHI)

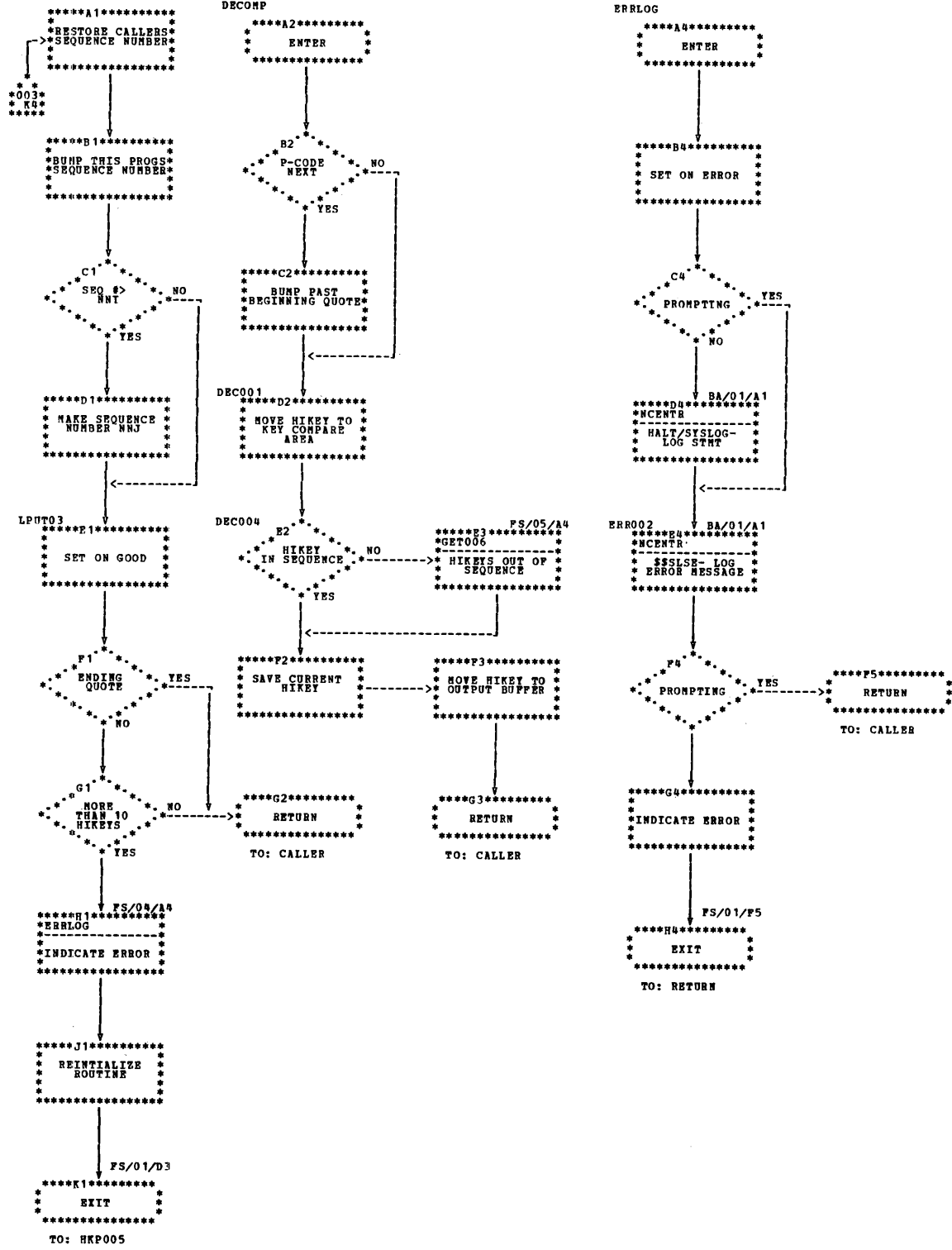


Chart FS (Part 4 of 5). HIKEY Processor (\$\$RBHI)

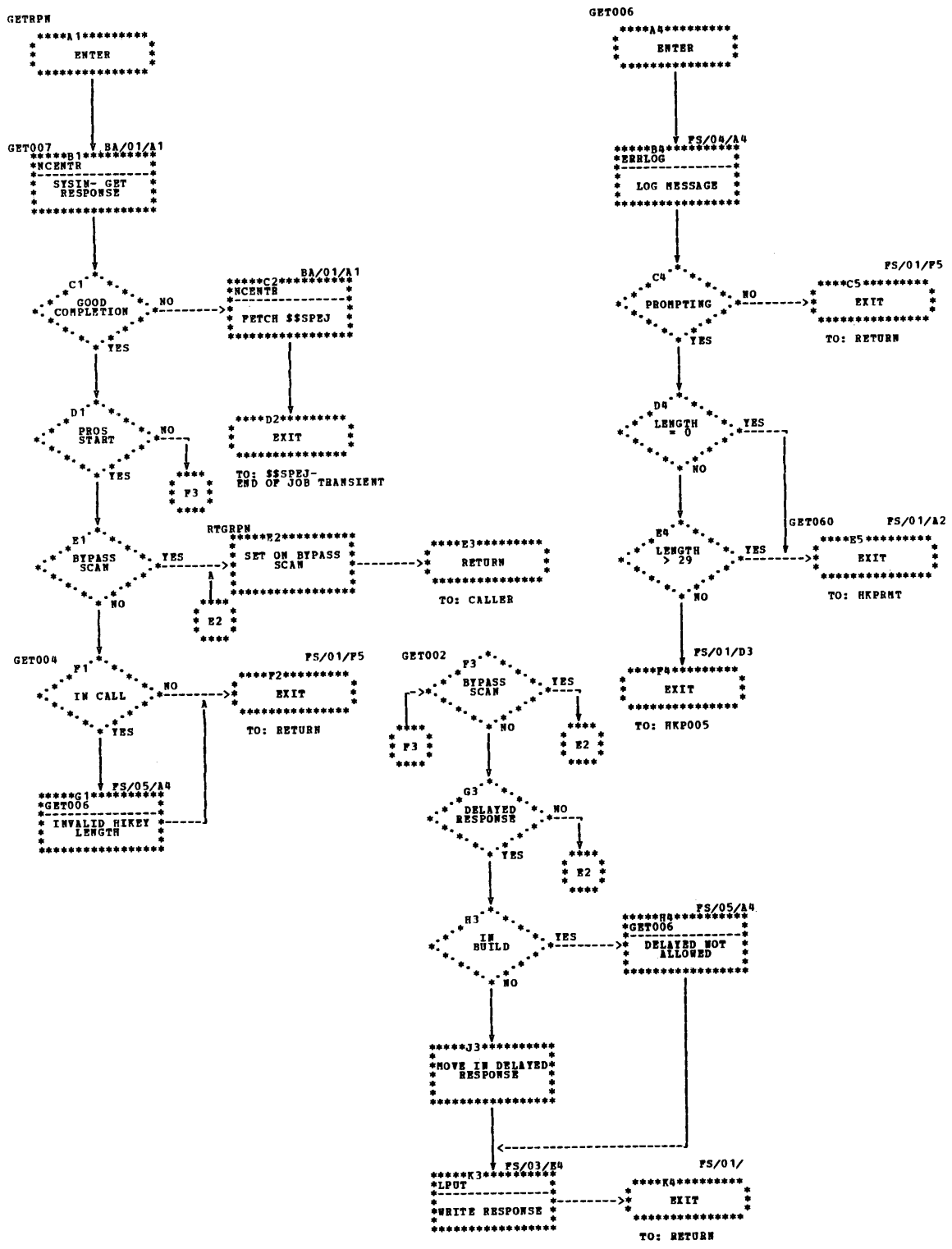


Chart FS (Part 5 of 5). HIKEY Processor (\$\$RBHI)

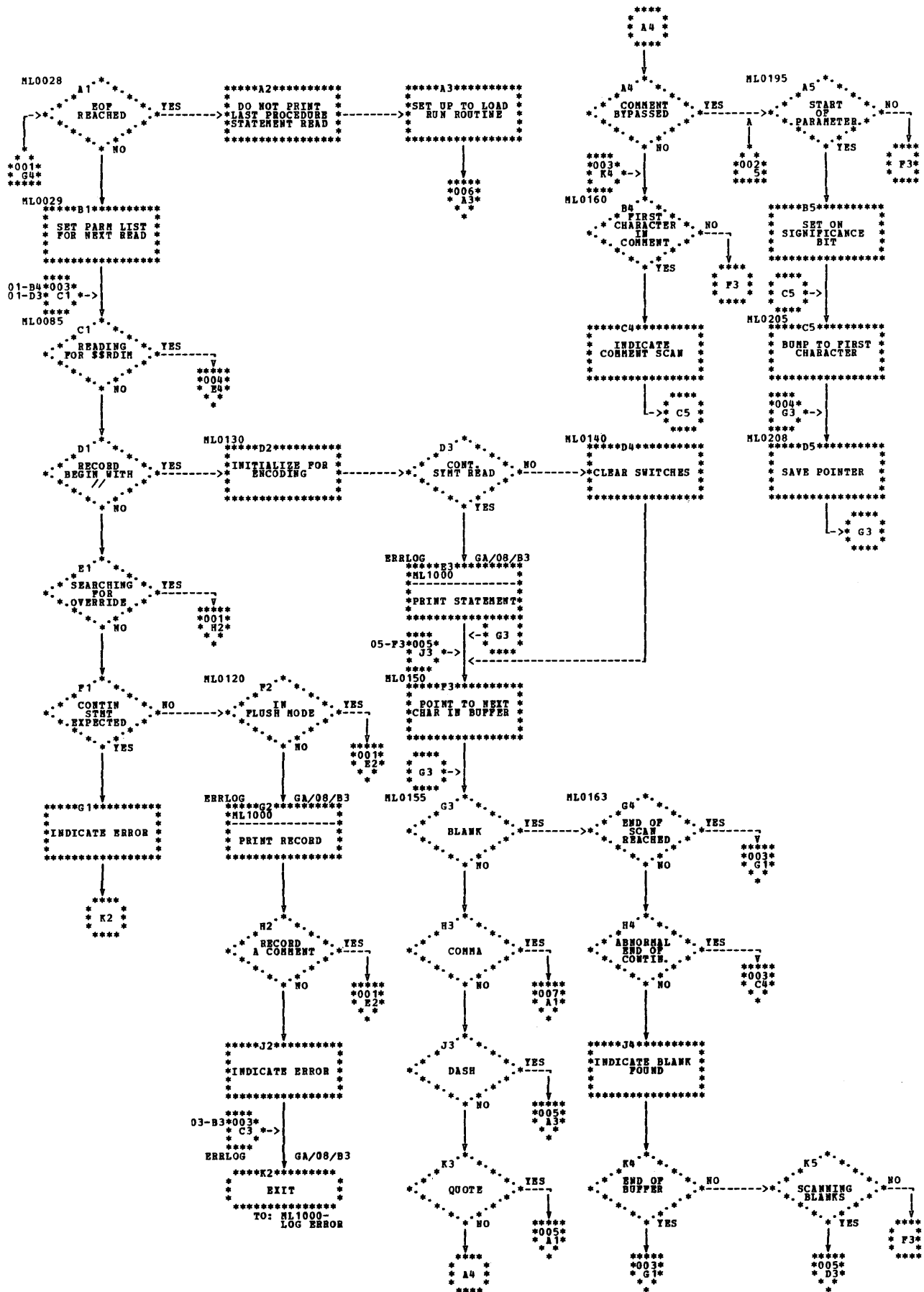


Chart GA (Part 2 of 8). Reader/Interpreter Mainline (\$\$RDML)

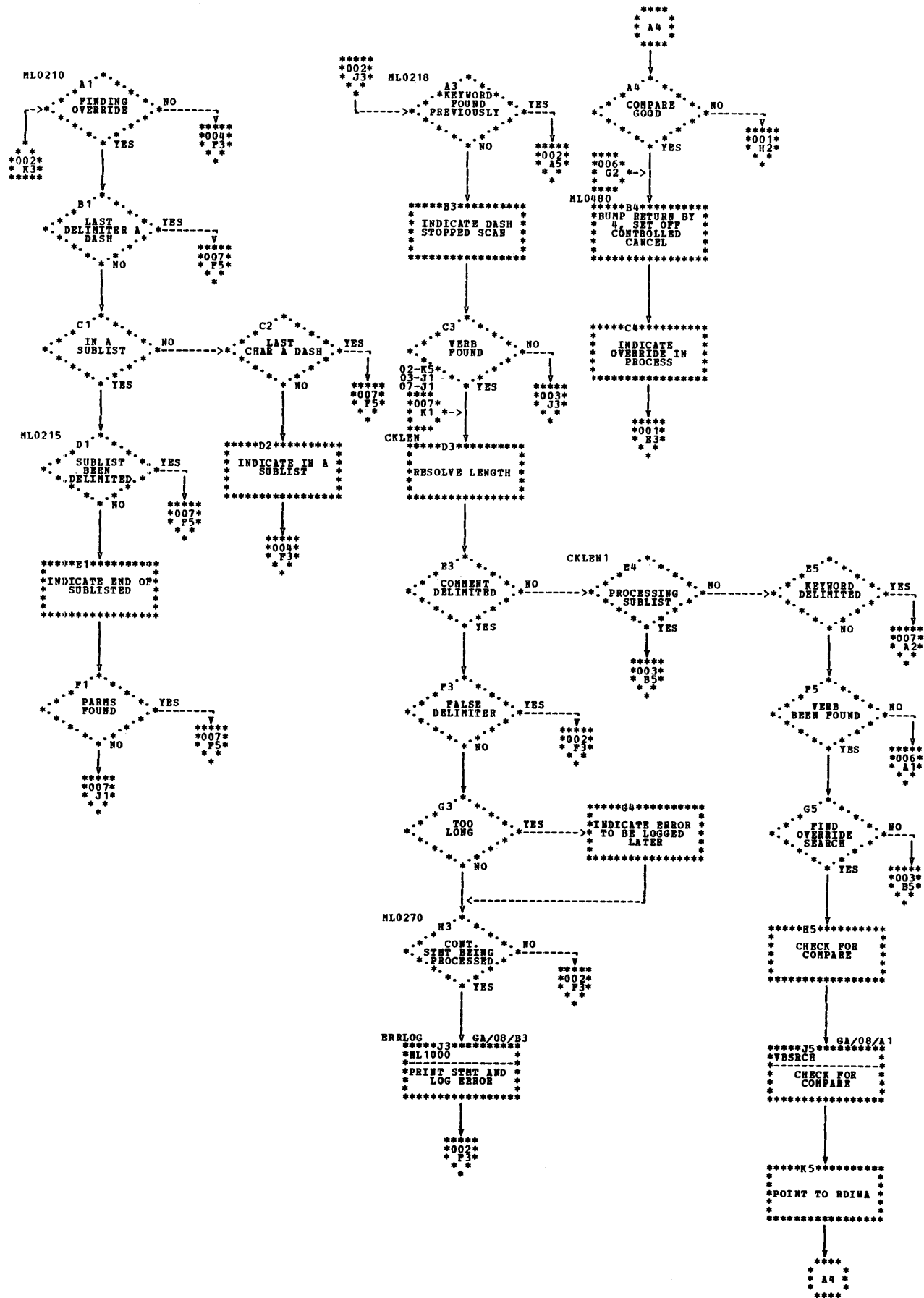


Chart GA (Part 5 of 8). Reader/Interpreter Mainline (\$\$RDML)

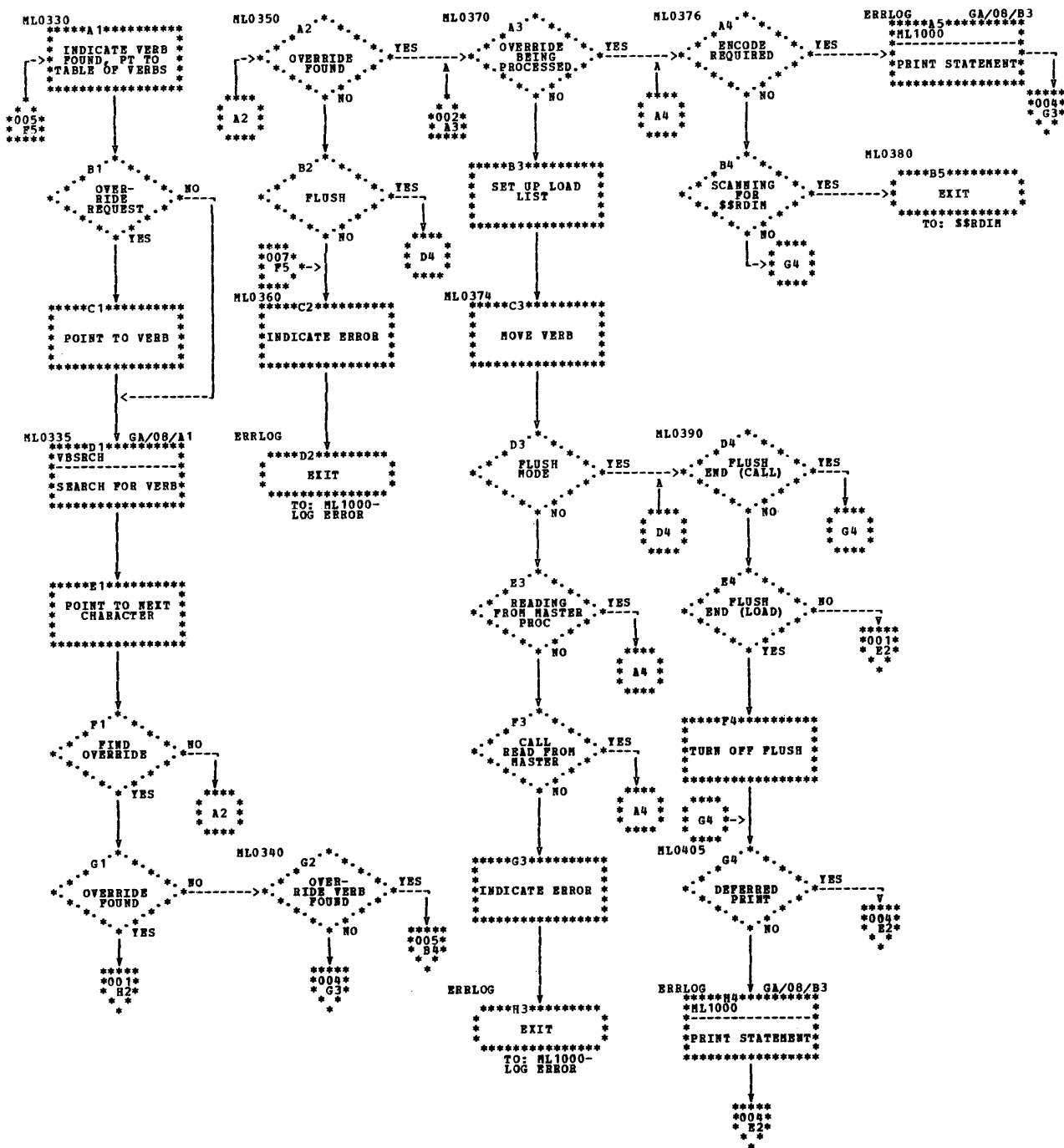


Chart GA (Part 6 of 8). Reader/Interpreter Mainline (\$\$RDML)

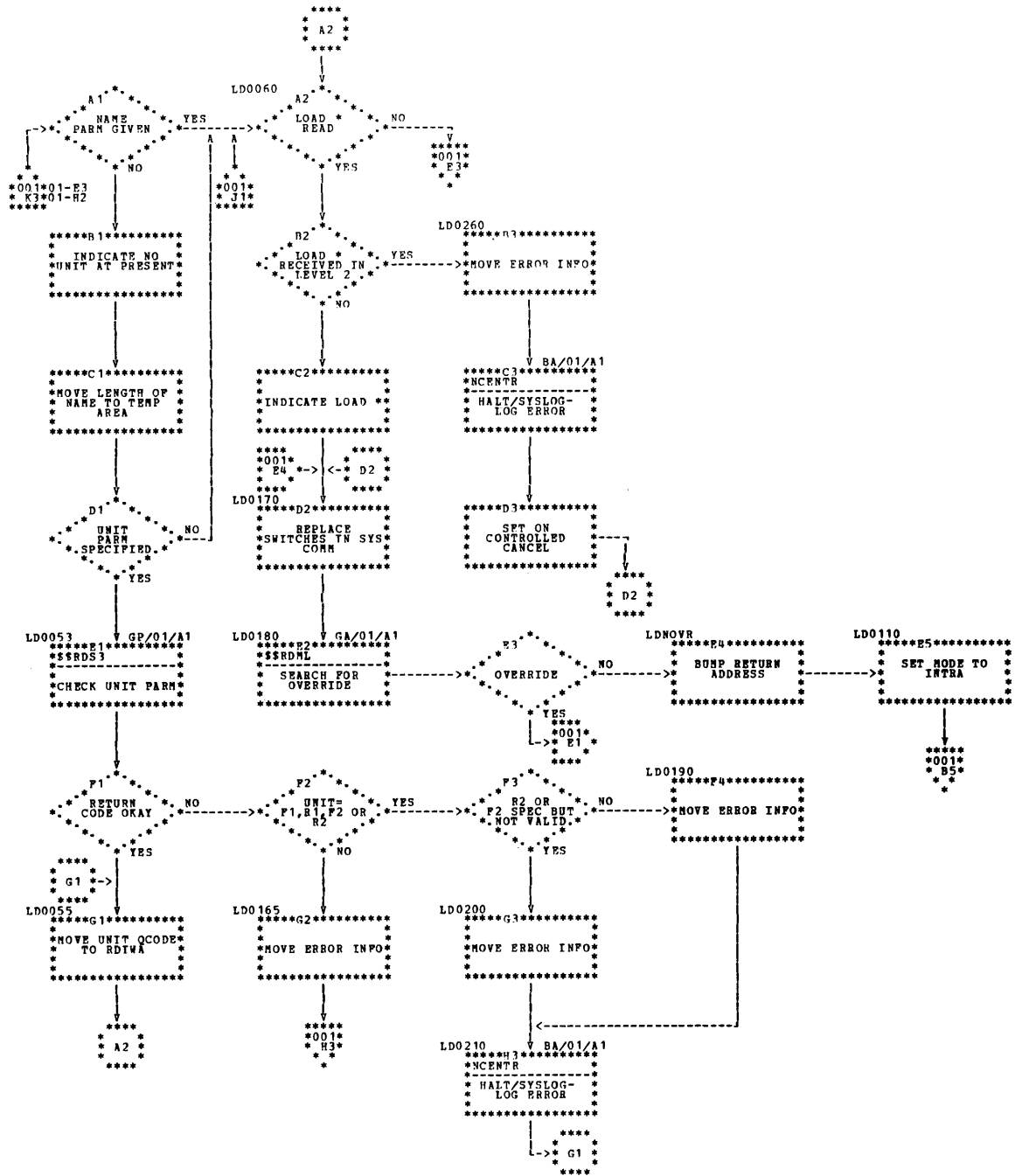


Chart GB (Part 2 of 2). // LOAD Control Card Processor (\$\$RDLD)

RN0000

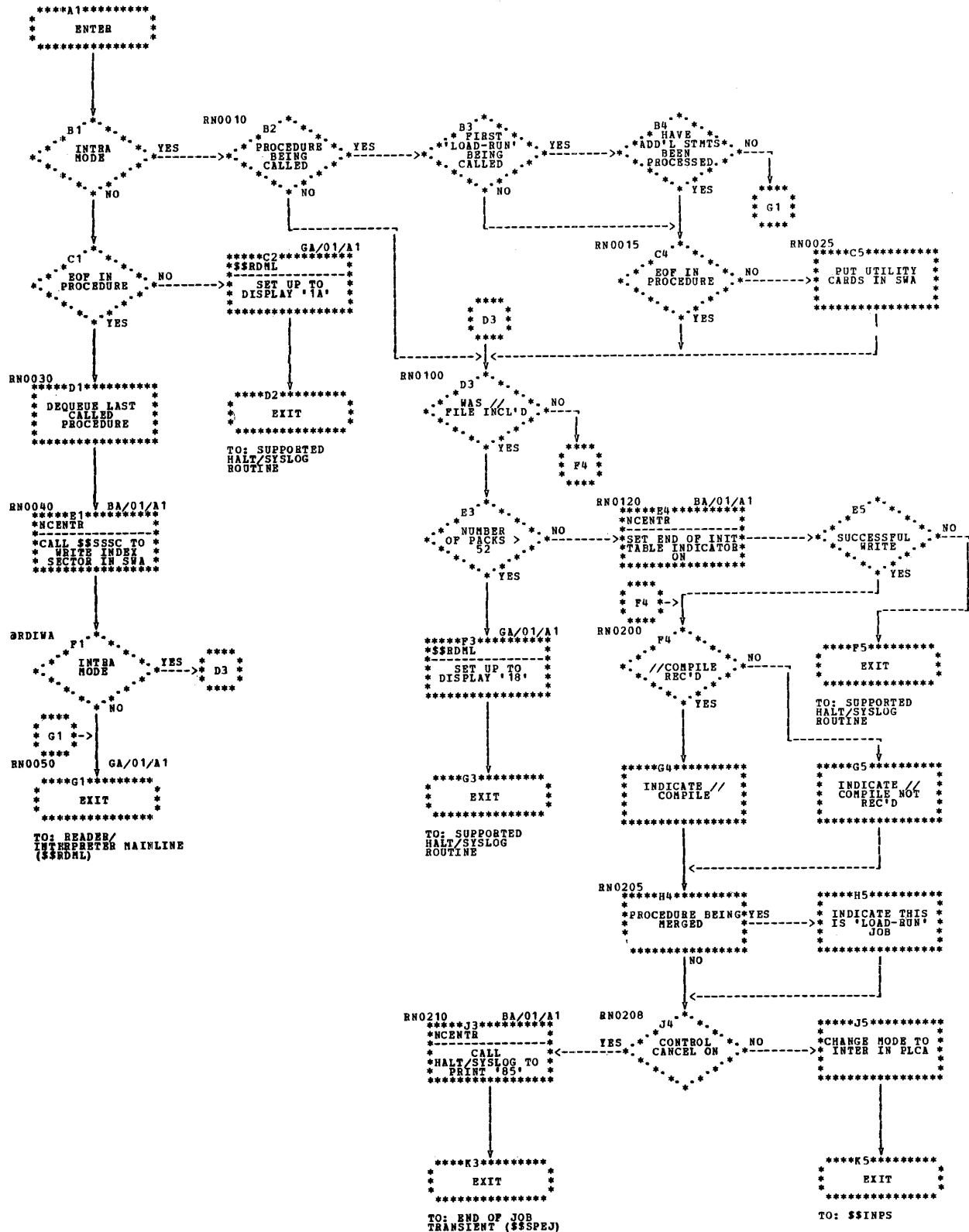


Chart GC. // RUN Control Card Processor (\$\$RDRN)

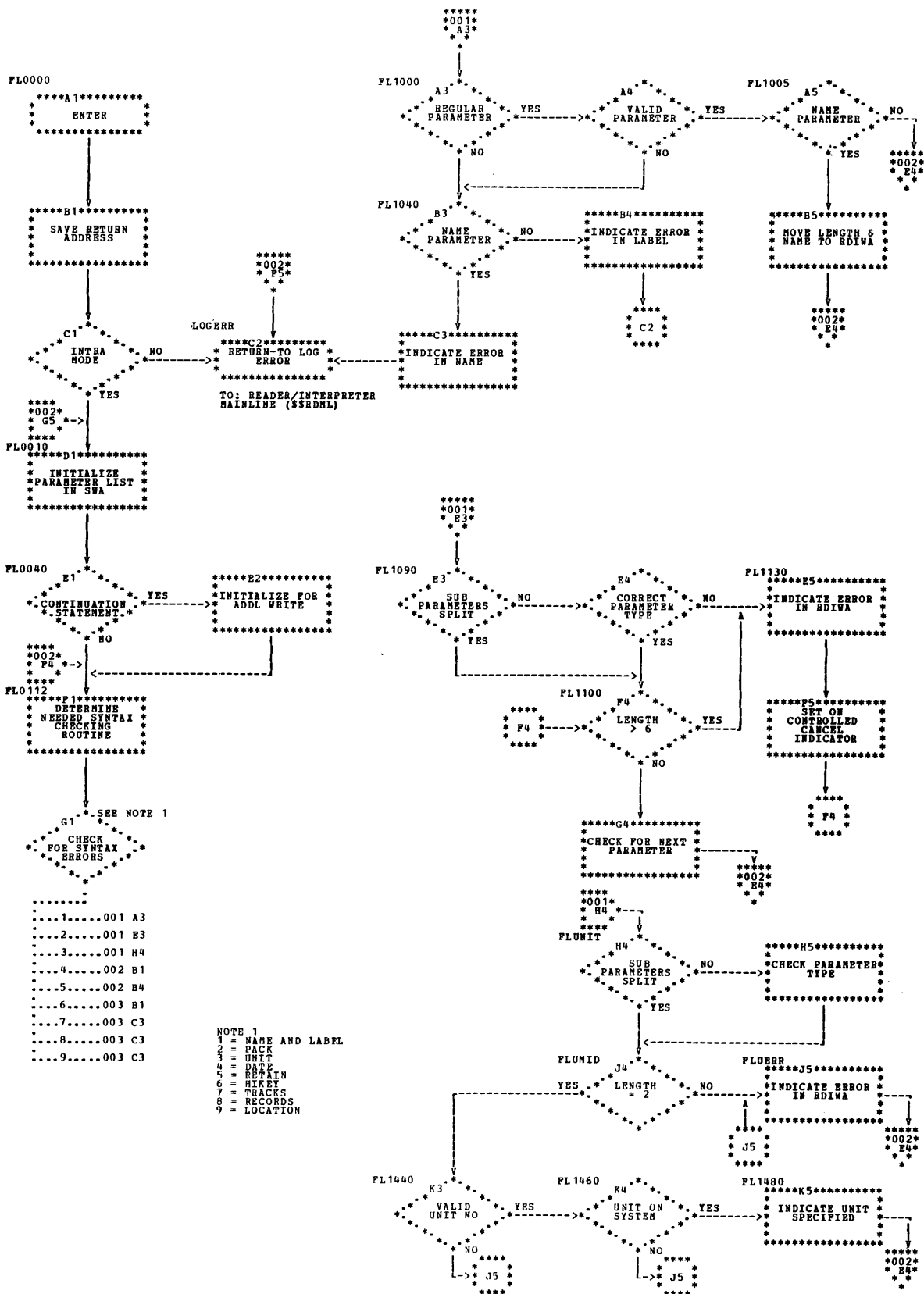


Chart GD (Part 1 of 3). // FILE Control Card Processor (\$\$RDFL)

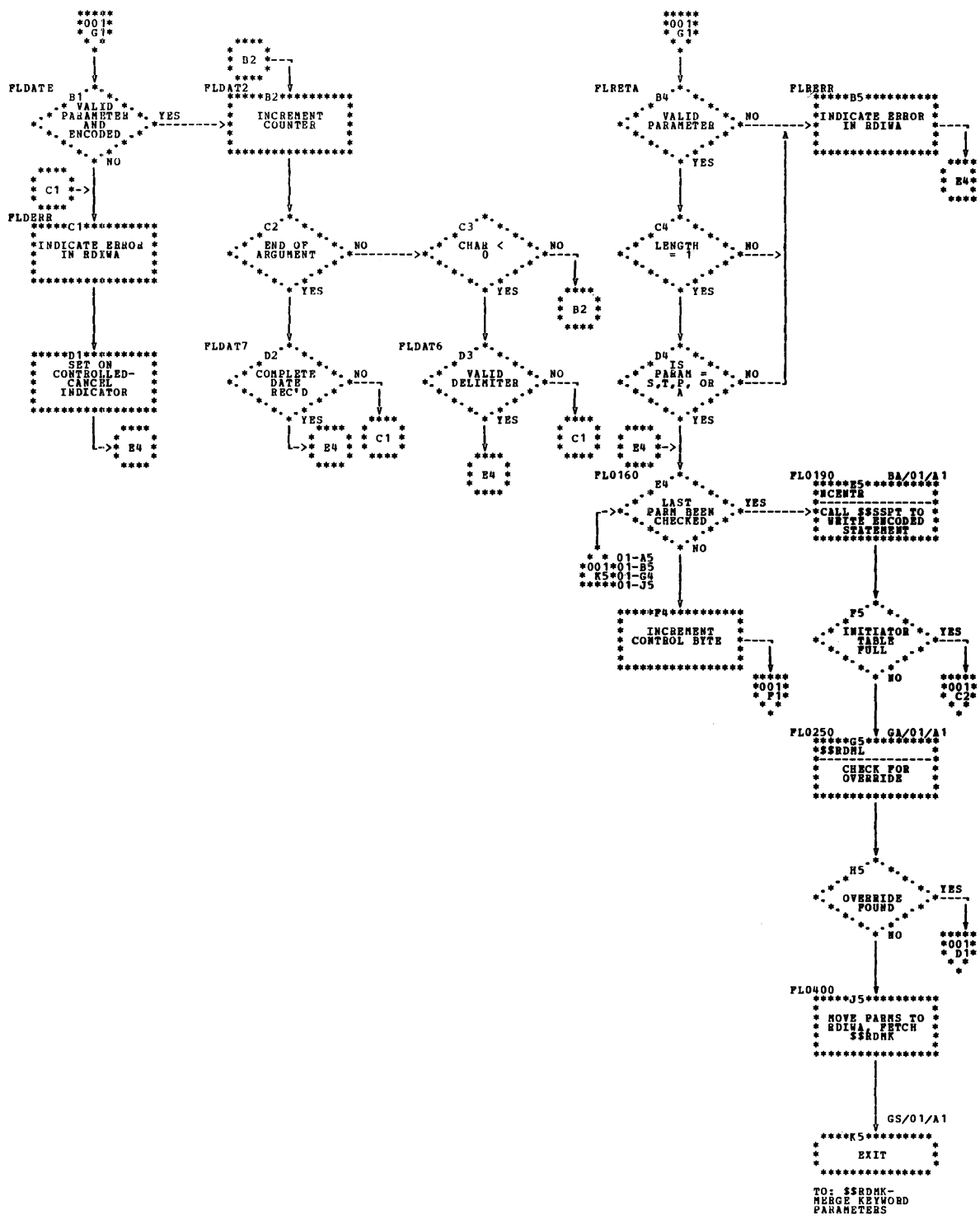


Chart GD (Part 2 of 3). // FILE Control Card Processor (\$\$RDFL)

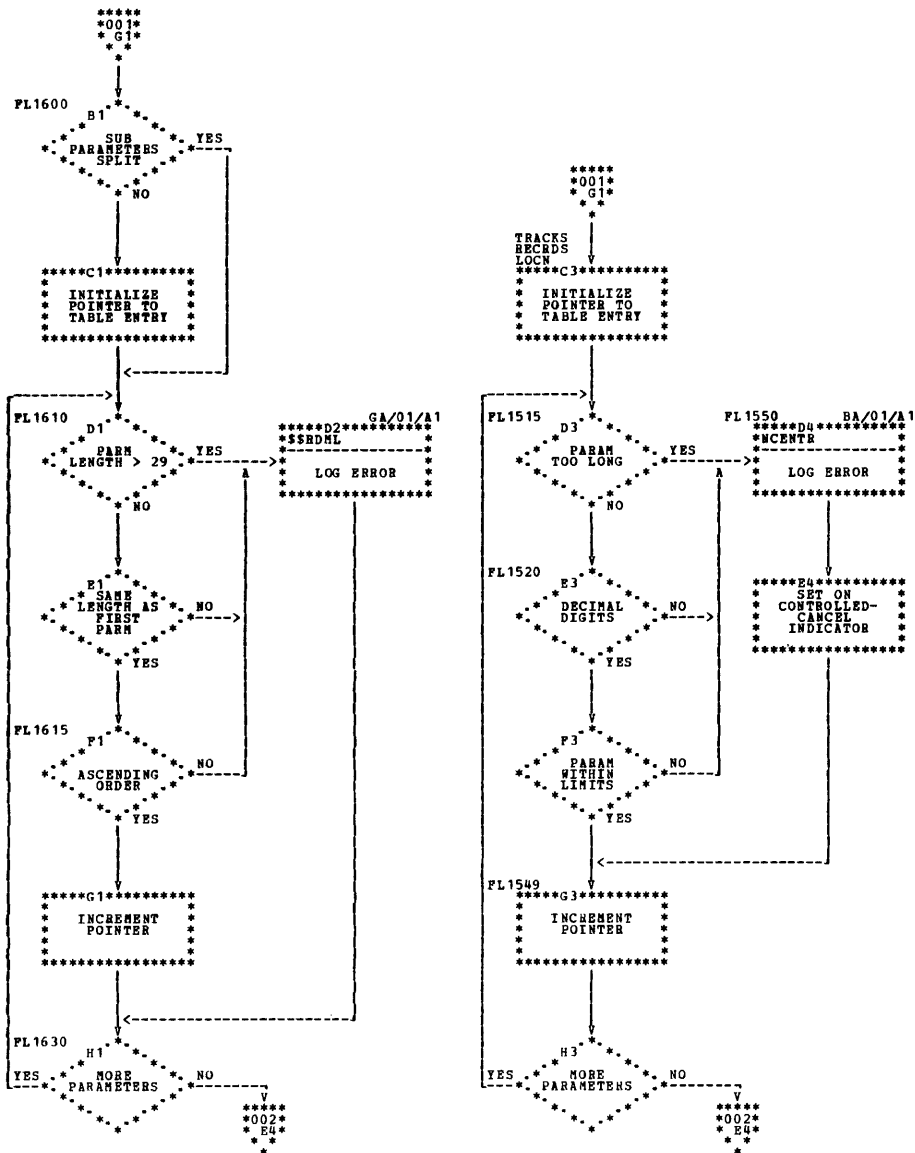
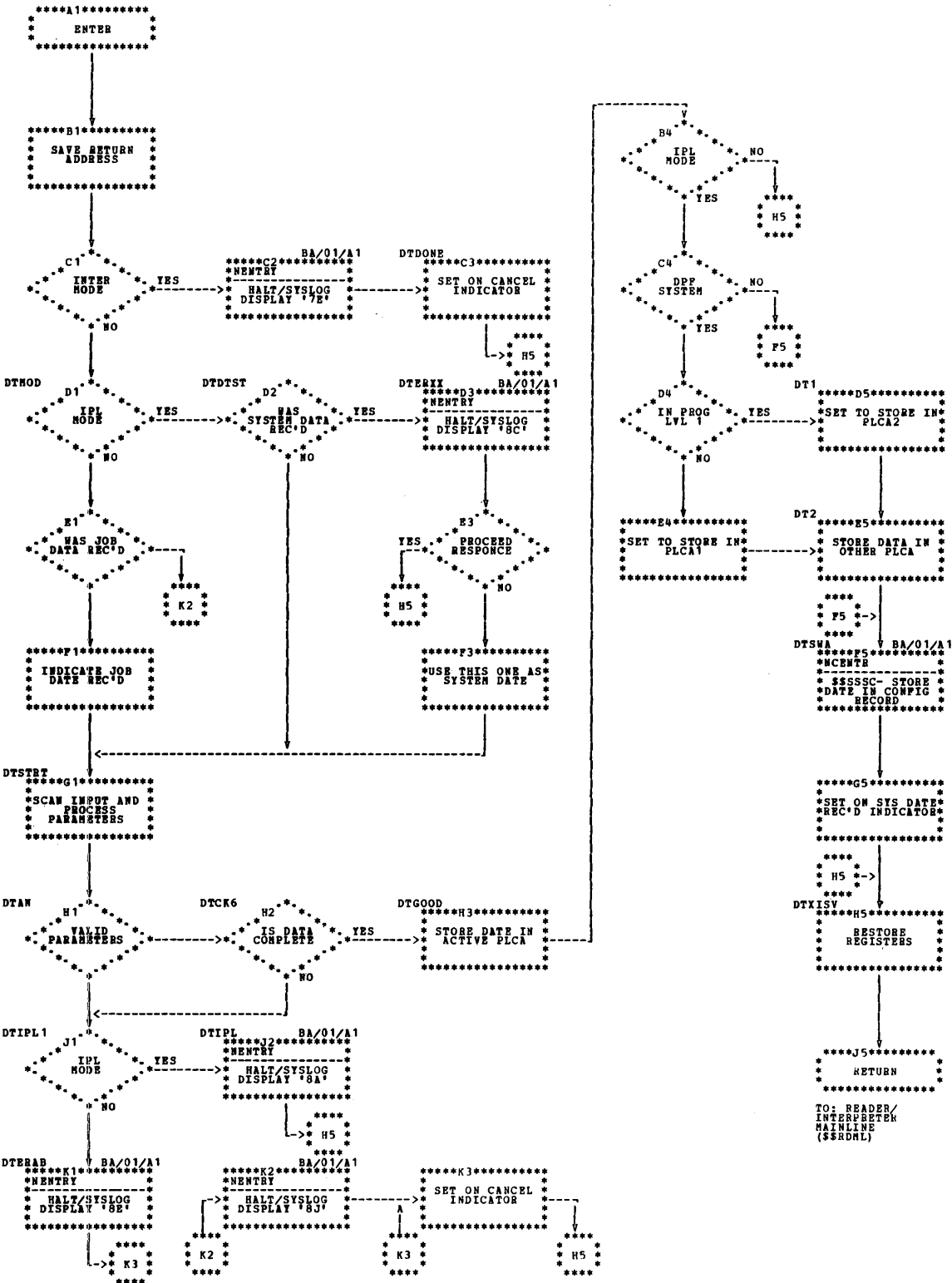


Chart GD (Part 3 of 3). // FILE Control Card Processor (\$\$RDFL)

DT0000



5

Chart GE. // DATE Control Card Processor (\$\$RDDT)

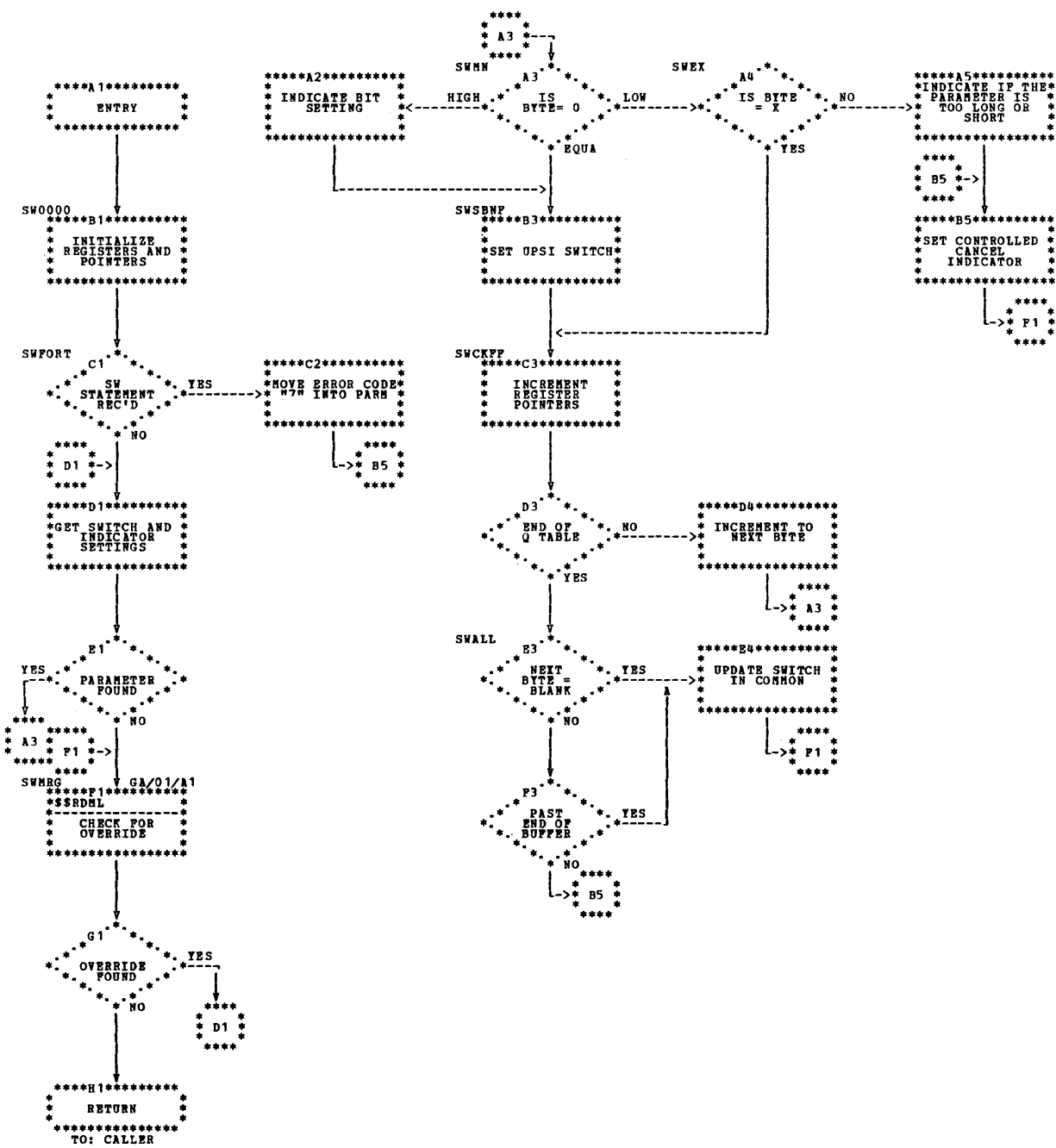


Chart GF. // SWITCH Control Card Processor (\$\$RDSW)

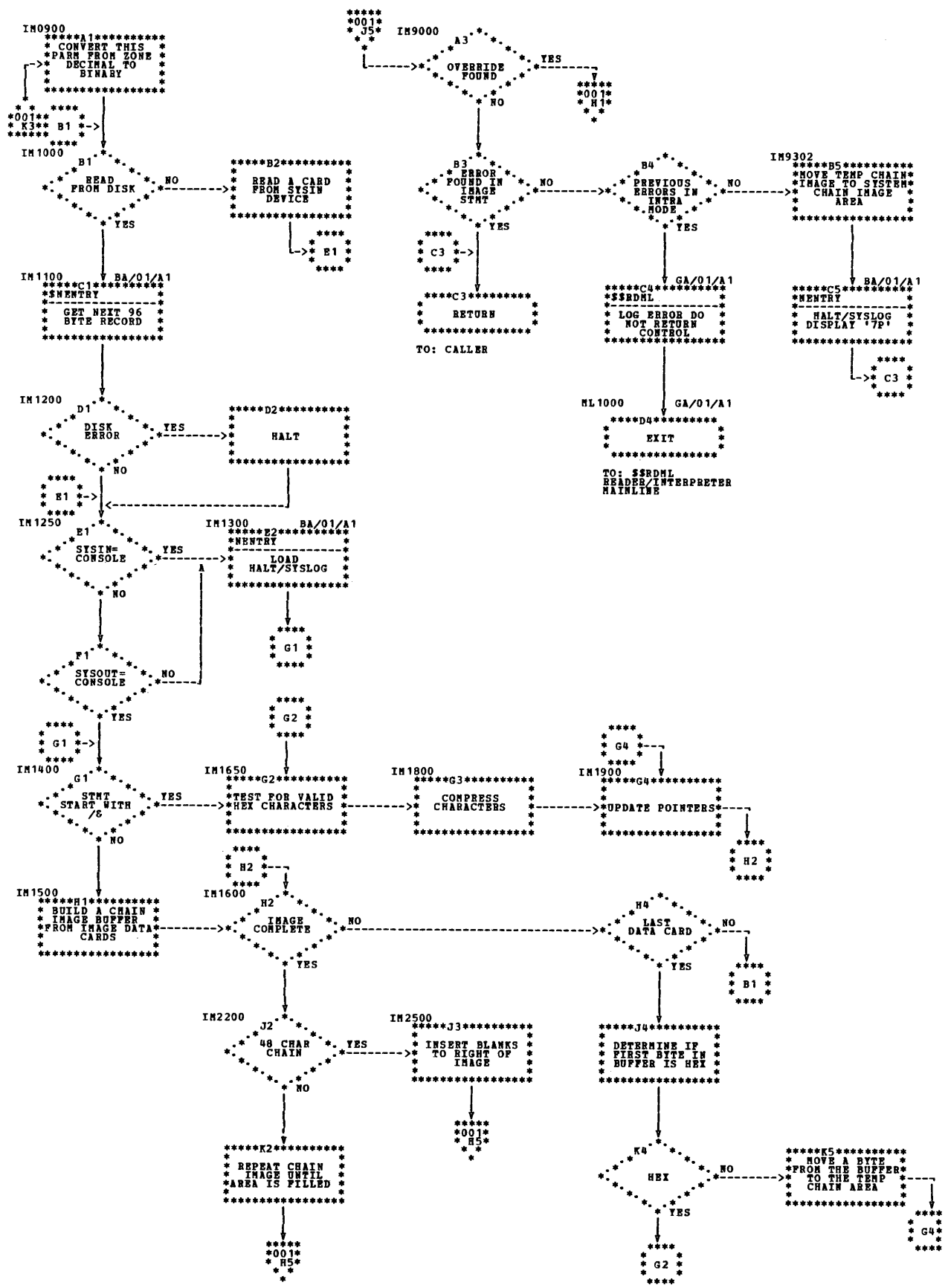


Chart GG (Part 2 of 2). // IMAGE Control Card Processor (\$\$RDIM)

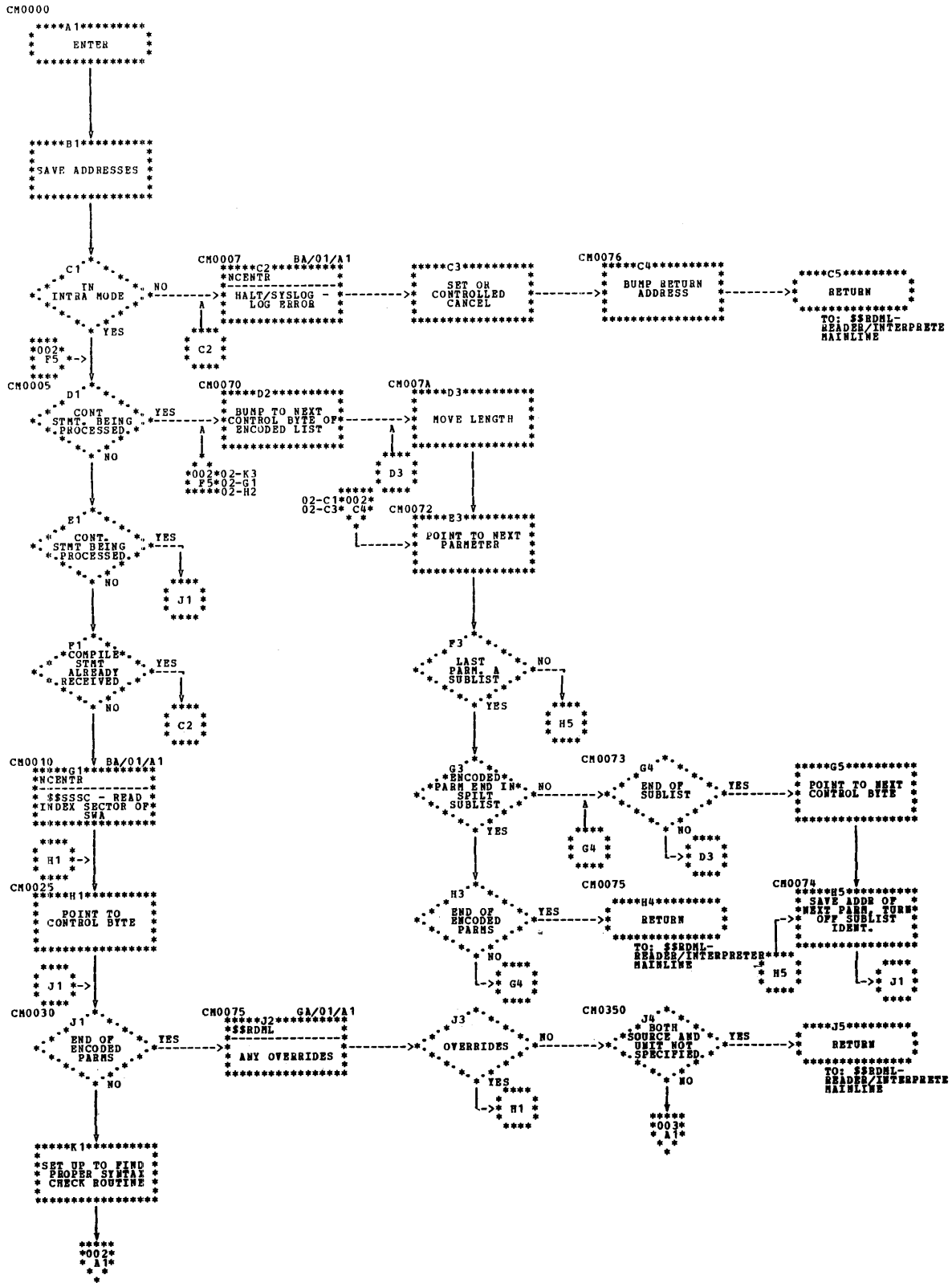


Chart GH (Part 1 of 3). // COMPILER Control Card Processor (\$\$RDCM)

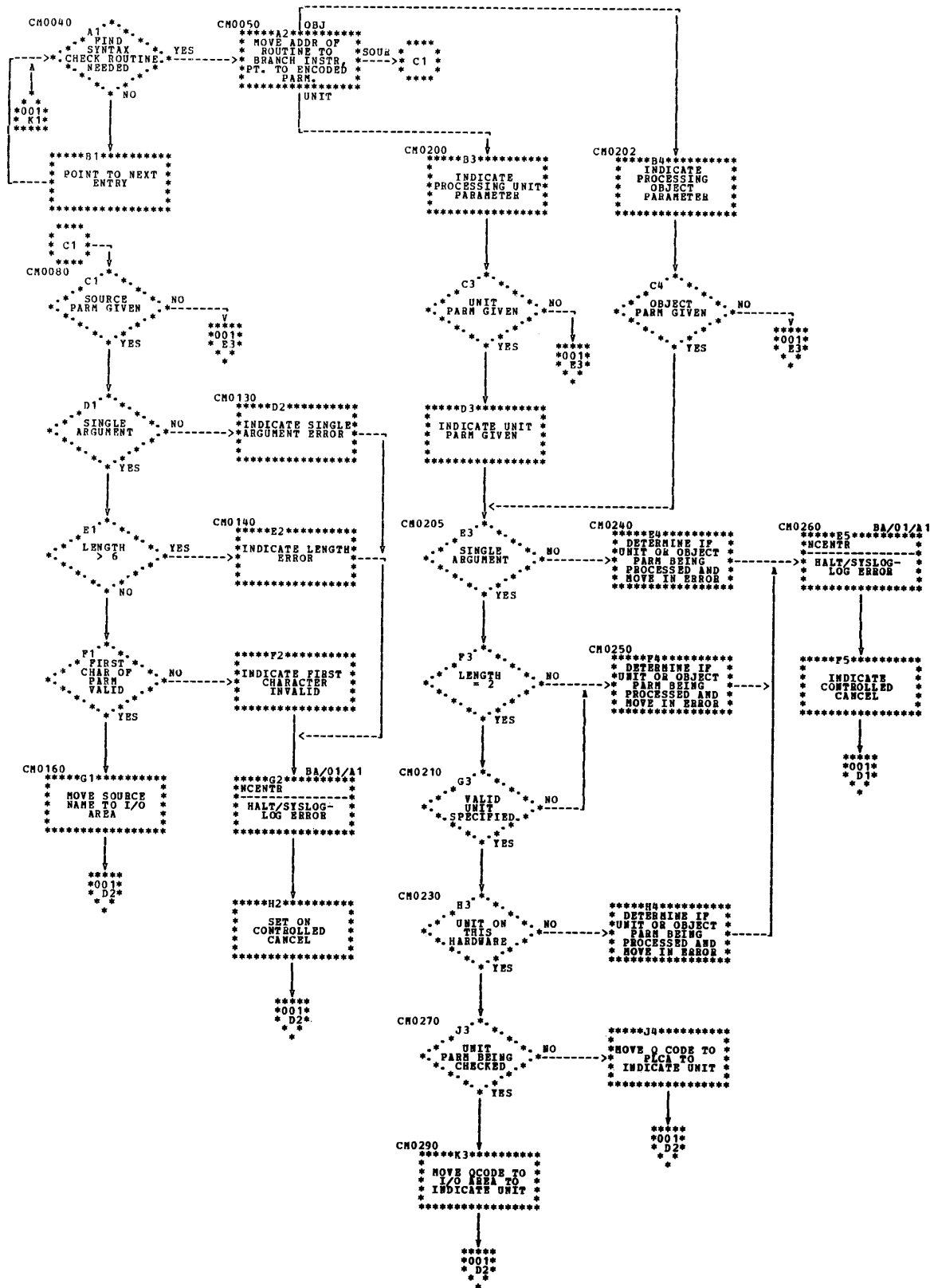


Chart GH (Part 2 of 3). // COMPILER Control Card Processor (RDCM)

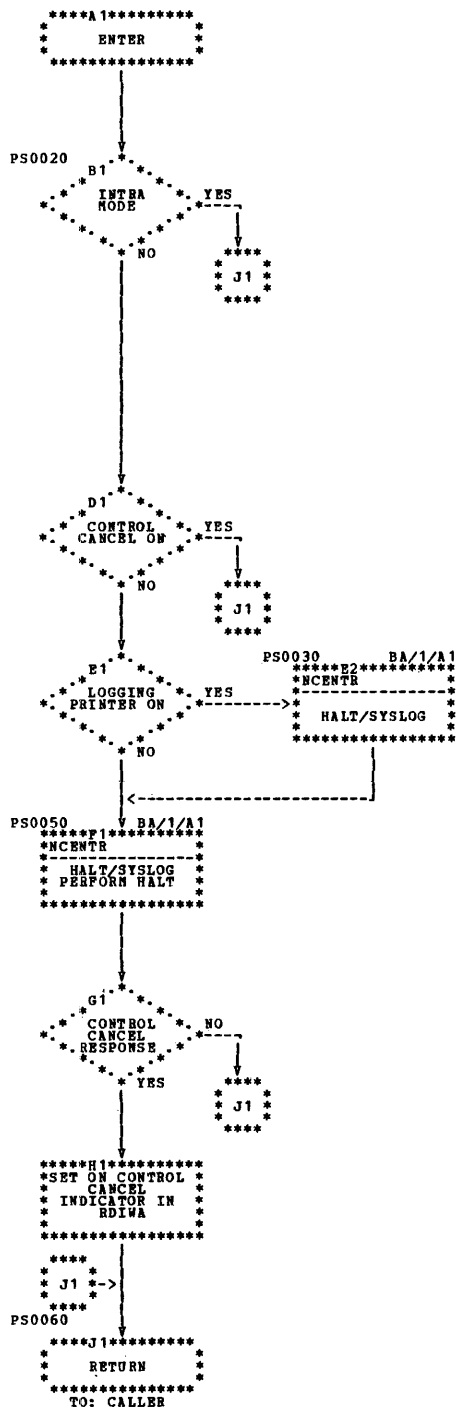


Chart GK. // PAUSE Control Card Processor (\$RDPS)

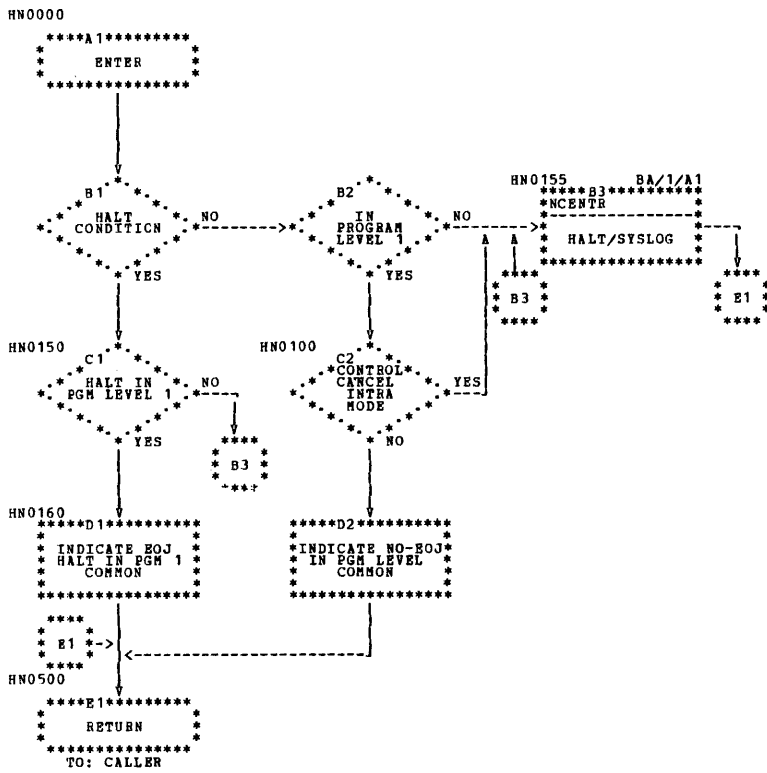


Chart GL. // HALT, // NOHALT Control Card Processor (\$\$RDHN)

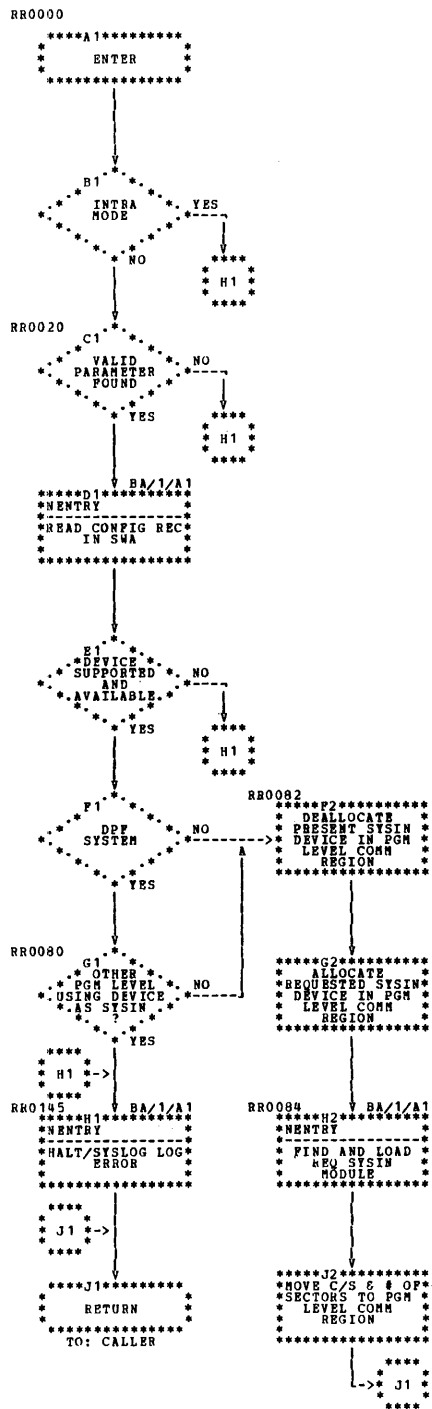


Chart GM. // READER Control Card Processor (\$\$RDRR)

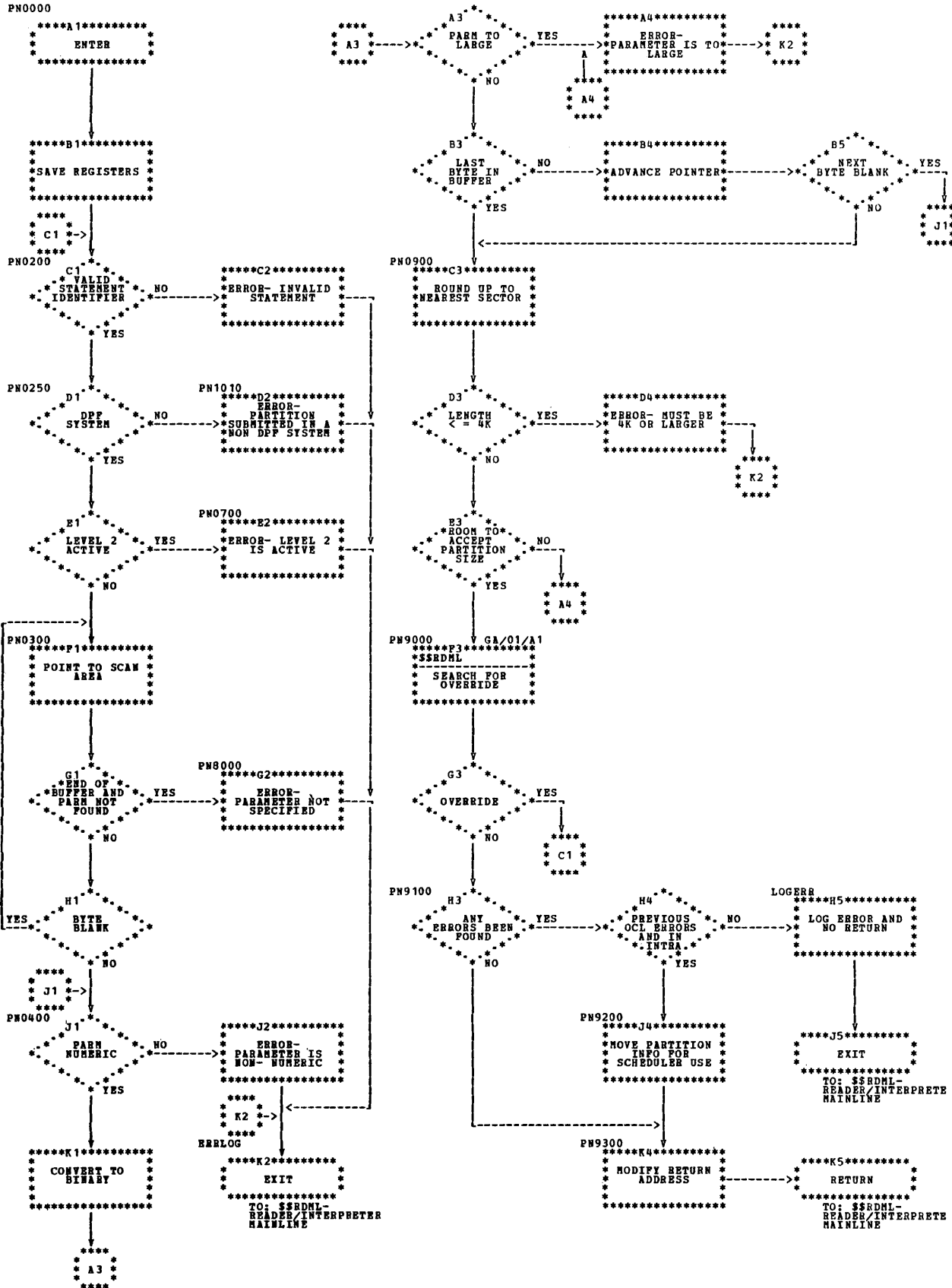


Chart GN. // PARTITION Control Card Processor (\$\$RDPN)

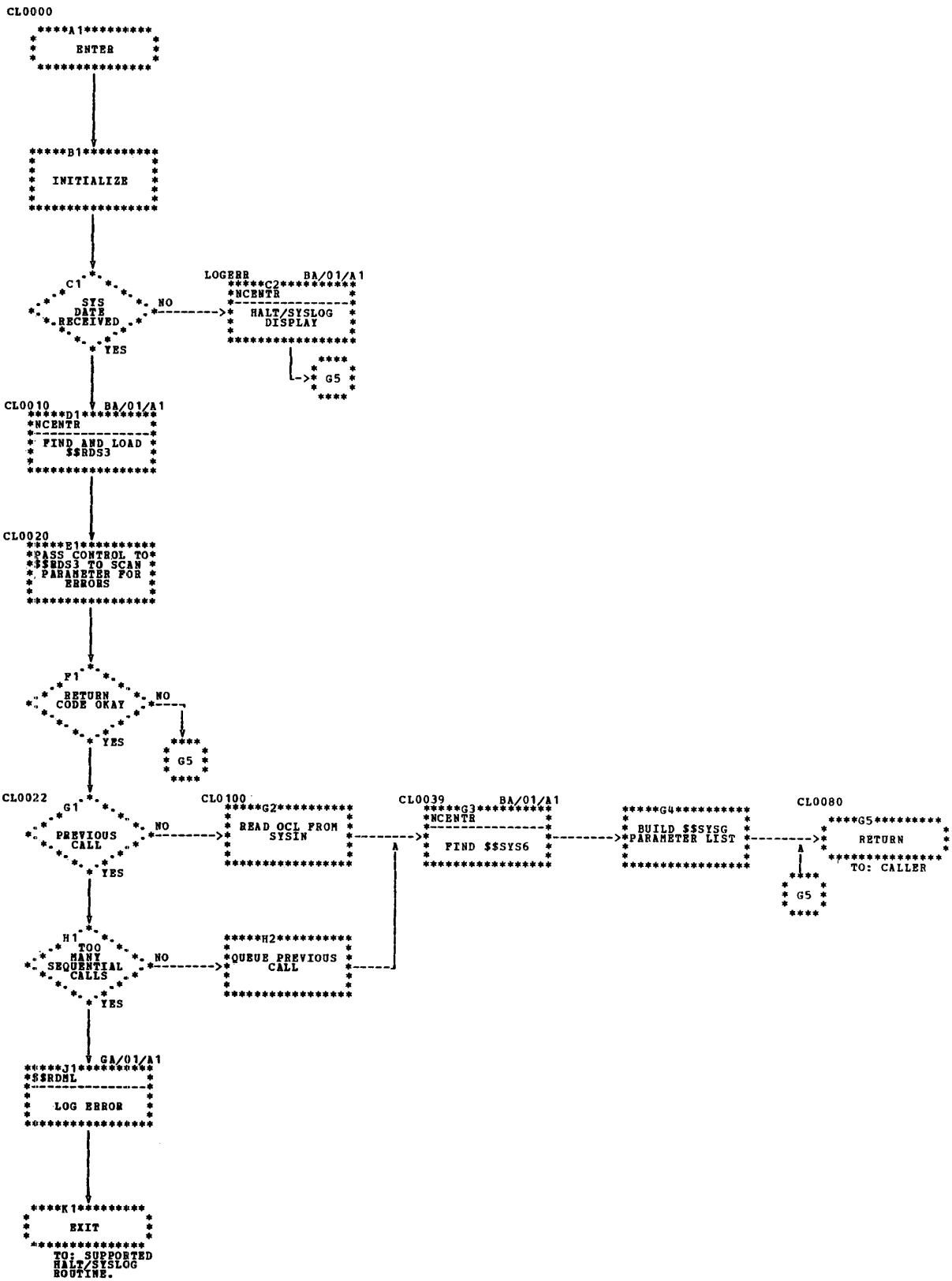


Chart GO. // CALL Control Card Processor (\$\$RDCL)

S30000

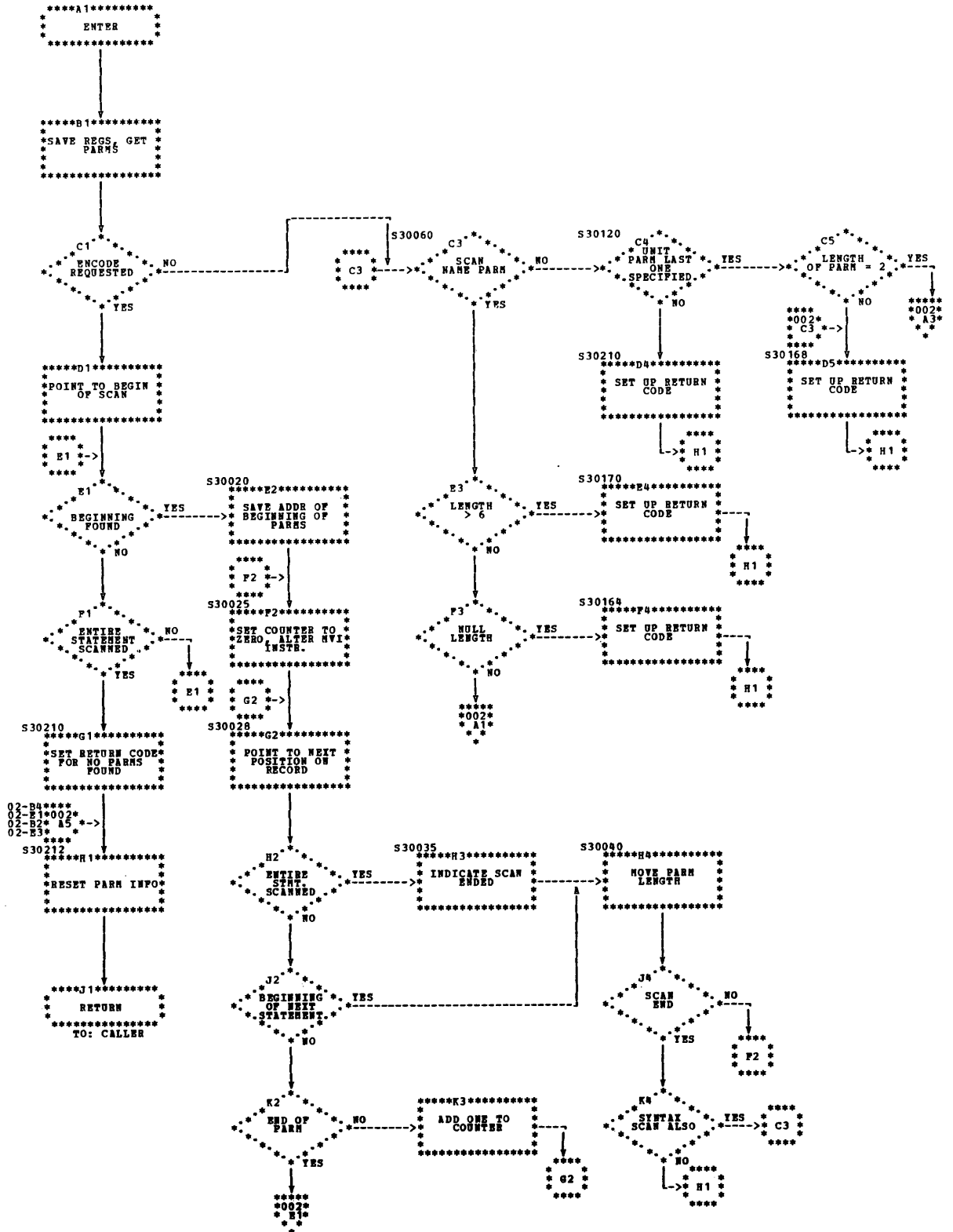


Chart GP (Part 1 of 2). Positional Parameter Syntax Scan Routine (\$\$RDS3)

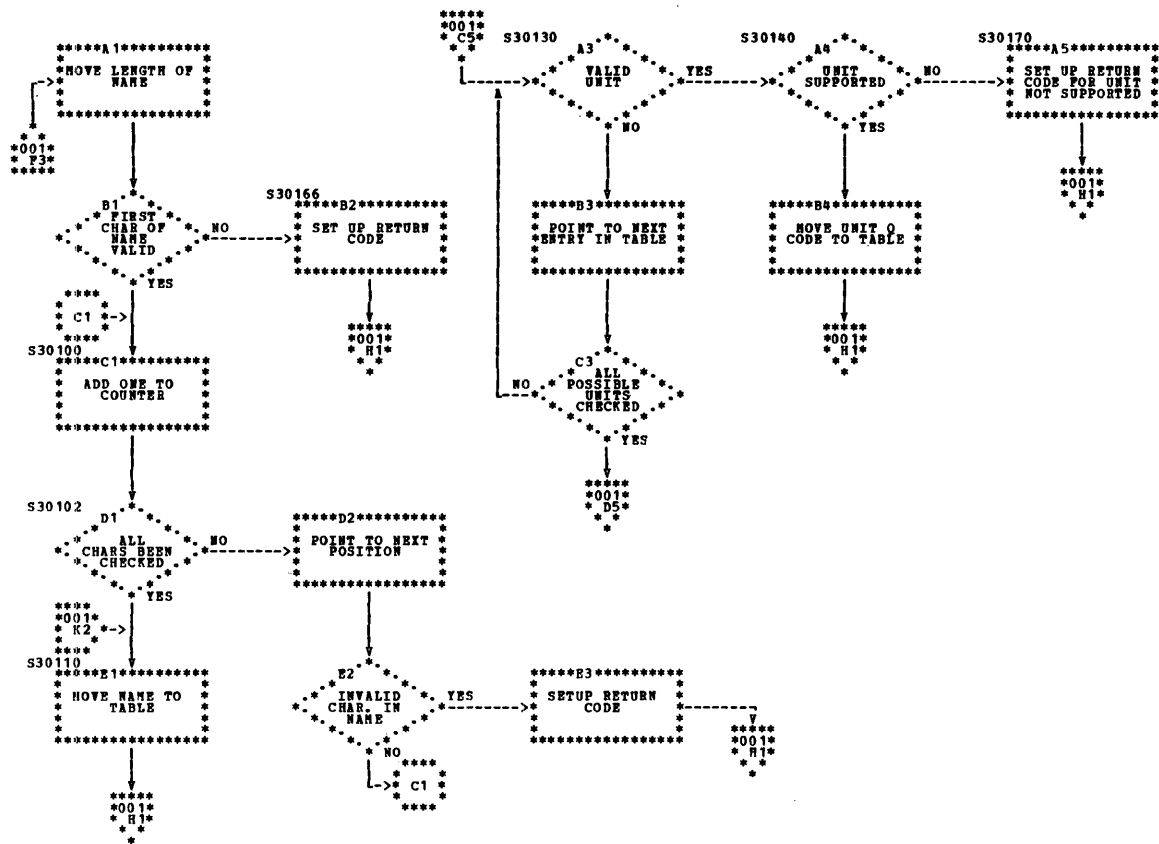


Chart GP (Part 2 of 2). Positional Parameter Syntax Scan Routine (\$\$RDS3)

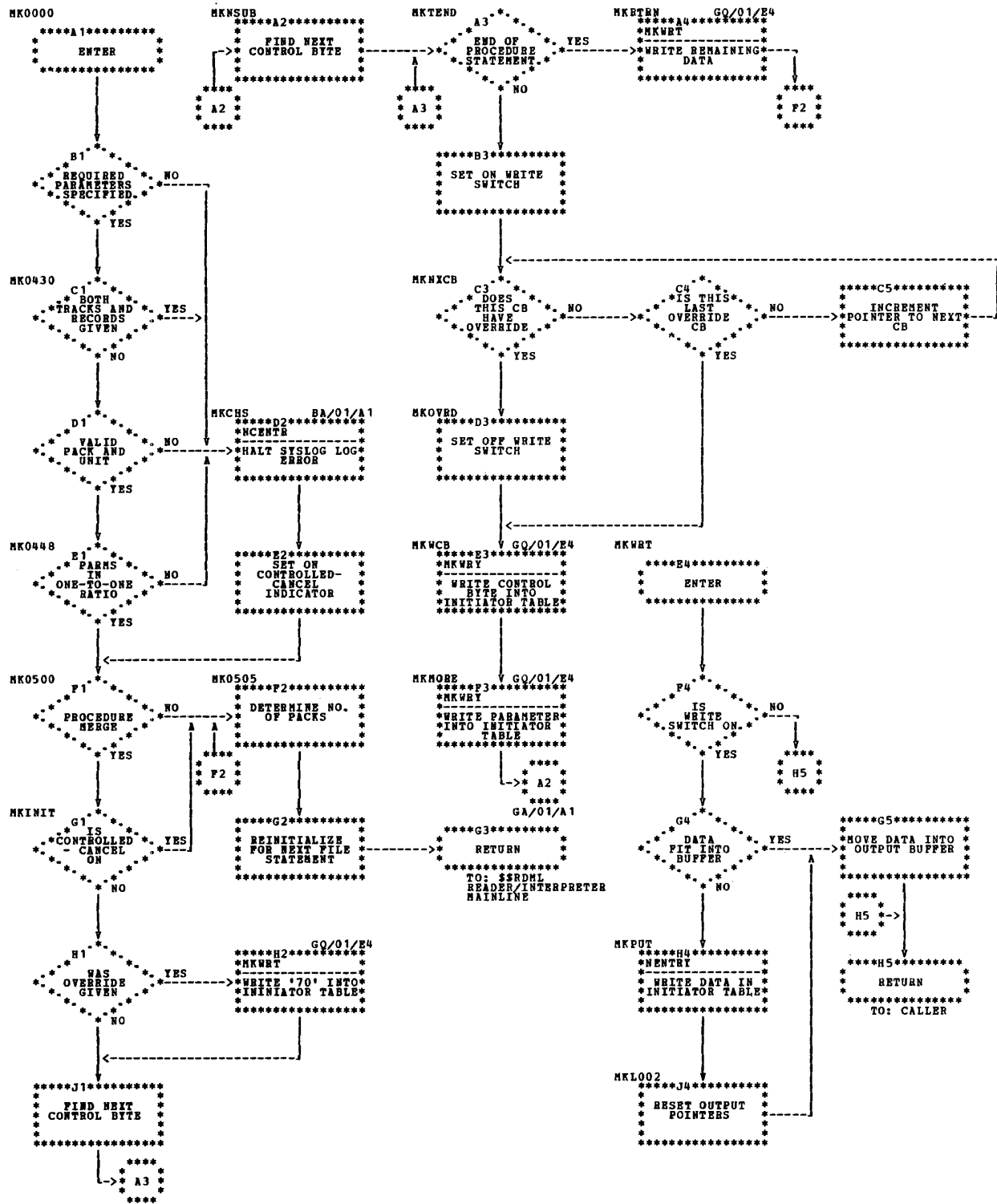


Chart GQ. FILE Diagnostics and Merge Keyword Parameters Routine (\$\$RDMK)

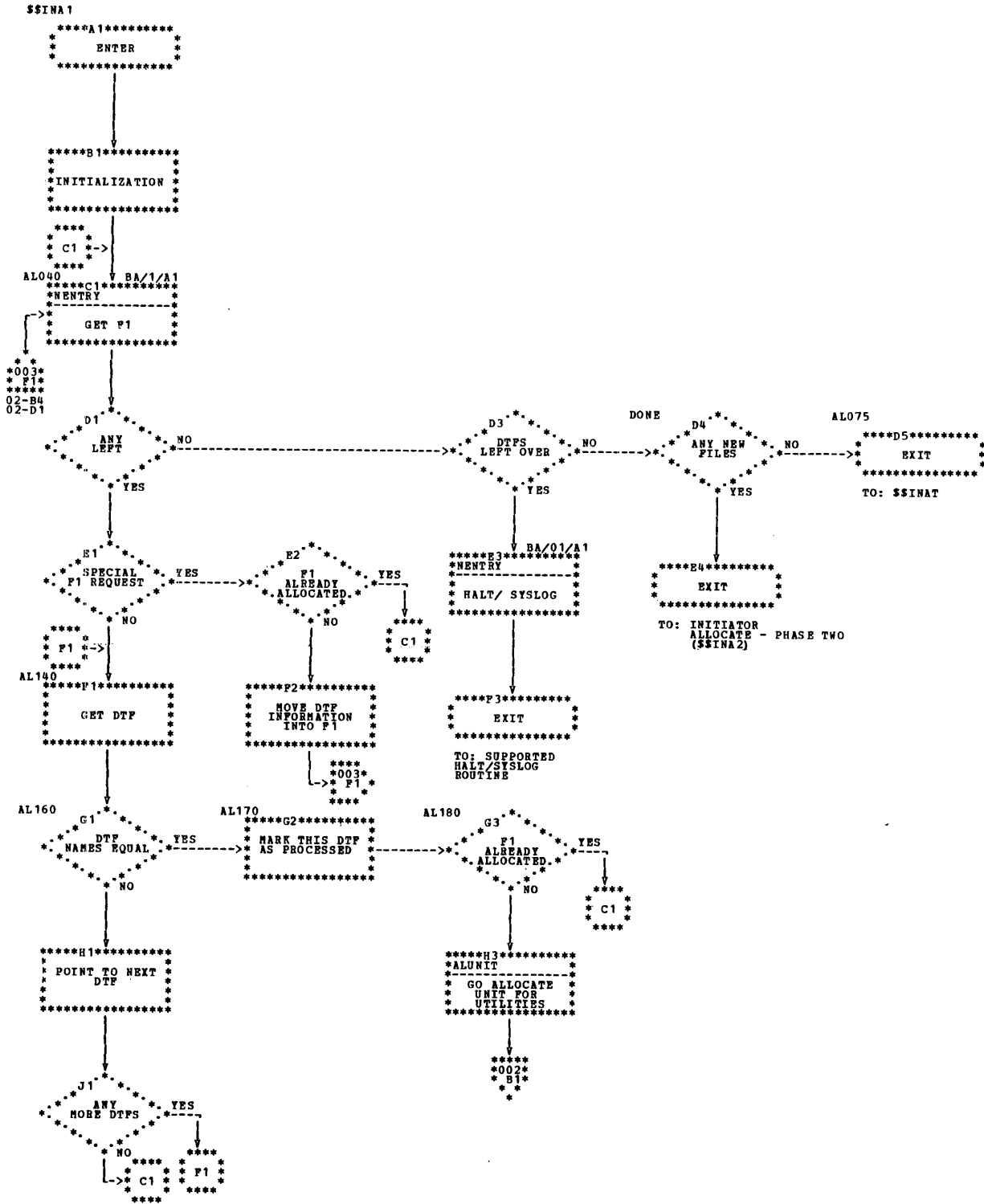


Chart HA (Part 1 of 3). Initiator Allocate-Phase One (\$\$INA1)

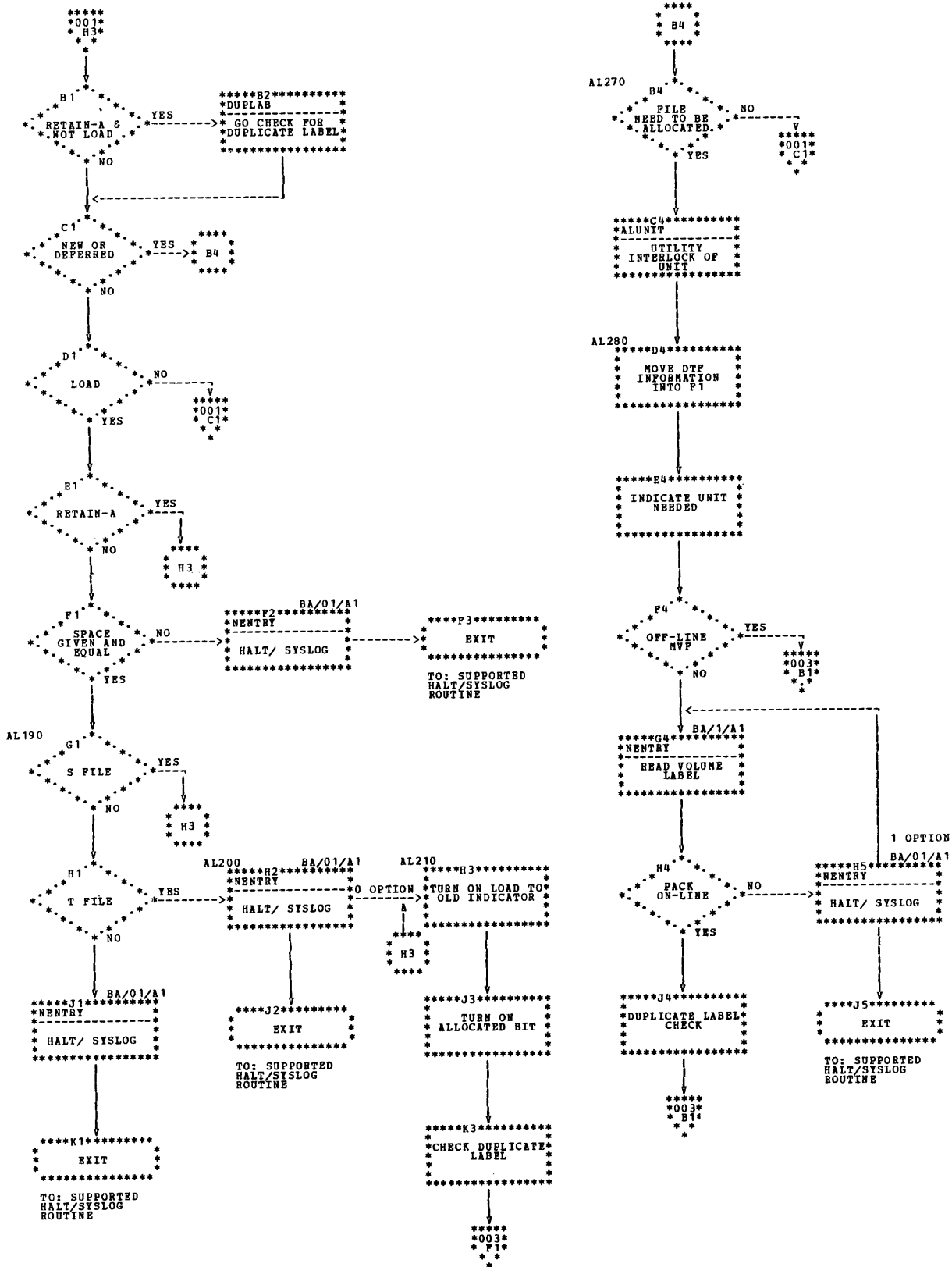


Chart HA (Part 2 of 3). Initiator Allocate-Phase One (S\$INA1)

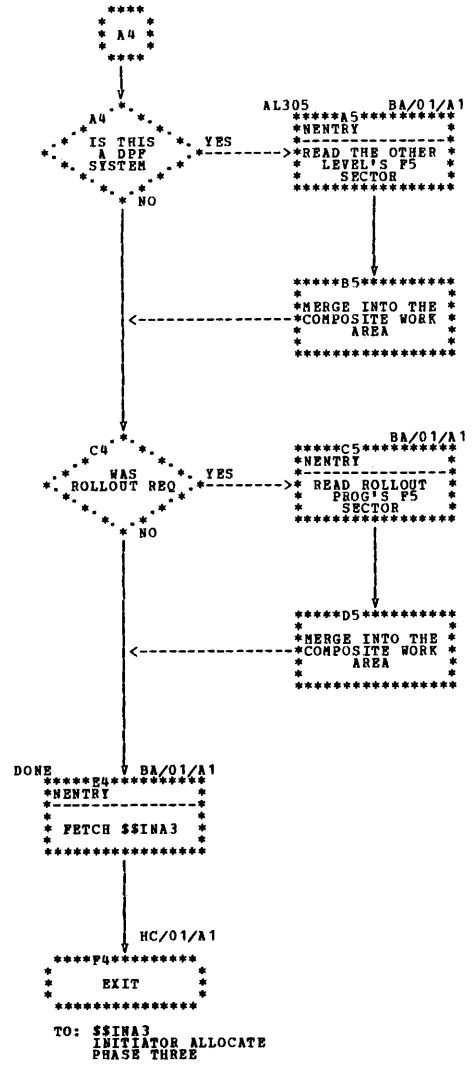
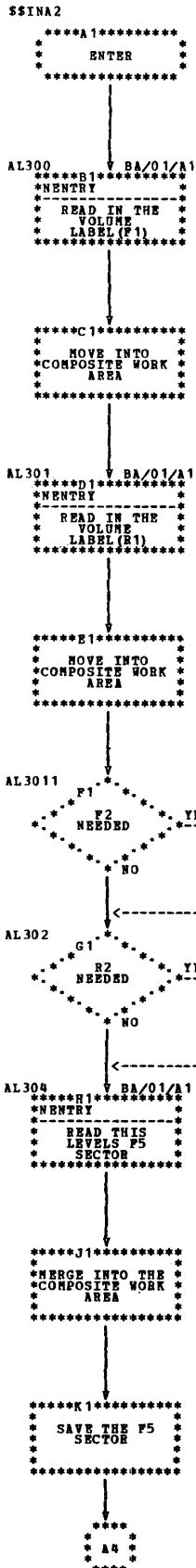


Chart HB. Initiator Allocate—Phase Two (\$\$INA2)

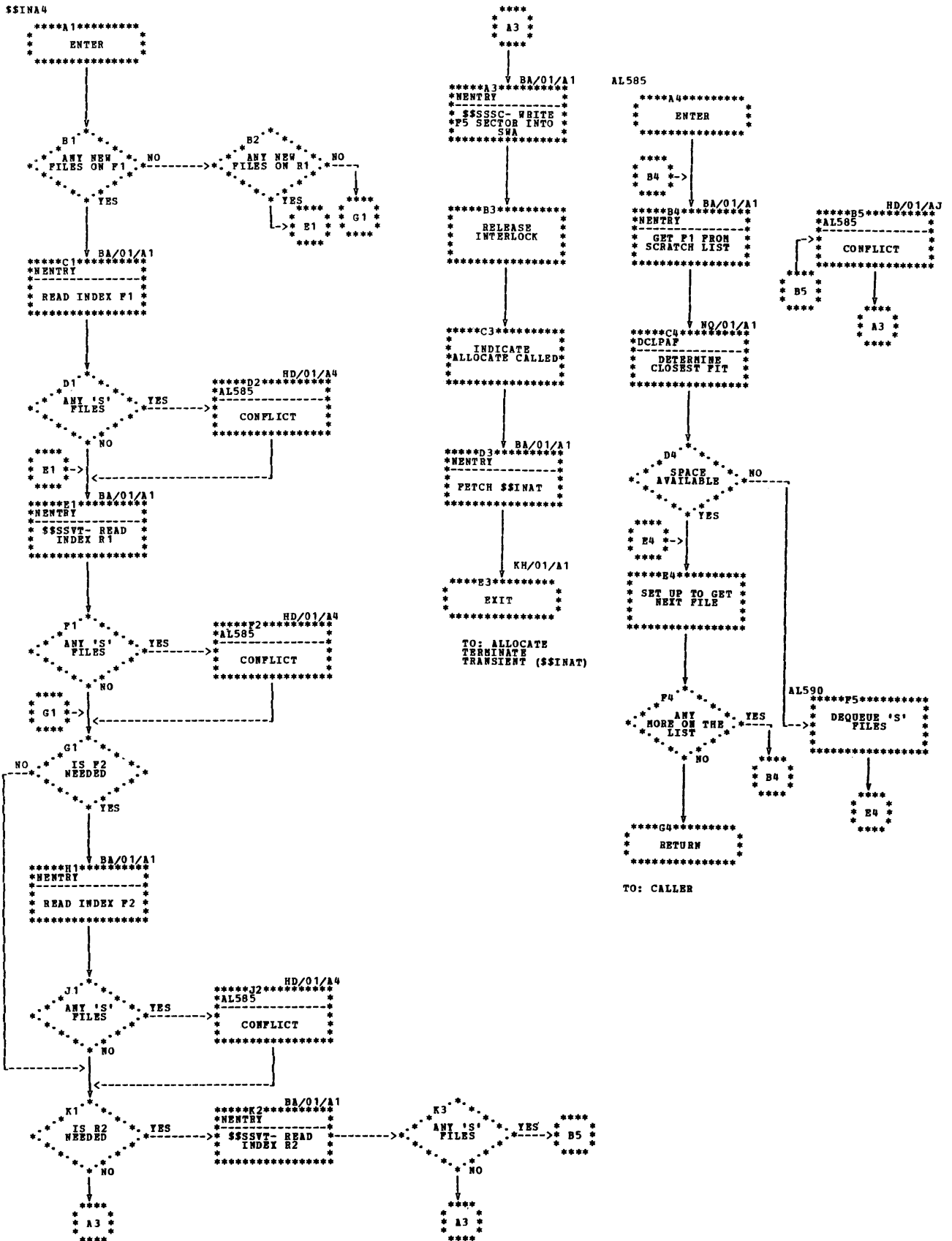


Chart HD. Initiator Allocate-Phase Four (\$\$INA4)

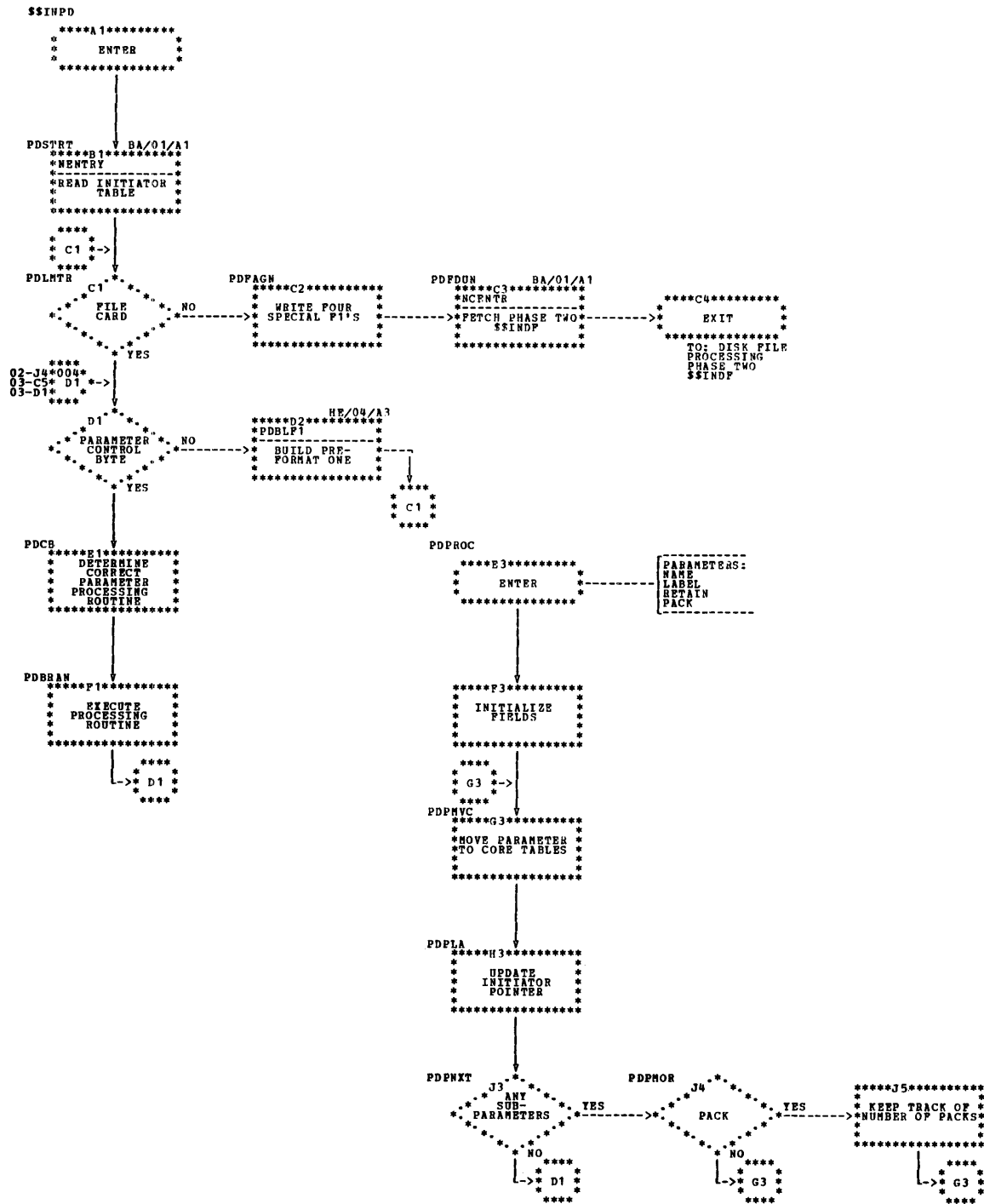


Chart HE (Part 1 of 4). Initiator Disk File Processor-Phase One (\$\$INPD)

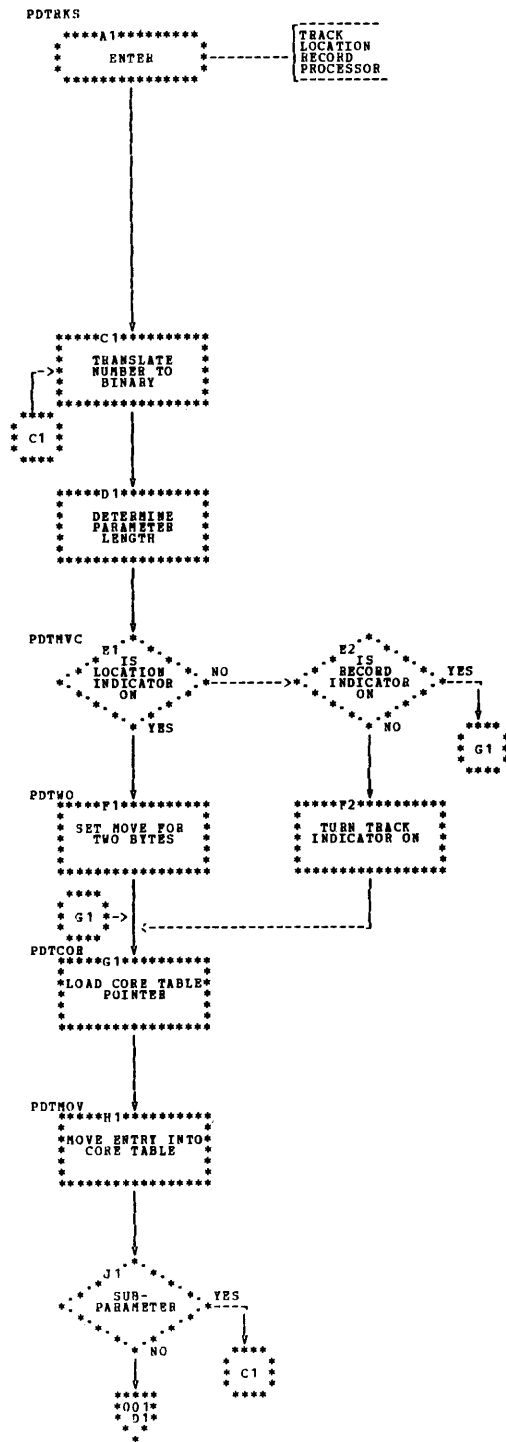


Chart HE (Part 2 of 4). Initiator Disk File Processor-Phase One (\$\$INPD)

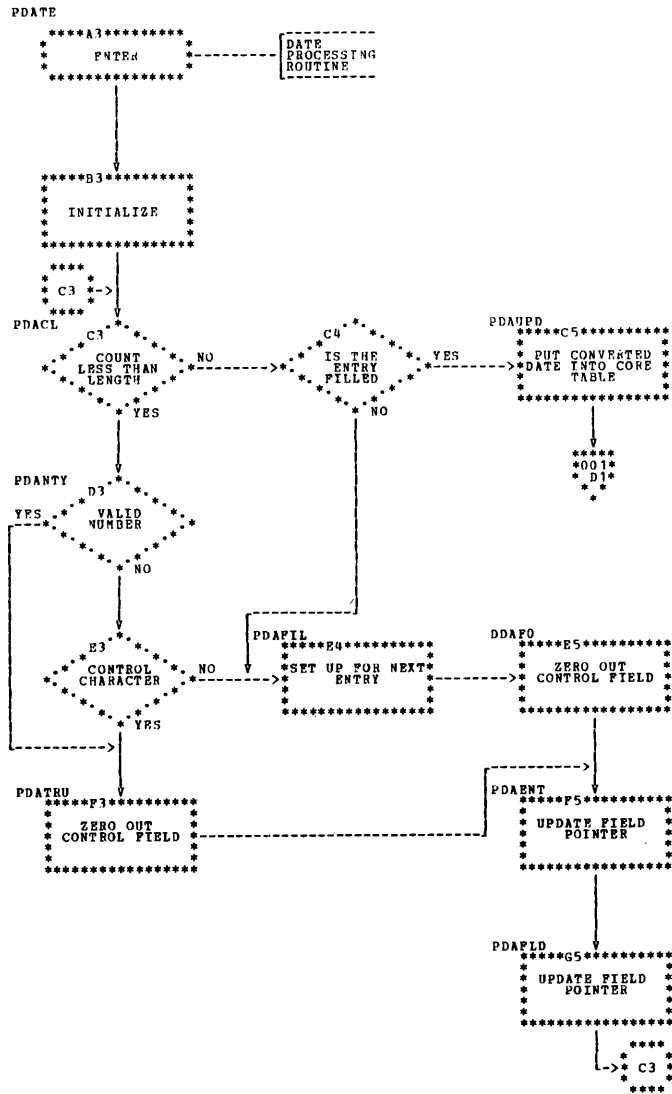
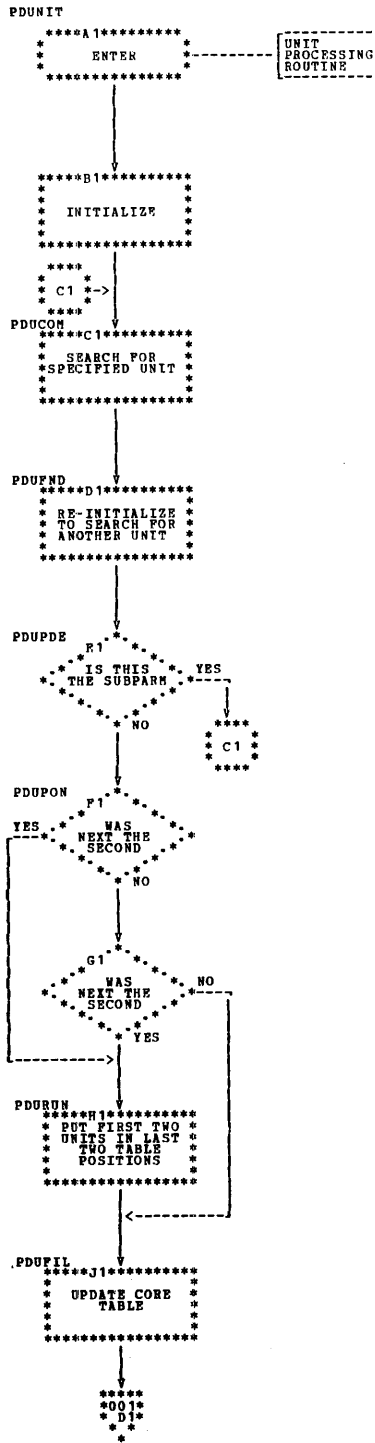


Chart HE (Part 3 of 4). Initiator Disk File Processor—Phase One (\$\$INPD)

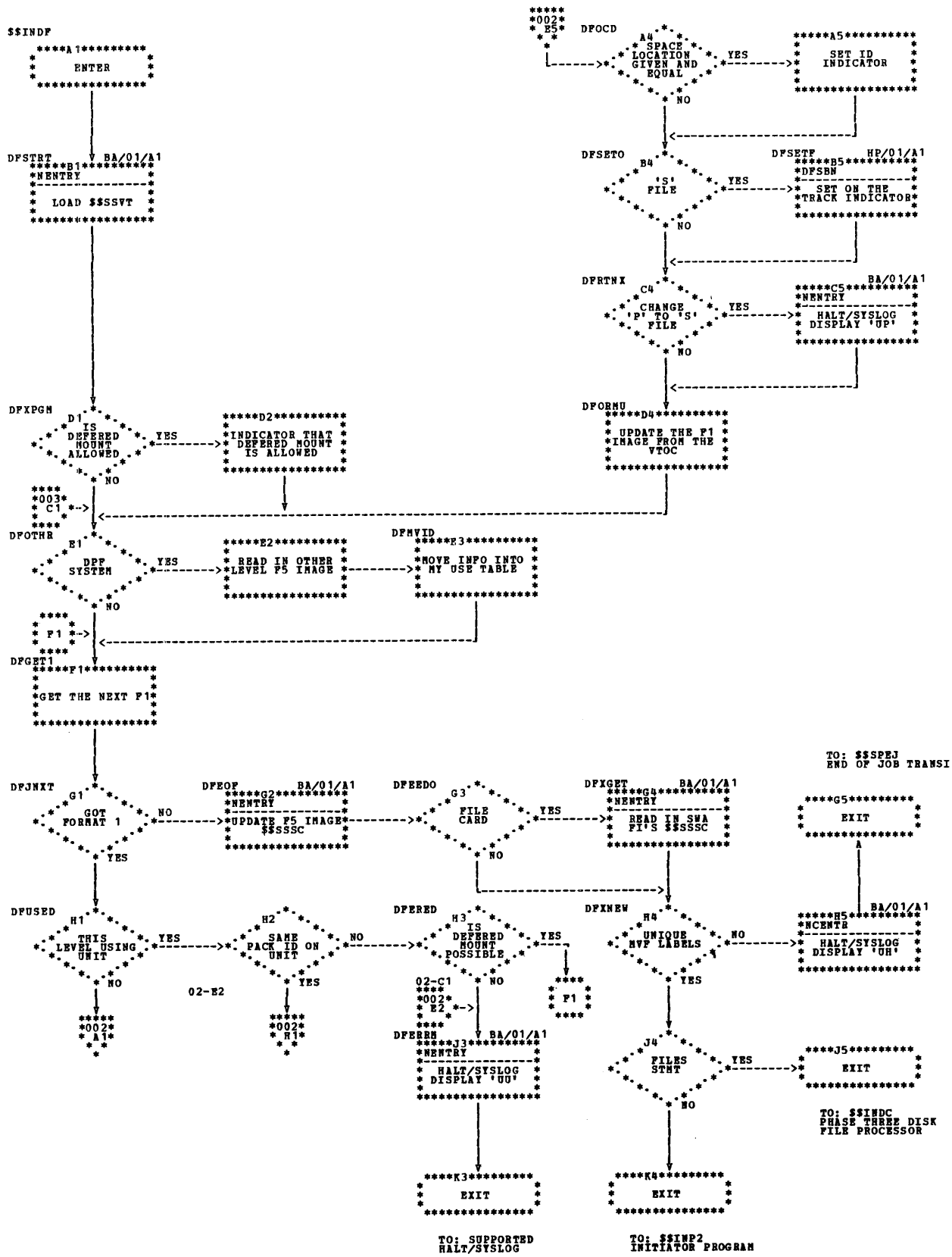


Chart HF (Part 1 of 3). Initiator Disk File Processor--Phase Two (\$\$INDF)

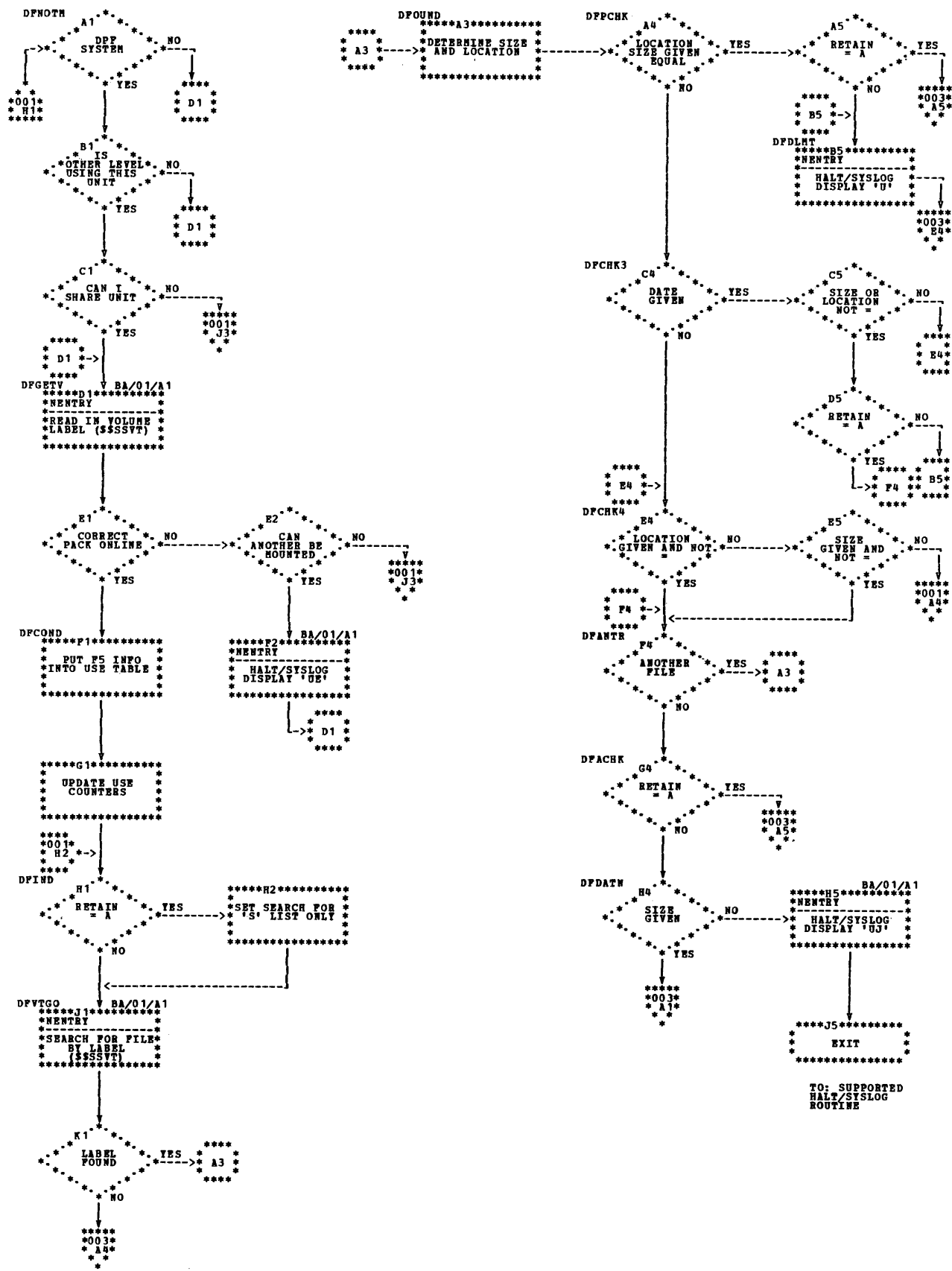


Chart HF (Part 2 of 3). Initiator Disk File Processor—Phase Two (\$\$INDF)

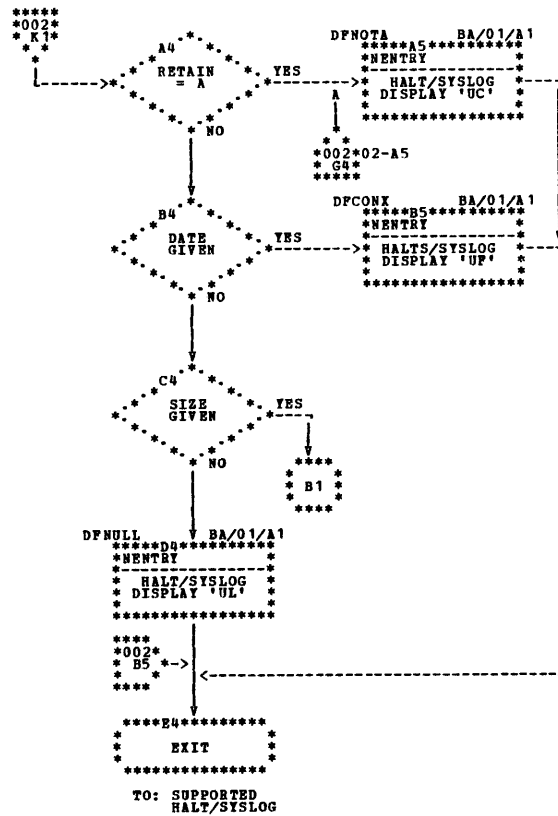
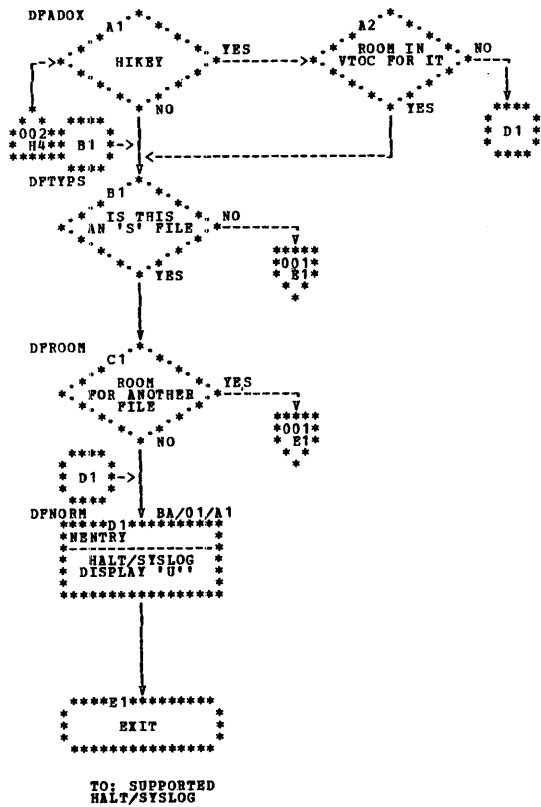


Chart HF (Part 3 of 3). Initiator Disk File Processor—Phase Two (\$\$INDF)

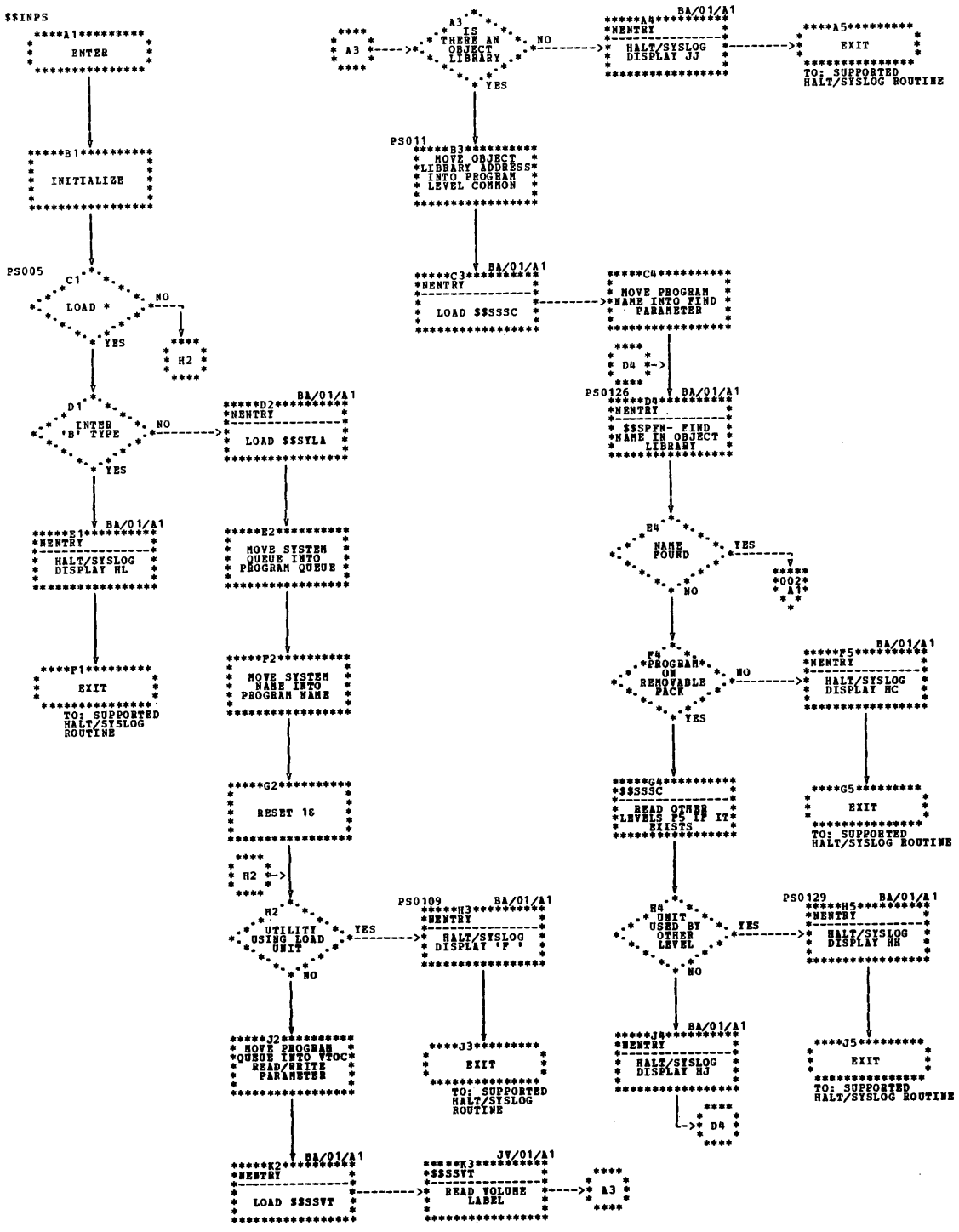


Chart HI (Part 1 of 3). Initiator Program Setup—Phase One (\$\$INPS)

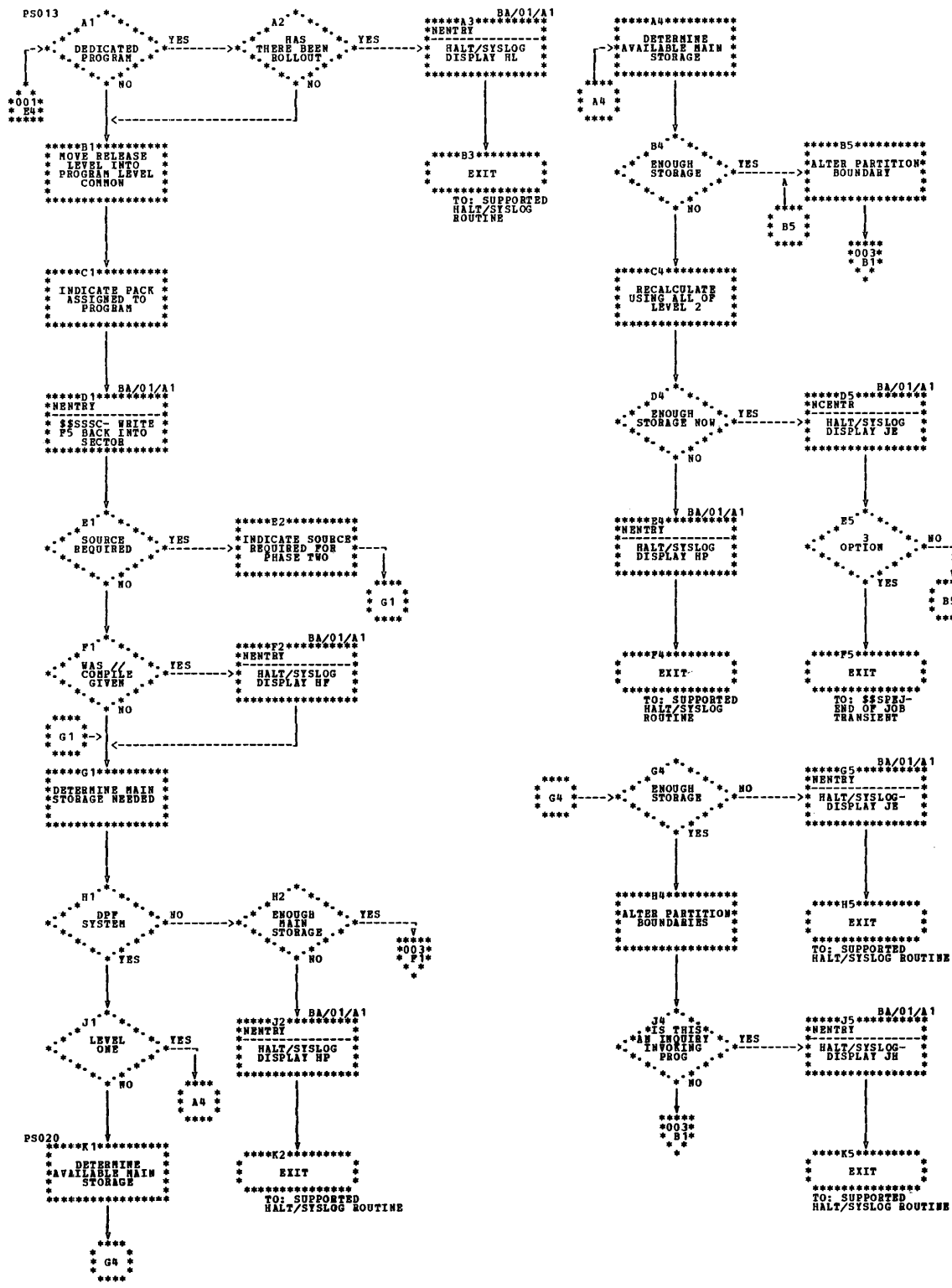


Chart HI (Part 2 of 3). Initiator Program Setup-Phase One (\$\$INPS)

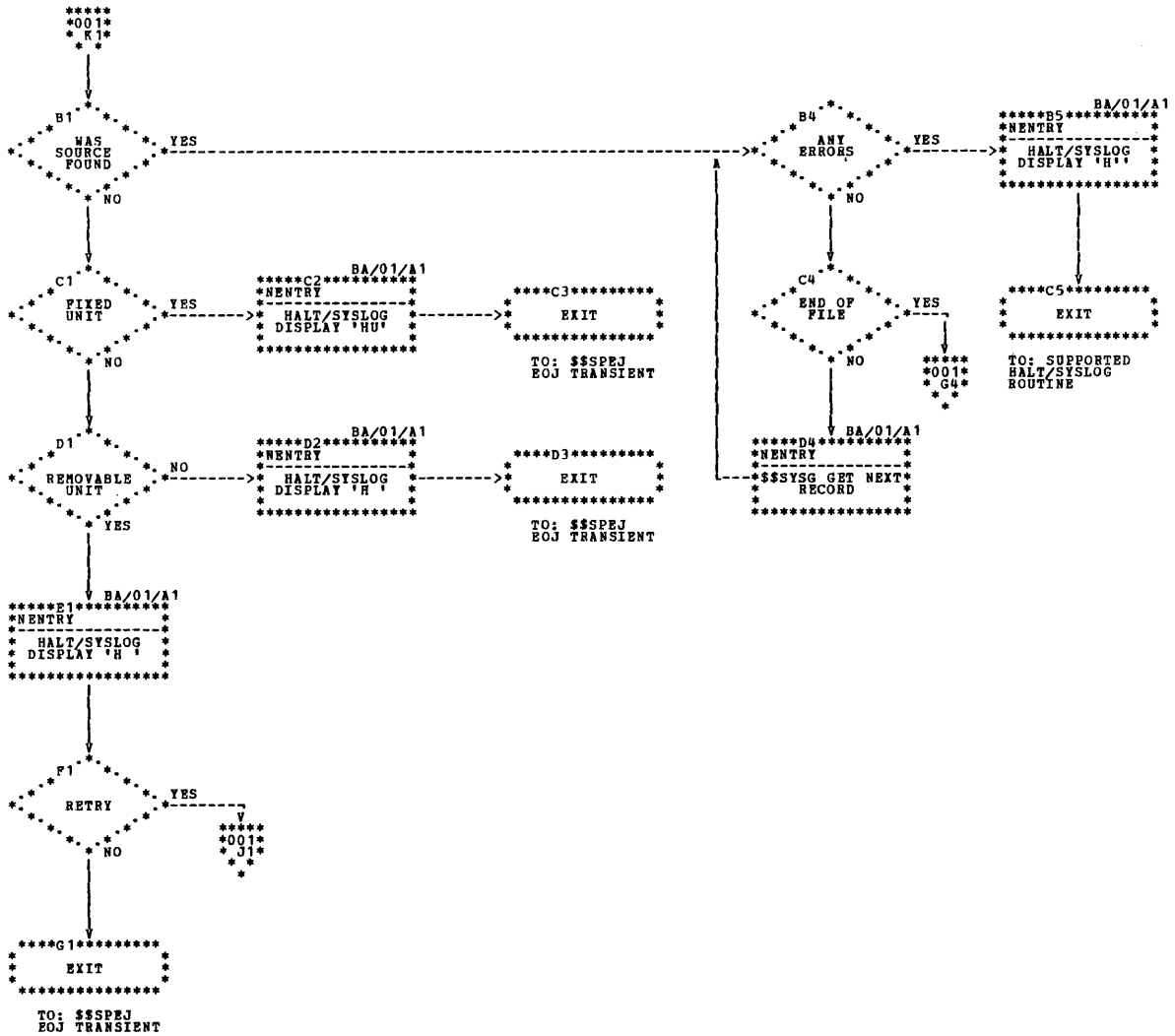


Chart HJ (Part 2 of 2). Initiator Program Setup—Phase Two (\$\$INP2)

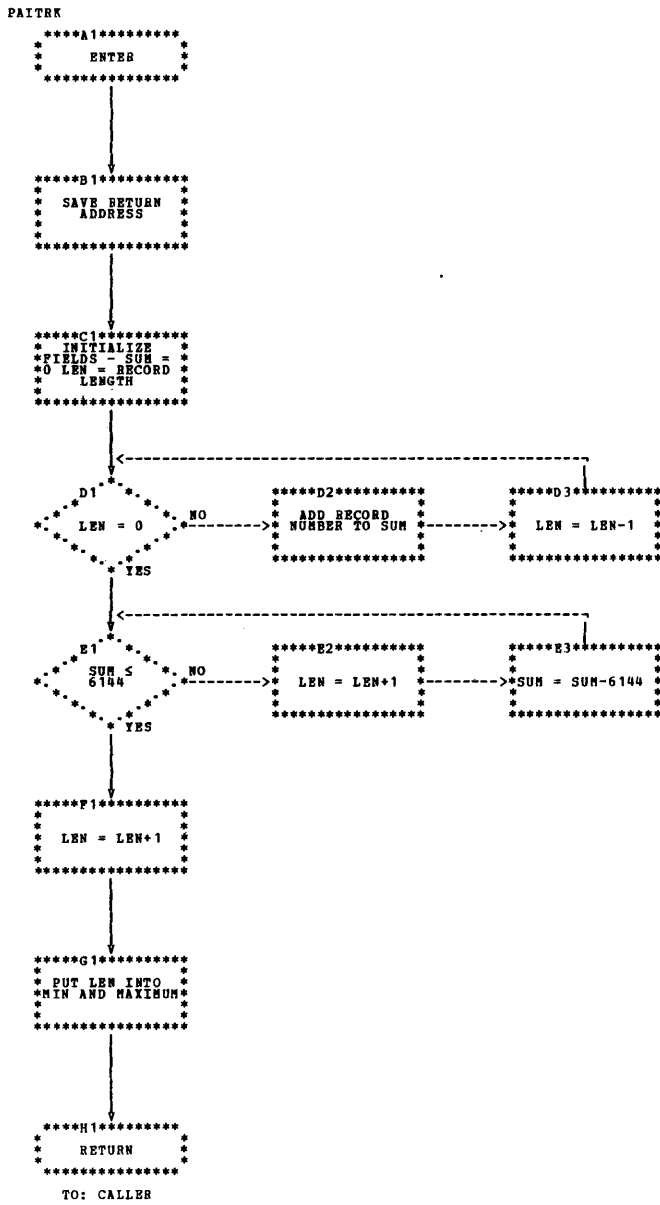


Chart HL. Initiator Allocate Support—Convert Records to Tracks (PAITRK)

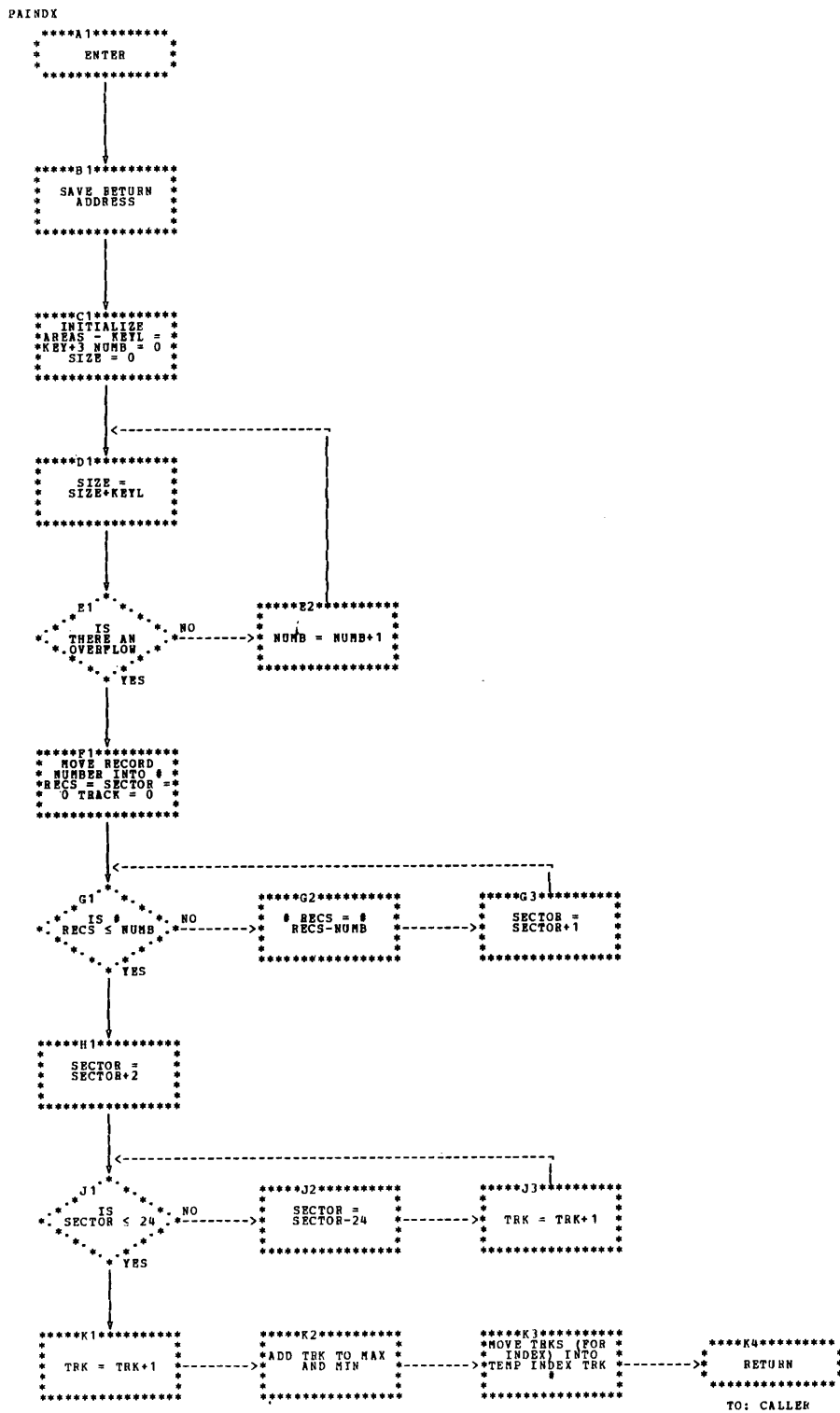


Chart HM. Initiator Allocate Support—Determine Size of Index (PAINDX)

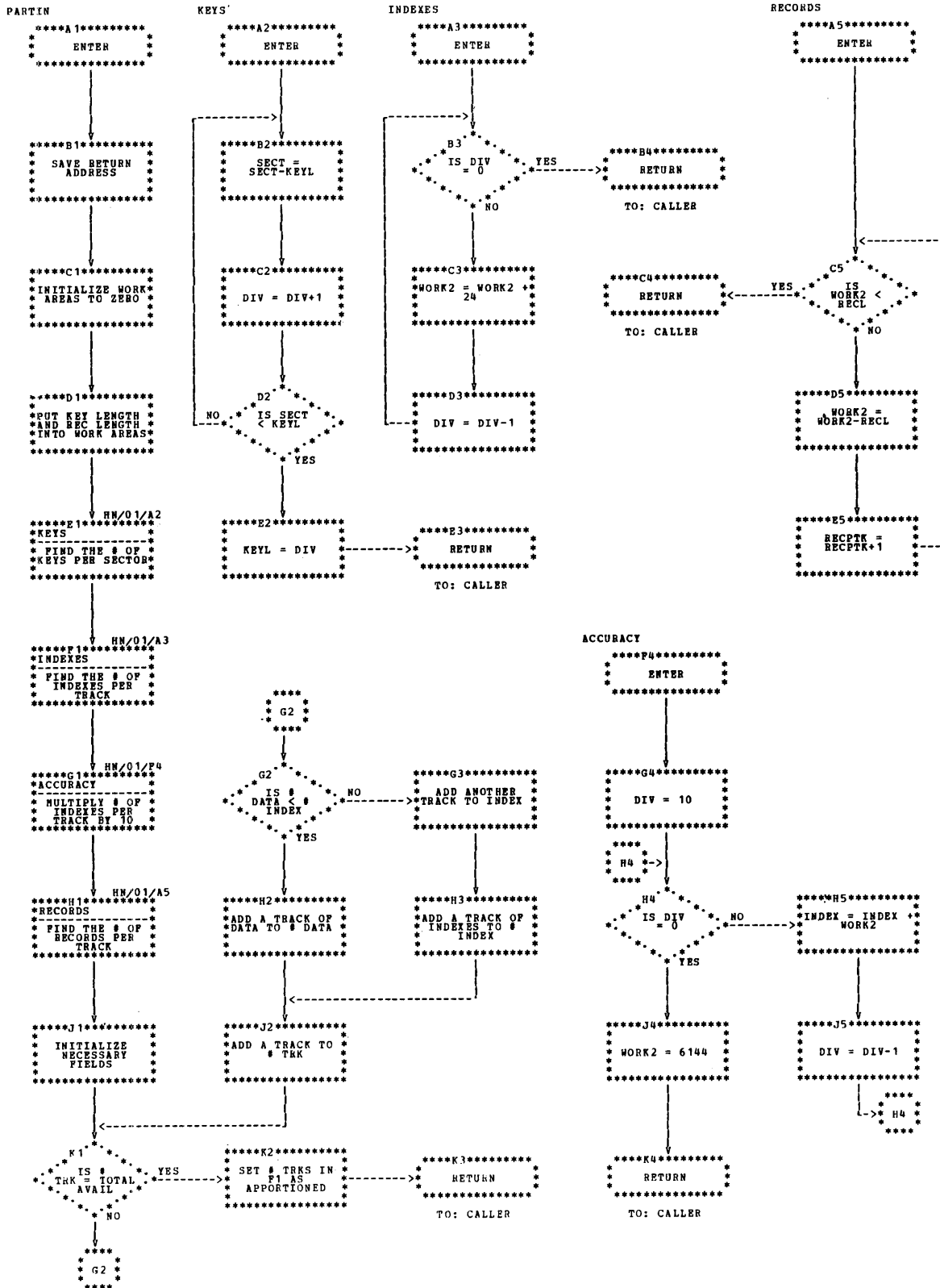


Chart HN. Initiator Allocate Support-Track Apportioning (PARTIN)

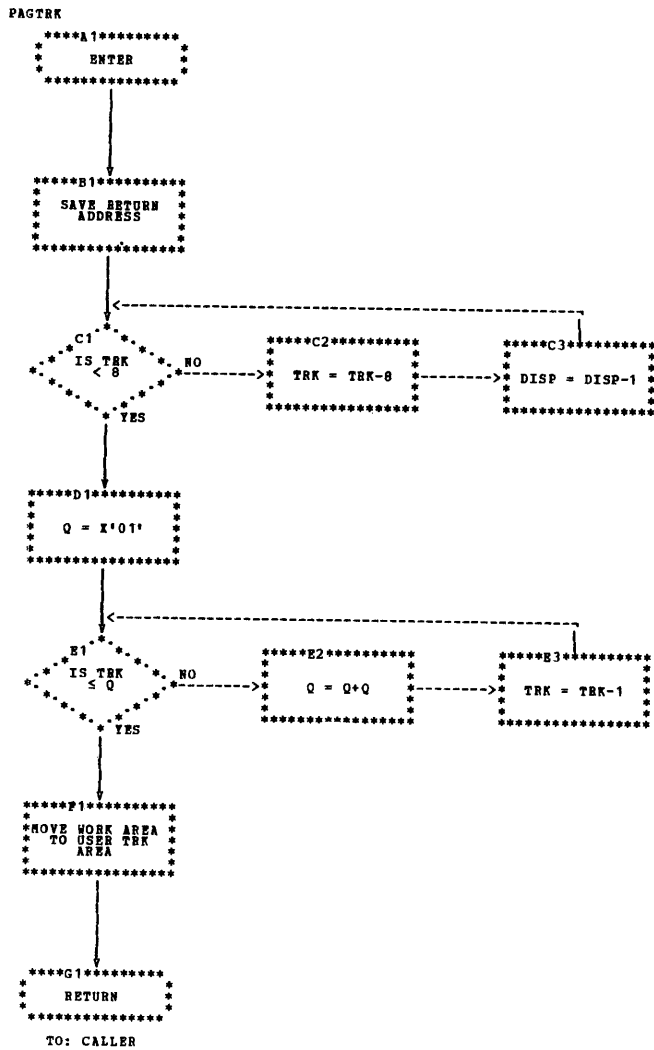
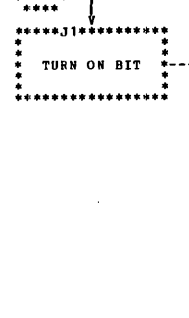
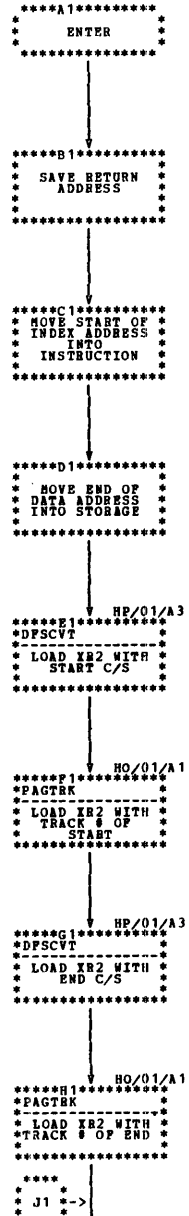


Chart HO. Initiator Allocate Support-Track Conversion (PAGTRK)

DPSBN



DFSCVT

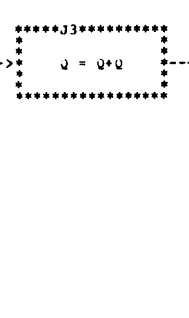
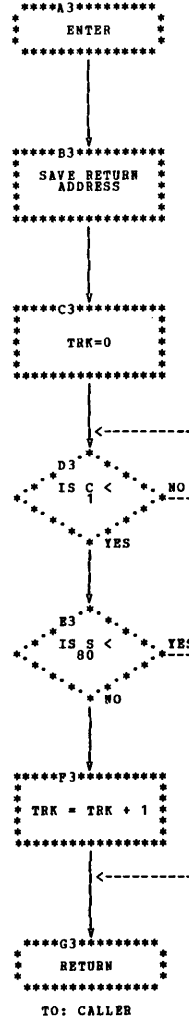


Chart HP. Initiator Allocate Support-Set Bits On (DPSBN)

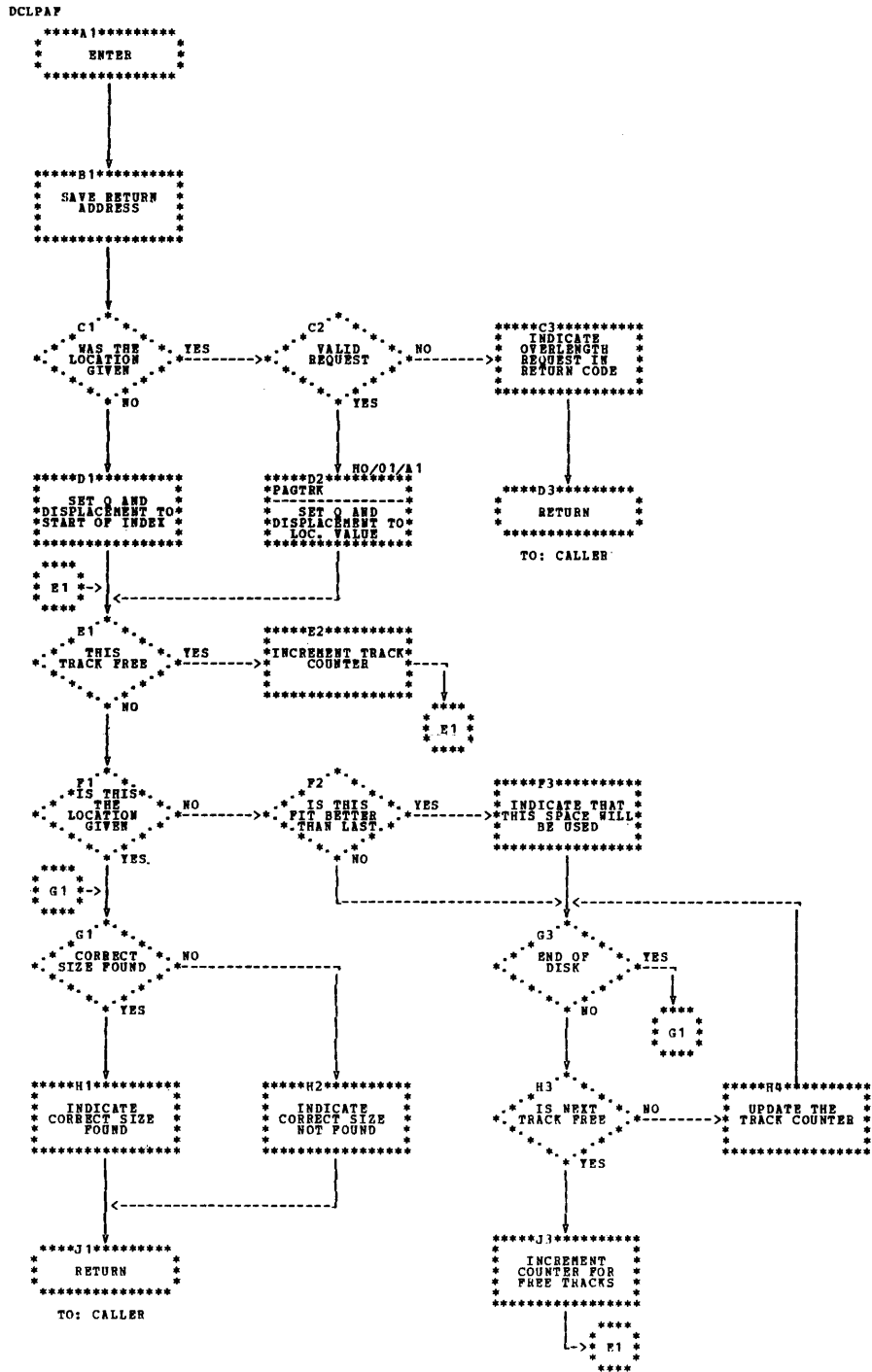


Chart HQ. Initiator Allocate Support—Determine Possible Allocation Fit (DCLPAF)

Figure 5-56 contains the Directory entries for the Scheduler.

Terminator			
Phase Name	Chart ID	Descriptive Name	Function
\$\$TMIP	EA	Terminator IPL Initialization	<ul style="list-style-type: none"> ● Initializes the scheduler area of the system communication region ● Allocates the system input/output devices
\$\$TMEO	EB	Terminator End of Job—Phase One	<ul style="list-style-type: none"> ● Performs the initial setup of the SWA
\$\$TMEJ	EC	Terminator End of Job—Phase Two	<ul style="list-style-type: none"> ● Creates an entry in the VTOC for every new file that was used during the past phase ● Updates the existing VTOC
\$\$TMRI	ED	Terminator Re-Initialization	<ul style="list-style-type: none"> ● Re-initializes the: Swa, program level and system communication regions
\$\$TMSB	EE	Terminator Standby Routine	<ul style="list-style-type: none"> ● Issues end of job halts _____ DPF _____ ● Ensures that the current level has enough main storage ● Expands to a minimum of 4K if current level is level two ● Determines the status of the Sysin device
\$\$TMSI	EF	Terminator Sysin Initialization	<ul style="list-style-type: none"> ● Set up the program level communication region to reflect the system input device
\$\$TMSK	EG	Terminator Key Sort	<ul style="list-style-type: none"> ● Determines the type of format one labels
\$\$TMDS	EH	Terminator Duplicate Key Scan	<ul style="list-style-type: none"> ● Reads the key of all indexed load or add files ● Scans the keys for duplicates
Reader/Interpreter			
Model 6			
\$\$RBIP	FA	Conversational Scheduler Initializer	<ul style="list-style-type: none"> ● Determines mode, conversational or disk system (D) ● Passes control to \$\$RBCO for conversational mode, to \$\$RDML for D mode
\$\$RBCO	FB	Scheduler Control	<ul style="list-style-type: none"> ● Loads Control Table ● Writes encoded control statements to the SWA ● Determines cycle type: LOAD, BUILD, CALL, or BUILD (chained)
\$\$RBBC	FC	Build-Chained Cycle	<ul style="list-style-type: none"> ● Prompts operator for responses to construct chained procedure
\$\$RBCC	FD	Call Control	<ul style="list-style-type: none"> ● Prompts operator for call name and unit ● Gets first record or procedure ● Passes control to Call (\$\$RBCL) ● Determines if Chained procedure was called
\$\$RBCL	FE	Call Cycle	<ul style="list-style-type: none"> ● Loads the processing modules for the OCL in the procedure ● Writes the OCL to the SWA ● Identifies type of statement ● Writes compressed statements to SWA
\$\$RBCP	FF	Positional Statement Processor	<ul style="list-style-type: none"> ● Processes positional type parameters
\$\$RBCD	FG	Data Type Statement Processor	<ul style="list-style-type: none"> ● Processes data type statements
\$\$RBCK	FH	Check Keyword	<ul style="list-style-type: none"> ● Processes keyword type statements

Figure 5-56. Scheduler Directory (Part 1 of 4)

Phase Name	Chart ID	Descriptive Name	Function
\$\$RBM Y	FI	Modify	<ul style="list-style-type: none"> Allows the operator to change and/or add to OCL given in LOAD, BUILD, or CALL
\$\$RBM1	FJ	Procedure Library Scheduler Interface	<ul style="list-style-type: none"> Reads responses from SWA and writes them to the procedure library
\$\$RBM2	FK	Utility OCL Processor	<ul style="list-style-type: none"> Includes and modifies utility OCL during BUILD, then writes it to the source library Displays and modifies utility OCL during CALL
\$\$RBRN	FL	Run Processor	<ul style="list-style-type: none"> Writes initiator information to the SWA
\$\$RBIN	FM	Interaction Routine	<ul style="list-style-type: none"> Completes building of the prompt buffers If error occurs, sets up error code to pass to Error Logging Routine (\$\$SLSE)
\$\$RBDS	FN	Date/Switch	<ul style="list-style-type: none"> Obtains OCL for Date and Switch statements during LOAD and BUILD Syntax Checks the OCL Writes correct OCL to the SWA
\$\$RBSW	FO	Switch Processor	<ul style="list-style-type: none"> Sets UPSI switches
\$\$RBDT	FP	Date Check	<ul style="list-style-type: none"> Syntax checks date parameters
\$\$RBLG	FQ	Log	<ul style="list-style-type: none"> Prompts for logging device Finds supported halt/syslog module Finds supported sysin module Writes the C/S of the halt/syslog and sysin modules to the system communication area
\$\$RBFM	FR	Forms	<ul style="list-style-type: none"> Prompts for the device and the number of lines per page Writes the device and the number of lines per page to the system communication area
\$\$RBHI	FS	HiKey	<ul style="list-style-type: none"> Prompts and processes responses for HIKEYs during LOAD, BUILD, or CALL
Disk System			
\$\$RDML	GA	Reader/Interpreter Mainline	<ul style="list-style-type: none"> Reads control cards Performs syntax check on control cards Encodes OCL statements containing keyword parameters Finds override statements to procedures Calls the appropriate control card processor
\$\$RDLD	GB	// LOAD Control Card Processor	<ul style="list-style-type: none"> Performs syntax check on // LOAD control card Places name and unit in program level communication region
\$\$RDRN	GC	// RUN Control Card Processor	<ul style="list-style-type: none"> Closes initiator table Sets up for reading procedure additions Checks the cancel condition code indicator Changes the system mode from INTRA to INTER
\$\$RDFL	GD	// FILE Control Card Processor	<ul style="list-style-type: none"> Insures the syntactical correctness of the // FILE control card Loads the control card data into the scheduler work area
\$\$RDDT	GE	// DATE Control Card Processor	<ul style="list-style-type: none"> Insures the syntactical correctness of the // DATE control card Loads the date found on the control card so that it can be used as the system date for the current program

Figure 5-56. Scheduler Directory (Part 2 of 4)

Phase Name	Chart ID	Descriptive Name	Function
\$\$RDSW	GF	// SWITCH Control Card Processor	<ul style="list-style-type: none"> ● Insures the syntactical correctness of the // SWITCH control card ● Sets the corresponding UPSI switches to reflect the control card
\$\$RDIM	GG	// IMAGE Control Card Processor	<ul style="list-style-type: none"> ● Modifies the chain image area to reflect the current print chain configuration
\$\$RDCM	GH	// COMPILE Control Card Processor	<ul style="list-style-type: none"> ● Syntactically checks the encoded parameters ● Places source name and unit in the scheduler work area
\$\$RDFM	GI	// FORMS Control Card Processor	<ul style="list-style-type: none"> ● Indicates the number of printable lines per page on the system printer
\$\$RDLG	GJ	// LOG Control Card Processor	<ul style="list-style-type: none"> ● Ensures the syntactical correctness of the // LOG control card ● Indicates the system's logging device
\$\$RDPS	GK	// PAUSE Control Card Processor	<ul style="list-style-type: none"> ● Issues an advance program level HALT
\$\$RDHN	GL	// HALT, // NOHALT Control Card Processor	<ul style="list-style-type: none"> ● Indicates whether or not a normal end of job HALT will occur
\$\$RDRR	GM	// READER Control Card Processor	<ul style="list-style-type: none"> ● Ensures that the specified device is correct ● Indicates the specified device as the system input device
\$\$RDPN	GN	// PARTITION Control Card Processor	<ul style="list-style-type: none"> ● Determines if the requested partition change can be supported ● Indicates the maximum amount of program level 2 storage
\$\$RDCL	GO	// CALL Control Card Processor	<ul style="list-style-type: none"> ● Ensures the syntactical correctness of the // CALL control card ● Locates the specified procedure ● Reads procedure additions and/or overrides into the scheduler work area
\$\$RDS3	GP	Positional Parameter Syntax Scan	<ul style="list-style-type: none"> ● Scans a specified area for parameters ● Encodes the parameters ● Ensures that the encoded parameters are syntactically correct
\$\$RDMK	GQ	FILE Diagnostics and Merge Keyword Parameters	<ul style="list-style-type: none"> ● Does end of statement diagnostics on // FILE statement ● Merges procedure // FILE statements and override // FILE statements
Initiator			
\$\$INA1	HA	Initiator Allocate—Phase One	<ul style="list-style-type: none"> ● Calculate the required disk storage needed for the format one images that have DTFs, updating SWA
\$\$INA2	HB	Initiator Allocate—Phase Two	<ul style="list-style-type: none"> ● Builds a composite image of all tracks allocated on all units needed for file allocating
\$\$INA3	HC	Initiator Allocate—Phase Three	<ul style="list-style-type: none"> ● Allocates file space in format ones
\$\$INA4	HD	Initiator Allocate—Phase Four	<ul style="list-style-type: none"> ● Dequeues all S files ● Recalls the Allocate Termination (\$\$INAT)
\$\$INPD	HE	Initiator Disk File Processor—Phase One	<ul style="list-style-type: none"> ● Decodes the // FILE control cards ● Transfers the // FILE information from the control card to the format one pre-image located in the SWA

Figure 5-56. Scheduler Directory (Part 3 of 4)

Phase Name	Chart ID	Descriptive Name	Function
\$\$INDF	HF	Initiator Disk File Processor—Phase Two	<ul style="list-style-type: none"> ● Performs pre-processing of disk files by creating format one images in the scheduler work area (SWA)
\$\$INDC	HG	Initiator Disk File Processor—Phase Three	<ul style="list-style-type: none"> ● Determines the current program level ● Ensures that within the program level, no two files have the same name, and that the file label and date is unique for each pack
\$\$INDS	HH	Initiator Disk File Processor—Phase Four	<ul style="list-style-type: none"> ● Sorts the format one images into the order required by the Allocate routine (\$\$INA1)
\$\$INPS	HI	Initiator Program Setup—Phase One	<ul style="list-style-type: none"> ● Determines if the program can run
\$\$INP2	HJ	Initiator Program Setup—Phase Two	<ul style="list-style-type: none"> ● Moves source information into a source file ● Allocates disk storage for the source and work files ● Fetches requested program
\$\$INMS	HL-HQ	Initiator Allocate—Support Routines	<ul style="list-style-type: none"> ● PAITRK—Converts records to tracks ● PAINDX—Determines the size of the index ● PARTIN—Apportions tracks ● PAGTRK—Converts tracks ● DFSBN—Sets bits on/off ● DCLPAF—Locates possible allocation fit

Figure 5-56. Scheduler Directory (Part 4 of 4)



PART 6.

TRANSIENT AND SCHEDULER SUPPORT



The routines that are described in this part of the logic manual are called by more than one program. The *Method of Operation* and *Data Area Formats* associated with the following routines are described within the discussion of the calling program. This part of the document contains only the Program Organization and Directory sections.

The following section describes, in detail, the organization of the Transient and Scheduler Support routines. Included in the following discussions are:

- Descriptions containing: name of the routine, input, output, function, and exits (both normal and error) from the routine
- Main storage map showing the routines that are found in main storage at the same time as the routine being described
- Flowcharts showing the functional flow of the routine being described
- Parameter list (when applicable) needed for the use of the routine being described

► System Input Device—Console (\$\$STIC)

CHART: JA
 FIGURE: 6-1, 6-2
 ENTRY POINT: \$\$STIC
 FUNCTION: Provides the logic needed to support the Console as a System input (Sysin) device.
 INPUT: Register 2 contains the address of a 7-byte parameter list shown in Figure 6-2.
 OUTPUT:
 — The function byte (first byte in parameter list) is changed to a return code for the caller.
 — The record is returned to the user in the buffer pointed to by CURENT.
 EXIT: Calling routine after the record has been read in
Note: \$\$STIC can be used within the user's core as well as within the transient area.

► System Input Device—MFCU/1442 (\$\$STIM)

CHART: JB
 FIGURE: 6-1, 6-2
 ENTRY POINT: \$\$STIM
 FUNCTION: Provides the logic needed to support the MFCU/1442 as a system input (Sysin) device.
 INPUT: Register 2 contains the address of a 7-byte parameter list shown in Figure 6-2.
 OUTPUT:
 — The function byte (first byte in parameter list) is changed to a return code for the caller.
 — The record is returned to the user in the buffer pointed to by CURENT. The buffer pointed to by NEXT is being filled, if double buffering is requested.
 EXIT: Calling routine after the record has been read in.
Note: \$\$STIM can be used within user's core as well as within the transient area.

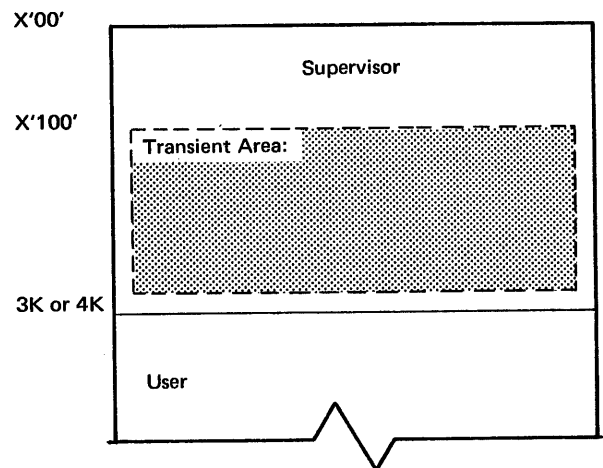


Figure 6-1. Main Storage Map showing Location of Transient Area

Byte	
0	1 2 3 4 5 6
Byte	Contents
0	This byte is used as a means of communication between the user and the Sysin routine.
	Function
X'00'	Indicates, swap the two buffers; wait on current and start filling the next buffer.
X'01'	Indicates that there is not a previous request pending.
X'02'	Indicates, swap the two buffers; wait on the current buffer, return when done — do not start on the next buffer.
X'04'	Determine the two buffer addresses and the work area address from a block of 271 bytes on a 128-byte boundary.
X'08'	This is a single buffering request; start filling the next buffer and wait until done, then swap the buffer addresses.
X'F0'	ENTER— allowed during conversational OCL entry.
X'E0'	ENTER— is not allowed during conversational OCL entry.
X'10'	ENTER— is allowed during non-conversational OCL entry.
	For all other codes of the form X'x0', ENTER— is not allowed.
	Return Codes
X'20'	The current request is complete. Input was terminated by ENTER—.
X'40'	The buffer pointed to by CURENT is ready. The buffer pointed to by NEXT is being loaded if double buffering is being used.
X'50'	End of file for buffer pointed to be CURENT.
X'60'	Hardware error. Cancel job.
X'80'	/* or /& was previously read. Only the Reader/Interpreter can now read from the device.
1-2	This area contains the address (at label NEXT) of the next buffer. The buffer is 96 bytes on a 128-byte boundary.
3-4	This area contains the address (at label CURENT) of the current buffer. The buffer is 96 bytes on a 128 byte boundary.
5-6	This area contains the address of a 47-byte work area.

Figure 6-2. Parameter List for the System Input Device Routines

► **Model 6 System Input Device—Keyboard/Printer (\$\$STIP, \$\$STIS)**

CHART: JC

FIGURE: 6-1

ENTRY POINTS:

- \$\$STIP
- \$\$STIS

FUNCTION:

- Provides the logic to support the Keyboard/Printer as a Sysin device.
- Provides Console support, input from the keyboard to be displayed on the printer.

INPUT: XR2 points to the sysin parameter list.

OUTPUT: Input is placed in the current buffer.

EXIT: To the next sequential instruction of the caller

► **Model 6 System Input Device—Data Recorder (\$\$STID)**

CHART: JD

FIGURE: 6-1

ENTRY POINT: \$\$STID

FUNCTION: Provides the logic to support the data recorder as a Sysin device by:

1. Building a DTF in the sysin work area as follows:

Byte		Byte
1	Data Recorder DTF	30
31	IOB 1	35
36	IOB 2	40
41	Register Save Area	44
45	Reserved	47

2. Calling the data management module that reads the cards (\$\$SLZO) (see *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Program Logic Manual, SY21-0512*).

INPUT: XR2 points to the sysin parameter list.

OUTPUT: Input from the Data Recorder is placed in the current buffer.

EXIT: To the next sequential instruction of the caller

► **Model 6 System Input Device—CRT (\$\$STIT, \$\$STIX)**

CHART: JE

FIGURE: 6-1

ENTRY POINTS:

- \$\$STIT
- \$\$STIX

FUNCTION: This routine supports CRT displays. It performs two transient loads:

1. \$\$STIT
2. \$\$STIX

INPUT: XR2 points to the sysin parameter list.

OUTPUT: Updated sysin data buffer and CRT displays on screen

EXIT: The next sequential instruction of the caller

► **System Output Printer Error Recovery (\$\$STEP)**

CHART: JF

FIGURE: 6-1

ENTRY POINT: \$\$STEP

FUNCTION:

- Calls the IOS Error Logging Routine (\$\$DLOG) to log any error found on the 22LC Printer.
- Issues a halt to indicate the error condition to the operator.

INPUT: The Q, R, and SNS values are passed from the Halt/Syslog—Printer routine (\$\$STOP).

OUTPUT: An error log parameter list is passed to the IOS Error Logging Routine (\$\$DLOG). See the *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0512*.

EXIT:

- Normal: Halt/Syslog—Printer routine (\$\$STOP)
- Error: EOJ transient (\$\$SPEJ)

► **Halt/Syslog—Console Printer (\$\$STOC)**

CHART: JG

FIGURE: 6-1

ENTRY POINT: \$\$STOC

FUNCTION: Supports the console printer as a system logging (Syslog) device by performing any of the following four functions:

1. Output only.
2. Halt with system message.
3. System message only.
4. Halt only.

INPUT: Register 2 contains the address of the Halt/Syslog parameter list. See *Appendix B. Diagnostic Aids* for Halt/Syslog parameter list format.

OUTPUT: Halt and/or message printed on the console printer.

EXIT:

- Normal:
 1. Calling routine after completion of the system message
 2. \$\$STOH if a halt was requested
- Error: End-of-Job transient (\$\$SPEJ) if an immediate cancel was requested.
Note: \$\$STOC can be used within user's core as well as within the transient area.

► **Halt/Syslog—Printer (\$\$STOP)**

CHART: JH

FIGURE: 6-1

ENTRY POINT: \$\$STOP

FUNCTION: Supports the printer as a system logging (Syslog) device by performing any of the following four functions:

1. Program output only.
2. Halt with system message.
3. System message only.
4. Halt only.

INPUT: Register 2 contains the address of a parameter list. See *Appendix B. Diagnostic Aids* for the format of the Halt/Syslog format.

OUTPUT: Halt and/or message printed on the 22LC printer

EXIT:

- Calling routine after normal completion
- System Output—Error Recovery (\$\$STEP) if a printer error occurs
- System Halt transient (\$\$STOH) if a halt is requested
Note: \$\$STOP can be used within user's core as well as within the transient area.

► **System Halt Transient (\$\$STOH)**

CHART: JI
 FIGURE: 6-1
 ENTRY POINT: \$\$STOH
 FUNCTION:

- Converts the halt codes from printable characters to valid stick light combinations.
- Checks for DPF system.
- Calls the End-of-Job transient (\$\$SPEJ) if an immediate cancel is requested.
- Passes control to the resident halt routine (NPHALT) if the system is operating with the Dual Programming Feature (DPF).

INPUT: Register 2 contains the address of the Halt/Syslog parameter list. See *Appendix B. Diagnostic Aids* for Halt/Syslog parameter format.

OUTPUT: The reply is returned to the caller in the fifth byte of the parameter list.

EXIT:

- Resident Halt/Syslog routine (NPHALT) if the system is operating in the DPF mode. Control is passed to the EOJ transient (\$\$SPEJ) if an immediate cancel is requested.
- All other conditions cause control to be returned to the caller, after the halt is reset.

► **Model 6 Printer Syslog Error Recovery Procedures Routine (\$\$STP6)**

CHART: JJ
 FIGURE: 6-1
 ENTRY POINT: \$\$STP6
 FUNCTION: Senses printer errors and attempts recovery.
 INPUT: None (called by \$\$STIP or \$\$STON)
 Output: Halt display
 EXIT:

- Normal: To the next sequential instruction of the caller
- Error: To End of Job transient (\$\$SPEJ)

► **Halt/Syslog-Halt Code Conversion Routine (\$\$STOK)**

CHART: JK
 FIGURE: 6-1
 ENTRY POINT: \$\$STOK
 FUNCTION: Converts Disk System halt characters to Model 6 halt characters and Model 6 halt codes.

The Halt Code Conversion Routine (\$\$STOK) converts two Disk System halt characters to a unique 9-bit binary equivalent, then converts the binary equivalent to a Model 6 halt instruction and to the Model 6 halt characters. The Model 6 halt characters are placed in the system buffer to be printed if requested. The Model 6 halt instruction is put in the program level communication region to be passed to Model 6 Halt/Syslog-Halt Routine (\$\$STOS).

The halt character conversion requires five steps:

1. Determine the displacement in the table (see Figure 6-3) of the first character.
2. Multiply the displacement by 22.
3. Look up the displacement of the second character and add it to the result of step 2.
4. Convert this decimal number to its hexadecimal value.
5. Write the 9-bit binary equivalent of the hex value. The bits on indicate the display lights on.

For example, to convert an 'E2' halt:

E displacement = 3
 2 displacement = 4

3 x 22 = 66
 66 + 4 = 70 (dec)
 70 (dec) = 46 (hex)

46 (hex) = 0000 0100 0110
 ABCD1 2345

* **
 Model 6 halt = C 34

Note: The following are exceptions to the conversion rule:

Disk System halt converts to Model 6 halt

EJ	ABCD12345
HE	BCD12345
5Y	A CD12 5
80	34
blank0	12 45
0C	AB D12 45
7E	CD1 3
0A	AB D 2345
'Y	CD123

Any Disk System code that ends with - except for -- cannot be converted. If Model 6 receives a Disk System halt it cannot convert, Model 6 issues halt '34'.

INPUT: Disk System halt characters in the Halt/Syslog parameter list (addressed by XR2)

OUTPUT:

- Model 6 halt characters in the system buffer
- Model 6 halt codes in the transient storage area

EXIT: Return to next sequential instruction in calling halt/syslog routine

Character		8	6	E	2	0	C	U	L	J	9	5	3	Y	A	P	F	H	4	'	7	1	-
Displacement	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
1	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
2	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
3	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
4	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
5	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
6	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
7	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
8	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
9	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511



Figure 6-3. Halt Conversion Tables

► **Model 6 Halt/Syslog Halt Routine—Keyboard (\$\$STOS)**

CHART: JL

FIGURE: 6-1

ENTRY POINTS:

- \$\$STOS
- OMTEST

FUNCTION: Issues HPL instruction and accepts options from the keyboard.

INPUT:

- Model 6 halt code in program level communication region at displacement X'52'
- Options allowed in parameter list (XR2 points to the parameter list)

OUTPUT: Reply option in parameter list

EXIT:

- Normal: Return to caller
- Error: End of Job routine (\$\$SPEJ)

► **Model 6 Halt/Syslog Routine—Matrix Printer (\$\$STOM)**

CHART: JM

FIGURE: 6-1

ENTRY POINT: \$\$STOM

FUNCTION: Performs halt/syslog functions for the matrix printer.

INPUT: XR2 points to the halt/syslog parameter list

OUTPUT: System message printed

EXIT:

- \$\$STOK
- \$\$STOS

► **Model 6 Halt/Syslog Routine—Output Only, Matrix Printer (\$\$STON)**

CHART: JN

FIGURE: 6-1

ENTRY POINT: \$\$STON

FUNCTION: Performs output functions for the matrix printer.

INPUT: XR2 points to the halt/syslog parameter list

OUTPUT: Requested printed output

EXIT: To the next sequential instruction of the calling program

► **Model 6 Halt/Syslog Halt Routine—CRT (\$\$STOT)**

CHART: JO

FIGURE: 6-1

ENTRY POINT: \$\$STOT

FUNCTION: Performs halt/syslog functions using the CRT for display.

INPUT: XR2 points to the halt/syslog parameter list

OUTPUT: CRT display

EXIT:

- \$\$STOK to translate halt code
- \$\$STOS to handle halt response
- Next sequential instruction of caller

► **Model 6 Halt/Syslog Routine—CRT On and Off Transient (\$\$STOX)**

CHART: JP

FIGURE: 6-1

ENTRY POINT: \$\$STOX

FUNCTION:

- Start CRT display when X'00' is encountered in the parameter list.
- Stop CRT display when X'80' is encountered in the parameter list.
- Set a return code of X'41' when any other command byte is encountered in the parameter list.
- Set return code of X'40' when the start or stop function is performed correctly.

INPUT: XR2 points to the halt/syslog parameter list

OUTPUT: Return code of X'40' or X'41'

EXIT: To the next sequential instruction of the caller

► **Model 6 Halt/Syslog Scheduler Error Messages Routine (\$\$SSLSE, \$\$SLS1)**

CHART: JQ, JR

FIGURE: 6-1

ENTRY POINTS:

- \$\$SSLSE
- \$\$SLS1

FUNCTION: Sets up halt/syslog to display all conversational scheduler error messages.

INPUT: XR2 points to a syslog buffer to be used

OUTPUT:

- Updated halt/syslog parameter list
- Message to be printed is in buffer

EXIT: To halt/syslog routine

► Scheduler Work Area—Get (\$SSGT)

CHART: JS

FIGURE: 6-1, 6-4

ENTRY POINT: \$SSGT

FUNCTION: Retrieves the next consecutive logical record from the SWA.

INPUT: Register 2 contains the address of the request parameter list. See Figure 6-4 for the SWA-GET parameter format.

OUTPUT: The next consecutive record is moved into the user's input/output area.

EXIT:

- Normal: Calling routine
 - Error: A halt is issued and displayed on the halt lights.
- Note:* \$SSGT can be used within user's core as well as within the transient area.

Register 2 →	Function Byte 1	Function Byte 2	Completion Code	I/O Area Pointer (2 Bytes)
Function Byte 1	<p>Bit 0 (X'80') 0 – Must be 0</p> <p>1 (X'40') 0 – Format one operation 1 – Utility control card operation</p> <p>2 (X'20') 0 – Not used 1 – Not used } setting must be 0</p> <p>3 (X'10') Level definition (this bit setting determines the meaning of bit 5)</p> <p>0 bit 5 (X'04') – 0 = level 1 1 = level 2</p> <p>1 bit 5 (X'04') – 0 = my level 1 = other level</p> <p>4 (X'08') On for first operation of its type</p> <p>5 (X'04') See bit 3</p> <p>6 (X'02') Update (format ones only)</p> <p>7 (X'01') On if Rolled-out SWA is wanted Note: Bit 5 must be off.</p>			
Function Byte 2	Length byte indicating the number of bytes in this logical record (1-255)			
Completion Code	X'40' = Successful completion X'80' = End of file			
I/O Pointer	Contains the address of the leftmost byte of the user I/O area			

Figure 6-4. Parameter List for Scheduler Work Area—Get Routine

► Scheduler Work Area—Put (\$\$\$SPT)

CHART: JT

FIGURE: 6-1, 6-5

ENTRY POINT: \$\$\$SPT

FUNCTION: This routine loads consecutive records into the SWA.

INPUT: Register 2 contains the address of the request parameter list. See Figure 6-5 for the format of the parameter list.

OUTPUT: The record is placed in the next consecutive area within the SWA.

EXIT:

- Normal: Calling routine
 - Error: A halt is issued and displayed on the halt lights
- Note:* \$\$\$SPT can be used within user's core as well as within the transient area.

Register 2 →		Function Byte 1	Function Byte 2	Completion Code	I/O Area Pointer (2 Bytes)
Function Byte 1	Bit	0 (X'80')	0 – 1 – Initiator Table		
		1 (X'40')	0 – Format one operation 1 – Utility control card operation		
		2 (X'20')	0 – Not used 1 – Not used	} setting must be 0	
		3 (X'10')	Level definition (this bit setting determines the meaning of bit 5)		
			0 bit 5 (X'04') – 0 = level 1 1 = level 2		
			1 bit 5 (X'04') – 0 = my level 1 = other level		
		4 (X'08')	On for first operation of its type		
		5 (X'04')	See bit 3		
		6 (X'02')	Update (format ones and utility cards only)		
		7 (X'01')	Not used		
Function Byte 2	Length byte indicating the number of bytes in hex in this logical record (1-255)				
Completion Code	X'40' = Successful completion X'80' = End of file				
I/O Pointer	Contains the address of the leftmost byte of the user I/O area				

Figure 6-5. Parameter List for Scheduler Work Area—Put Routine

► Scheduler Work Area—Read/Write (\$SSSSC)

CHART: JU

FIGURE: 6-1, 6-6

ENTRY POINT: \$SSSSC

FUNCTION:

- Reads consecutive sectors from the SWA.
- Writes consecutive records into the SWA.

INPUT: Register 2 contains the address of the request parameter list. See Figure 6-6 for the format of this parameter list.

OUTPUT: The consecutive records are read from, or written into the SWA.

EXIT:

- Normal: Calling routine
 - Error: A halt is issued and displayed on the halt lights
- Note:* \$SSSSC can be used within user's core as well as within the transient area.

Register 2 →		Function Byte 1	Function Byte 2	Completion Code	I/O Area Pointer (2 Bytes)
Function Byte 1		Bit 0 (X'80')	On for write sector	} One or the other is on.	
		Bit 1 (X'40')	On for read sector		
		Bit 2 (X'20')	Not used		
		Bit 3 (X'10')	Level definition (this bit setting determines the meaning of bit 5)		
		0 bit 5 (X'04')	0 = level 1 1 = level 2		
		1 bit 5 (X'04')	0 = my level 1 = other level		
		Bit 4 (X'08')	On for first operation of its type		
		Bit 5 (X'04')	See bit 3		
		Bit 6 (X'02')	(Read, write—last record read) Operation depends on setting of bits 0 and 1.		
		Bit 7 (X'01')	On if rolled-out SWA is wanted. Bit 5 must be off.		
Function Byte 2		Bit 0 (X'80')	Not used		
		Bit 1 (X'40')	Work area		
		Bit 2 (X'20')	Initiator table		
		Bit 3 (X'10')	Utility statements		
		Bit 4 (X'08')	Format 1's		
		Bit 5 (X'04')	Format 5's		
		Bit 6 (X'02')	Partition index		
		Bit 7 (X'01')	Common sector		
Completion Code		40 = Successful completion			
		80 = End of file			
I/O Pointer		Points to leftmost byte of user I/O area			

Figure 6-6. Parameter List for Scheduler Work Area Read/Write Routine

► **Volume Table of Contents—Read/Write (\$SSVT)**

CHART: JV

FIGURES: 6-1, 6-7

ENTRY POINT: \$SSVT

FUNCTION: Performs the read/write functions for the following data areas. These areas are located in the volume table of contents (VTOC):

1. Volume label
2. Volume table of contents (VTOC) index
3. Configuration record
4. Format-1 records

INPUT: Register 2 contains the address of the request parameter list. See Figure 6-7 for the format of this parameter list.

OUTPUT: Register 2 contains the address of the parameter list found in Figure 6-7.

EXIT:

- Normal: Calling routine
 - Error: A halt is issued and displayed on the halt lights
- Note:* \$SSVT can be used within user's core as well as within the transient area.

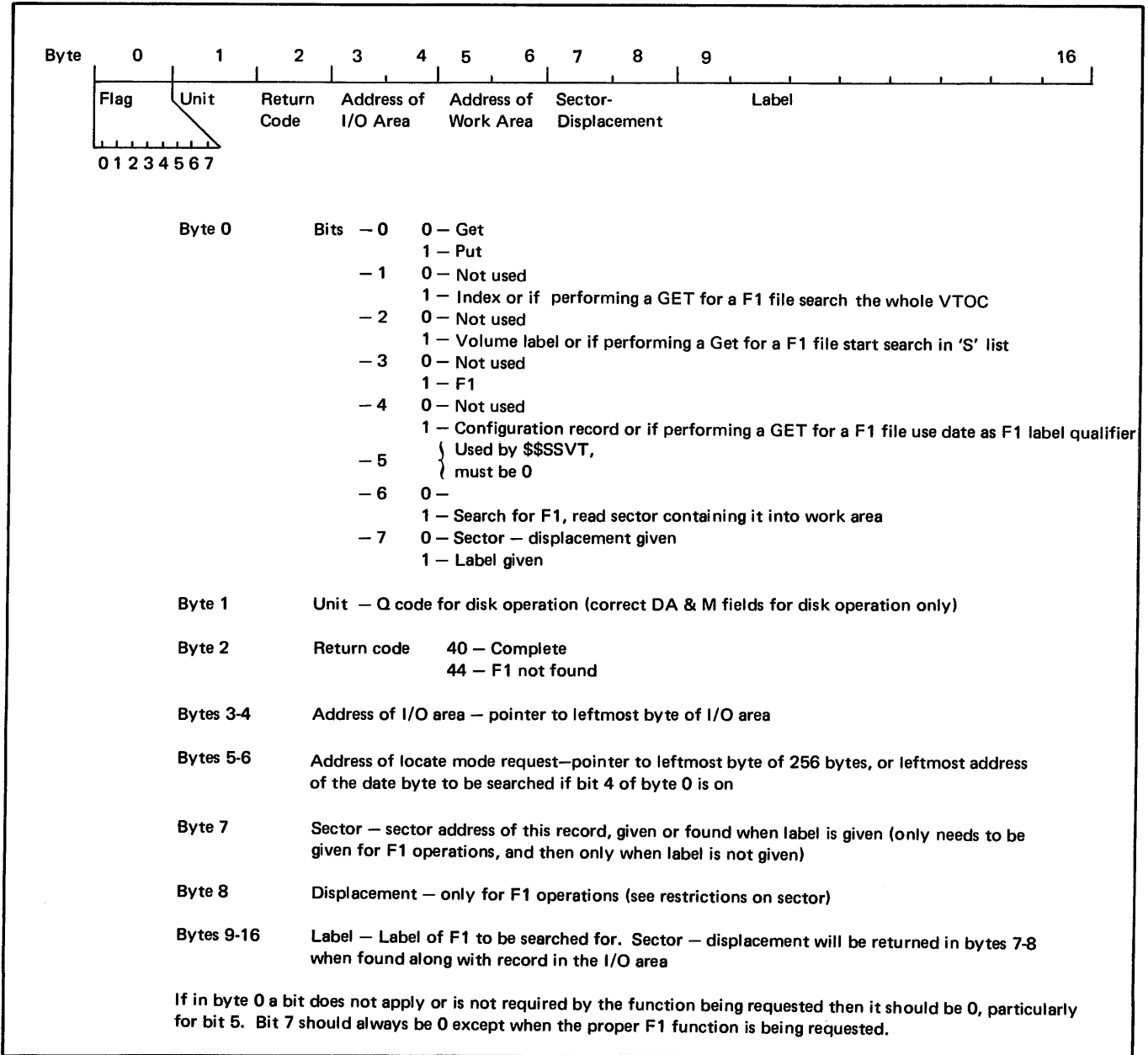


Figure 6-7. Parameter List for Volume Table of Contents Read/Write Routine

► Rollin—Phase One (\$\$STRI)

CHART: JW
FIGURE: 6-1
ENTRY POINT: \$\$STRI
FUNCTION: Restores the SWA from Roll-out disk area storage back to system disk area storage.
INPUT: None
OUTPUT: The SWA is restored.
EXIT:
– Normal: Rollin—Phase Two (\$\$STRN)
– Error: A terminal halt is issued upon encountering a permanent disk error

► Rollin—Phase Two (\$\$STRN)

CHART: JX
FIGURE: 6-1
ENTRY POINT: \$\$STRN
FUNCTION:
– Restores an interrupted RPG program.
– Restores all data areas associated with the interrupted RPG program.
– Determines if the Multi-Function Card Unit (MFCU) was in use at the time the RPG program was rolled out and repositions cards.
– Determines if the correct disk pack was remounted.
– Passes control to the restored RPG program.
INPUT: None
OUTPUT: The user's pre-empted program and associated data areas are loaded back into main storage.
EXIT:
– Normal: User's pre-empted RPG program
– Error: Supported Halt/Syslog routine

► Rollout—Phase One (\$\$STRO)

CHART: JY
FIGURE: 6-1
ENTRY POINT: \$\$STRO
FUNCTION: Waits for the completion of the I/O operations for the devices allocated to the program level.
INPUT: None
OUTPUT: The user's RPG program and necessary data areas are saved on disk.
EXIT:
– Normal: Rollout—Phase Two (\$\$STRU)
– Error: RPG calling program

► Rollout—Phase Two (\$\$STRU)

CHART: JZ
FIGURE: 6-1
ENTRY POINT: \$\$STRU
FUNCTION:
– Saves the SWA for the program level.
– Moves the user's RPG program from main storage onto disk.
INPUT: None
OUTPUT:
– The SWA for the program level is saved.
– The user's RPG program is stored on disk.
EXIT: Rollout—Phase Three (\$\$STRT)

► Rollout—Phase Three (\$\$STRT)

CHART: KA
FIGURE: 6-1
ENTRY POINT: \$\$STRT
FUNCTION: Stores and builds a disk location table containing entries for each of the following data areas.
1. Print chain image
2. Transient scheduler table
3. Printer page size
4. Scheduler/data management switches
5. Volume labels
6. Program level communication region
7. Pack IDs for volumes used by program
INPUT: None
OUTPUT: Disk location table entries with the following format:
1. Address of the data
2. Length of data
3. Main storage location from which the data was taken
EXIT:
– Normal: Terminator's Re-initialization routine (\$\$TMRI)
– Error: Supported Halt/Syslog routine

► Disk System System List—Print (\$\$SYP1)

CHART: KB

FIGURE: 6-8

ENTRY POINT: \$\$SYP1

FUNCTION: Allows the user to be printer independent by supporting the following functions:

1. Open
2. Close
3. Print, skip and/or space
4. Skip and/or space without print
5. Allocate printer

INPUT: Register 2 points to a 7-byte parameter list:

- Byte 1—Op code
- Byte 2—Skip N lines before printing
- Byte 3—Space N lines before printing
- Byte 4—Skip N lines after printing
- Byte 5—Space N lines after printing
- Bytes 6 and 7—Logical record address

OUTPUT: Printed Output

EXIT:

- Normal: Next sequential instruction+8 of caller
- Error: Next sequential instruction+4 of caller
- Overflow: Next sequential instruction of the caller

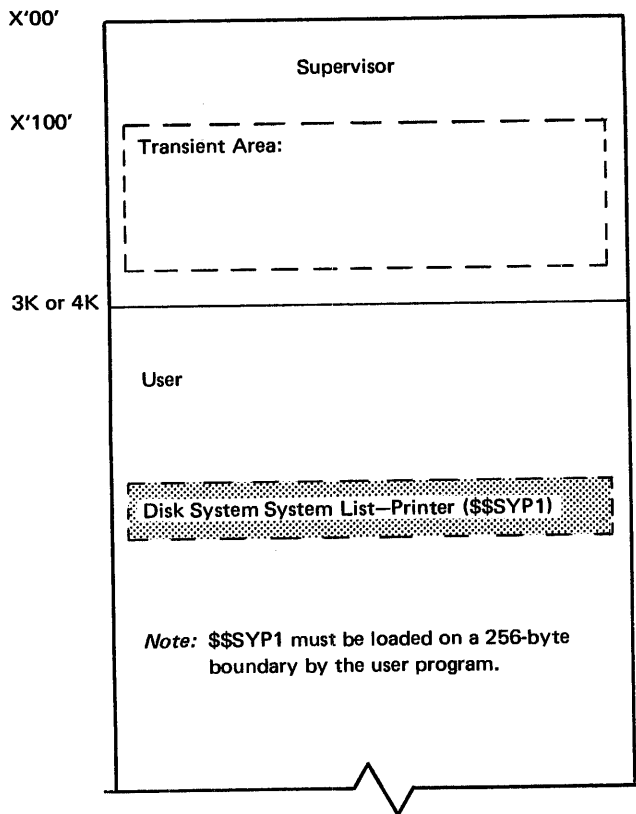


Figure 6-8. Main Storage Map for Disk System System List Routine for the Printer (\$\$SYP1)

► Model 6 System List—Print (\$\$SYP2)

CHART: KC

FIGURE: 6-9

ENTRY POINT: \$\$SYP2

FUNCTIONS:

- Derives the required halt/syslog parameter list from the print parameter list of the calling program.
- Calls halt/syslog.

INPUT: XR2 contains the address of the print parameter list of the calling program.

OUTPUT: Required parameter list is passed to halt/syslog.

EXIT: To next sequential instruction of calling program

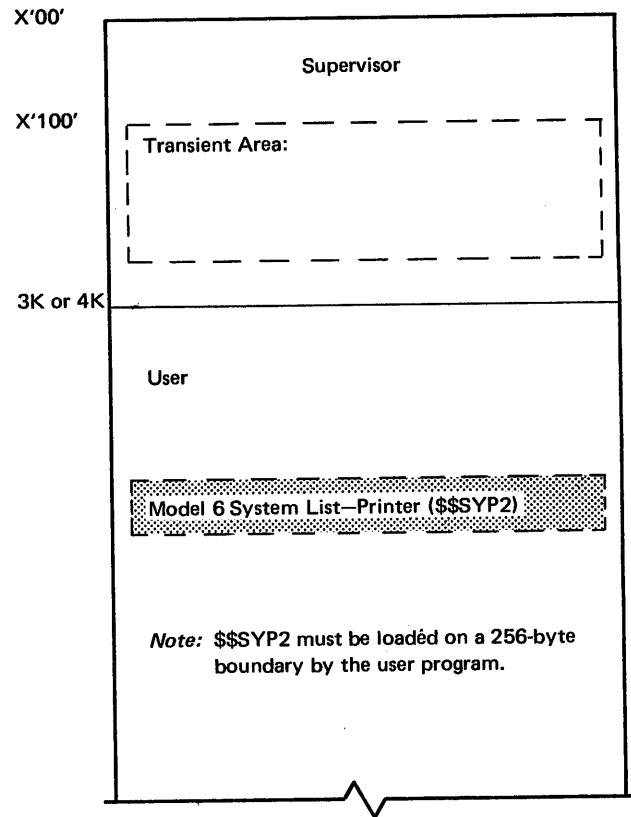


Figure 6-9. Main Storage Map for Model 6 System List Routine for the Printer (\$\$SYP2)

► **Disk System System List–Print/Punch (\$\$SPC1)**

CHART: KD

FIGURE: 6-10

ENTRY POINT: \$\$SPC1

FUNCTION: Determines if request was for the MFCU:

1. Punch only
2. Punch/Print 96 columns
3. Punch 96 columns and Print 128 columns

INPUT: The following 8-byte parameter list is passed via Register 2:

Byte 1–Operation code

X'00' = CLOSE

X'10' = OPEN

X'40' = PUNCH CARD

X'60' = PRINT/PUNCH CARD

Byte 2–Print Length

X'60' = 96 columns

X'80' = 128 columns

Bytes 3 and 4–Address of 10-byte permanent save area

Bytes 5 and 6–Address of 96-byte punch buffer

Bytes 7 and 8–Address of 96 or 128-byte print buffer

OUTPUT: One card is produced, by the MFCU, per request.

EXIT:

- Normal: Calling routine
- Error: End-of-Job transient (\$\$SPEJ) routine for an immediate cancel

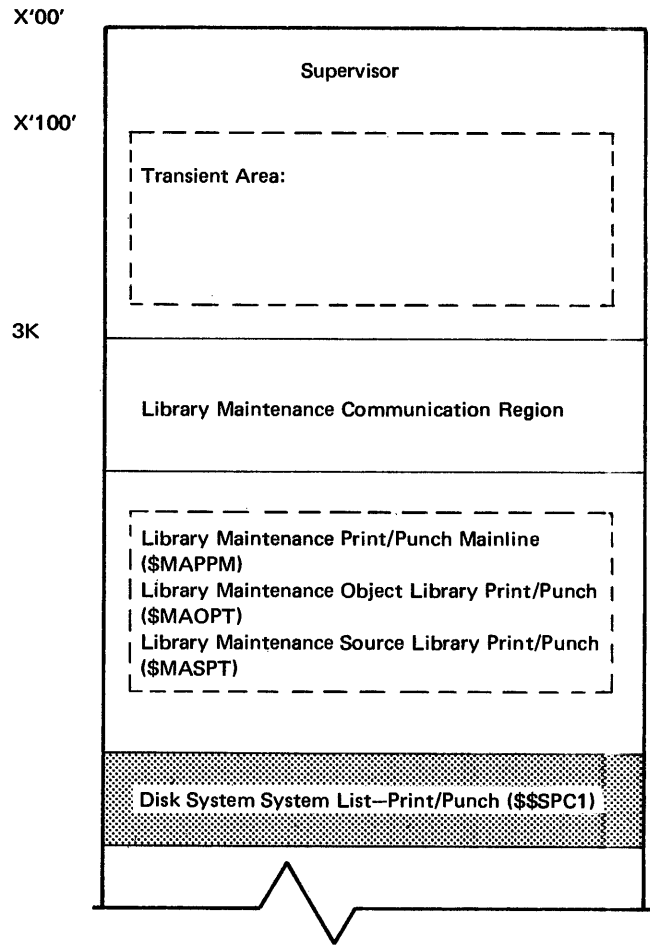


Figure 6-10. Main Storage Map for Disk System System List–Print/Punch Routine (\$\$SPC1)

► Model 6 System List—Punch Routine (\$\$SPC2)

CHART: KE

FIGURE: 6-11

ENTRY POINT: \$\$SPC2

FUNCTIONS:

- Punches information into the 96 column card.
- Prints information on the 96 column card.

INPUT: The following 8-byte parameter list is passed via XR2:

Byte 1—Operation code

X'00' = CLOSE

X'10' = OPEN

X'40' = PUNCH ON CARD

Byte 2—Print length

X'60' for 96 character print

X'80' for 128 character print (4 lines)

Bytes 3 and 4—address of 15-byte permanent save area used for error recovery

Bytes 5 and 6—address of 96-byte buffer which will contain information to be punched

Bytes 7 and 8—address of 96-byte print buffer

OUTPUT: One punched and printed card for each request
EXIT:

- Normal: Next sequential instruction of calling program
- Error: Halt/syslog and immediate cancel

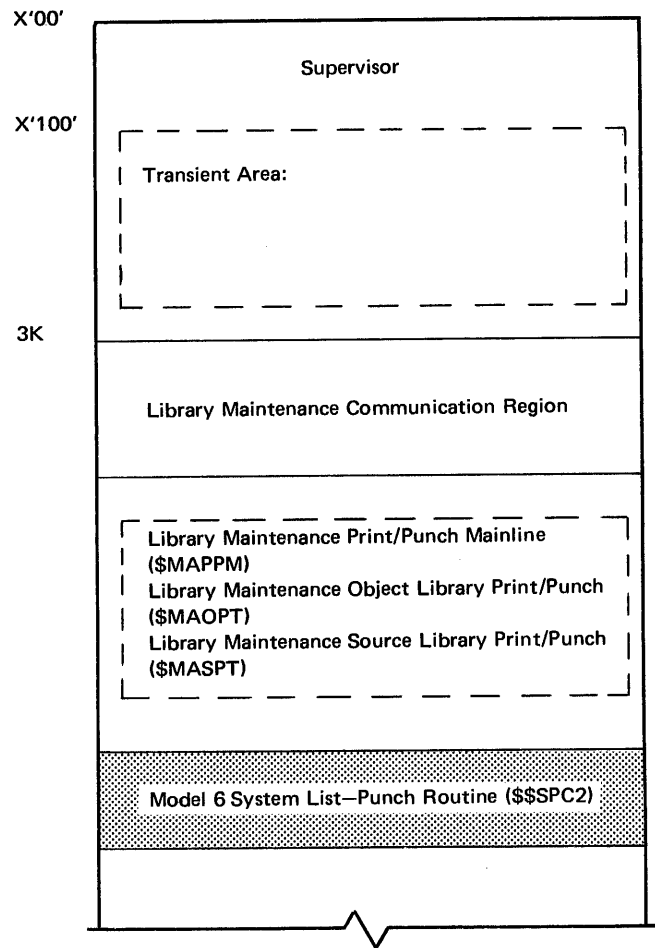


Figure 6-11. Main Storage Map for System List—Punch Routine (\$\$SPC2)

► **Source Library Get (\$\$SYSG)**

CHART: KF

FIGURE: 6-1

ENTRY POINT: \$\$SYSG

FUNCTION: Finds a specified member in the Source Library and/or passes a 96-byte record, from the member, back to the caller.

INPUT: The get parameter list is passed from the caller to this routine. The parameter list has the following two formats:

First call F/Q/T/NNNNNN/AA
All other calls F/Q/XX/YY/ZZZ/AA

F → F Is the function byte

<i>Bit</i>	<i>Request</i>
------------	----------------

0	Find
1	Get first rec
2	Get next rec
3	Not used

<i>Bit</i>	<i>Reply</i>
------------	--------------

4	Not used
5	Unused
6	No find or no library
7	End of file

Q → Q Is Q byte of pack

R1 = A1

F1 = A9

R2 = B1

F2 = B9

T → T Is type of member

S is source

P is procedure

N → NNNNNN Is 6 character name

X → XX Is cyl/sect of beginning of member

Y → YY Is cyl/sect of end of member

Z → ZZZ Is cyl/sect/disp of next record to be processed

A → AAAA Is 2 byte addr of first byte of 96 byte record

OUTPUT:

– Updated get parameter list

– 96-byte record

EXIT: Transient Area Scheduler (NENTRY)

► **Find Transient (\$\$SPFN)**

CHART: KG

FIGURES: 6-1

ENTRY POINT: \$\$SPFN

FUNCTION:

– Locates a specific overlay or program in the object library.

Note: If invoked by the Loader, the requested program is specified by either fetch or load request operand.

– Passes loading information, contained in the directory entry, to the caller.

– Issues a halt if the entry is not found.

INPUT: Operand specifying the requested program or overlay

OUTPUT: Information contained in the program's directory entry needed for loading is passed to the calling routine.

EXIT:

– Normal: Calling routine

– Error:

1. A halt of 'b4' on the Disk System, '14' on Model 6, is issued and displayed on the halt lights.

2. If log is specified on, the module name is printed on the Syslog device.

► Allocate Initiator (\$\$STAI)

CHART: KH

FIGURES: 6-12, 6-13

ENTRY POINT: \$\$STAI

FUNCTION:

- Allocates devices to the calling program.
- Allocates disk files for use by the calling program.
- Copies the last 4096 bytes (4K) of the user's program area from main storage into the SWA on the system disk.
- Compresses specified DTFs into the last 1024 bytes (1K) of the user's main storage area for \$\$INA1. See Figure 6-13 for the format of the compressed DTFs.

INPUT: Register 2 contains the starting address of the chained DTFs. The DTFs contain the chain pointer B.

OUTPUT:

- The last 4K of the user's main storage is copied into the SWA.
- The user's disk DTFs are compressed and loaded into the last 1K of the user's main storage.

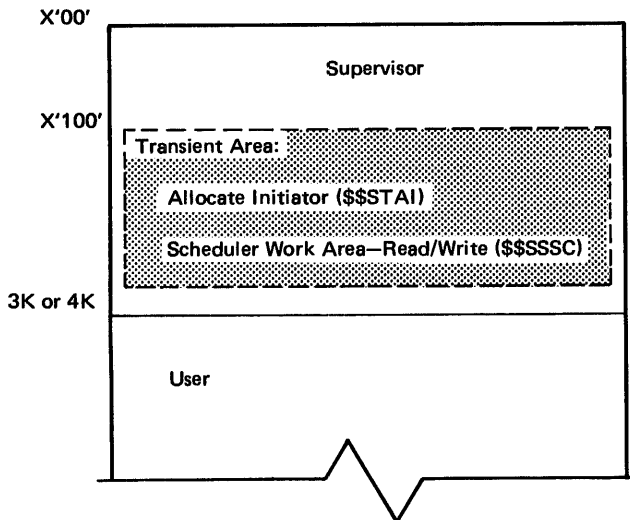


Figure 6-12. Main Storage Map for Allocate Initiator Transient (SSSTAI)

EXIT:

- Normal:
 1. When all non-disk DTFs are given, control is returned to the caller.
 2. When disk DTFs are given, control is passed to the Initiator Allocate—Phase One Routine (\$\$INA1).
- Error: Supported Halt/Syslog routine

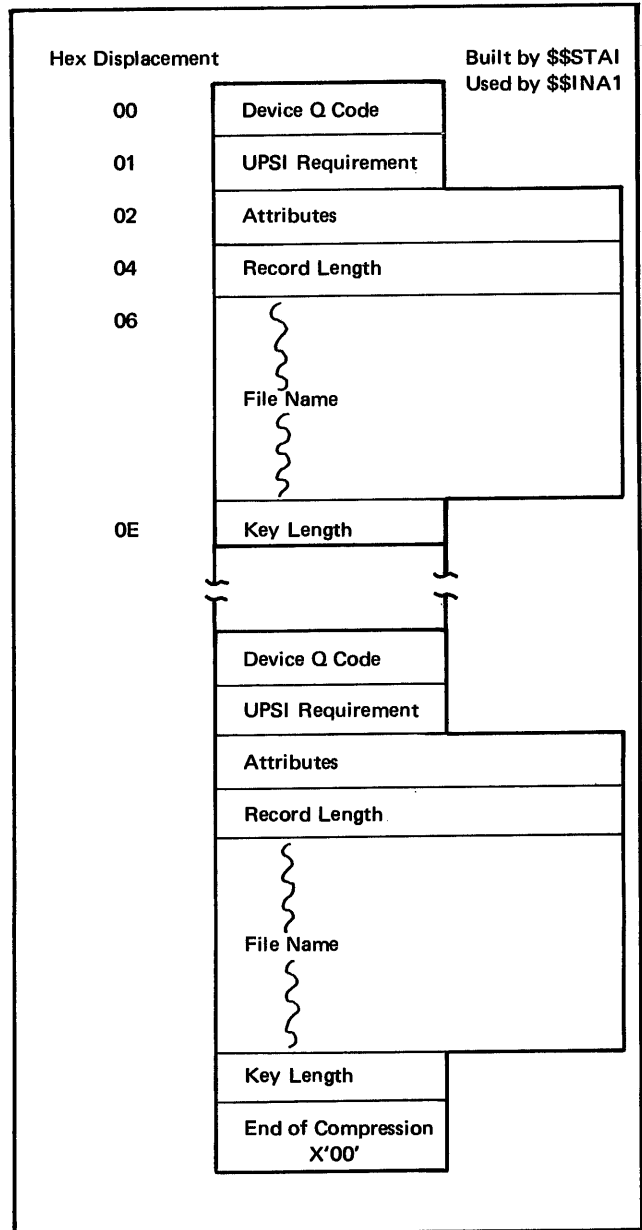


Figure 6-13. Compressed DTF Format—Built by the Allocate Initiator Transient

► **LOAD * Mainline (\$\$SYLA)**

CHART: KI

FIGURES: 6-14

ENTRY POINT: \$\$SYLA

FUNCTION: Builds a temporary object library entry containing the following:

1. Directory entry
2. Object program
3. Volume label

INPUT:

- Header card
- Object program (R. module) built by the Linkage Editor

OUTPUT: The object program and the directory entry are written onto the system disk pack.

EXIT:

- Normal: Initiator Program Setup—Phase One (\$\$INPS) routine
- Error: Supported Halt/Syslog routine

► **End of Job Transient (\$\$SPEJ)**

CHART: None

FIGURES: 6-1

ENTRY POINT: START

FUNCTION:

- Reset the partition to 4K.
- Sets the relocation factor and the overlay factor to zero.
- Closes all chained DTFs (for control cancel only).
- Reset console I/O.
- Clears the transient queue.
- Turns off library maintenance control indicators.
- Calls the Terminator EOJ—Phase One routine (\$\$TMEO).

INPUT: Register 1 points to the program level communication area.

OUTPUT: Terminates the calling routine

EXIT:

- Normal: Terminator End of Job—Phase One (\$\$TMEO)
- Error: Display halts 'HE' followed by 'OA' on the Disk System, halts 'BCD12345' followed by 'ABD2345' on Model 6, if the IPL contained an error before EOJ was called.

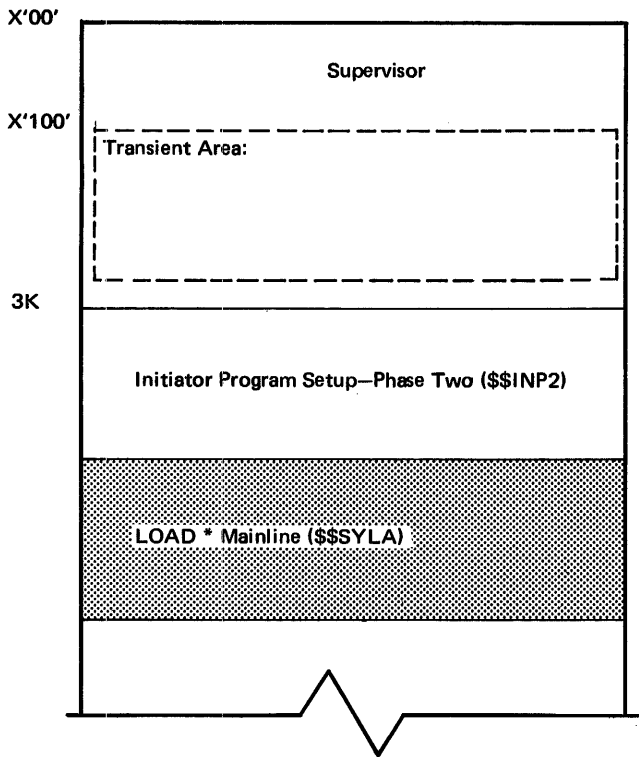


Figure 6.14. Main Storage Map for LOAD*Mainline (SSSYLA)

► End of Job—Disk Status (\$\$TMST)

CHART: KJ

FIGURES: 6-1, 6-15

ENTRY POINT: \$\$TMST

FUNCTION:

- Determines the mask to be used for the input Q-byte.
- Serves as a mediator between the Disk IOS Error Logging routines, used to log the SIO table, and the Terminator End of Job phases one and two (\$\$TME0, \$\$TMEJ).

INPUT: A Q-byte is passed from the Terminator End of Job phases (one or two).

OUTPUT: A 2-byte parameter list is generated describing the Q-byte. This parameter list is passed to the correct IOS Error Logging routine. See Figure 6-15 for the format of this parameter list.

EXIT: IOS Error Logging routine (\$\$DLOG)

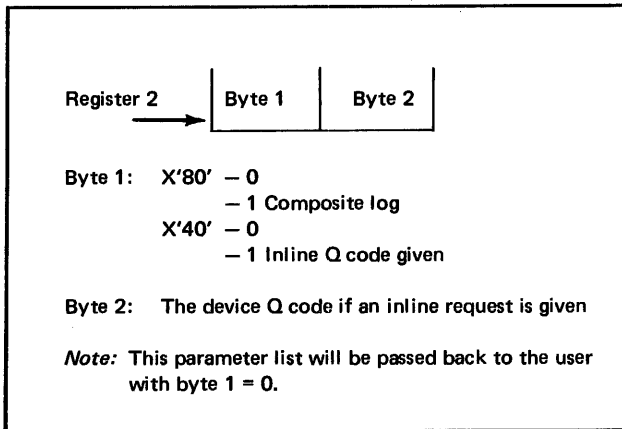


Figure 6-15. Parameter List for End of Job—Disk Status Routine

► Copy Main Storage to Source Library (\$\$SYSP)

CHART: KK

FIGURES: 6-16

ENTRY POINT: \$\$SYSP

FUNCTION:

- Reads 96-byte records from main storage.
- Compresses the 96-byte records and loads them into the source library.
- Updates the volume label and library directory.

INPUT: Source statements or procedures from main storage and a source library on a specified disk pack

OUTPUT: The input source statements or procedures are compressed and loaded into the source library.

EXIT:

- Normal: Calling routine
- Error: A halt is issued if a permanent disk error is encountered

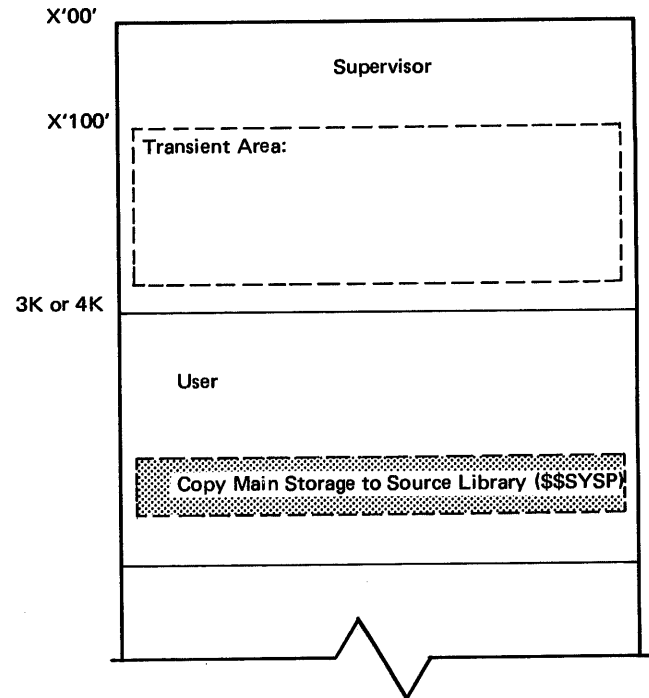


Figure 6-16. Main Storage Map for Copy Main Storage to Source Library Routine (\$\$SYSP)

► **Resource Allocation—Phase One (\$\$STR1)**

CHART: KL

FIGURE: 6-1

ENTRY POINT: \$\$STR1

FUNCTION:

- Reads temporary configuration records from the SWA.
- Calls Resource Allocation—Phase Two (\$\$STR2) to allocate the requested devices.
- Restores the updated configuration records to the SWA.

INPUT: Register 2 contains the address of the first DTF in the DTF chain. Each DTF contains, in the leftmost byte, the address of the device.

OUTPUT: Updated temporary configuration record

EXIT:

- Normal: Calling routine
 - Error: A halt condition results
- Note:* The user must not have any DTFs in the last 1K of the program level.

► **Resource Allocation—Phase Two (\$\$STR2)**

CHART: KM, Disk System
KN, Model 6

FIGURE: 6-1

ENTRY POINT: \$\$STR2

FUNCTION: Allocates non-disk devices for a program level.

INPUT: Register 2 contains the address of the 1K work area built in \$\$STR1.

OUTPUT: Updated temporary configuration records in the SWA.

EXIT:

- Normal: Resource Allocation—Phase One (\$\$STR1)
 - Error: Supported Halt/Syslog routine
- Note:* The user must not have any DTFs in the last 1K of the program level.

► **Transient Resolver (\$\$OXRF)**

CHART: KO

FIGURE: 6-17

ENTRY POINT: \$\$OXRF

FUNCTION: Resolves absolute disk addresses among certain system transients to facilitate the passing of control from one transient to another.

INPUT: None

OUTPUT: Resolved caller transient addresses

EXIT:

- Normal: Calling routine
- Error: End-of-Job transient (\$\$SPEJ)

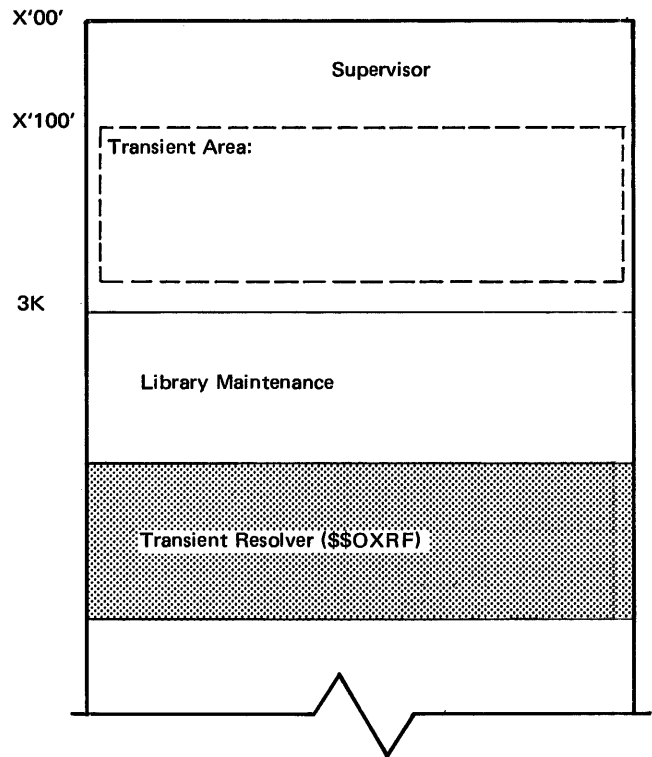


Figure 6-17. Main Storage Map for Transient Resolver (SSOXRF)

► Allocate Terminator (\$\$INAT)

CHART: KP

FIGURE: 6-18

ENTRY POINT: \$\$INAT

FUNCTION: Sets up the user's main storage to reflect the settings before the allocation routines were called.

INPUT:

- The completion code found at label NPSCH8 located within the program communication region (special requests only)
- The user's main storage (that was saved) is located at initiator table plus three sectors.

OUTPUT:

- The completion code is returned in the caller's first DTF (special requests only).
- The user's saved main storage settings are reloaded into main storage.

EXIT: Calling routine

► System Queue Routine (\$\$STNQ)

CHART: None

FIGURE: 6-1

ENTRY POINT: \$\$STNQ

FUNCTION: This routine is called if a transient must be removed from the transient area X'100', but the transient to be removed is interlocked with the Scheduler within a DPF system. This routine provides all of the necessary linkages to remove the transient via the Resident Halt Routine (HPLNUC).

INPUT: None

OUTPUT:

- A request for an APL is sent to the Resident Halt Routine (HPLNUC).
- The first seven bytes of the transient storage area (NCSTOR), located within the system communication region, are saved

EXIT: Resident Halt Routine (HPLNUC)

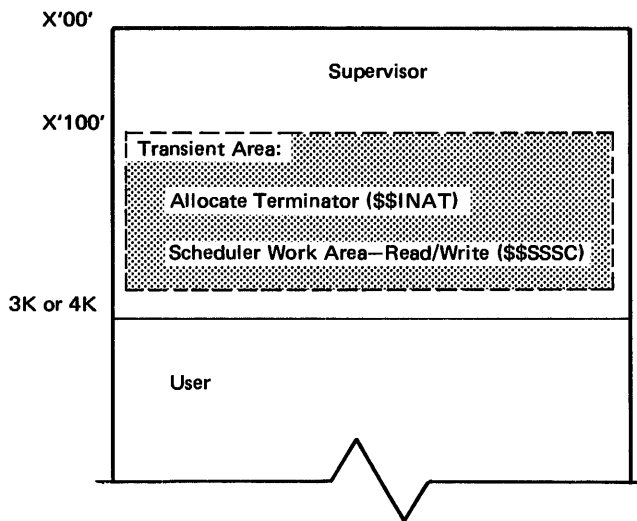


Figure 6-18. Main Storage Map for Allocate Terminator Transient (SSINAT)

► **Halt/Syslog Parameter Build Routine (\$\$STHB)**

CHART: KQ

FIGURE: 6-19

ENTRY POINT: \$\$STHB

FUNCTION:

- Builds the appropriate parameter list to call the supported Halt/Syslog routine.
- Loads the parameter list into the Reader/Interpreter work area (RDIWA).

INPUT: The Reader/Interpreter work area contains the error number and a 2-byte module identifier indicating the module that caused the error.

OUTPUT: A 5 or 7-byte parameter list is built in the Reader/Interpreter work area (RDIWA).

EXIT: Calling routine

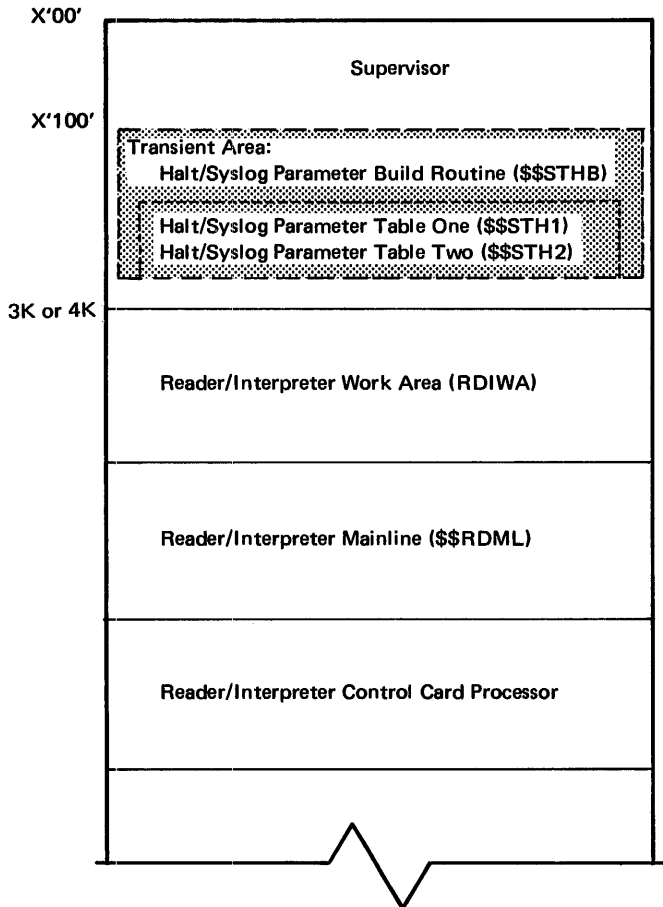


Figure 6-19. Main Storage Map for Halt/Syslog Parameter Build Routine (\$\$STHB)

► **Keyword Syntax Scan Routine (\$\$RDS1)**

CHART: KR

FIGURE: 6-20

ENTRY POINT: \$\$RDS1

FUNCTION:

- Scans a specified area containing keyword parameters.
- Checks the validity of the parameters.
- Builds a table of corresponding arguments.
- Returns a completion status code to the caller.

INPUT: An 8-byte parameter list (passed from the caller) containing the following:

1. Address of the keyword table
2. Address of the argument table that is to be built
3. Addresses (start and end) of the area to be scanned for the keywords

OUTPUT: A return code is passed back to the caller indicating one of the following conditions:

1. Syntax error
2. Invalid keyword
3. Continuation card found
4. Invalid parameters for the area to be scanned
5. Successful completion

EXIT: Calling routine

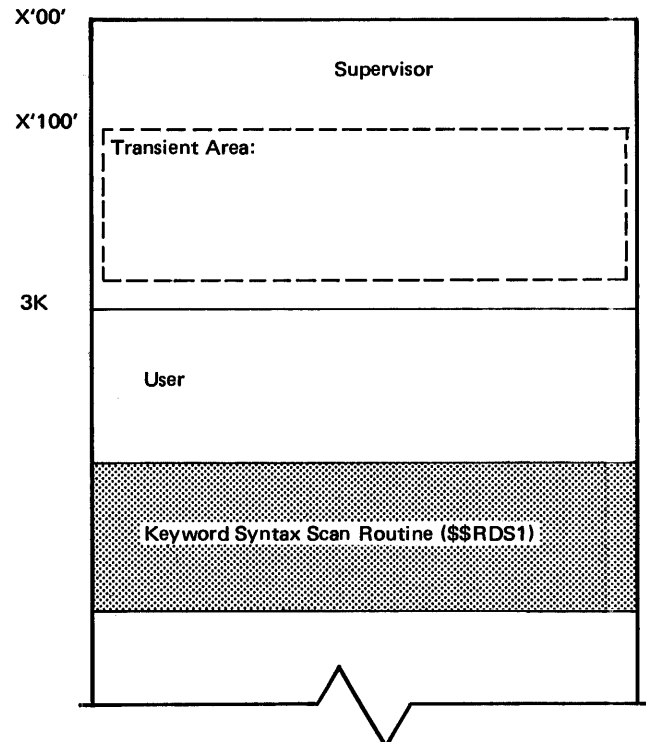


Figure 6-20. Main Storage Map for Keyword Syntax Scan Routine (\$\$RDS1)

► **Transient Resolver No-Op Routine (\$\$ODNP)**

CHART: None

FIGURE: 6-1

ENTRY POINT: DMODNP

FUNCTION:

- The Transient Resolver No-op Routine (\$\$ODNP) issues a halt of "42" with one option (immediate cancel).
- This transient (\$\$ODNP) is loaded into the transient area, by the calling system transient, after the following sequence of events:
 1. A system transient requests that another system transient be loaded into main storage.
 2. The Transient Resolver (\$\$OXRF) cannot locate the requested system transient.
 3. The Transient Resolver (\$\$OXRF) returns the address of the Transient Resolver No-op Routine (\$\$ODNP) to the calling system Transient.

INPUT: None

OUTPUT: None

EXIT: Supported Halt/Syslog transient

► **HIKEY Transient (\$\$STF7)**

CHART: KS

FIGURE: 6-1, 6-21

ENTRY POINT: \$\$STF7

FUNCTION: Performs the reads and writes of format sevens in the SWA and VTOC.

INPUT: Register 2 points to a 6-byte parameter list. The format of the parameter list is shown in Figure 6-21.

OUTPUT: The format seven is read or written in the SWA or VTOC.

EXIT:

- Normal: Next sequential instruction of caller
- Error: Issue HE halt for permanent disk error and then call EOJ transient (\$\$SPEJ)

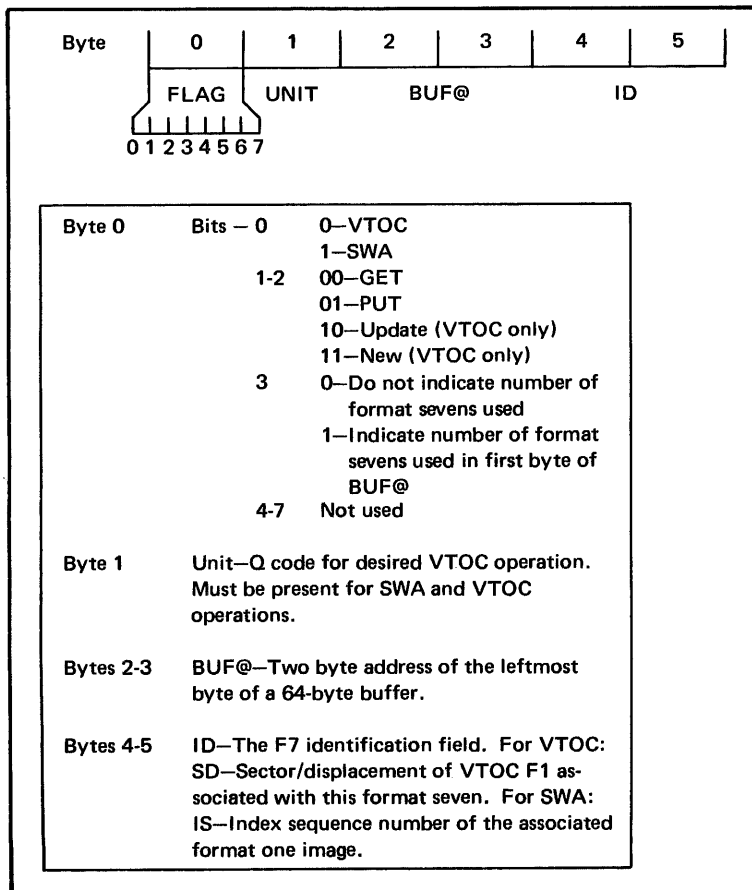


Figure 6-21. Parameter List for HIKEY Transient (\$\$STF7)

► **HIKEY Parameter Scan Routine (\$\$RDHK)**

CHART: KT

FIGURE: 6-1

FUNCTION: Scan a specified area containing the HIKEY parameters.

INPUT: Register 2 contains the address of the starting position of the scan.

OUTPUT:

- Parameters are encoded and placed in the ENCODE field located in \$\$RDML.
- Sets up for halt if syntax errors are found.

EXIT:

- Normal: Reader/Interpreter Mainline (\$\$RDML)
- Error: Supported Halt/Syslog routine

► **Model 6 Logical Put Routine (\$\$RBLP)**

CHART: KU

FIGURE: 6-1

ENTRY POINT: \$\$RBLP

FUNCTION:

- Compresses records.
- Writes compressed records to disk.

INPUT: XR2 points to an 11-byte parameter list. The parameter list has the following format:

F/Q/XX/YY/ZZ/AA

F F is the function byte:

Value Request

- | | |
|-------|---|
| X'C0' | Put first record in buffer, but do not write unless sector overflows. |
| X'A0' | Put next record in buffer, but do not write unless sector overflows. |
| X'80' | Put buffer to disk. |
| X'40' | Put first record to disk. |
| X'20' | Put next record to disk. |

Value Reply

- | | |
|-------|-----------------------------|
| X'08' | Permanent disk error |
| X'01' | End of disk space specified |

Q Q byte of pack

Value Pack

- | | |
|-------|----|
| X'A1' | R1 |
| X'A9' | F1 |
| X'B1' | R2 |
| X'B9' | F2 |

X XX contains the C/S of beginning PUT.

Y YY contains the C/S of ending PUT:

Z ZZZ contains the C/S/Displacement of the next record to be processed.

A AA contains the address of the first byte of the 96-byte record.

OUTPUT:

- Compressed record written to disk
- Updated parameter list

EXIT: To the transient area scheduler

► Model 6 Syntax Checker (\$\$RBSX)

CHART: KV
 FIGURE: 6-22
 ENTRY POINT: \$\$RBSX
 FUNCTION: Syntax checks data in a 96-byte buffer for Load, Build, and Call operations.
 INPUT: XR2 contains the address of the required parameter list.

The 7-byte parameter list has the following format:

O/F/SS/EE/L

O Operation byte X'80' = BUILD
 X'40' = LOAD
 X'20' = CALL

F Function Byte

Value Request

X'80' NAME-6 character
 ● LOAD
 ● BUILD
 X'90' NAME-6 character
 ● PACK
 X'A0' NAME-8 character
 ● FILE
 ● LABEL
 X'B0' CREATION DATE
 X'E0' SWITCH
 X'60' LOCATION
 X'40' UNIT
 X'30' RECORDS
 X'20' TRACKS
 X'10' RETAIN

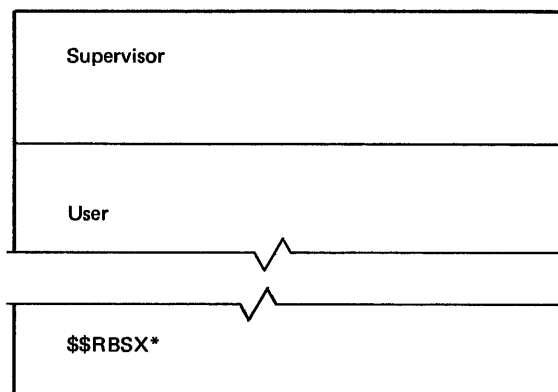
Value Reply

X'0A' END MVF
 X'09' START MVF
 X'08' OTHER
 X'04' NO RESPONSE
 X'02' INVALID RESPONSE
 X'01' NULL RESPONSE

S SS is a 2-byte address which identifies the first byte of a 96-byte buffer.
 E EE is a 2-byte address which identifies the last byte of the response returned to the caller.
 L L is a byte which contains the length of the response minus 1, and is returned to the caller.

OUTPUT: Updated parameter list
 EXIT: To next sequential instruction of caller

OK



*Syntax Checker (\$\$RBSX) is usually loaded at the ending address of the calling program. Examine the calling program to determine the exact loading address.

Figure 6-22. Main Storage Map for Model 6 Syntax Checker (\$\$RBSX)

► **Model 6 Syntax Checker Interface (\$\$RBSI)**

CHART: KW

FIGURE: 6-1

ENTRY POINT: \$\$RBSI

FUNCTION: This routine functions as an interface between Model 6 Syntax Checker (\$\$RBSX) and the calling program when a series of subparameters is to be checked.

INPUT: XR2 points to the parameter list used by \$\$RBSX.

OUTPUT: These codes are returned to the caller:

X'0A' – end of multivolume received

X'09' – multivolume started, more responses needed

X'02' – error condition exists

EXIT: To the next sequential instruction of caller

► **Model 6 Library Interaction Routine (\$\$RBLI)**

CHART: KX

FIGURE: 6-1

ENTRY POINT: \$\$RBLI

FUNCTION: This routine scans each 96-byte record taken from a procedure in the source library for LOAD, or any verb, or a continuation statement.

INPUT: XR2 points to a 3-byte parameter list containing a flag byte and the address of the 96-byte record.

The flag byte codes are:

X'02' – scan for the verb LOAD

X'01' – scan for any verb

X'00' – scan for a continuation statement

OUTPUT:

– A code returned to the flag byte:

X'80' – P code of verb found

X'40' – Non-Model 6 verb found

X'10' – Record was extraneous or in error

X'08' – RUN found

X'04' – Continuation record found

– Register 1 points one byte beyond the verb, comment indicator (*), or continuation indicator (//).

EXIT: To the next sequential instruction of the caller

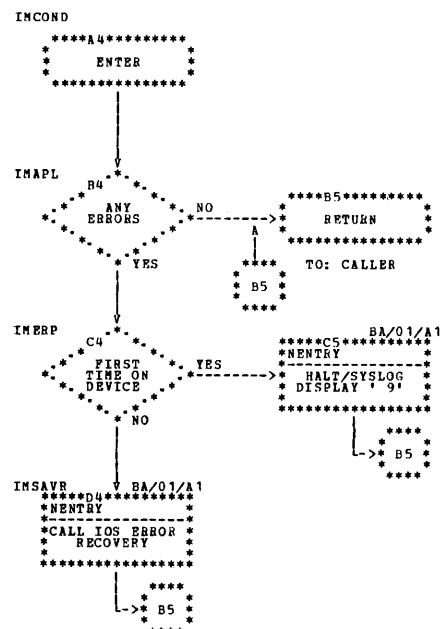
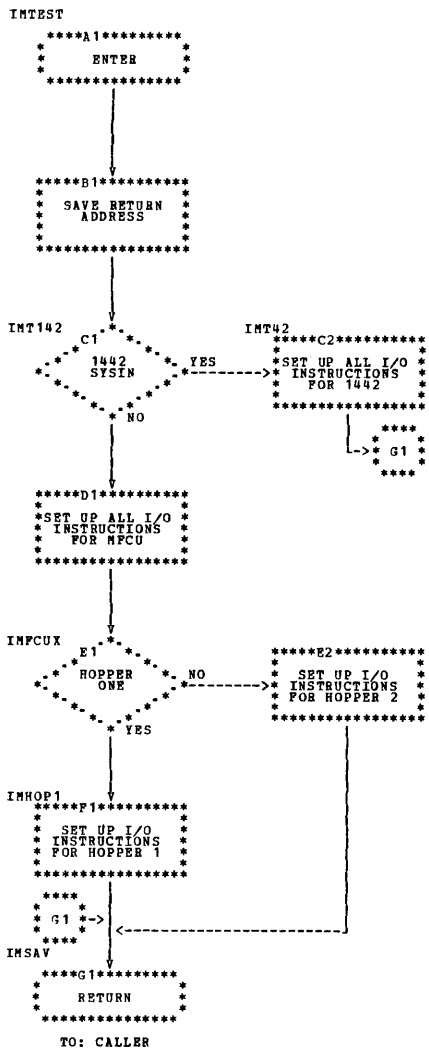


Chart JB (Part 2 of 2). System Input Device-1442/MFCU (\$\$STIM)

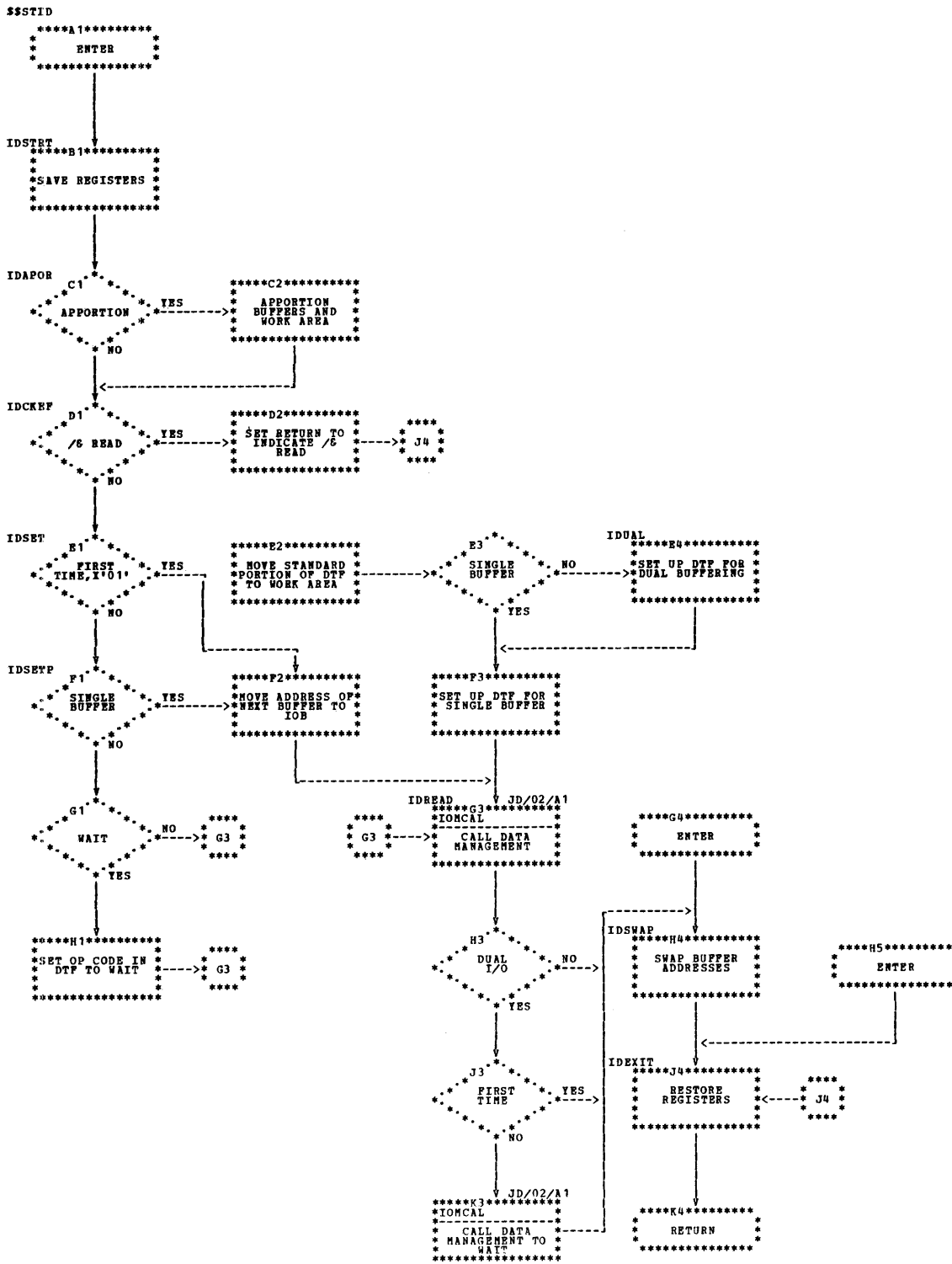


Chart JD (Part 1 of 2). Model 6 System Input Device-Data Recorder (\$\$STID)

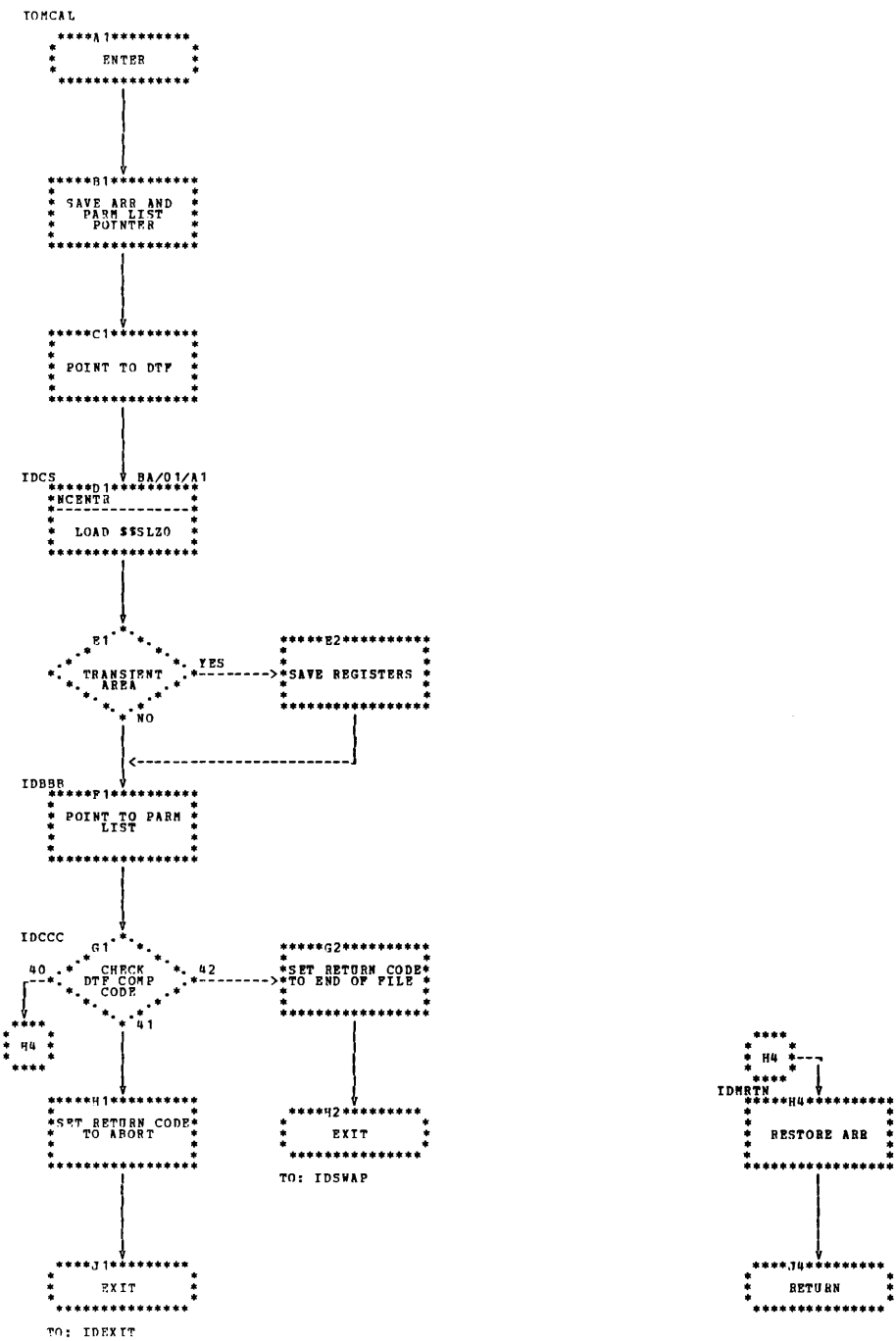


Chart JD (Part 2 of 2). Model 6 System Input Device--Data Recorder (\$\$STID)

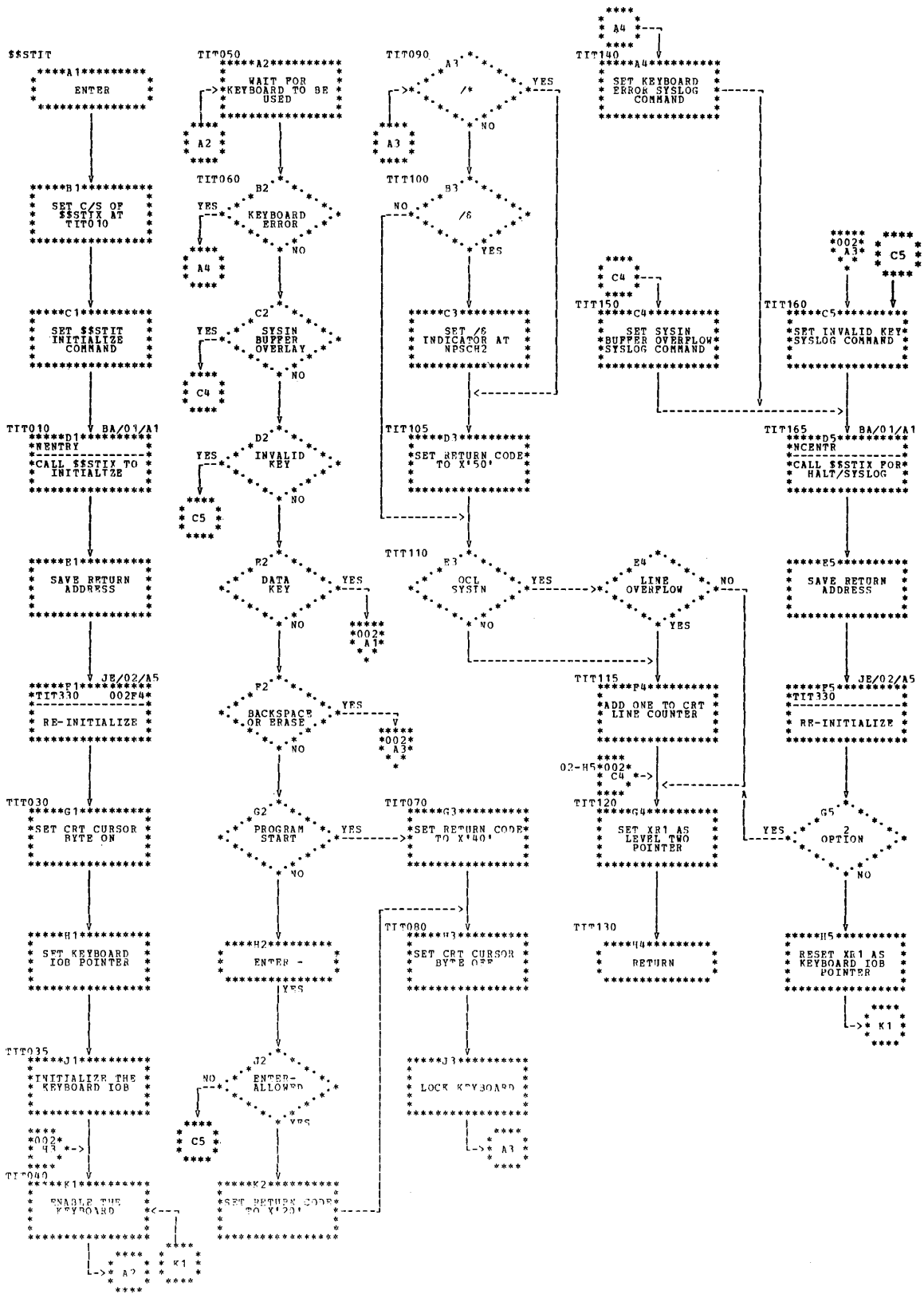


Chart JE (Part 1 of 4). Model 6 System Input Device-CRT (\$\$STIT, \$\$STIX)

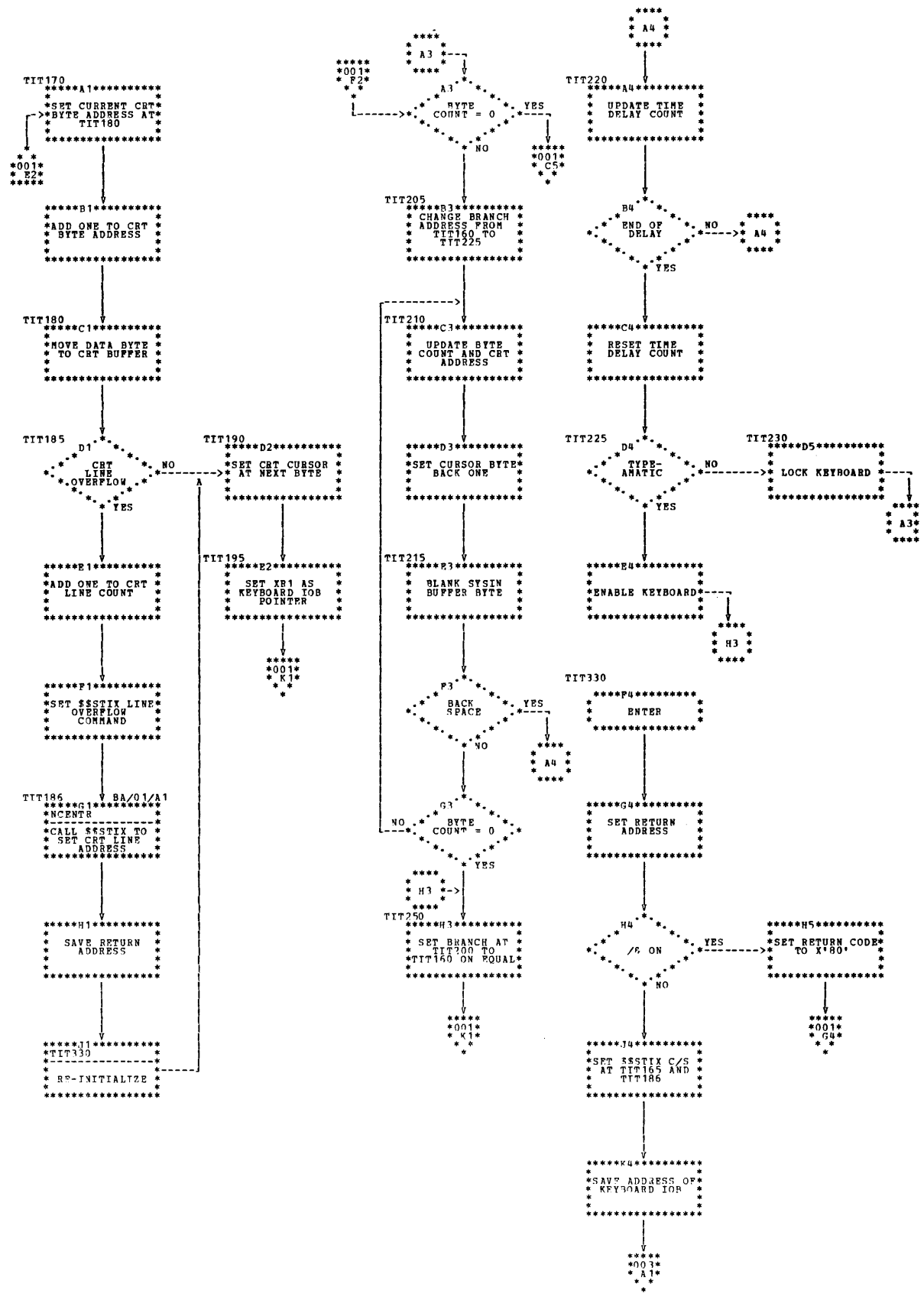
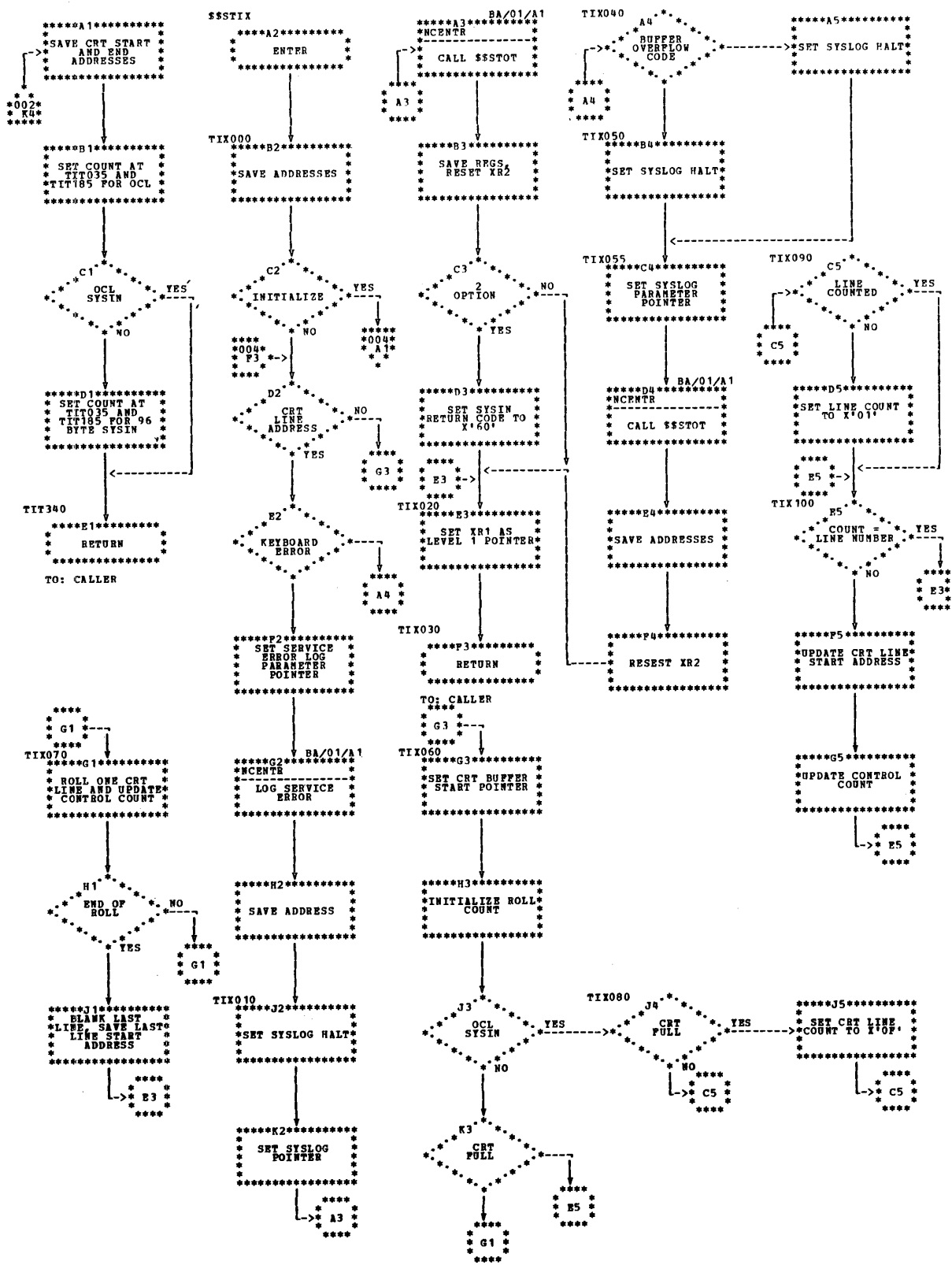


Chart JE (Part 2 of 4). Model 6 System Input Device-CRT (\$\$STIT, \$\$STITX)



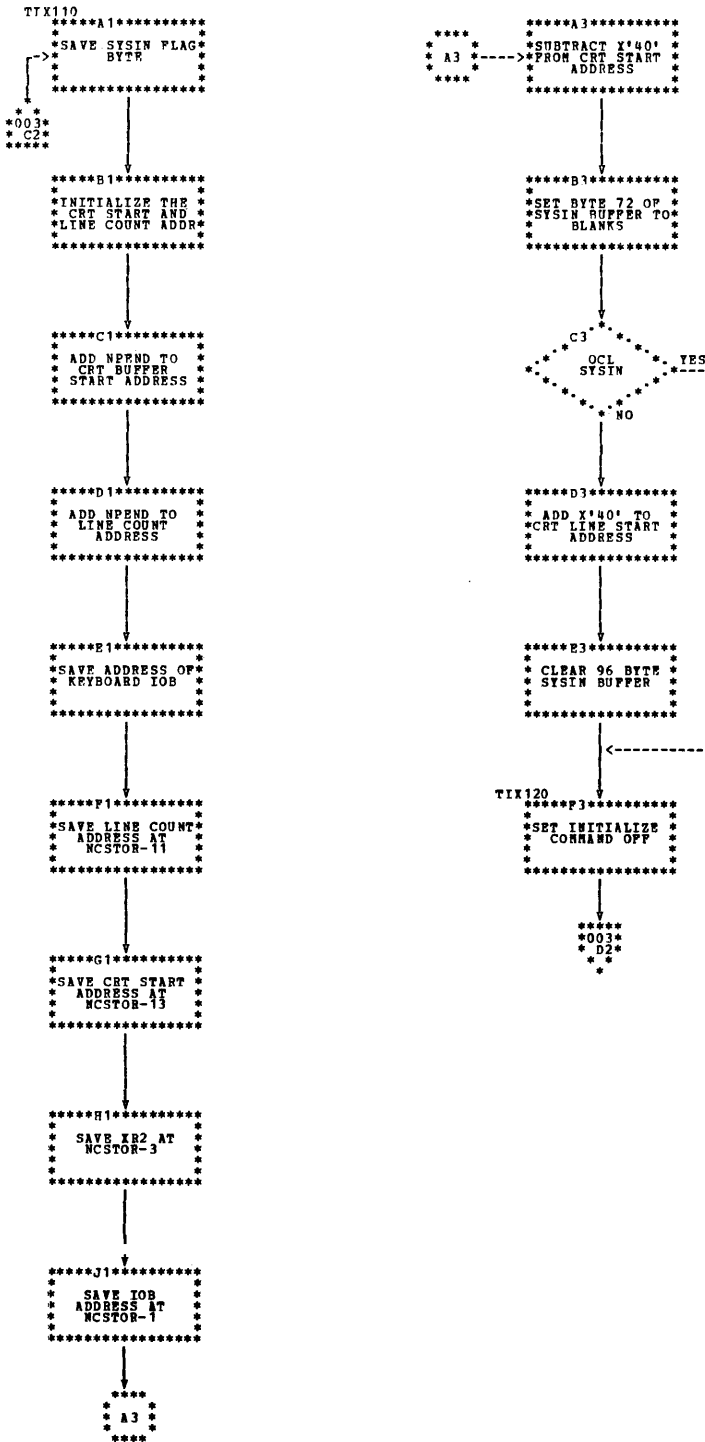


Chart JE (Part 4 of 4). Model 6 System Input Device—CRT (\$\$STIT, \$\$STIX)

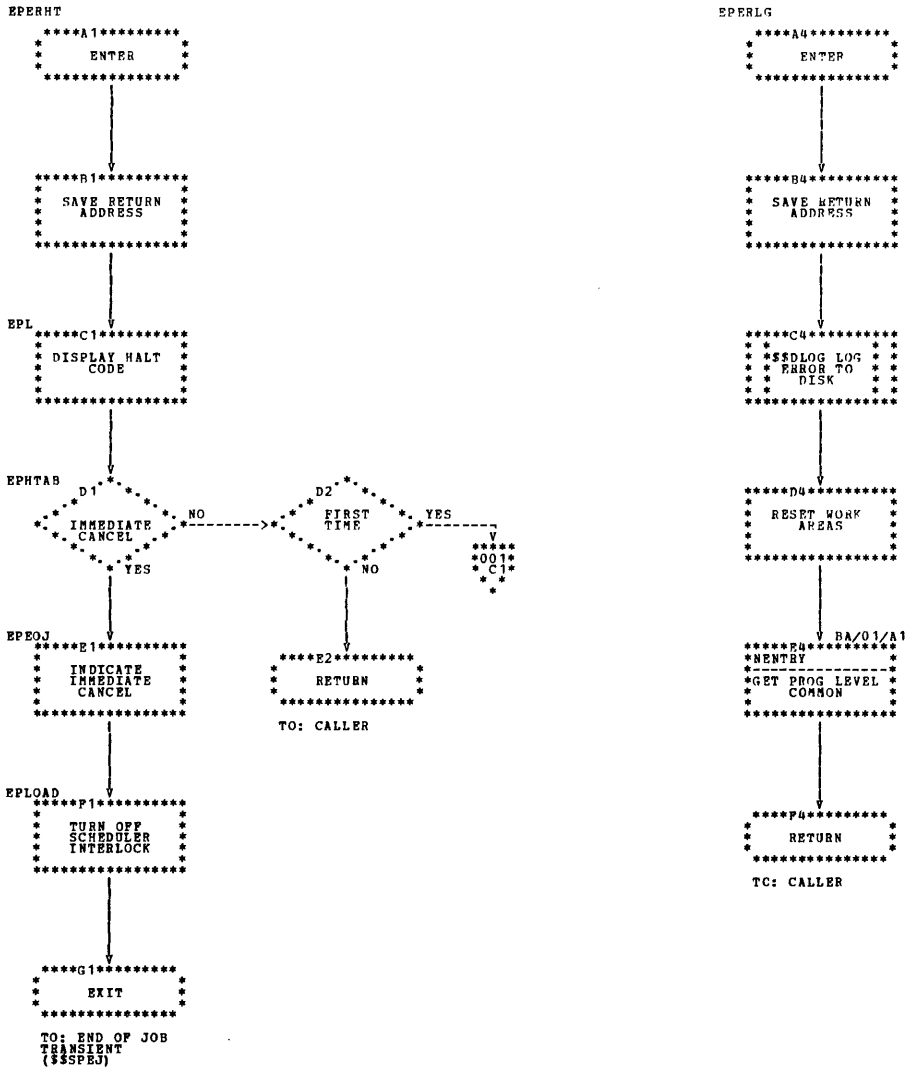


Chart JF (Part 2 of 2). Printer Error Recovery Routine (\$\$STEP)

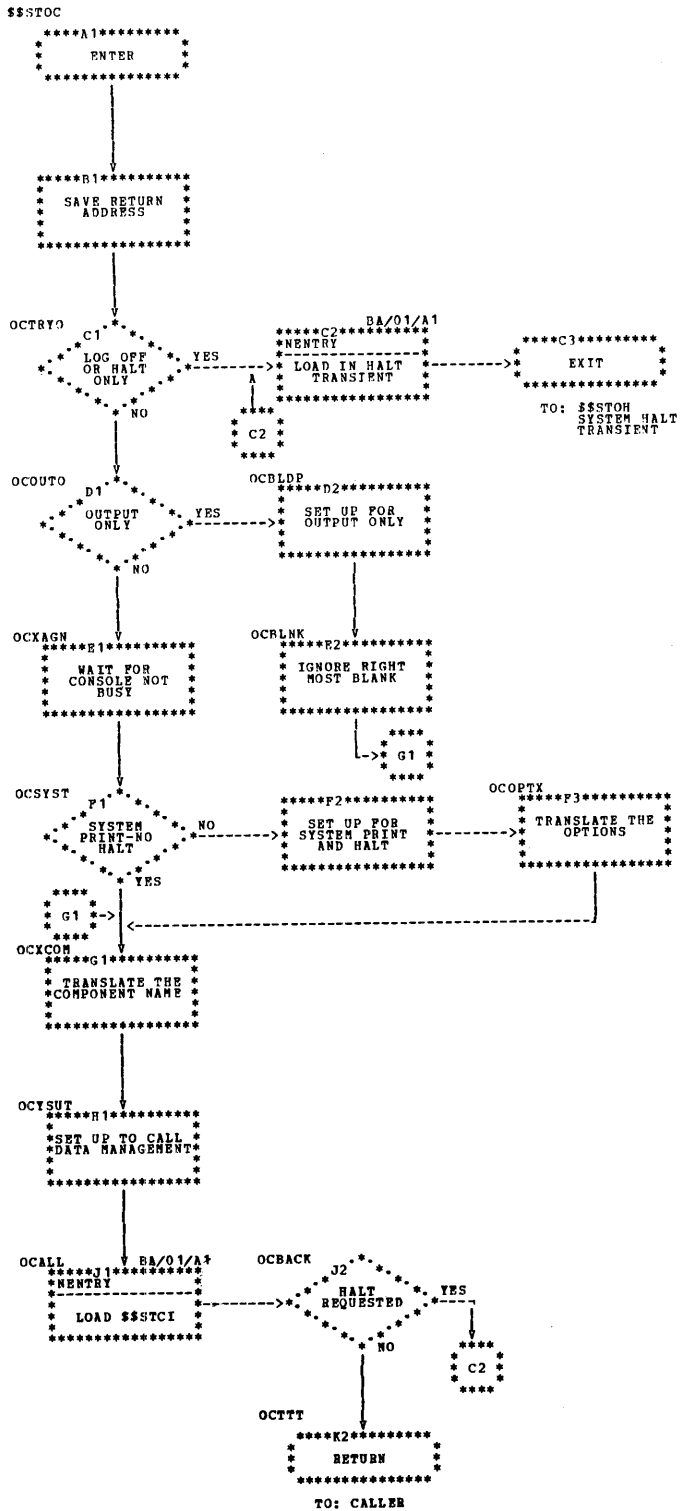


Chart JG. Halt/Syslog for the Console Printer (\$\$STOC)

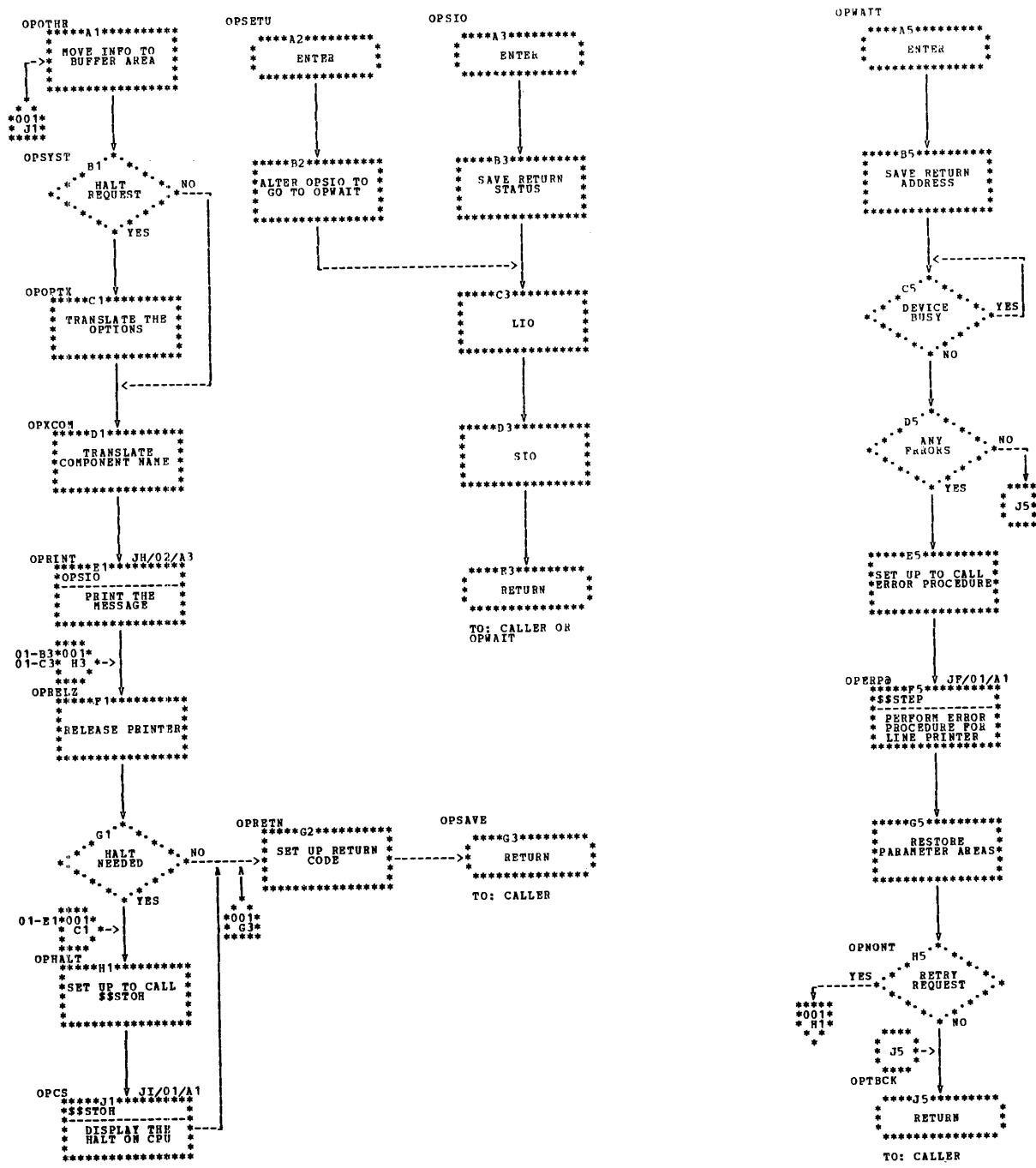


Chart JH (Part 2 of 2). Halt/Syslog for the Line Printer (\$\$STOP)

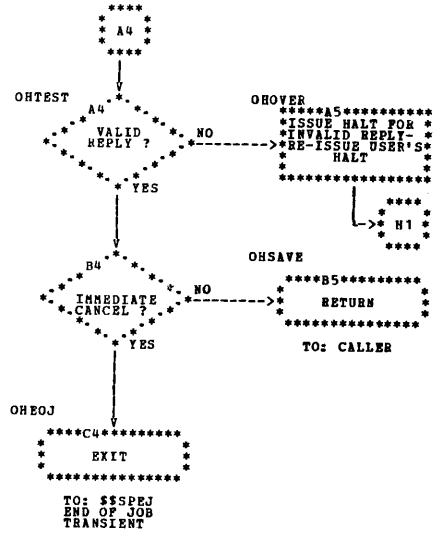
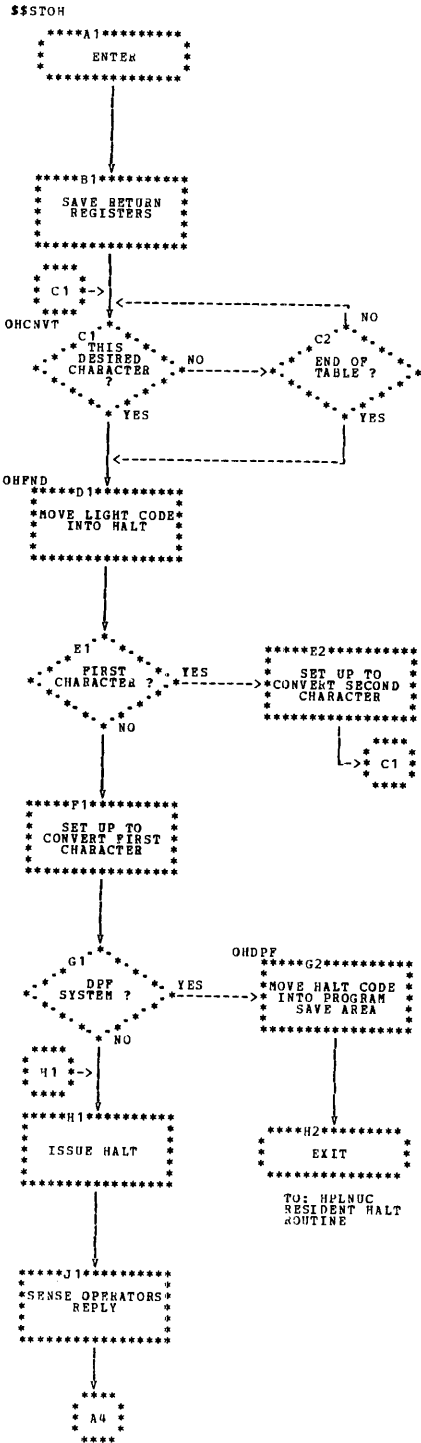


Chart JI. System Halt Transient (\$\$STOH)

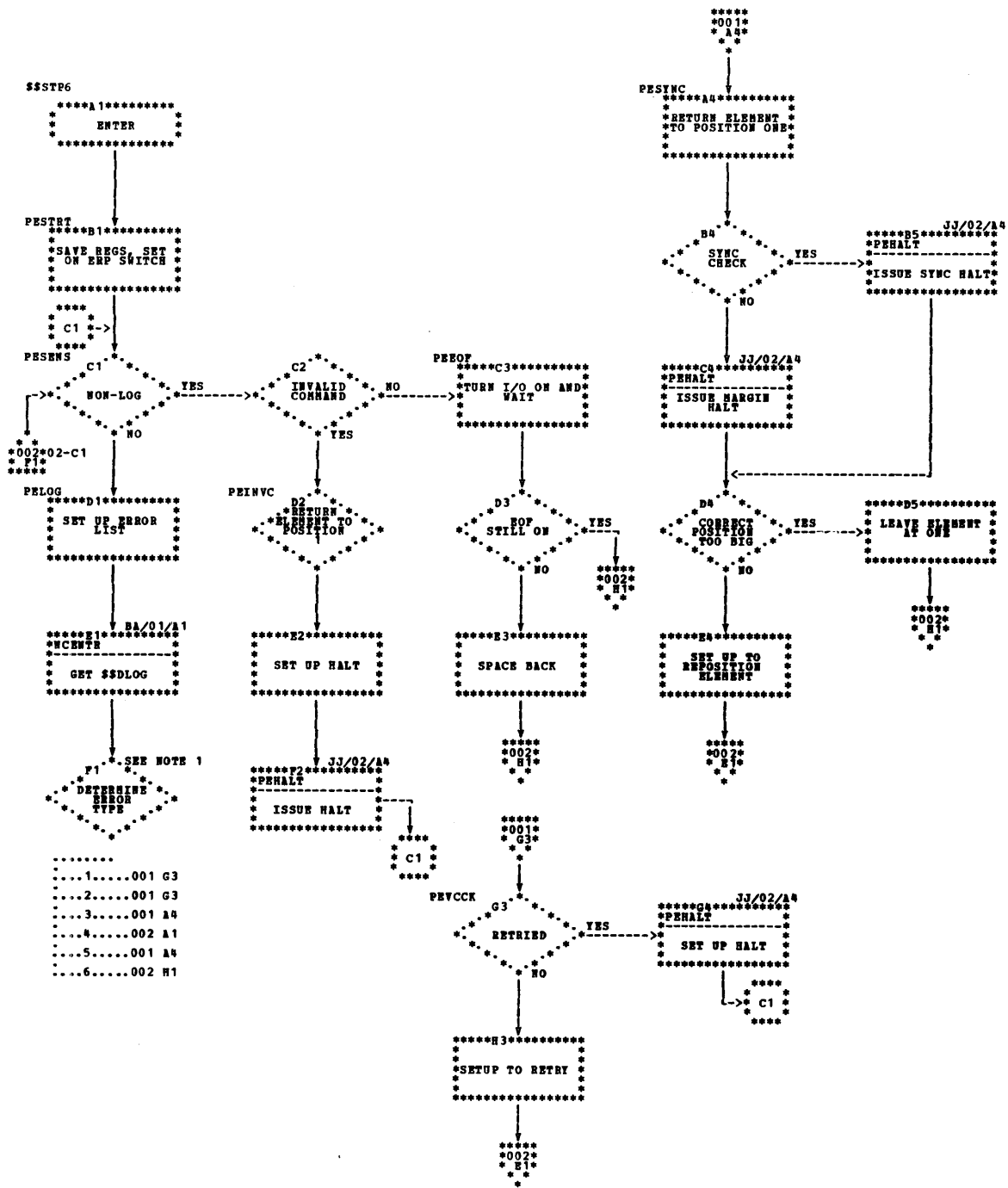


Chart JJ (Part 1 of 2). Model 6 Printer Syslog Error Recovery Procedures Routine (\$\$STP6)

NOTE 1
 1 = HORZ CYCLE CHECK
 2 = VERT CYCLE CHECK
 3 = SYNC CHECK
 4 = DATA ON EOS CHECK
 5 = MARGIN CHECK
 6 = NO ERROR

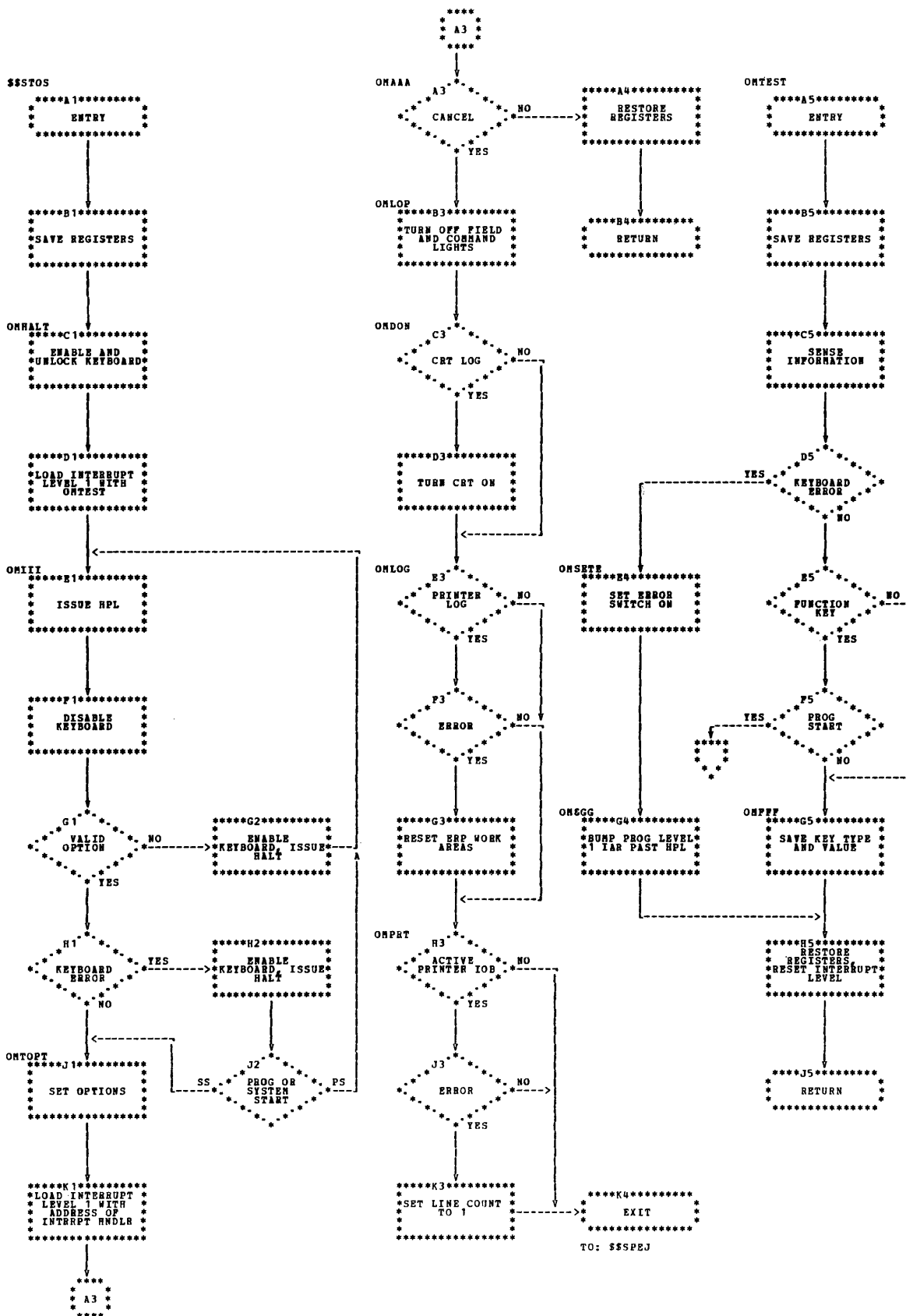


Chart JL. Model 6 Halt/Syslog Routine-Keyboard (\$\$STOS)

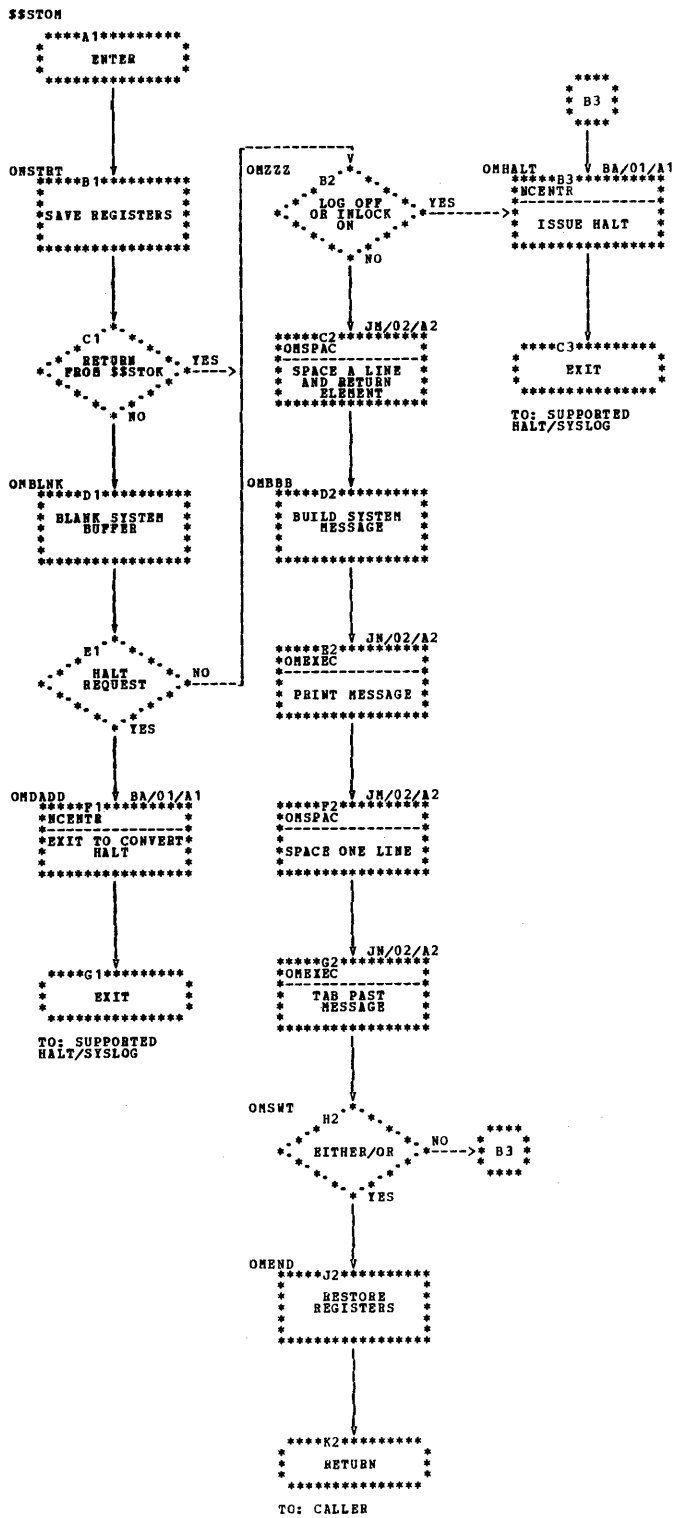


Chart JM (Part 1 of 2). Model 6 Halt/Syslog Routine—Matrix Printer (\$\$STOM)

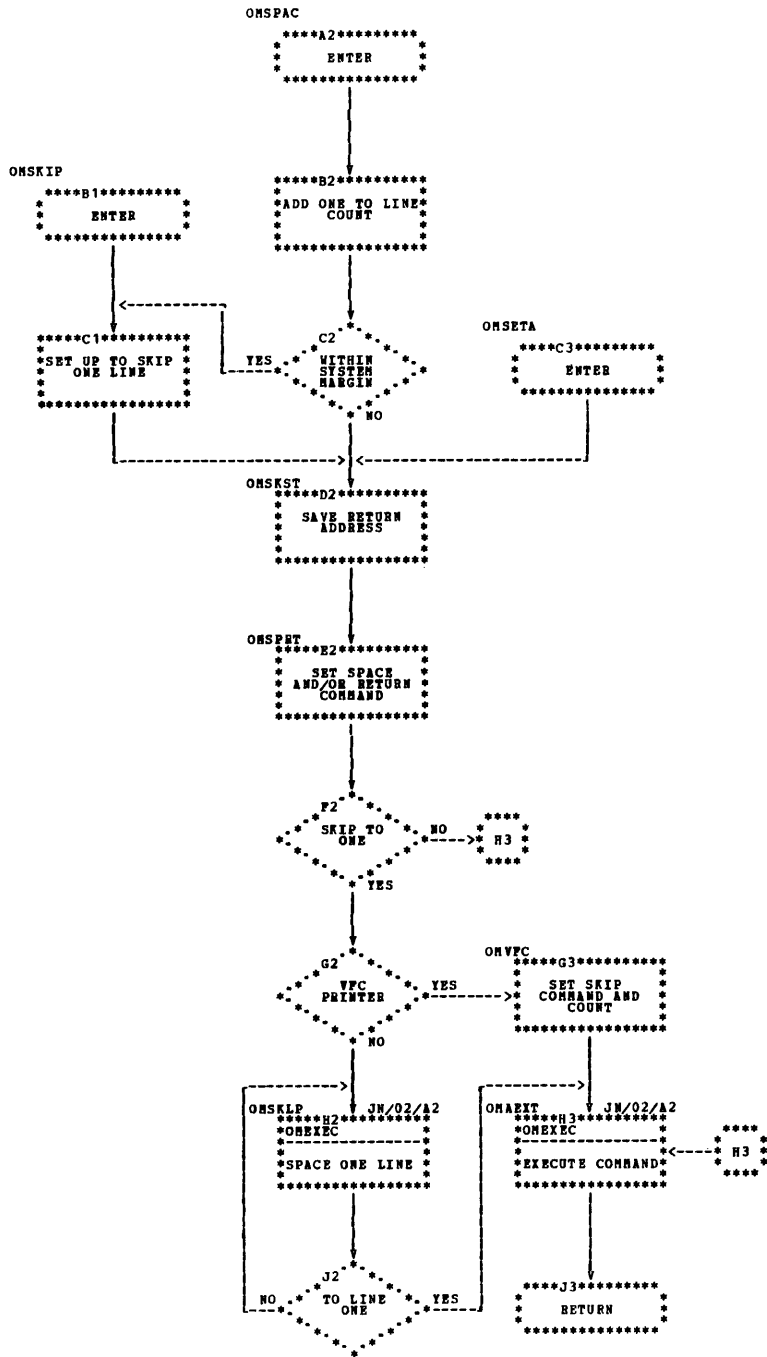


Chart JM (Part 2 of 2). Model 6 Halt/Syslog Routine—Matrix Printer (\$\$STOM)

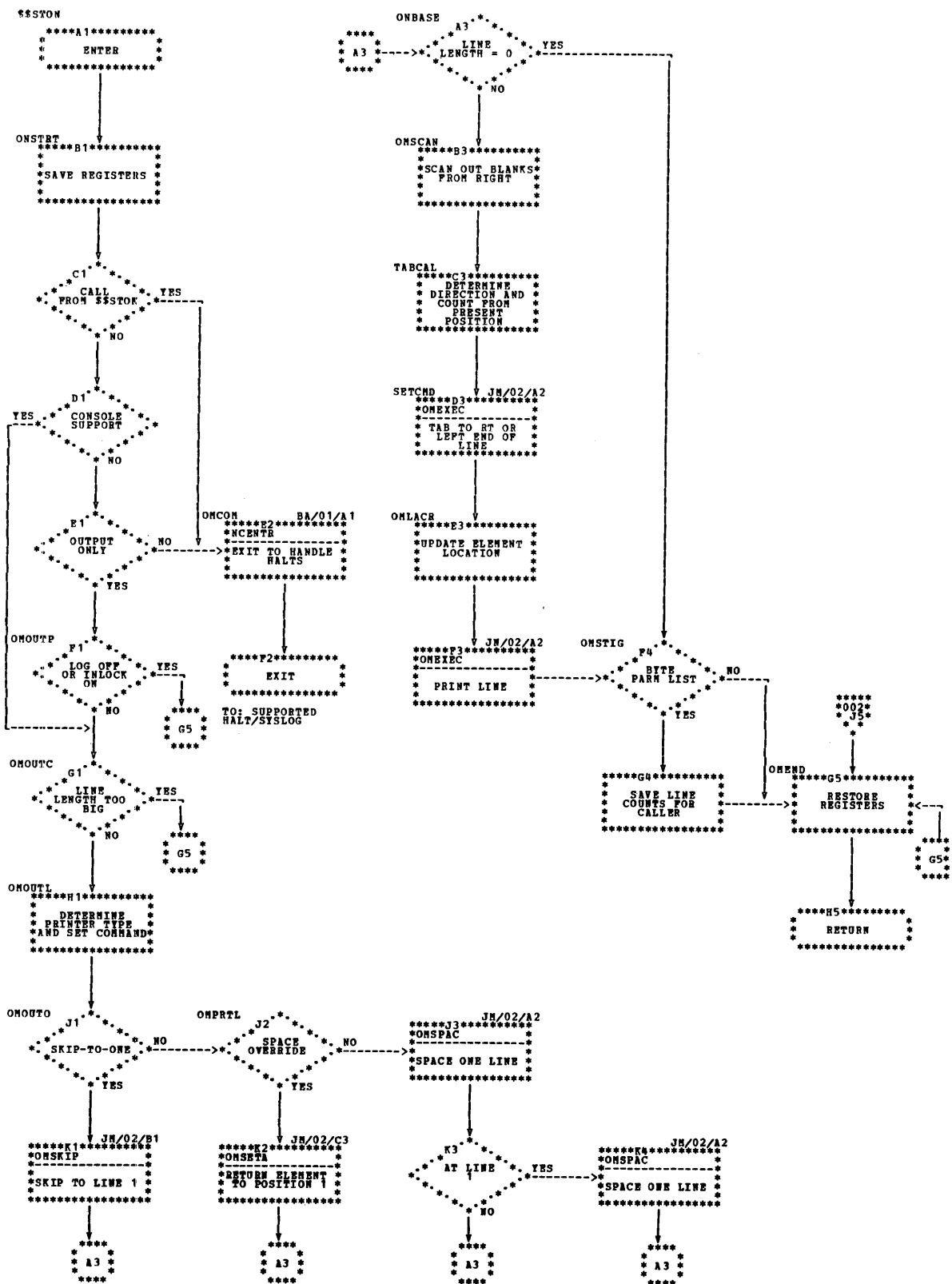


Chart JN (Part 1 of 2). Model 6 Halt/Syslog Routine-Output Only, Matrix Printer (\$\$STON)

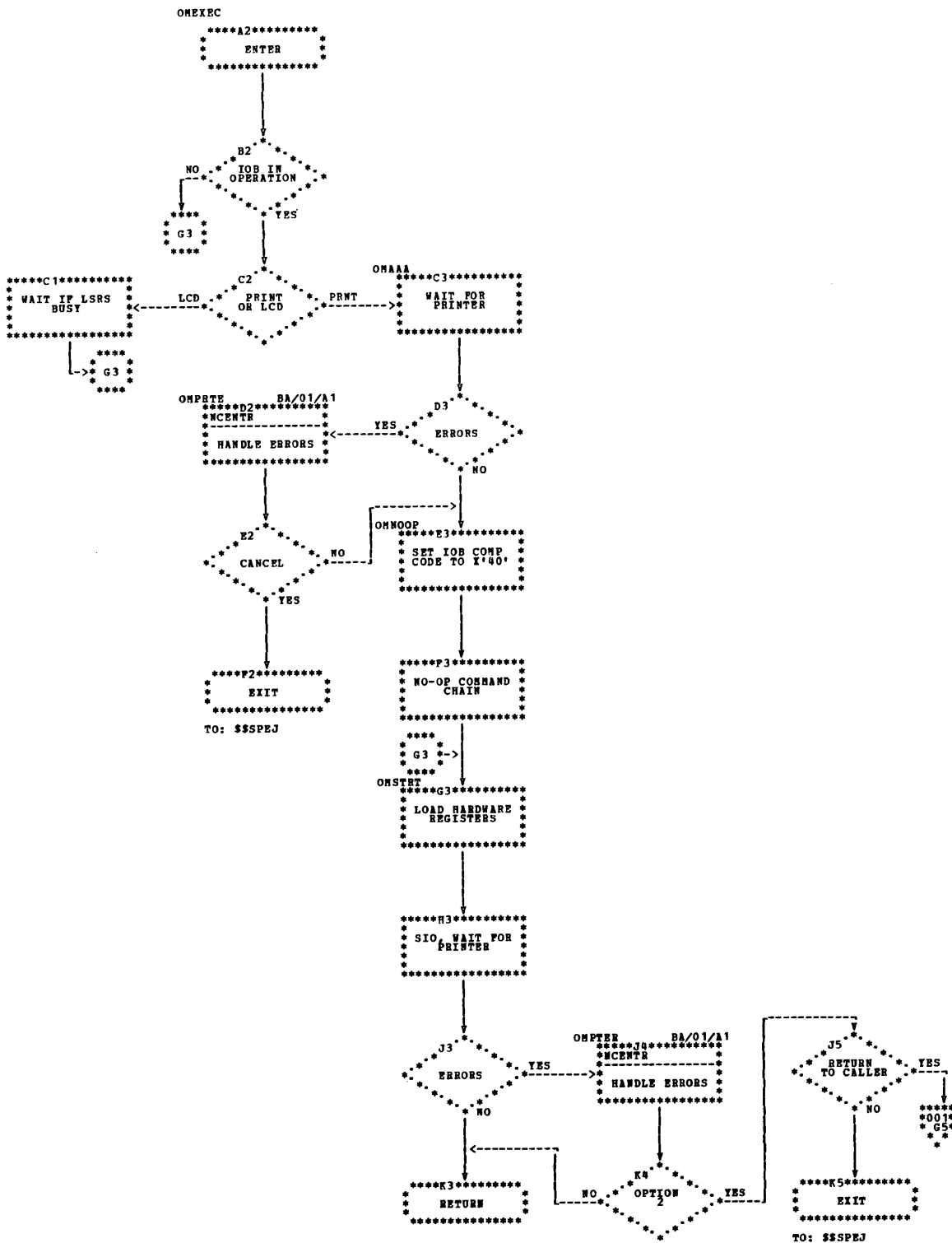


Chart JN (Part 2 of 2). Model 6 Halt/Syslog Routine—Output Only Matrix Printer (\$\$STON)

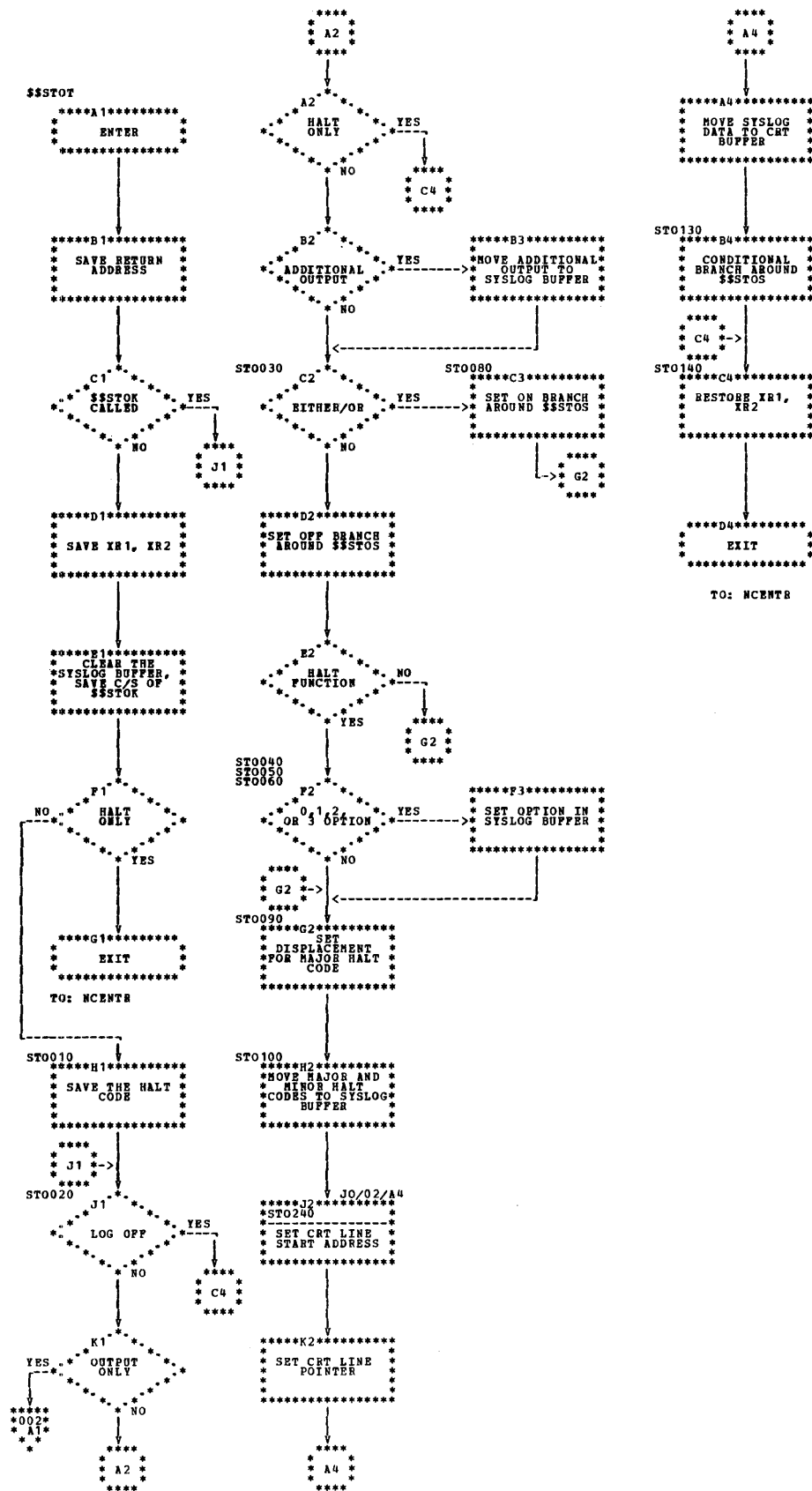


Chart JO (Part 1 of 2). Model 6 Halt/Syslog Halt Routine-CRT (\$\$STOT)

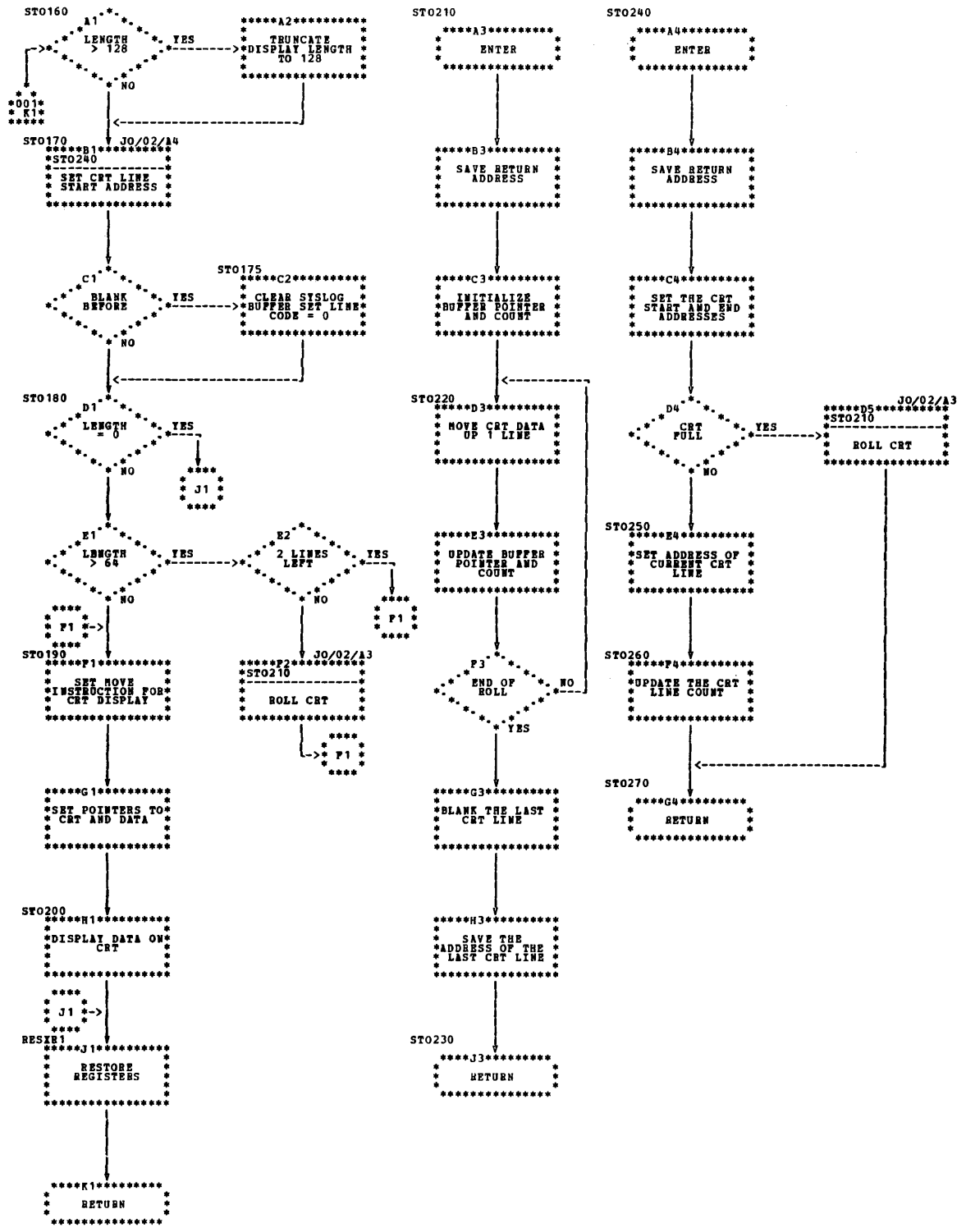


Chart JO (Part 2 of 2). Model 6 Halt/Syslog Halt Routine-CRT (\$\$STOT)

\$\$\$STOX

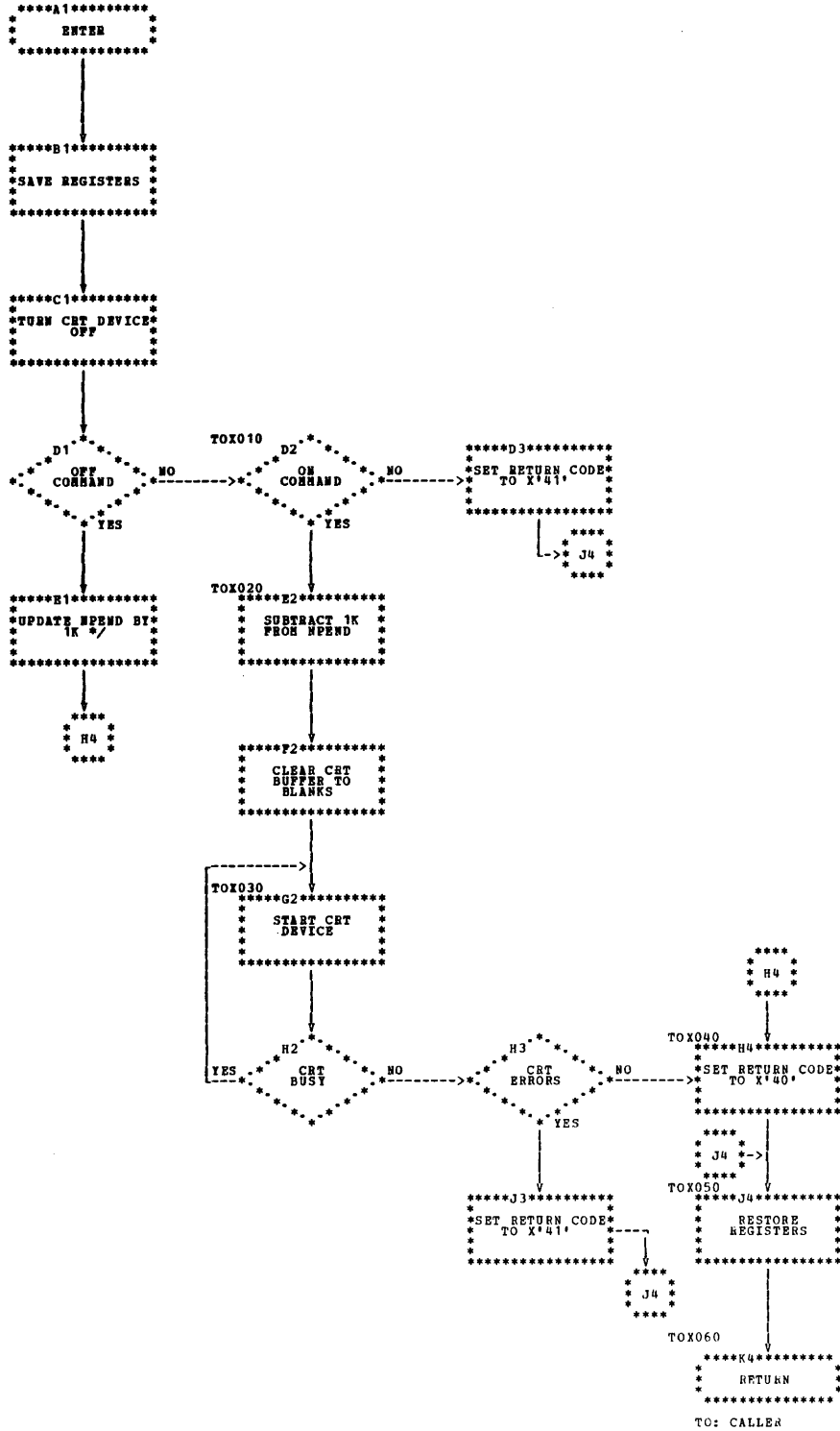


Chart JP. Model 6 Halt/Syslog Routine—CRT On and Off Transient (\$\$\$STOX)

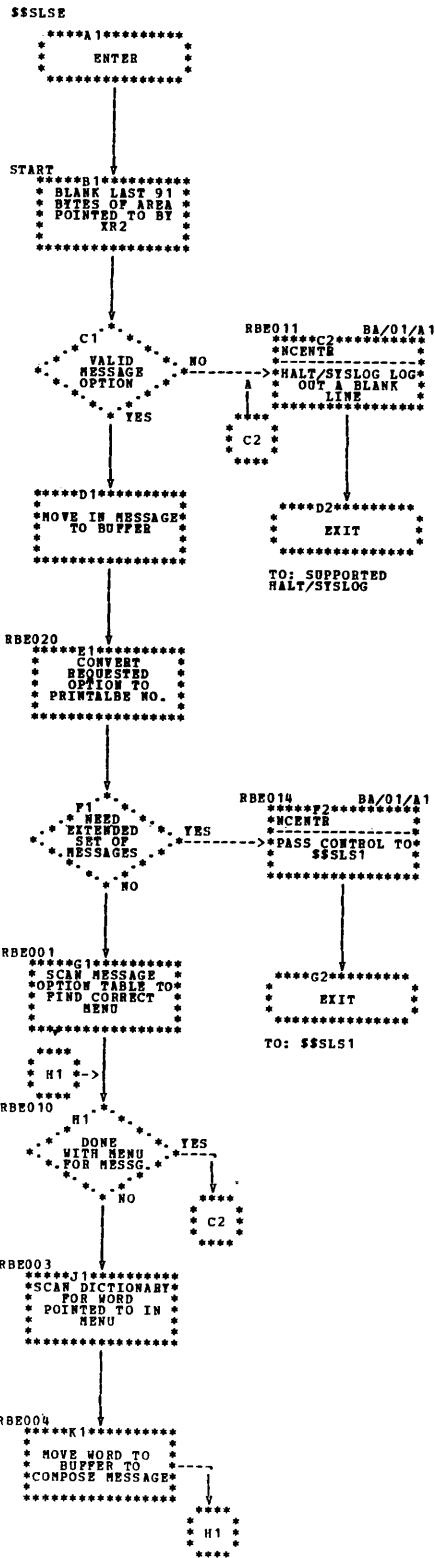


Chart JQ. Model 6 Halt/Syslog Scheduler Error Messages Routine (\$\$SLSE)

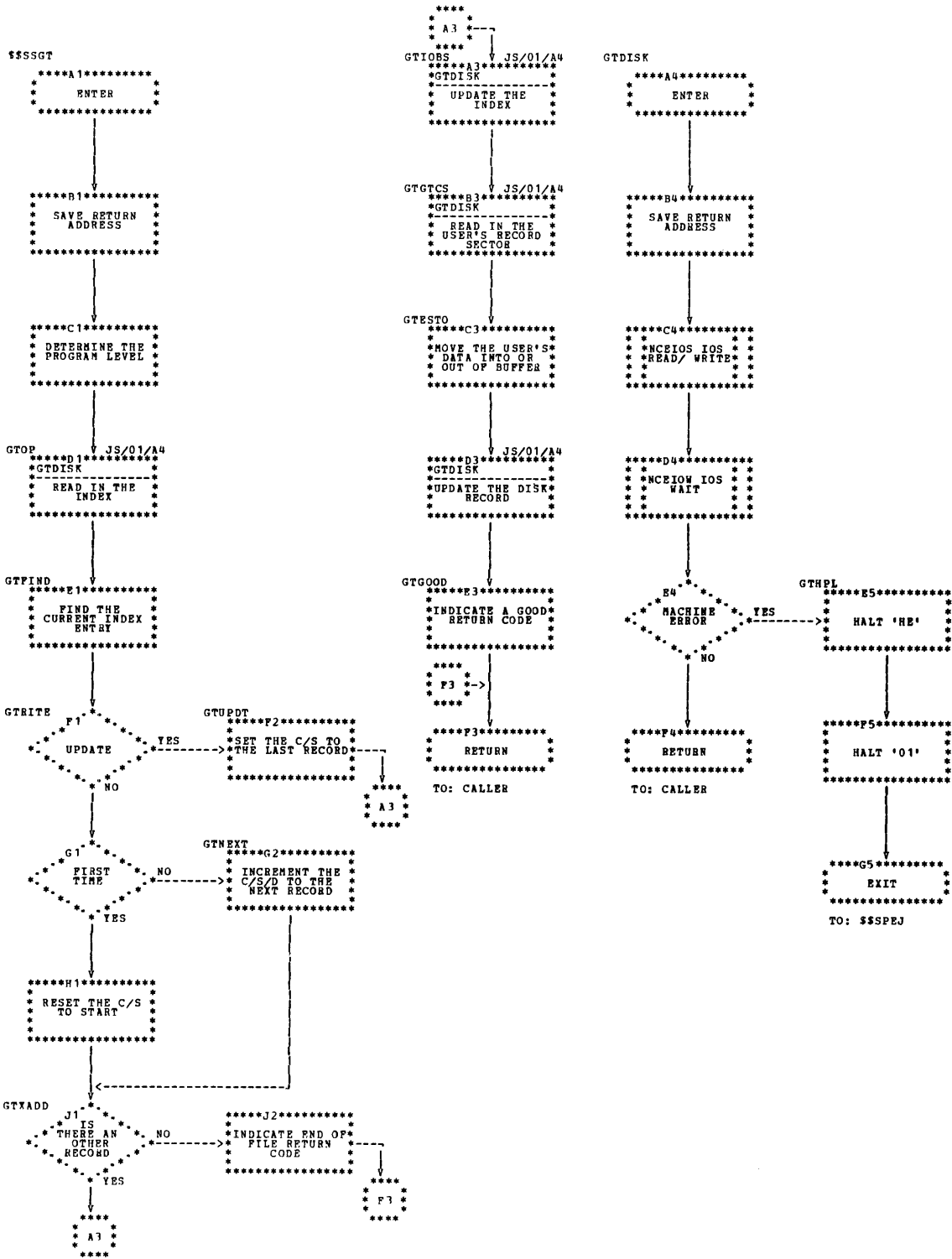


Chart JS. Scheduler Work Area-Get (\$SSSGT)

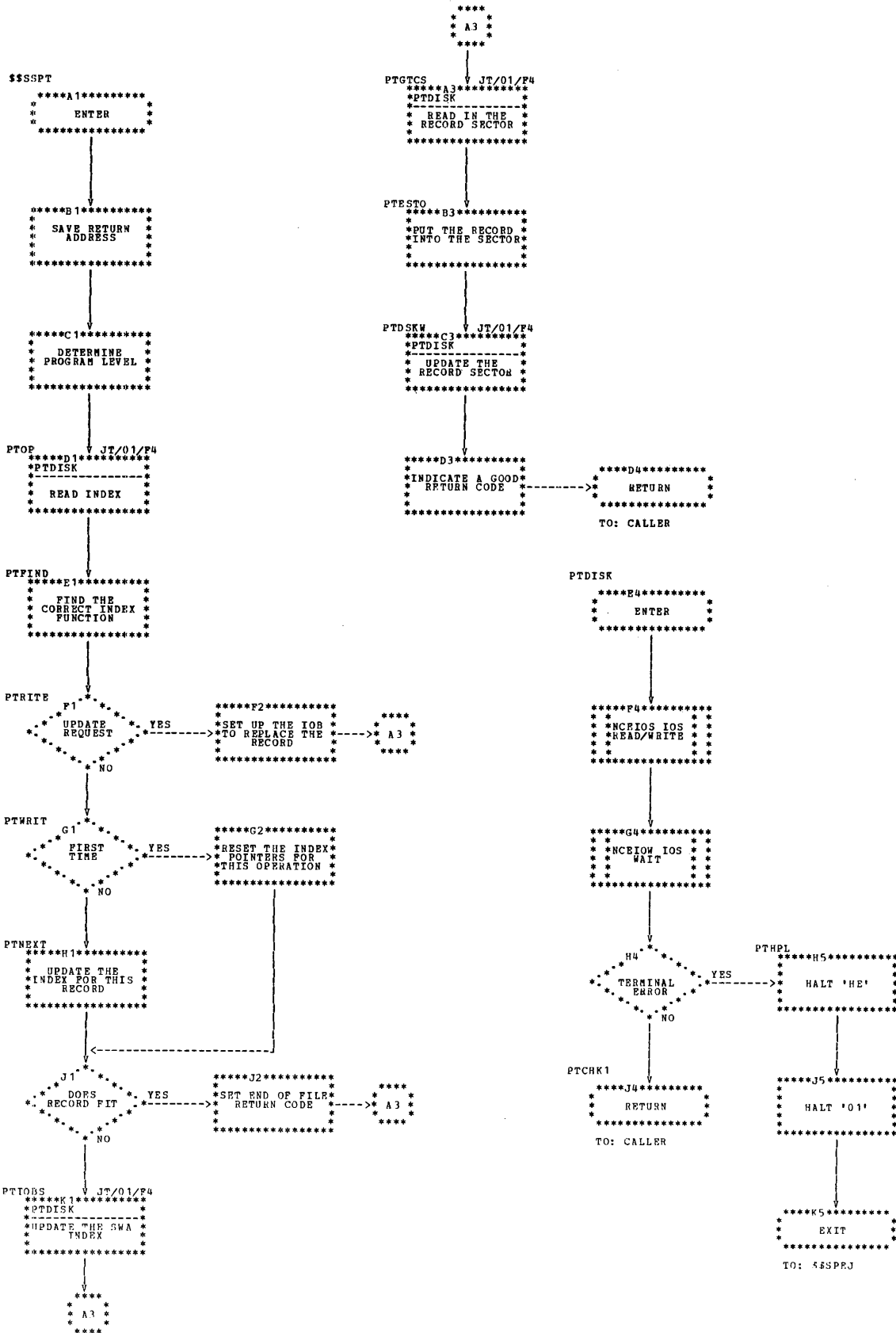


Chart JT. Scheduler Work Area—Put (\$\$\$SPT)

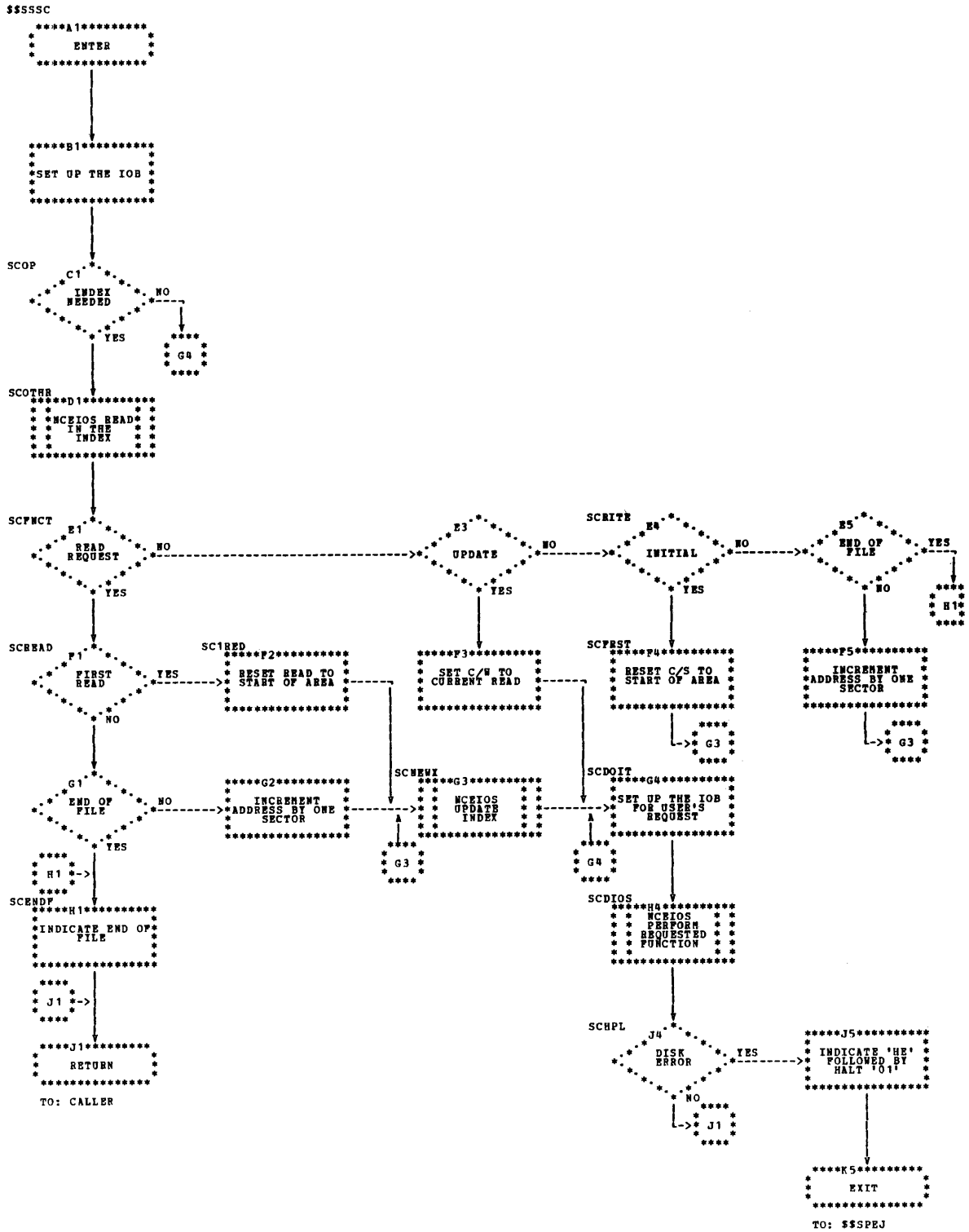


Chart JU. Scheduler Work Area—Read/Write (\$\$SSSC)

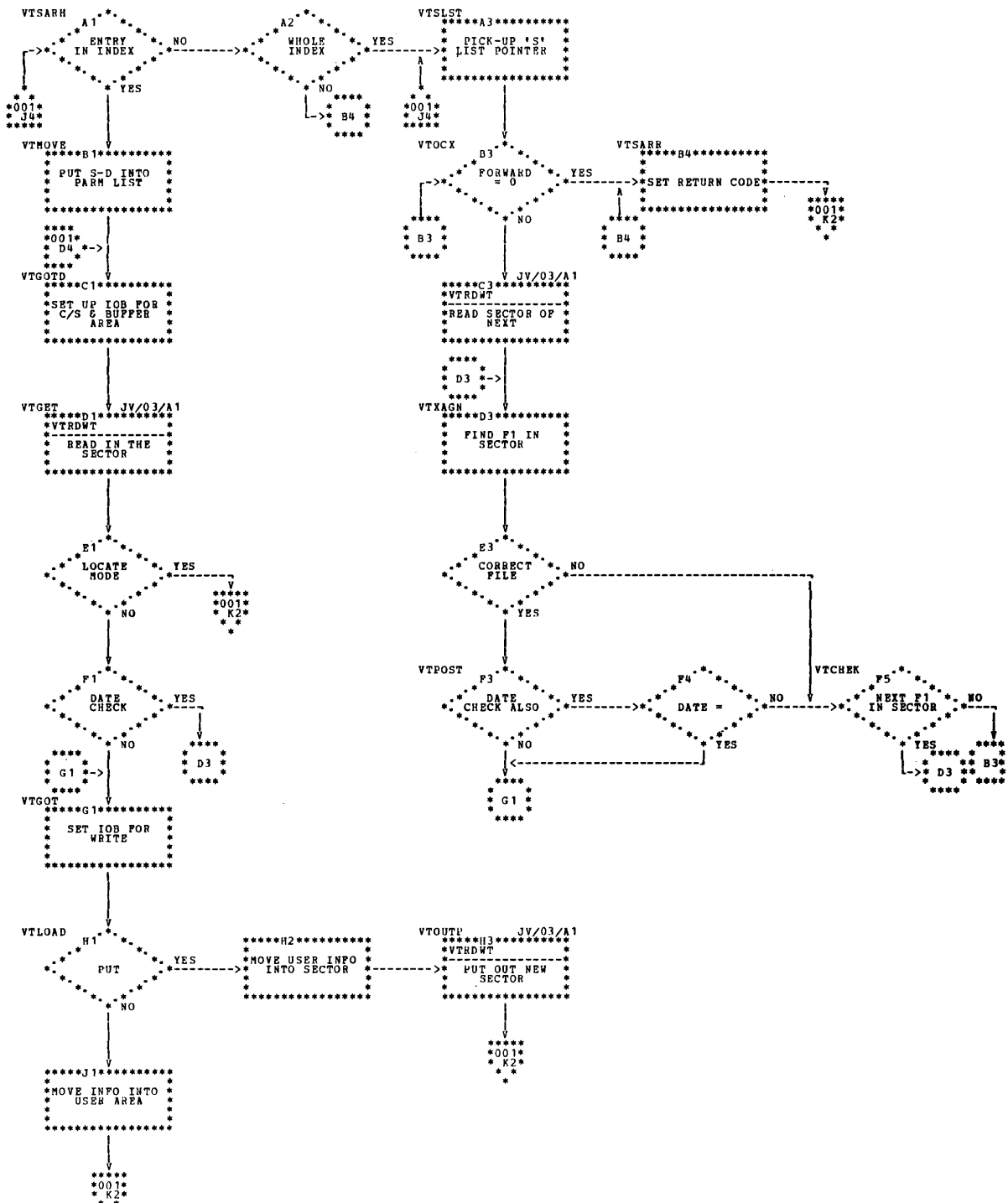


Chart JV (Part 2 of 3). Volume Table of Contents-Read/Write (\$SSVT)

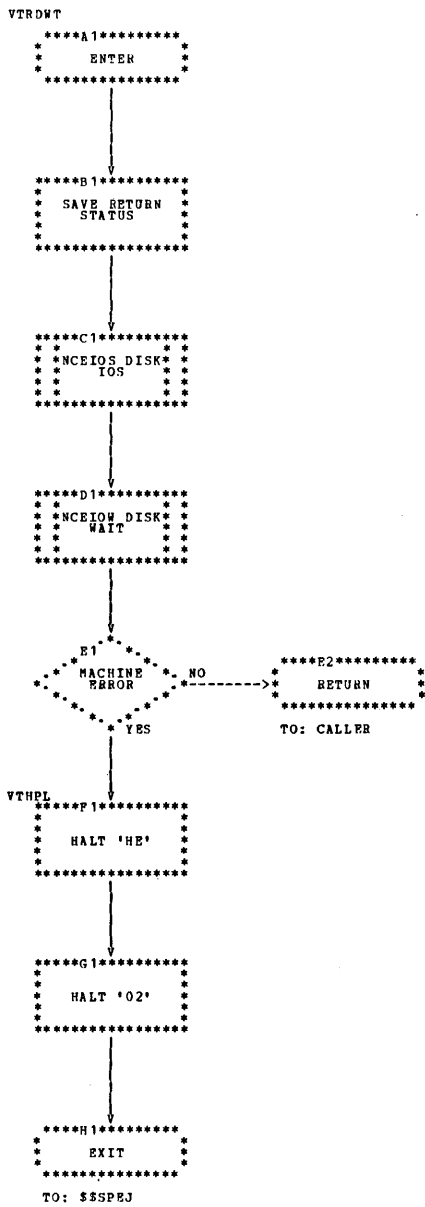


Chart JV (Part 3 of 3). Volume Table of Contents—Read/Write (\$\$SSVT)

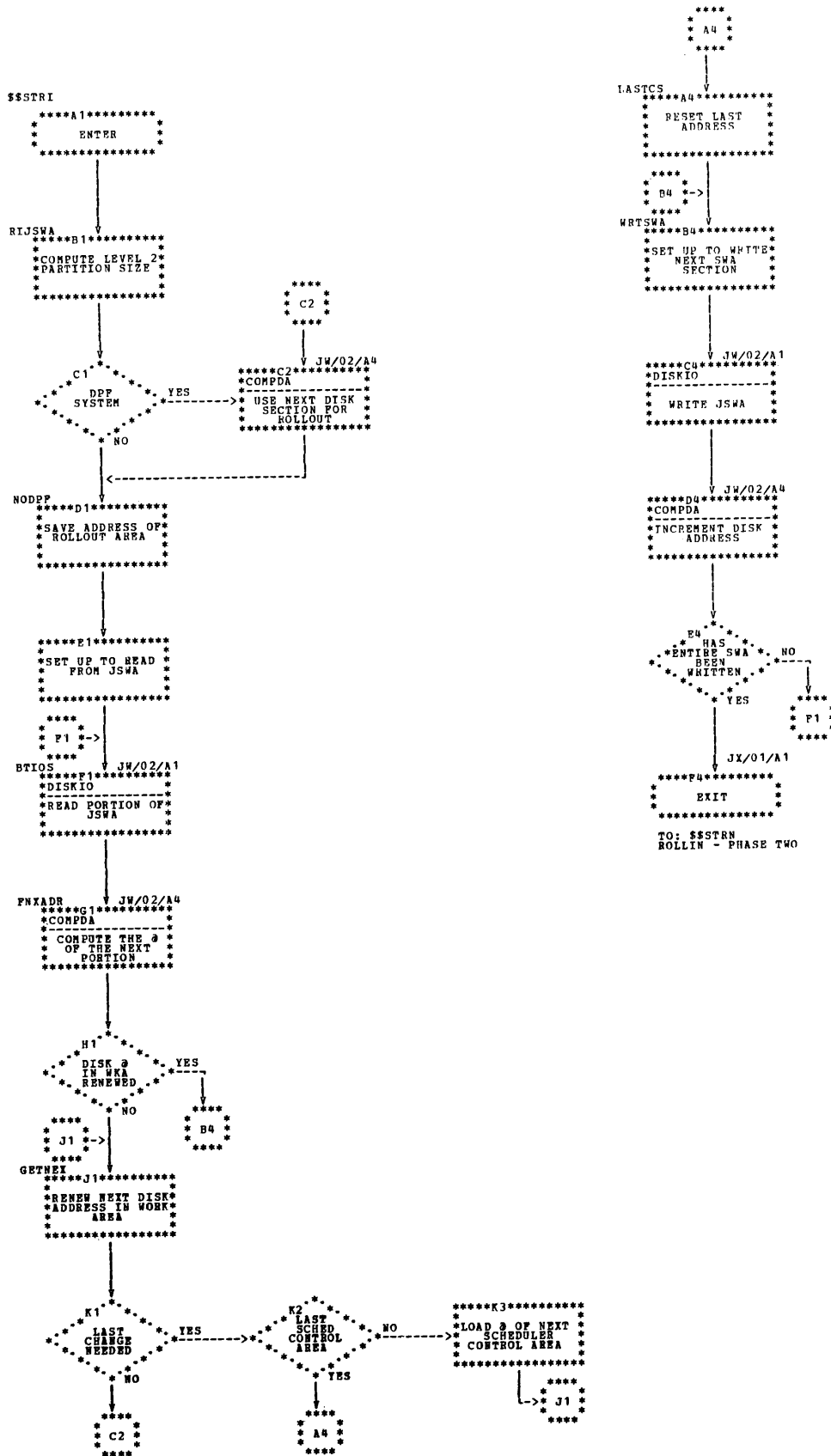
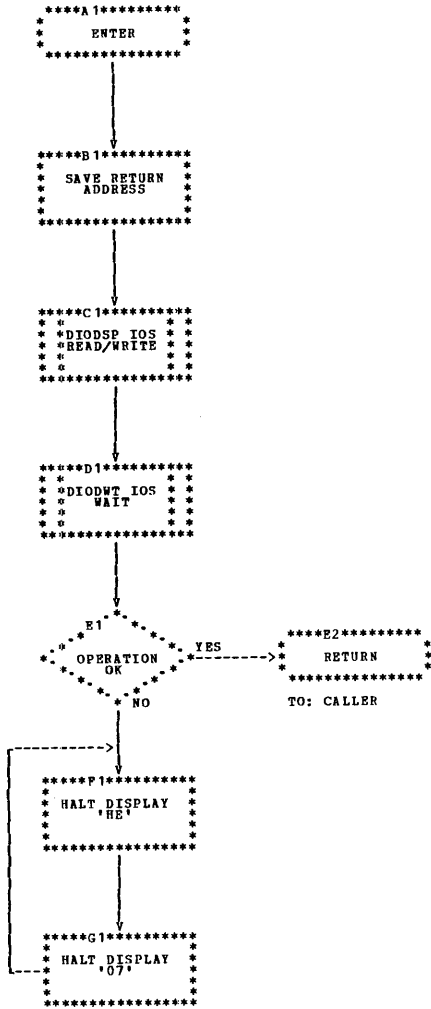


Chart JW (Part 1 of 2). Rollin-Phase One (\$\$STRI)

DISKIO



COMPDA

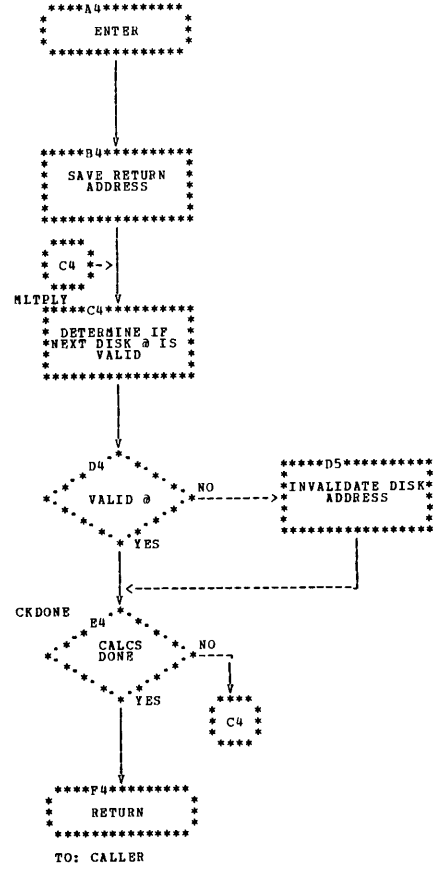


Chart JW (Part 2 of 2). Rollin-Phase One (\$\$STRI)

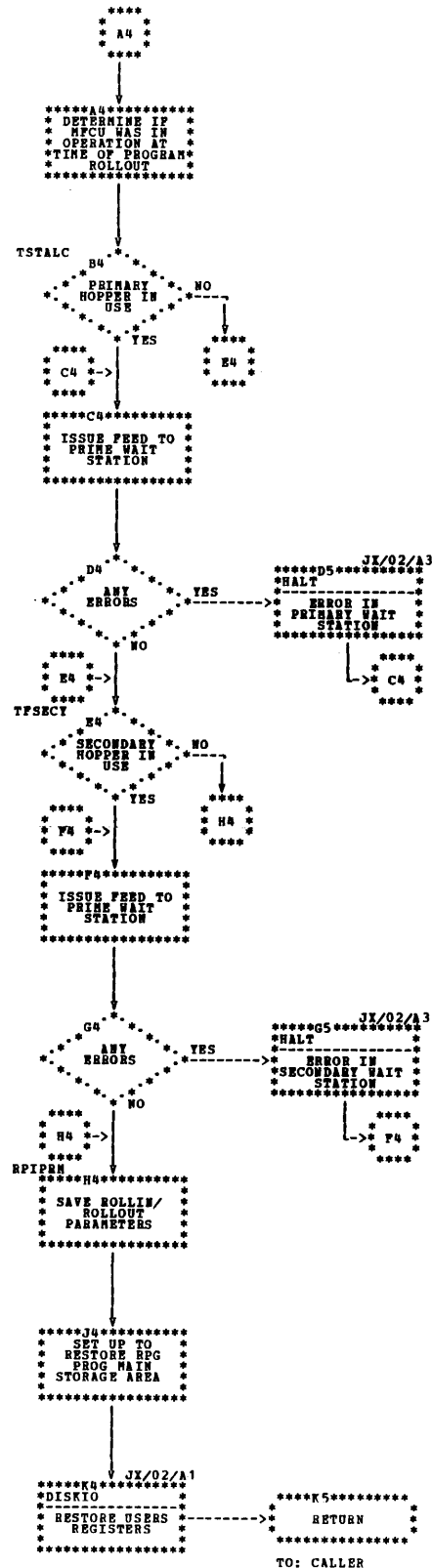
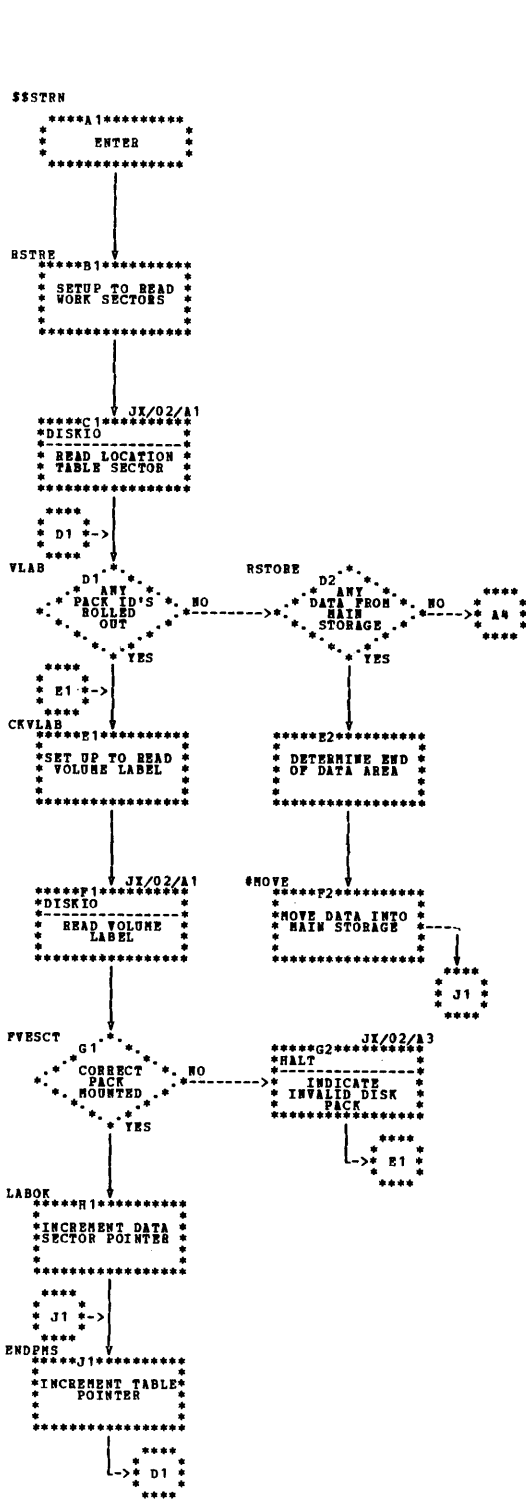


Chart JX (Part 1 of 2). Rollin-Phase Two (\$\$STRN)

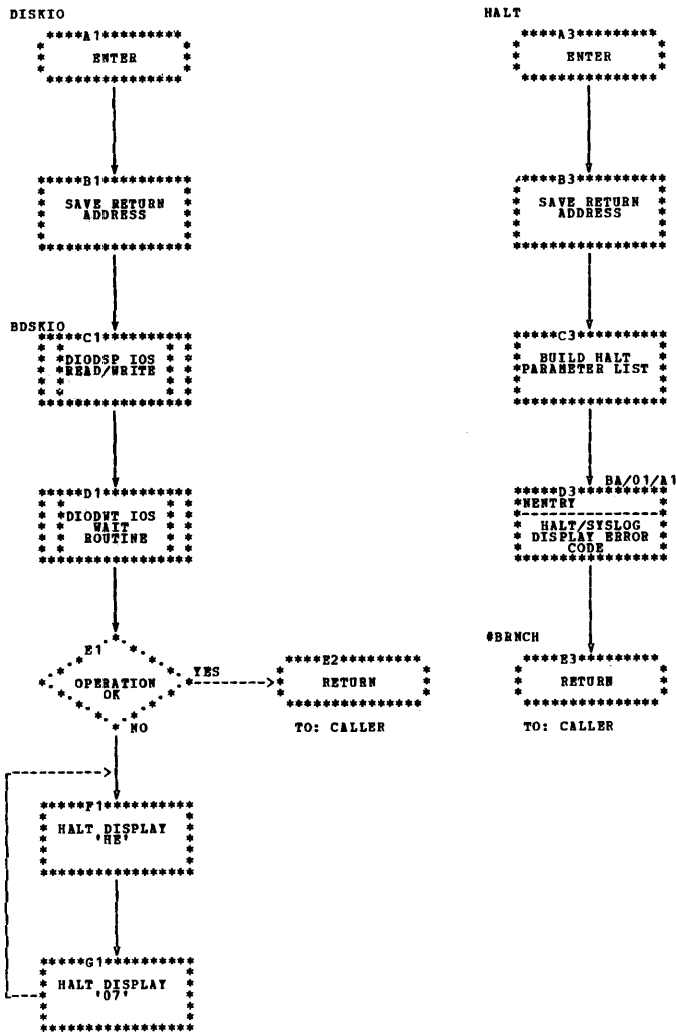


Chart JX (Part 2 of 2). Rollin-Phase Two (\$\$STRN)

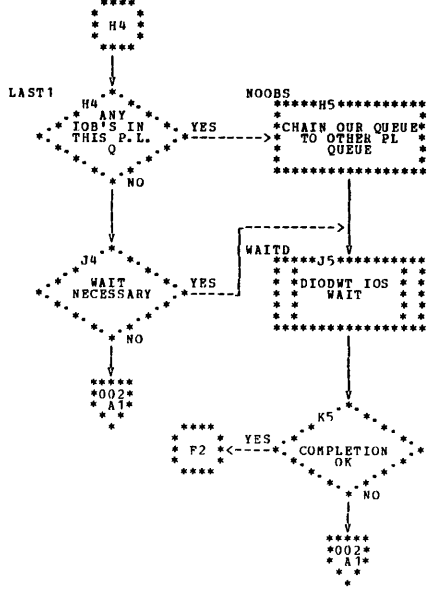
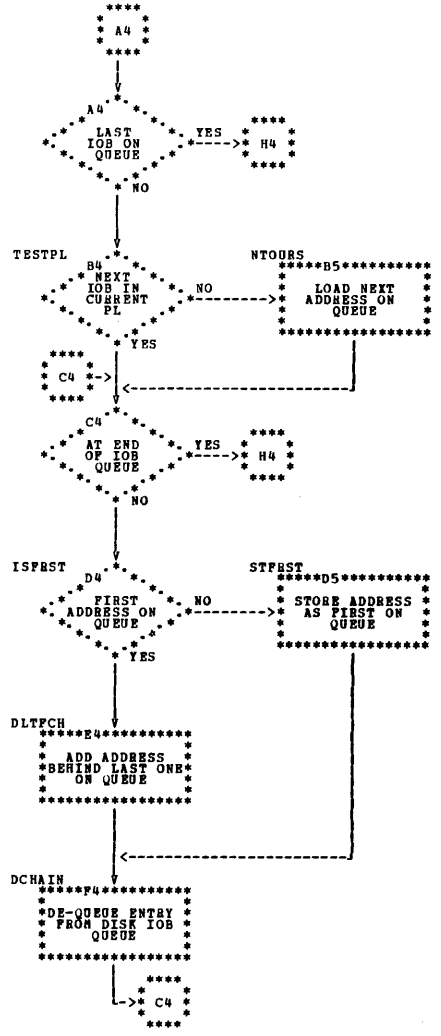
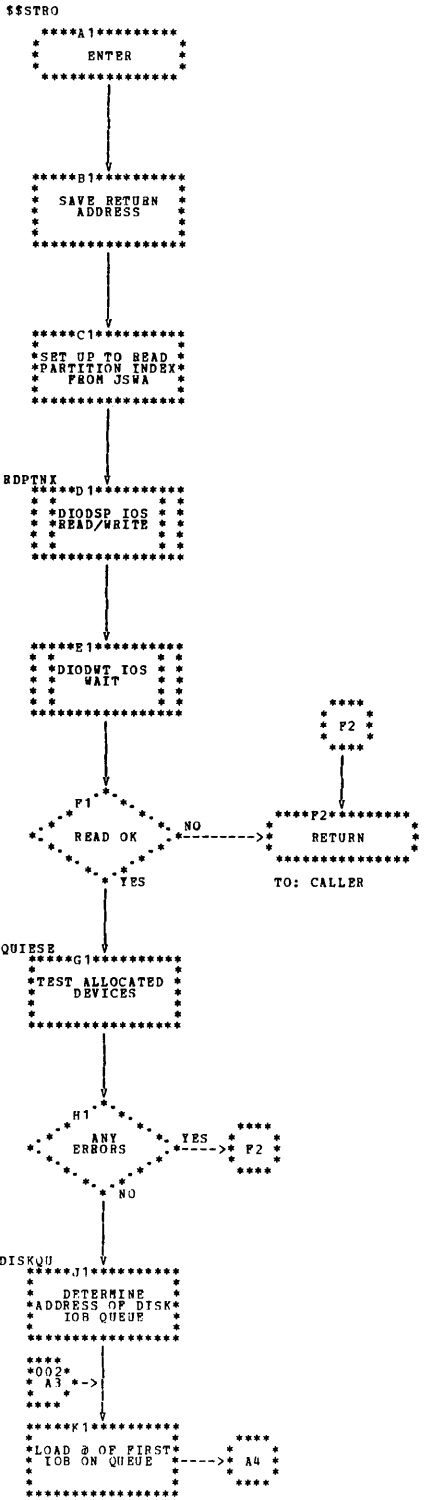


Chart JY (Part 1 of 2). Rollout-Phase One (\$\$STRO)

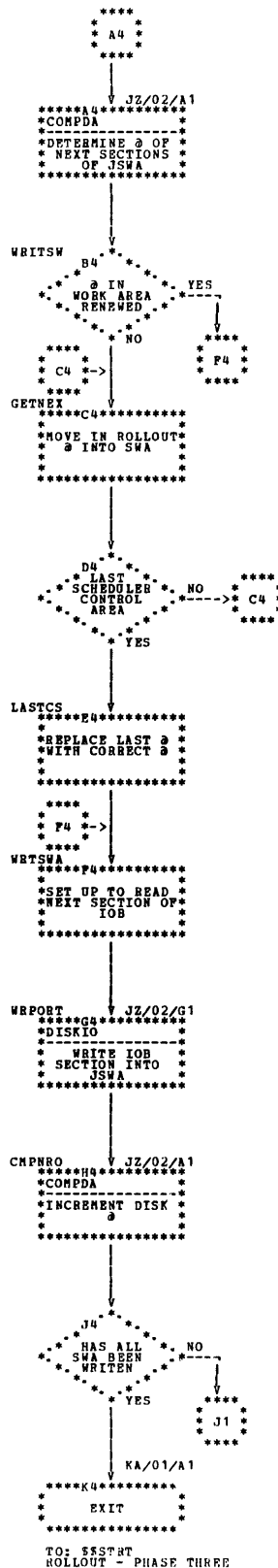
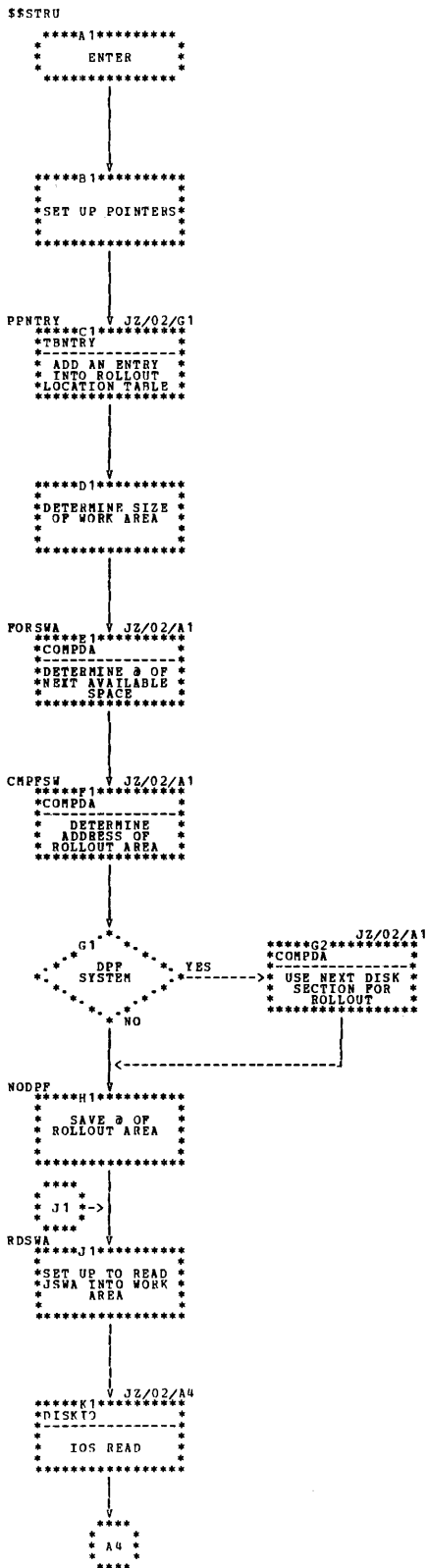


Chart JZ (Part 1 of 2). Rollout-Phase Two (\$\$STRU)

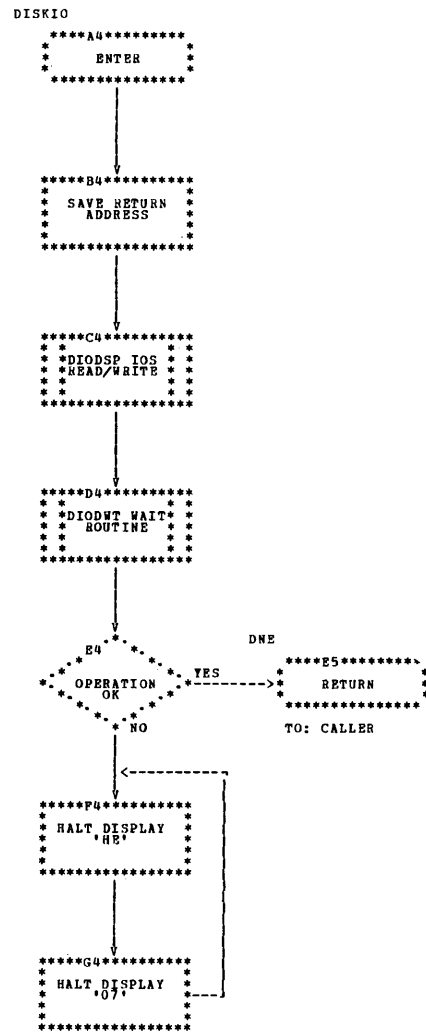
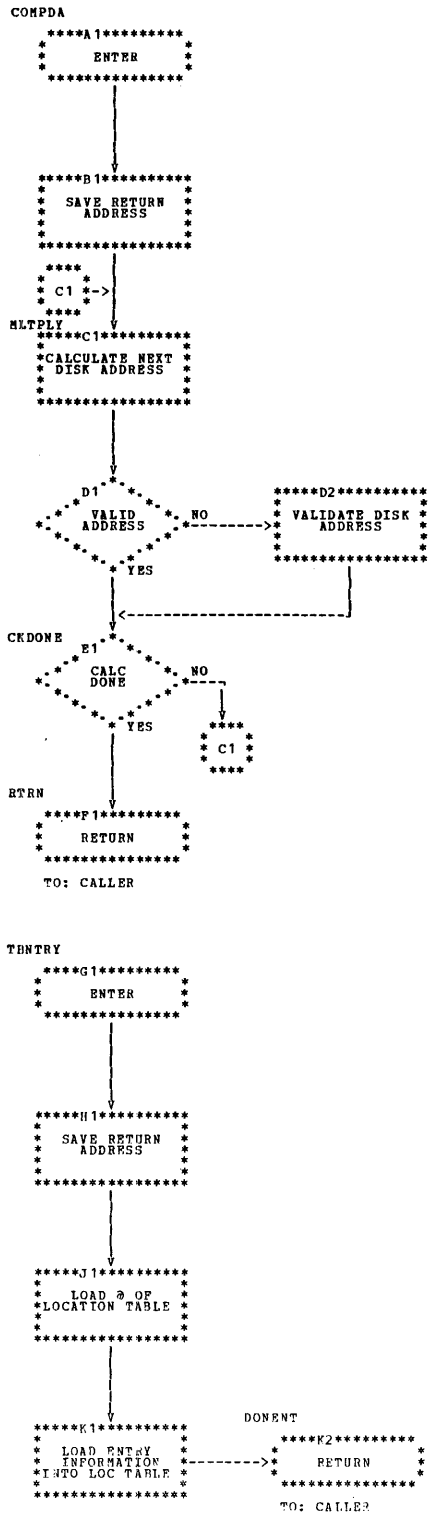


Chart JZ (Part 2 of 2). Rollout-Phase Two (\$\$STRU)

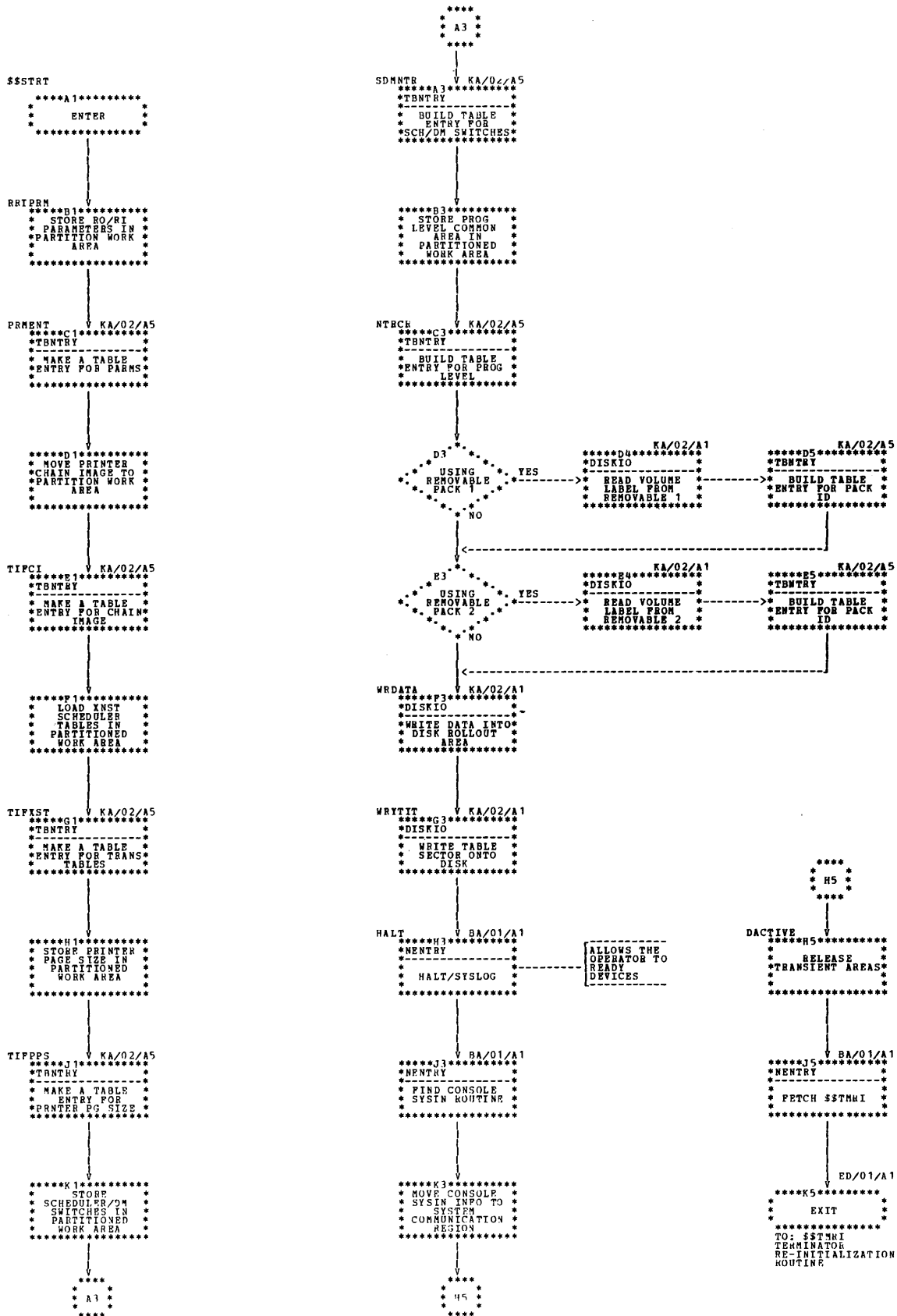
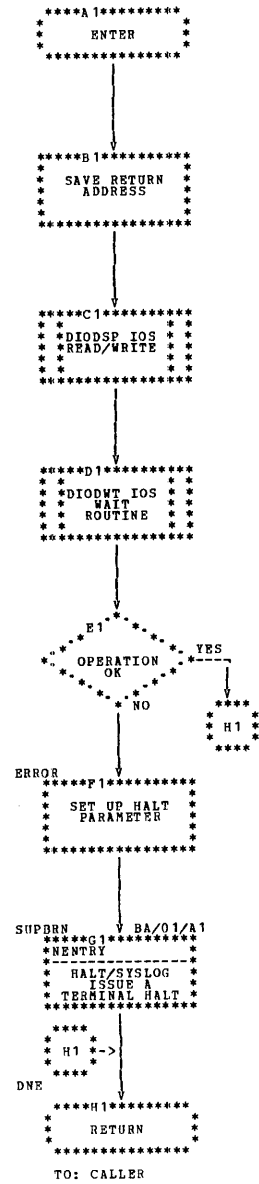
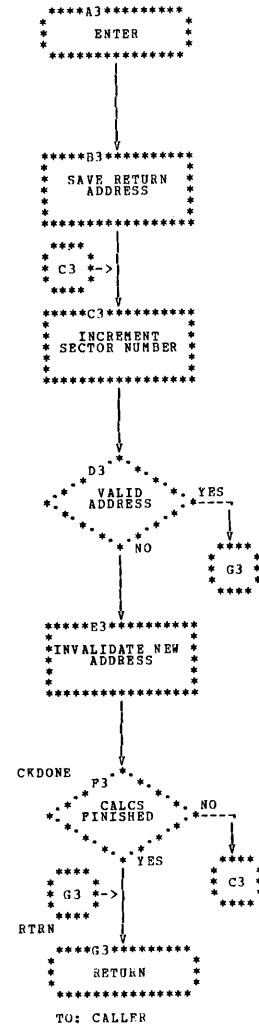


Chart KA (Part 1 of 2). Rollout-Phase Three (\$\$STRT)

DISK IO



COMPDA



TBNTAY

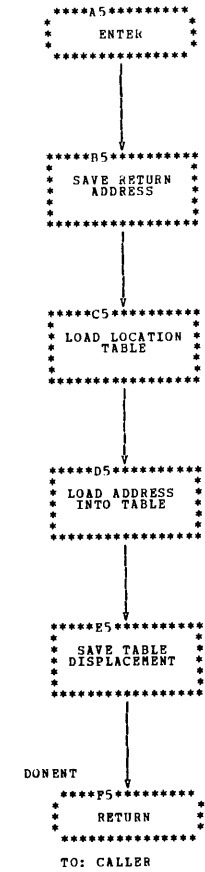


Chart KA (Part 2 of 2). Rollout-Phase Three (\$\$STRT)

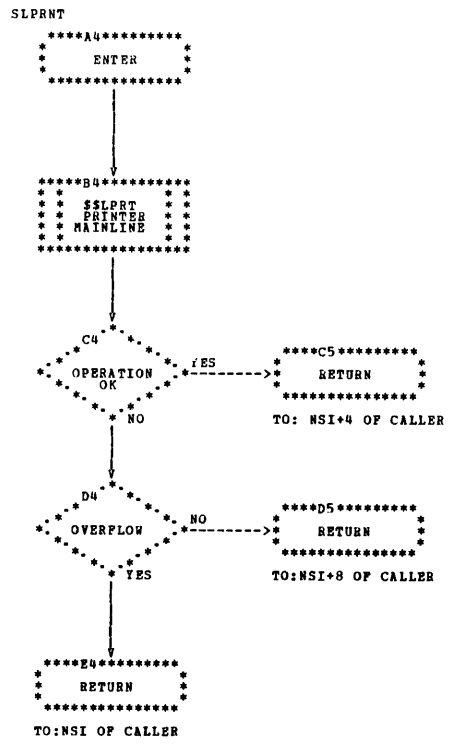
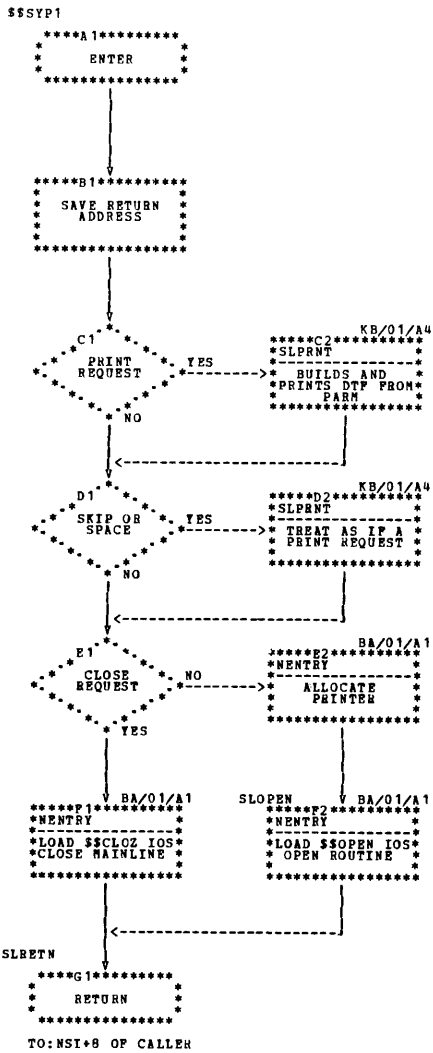


Chart KB. Disk System System List-Printer (\$\$SYP1)

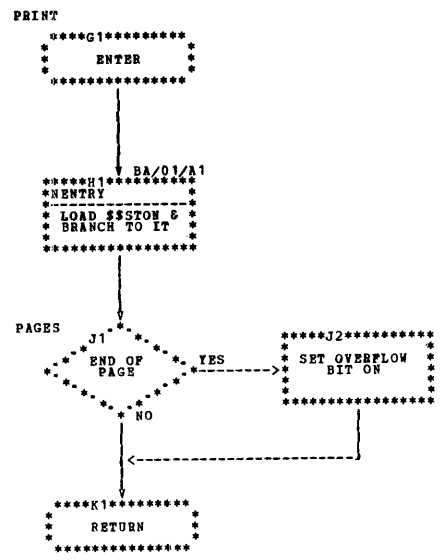
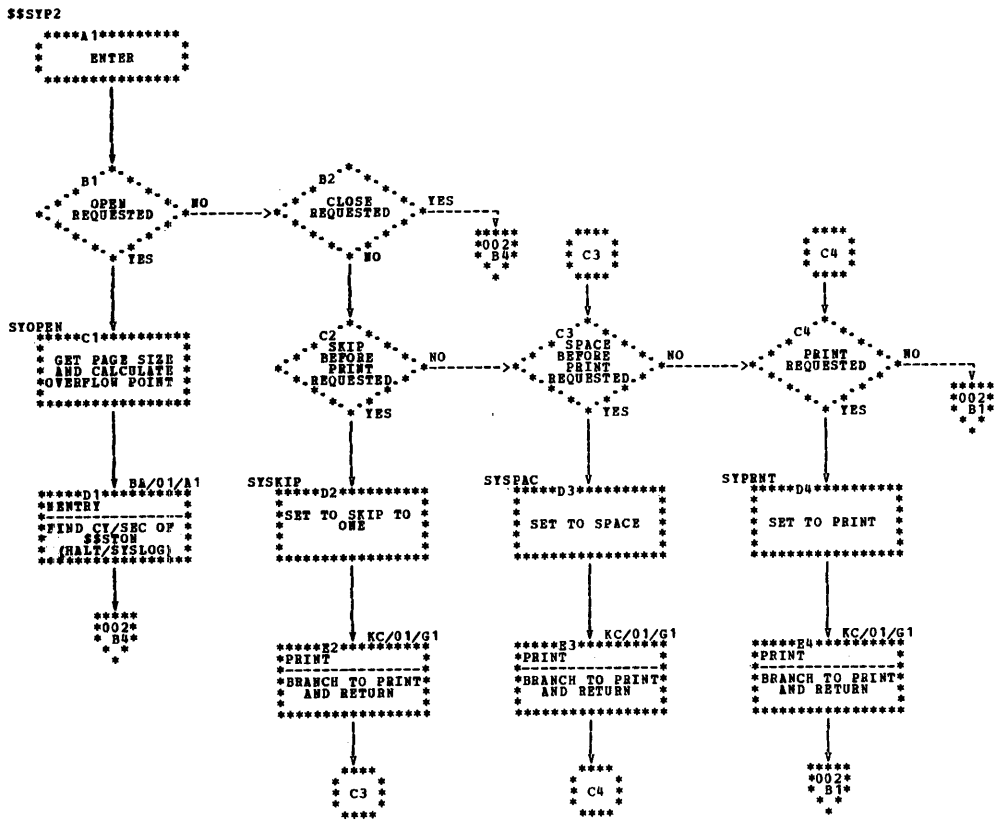


Chart KC (Part 1 of 2). Model 6 System List-Printer (\$\$SYP2)

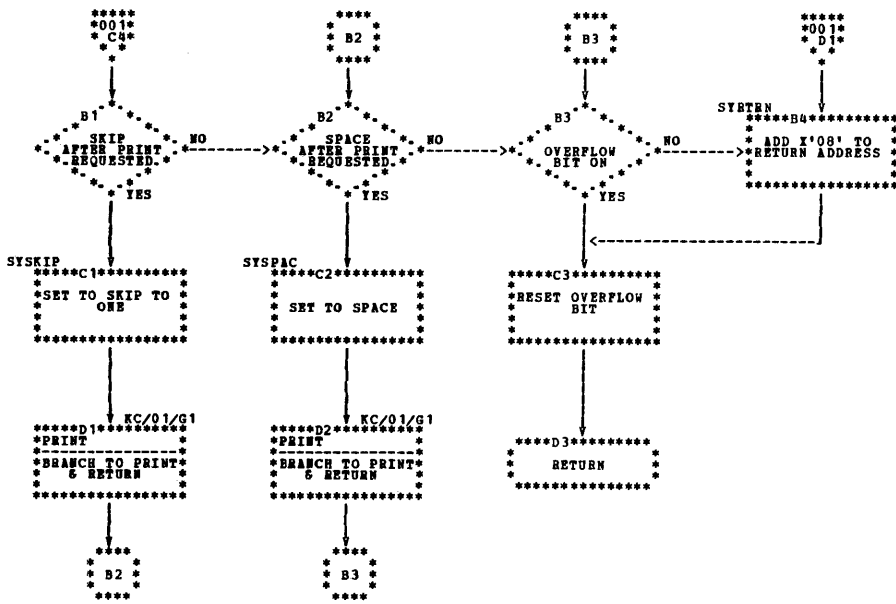


Chart KC (Part 2 of 2). Model 6 System List-Printer (\$\$SYP2)

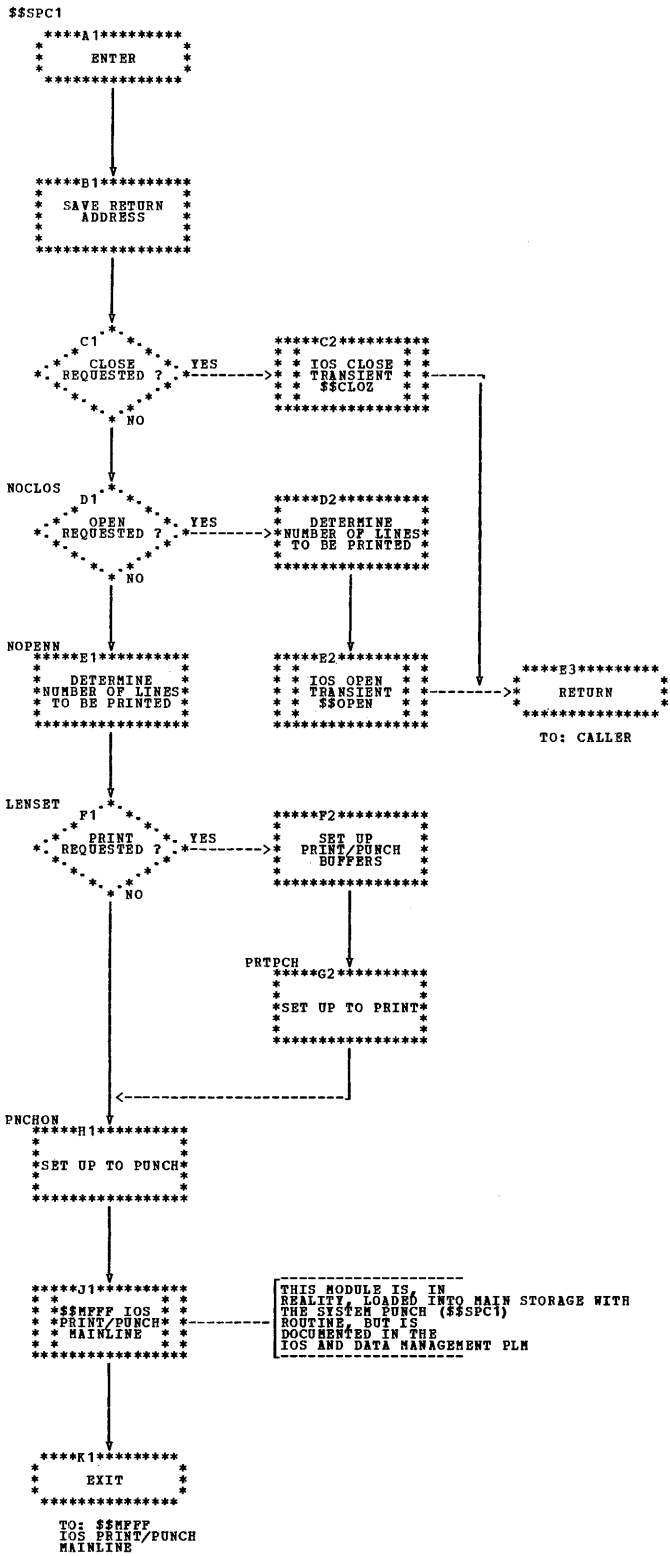


Chart KD. Disk System System List-Print/Punch (\$\$SPC1)

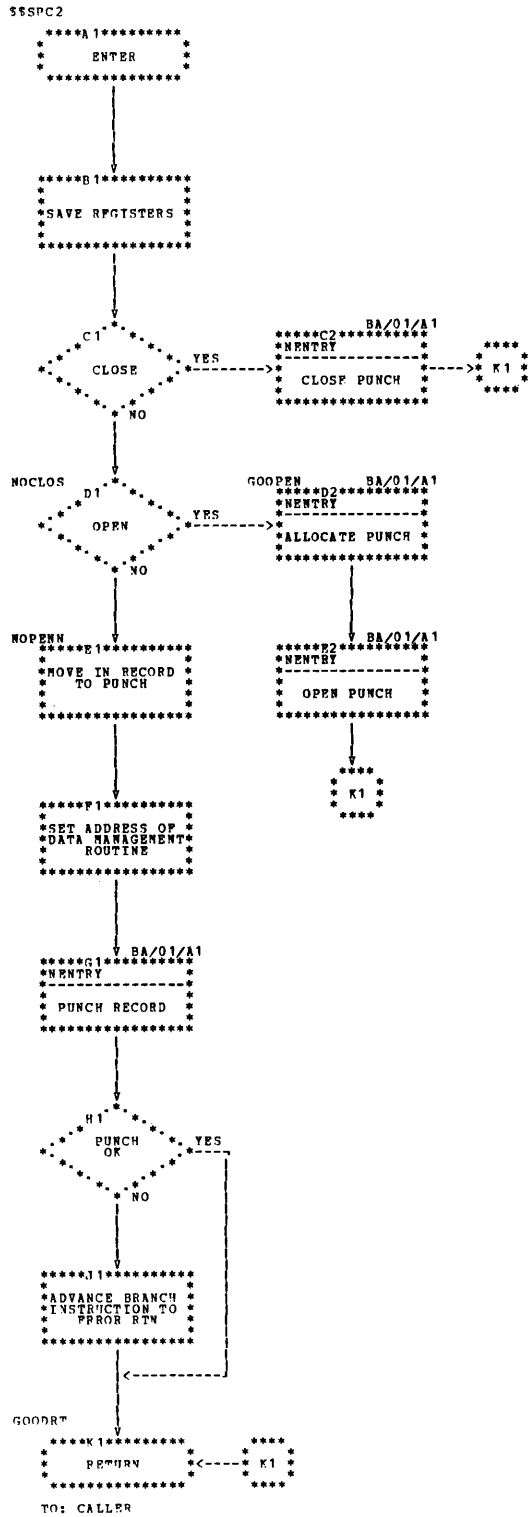


Chart KE. Model 6 System List—Punch (\$\$SPC2)

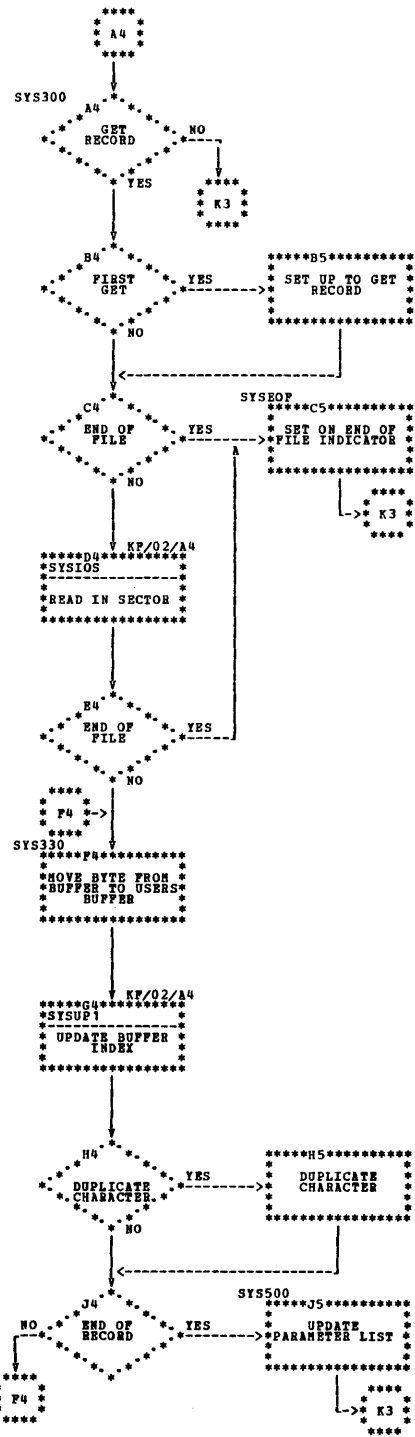
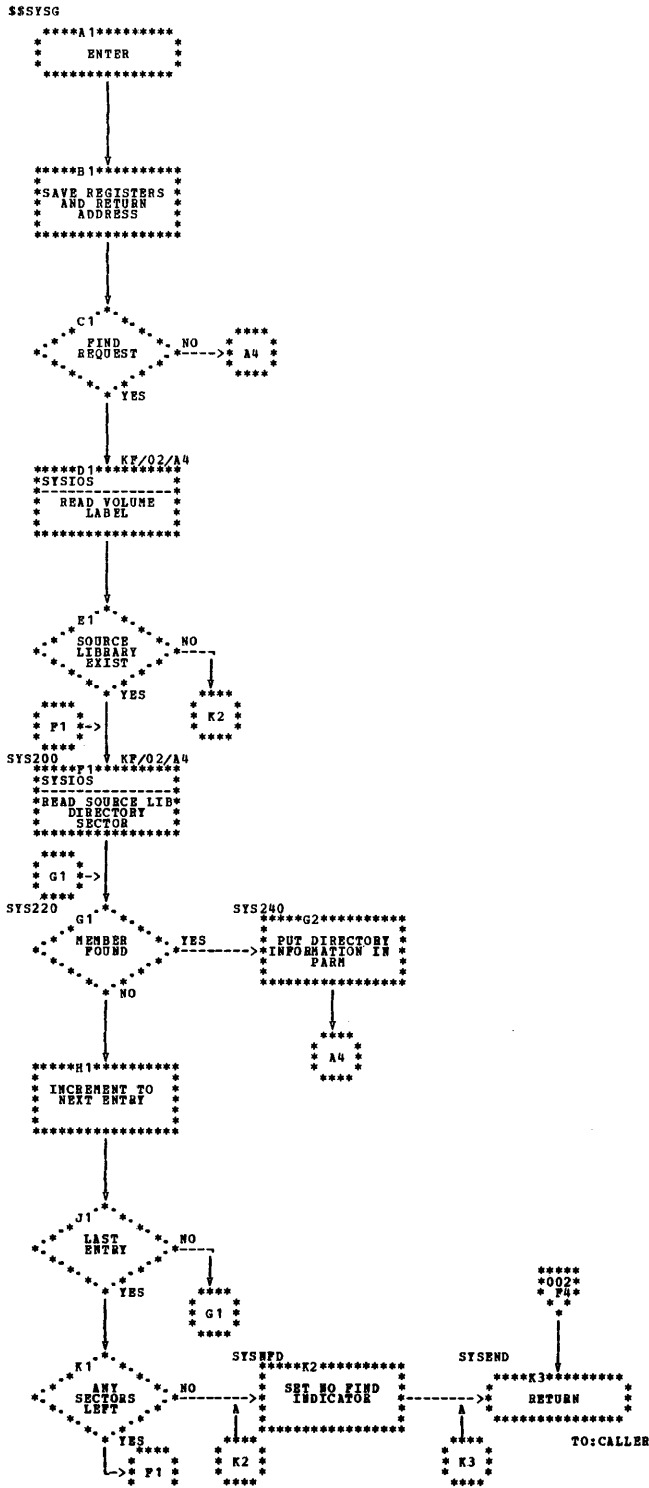
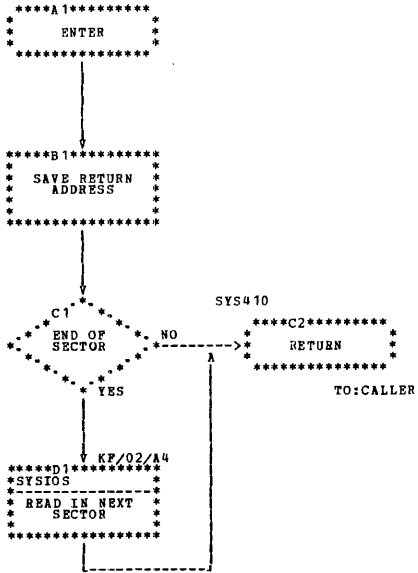


Chart KF (Part 1 of 2). Source Library Get (\$\$SYSG)

SYSUP1



SYSIOS

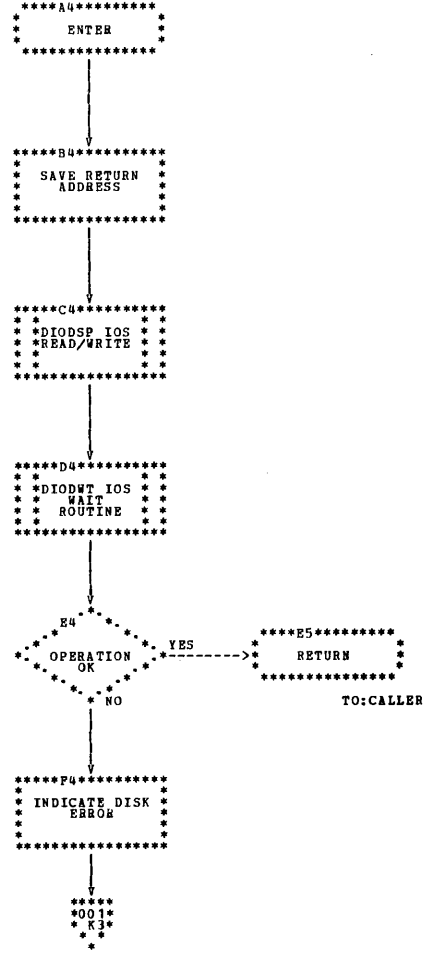


Chart KF (Part 2 of 2). Source Library Get (\$\$SYSG)

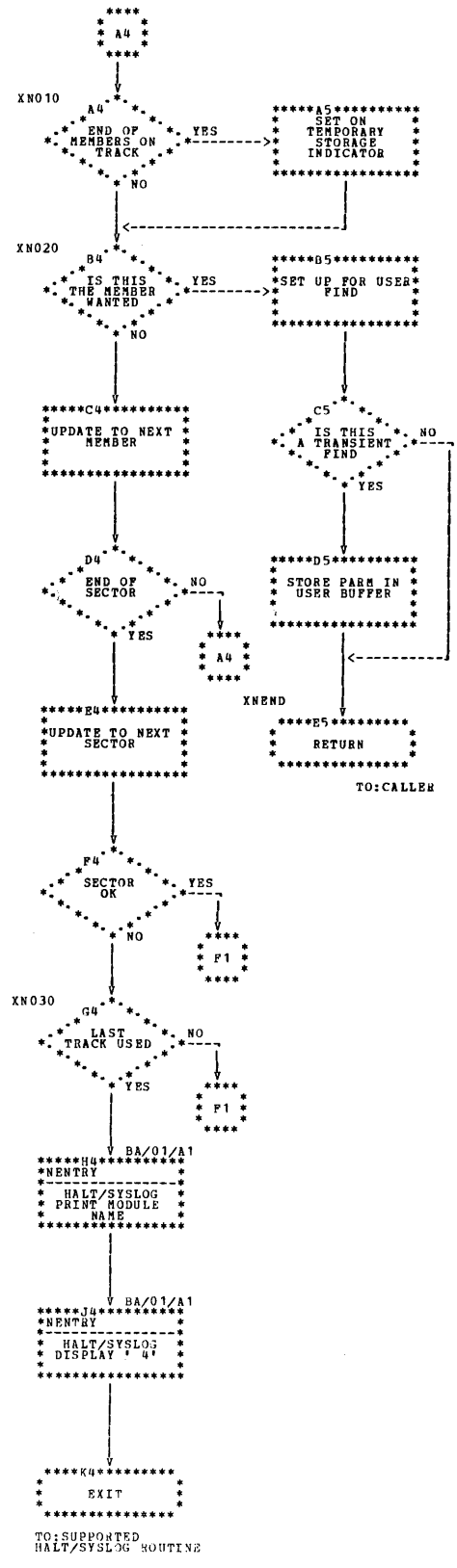
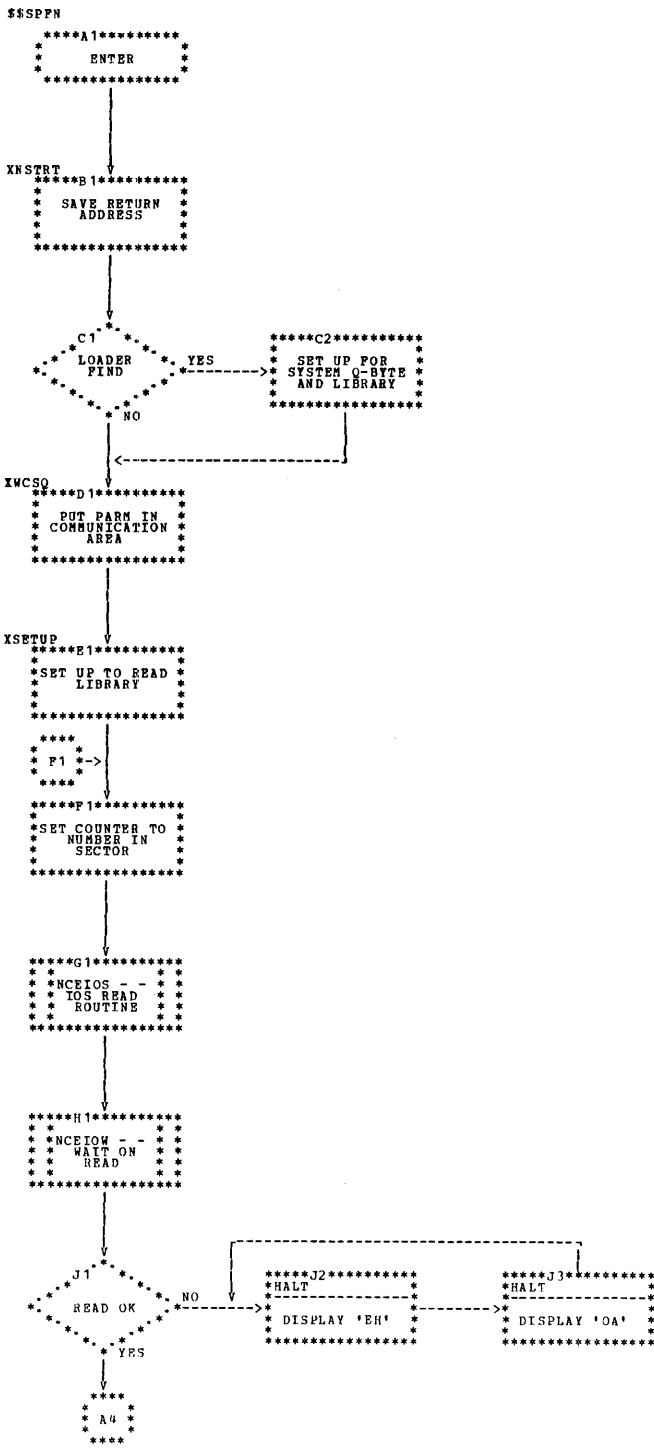


Chart KG. Find Transient (\$\$SPFN)

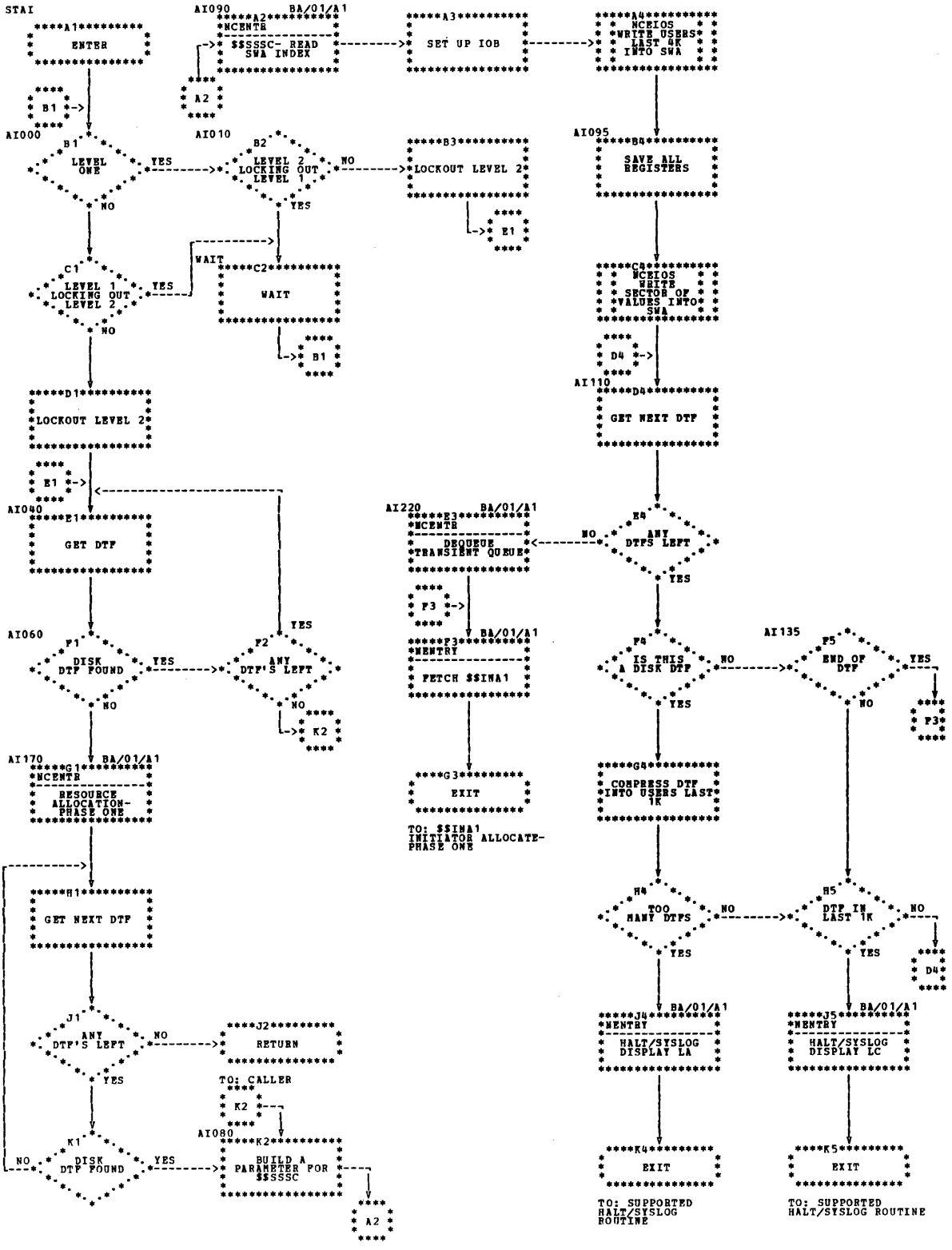


Chart KH. Allocate Initiator (\$\$STAI)

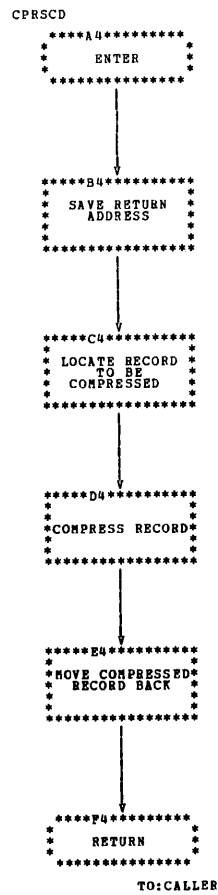
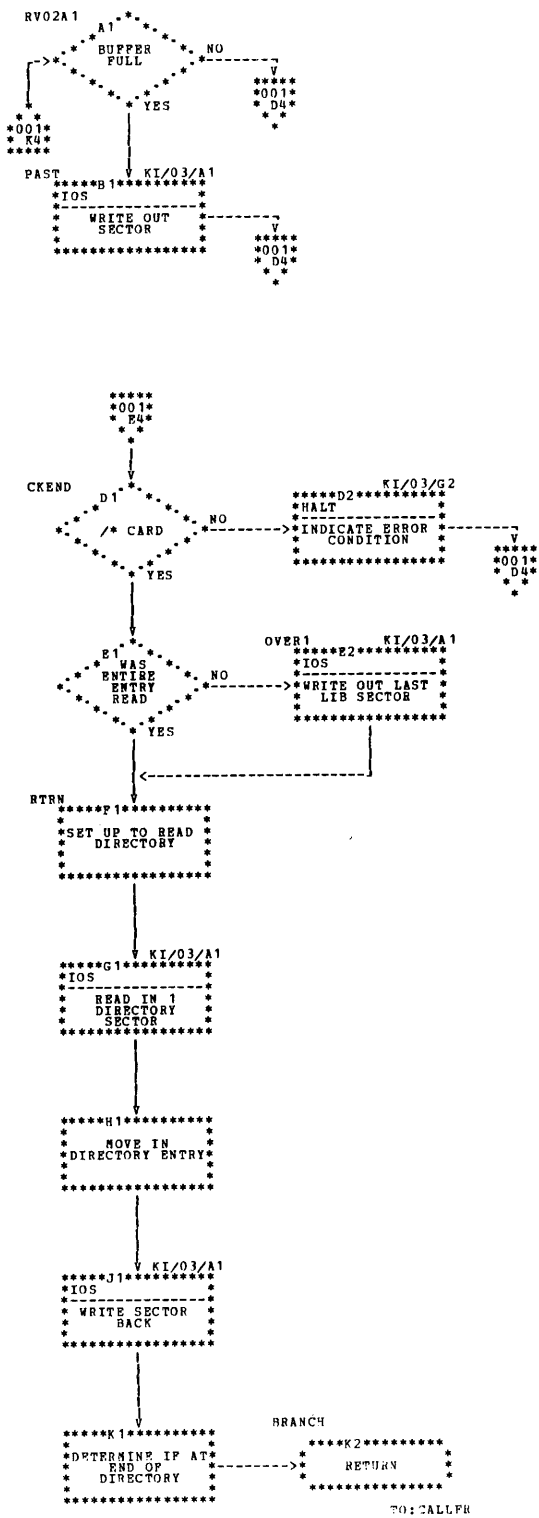


Chart KI (Part 2 of 3). LOAD * Mainline (\$SYLA)

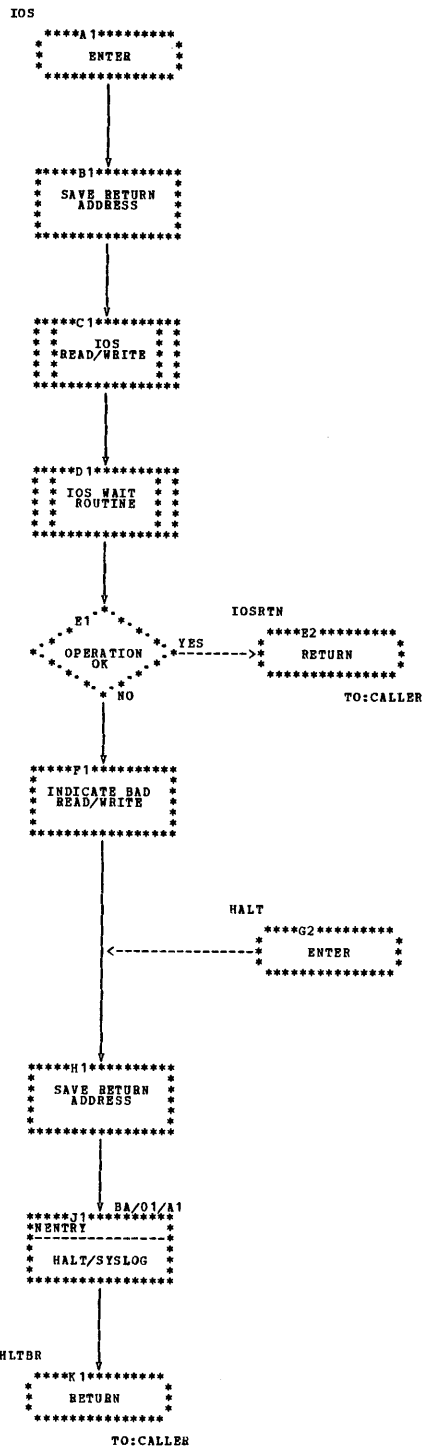


Chart KI (Part 3 of 3). LOAD * Mainline (\$\$SYLA)

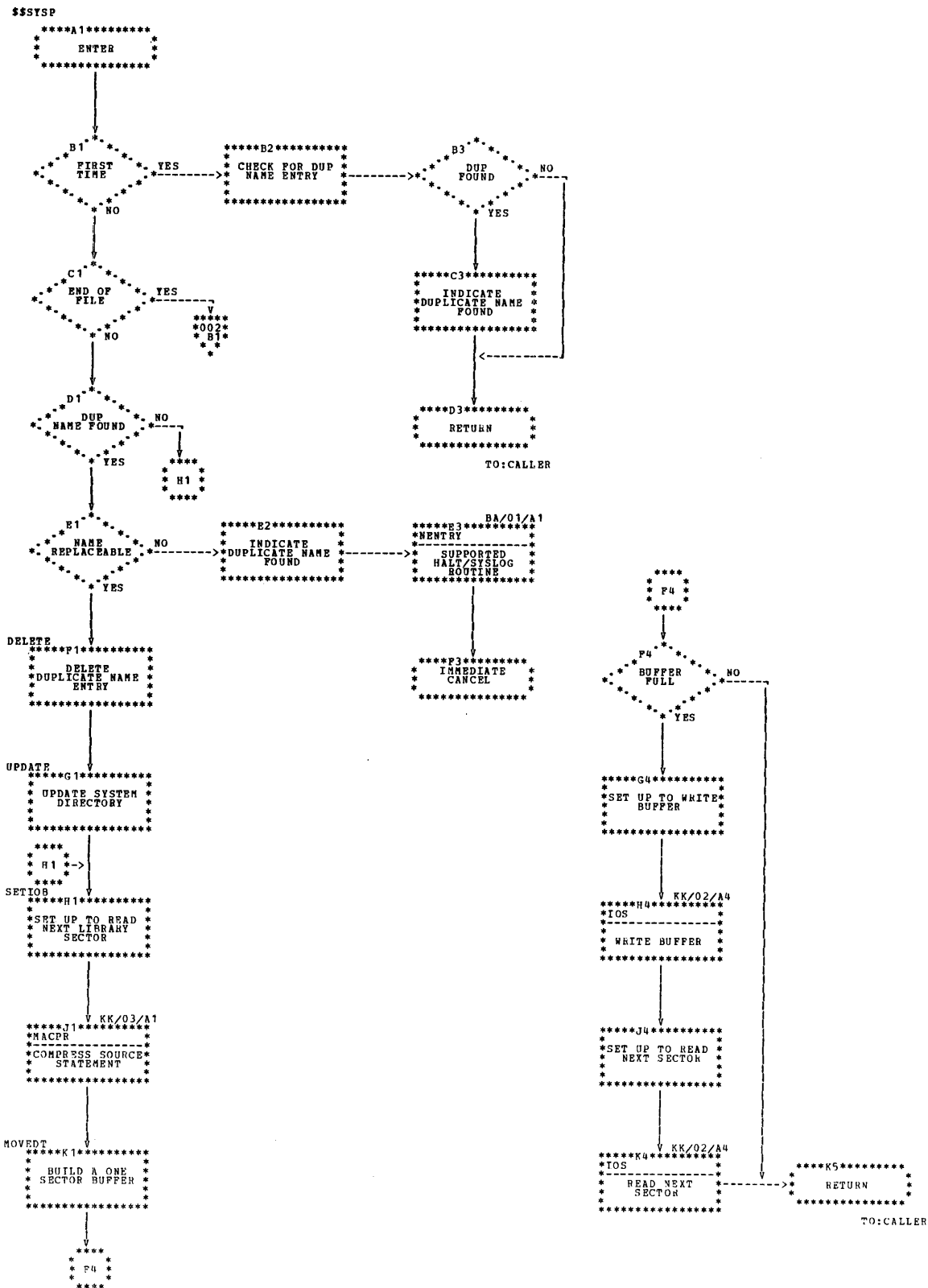


Chart KK (Part 1 of 3). Copy Main Storage to Source Library (\$SYSP)

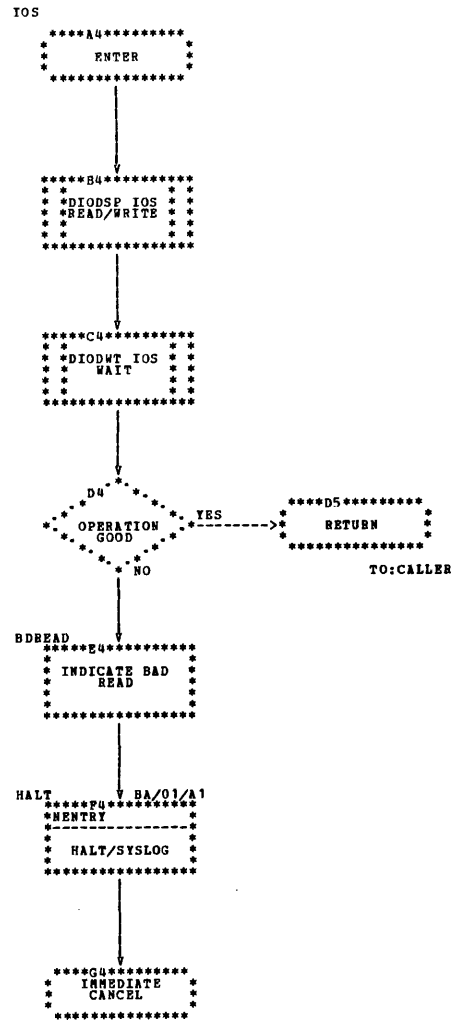
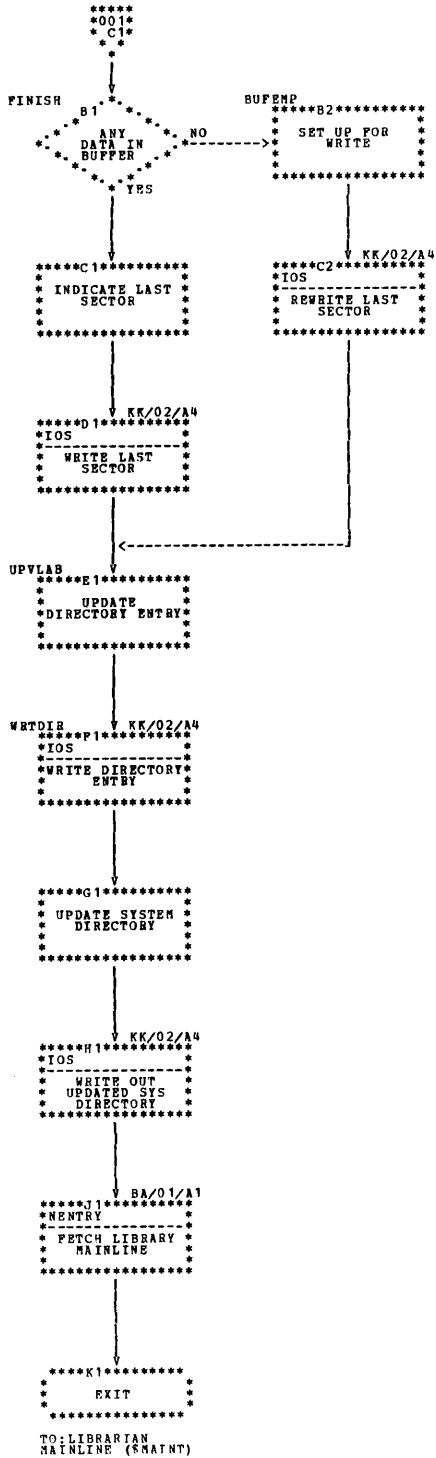


Chart KK (Part 2 of 3). Copy Main Storage to Source Library (\$\$SYSP)

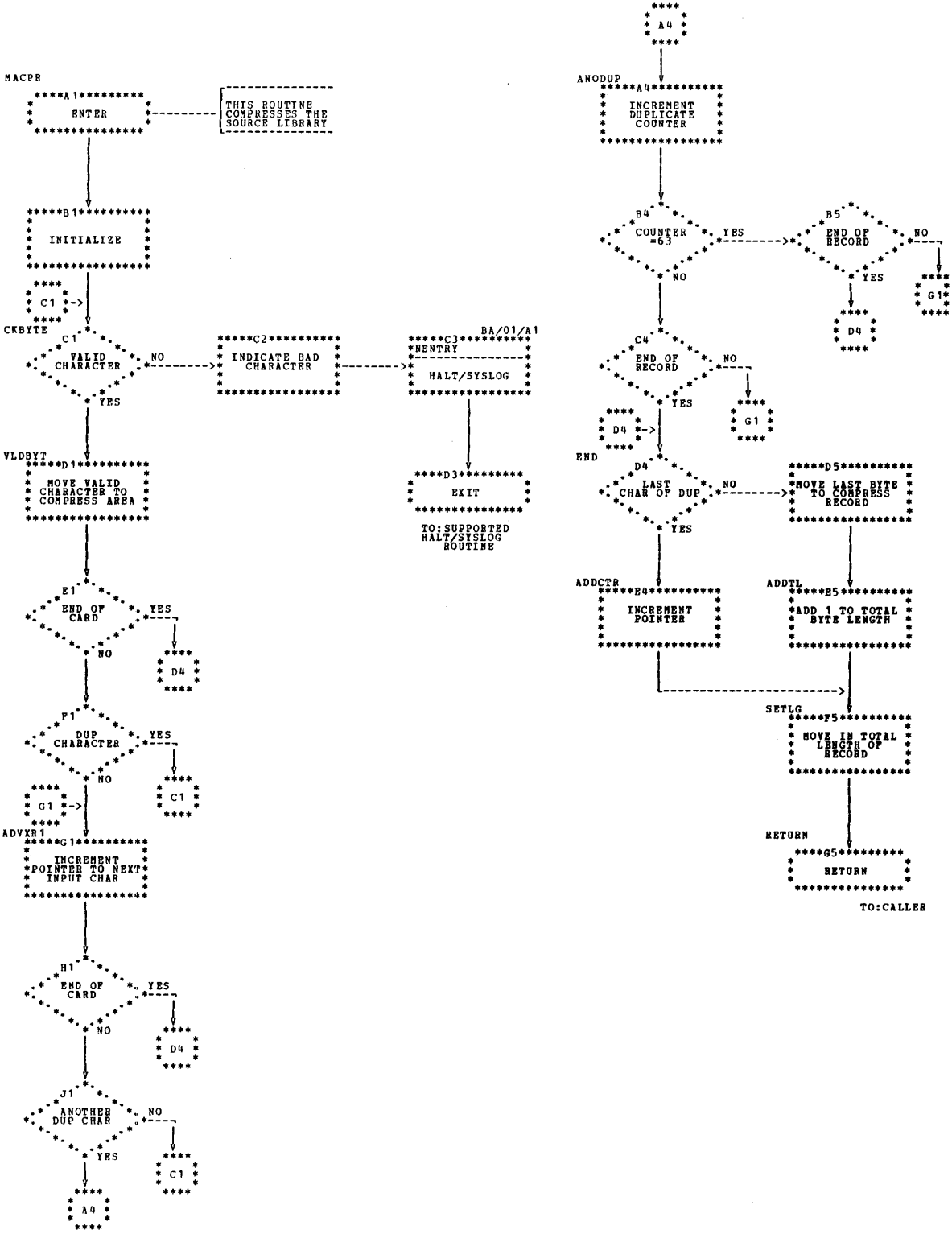


Chart KK (Part 3 of 3). Copy Main Storage to Source Library (\$SYSP)

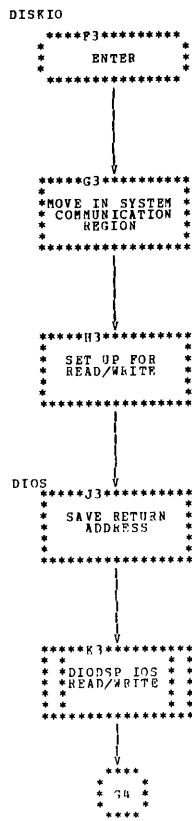
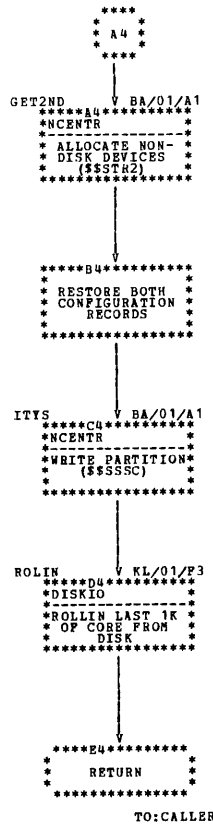
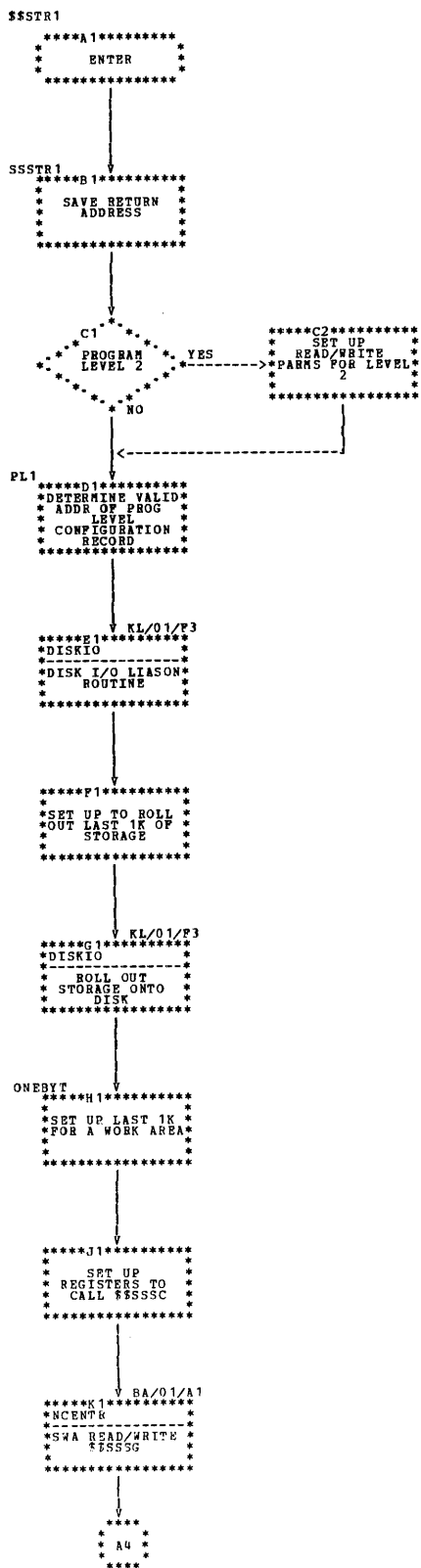


Chart KL. Resource Allocation—Phase One (\$\$STR1)

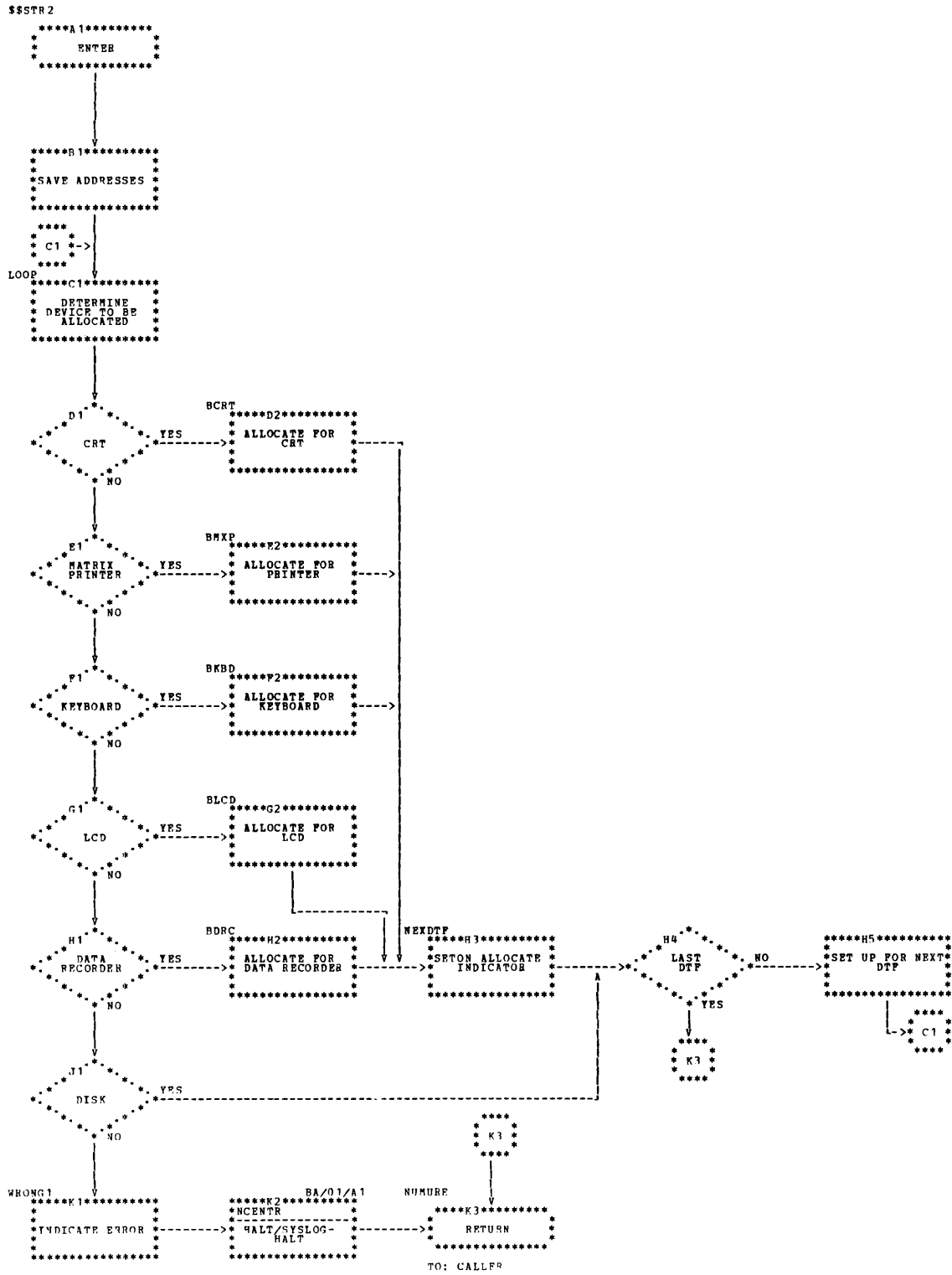


Chart KN. Resource Allocation—Phase Two, Model 6 (\$\$STR2)

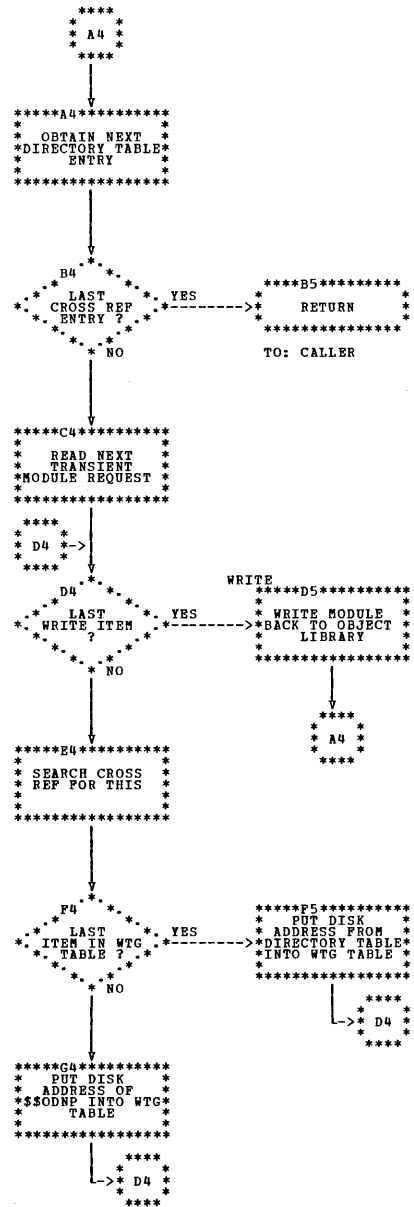
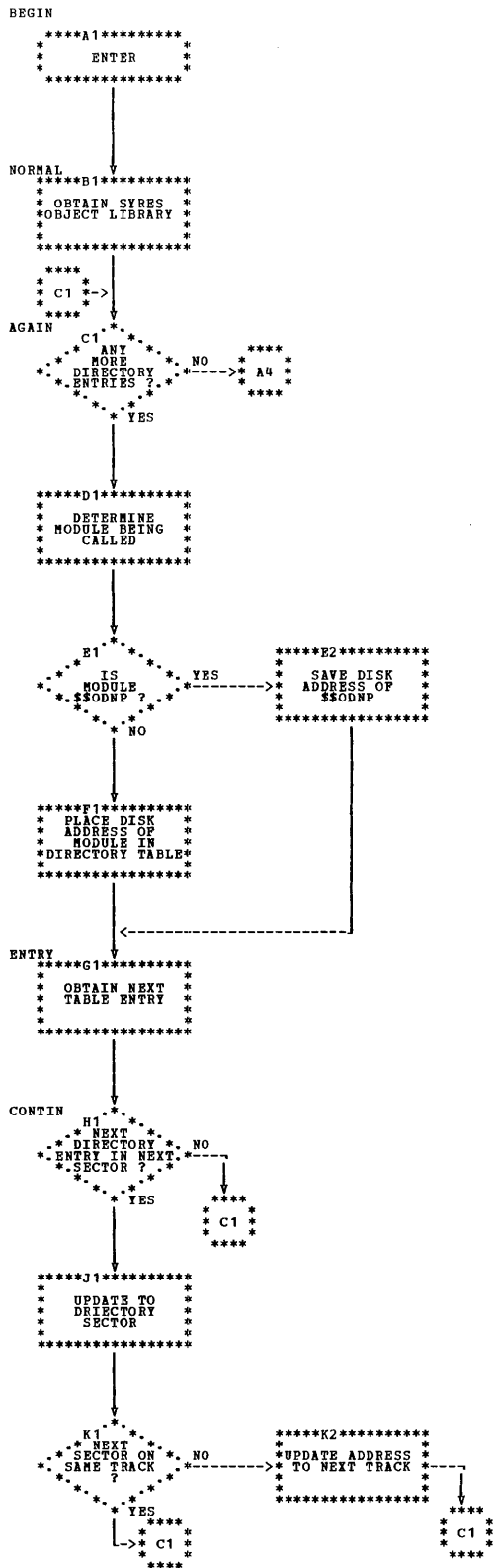


Chart KO. Transient Resolver (\$\$OXRF)

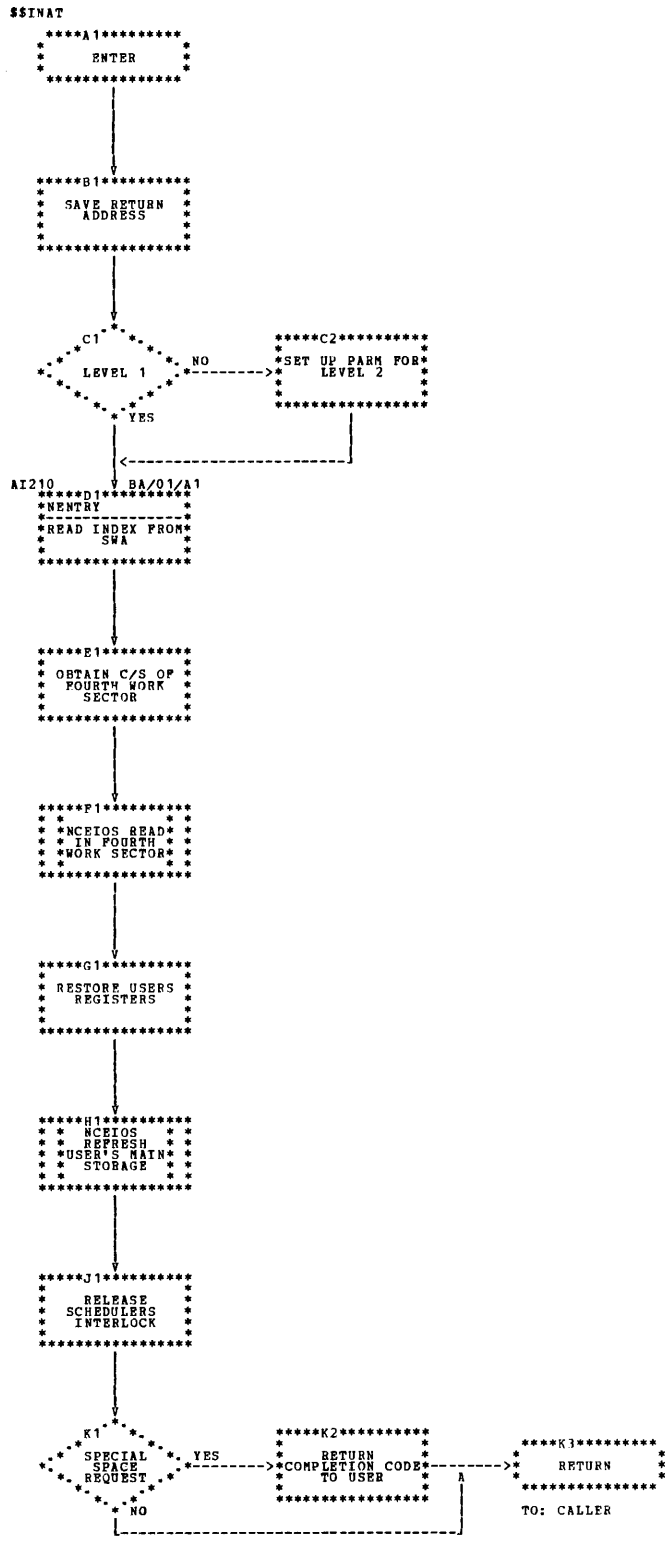


Chart KP. Allocate Terminator (\$\$INAT)

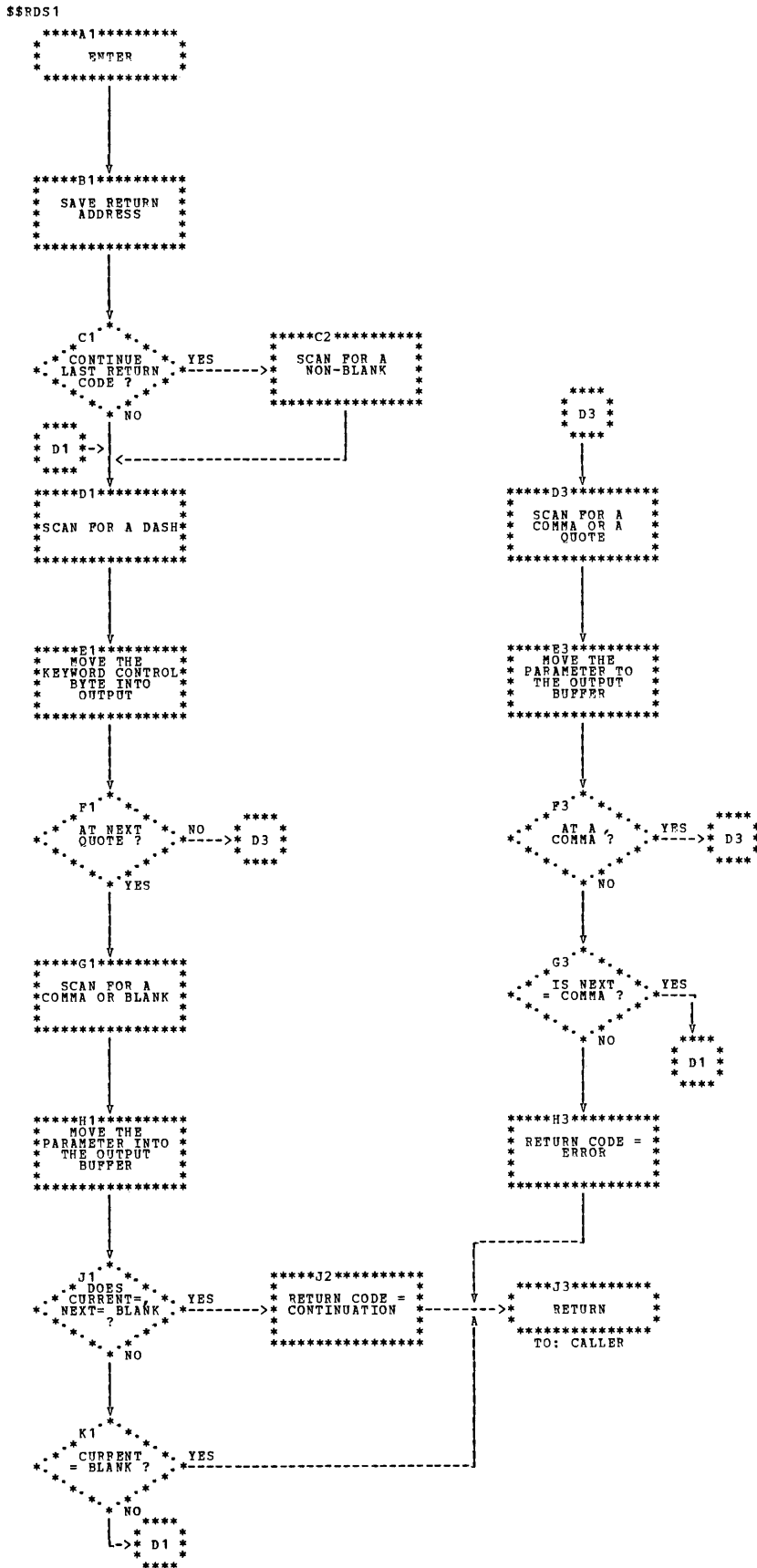


Chart KR. Keyword Syntax Scan Routine (\$\$RDS1)

\$\$\$TF7

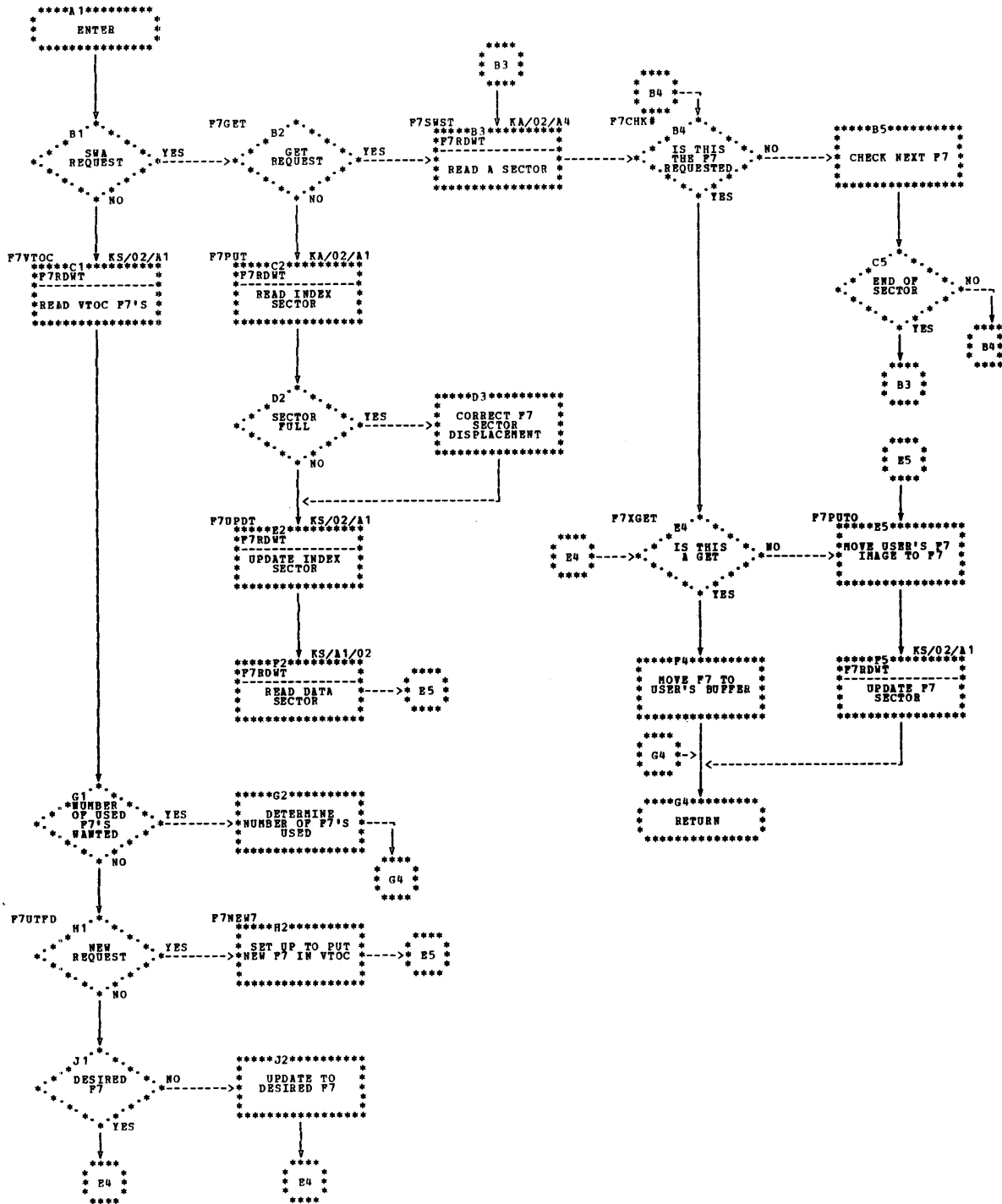
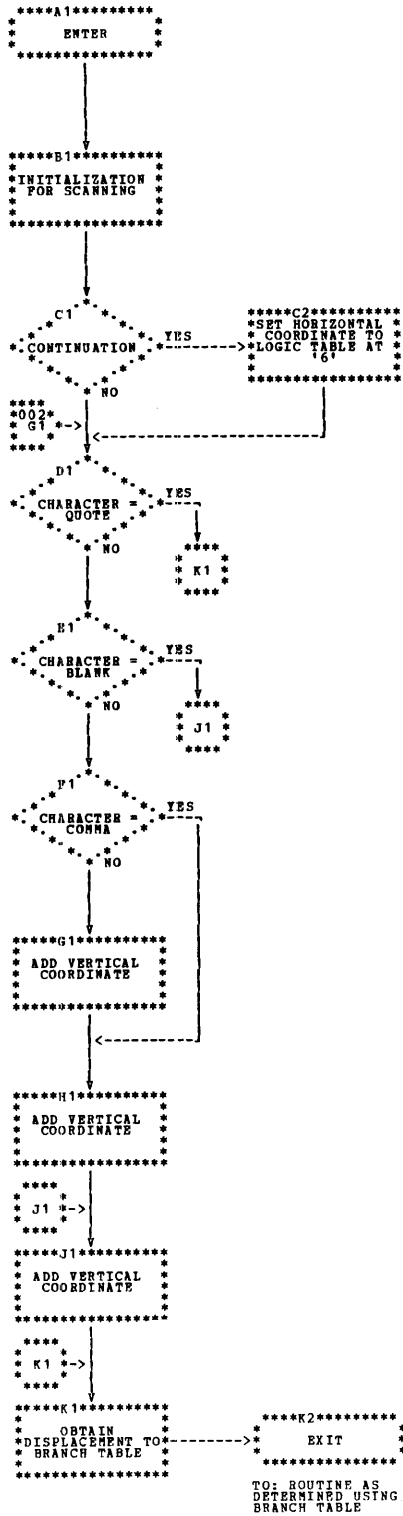


Chart KS (Part 1 of 2). HIKEY Transient (\$\$STF7)

\$\$\$RDHK



BRANCH TABLE

	1	2	3	4	5	6
Quote	BHK 230 2	BHK 120 4	BHK 120 4	BHK 110 3	BHK 120 4	BHK 120 4
Blank	BHK 150 F	BHK 110 3	BHK 110 3	BHK 140 F	BHK 160 3	BHK 110 3
Comma	BHK 220 F	BHK 150 F	BHK 130 5	BHK 140 F	BHK 150 F	BHK 150 F
Other	BHK 150 F	BHK 110 3	BHK 110 3	BHK 150 F	BHK 110 3	BHK 110 3

Values in this portion of the squares provide the displacement to the branch table. It identifies the internal subroutines.

Values in this portion of the squares will be used as the horizontal coordinate next time through.

Chart KT (Part 1 of 2). HIKEY Parameter Scan Routine (\$\$RDHK)

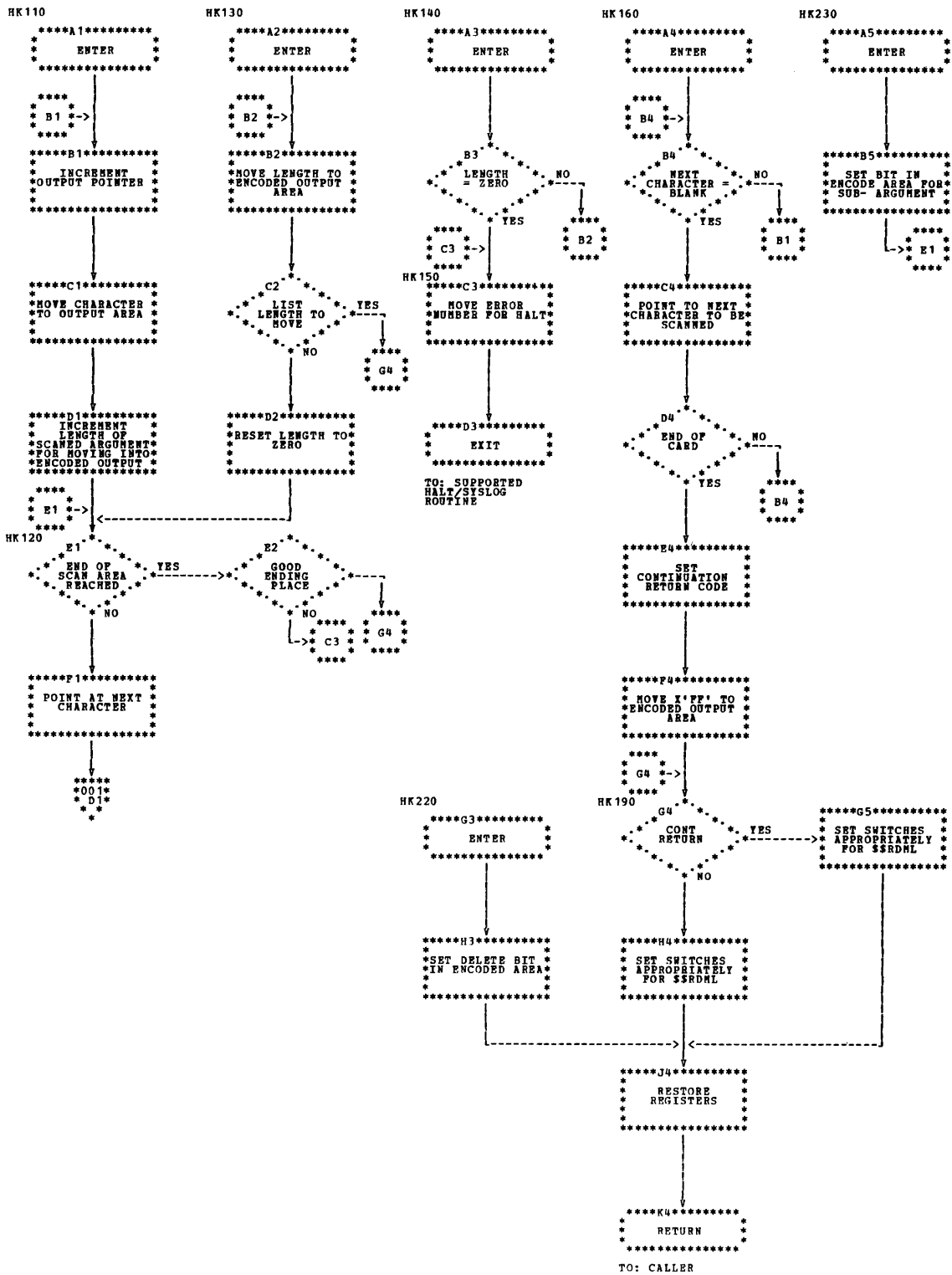


Chart KT (Part 2 of 2). HIKEY Parameter Scan Routine (\$\$RDHK)

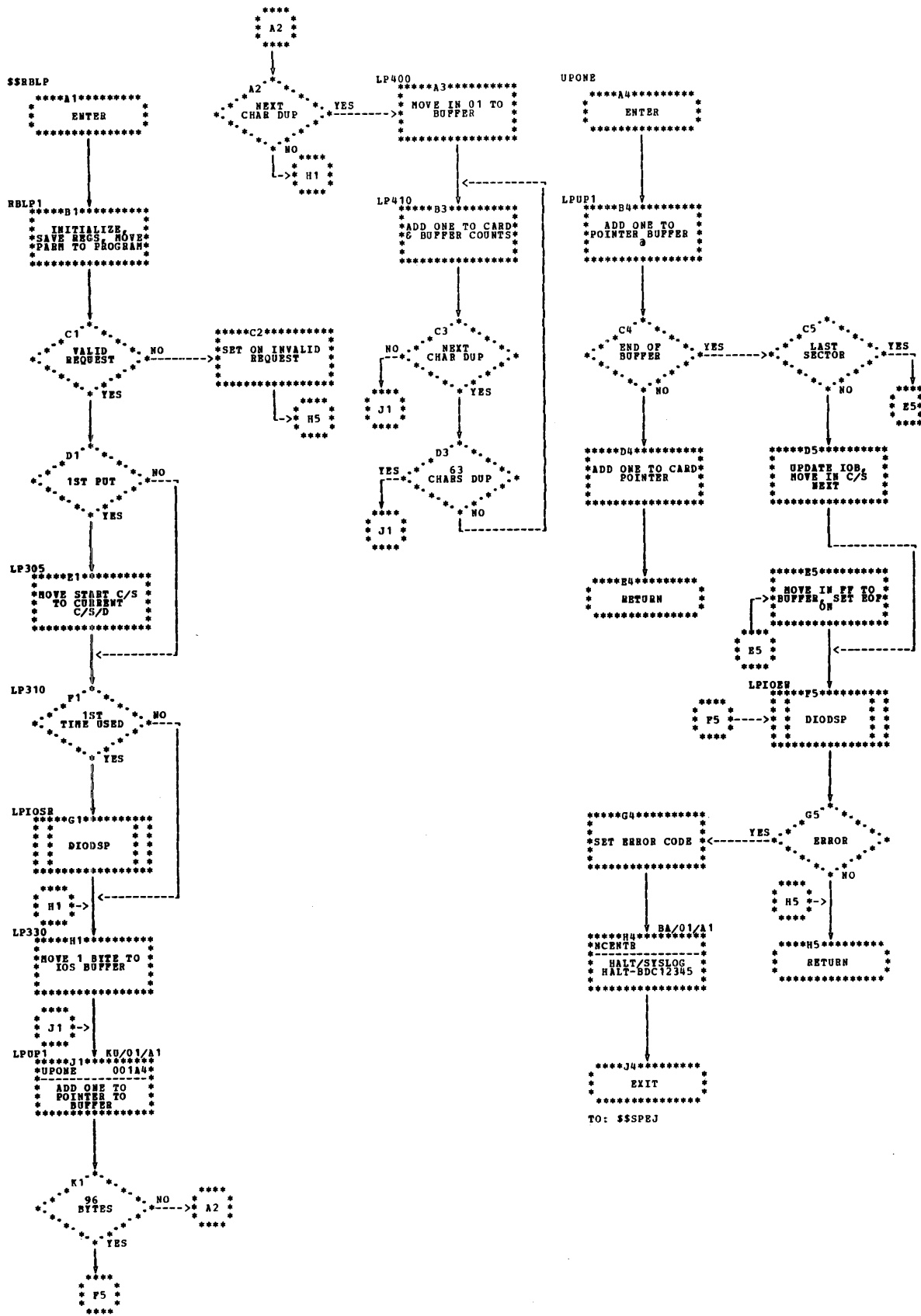


Chart KU. Model 6 Logical Put Routine (\$\$RBLP)

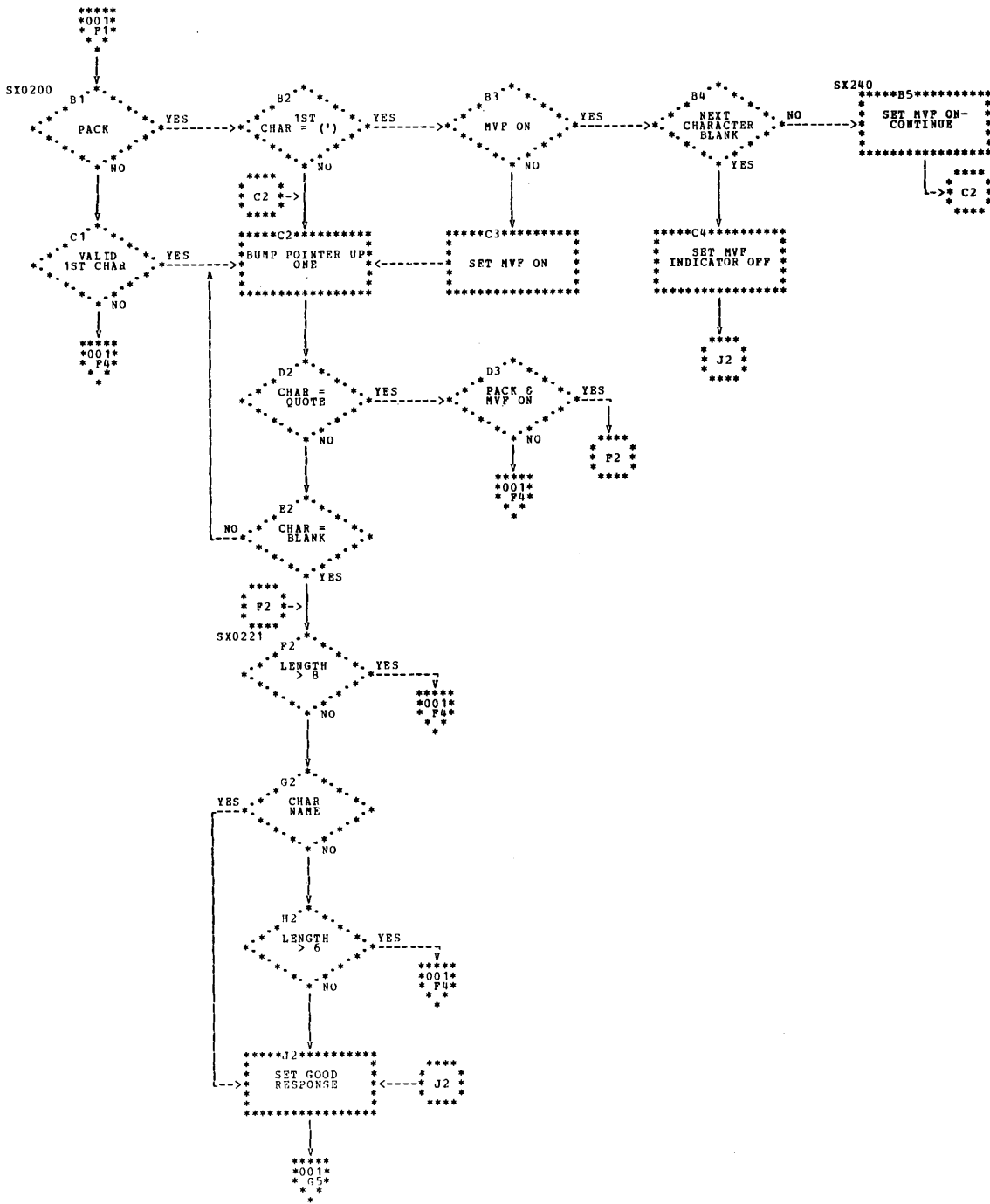


Chart KV (Part 2 of 4). Model 6 Syntax Checker (\$RBSX)

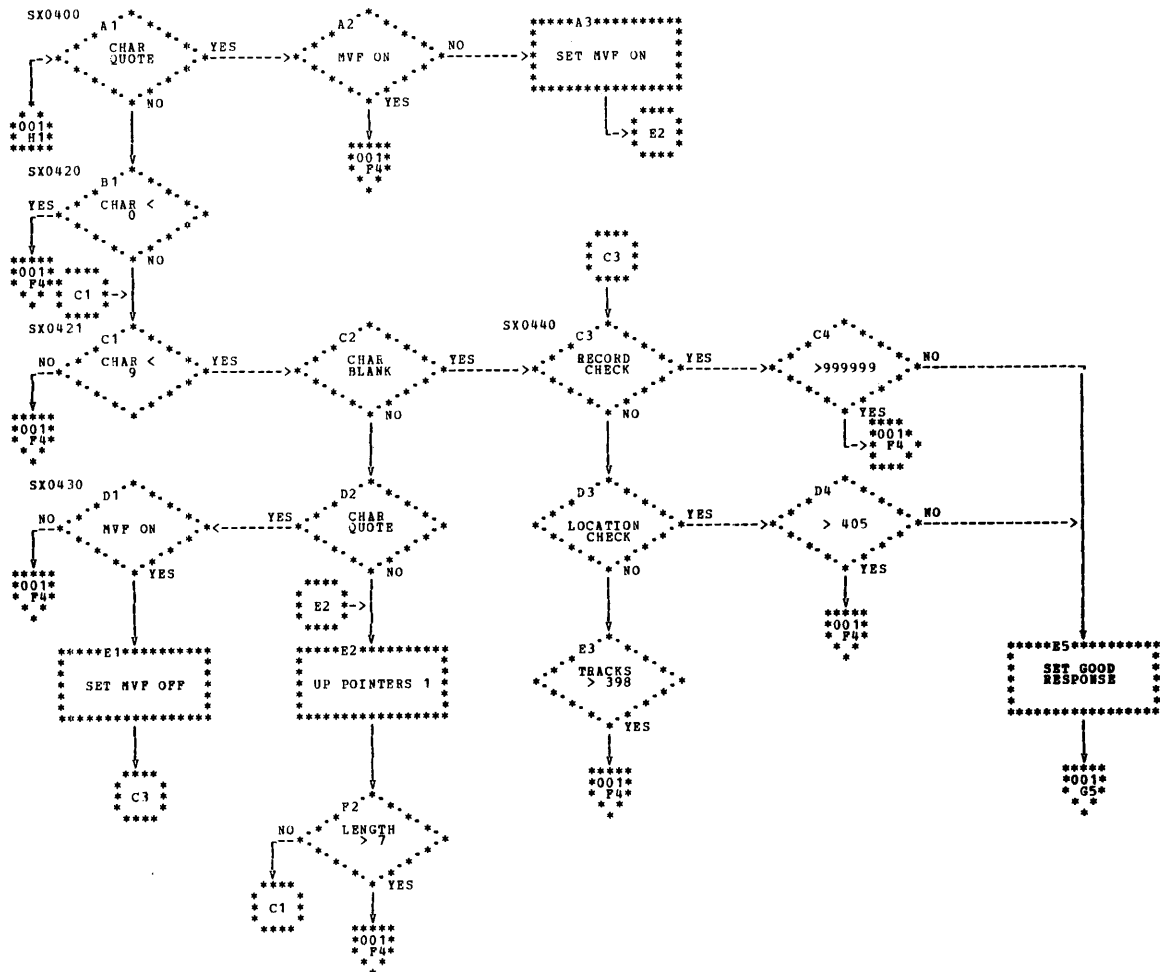


Chart KV (Part 4 of 4). Model 6 Syntax Checker (SXRBSX)

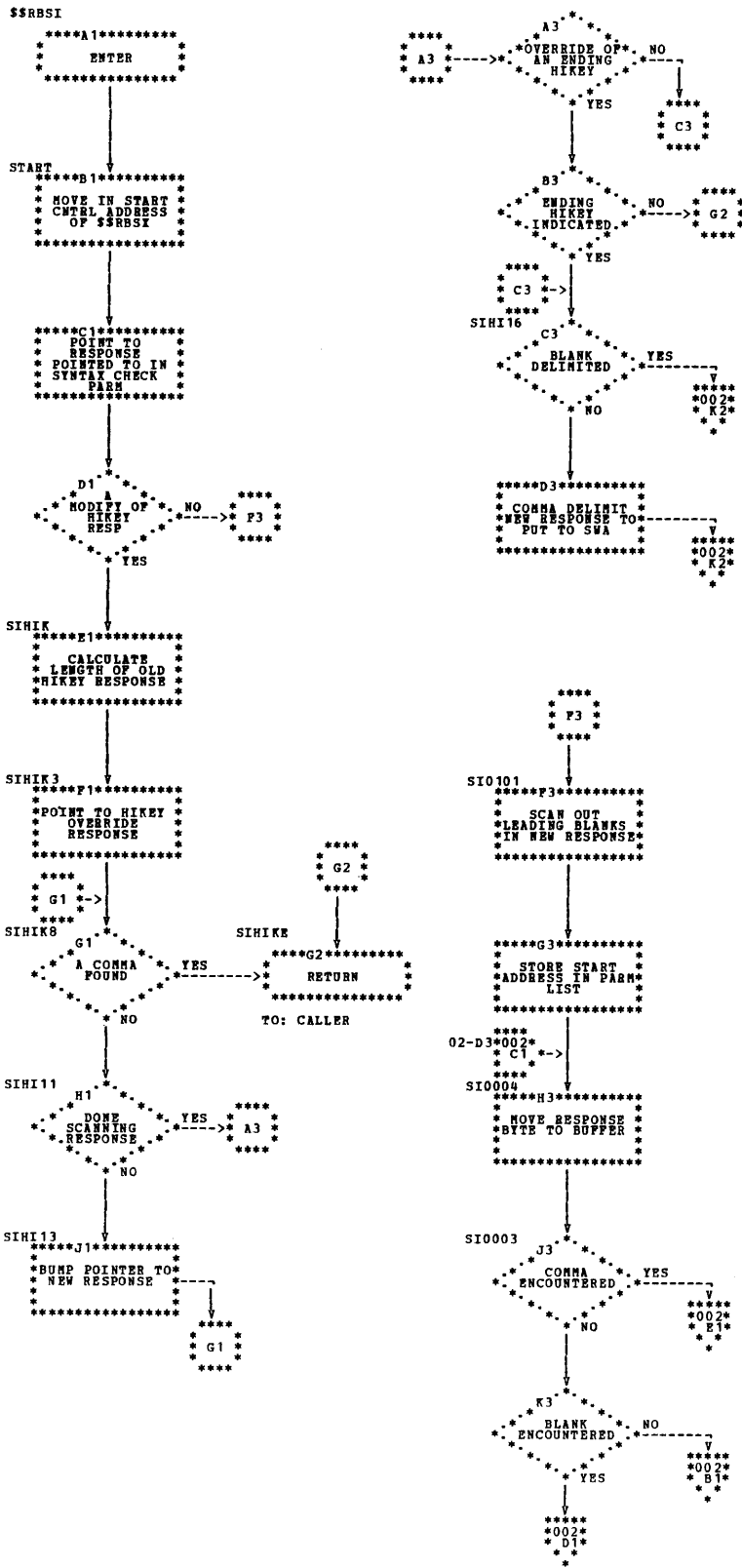


Chart KW (Part 1 of 2). Model 6 Syntax Checker Interface (\$\$RBSI)

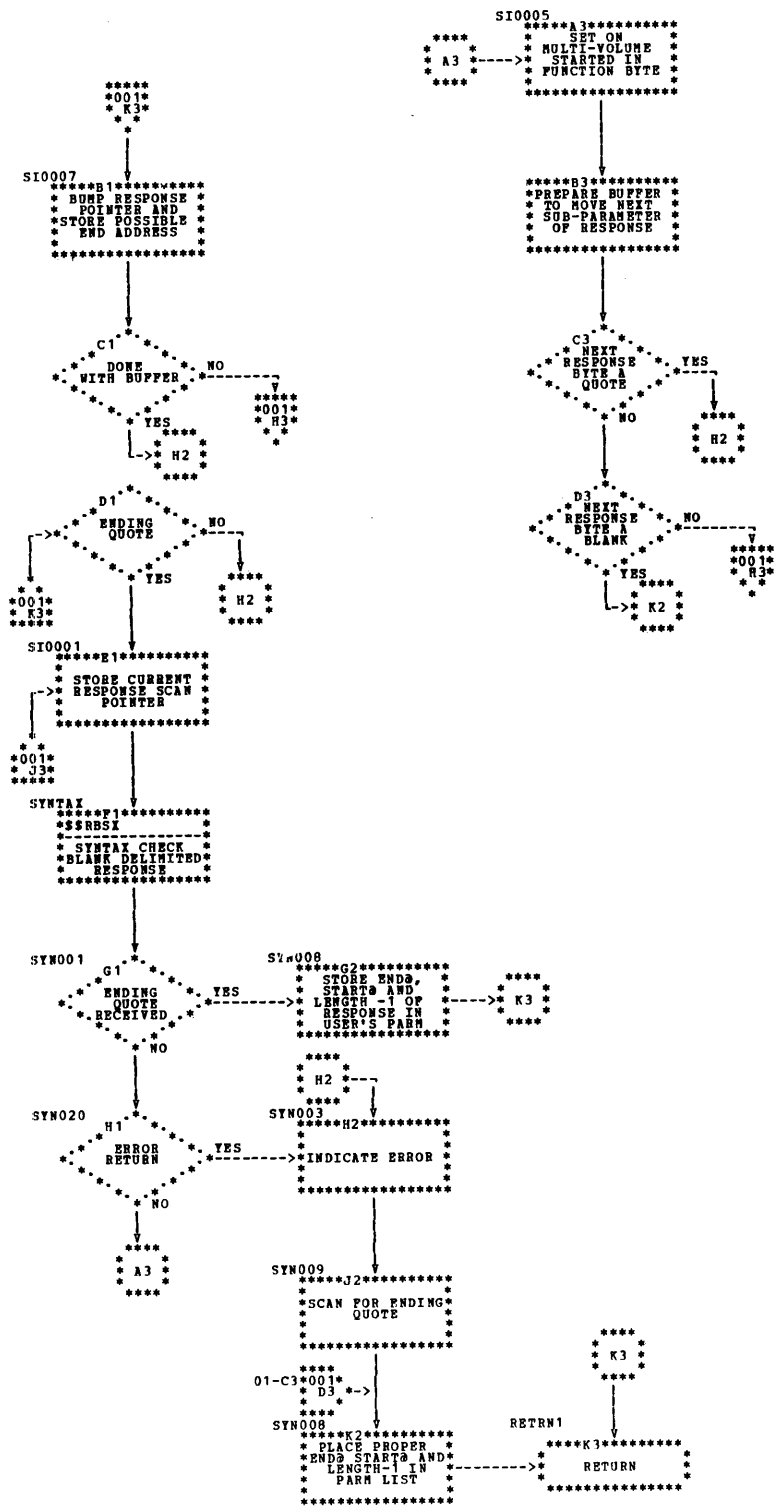


Chart KW (Part 2 of 2). Model 6 Syntax Checker Interface (\$RBSI)

Figure 6-23 contains the Directory for the Transient and Scheduler Support routines.

Phase Name	Chart ID	Descriptive Name	Function
\$\$STIC	JA	System Input Device—Console	<ul style="list-style-type: none"> ● Reads a record from the console (5471)
\$\$STIM	JB	System Input Device—MFCU/1442	<ul style="list-style-type: none"> ● Reads a record from the MFCU or the 1442
\$\$STIP/ \$\$STIS	JC	Model 6 System Input Device— Keyboard/Printer	<ul style="list-style-type: none"> ● Supports the keyboard/printer as a sysin device
\$\$STID	JD	Model 6 System Input Device— Data Recorder	<ul style="list-style-type: none"> ● Supports the data recorder as a sysin device
\$\$STIT/ \$\$STIX	JE	Model 6 System Input Device— CRT	<ul style="list-style-type: none"> ● Supports CRT displays
\$\$STEP	JF	System Output—Printer Error Logging	<ul style="list-style-type: none"> ● Calls the IOS Error Logging routine (\$\$DLOG) ● Issues a descriptive halt
\$\$STOC	JG	Halt/Syslog—Console (5471)	<ul style="list-style-type: none"> ● Supports the console as a system logging device
\$\$STOP	JH	Halt/Syslog—Printer	<ul style="list-style-type: none"> ● Supports the printer as a system logging device
\$\$STOH	JI	System Halt Transient	<ul style="list-style-type: none"> ● Converts the halt codes into valid stick light combinations ● Checks for a DPF system ● Passes control to the Resident Halt routine (NPHALT) if in a DPF system
\$\$STP6	JJ	Model 6 Printer Syslog Error Recovery Procedures Routine	<ul style="list-style-type: none"> ● Senses printer errors and attempts recovery
\$\$STOK	JK	Halt/Syslog—Halt Code Conversion Routine	<ul style="list-style-type: none"> ● Converts Disk System halt characters to Model 6 halt characters and Model 6 halt codes
\$\$STOS	JL	Model 6 Halt/Syslog Routine— Keyboard	<ul style="list-style-type: none"> ● Issues HPL instructions and accepts options from keyboard
\$\$STOM	JM	Model 6 Halt/Syslog Routine— Matrix Printer	<ul style="list-style-type: none"> ● Performs halt/syslog functions for the matrix printer
\$\$STON	JN	Model 6 Halt/Syslog Routine— Matrix Printer	<ul style="list-style-type: none"> ● Performs halt/syslog functions for the matrix printer
\$\$STOT	JO	Model 6 Halt/Syslog Routine—CRT	<ul style="list-style-type: none"> ● Performs halt/syslog functions using the CRT for display
\$\$STOX	JP	Model 6 Halt/Syslog Routine— CRT On and Off	<ul style="list-style-type: none"> ● Starts and stops the CRT display
\$\$SLSE/ \$\$SLS1	JQ/ JR	Model 6 Halt/Syslog Scheduler Error Messages Routine	<ul style="list-style-type: none"> ● Sets up halt/syslog to display all conversational scheduler error messages
\$\$SSGT	JS	Scheduler Work Area—GET	<ul style="list-style-type: none"> ● Retrieves the next consecutive logical record from the scheduler work area

Figure 6-23. Transient Directory (Part 1 of 3)

Phase Name	Chart ID	Descriptive Name	Function
\$\$\$SPT	JT	Scheduler Work Area—PUT	<ul style="list-style-type: none"> ● Loads consecutive records into the scheduler work area
\$\$\$SSC	JU	Scheduler Work Area—Read/Write	<ul style="list-style-type: none"> ● Reads consecutive sectors from the SWA ● Writes consecutive sectors into the SWA
\$\$\$SVT	JV	Volume Table of Contents Read/Write	<ul style="list-style-type: none"> ● Performs read/write functions for the following data areas: <ol style="list-style-type: none"> 1. Volume label 2. VTOC index 3. Configuration record 4. Format one records ● Restores the scheduler work area
\$\$STRI	JW	Rollin—Phase One	
\$\$STRN	JX	Rollin—Phase Two	<ul style="list-style-type: none"> ● Restores the interrupted RPG program and data areas ● Determines if the correct disk was remounted ● Passes control to the restored RPG program
\$\$STRO	JY	Rollout—Phase One	<ul style="list-style-type: none"> ● Tests the allocated devices ● Enqueues IOB entries ● Rolls out the RPG program onto disk
\$\$STRU	JZ	Rollout—Phase Two	<ul style="list-style-type: none"> ● Stores the scheduler work area
\$\$STRT	KA	Rollout—Phase Three	<ul style="list-style-type: none"> ● Builds a table entry for the following: <ol style="list-style-type: none"> 1. Print chain image 2. Transient area scheduler 3. Printer page size 4. Scheduler/Data Management Switches 5. Volume labels
\$\$SYP1	KB	Disk System System List—Print	<ul style="list-style-type: none"> ● Allows the user to be printer independent by supporting: <ol style="list-style-type: none"> 1. Open 2. Close 3. Print, skip and/or space 4. Skip and/or space without print
\$\$SYP2	KC	Model 6 System List—Print	<ul style="list-style-type: none"> ● Passes print parameter list to halt/syslog ● Calls halt/syslog
\$\$SPC1	KD	Disk System System List— Print/Punch	<ul style="list-style-type: none"> ● Supports the following functions: <ol style="list-style-type: none"> 1. Punch only 2. Print/punch 96 columns 3. Punch 96 columns and print 128 columns
\$\$SPC2	KE	Model 6 System List—Punch Routine	<ul style="list-style-type: none"> ● Supports the following functions: <ol style="list-style-type: none"> 1. Punch only 2. Print/punch 96 columns 3. Punch 96 columns and print 128 columns
\$\$SYSG	KF	Source Library Get	<ul style="list-style-type: none"> ● Finds a specified member of the source library ● Passes a 96-byte record back to the caller
\$\$SPFN	KG	Find Transient	<ul style="list-style-type: none"> ● Locates a specific program in the object library ● Passes loading information to the caller ● Issues a halt if entry is not found
\$\$STAI	KH	Allocate Initiator	<ul style="list-style-type: none"> ● Loads the Allocate routine (\$\$INAI) ● Compresses specified DTFs into the last 1024 bytes of the user's RPG program ● Copies the last 4K of the user's program into SWA

Figure 6-23. Transient Directory (Part 2 of 3)

Phase Name	Chart ID	Descriptive Name	Function
\$\$SYLA	KI	Load* Mainline	<ul style="list-style-type: none"> ● Builds a temporary object library ● Dynamically allocates needed space for the above library ● Updates the volume label if space was allocated
\$\$SPEJ	none	End of Job Transient	<ul style="list-style-type: none"> ● Terminates the calling routine
\$\$TMST	KJ	End of Job—Disk Status	<ul style="list-style-type: none"> ● Determines the mask to be used for the input Q-byte ● Calls the Disk IOS Error Logging routine and the End of Job phases one and two
\$\$SYSP	KK	Copy Main Storage to Source Library	<ul style="list-style-type: none"> ● Reads 96-byte records from main storage ● Compresses and loads the records into the source library ● Updates the volume label and the library directory
\$\$STR1	KL	Resource Allocation Phase One	<ul style="list-style-type: none"> ● Reads the temporary configuration records from the SWA ● Calls \$\$STR2 to allocate the requested devices ● Restores the updated configuration records
\$\$STR2	KM, KN	Resource Allocation Phase Two	<ul style="list-style-type: none"> ● Allocates non-disk devices
\$\$OXRF	KO	Transient Resolver	<ul style="list-style-type: none"> ● Resolves absolute disk addresses among certain system transients in order to facilitate the passing of control from one to another
\$\$INAT	KP	Allocate Terminator	<ul style="list-style-type: none"> ● Sets up the user's main storage to reflect the settings before the Allocation routines were called
\$\$STNQ	none	System Queue Routine	<ul style="list-style-type: none"> ● Removes transients that are interlocked with the scheduler and are located on the transient queue
\$\$STHB	KQ	Halt/Syslog Parameter Build Routine	<ul style="list-style-type: none"> ● Builds and loads the appropriate Halt/Syslog parameter list into the Reader/Interpreter Work Area (RDIWA).
\$\$RDS1	KR	Keyword Syntax Scan Routine	<ul style="list-style-type: none"> ● Scans a specified area for keywords ● Ensures that each keyword is syntactically correct ● Passes a return code to the caller indicating the result of the scan
\$\$ODNP	none	Transient Resolver No-op Routine	<ul style="list-style-type: none"> ● Issues a halt (42) if a system transient (requested by another system transient) cannot be located by the Transient Resolver (\$\$OXRF)
\$\$STF7	KS	HIKEY Transient	<ul style="list-style-type: none"> ● Reads and writes format seven's in the SWA and VTOC
\$\$RDHK	KT	HIKEY Parameter Scan Routine	<ul style="list-style-type: none"> ● Scans a specified area containing the HIKEY parameters
\$\$RBLP	KU	Model 6 Logical Put Routine	<ul style="list-style-type: none"> ● Compresses records ● Writes compressed records to disk
\$\$RBSX	KV	Model 6 Syntax Checker	<ul style="list-style-type: none"> ● Checks syntax for Load, Build, and Call cycles
\$\$RBSI	KW	Model 6 Syntax Checker Interface	<ul style="list-style-type: none"> ● Serves as the interface between \$\$RBSX and the calling routine when MVFs are being processed
\$\$RBLI	KX	Model 6 Library Interaction Routine	<ul style="list-style-type: none"> ● Scans each 96-byte record taken from a procedure in the source library for LOAD, or any verb, or a continuation statement ● Returns statement type to caller

Figure 6-23. Transient Directory (Part 3 of 3)



PART 7.

LINKAGE EDITOR

INCLD Control Records

The INCLD record is used to specify an object (R.) module that is to be link edited into the current load (O.) module. The specified object (R.) module resides in the object library on disk, not in the \$WORK data file. The Linkage Editor locates the specified object (R.) module in the object library and incorporates it into the current load (O.) module.

Position

1

#INCLD#	module name
---------	-------------

ENTRY Control Record

The ENTRY record is used to override the language translator-generated END record. The ENTRY record is used to specify the first instruction to be executed in the load (O.) module.

Position

1

#ENTRY#	entry point name
---------	------------------

OPTNS Control Record

The OPTNS record is used to specify one or more of the following attributes for the current load (O.) module.

1. INQRY—indicates that this is an inquiry program
2. INQEVK—indicates that this is an inquiry invoking program
3. DEDIC—indicates that this program must be run in a dedicated system
4. SOURCE—indicates that this program requires source input
5. DFRMNT—indicates to defer pack-not-on-line diagnostics
6. PUNCH—indicates punched output to be placed into a LOAD * deck
7. NOAUTO—indicates do not use the AUTOLINK function
8. PTFAPP—indicates a PTF has been applied to the O. module

Position

1

#OPTNS	[INQRY,INQEVK,DEDIC,SOURCE,DFRMNT,PTFAPP,PUNCH,NOAUTO]
--------	--

SOURCE RECORDS—INPUT TO THE LINKAGE EDITOR

The Linkage Editor receives three types of source input records from the \$WORK data file (Figure 7-2):

1. External symbol list (ESL) input records
2. TEXT-RLD input records
3. END input record

Each input record is 64 bytes in length. These records define the object (R.) modules that are to be link edited together to form the current load (O.) module. The following describes the format and function of each source input record type.

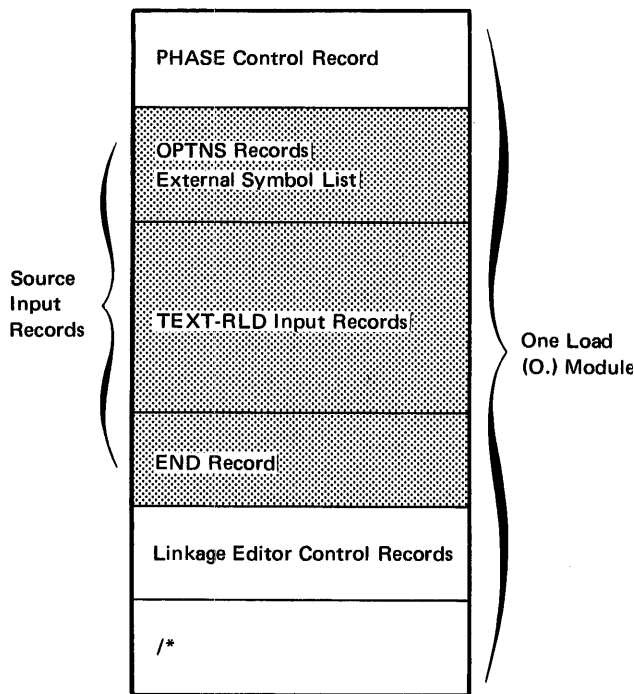
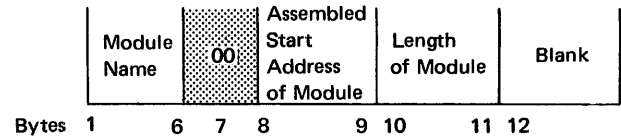


Figure 7-2. Source Input Records Within the Language Translator Output Work File (\$WORK)

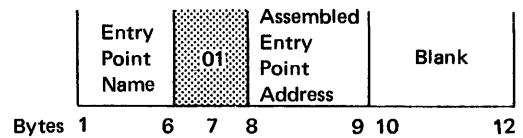
External Symbol List (ESL) Input Records

There are three valid types of ESL entries, each type is 12 bytes in length. Each ESL record may contain from one to five of the following entries:

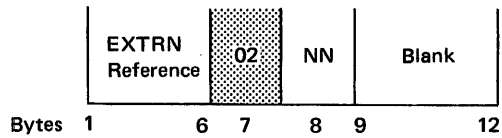
Type X'00'—Module Name: This ESL entry indicates the name assigned to the object (R.) module.



Type X'01'—Entry Point: This ESL entry indicates the name of an entry point within the object (R.) module.



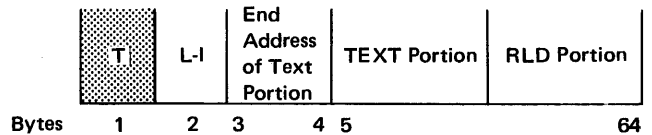
Type X'02'—EXTRN Reference: This ESL entry indicates that the symbol or entry point referenced on this record is defined in another specified module.



NN = Load module number (00 specifies this module)

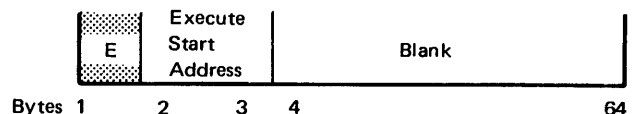
TEXT-RLD Input Records

These input records contain in the TEXT portion the data and instructions that comprise the object (R.) modules. The RLD portion of the same record contains pointers to the relocatable operands located in the TEXT portion of the record.



END Input Record

This record is generated by the language translator and contains the address of the first instruction to be executed in the load (O.) module.



OUTPUT FROM THE LINKAGE EDITOR

The user can expect the following output from the Linkage Editor:

- Updated volume label sector to indicate the next available directory and data locations in the object library
- The object (R.) modules that have been linked to form one or more load (O.) modules
- An entry in the object library directory for every non-RPG overlay module. (See *Appendix A. Data Area Formats* for format of directory.)
- An entry, for every RPG overlay load (O.) module, in the RPG root load overlay fetch table. (See *Appendix A. Data Area Formats* for format of table.)
- Printed output specifying any errors that might have occurred during the Linkage Editor phase.
- A LOAD * deck if punched output is requested
- A listing of the RPG overlay table if the output is an RPG overlay program
- TEXT output records: This portion of the load (O.) module in one continuous string of instructions and data taken from the object (R.) modules. The TEXT records are in machine language without identifiers, length bytes or address bytes (Figure 7-3).
- RLD output records: This portion of the load (O.) module is one continuous string of relocation directory (RLD) entries (Figure 7-3). Each RLD byte indicates the number of TEXT bytes displaced from the preceding RLD-referenced TEXT operand.

There are four types of valid RLD entries:

1. Type X'00'-X'7F'—indicating normal RLD displacement
2. Type X'80'—indicating a 128-byte text area with no relocatable operands
3. Type X'FE'—indicating a null RLD
4. Type X'FF'—indicating the end of the RLDs

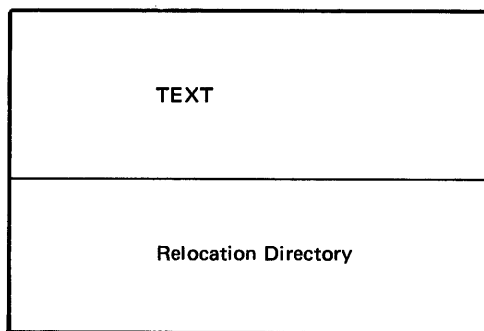


Figure 7-3. Relative Position of the TEXT and RLD Portions of the Object Library

SYSTEM REQUIREMENTS

The Linkage Editor requires the following minimum system configurations:

Disk System:

- IBM 5203 Line Printer, Model A1
- IBM 5410 Processing Unit, Model A13
- IBM 5444 Disk Storage Drive, Model 2

Model 6:

- IBM 5213 Line Printer, Model 1
- IBM 5406 Processing Unit, Model B2 (Includes the keyboard and eight command keys)
- IBM 5444 Disk Storage Drive, Model 2

The Linkage Editor performs the following functions:

1. Links separately assembled or compiled object (R.) modules* together to form one or more load (O.) modules. The physical size of each load (O.) module is determined by the total size of the object (R.) modules that are referenced between two PHASE control records or a PHASE control record and the /* control record. The control records are located in the language translator output work area (\$WORK).
2. Builds an entry in the object library directory for every non-RPG overlay load (O.) module that is linked into the object library.
3. Builds the overlay fetch table located in the user's RPG mainline program.
4. Updates the volume label sector on the disk pack containing the load (O.) modules.
5. Provides printed output in the form of:
 - Diagnostic error messages
 - A listing of unresolved external references
 - An image of the overlay fetch table
6. Punches a LOAD * deck if punched output is requested.

Note: When using a Supervisor:

1. Object (R.) modules are not relocatable, executable, loadable, or in main storage image.
2. Load (O.) modules are relocatable, executable, loadable, and in main storage image.

The Linkage Editor consists of five processing steps:

1. PASS1 Root Phase (\$LINKB) consisting of:
 - \$LINKC—INCLD, ENTRY Control Record and AUTOLINK Processor
 - \$LINKD—TEXT-RLD and END Control Record Processor
 - \$LINKE—ESL and OPTNS Control Record Processor
 - \$LINKF—PHASE Control Record Processor
2. External Symbol List (ESL) Table—Process Phase (\$LINKM)
3. PASS2 Root Phase (\$LINKG) consisting of:
 - \$LINKH—TEST-RLD Conversion routine
 - \$LINKJ—Directory Build routine
4. Error and Overlay Fetch Table—Print Phase (\$LINKK)
5. Punch Output Processor (\$LINKN)

Figure 7-4 shows these processing steps. The following discussion will take each processing step and break it down into its components and functions.

PASS1 ROOT PHASE (\$LINKB)

The PASS1 Root Phase (\$LINKB) initially receives control when the Linkage Editor is called by the Scheduler (Figure 7-4). The PASS1 Root Phase performs the following functions:

- Initializes the Linkage Editor communications region (COMMON)
- Opens the linkage editor input work file (\$WORK)
- Reads the object (R.) modules from the \$WORK file
- Determines the type of records being processed
- Calls the Find transient (\$\$SPFN) to build the linkages required for loading the following six modules into main storage: (1) \$LINKC, (2) \$LINKD, (3) \$LINKE, (4) \$LINKF, (5) \$LINKK, (6) \$LINKM
- Builds entries in the linkage editor error table (ERRTAB)
- Passes control to one of the control record processors

INCLD and ENTRY Control Record and AUTOLINK Processor (\$LINKC)

Control is passed to this routine after either an INCLD, ENTRY, PHASE or /* record has been read by the PASS1 Root Phase (\$LINKB). This record processor (\$LINKC) will in all cases determine the syntactical correctness of the name located on the control record. The processor then performs one of the following functions depending upon the type of record being processed.

INCLD Control Record

When the INCLD control record is being processed, the system Find transient (\$\$SPFN) is called to search the object library directory to locate the requested object (R.) module. The object (R.) module is then processed and control is returned to the PASS1 Root Phase (\$LINKB). The PASS1 Root Phase (\$LINKB) will then read the next record from \$WORK. The INCLD statement is changed to *FOUND if the module is located. A *NOFND is set if the module cannot be located.

ENTRY Control Record

When the ENTRY control record is being processed, the processor (\$LINKC) builds an entry control field in the PASS1 communication region. Control is then returned to the PASS1 Root Phase (\$LINKB). The PASS1 Root Phase then reads the next record from \$WORK.

PHASE or / Control Records*

When the second PHASE or a /* control record is being processed, the processor (\$LINKC) performs the AUTOLINK function. The AUTOLINK function is the scanning of the records in the ESL table that were built during the last PASS1 Phase, for unresolved EXTRN references. The name of the located unresolved EXTRN is used in a search of the object library directory. If a match is found, the text from the object (R.) module, referenced in the object library directory, is included in the current load (O.) module generated by the PASS1 Root Phase (\$LINKB). If no match is found, \$LINKC generates an error code in the error table (ERRTAB). The AUTOLINK process continues until all EXTRN records have either been resolved or indicated as unresolvable. Control is then returned to the PASS1 Root Phase (\$LINKB). The PASS1 Root Phase then reads the next record from \$WORK.

Note: The AUTOLINK function is bypassed if the NOAUTO option was chosen.

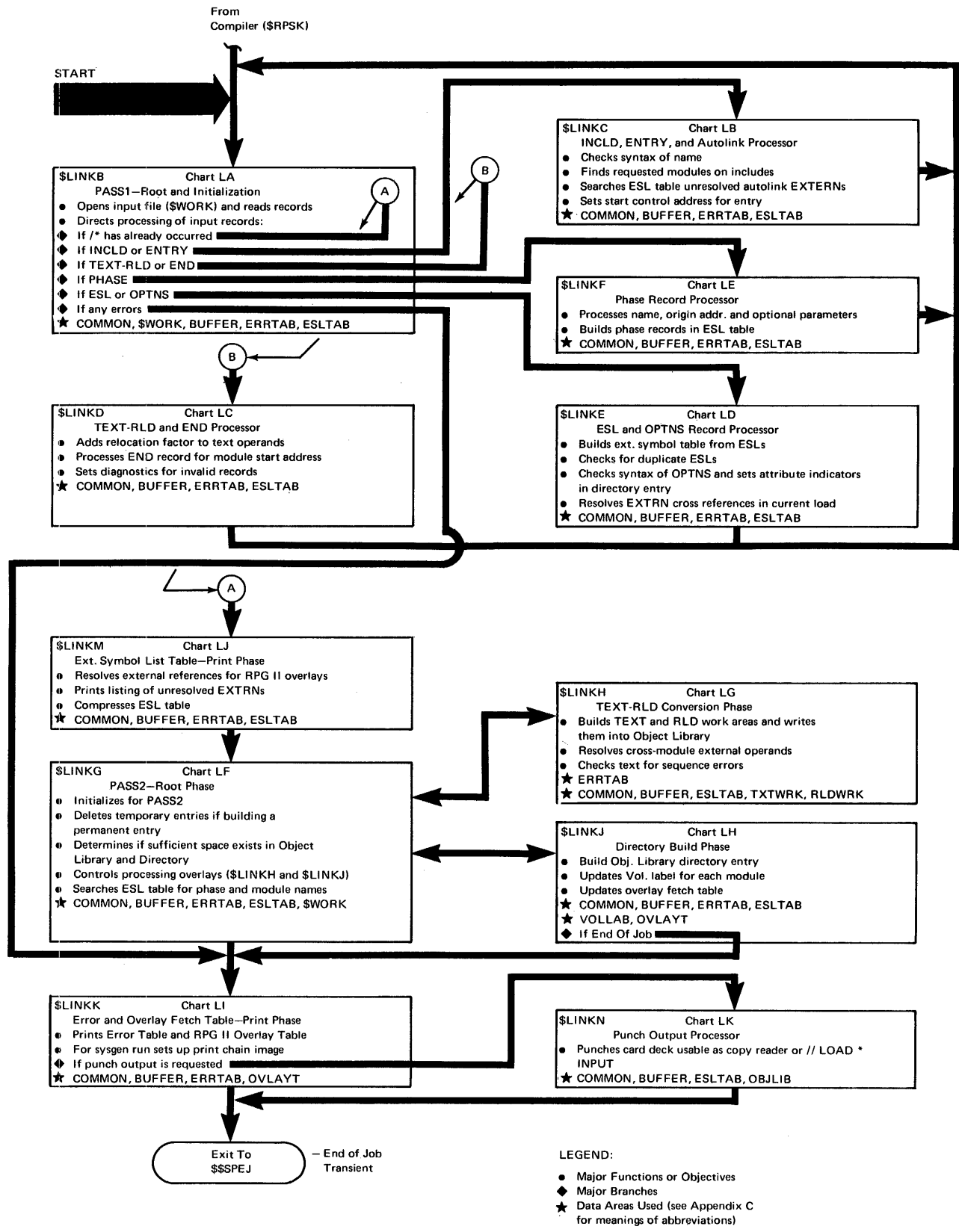


Figure 7-4. Main Logic of Linkage Editor

TEXT-RLD and END Control Records Processor (\$LINKD)

Control is passed to this processor (\$LINKD) if either a TEXT-RLD input record or an END control record is being processed. This processor (\$LINKD) performs the following functions, depending upon the type of record being processed:

TEXT-RLD Input Record

When a TEXT-RLD input record is being processed, \$LINKD updates the following items:

1. The TEXT record starting address
2. The TEXT length
3. The relocation directory (RLD) sequence
4. The value of the non-external operands

The updated TEXT-RLD record then returns to the language translator output work area (\$WORK).

END Control Record

When an END control record is being processed, \$LINKD determines the placement and value of the END record. The following functions are performed by \$LINKD depending upon the placement and value of the END control record:

1. If SWITCH0 in the PASS1 communication region (COMMON) indicates that a valid ENTRY control record was processed during this phase (before a /* or second PHASE control card was processed), the END control record is not processed. Control is returned to the PASS1 Root Phase (\$LINKB), which in turn reads the next record from the input file (\$WORK).
2. If the END record is part of an object (R.) module, the END record is not processed. Control is returned to the PASS1 Root Phase (\$LINKB), which in turn reads the next record from the input file (\$WORK), except during AUTOLINK.
3. If the current END control record is the first mainline END record, but has a null (X'FFFF') start address field, the END record is not processed. Control is returned to the PASS1 Root Phase (\$LINKB), which in turn reads the next record from the input file (\$WORK).
4. If the current END control record is not the first END record with a valid start address, this record is not processed. Control is returned to the PASS1 Root Phase (\$LINKB), which in turn reads the next record from the input file (\$WORK).
5. If the current END control record is the first mainline END record in the current phase, a valid ENTRY control record has not been processed and the start address field does not contain a null value (X'FFFF'). Then the address found on this record is moved into the area DIREPT located in the PASS1 communication region (COMMON).

ESL and OPTNS Control Record Processor (\$LINKE)

Control is passed to this processor if either an ESL or OPTNS control record is being processed. This processor (\$LINKE) performs the following functions depending upon the type of records being processed.

ESL Control Record

When an ESL control record is being processed, \$LINKE determines which of the following ESL record type is being processed:

Type X'00'—Module Name: The processor (\$LINKE) searches the ESL table for a matching module name or entry point. If a match is located, \$LINKE indicates an error in the error table (ERRTAB). If a match is not located, the ESL table is searched again for EXTRN records with the same name. If matching EXTRN records are located, the address and relocation factor are moved from the module name ESL record into the EXTRN records. The EXTRN records are then indicated as being resolved. The C/S in \$WORK at the location where the module is referenced by the name record is then placed in the ESL table.

Type X'01'—Entry Point: The processor (\$LINKE) searches the ESL table for a matching module name or entry point. If a matching record is located, \$LINKE indicates an error in the error table (ERRTAB). If a match is not located, the ESL table is searched again for matching EXTRN records. If matching EXTRN records are located, the address and relocation factor are moved from the entry point ESL record to the EXTRN records and the EXTRNs are indicated as being resolved.

Type X'02'—EXTRN References: The processor (\$LINKE) searches the ESL table for either a module name (type X'00') or an entry point (type X'01') ESL record entry with the same name. If a matching record is located, the address and relocation factor are moved into the EXTRN ESL record before the EXTRN entry is moved into the ESL table. The EXTRN is indicated as being resolved (type X'03'). If a match is not found, the EXTRN ESL entry is indicated as being unresolved.

OPTNS Control Record

When an OPTNS control record is being processed, \$LINKE scans the OPTNS record to ensure the syntactical correctness. The parameters located on the control record are used to set indicators in the directory attribute byte (SWITCH1) located in the PASS1 communication region. Control is returned to the PASS1 Root Phase (\$LINKB), which in turn will read the next record.

PHASE Control Record Processor (\$LINKF)

Control is passed to this processor (\$LINKF) only if a PHASE control record is being processed. The processor (\$LINKF) will syntactically check the PHASE control record for the following:

- **PHASE Name:** If the PHASE name is not found, \$LINKF performs an immediate cancel. If a valid name is located, the name is moved into the DIRNAM located in the PASS1 communication region.
- **PHASE Origin Address:** If an invalid address was specified on the PHASE control record, the Linkage Editor performs an immediate cancel. A valid address is moved into DIRLAD located in the PASS1 communication region.
- **Load (O.) Module Release Number:** A valid release level number (01-FF) is used to build a 1-byte level indicator (DIRLEV) located in the PASS1 communication region.
- **Load Module Size:** The size attribute indicates the main storage needed by the load (O.) module. If the size was not specified, SIZE will be set to the number of text sectors of the O. module.
- **Attribute:** TEMP or PERM indicates whether the module is to be temporary or permanent.

After all of the above parameters have been processed, the PHASE control record is moved into the ESL table. Control is then returned to the PASS1 Root Phase (\$LINKB), which in turn will call \$LINKC to perform the AUTOLINK function, unless the NOAUTO option was chosen.

EXTERNAL SYMBOL LIST (ESL) TABLE—PROCESS PHASE (\$LINKM)

After all input records have been processed, the PASS1 Root Phase (\$LINKB) passes control to the External Symbol List (ESL) Table Process Phase (\$LINKM).

After receiving control from the PASS1 Root Phase (\$LINKB), this phase (\$LINKM) performs the following three functions:

- Formats and prints a listing of any unresolved EXTRN records
- Resolves cross phase (between load (O.) modules) EXTRN references for RPG overlay modules
- Compresses the external symbol list table into the high end of main storage

After the ESL Table—Process Phase (\$LINKM) has completed processing, control is passed to the PASS2 Root Phase (\$LINKG). If an error was found during \$LINKM, control is passed to the Error and Overlay Fetch Table—Print Phase (\$LINKK).

PASS2 ROOT PHASE (\$LINKG)

After receiving control from \$LINKM, this phase (\$LINKG) performs the following functions:

- Determines if enough disk storage is available in the object library and the object library directory for the load (O.) modules. If there is not enough room available, the PASS2 Root Phase (\$LINKG) issues an immediate cancel.
- Deletes all temporary entries in the object library if the current load (O.) module has an attribute of PERM.

- Enters the data found in the first PHASE record of the ESL table, into the PASS2 communication region (COMMON).
- Uses the module ESL records (type X'00') in the ESL table to locate the object (R.) text-RLD records to be converted to load (O.) module code.
- Calls the TEXT-RLD Conversion Phase (\$LINKH) to process the TEXT-RLD records.
- When the END control record is read, the PASS2 Root Phase (\$LINKG) searches the ESL table for the next PHASE or module record. If a PHASE record is located, the PASS2 Directory Build Phase (\$LINKJ) is called to build the object library directory or an overlay fetch table entry for the last load (O.) module. The volume label is also updated at the same time. If a module record is located, control is passed to the TEXT-RLD Conversion Phase (\$LINKH).
- The error table (ERRTAB) is updated if any errors occur during the PASS2 phase.

After all modules have been processed, control is passed to the Error and Overlay Fetch Table—Print Phase (\$LINKK).

ERROR AND OVERLAY FETCH TABLE—PRINT PHASE (\$LINKK)

This phase prints the error table (ERRTAB) and the RPG overlay fetch table. If punched output is requested, control is passed to the Punch Processor (\$LINKN), otherwise control is passed to the End-of-Job transient (\$\$SPEJ).

PUNCH PROCESSOR (\$LINKN)

This processor is called if punched output has been requested. The punched output is in the form of a LOAD * deck. Control is passed from this processor (\$LINKN) to the End-of-Job transient (\$\$SPEJ).

This section describes in detail the organization of the Linkage Editor routines shown in Figure 7-4. Included in the following discussions are:

- Description containing: name of routine, input, output, function, exits from this routine.
- Main storage map showing the routines that might be in main storage at the same time as the routine being described.
- Flowchart showing the functional flow of the routine being described.

► **PASS1—Root and Initialization Phase (\$LINKB)**

CHART: LA

FIGURE: 7-5

ENTRY POINTS:

- DEDAB0—Mainline entry from initialization
- DEDAB1—Data Management entry point from PASS1 overlay phase
- DEDAB2—Entry point from PASS1 overlays to continue AUTOLINK processing
- DEDAB3—Entry point to call \$LINKF
- DEDAB4—Entry point for building error table (ERRTAB)
- DEDAB5—Entry point for calling \$LINKK for job termination
- DEDAB6—Data Management entry point
- DEDAB7—Entry point for physical I/O linkage
- DEDAB8—Initialization routine

FUNCTION:

- Initializes the PASS1 communication region (COMMON).
- Opens the language translator output work area (\$WORK).
- Builds the linkage-editor error table.
- Reads the object (R.) module from \$WORK.
- Determines the type of input records being processed.
- Provides linkage for calling control record processors.

INPUT:

- The language translator output work area (\$WORK)
- The R. module in the object library

OUTPUT:

- Referenced object modules (R. modules) are written into the language translator output work file (\$WORK).
- The ESL table is built in main storage (controls the order of PASS2 processing).
- Error table for link-edit errors.

EXIT:

- Normal: Supervisor General Entry/Exit routine (NENTRY) after loading one of the following processors:
 1. \$LINKC—INCLD and ENTRY Control Record and AUTOLINK Processor
 2. \$LINKD—TEXT-RLD and END Record Processor
 3. \$LINKE—ESL and OPTNS Record Processor
 4. \$LINKF—PHASE Control Record Processor
 5. \$LINKM—ESL Table-Process Phase
- Error: Error Table Print (\$LINKK) if terminal error occurs

► **PASS1—INCLD and ENTRY Control Record and AUTOLINK Processor (\$LINKC)**

CHART: LB

FIGURE: 7-5

ENTRY POINT: DEDAC0

FUNCTION:

- Performs syntax checking of the name field in the INCLD and ENTRY control statements.
- Calls the Find transient (\$\$SPFN) to find modules that are requested by the INCLD input records.
- Builds the ENTRY control field in the object library directory record for the phase.
- Controls search of the ESL table for EXTRN words during AUTOLINK processing.
- Searches ESL table for duplicate INCLD modules.

INPUT: INCLD and ENTRY control records are located in the input/output buffer (BUFFER).

OUTPUT:

- The control start address entry (located in the library directory)
- Provides diagnostic messages for the following:
 1. Duplicate module calls.
 2. Syntax errors in the name field.
 3. Multiple ENTRY statements per phase.
 4. Invalid control starting points.

EXIT:

- Normal: PASS1 Root Phase entry point (DEDAB0)
- Error: Error Table Build entry point (DEDAB4) located in the PASS1 Root Phase (\$LINKB)

► **PASS1—TEXT-RLD and END Record Processor (\$LINKD)**

CHART: LC

FIGURE: 7-5

ENTRY POINT: DEDAD0

FUNCTION:

- Updates the non-external TEXT operands by adding the module's relocation factor to the operand.
- Processes END record operands for the load (O.) module start control address.
- Processes the load (O.) module ESL table to resolve non-cross module EXTRN records.
- Indicates and sets diagnostic messages for invalid TEXT length and invalid RLDs.

INPUT: Input for this phase comes in the form of TEXT-RLD and END control input records that are located in the input/output buffer.

OUTPUT: There are two areas used for the output from this phase depending upon the type of records that are processed:

1. TEXT-RLD Records—Output goes into the input/output buffer (BUFFER).
2. END Records—The END operand is processed and the start control address of load (O.) module is moved into DIREPT located within the PASS1 communication region (COMMON).

EXIT:

- Normal: PASS1 Root Phase (DEDAB0)
- Error: PASS1 Root Phase (DEDAB4). Invalid length text records will cause job termination after \$LINKK is called by the PASS1 Root Phase (\$LINKB) to print the error table (ERRTAB).

► **PASS1—ESL and OPTNS Record Processor (\$LINKE)**

CHART: LD

FIGURE: 7-5

ENTRY POINT: DEDAE0

FUNCTION:

- Locates and indicates multidefined module names or entry point ESL records that appear within the current load (O.) module.
- Resolves EXTRN cross references in the current load (O.) module when possible.
- Builds the external symbol list (ESL) table for the ESL records.
- Unblocks up to five ESL entries within one input ESL record.
- Performs a syntax check on the OPTNS record.
- Processes the OPTNS record and sets the indicators in the attribute portion of the object library directory entry for the phase.

INPUT: ESL or OPTNS input records located in the input buffer (BUFFER)

OUTPUT:

- The external symbol list (ESL) table is updated to reflect the inclusion of new ESL records.
- The OPTNS switches are set in the directory save area of the PASS1 communication region (COMMON).

EXIT:

- Normal: PASS1 Root Phase at entry point (DEDAB0)
- Error: Error Table Build entry point (DEDAB4) located in the PASS1 Root Phase (\$LINKB)

► **PASS1—PHASE Control Record Processor (\$LINKF)**

CHART: LE

FIGURE: 7-5

ENTRY POINT: DEDAFO

FUNCTION: Processes each PHASE control record for:

1. Phase name
2. Phase origin address
3. Load module level number
4. Temporary or permanent attribute of output
5. Program execution size

INPUT: PHASE control record contained in the input/output buffer (BUFFER).

OUTPUT: PHASE control record information is placed in the PASS1 communication region (COMMON) until all of the PHASE parameters have been processed. The PHASE record is then moved to the external symbol list table.

EXIT:

- Normal: PASS1 Root Phase (\$LINKB) at the entry point (DEDAB1)
- Error: PASS1 Root Phase (\$LINKB) at the entry point (DEDAB4)

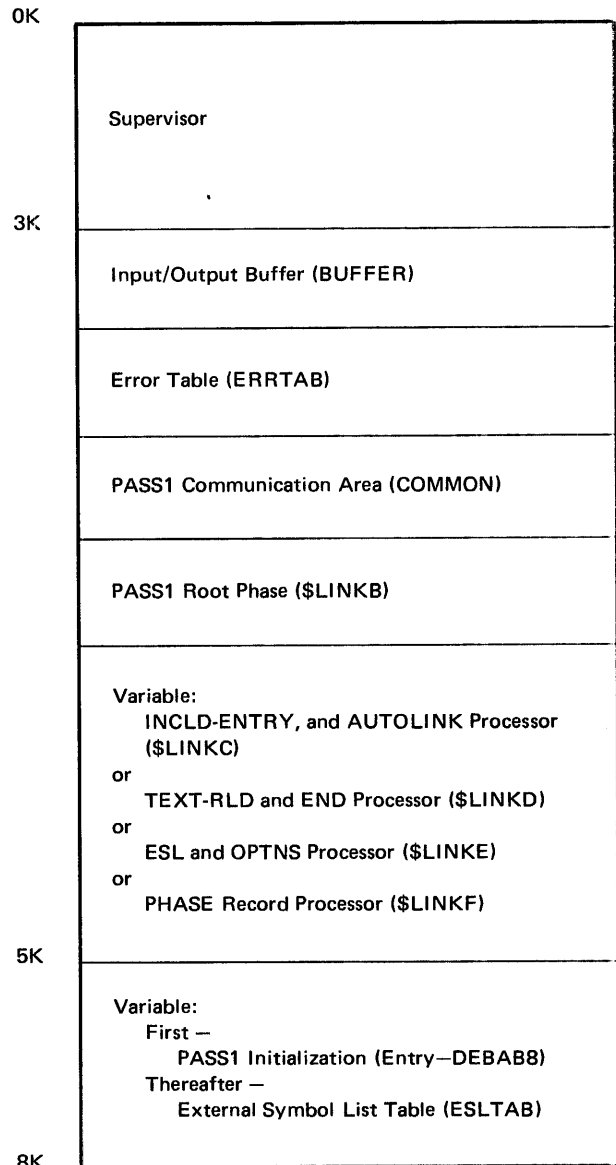


Figure 7-5. Main Storage Map for PASS1

► **PASS2—Root Phase (\$LINKG)**

CHART: LF

FIGURE: 7-6

ENTRY POINT:

- DEDAG0—Initialization phase
- DEDAG1—Mainline entry point from \$LINKH-overlay phase
- DEDAG2—Error table build
- DEDAG3—\$LINKK call entry point
- DEDAG4—Mainline entry point from \$LINKJ—Directory Build phase

FUNCTION:

- Deletes all temporary entries in the object library if the current O. module is PERM.
- Initializes the PASS2 communication region (COMMON).
- Determines if there is sufficient space for the load module in the object library and the object library directory.
- Searches the external symbol list (ESL) table for phase and module record names.
- Reads the input records from the input buffer.
- Updates the error table (ERRTAB).
- Calls the appropriate overlay (\$LINKJ or \$LINKH) to process the records.

INPUT:

- Linkage editor input work file (\$WORK)
- PASS1 communication area (COMMON)
- External symbol list (ESL) table
- Object library portion of the volume label sector, see Figure 5-64 for this data area

OUTPUT: None

EXIT:

- Normal: Control is passed to the appropriate overlay phase (\$LINKJ or \$LINKH) or to the Error and Overlay Fetch Table Print Phase (\$LINKK) upon normal end of job.
- Error: Error and Overlay Fetch Table Print Phase (\$LINKK) before end of job

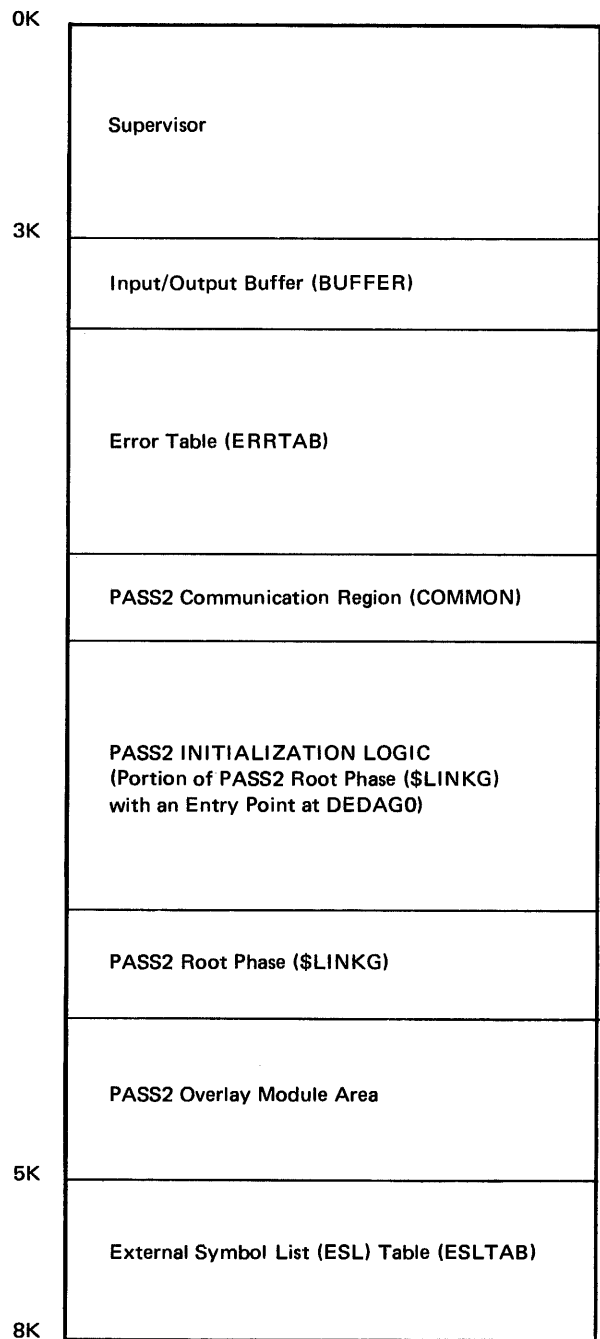


Figure 7-6. Main Storage Map for PASS2 Root Phase (\$LINKG)

► **PASS2–TEXT-RLD Conversion Phase (\$LINKH)**

CHART: LG

FIGURE: 7-7

ENTRY POINT: DEDAHO

FUNCTION:

- Resolves cross-module external operands.
- Builds TEXT and relocation directory (RLD) work areas (TXTWRK and RLDWRK).
- Checks the text for sequence errors.
- Writes the RLD and TEXT work areas into the object library.

INPUT:

- The external symbol list table (ESLTAB)
- TEXT-RLD records from the input file (BUFFER)

OUTPUT: TEXT and RLD records are written into the object library.

EXIT:

- Normal: PASS2 Root Phase (\$LINKG) at label DEDAG1
- Error: Error and Overlay Fetch Table–Print Phase (\$LINKK)

► **PASS2–Directory Build Phase (\$LINKJ)**

CHART: LH

FIGURE: 7-7

ENTRY POINT: DEDAJO

FUNCTION:

- Builds object library directory records for load modules (O. modules) with unreserved names.
- Updates the overlay fetch table for phases with reserved names.
- Updates the object library directory portion of the volume label sector for each load (O.) module.

INPUT:

- Volume label sector (See Figure 5-64)
- External symbol list table
- PASS2 communication region (COMMON)
- Object library directory
- Overlay fetch table

OUTPUT:

- Entries in the object library portion of the volume label sector
- Entries in the object library directory or RPG Root Phase overlay fetch table
- Updated volume label object library pointers

EXIT:

- Normal: Entry point (DEDAG4) of the (\$LINKG) to process next record
- Error: Error table entry point (DEDAG2) of the PASS2 Root Phase (\$LINKG)

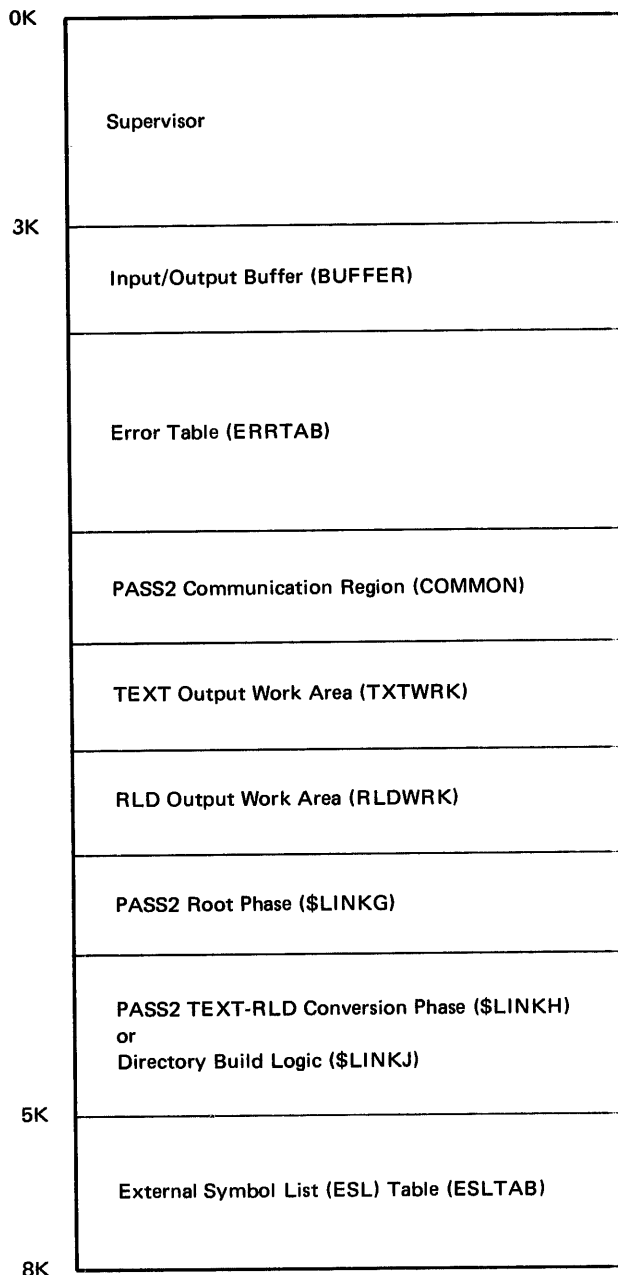


Figure 7-7. Main Storage Map for PASS2 TEXT-RLD Conversion Phase (\$LINKH)

► **Error and Overlay Fetch Table—Print Phase (\$LINKK)**

CHART: LI

FIGURE: 7-8

ENTRY POINT: DEDAKO

FUNCTION:

- Print the error table (ERRTAB).
- Set up the print chain image for System Generation.
- Print the overlay fetch table for RPG overlay program.
- Print linkage-edited supervisor size for System Generation.

INPUT:

- Error table (ERRTAB)
- Print chain image located at X'400'
- Overlay fetch table

OUTPUT:

- Printed listing of the error table (ERRTAB) and overlay fetch table
- Print chain image in Sysgen output
- Supervisor size message for Sysgen run

EXIT: End-of-Job transient (\$\$SPEJ) if punched output is not required. If punched output is required, control is passed to Punch Processor (\$LINKN).

► **External Symbol List (ESL) Table—Process Phase (\$LINKM)**

CHART: LJ

FIGURE: 7-8

ENTRY POINT: DEDAMO

FUNCTION:

- Prints a listing of the unresolved EXTRNs in the external symbol list table.
- Resolves external, cross-phase references for RPG overlays.
- Compresses and moves the external symbol list table into the high end of main storage.

INPUT: The external symbol list table (ESLTAB)

OUTPUT:

- A listing of the unresolved EXTRNs
- Compressed ESL table
- Resolved cross-phase EXTRNs

EXIT:

- Normal: PASS2 Root Phase (\$LINKG)
- Error: Supported Halt/Syslog routine to log the permanent printer errors

► **Punch Output Processor (\$LINKN)**

CHART: LK

FIGURE: 7-8

ENTRY POINT: \$LINKN

FUNCTION:

- Punches a header card with information required to use the punched deck as COPY READER or // LOAD * input.
- Punches a card deck (LOAD * deck) containing Linkage Editor information normally placed in the object library.

INPUT: The TEXT and RLD records, generated by the PASS2 phases, located in the object library

OUTPUT: A punched card deck that can be used as either COPY READER or // LOAD * input. This card deck contains the information normally contained in the object library.

EXIT: Calling routine

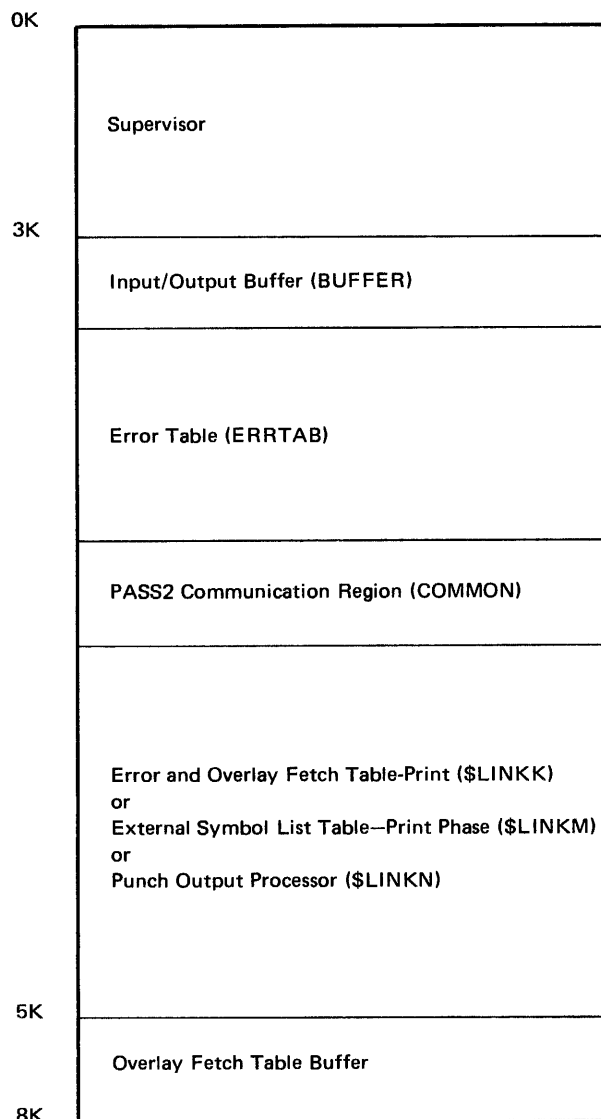


Figure 7-8. Main Storage Map for \$LINKK, \$LINKM, and \$LINKN

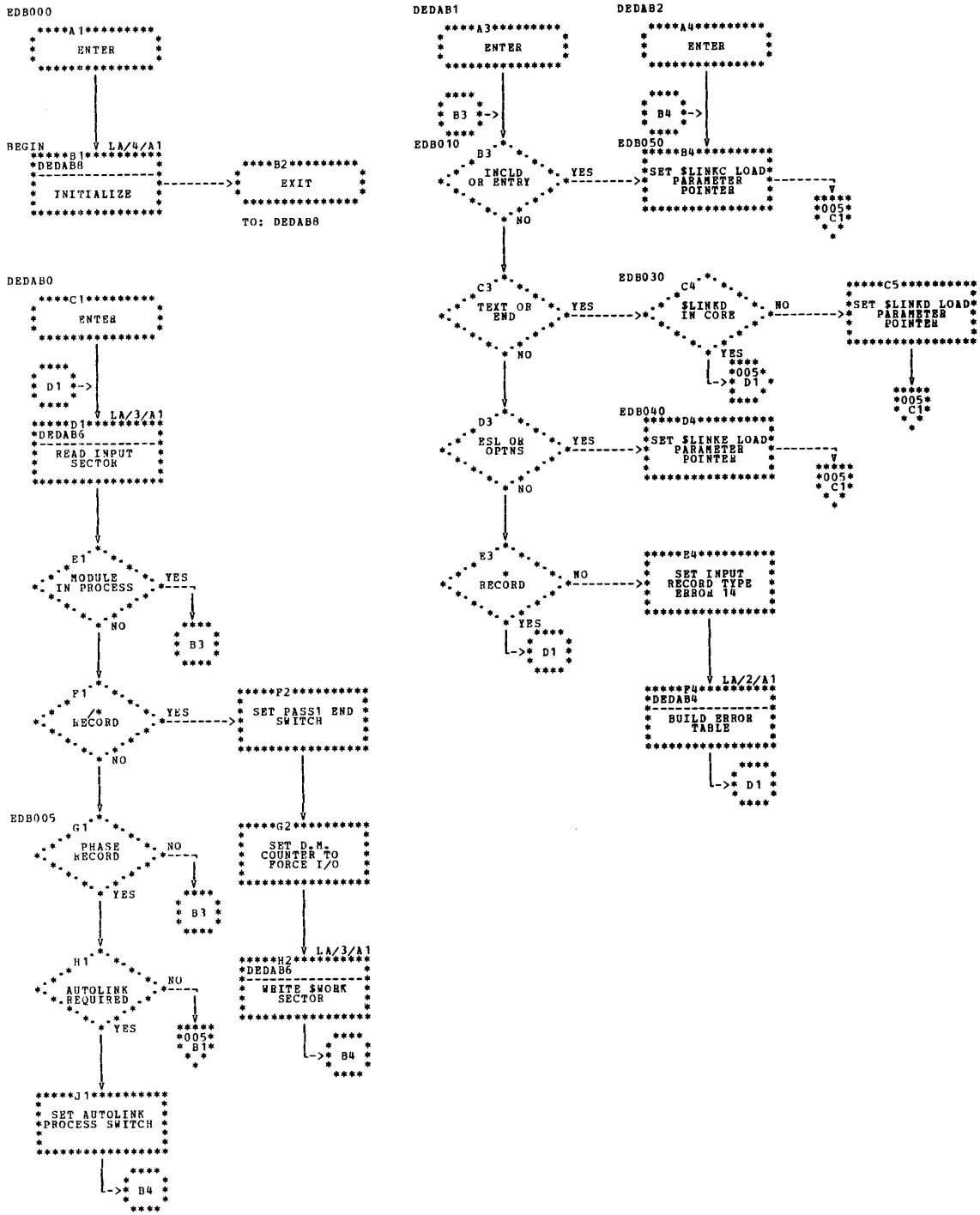


Chart LA (Part 1 of 5). Linkage Editor PASS1 Root and Initialization Phase (\$LINKB)

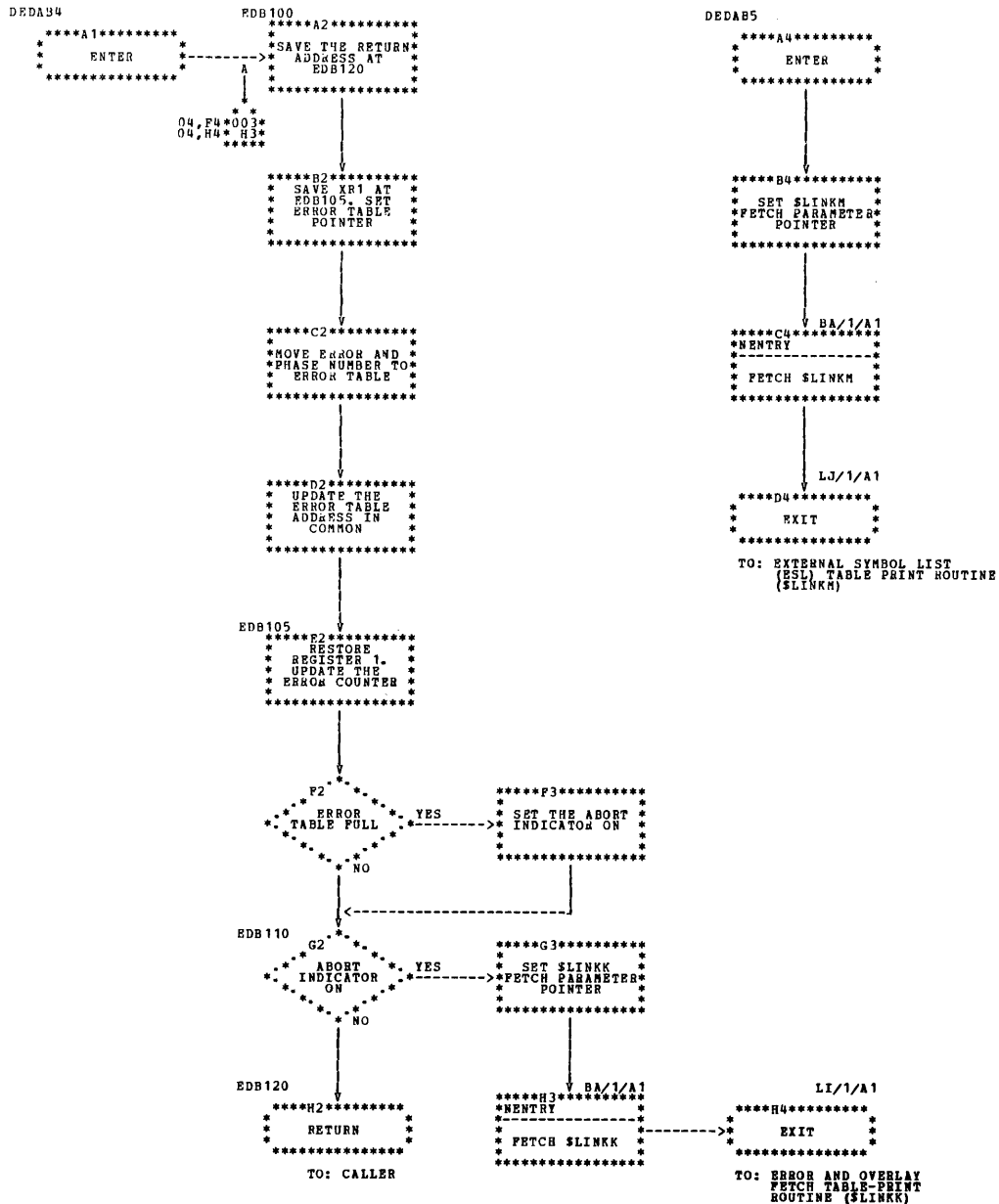


Chart LA (Part 2 of 5). Linkage Editor PASS1 Root and Initialization Phase (\$LINKB)

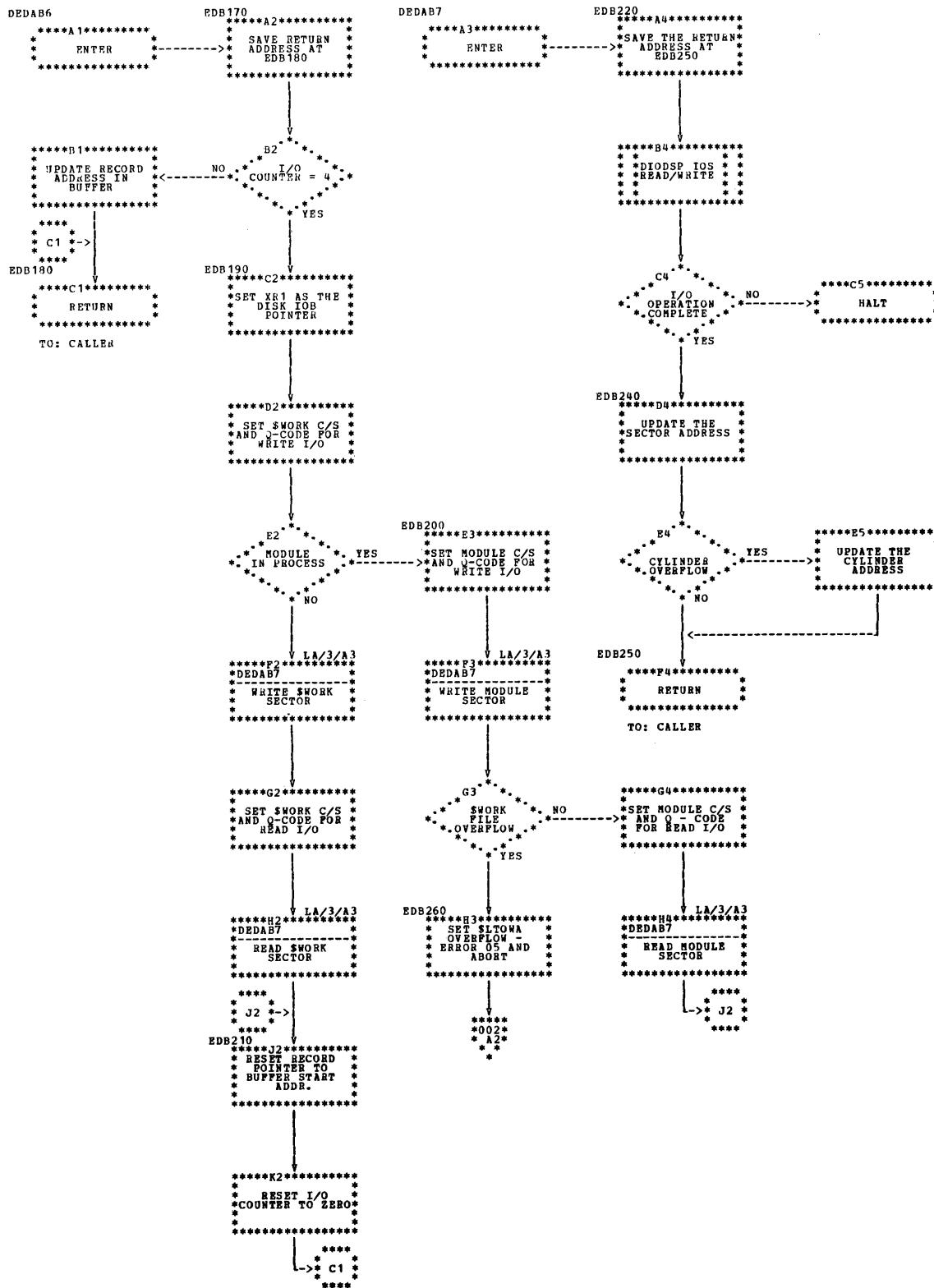


Chart LA (Part 3 of 5). Linkage Editor PASS1 Root and Initialization Phase (\$LINKB)

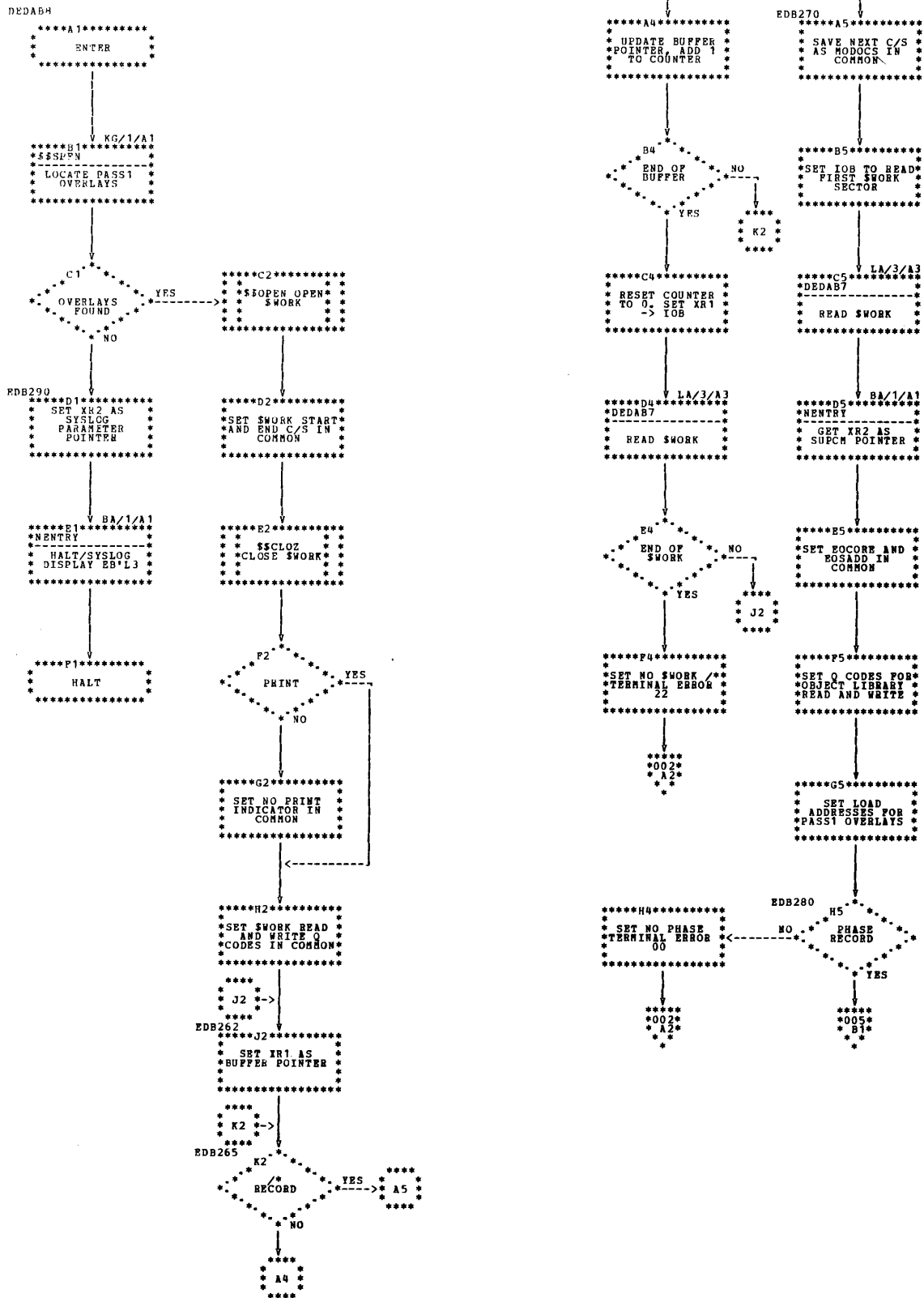


Chart LA (Part 4 of 5). Linkage Editor PASS1 Root and Initialization Phase (\$LINKB)

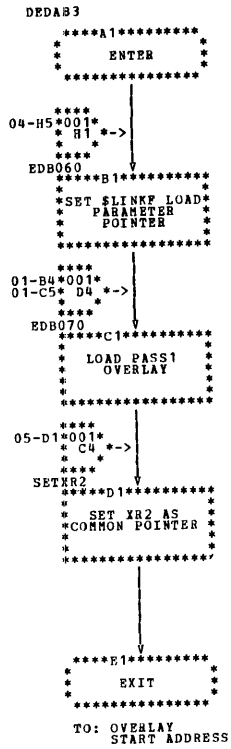


Chart LA (Part 5 of 5). Linkage Editor PASS1 Root and Initialization Phase (\$LINKB)

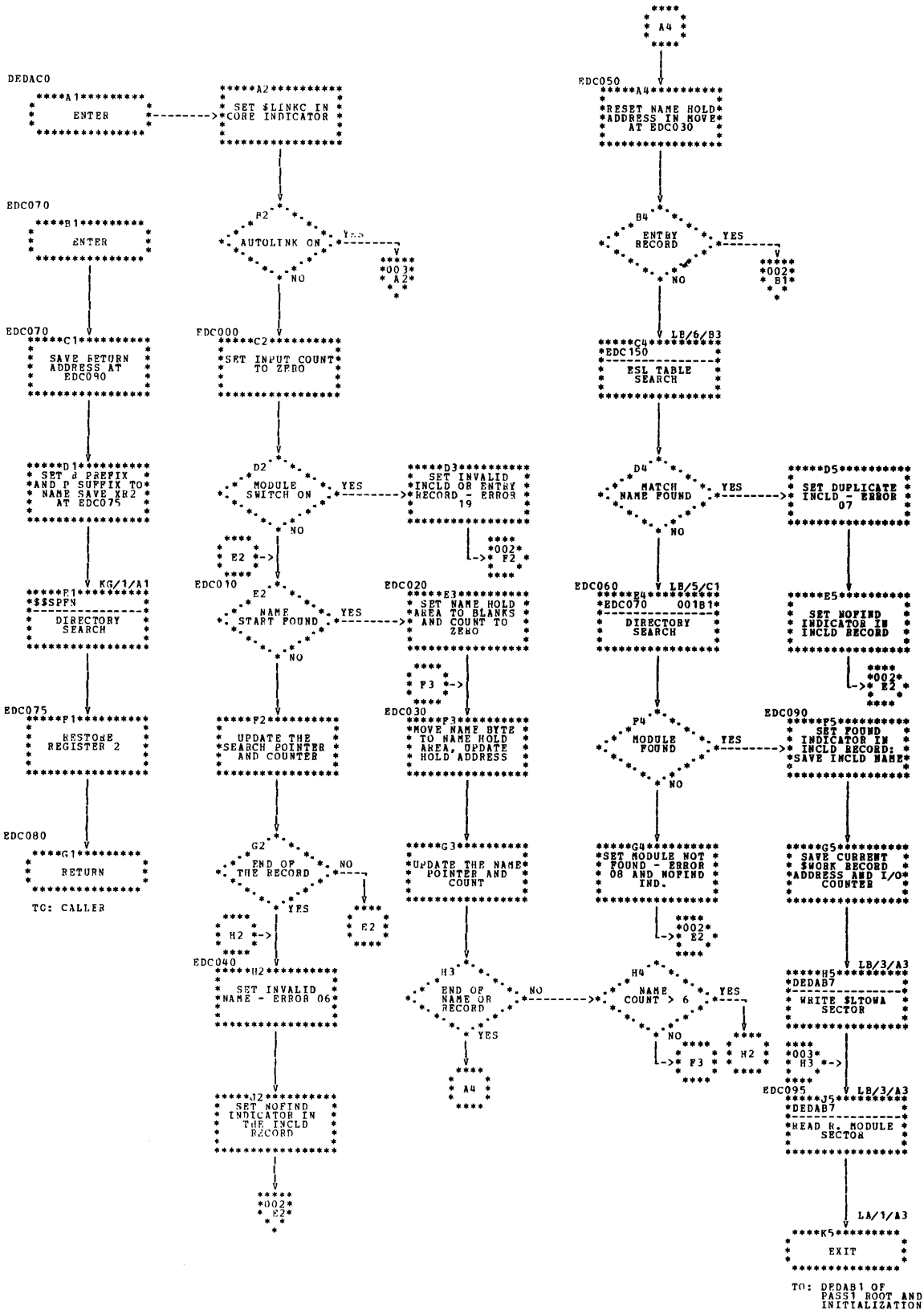


Chart LB (Part 1 of 3). PASS1 INCLD and ENTRY Control Records and AUTOLINK Processor (\$LINKC)

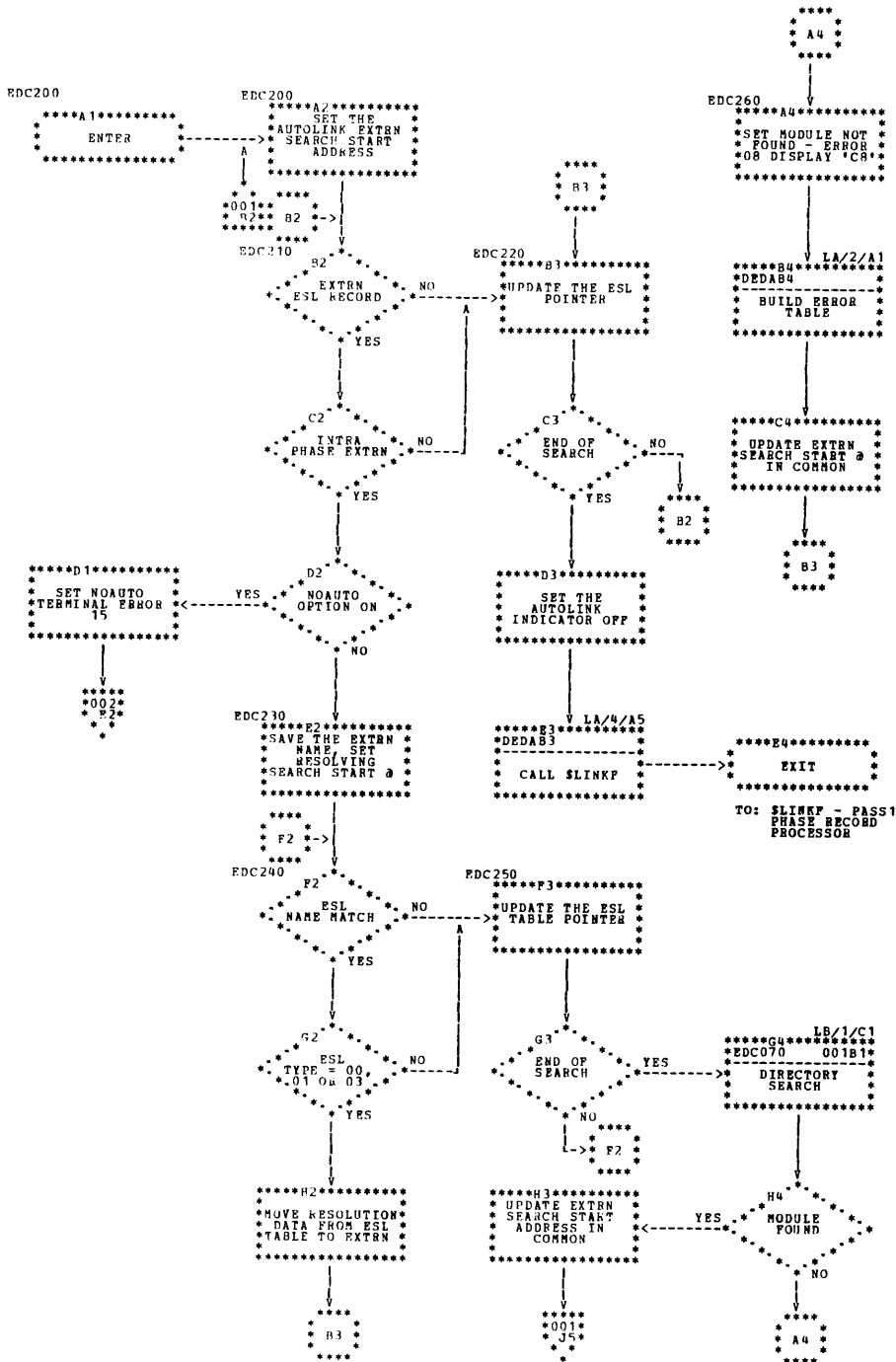


Chart LB (Part 3 of 3). PASS1 INCLD and ENTRY Control Records and AUTOLINK Processor (\$LINKC)

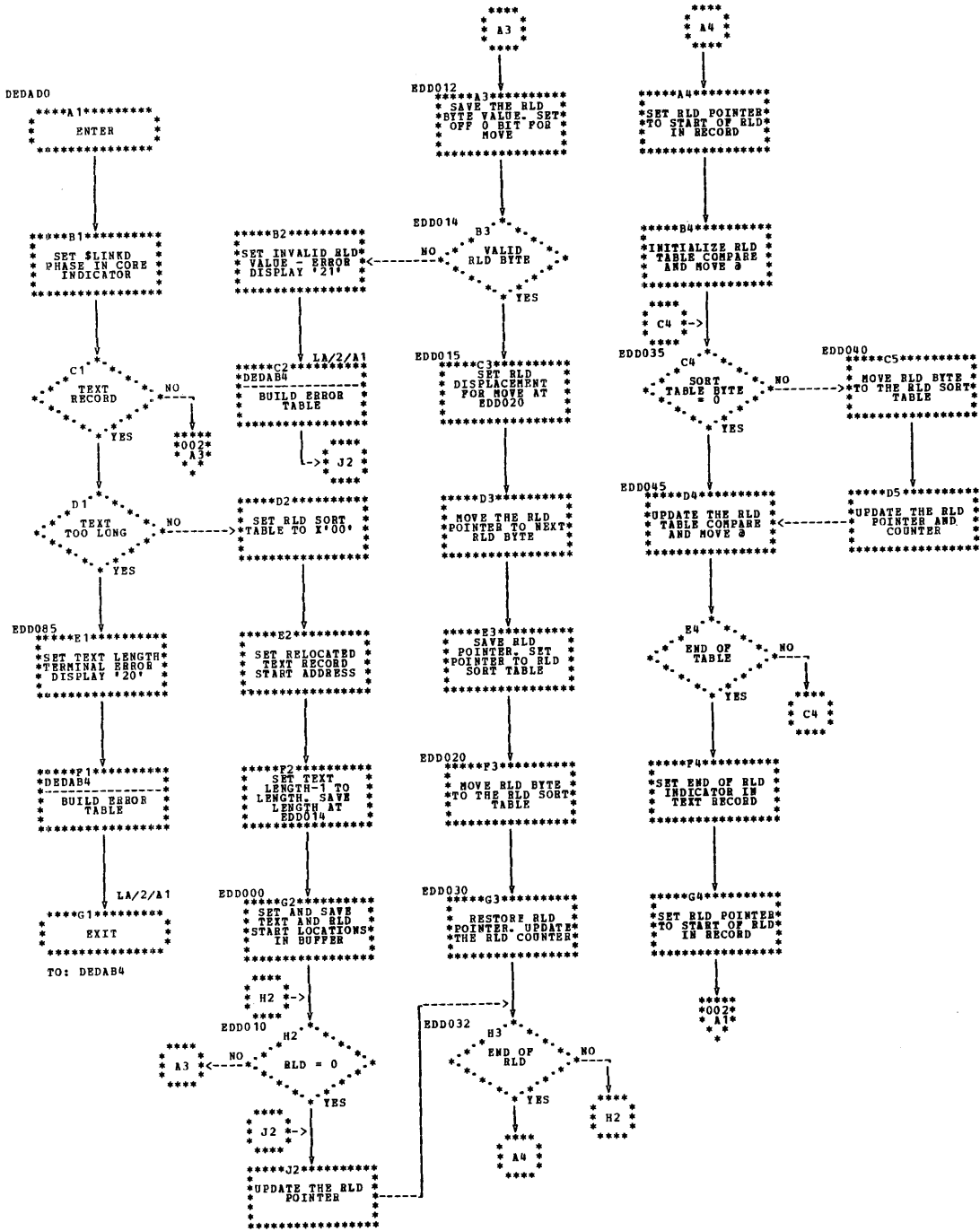


Chart LC (Part 1 of 2). PASS1 TEXT-RLD and END Record Processor (\$LINKD)

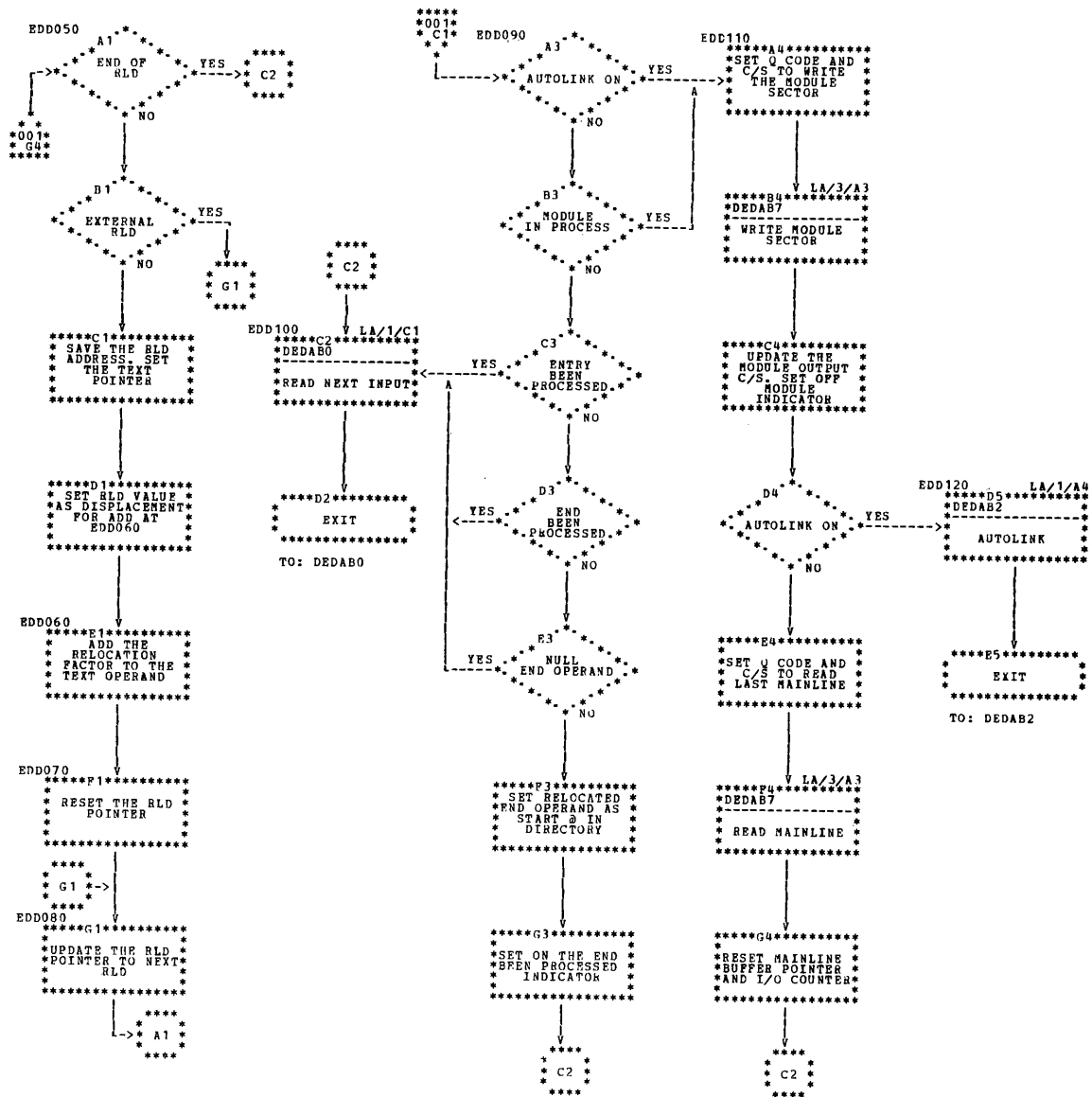


Chart LC (Part 2 of 2). PASS1 TEXT-RLD and END Record Processor (\$LINKD)

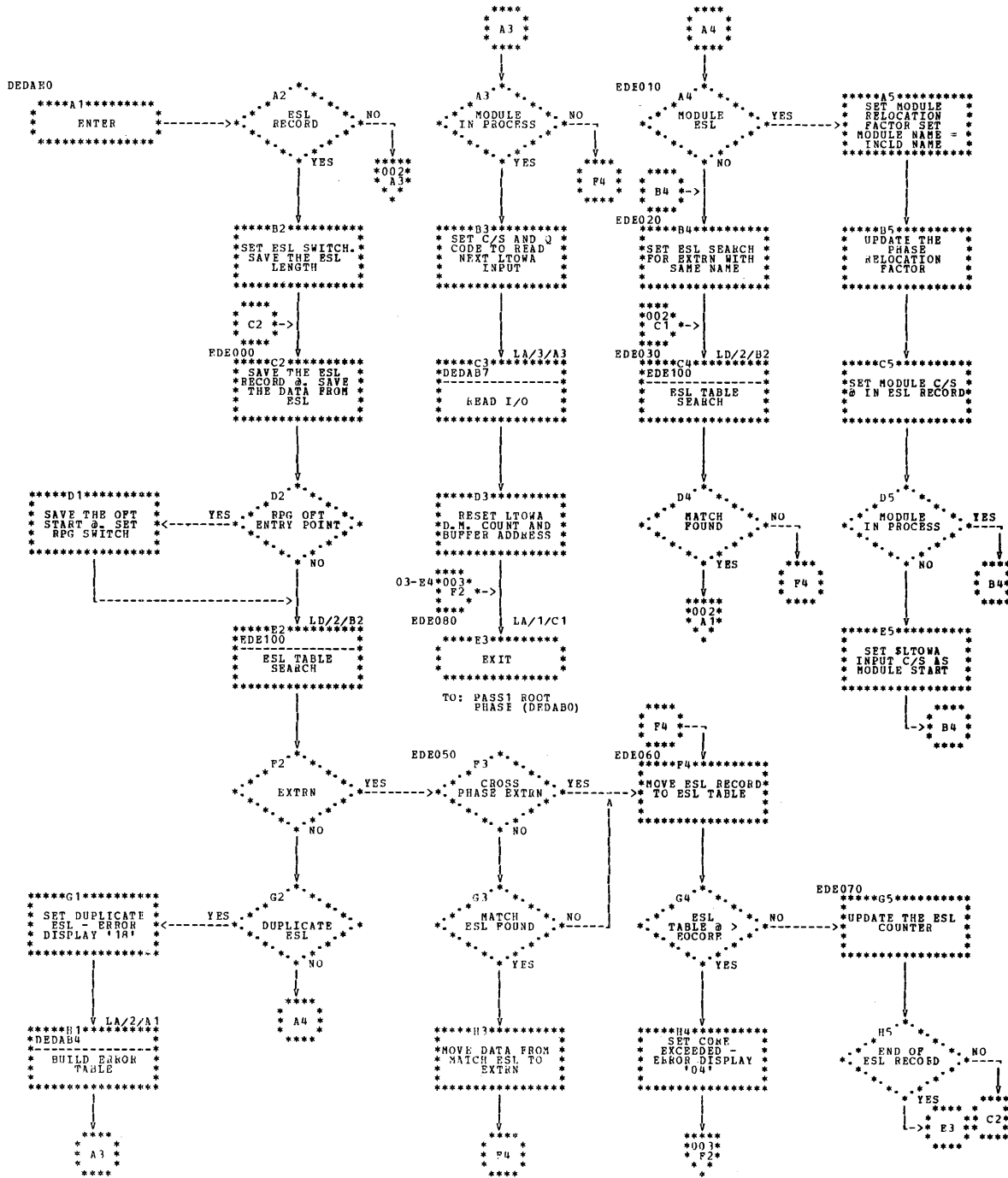


Chart LD (Part 1 of 3). PASS1 External Symbol List OPTNS Record Processor (\$LINKE)

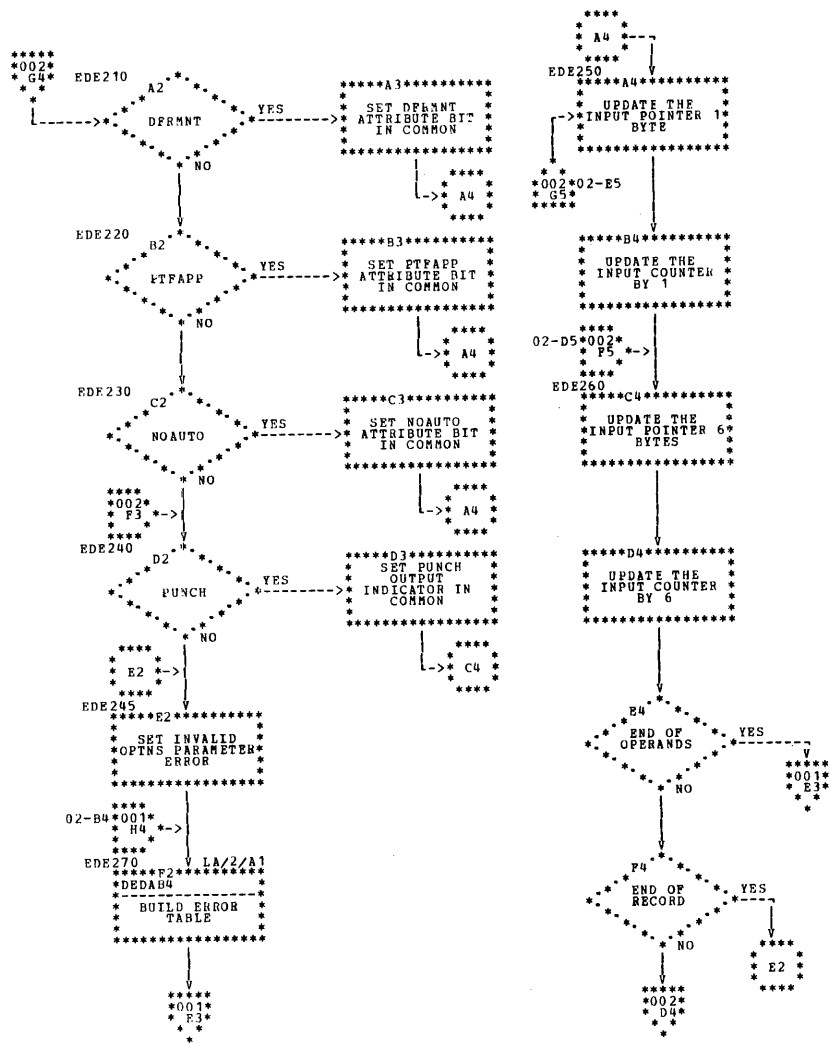


Chart LD (Part 3 of 3). PASS1 External Symbol List OPTNS Record Processor (\$LINKE)

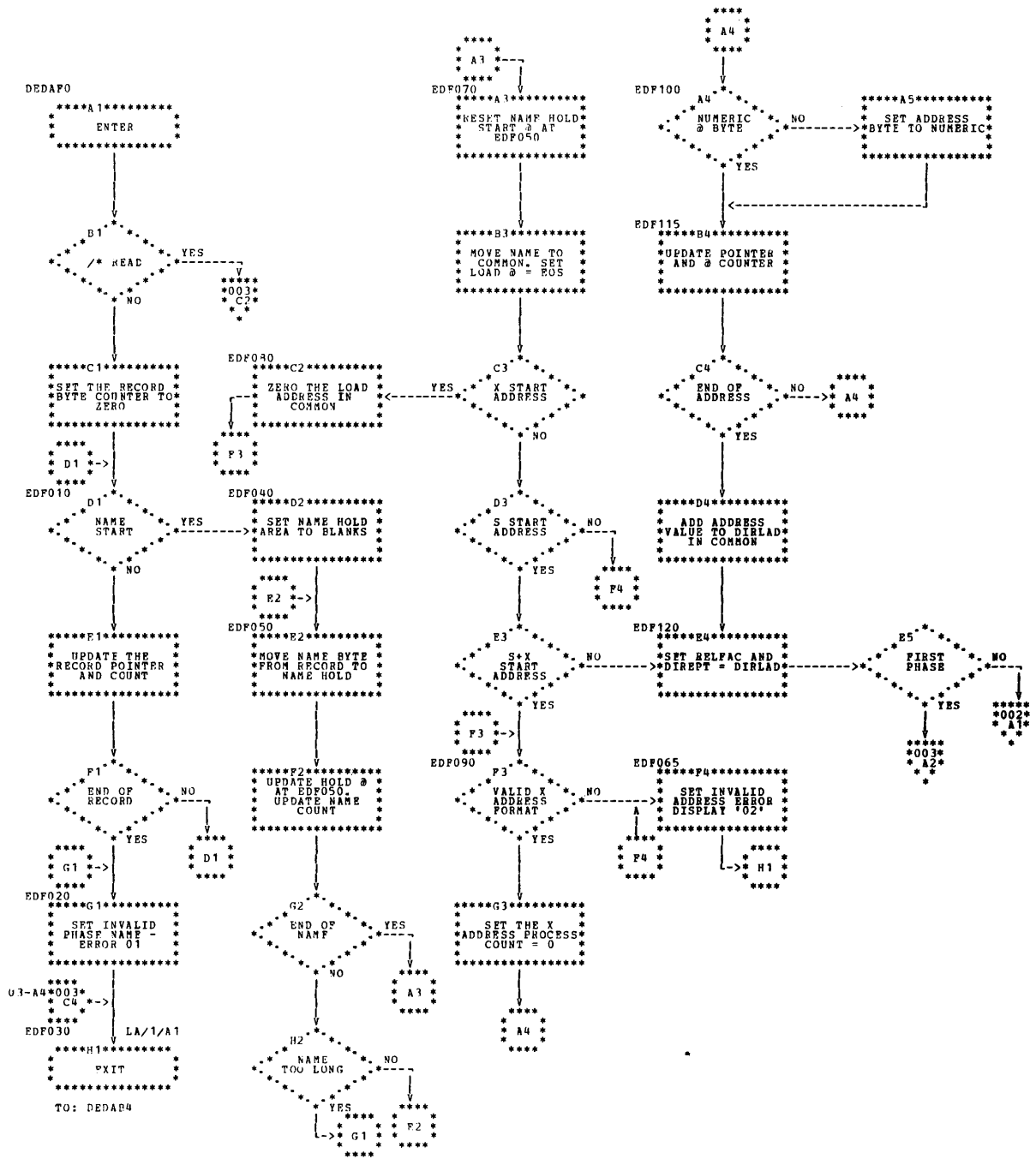


Chart LE (Part 1 of 3). PASS1 PHASE Record Processor (\$LINKF)

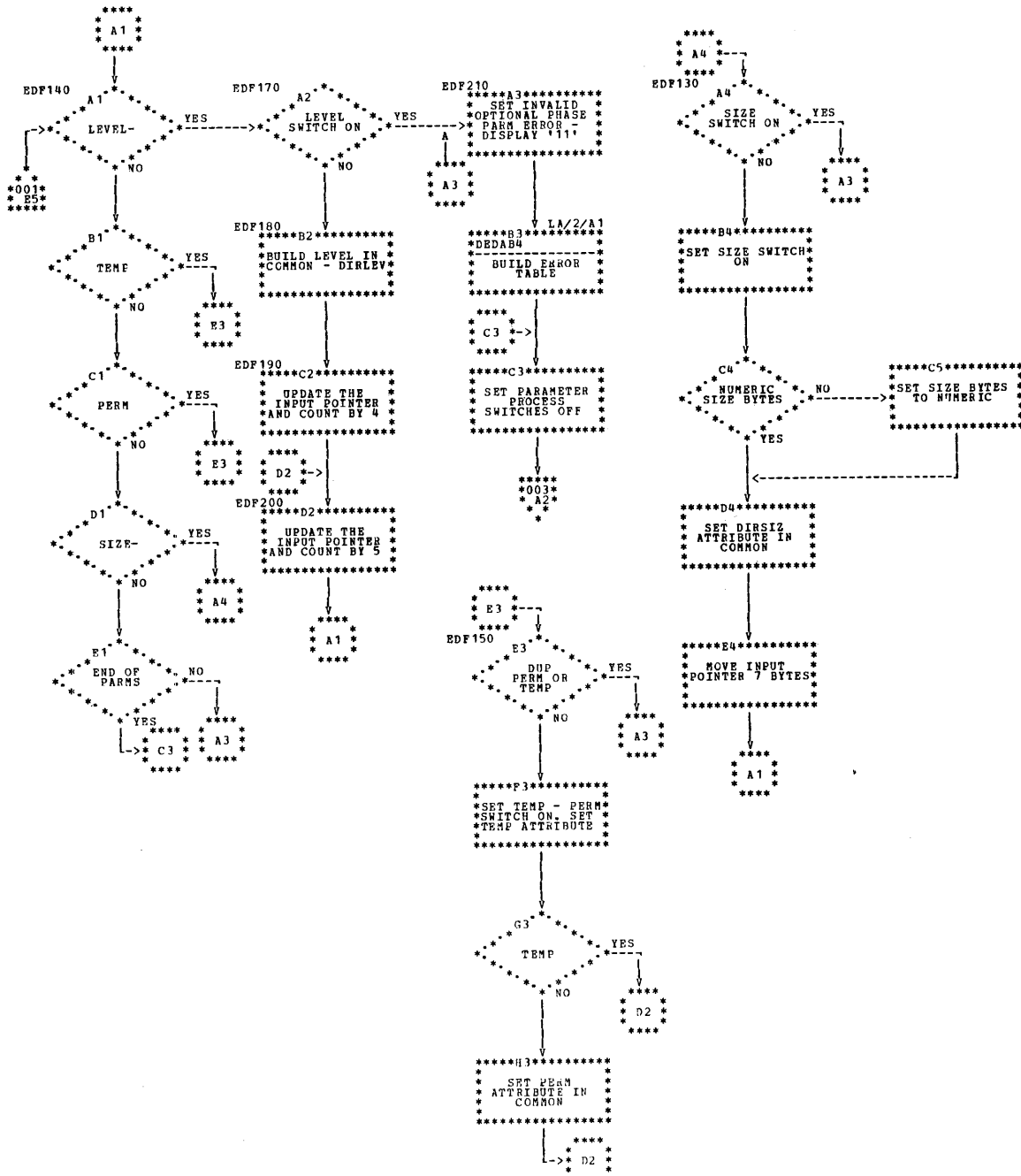


Chart LE (Part 2 of 3). PASS1 PHASE Record Processor (\$LINKF)

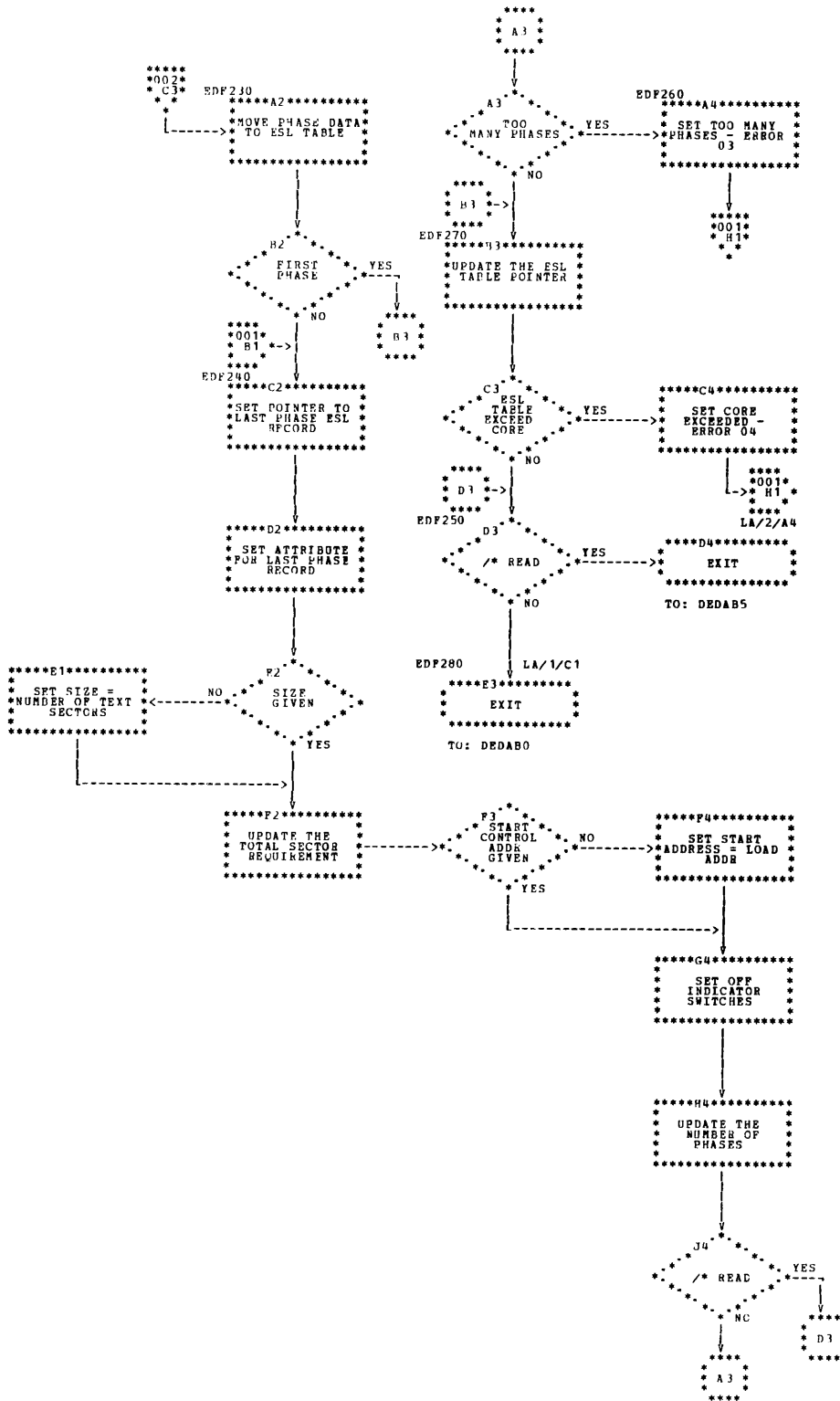


Chart LE (Part 3 of 3). PASS1 PHASE Record Processor (\$LINKF)

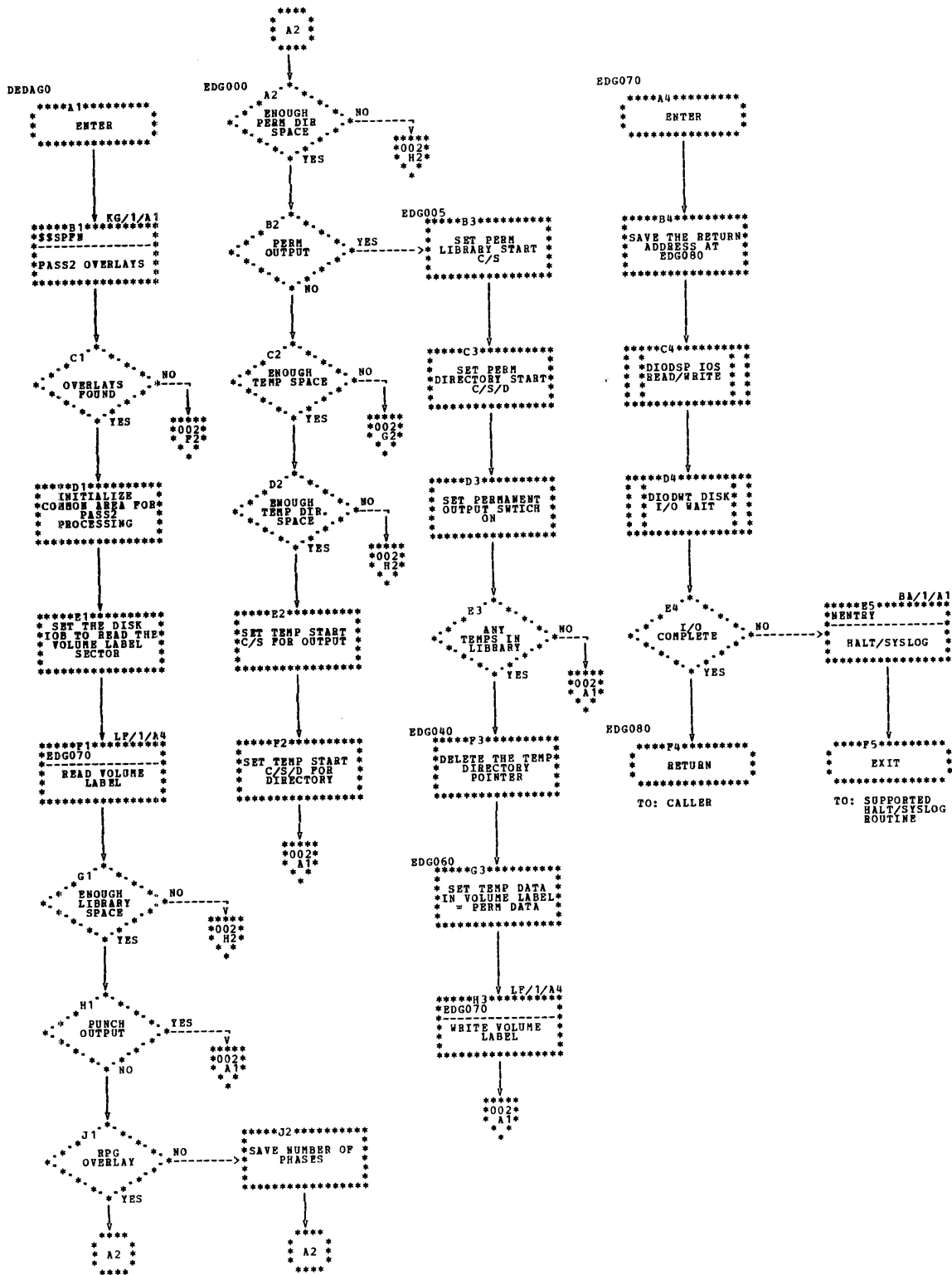


Chart LF (Part 1 of 3). Linkage Editor PASS2 Root Phase (\$LINKG)

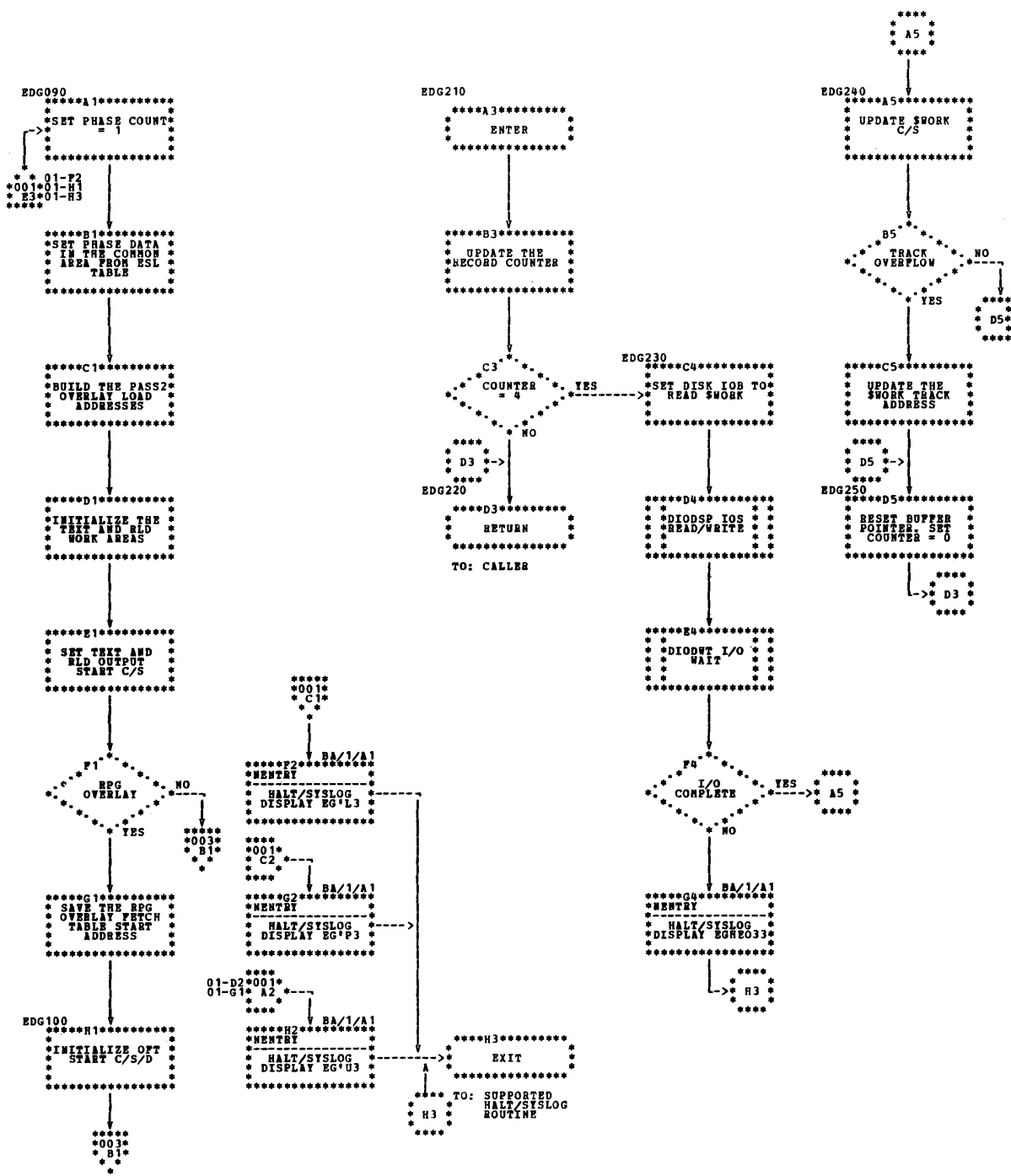
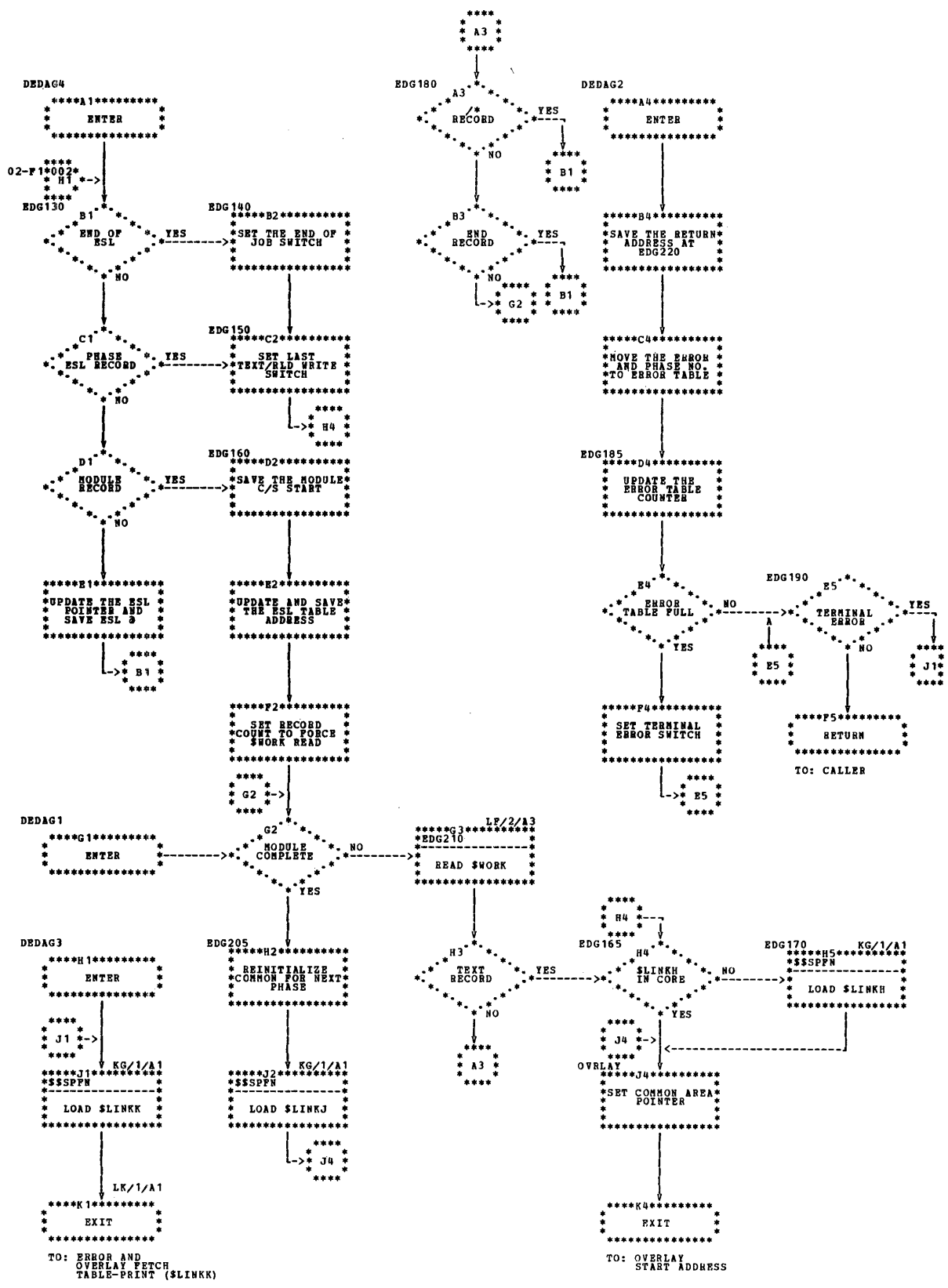


Chart LF (Part 2 of 3). Linkage Editor PASS2 Root Phase (\$LINKG)



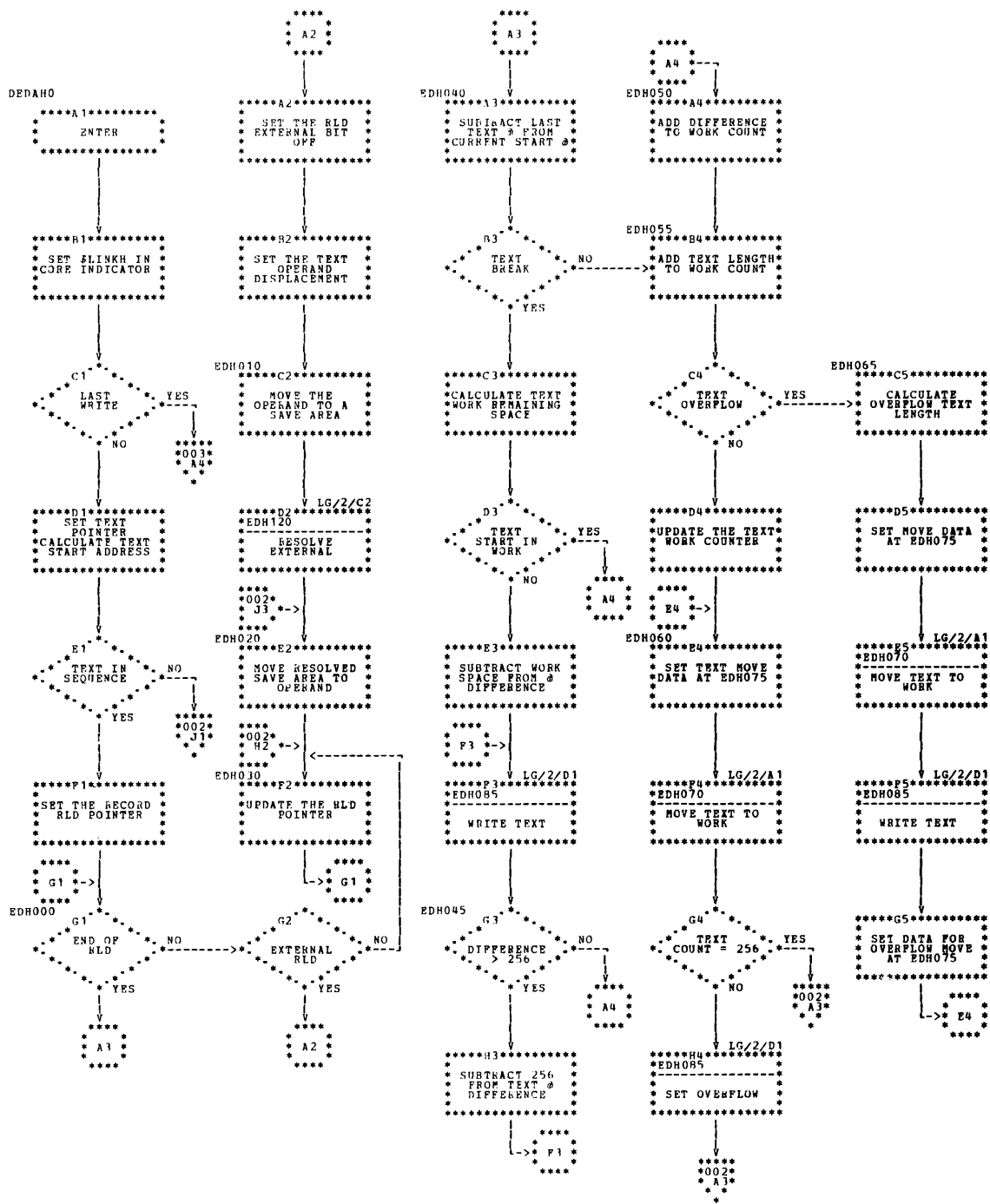


Chart LG (Part 1 of 4). PASS2 TEXT-RLD Conversion Phase (\$LINKH)

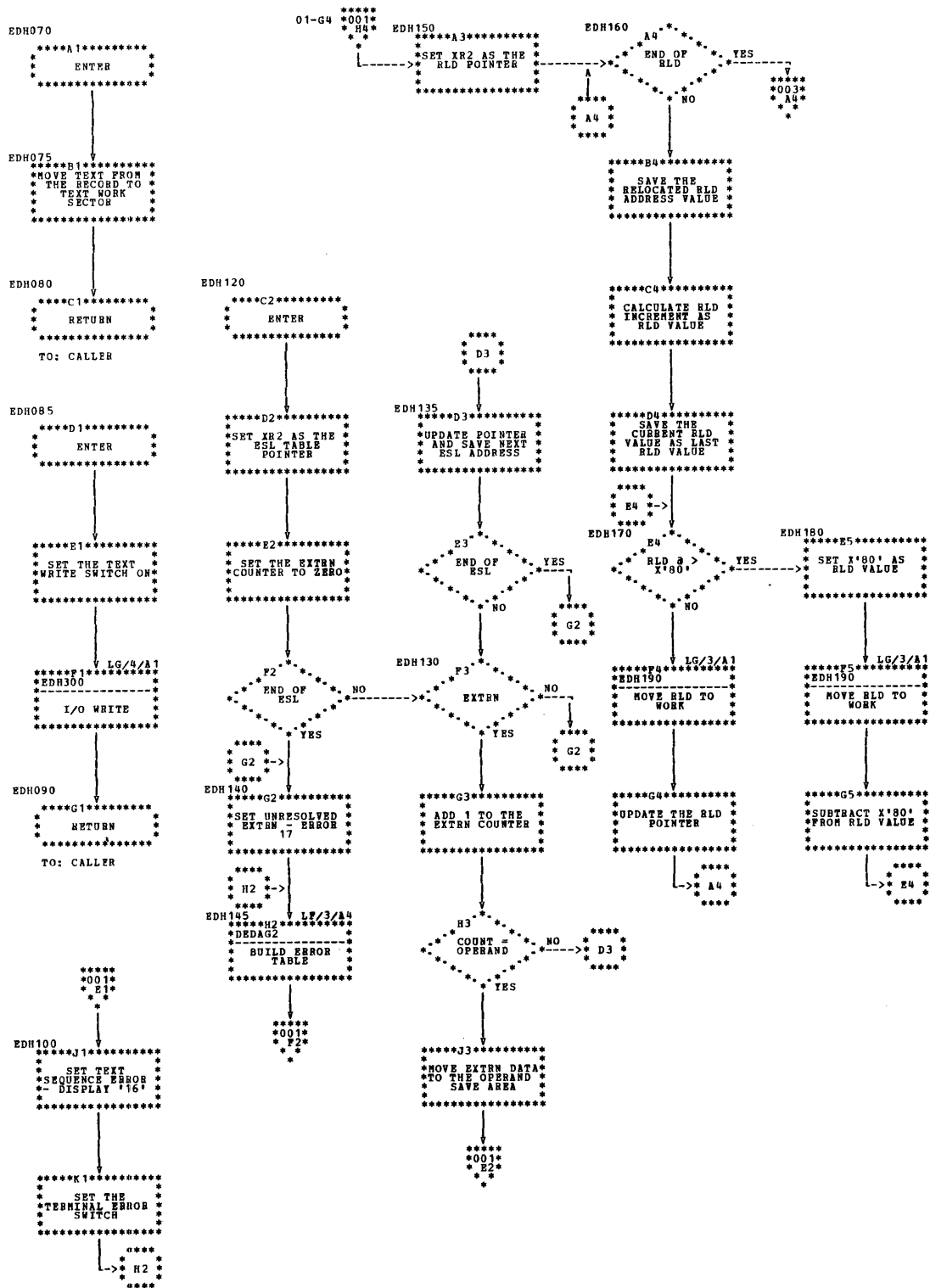


Chart LG (Part 2 of 4). PASS2 TEXT-RLD Conversion Phase (\$LINKH)

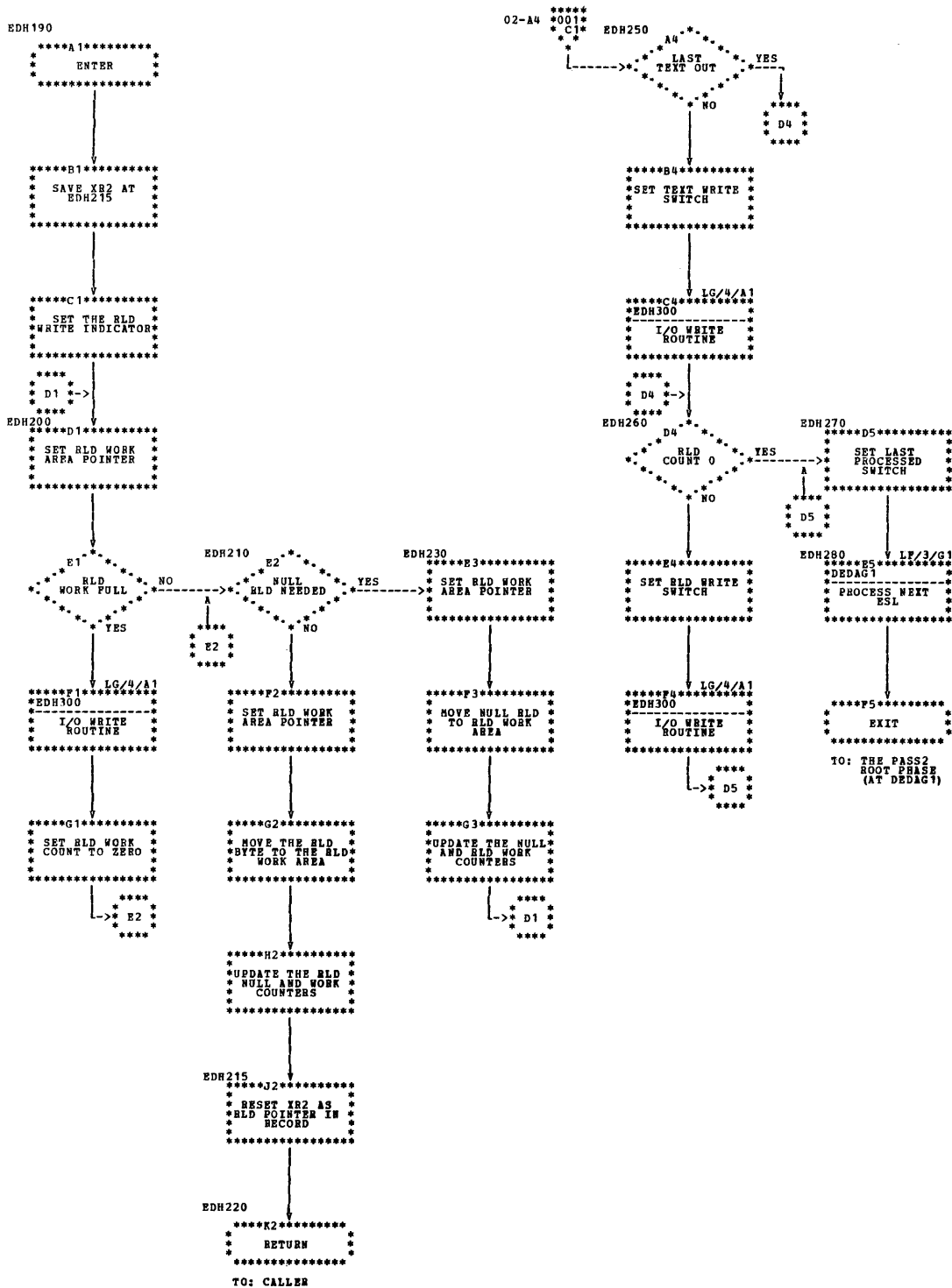


Chart LG (Part 3 of 4). PASS2 TEXT-RLD Conversion Phase (\$LINKH)

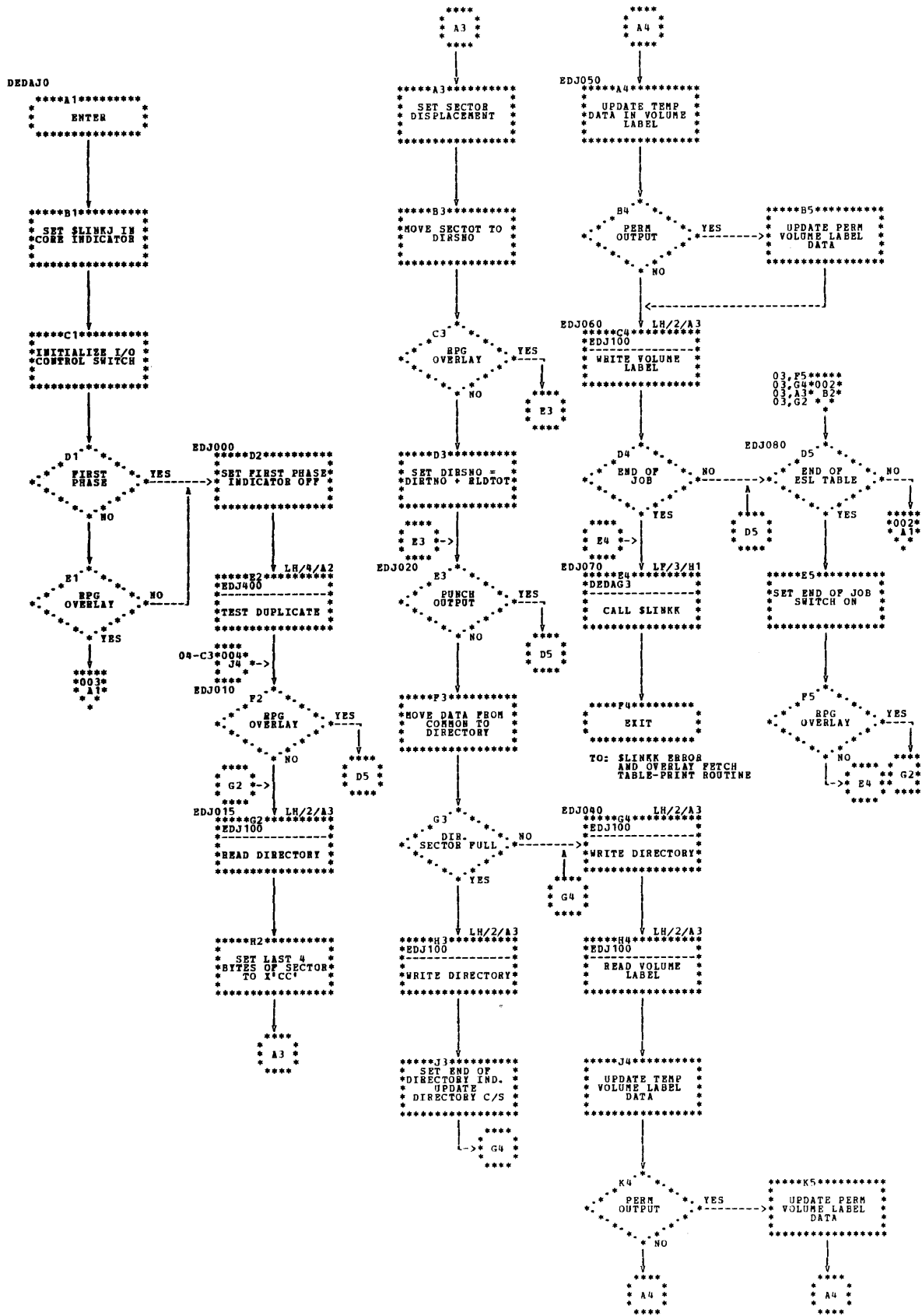
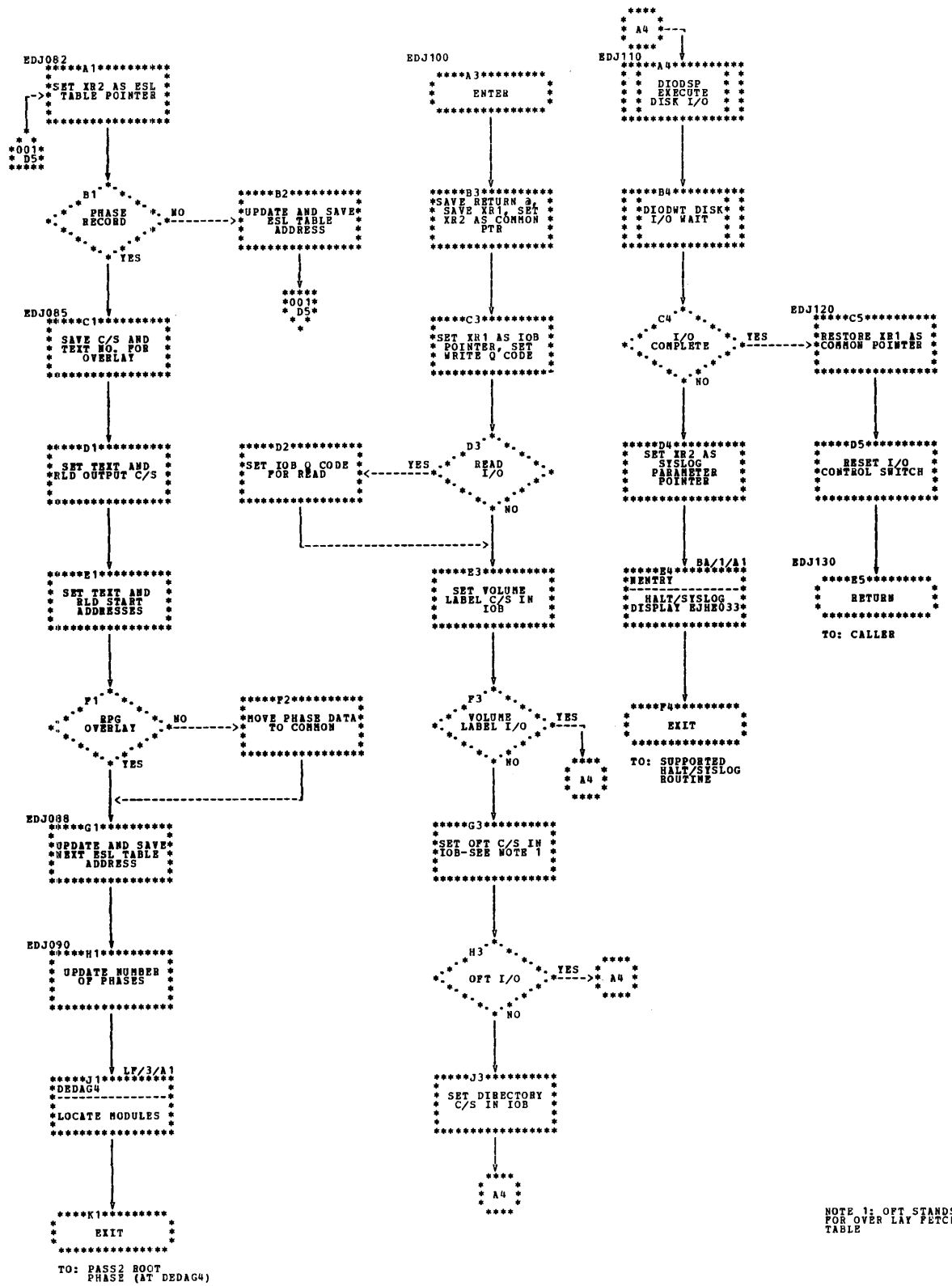


Chart LH (Part 1 of 4). PASS2 Directory Build Phase (\$LINKJ)



NOTE 1: OPT STANDS FOR OVER LAY FETCH TABLE

Chart LH (Part 2 of 4). PASS2 Directory Build Phase (\$LINKJ)

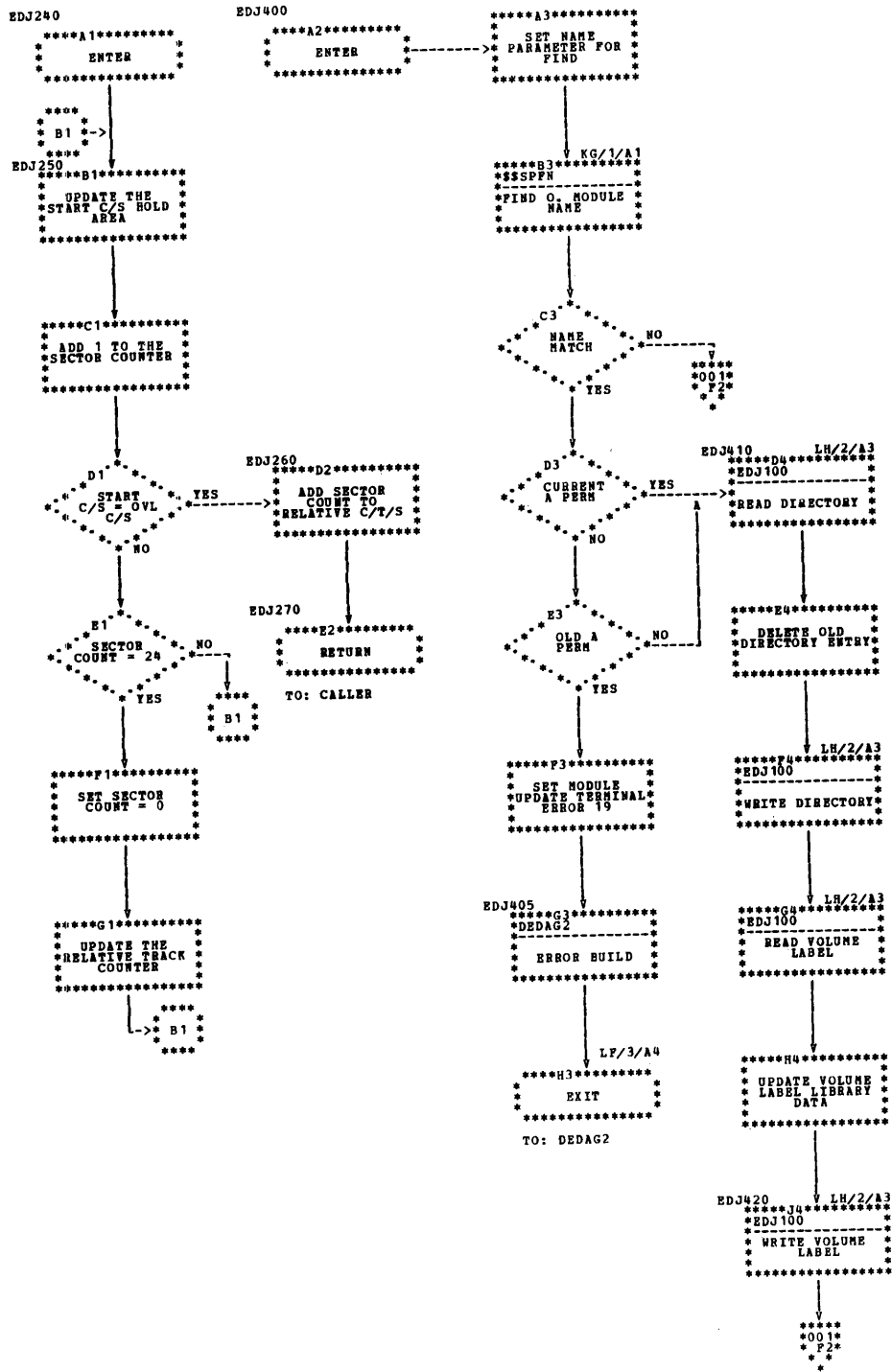


Chart LH (Part 4 of 4). PASS2 Directory Build Phase (\$LINKJ)

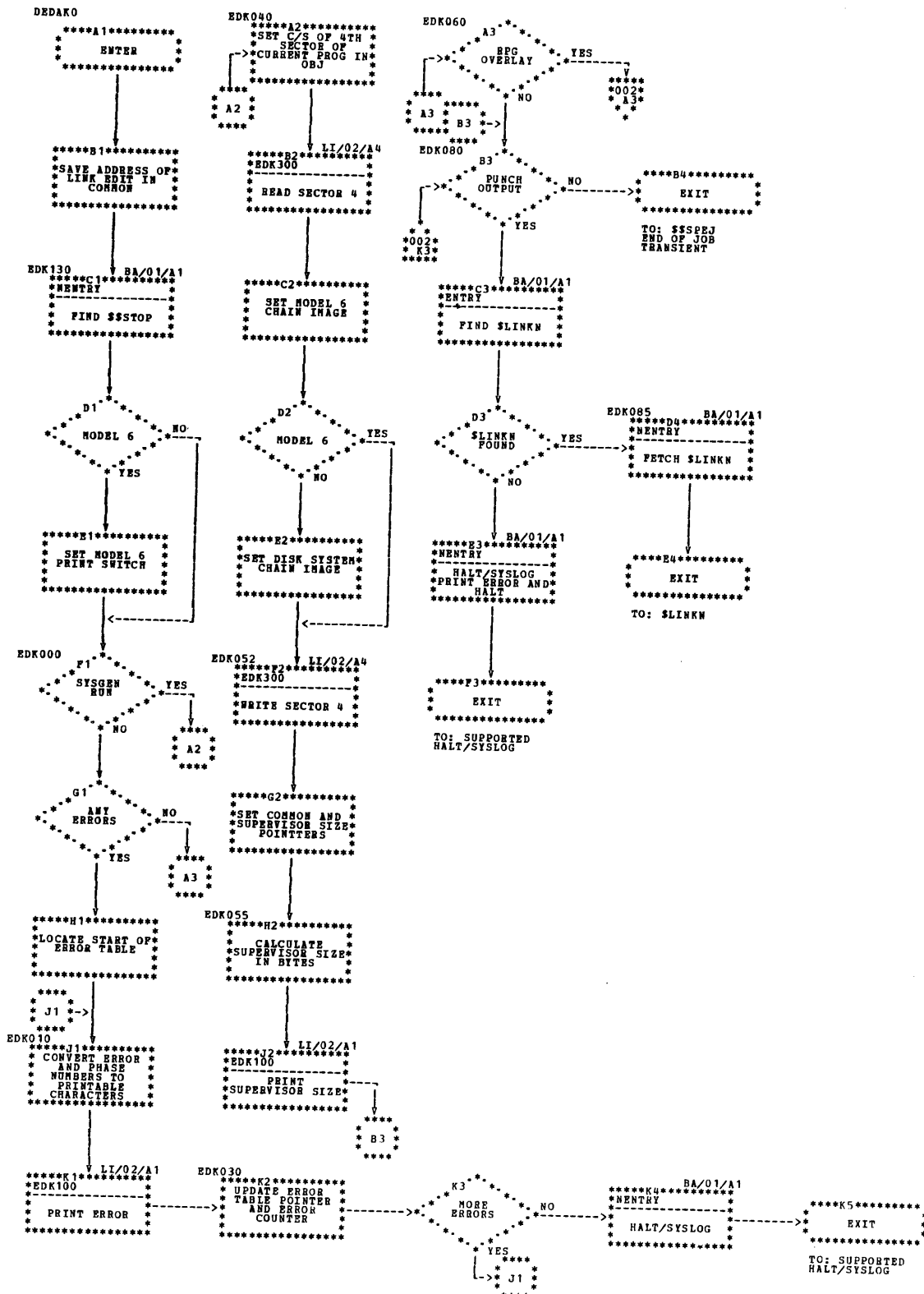


Chart LI (Part 1 of 2). Error and Overlay Fetch Table—Print Phase (\$LINKK)

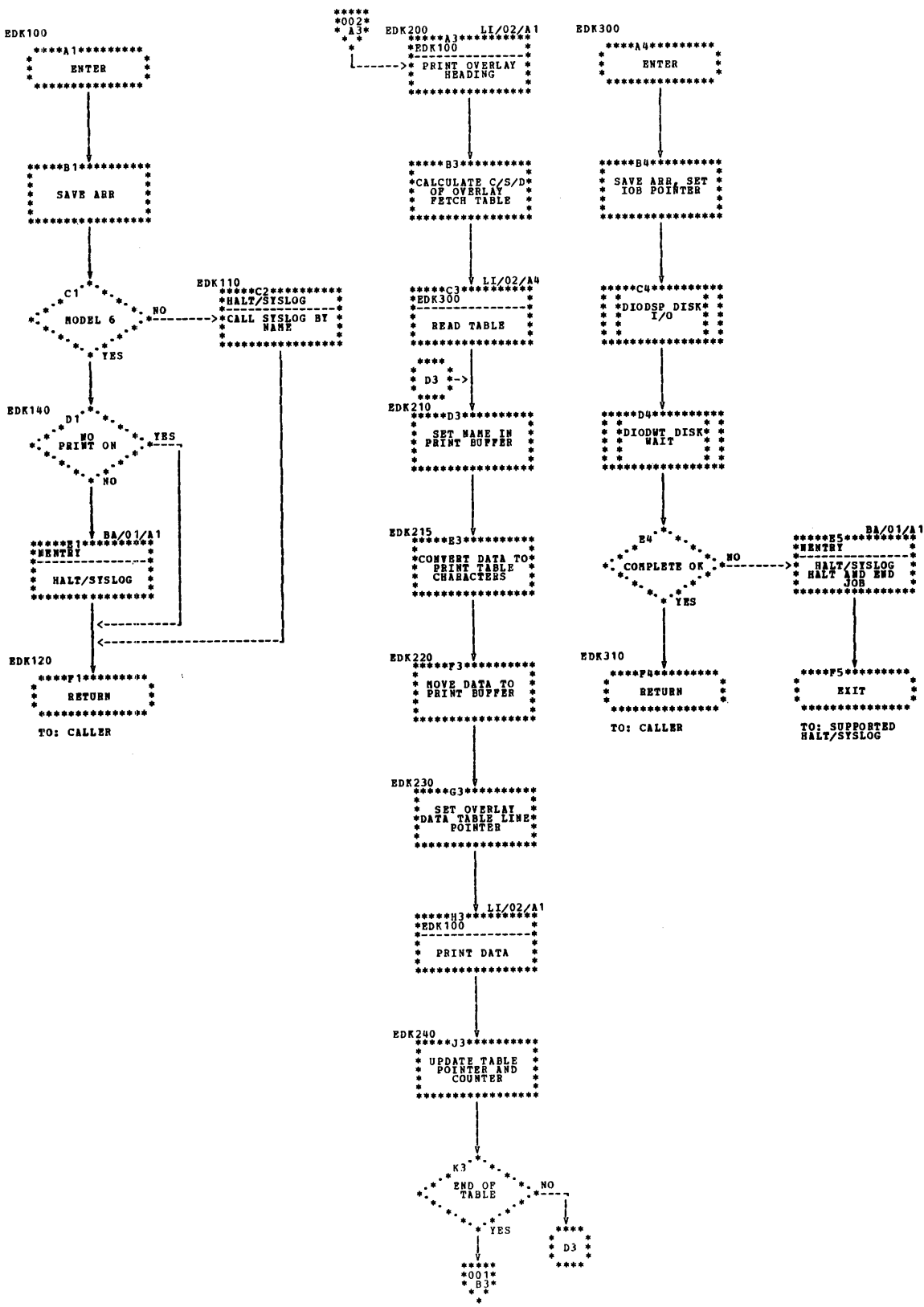


Chart LI (Part 2 of 2). Error and Overlay Fetch Table-Print Phase (\$LINKK)

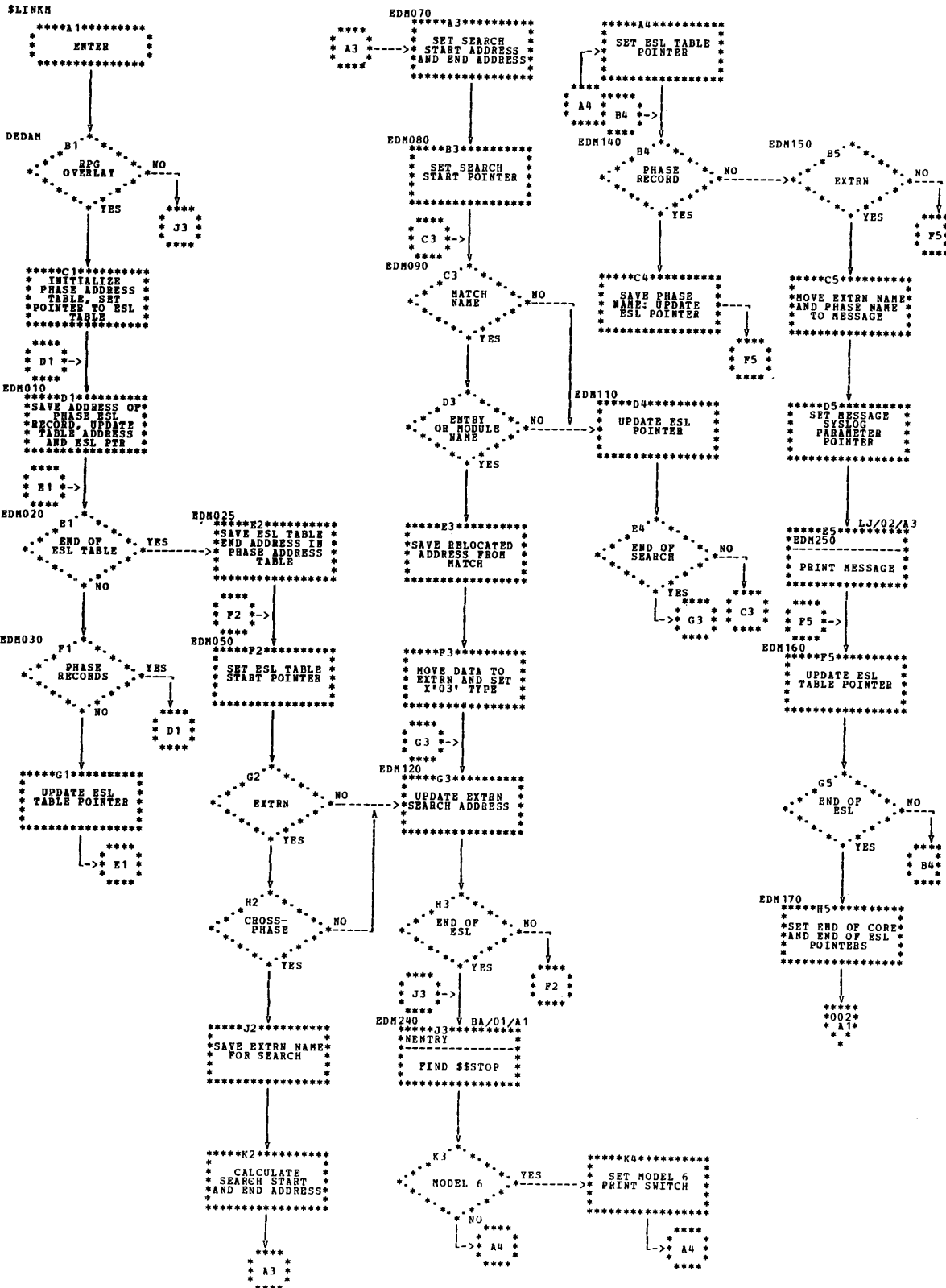


Chart LJ (Part 1 of 2). External Symbol List (ESL) Table-Process Phase (\$LINKM)

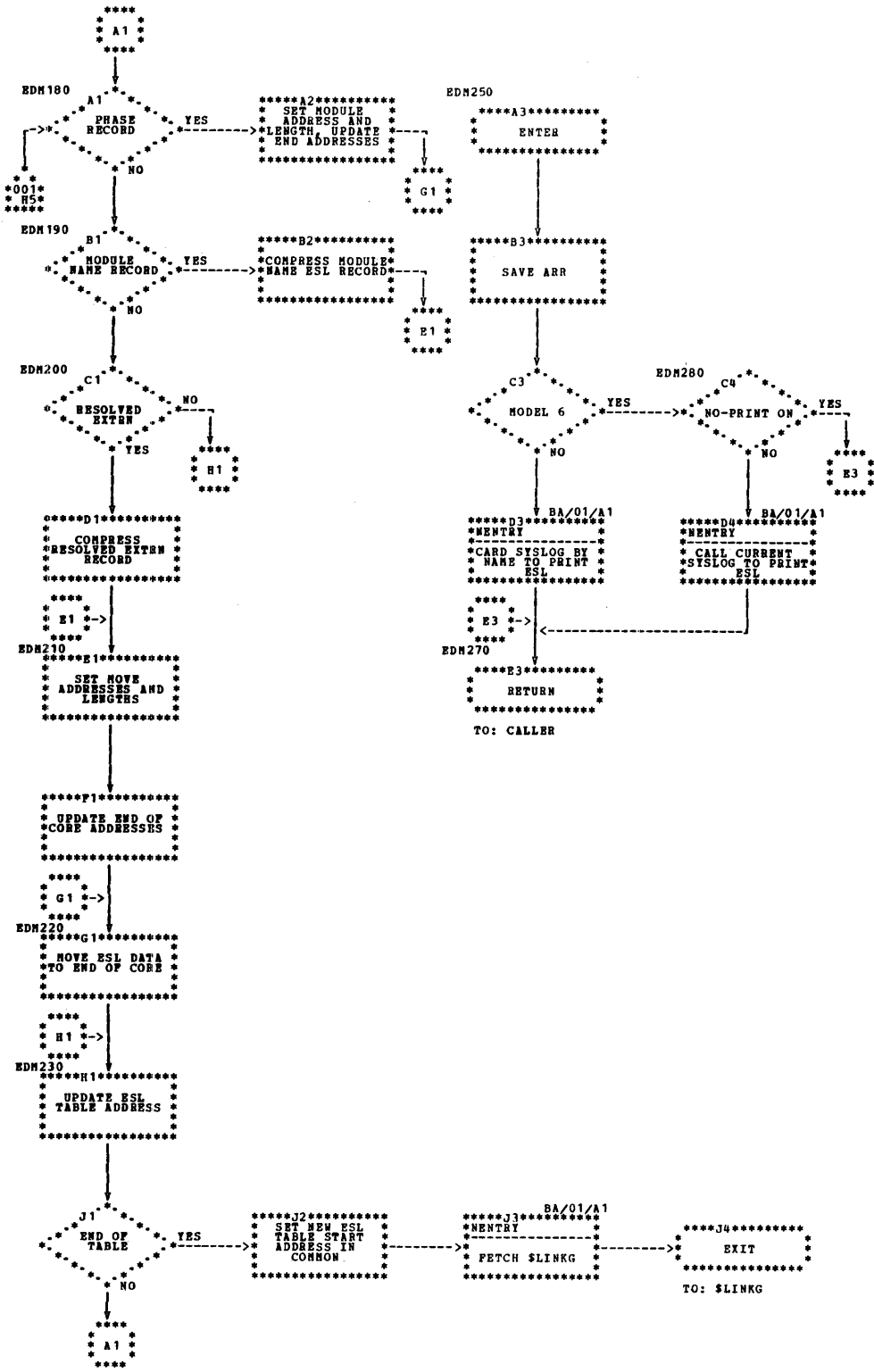


Chart LJ (Part 2 of 2). External Symbol List (ESL) Table-Process Phase (\$LINKM)

SLINKN

*****A1*****
* ENTER *

*****B1*****
* SAVE ADDRESS OF *
* COMMON *
* INITIALIZE *
* COMMON AREA FOR *
* PUNCH *

C1
MODEL 6

BA/01/A1
*****C2*****
* NENTRY *

BA/01/A1
*****D1*****
* NENTRY *

*****E1*****
* OPEN HFCU OR *
* DATA RECORDER *

F1
OPEN OK

B5

*****G1*****
* SET PUNCH JOB *
* TO PRINT AND *
* PUNCH CARDS *

*****H1*****
* MOVE HEADER *
* NAME, TYPE AND *
* SEQUENCE TO *
* PUNCH *

*****J1*****
* SET POINTERS TO *
* MOVE DIRECTORY *
* DATA TO BUFFER *

A3

EDN020

*****G2*****
* ENTER *

*****H2*****
* SET NUMERIC *
* ZONE ON *

J2
BYTE >
I'F9'

*****J3*****
* SET BYTE TO *
* ALPHA VALUE *

EDN030

*****K2*****
* RETURN *

TO: CALLER

EDN010

*****A3*****
* MOVE DATA FROM *
* COMMON TO PUNCH *
* BUFFER *

A3

LK/01/G2

EDN020
*****B3*****
* CONVERT DATA *

*****C3*****
* UPDATE THE DATA *
* AND BUFFER *
* POINTERS *

D3
END OF DATA

LK/01/A1
*****D4*****
* CHKSUM *

*****E3*****
* RESET THE DATA *
* POINTER IN *
* COMMON *

A3

PCHCHK

LK/01/F4
*****E4*****
* PCHCHK *

*****F4*****
* ENTER *

*****G4*****
* SAVE RETURN *
* ADDRESS AT *
* PCHEND *

*****H4*****
* SET THE PUNCH *
* JOB POINTER *

*****J4*****
* \$\$\$PCH1 *
* PUNCH THE CARD *

F5

BADPCH

*****A5*****
* ENTER *

B5

*****B5*****
* SET SYSLOG *
* PARAMETER *
* POINTER *

BA/01/A1
*****C5*****
* NENTRY *

*****D5*****
* HALT *

F5

P5
PUNCH OK

*****G5*****
* CLEAR THE PUNCH *
* AREA TO BLANKS *

PCHEND
*****H5*****
* RETURN *

TO: CALLER

Chart LK (Part 1 of 4). Linkage Editor Punch Output Processor (\$LINKN)

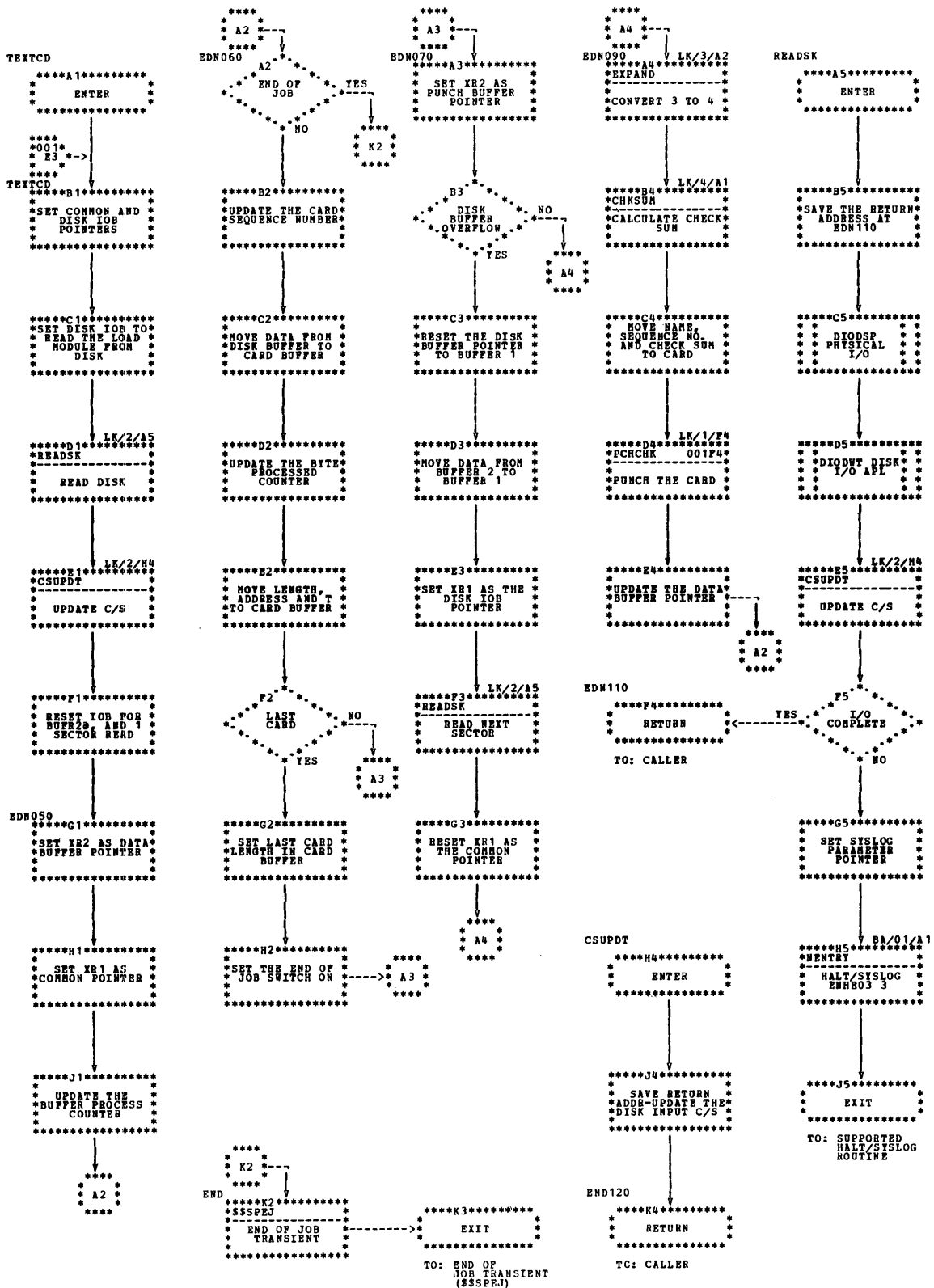


Chart LK (Part 2 of 4). Linkage Editor Punch Output Processor (\$LINKN)

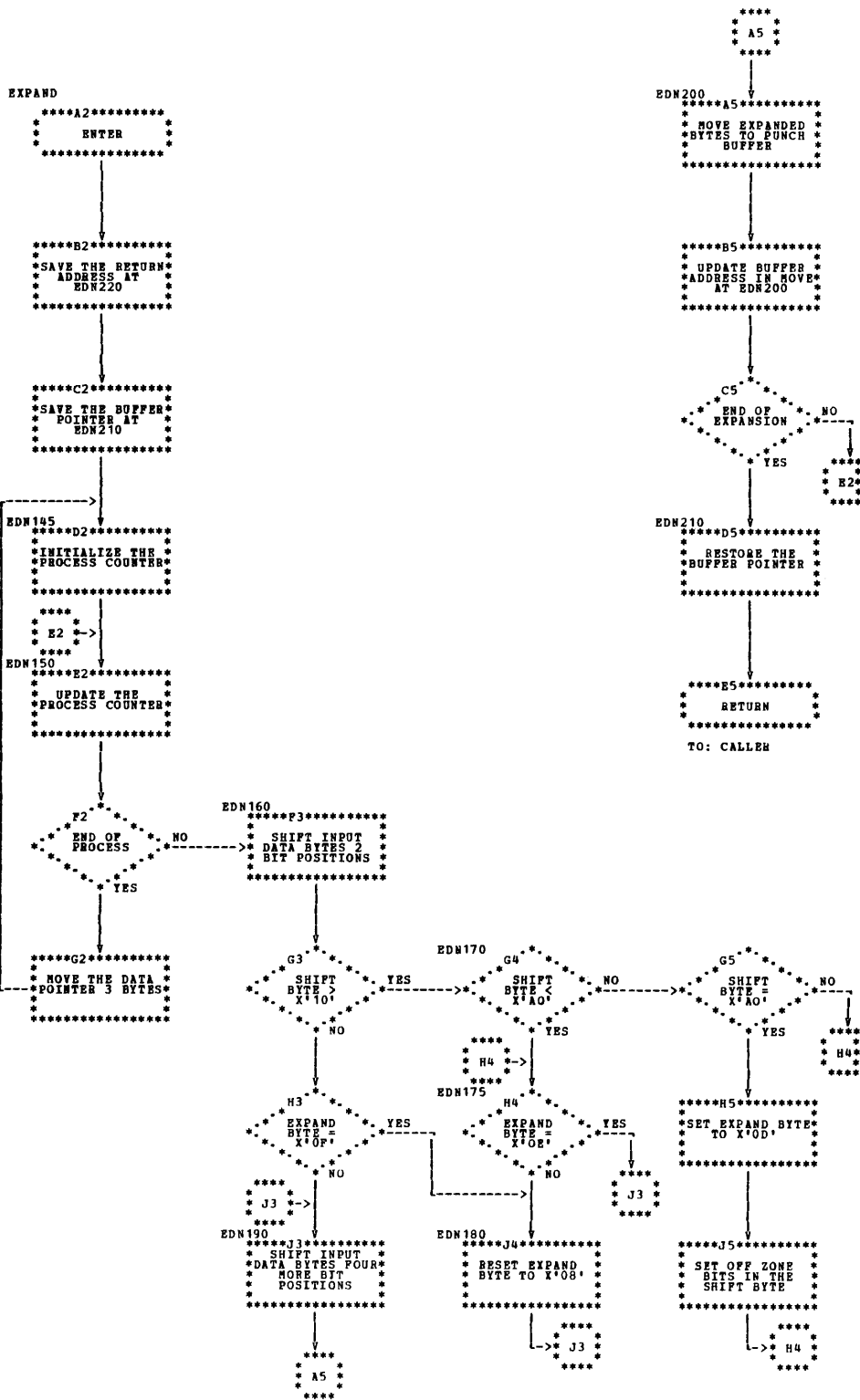


Chart LK (Part 3 of 4). Linkage Editor Punch Output Processor (\$LINKN)

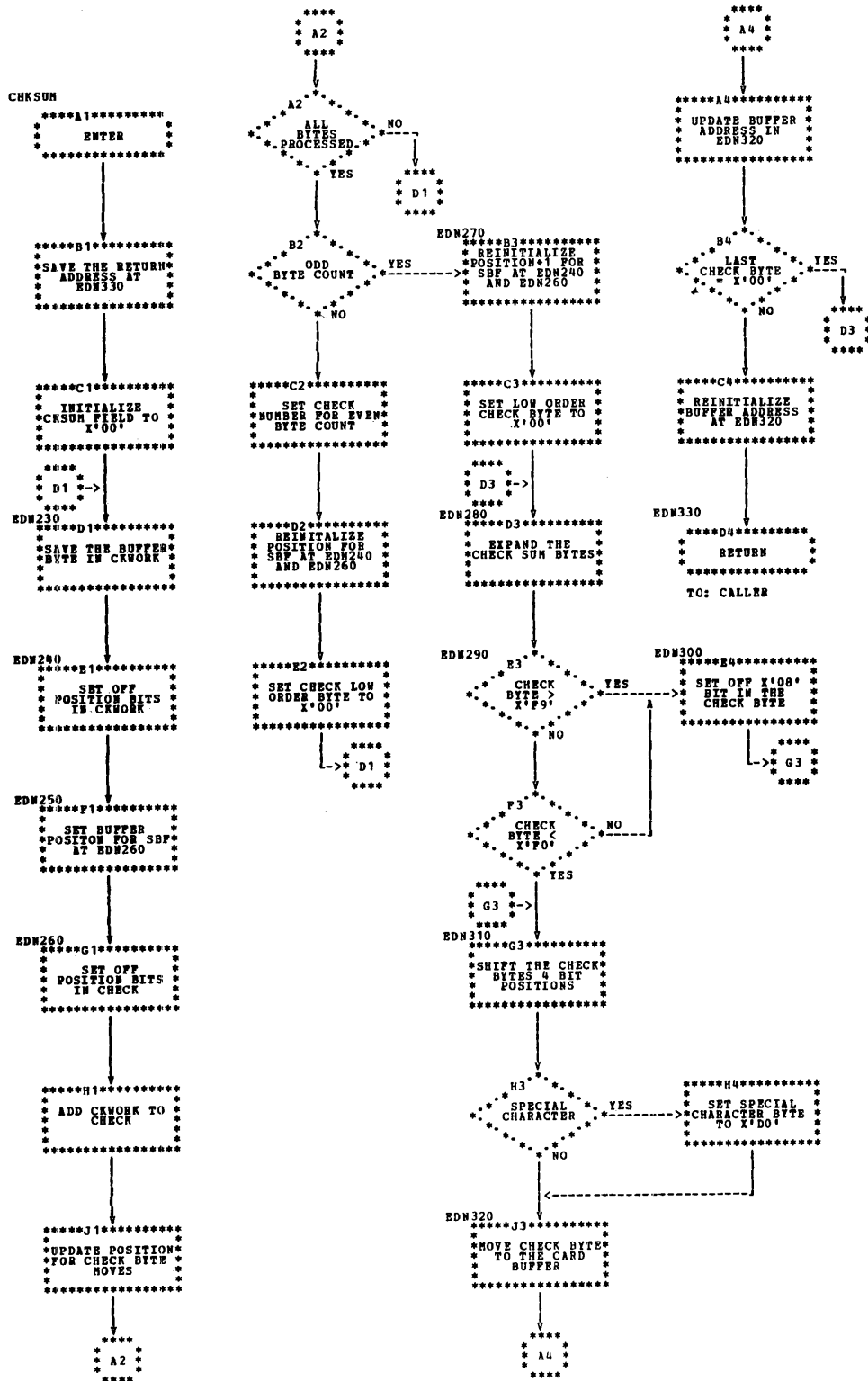


Chart LK (Part 4 of 4). Linkage Editor Punch Output Processor (\$LINKN)

Figure 7-9 contains the Directory entries for the Linkage Editor.

Module/Phase Name	Chart ID	Descriptive Name	Entry Point	Function
PASS1 \$LINKB	LA	PASS1 Root Phase	DEDAB0 DEDAB1 DEDAB2 DEDAB3 DEDAB4 DEDAB5 DEDAB6 DEDAB7	<ul style="list-style-type: none"> ● Mainline entry from Initialization ● Data Management entry from overlay ● Input record checking ● Phase processor entry point ● Build error table ● Print MAPTBL errors ● Data management ● Physical I/O linkage
PASS1 \$LINKC	LB	INCLD and ENTRY Record Processor	DEDAC0	<ul style="list-style-type: none"> ● Provides syntax checking ● Processes INCLD statements ● Provides diagnostic messages ● Builds ENTRY field in object library ● Locates unresolved EXTERNS
PASS1 \$LINKD	LC	TEXT-RLD and END Record Processor	DEDAD0	<ul style="list-style-type: none"> ● Adds relocation factor to TEXT operands ● Processes END records
PASS1 \$LINKE	LD	ESL and OPTNS Record Processor	DEDAE0	<ul style="list-style-type: none"> ● Locates and provides error messages for multiply defined names and entry points ● Resolves EXTRN cross references ● Builds ESL table ● Unblocks ESL entries ● Sets phase optional attributes
PASS1 \$LINKF	LE	PHASE Record Processor	DEDAF0	<ul style="list-style-type: none"> ● Interrogates PHASE statement and builds phase directory records
PASS2 \$LINKG	LF	PASS2 Root Phase	DEDAG0 DEDAG1 DEDAG2 DEDAG3 DEDAG4	<ul style="list-style-type: none"> ● Initialization of PASS2 Root Phase ● Return point from TEXT-RLD phase (\$LINKH) ● Build error table ● Error and overlay fetch table ● Print phase (\$LINKK) calling routine ● Return point from Directory Build Phase (\$LINKJ)
PASS2 \$LINKH	LC	TEXT-RLD Conversion Phase	DEDAH0	<ul style="list-style-type: none"> ● Resolves external RLDs ● Builds TEXT and RLD work area ● Calls error build ● Writes RLD and TEXT into object library
PASS2 \$LINKJ	LH	Directory Build Phase	DEDAJ0	<ul style="list-style-type: none"> ● Builds an object library directory record for the load module ● Updates the object library volume label sector for phases with unreserved names ● Updates the overlay fetch table for phases with unreserved names
\$LINKK	LI	Error and Overlay Fetch Table—Print Phase	DEDAK0	<ul style="list-style-type: none"> ● Prints Supervisor size for Sysgen ● Prints the error table ● Provides the chain image for Sysgen program ● Prints the overlay fetch table for the RPG overlay program
\$LINKM	LJ	ESL Table—Process Phase	DEDAM0	<ul style="list-style-type: none"> ● Prints unresolved EXTRNS ● Prints the ESL table ● Attempts to resolve RPG cross-phase EXTRNS ● Compresses the ESL table into high end of main storage
\$LINKN	LK	Punch Processor	DEDAN0	<ul style="list-style-type: none"> ● Punches a header card with information required to use the punched deck as COPY READER or // LOAD * input ● Punches a card deck with Linkage Editor information normally contained in the object library

Figure 7-9. Linkage Editor Directory



PART 8.

SYSTEM GENERATION

8

System Generation is the means by which the user can generate his IBM System/3 Disk Systems to reflect the desired options that are supported by his current machine configuration. The initial input to the System Generation program consists of these control statements:

Disk System

1. // DATE mmddy
2. // CALL \$SGPCH,R1
3. // RUN

Model 6 System

1. DATE mmddy
2. CALL \$SGCAL, R1

Disk System

After the system reads these user-supplied control cards, it punches a card deck. The punched deck is then modified by the user to reflect the desired, supported options. The user then reloads the punched deck back into the MFCU.

The System Generation program reads and uses this modified card deck to:

1. Initialize F1
2. Build the configuration record
3. Call the Linkage Editor to build a new Supervisor
4. Call the Library Maintenance program to copy the required system control programs

Model 6 System

After reading these user-supplied control statements, the system calls the procedure \$SGKBD to allow the user to select the appropriate keyboard image table for his keyboard. Procedures \$SGINT, \$SGCOR, and \$SGENB are then called to initialize F1, allow the user to save space for the BASIC system, and to obtain through operator responses, the user's hardware configuration and any optional system support desired. The System Generation program reads and uses the operator responses to:

1. Build the configuration record
2. Call the Linkage Editor to build a new supervisor
3. Call the Library Maintenance program to copy the required system control programs

CONTROL RECORDS USED BY SYSTEM GENERATION

The following control records are used exclusively by the System Generation program.

LINK—Definition Header Record

The LINK-definition header record denotes the beginning of a LINK-definition and must appear first in each LINK-definition. The format of this record is as follows:

Position	1	8	14
	Name	Operation	Operand
	Not used, must not be present	LINK	Not used, must not be present

MEND—LINK-Definition End Record

The LINK-definition end record denotes the end of and must be placed after, the LINK-definition. The format for this record is as follows:

Position	1	8	14
	Name	Operation	Operand
	Not used	MEND	Not used, must not be present

Keyword Prototype Record

A keyword prototype record enables the user to reduce the number of operands in each LINK-instruction that corresponds to the LINK-definition, and to write the operands in any order. The format for this record is as follows:

Position	1	8	14
	Name	Operation	Operand
	A symbolic parameter or not used	A symbol	Operands of the form described, separated by commas or blank

Name: The name entry of the prototype statement may be unused or it may contain a symbolic parameter.

Operation: The symbol in the operation entry is the mnemonic operation code that must appear in all LINK-instructions that refer to this LINK-definition.

Operand:

- Each operand must consist of a symbolic parameter, immediately followed by a dash and optionally followed by a value.
- A value that is part of an operand must immediately follow a dash. If there is no value, a blank, or comma, and another keyword operand must follow the dash.
- Anything that may be used as an operand in a LINK-instruction except variable symbols, may be used as a value in a keyword prototype record.
- The symbolic parameters are used in the LINK-definition to represent the name entry and operands of the corresponding LINK-instruction.

System Configuration Statement

The system configuration statement is used by the user to specify the values for the keywords that appear in the operand of the keyword prototype record. The format for the system configuration statement is as follows:

Position	1	8	14
	Name	Operation	Operand
	A symbol, or not used	Mnemonic	Zero or more operands of the form described, separated by commas

Name:

- If a symbolic parameter appears in the name entry of the keyword prototype record, and the name entry of the LINK record contains a symbol, the symbolic parameter is replaced by the symbol. If the name entry of the LINK record is unused, the symbolic parameter is replaced by a null character value.
- If a symbolic parameter appears in the operand of the keyword prototype record, and the LINK record contains a keyword that corresponds to the symbolic parameter, the value assigned to the keyword replaces the symbolic parameter.
- If a symbolic parameter was assigned a value by a keyword prototype record, and the LINK record does not contain a keyword that corresponds to the symbolic parameter, the standard value assigned to the symbolic parameter is replaced by a null value with an undefined attribute.

Operand:

- Each operand consists of a keyword immediately followed by a dash sign and an optional value. Nested keywords are not permitted.
- A keyword consists of one through six letters and digits, the first of which must be a letter.
- The keyword part of each keyword LINK record operand must correspond to one of the symbolic parameters that appears in the operand of the keyword prototype record. A keyword corresponds to a symbolic parameter if the characters of the keyword are identical to the characters of the symbolic parameter that follow the ampersand.
- The operands in a system configuration statement may be written in any order. If an operand appeared in a keyword prototype record, a corresponding operand does not have to appear in the system configuration statements.

The following is an example of a complete LINK-Definition. Note that the symbolic parameters that are found in the model records appear in the keyword prototype record and were originated by the user's system configuration statements. The model records are generated by the system and change only to reflect the values found in the keyword prototype record.

Position	1	8	14
Statement	Name	Operation	Operand
Header		LINK	
Prototype	&NAME	MOVE	&P=&S=&R1=&R2=
Model	&NAME	ST	&R1,&S.(,&R2)
Model		L	&R1,&P.B
Model		ST	&R1,&P.A
Model		L	&R1,&S.(,&R2)
END		MEND	

TABLE Record

A TABLE record is used to assign a value to its associated variable symbol. The format for this record is as follows:

Position	1	8	14
	Not used	TABLE	A variable symbol

TABLE Definition Record

The TABLE definition records are used to define the values that are to be assigned by the TABLE record. At least one TABLE definition record must follow a TABLE record. The format for the TABLE definition record is as follows:

Position	1	13	14
	Argument	b	Value

TEXT Record

The TEXT record must be specified and is used to indicate the beginning of the conditional processing instructions. The valid records that can appear ahead of the TEXT record in the job stream are: LINK, LINK-prototype, GBLB, LCLB, TABLE and TABLE definition records. If any of these records appear after the TEXT record, they will be considered invalid and an error will result. The format for the TEXT record is as follows:

Position	1	8	14
	Name	Operation	Operand
	Not used	TEXT	Not used

AGO Record—Unconditional Branch

The AGO record is used to unconditionally alter the sequence in which the source library LINK-definition records are processed by the System Generation Pre-Processor program. The format for the AGO record is as follows:

Position	1	8	14
	Name	Operation	Operand
	A sequence symbol or not used	AGO	A sequence symbol

ANOP Record—Assembly No Operation

The ANOP record facilitates conditional and unconditional branching to records named by symbols or variable symbols located within the name field. The format for the ANOP record is as follows:

Position	1	8	14
	Name	Operation	Operand
	A sequence symbol	ANOP	Not used, must not be present

AIF Record—Conditional Branch

The AIF record is used to alter conditionally the sequence in which the LINK-definition records are processed. The format for the AIF record is as follows:

Position	1	8	14
	Name	Operation	Operand
	A sequence symbol or not used	AIF	A logical expression enclosed in parentheses, immediately followed by a sequence symbol

The sequence symbol, located within the AIF operand, must immediately follow the closing parenthesis of the logical expression. The logical expression in the operand evaluated to determine if it is true or false. If the expression is true, the record named by the sequence symbol in the operand is the next record processed by System Generation Pre-Processor. If the expression is false, the next sequential LINK-definition record is processed.

Whenever a compare is issued between operands of the AIF statement whose lengths (after substitution of variable symbols), are unequal, the compare will be the result of a length comparison and not the operand contents.

This means that an operand of 001 would be greater than an operand of 99. Since no attribute checking is done by the System Generation Pre-Processor for AIF statement operands (other than for options 1 and 2 below), an operand of '001' would compare equal to an operand of 001. Note also that the quotes surrounding a character string do not count toward the length and do not appear in the operand being compared.

Option

1 Type Attribute (T') Checking: The following conditions are checked when option 1 is specified.

Note: The user may refer to the type attribute only in the operand of the AIF record.

Condition	Meaning
AIF (T'&name { NE } D) .A { EQ }	&name numeric?
AIF (T'&name { NE } B) .B { EQ }	&name binary?
AIF (T'&NAME { NE } C) .D { EQ }	&NAME character
AIF (T'&NAME { NE } ") .E { EQ }	&NAME specified—i.e. null attribute
AIF (T'&name { NE } 'T'&NAME1) .C { EQ }	Do &NAME and &NAME1 have the same attributes
<i>Note:</i> No concatenation of symbols in an AIF operand is supported when T' processing. If concatenation is specified, it will terminate the Pre-Processor.	

2 Binary Condition Checking

AIF (& Symbol) . name
<i>Note:</i> Valid only if & symbol is a binary set symbol.

3 Value Checking

AIF $\left(\left\{ \begin{array}{l} \& \text{symbol} \\ \text{'char string'} \\ \text{number} \end{array} \right\} \& \left\{ \begin{array}{l} \text{GT} \\ \text{GE} \\ \text{EQ} \\ \text{NE} \\ \text{LT} \\ \text{LE} \end{array} \right\} \& \left\{ \begin{array}{l} \& \text{symbol} \\ \text{'char string'} \\ \text{number} \end{array} \right\} \right) . \text{seq symbol}$
<i>Note:</i> & symbol = any variable symbol 'char string' = up to 60 characters, enclosed in quotes, number = up to 60 digits (0-9) GT = greater than GE = greater than or equal EQ = equal NE = not equal LT = less than LE = less than or equal

GBLB or LCLB Set Records

Local set symbols are defined when they appear within the operand entry of a LCLB record. Global set symbols are defined when they appear within the operand entry of a GBLB record. The format for both the LCLB and GBLB records is as follows:

Position	1	8	14
Name	Operation	Operand	
Not used, must not be present	GBLB or LCLB	One or more variable symbols that are to be used as global or local binary SET symbols separated by commas	

SETB Record—Set Binary

The SETB record may be used to assign the binary value of 0 or 1 to a SETB symbol referenced within the name field. The format for the SETB record is as follows:

Position	1	8	14
Name	Operation	Operand	
A SETB symbol	SETB	A 0 or 1	

Comment Records

The comment record is used to generate comments just as other model records are used to generate program control, OCL, or other record types. The format for the comment record is as follows:

Position	1	8	14
Name	Operation	Operand	
* THIS STATEMENT WILL BE GENERATED * THIS ONE WILL NOT BE GENERATED			

MEXIT Record—Link-Definition Exit

The MEXIT record is used to indicate that processing of a LINK-definition is to be terminated with this record. The format for the MEXIT record is as follows:

Position	1	8	14
Name	Operation	Operand	
A sequence symbol or not used	MEXIT	Not used, must not be present	

MNOTE Record

The MNOTE record may be used to generate a message and to indicate what error severity, if any, is to be associated with the printed message. The format for the MNOTE record is as follows:

Position	1	8	14
Name	Operation	Operand	
A sequence symbol or not used	MNOTE	SC or SC, 'message'	

The operand entry of the MNOTE assembler-instruction may be written in one of the following forms:

1. Severity-code
2. Severity-code, 'message'

Severity code — a 2-digit response — any characters other than blank are valid.

Message — up to 60 characters, enclosed in quotes.

Model Records

The model records are those records that are generated by the System Generation Pre-Processor program to be executed in such a manner that the user specified system will be generated. The format of the model record is as follows:

Position	1	8	14
Name	Operation	Operand	
Unused, symbol, sequence symbol, or variable	Any non-LINK processor statement		

SYSTEM RESIDENCE DISK PACK ORGANIZATION

The user of the IBM System/3 Disk Systems can have three types of disk packs:

1. A system pack – This disk pack contains the system, IPL programming and system libraries (object and source). If the user has ordered co-resident systems (System/3 BASIC and disk system management for Model 6 only), both systems reside on the same pack. See *IBM System/3 Model 6 BASIC Interactive System Program Logic Manual*, LY34-0001.

2. User library pack – This disk pack contains just libraries without the system.
3. User data pack – This disk pack contains the user's data without either libraries or the system.

Figure 8-1 illustrates the organization of the user's system disk pack. This diagram is a directory to the various parts of this book that describe the individual areas in greater detail.

Cylinder		Sector
0	IPL Bootstrap—(START)	0
0	Configuration Record—See Figure A-5 for Configuration format	1
0	Volume Label System Directory—See Figure A-6 for directory format	2
0	Error Logging Area	3-8
0	Volume Table of Contents—See Figures A-8 and A-9 for VTOC format	9-23
0	Reserved	24-35
0	Program Logging Sectors	36-38
0	Nucleus Initialization Program—(IPLNIP)	39-46
0	PTF Logging Sector	47
1-3	Alternate Tracks	
	<i>BASIC System Work Area and BASIC System Program File (co-resident system). Present only if user has a co-resident system.</i>	
4	Source Library Directory—See Figure A-26 for directory format	0 +
—	Source Library—See Figure A-27 for source library entry format	
—	Scheduler Work Area—See Figure A-7 for SWA format	
—	Object Library Directory—See Figure A-20 for directory format	
—	Object Library—See <i>Part 2 Linkage Editor</i> of this document	
—	User Area	

Figure 8-1. Map of the System Residence Organization

SYSTEM REQUIREMENTS

The following devices are required to support the System Generation program.

Disk System:

- One 5203 Line Printer (Model A1)
- One 5410 Processing Unit (Model A13)
- One 5424 Multi-Function Card Unit (Model 1)
- One 5444 Disk Storage Drive (Model 2)

Model 6 System:

- One 2275 Keyboard (Model 3)
- One 5213 Printer (Model 3)
- One 5406 Processing Unit (8K)
- One 5444 Disk Storage Drive (Model 2)

The overall flow of logic for the System Generation program is illustrated in Figure 8-2. As indicated, the System Generation—Phase One (\$\$GEN for Disk System or \$\$GENB for Model 6 System) routine obtains the user's system configuration from the responses on the control cards. Phase one in turn passes this information on to the first phase of the System Generation Pre-Processor Mainline (\$\$GROC). This mainline (\$\$GROC) remains in main storage throughout phase one of System Generation. The mainline (\$\$GROC) calls in five other routines to perform the following functions:

- \$\$GMN1—Builds the variable symbol table
- \$\$GMN2—Updates the variable symbol table
- \$\$GMN3—Controls processing of action statements
- \$\$GAIF—Processes the action control statements
- \$\$GMST—Translates the model statements

The configuration output from \$\$GROC is recorded in a file on disk file F1. This file is then used as the input to the System Generation—Phase Two (\$\$GEN1) routine. This phase creates a new configuration record, builds the procedures necessary for any optional SCP support generation and produces the linkage editor input file (\$\$WORK).

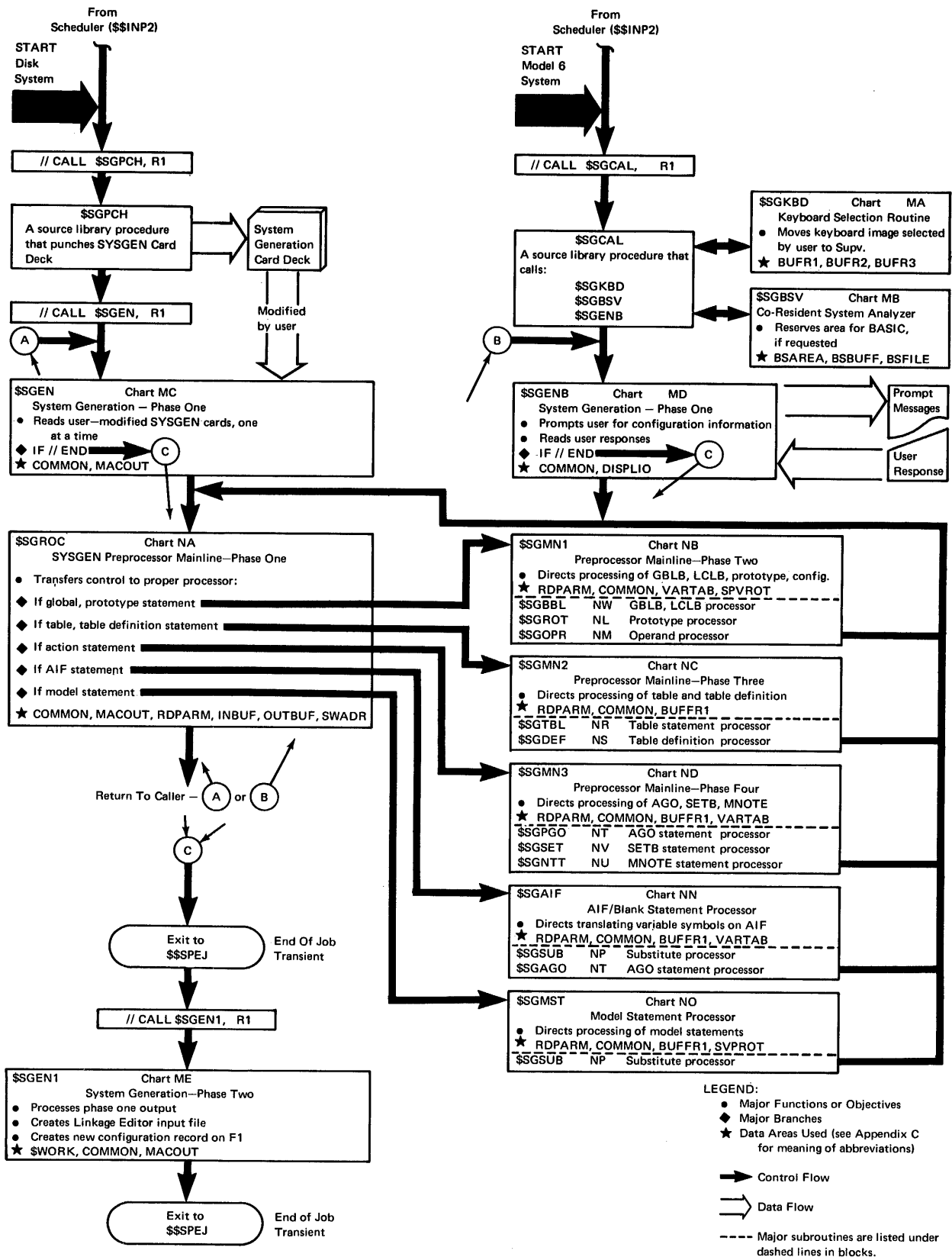


Figure 8-2. Main Logic of System Generation for Disk and Model 6 Systems

This section describes in detail the organization of the System Generation routines shown in Figure 8-2. Included in the following discussions are:

- Description containing: name of the routine, input, output, function, exits (both normal and error) from the routine.
- Main storage maps showing the routines that are in main storage at the same time as the routine being described.
- Flowchart showing the functional flow of the routine being described.

SYSTEM GENERATION

► **Keyboard Selection Routine (\$SGKBD)—Model 6 System Routine Only**

CHART: MA
 FIGURE: 8-3
 ENTRY POINT: SGKBD
 FUNCTION:

- Allows the user to select the appropriate 64-byte EBCDIC keyboard image table for the keyboard.
- Moves the chosen table to the keyboard area in the supervisor.
- Sets the appropriate byte in the configuration record.

INPUT: A response from the operator (a digit of 1 through 9) indicating one of the following tables:

1. Domestic (English)
2. Austria/Germany
3. Belgium/France
4. Denmark
5. Norway
6. Finland/Sweden
7. Spain
8. Brazil/Portugal
9. United Kingdom

(See *Appendix A. Data Area Formats* for a complete listing of the tables.) If no response is given, the default value is to the image table already in use.

OUTPUT: The appropriate translate table is in the supervisor.

- EXIT:
- Normal: End of Job Transient (\$\$SPEJ)
 - Error: Halt/Syslog Routine

► **Co-Resident System Analyzer (\$SGBSV)—Model 6 System Routine Only**

CHART: MB
 FIGURE: 8-3
 ENTRY POINT: SGBSV
 FUNCTION: Determines by OCL SWITCH parameter if a co-resident system is to be generated and saves space for BASIC if necessary. A switch setting of X'CO' means to save space for BASIC. A switch setting of X'80' means not to save space for BASIC.
 INPUT: UPSI switch
 OUTPUT: Disk space for BASIC is reserved.
 EXIT:

- Normal: End of Job Transient (\$\$SPEJ)
- Error: Supported Halt/Syslog routine

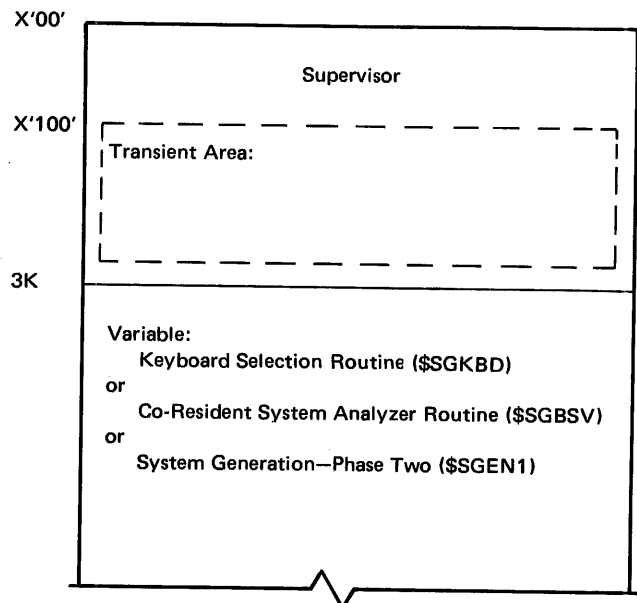


Figure 8-3. Main Storage Map for \$SGKBD, \$SGBSV, or \$SGEN1

► **System Generation—Phase One (\$\$GEN for the Disk System, \$SGENB for the Model 6 System)**

CHART: MC (Disk System)
MD (Model 6 System)

FIGURE: 8-4

ENTRY POINT: SYSGEN (Disk System)
SGENB (Model 6 System)

FUNCTION:

- \$SGENB prompts and allows the user to modify the system configuration statements.
- Reads and logs the system configuration statements, one statement at a time, into the Sysin input buffer (COMMON).
- Passes the address of this buffer to the Pre-Processor Mainline (\$SGROC).

INPUT:

- Disk System: User-modified system configuration, card deck read from the MFCU
- Model 6 System: User-modified system configuration statements via the keyboard

OUTPUT: The address of the system generation communication region (COMMON) is passed to the Pre-Processor Mainline (\$SGROC).

EXIT:

- Normal: End-of-Job transient (\$\$SPEJ) after the // END statement has been processed
- Error:
 1. If the note error switch (NOTESW) located in the system generation communication region (COMMON) is on when control is returned from the Pre-Processor Mainline (\$SGROC), \$SGEN or \$SGENB passes control to the supported Halt/Syslog routine.
 2. If there is a read error, \$SGEN or \$SGENB halts and displays the error code of '0'.

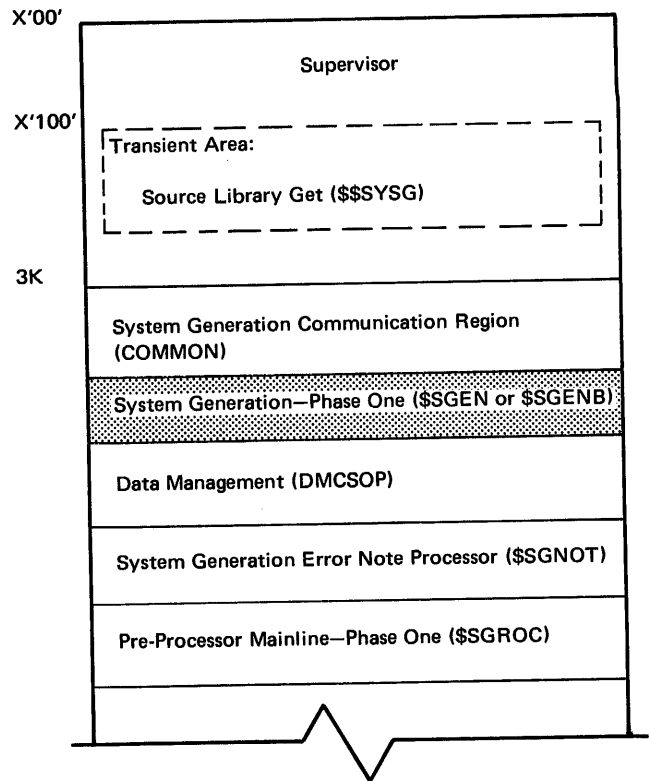


Figure 8-4. Main Storage Map for System Generation—Phase One (\$\$GEN for Disk System or \$SGENB for Model 6)

► **System Generation—Phase Two (\$\$GEN1)**

CHART: ME

FIGURE: 8-3

ENTRY POINT: GEN1

FUNCTION:

- Processes output file MACOUT of Pre-Processor.
- Creates a new configuration record on disk pack F1.
- Builds the procedure required for any optional SCP support generation.
- Creates a linkage editor input file (\$WORK) used to obtain a new Supervisor.

INPUT: Records located in the System Generation—Phase One output work file (MACOUT)

OUTPUT:

- Linkage editor input file (\$WORK) is built on disk pack F1
- The configuration record built on disk pack F1
- Procedure containing library maintenance—copy statements for optional system control programs (SCP) built on disk pack R1
- Sets the SYSGEN byte (NCSGEN) on, this byte is located in the system communication region

EXIT:

- Normal: End-of-Job transient (\$\$SPEJ)
- Error: Supported Halt/Syslog routine

► **System Generation Error Note Processor (\$\$GNOT)**

CHART: MF

FIGURE: 8-4

ENTRY POINT: \$\$GNOT

FUNCTION:

- Loads the parameter list for the supported Halt/Syslog routine with a severity code and error text (See *Halt/Syslog Procedure* in *Appendix A. Diagnostic Aids*).
- Sets the note error switch (NOTESW) located in the system generation communication region (COMMON).

INPUT:

- Register 1 contains the address of the system generation communication region (COMMON).
- The address of the note buffer is located at the label NERMES within COMMON.

OUTPUT: Address of the halt parameter list (HLTBKT) located in \$\$GNOT

EXIT: Calling routine

PRE-PROCESSOR ROUTINES

► **Pre-Processor Mainline—Phase One (\$SGROC)**

CHART: NA
 FIGURE: 8-4, 8-5
 ENTRY POINT: \$SGROC
 FUNCTION:

- Acts as the Pre-Processor Mainline driver. Once the Pre-Processor is called, this routine (\$SGROC) is loaded in main storage and remains there until all the system configuration statements have been processed.
- Calls five routines (\$SGMN1, \$SGMN2, \$SGMN3, \$SGAIF, and \$SGMST) to perform the pre-processor functions.
- Passes the address of the pre-processor communication region (RDPARM) to the above routines via register 1.

INPUT: Register 1 contains the address of the system generation communication region (COMMON).

OUTPUT: Translated model statements are written into the requester’s assigned output file (MACOUT), located on disk pack F1.

- EXIT:
- Normal: Phase one of the System Generation—Phase One (\$SGEN for Disk System or \$SGENB for Model 6)
 - Error:
 1. Error section (CALLNP), located in \$SGROC, if an error is located in the system configuration statement
 2. Error section (SETNOT), located within the Pre-Processor Mainline (\$SGROC), if a pre-processor error occurs

► **Pre-Processor Mainline—Phase Two (\$SGMN1)**

CHART: NB
 FIGURE: 8-5
 ENTRY POINT: \$SGMN1

FUNCTION: Directs the processing of the GBLB, LCLB, prototype, and system configuration statement keyword operands.

- INPUT:
- The address of the pre-processor communication region (RDPARM), in Register 1
 - The address of the system configuration statement (located within RDPARM)
 - The address of the GBLB, LCLB and prototype pre-processor statements (located within RDPARM)

OUTPUT: Keyword operands and GBLB entries are placed within the variable symbol table.

- EXIT:
- Normal: Phase one of the Pre-Processor Mainline (\$SGROC)
 - Error: Error section (SETNOT located within the Pre-Processor Mainline (\$SGROC)

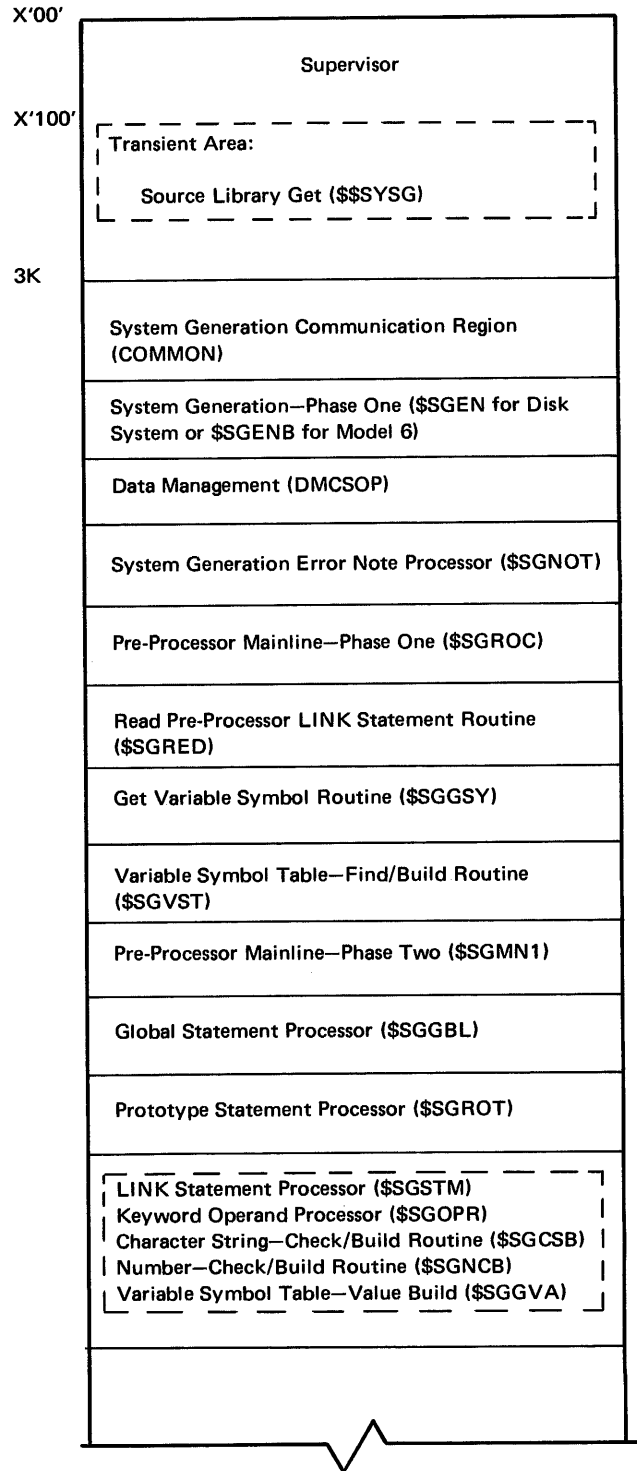


Figure 8-5. Main Storage Map for Phase Two Routines

► Pre-Processor Mainline—Phase Three (\$SGMN2)

CHART: NC

FIGURE: 8-6

ENTRY POINT: \$SGMN2

FUNCTION: Directs the processing of the pre-processor TABLE and TABLE definition statements.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address (located with RDPARM) of the input buffer (BUFFR1) that contains the TABLE and TABLE definition statements

OUTPUT: Updated variable symbol table

EXIT:

- Normal: Phase one of the Pre-Processor Mainline (\$SGROC)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

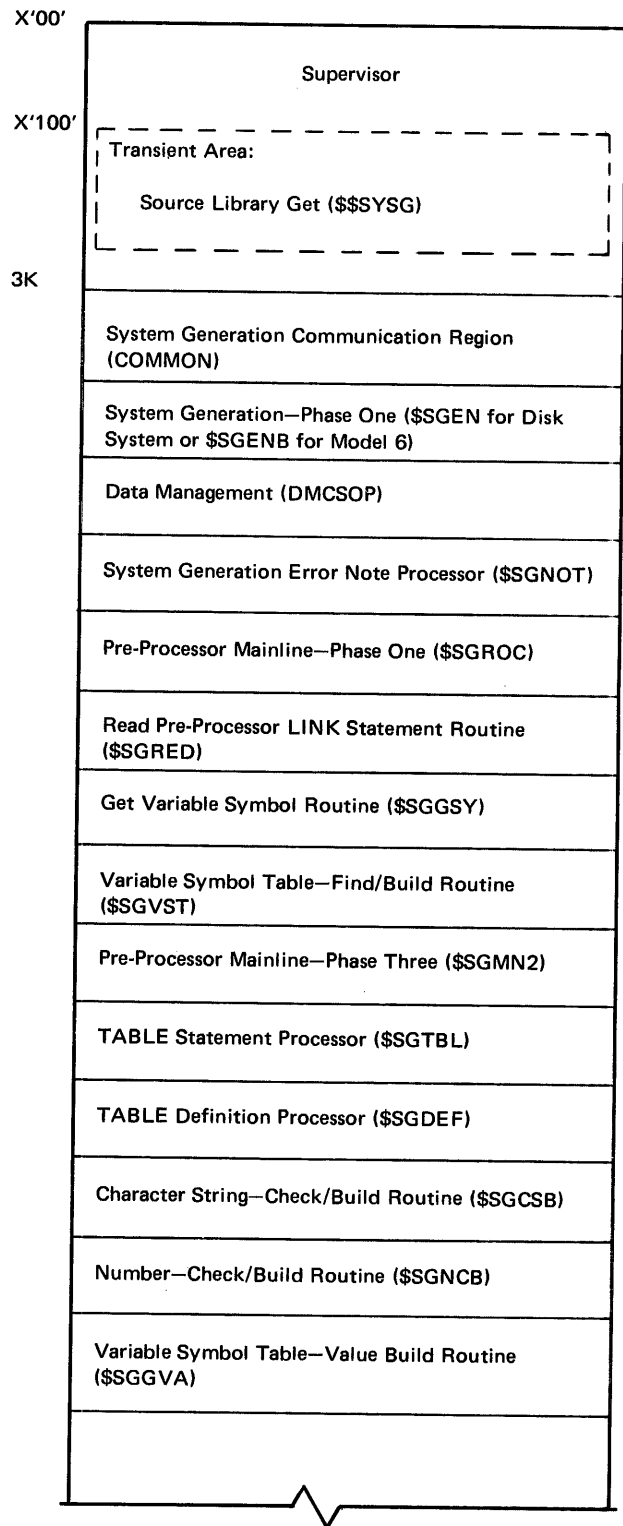


Figure 8-6. Main Storage Map for Phase Three Routines

► **Pre-Processor Mainline—Phase Four (\$SGMN3)**

CHART: ND

FIGURE: 8-7

ENTRY POINT: \$SGMN3

FUNCTION: Controls the processing of the AGO, ANOP, MEXIT, MEND, MNOTE and SETB action (conditional) processing statements.

INPUT:

- The address of the pre-processor communication region (RDPARM) in Register 1
- The address of the input buffer (BUFFR1) (located at label BUFF within RDPARM) containing the pre-processor statement

OUTPUT: The output is dependent upon the statement processed:

- AGO – Causes the Pre-Processor to read pre-processor statements from the source library until a match, for its (AGO) sequence-symbol, is found.
- MNOTE – An error message is returned via BUFFR1 located in \$SGROC.
- SETB – Searches the variable symbol table for the GBLB and LCLB referred to on the SETB statement. Sets the binary value of the found GBLB and LCLB either on or off.
- ANOP – Reads the next pre-processor statement.
- MEXIT – Returns control to caller (indicates logical end or pre-processor statements for this source library entry).
- MEND – Returns control to caller (indicates physical end of pre-processor statements for this source library entry).

EXIT:

- Normal: Phase one of the Pre-Processor Mainline (\$SGROC)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Get Variable Symbol Routine (\$SGGSY)**

CHART: NE

FIGURE: 8-5, 8-6, 8-7

ENTRY POINT: \$SGGSY

FUNCTION:

- Takes a variable symbol and builds it in an output buffer (OUTBUF). (Variable symbols start with &.)
- Computes the symbol length and stores the length along with the symbol in the output buffer (OUTBUF).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The addresses of the input (BUFFR1) and output (OUTBUF) buffers are located within the pre-processor communication region (RDPARM) at the labels BUFF and OUTBUA

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Variable Symbol Table—Find/Build Routine (\$SGVST)**

CHART: NF

FIGURE: 8-5, 8-6, 8-7

ENTRY POINT: \$SGVST

FUNCTION:

- Takes the variable symbol located in the output buffer (OUTBUF) and searches the variable symbol table for a matching entry.
- If no match is found, the build switch (VSTBSW) located in the pre-processor switch area is tested. If the switch is on, the symbol is added to the variable symbol table.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the variable symbol table, located at the label VSTST within RDPARM
- The address of the variable symbol buffer (OUTBUF), located at the label (OUTBUA) within RDPARM

OUTPUT:

- The address of the current symbol entry is placed in the pre-processor communication region (RDPARM).
- The current symbol entry is placed within the variable symbol table.

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Variable Symbol Table—Value Build Routine (\$SGGVA)**

CHART: NG

FIGURE: 8-5, 8-6, 8-7

ENTRY POINT: \$SGGVA

FUNCTION: Takes the value built in the output buffer (OUTBUF) and places it in the variable symbol table opposite the corresponding symbol name.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the output buffer (OUTBUF), located at the label OUTBUA of RDPARM

OUTPUT: The variable symbol table is updated. (The value is placed in the VST located at label VSTENT within RDPARM opposite its corresponding variable symbol name.)

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Number—Check/Build Routine (\$SGNCB)**

CHART: NH

FIGURE: 8-5, 8-6

ENTRY POINT: \$SGNCB

FUNCTION: Builds a number, with corresponding length and attribute, in the output buffer (OUTBUF).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the first byte of the input number, located at the label OUTGSY of RDPARM
- The address of the output buffer (OUTBUF), located at the label OUTBUA of RDPARM

OUTPUT: The output buffer (OUTBUF) contains a number (0-60 bytes in length) and its corresponding attribute.

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

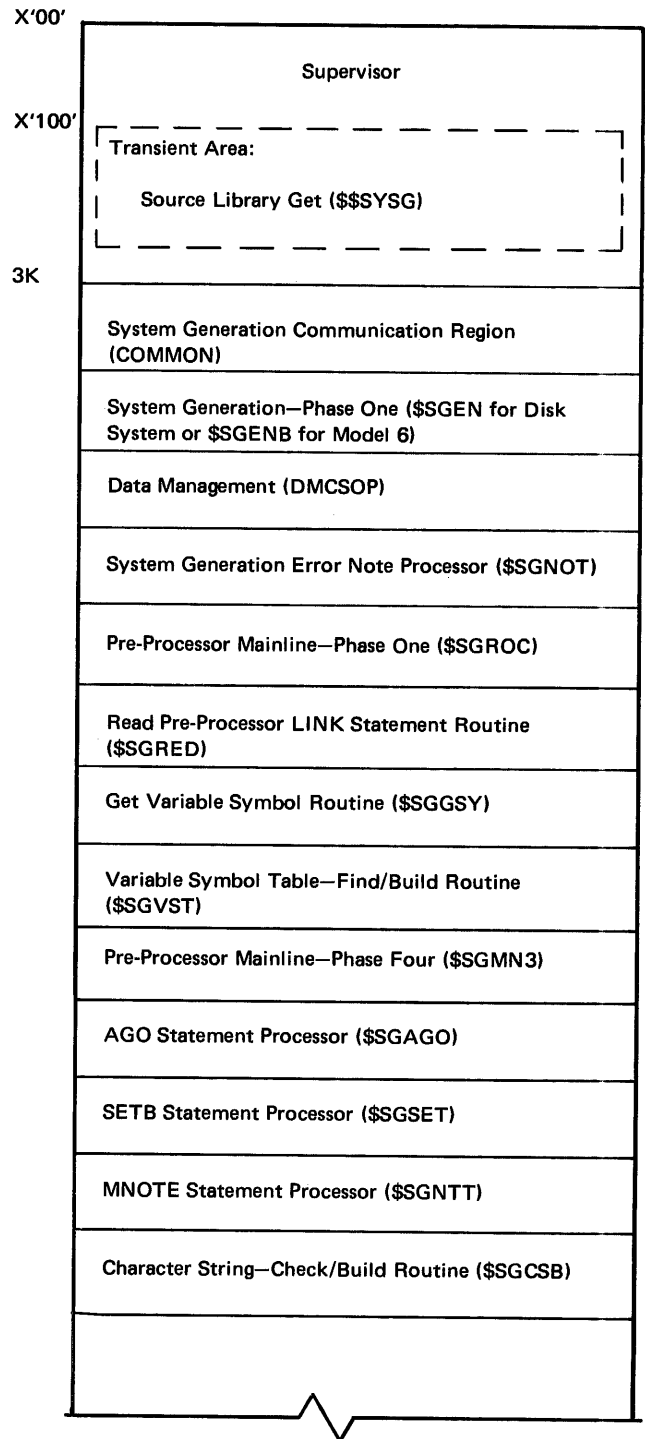


Figure 8-7. Main Storage Map for Phase Four Routines

► **Character String—Check/Build Routine (\$SGCSB)**

CHART: NI

FIGURE: 8-5, 8-6

ENTRY POINT: \$SGCSB

FUNCTION: Builds a character string, with its corresponding attribute and length, in the output buffer (OUTBUF).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the starting quote of the character string, located at the label OUTGSY of RDPARM
- The address of the output buffer (OUTBUF), located at the label OUTBUA of RDPARM

OUTPUT: The output buffer (OUTBUF) contains the character string with its corresponding attribute and length.

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Read Pre-Processor Statement Routine (\$SGRED)**

CHART: NJ

FIGURE: 8-5, 8-6, 8-7

ENTRY POINT: \$SGRED

FUNCTION: Reads the source pre-processor statements from the appropriate source library entry.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the Source Library—Get transient (\$SSYSG), located at the label PA of RDPARM

OUTPUT: The pre-processor statements are read into the output buffer (BUFFR1) located at label BUFF within RDPARM from the source library

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Link Statement Processor (\$SGSTM)**

CHART: NK

FIGURE: 8-5

ENTRY POINT: \$SGSTM

FUNCTION: Processes the LINK statement and loads the prototype statement into the save buffer (SVPROT). The prototype statement resides in SVPROT until all of the GBLB and LCLB statements have been processed.

INPUT: Register 1 contains the address of the pre-processor communication region (RDPARM).

OUTPUT: The prototype statement is loaded into the save buffer (SVPROT) located at label PROTSV within RDPARM.

EXIT:

- Normal: Calling routine (\$SGMN1)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Prototype Statement Processor (\$SGROT)**

CHART: NL

FIGURE: 8-5

ENTRY POINT: \$SGROT

FUNCTION: Directs the building of the prototype statement entries in the variable symbol table.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The prototype statement is located in the save buffer (SVPROT) located at label PROTSV within RDPARM

OUTPUT: The prototype statement entries are built in the variable symbol table.

EXIT:

- Normal: Pre-Processor Mainline (\$SGROC)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Keyword Operand Processor (\$SGOPR)**

CHART: NM

FIGURE: 8-5

ENTRY POINT: \$SGOPR

FUNCTION:

- Takes the keyword operands and values from the PROTOTYPE Statement, located in BUFFR1 and the system configuration statements, passed from \$SGEN (Disk System) or \$SGENB (Model 6) in the buffer (COMMON).
- Builds the variable symbol table entries.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the prototype statement, located at the label PROTSV in RDPARM
- The address of the buffer (COMMON) containing the system configuration statements, located at the label COMMON within RDPARM

OUTPUT: The variable symbol table contains the entries and values obtained from the prototype and system configuration statements.

EXIT:

- Normal: Prototype Statement Processor (\$SGROT)
- Error:
 1. Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC) if an error is found in a prototype statement
 2. Error section (CALLNP) of the Pre-Processor Mainline (\$SGROC) if an error is found in a system configuration statement

► **AIF Statement Processor (\$SGAIF)**

CHART: NN

FIGURE: 8-8

ENTRY POINT: \$SGAIF

FUNCTION:

- Directs the translating of the variable symbols on the AIF statement into their respective values.
- Tests to see if the respective values can be resolved according to the condition stated on the AIF statement.
- Directs the internal conditional processing of the pre-processor (AIF) statements.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the AIF statement, located at the label BUFF in RDPARM

OUTPUT: The output buffer (BUFFR1) contains the next pre-processor statement to be executed.

EXIT:

- Normal: Pre-Processor Mainline \$SGROC
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Model Statement Build Routine (\$SGMST)**

CHART: NO

FIGURE: 8-8

ENTRY POINT: \$SGMST

FUNCTION:

- Reads and analyzes the model statement located in BUFFR1.
- The variable symbols are replaced by their corresponding values located in the variable symbol table.
- The analyzed model statement is rebuilt, with its replaced values, in the save buffer (SVPROT).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The model statement that is to be analyzed, located in BUFFR1

OUTPUT: The rebuilt model statement, located in the save buffer SVPROT

EXIT:

- Normal: Pre-Processor Mainline (\$SGROC)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Subfield Analyzer (\$SGSUB)**

CHART: NP

FIGURE: 8-8

ENTRY POINT: \$SGSUB

FUNCTION:

- Analyzes the following data fields, one field at a time. The following data fields are located on the model and AIF statements.

Field one—Positions 1-7

Field two—Positions 8-13

Field three—Positions 14-96

- Translates the variables found in the above fields into their corresponding variable symbol table values.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the first byte of the field to be analyzed, located at the label INGSY in RDPARM

OUTPUT: The output buffer (SVPROT) contains the analyzed/translated field.

EXIT:

- Normal: Calling routine
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Model Statement Output Processor (\$SGOUT)**

CHART: NQ

FIGURE: 8-8

ENTRY POINT: \$SGOUT

FUNCTION: Writes the analyzed model statement into the output file (MACOUT) designated by the caller (\$SGEN for Disk System or \$SGENB for Model 6).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of Data Management, located at the label ADDMGN within the system generation communication region (COMMON)
- The address of the DTF output file, located at the label ADDDTF within the system generation communication region (COMMON)

OUTPUT: The analyzed model statement is located in the requester's designated output file (MACOUT).

EXIT:

- Normal: Calling routine (\$SGMST)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Table Statement Processor (\$SGTBL)**

CHART: NR

FIGURE: 8-5

ENTRY POINT: \$SGTBL

FUNCTION: Takes the variable symbol, located in positions 14-20 on the TABLE statement, and locates or builds an entry for it in the variable symbol table.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the TABLE statement, located at the label BUFF within RDPARM

OUTPUT: The address of the variable symbol (found/built), located at the label VSTENT within RDPARM

EXIT:

- Normal: Calling routine (\$SGMN2)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Table Definition Processor (\$SGDEF)**

CHART: NS

FIGURE: 8-8

ENTRY POINT: \$SGDEF

FUNCTION:

- Using the value from the variable symbol table located at label VSTENT, \$SGDEF reads the TABLE definition statements into an input buffer (BUFFR1). These statements are read until a match is located (columns 1-12).
- The argument (columns 14-96), on the matched TABLE definition statement (located in BUFFR1), replaces the matched value in the variable symbol table.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the variable symbol in the variable symbol table, located at the label VSTENT within RDPARM

OUTPUT: Updated variable symbol table

EXIT:

- Normal: Calling routine (\$SGMN2)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

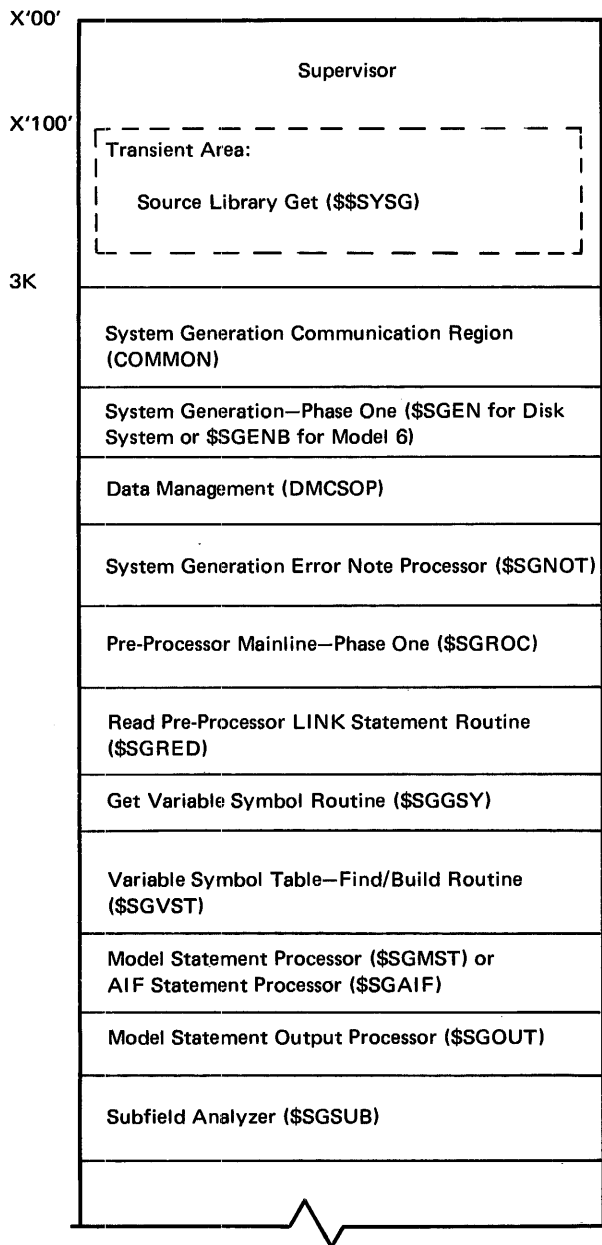


Figure 8-8. Main Storage Map for Various Routines

► **AGO Statement Processor (\$\$GAGO)**

CHART: NT

FIGURE: 8-7

ENTRY POINT: \$\$GAGO

FUNCTION:

- Determines if the sequence symbol, located in positions 1-6, is valid.
- Performs a forward search through the source library until a matching sequence symbol is located.
- Passes the corresponding pre-processor statement from the source library to the Pre-Processor Mainline (\$\$GROC).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the sequence symbol, located at the label INGSY within RDPARM

OUTPUT: The address of the pre-processor statement, with the sequence symbol, located at the label BUFF within RDPARM

EXIT:

- Normal: Pre-Processor Mainline (\$\$GROC)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$\$GROC)

► **MNOTE Statement Processor (\$\$GNNT)**

CHART: NU

FIGURE: 8-7

ENTRY POINT: \$\$GNNT

FUNCTION: Passes the pre-processor MNOTE statement (located in BUFFER1) to the System Generation Error Note Processor (\$\$GNNT).

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the Error Note Processor, located at the label ADNOTE located within the system generation communication region (COMMON)

OUTPUT: The MNOTE statement, passed to the Error Note Processor (\$\$GNNT)

EXIT:

- Normal: Calling routine (\$\$GMN3)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$\$GROC)

► **SETB Statement Processor (\$SGSET)**

CHART: NV

FIGURE: 8-7

ENTRY POINT: \$SGSET

FUNCTION:

- Locates the variable symbol, as specified on the SETB pre-processor statement, in the variable symbol table.
- Loads the SETB value into this variable symbol table entry.

INPUT:

- The address of the pre-processor communication region (RDPARM), in Register 1
- The address of the SETB statement, located at the label BUFF within RDPARM

OUTPUT: The variable symbol table entry is set to either 0 (off) or 1 (on).

EXIT:

- Normal: Calling routine (\$SGMN3)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **Global and Local Statement Processor (\$SGGBL)**

CHART: NW

FIGURE: 8-5

ENTRY POINT: \$SGGBL

FUNCTION: Creates LCLB or GBLB entries in the variable symbol table.

INPUT:

- The address of the pre-processor communication region (RDPARM)
- The address of the pre-processor statement (LCLB or GBLB), located at the label BUFF within RDPARM.

OUTPUT: The GBLB or LCLB entries, built in the variable symbol table

EXIT:

- Normal: Calling routine (\$SGMN1)
- Error: Error section (SETNOT) of the Pre-Processor Mainline (\$SGROC)

► **IPL Bootstrap Modify (\$SGPRI)—Model 6 Only**

CHART: NX

FIGURE: None

ENTRY POINT: SGPR1

FUNCTION: Modifies the IPL Bootstrap according to the setting of the UPSI switch:

- B'11000000'—set BASIC as the primary IPL system.
- B'10000000'—set DSM as the primary IPL system.

INPUT: UPSI switch

OUTPUT: Modified IPL bootstrap

EXIT:

- Normal: End of Job transient (\$SPEJ)
- Error: Supported Halt/Syslog routine

► **System Verification Message Routine (\$SGIVP)**

CHART: None

FIGURE: None

ENTRY POINT: SGVIP

FUNCTION: Prints the following message:

END OF SYSTEM VERIFICATION

INPUT: None

OUTPUT: Printed message

EXIT:

- Normal: End of Job Transient (\$SPEJ)
- Error: None

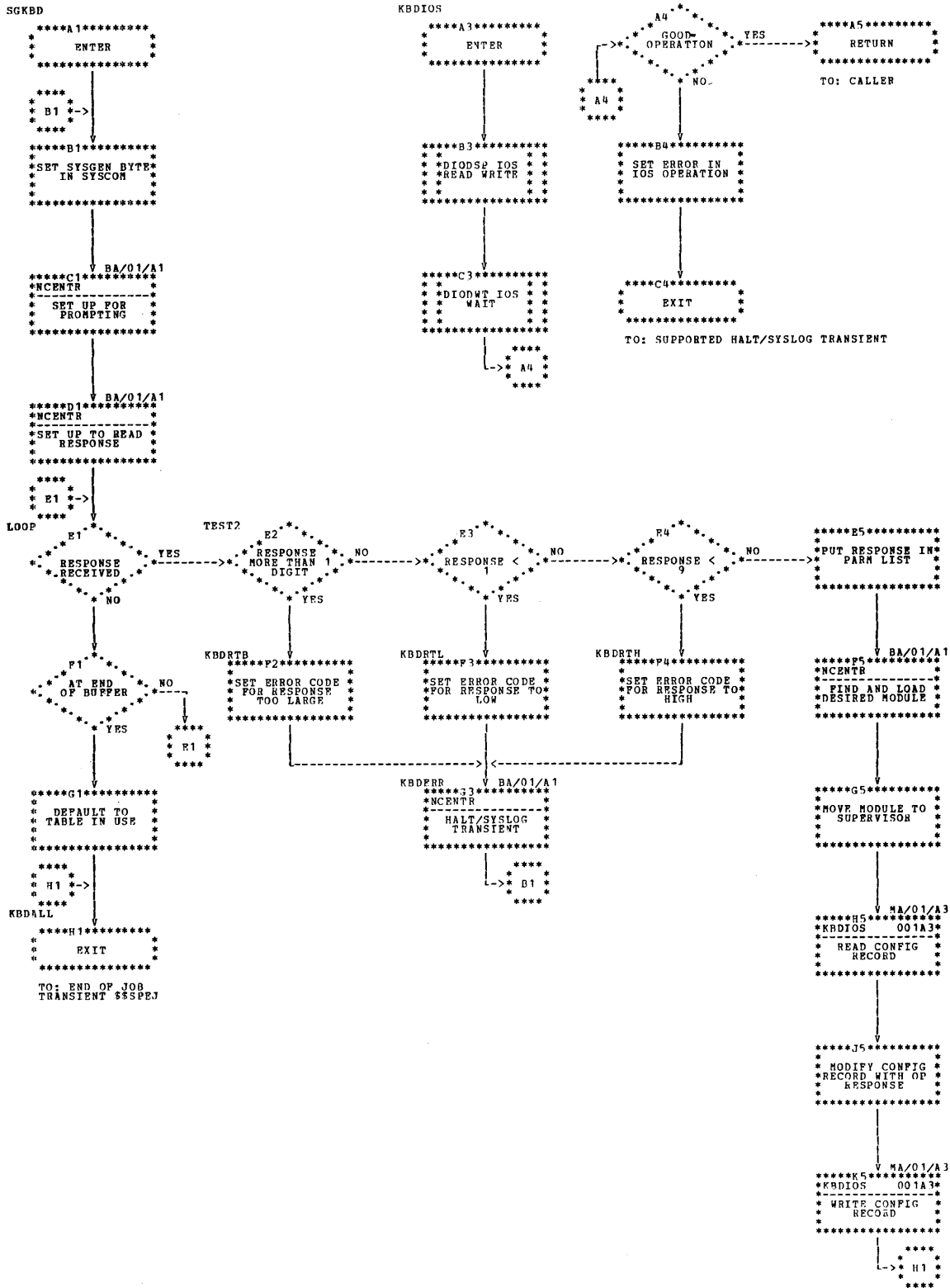


Chart MA. Keyboard Selection Routine (\$SGKBD)–Model 6 Only

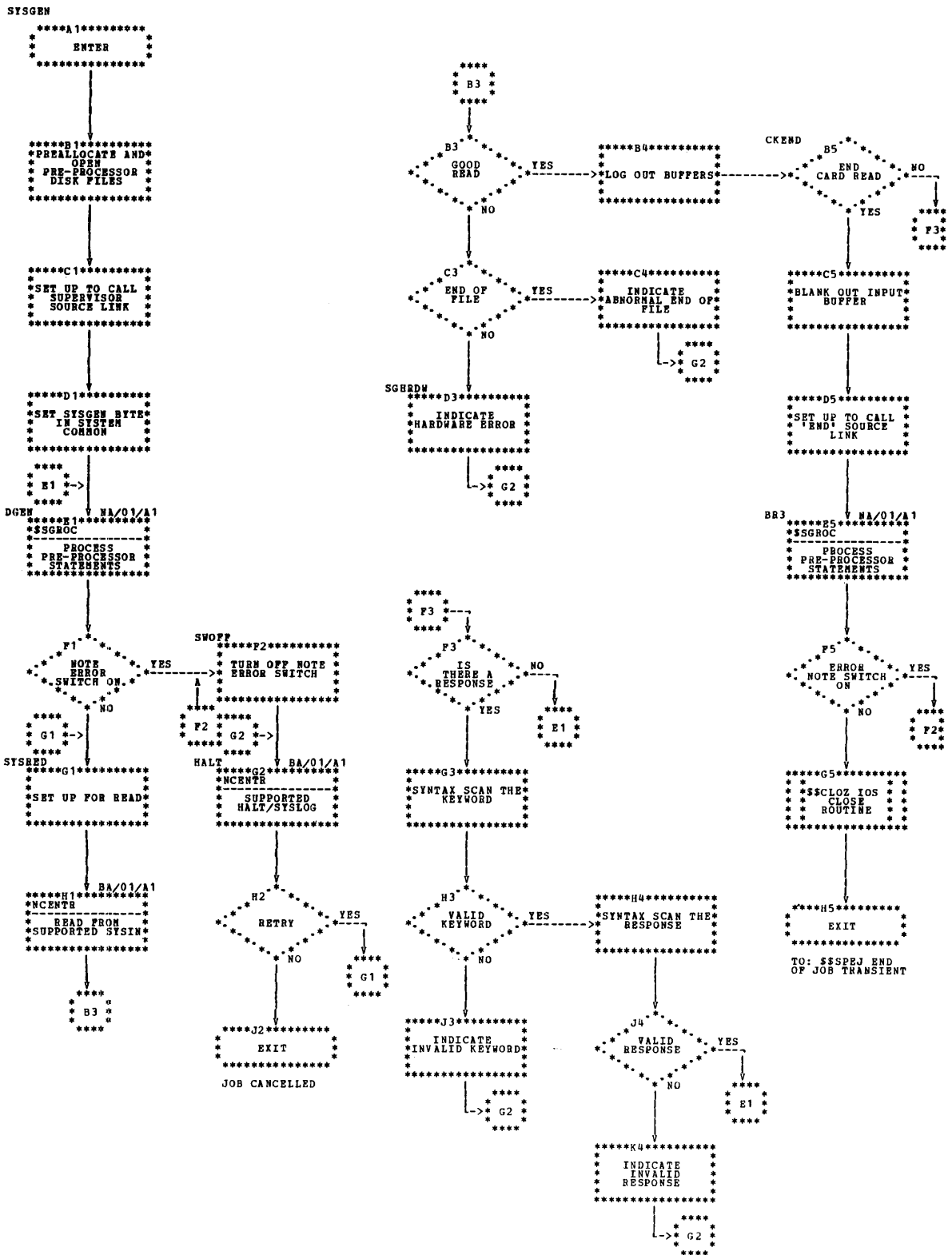


Chart MC. System Generation—Phase One (\$SGEN)—Disk System

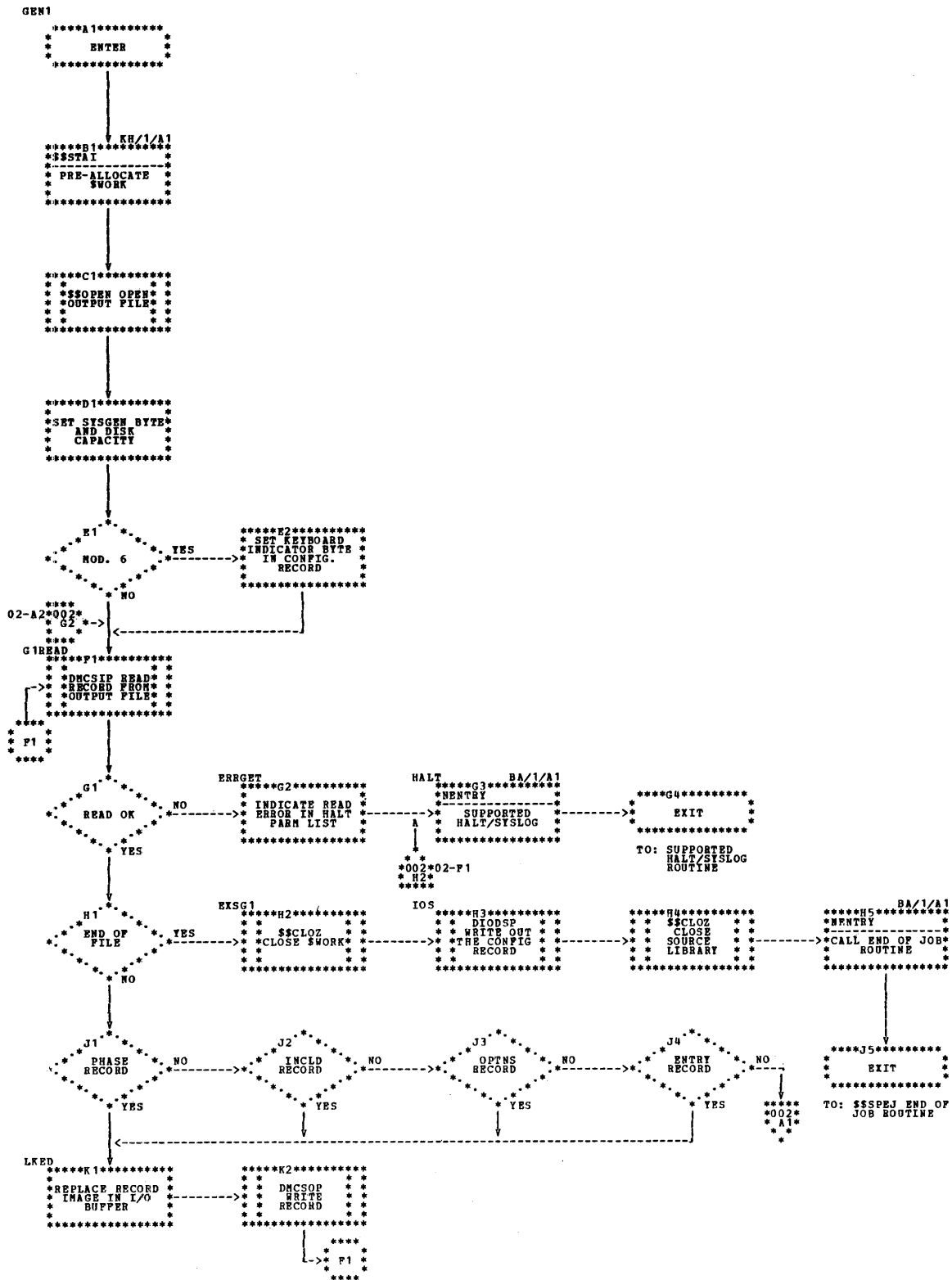


Chart ME (Part 1 of 2). System Generation—Phase Two (\$\$GEN1)

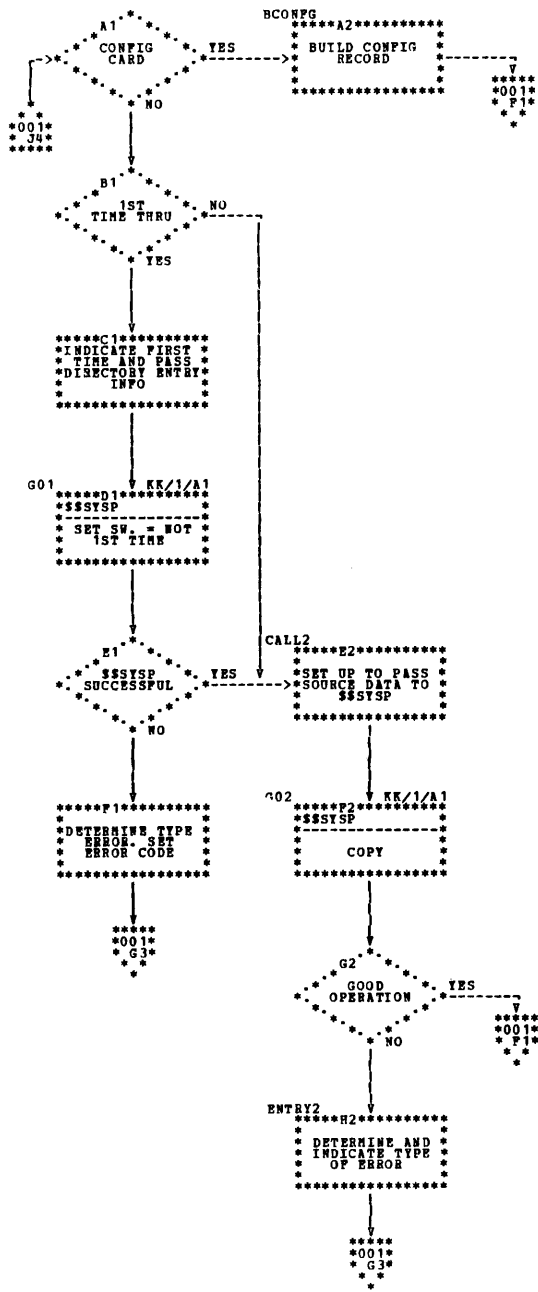


Chart ME (Part 2 of 2). System Generation—Phase Two (\$\$GEN1)

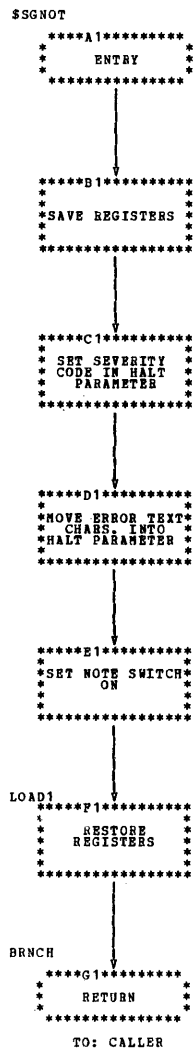


Chart MF. System Generation Error Note Processor (\$SGNOT)

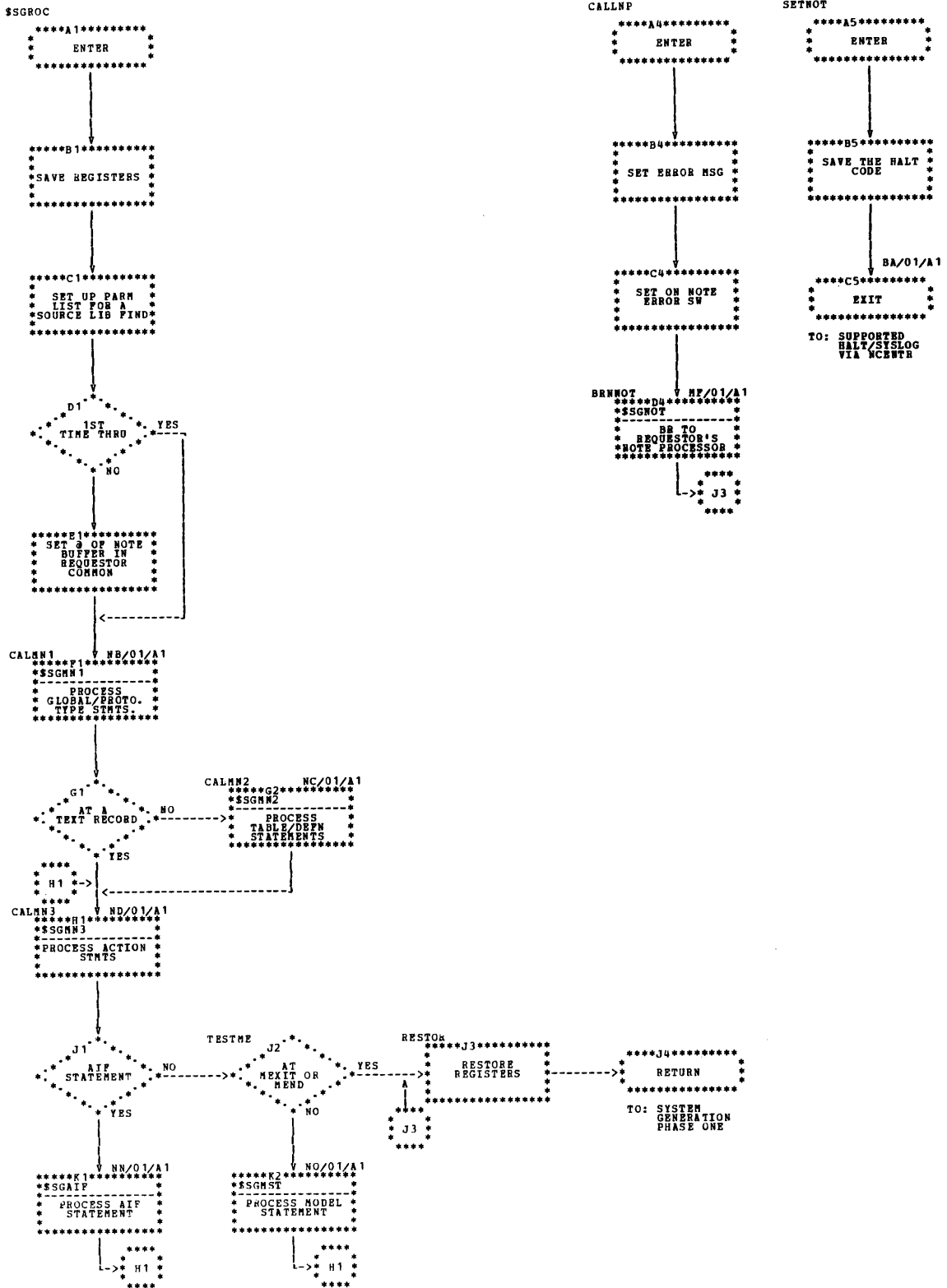


Chart NA. System Generation Pre-Processor Mainline-Phase One (\$SGROC)

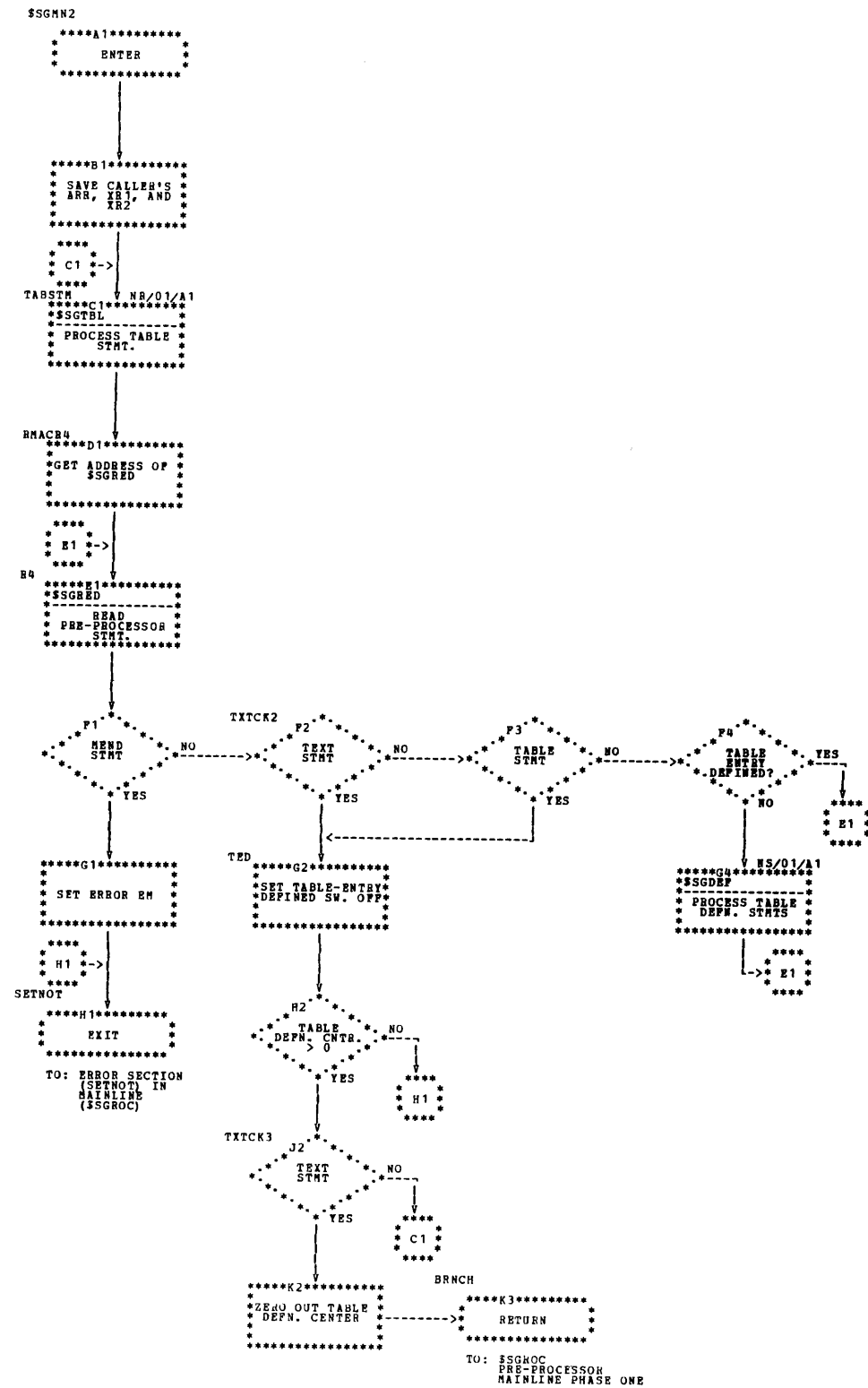


Chart NC. System Generation Pre-Processor Mainline-Phase Three (\$SGMN2)

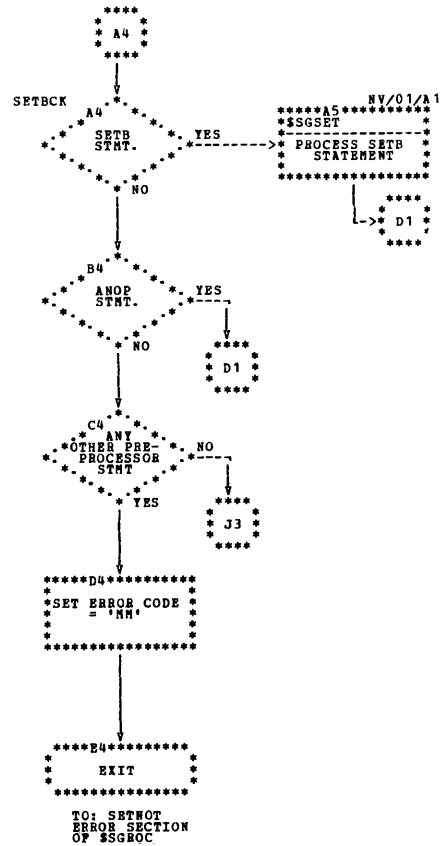
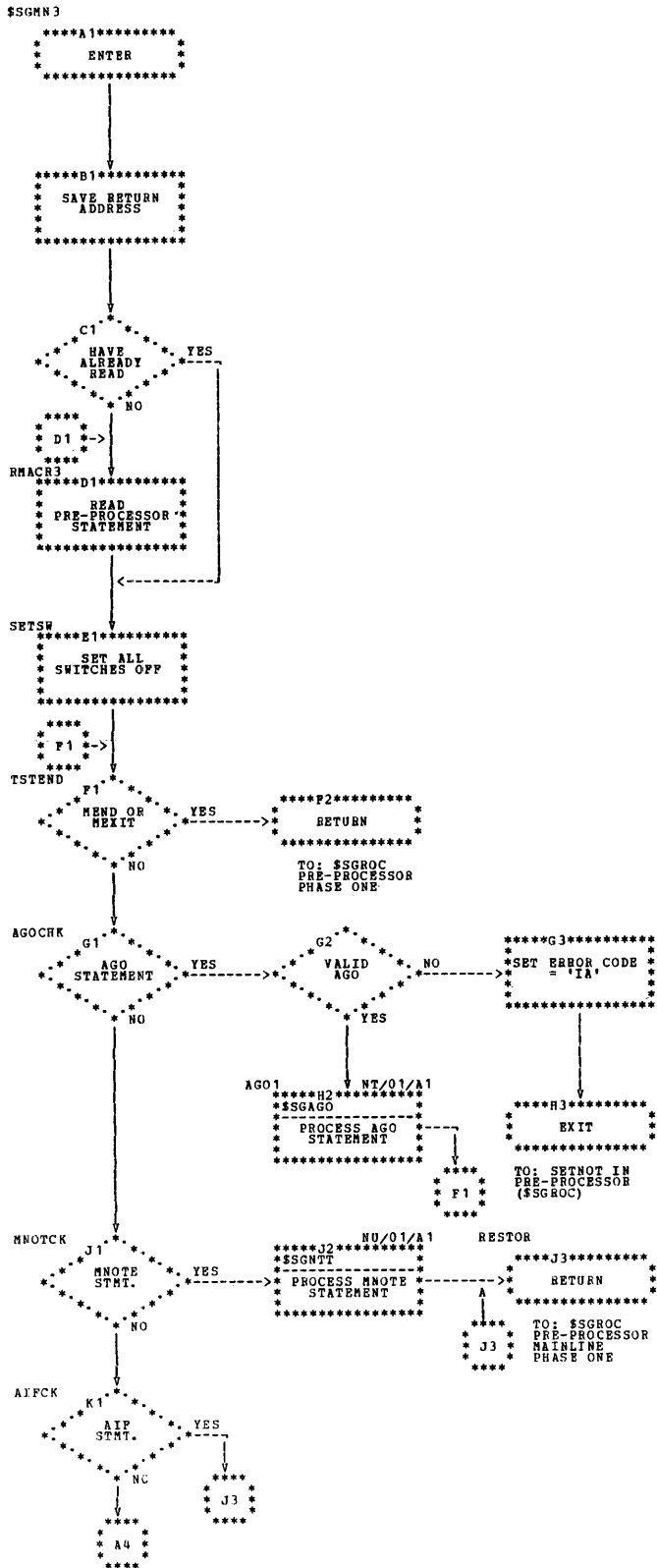
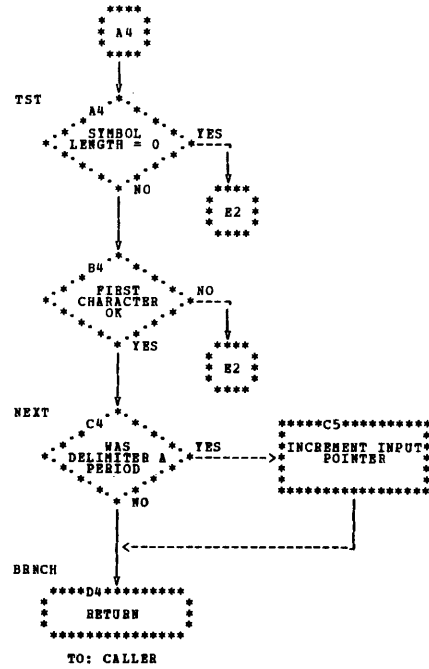
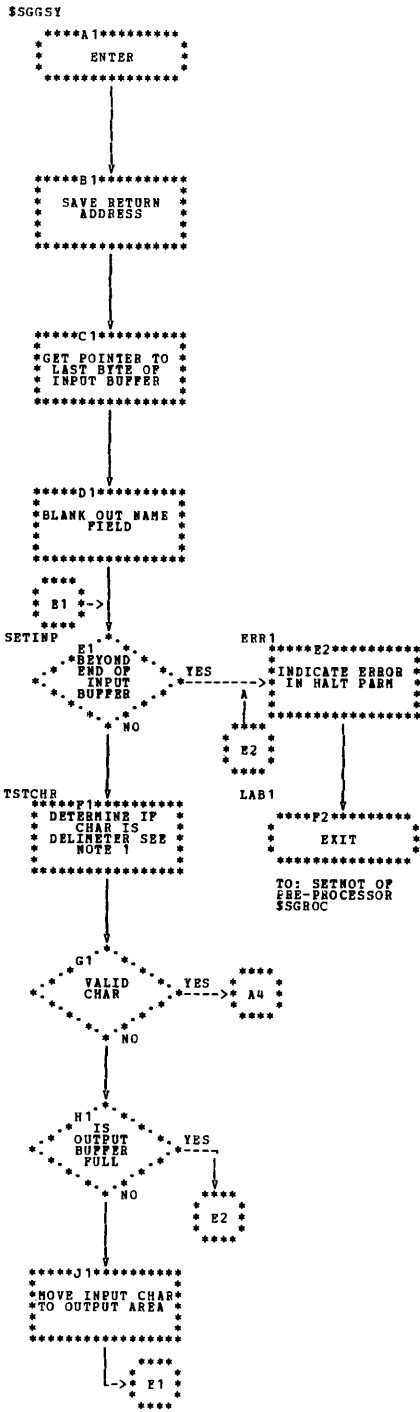


Chart ND. System Generation Pre-Processor-Phase Four (\$SGMN3)



NOTE 1: VALID DELIMITERS ARE:
 BLANK EQUAL SIGN
 PERIOD QUOTE
 COMMA SLASH
 DASH LEFT PARENTH
 PLUS RIGHT PARENTH
 AMPERSAND

Chart NE. Get Variable Symbol Routine (\$SGGSY)

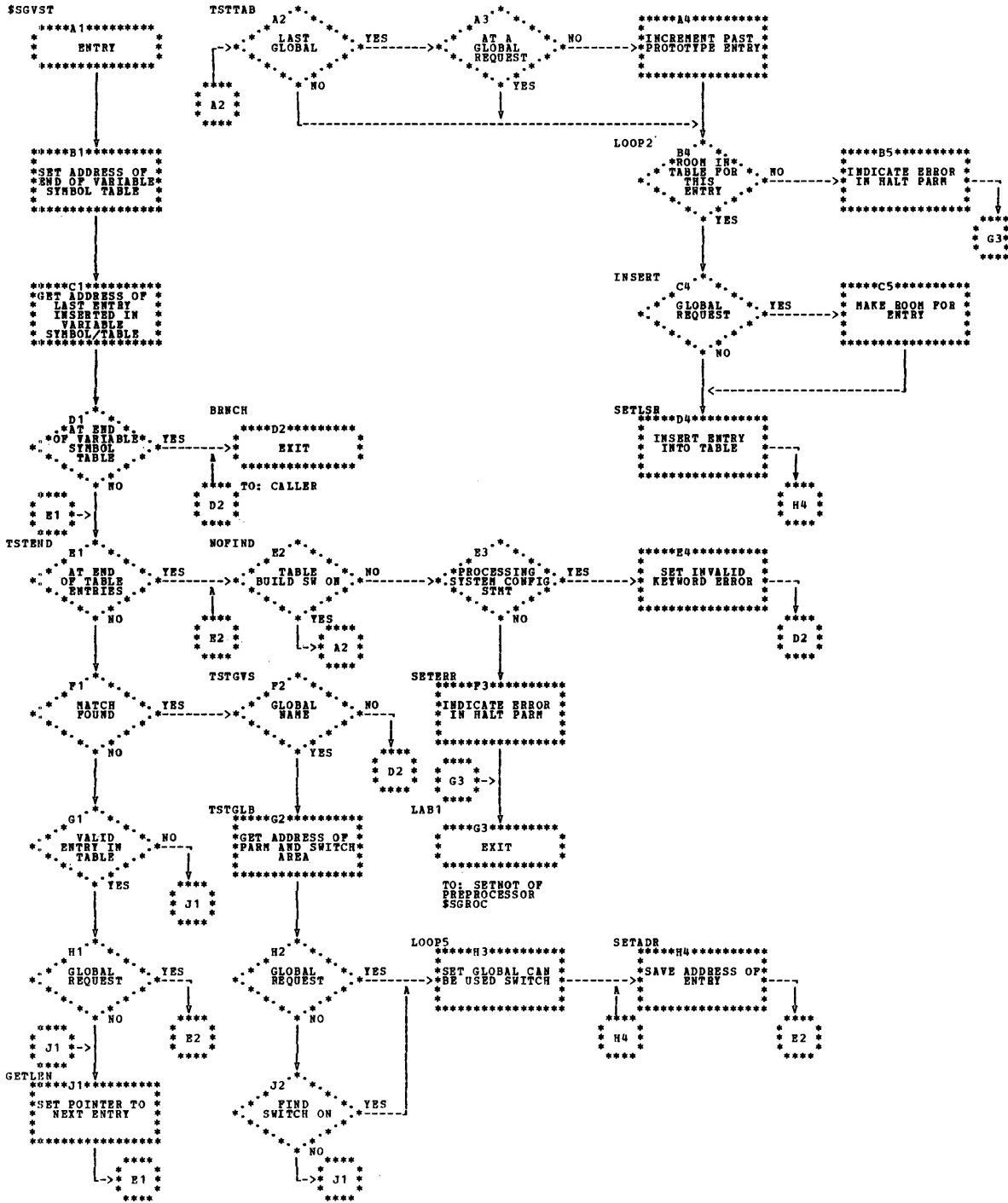


Chart NF. Variable Symbol Table-Find/Build Routine (\$SGVST)

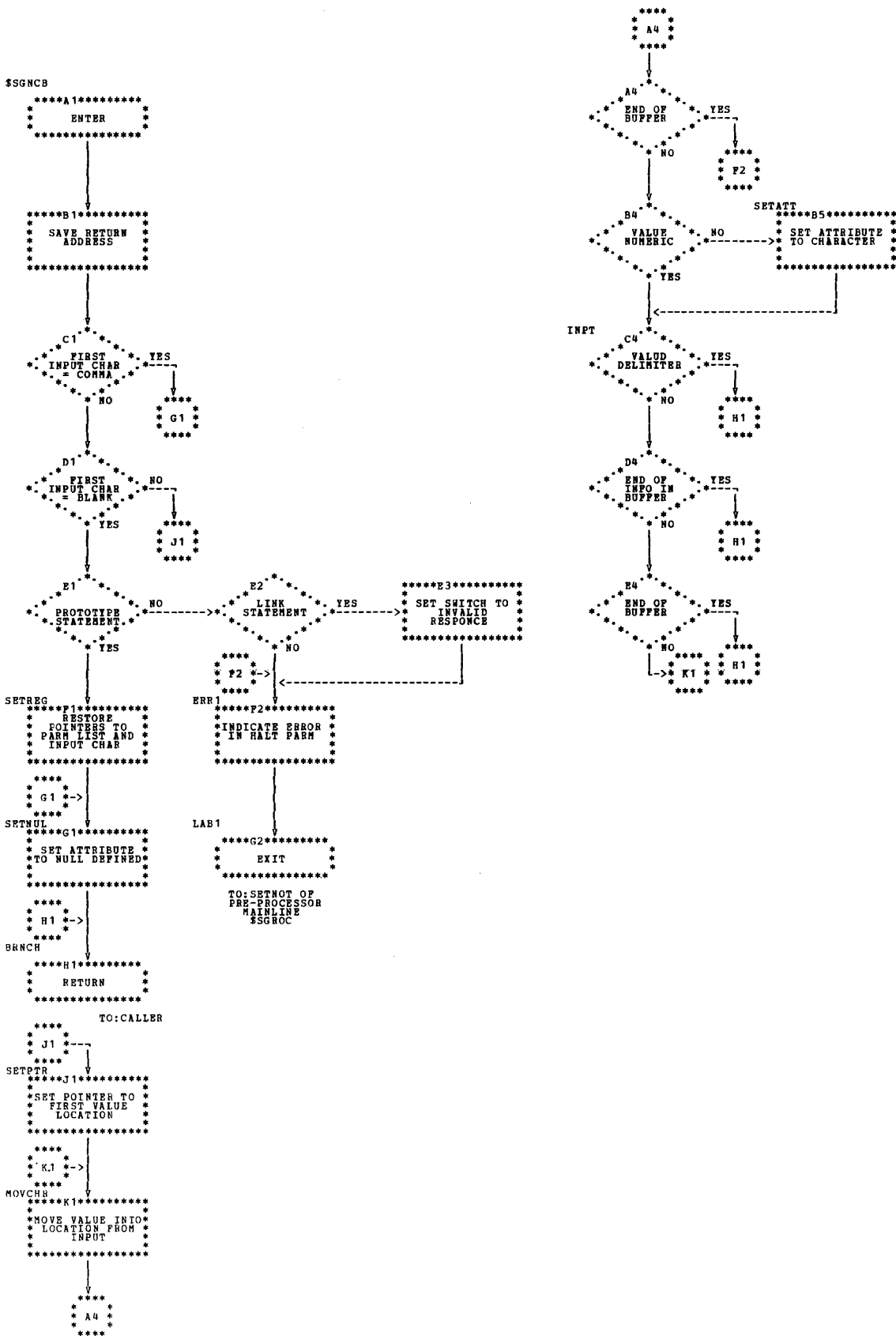


Chart NH. Number-Check/Build Routine (\$SGNCB)

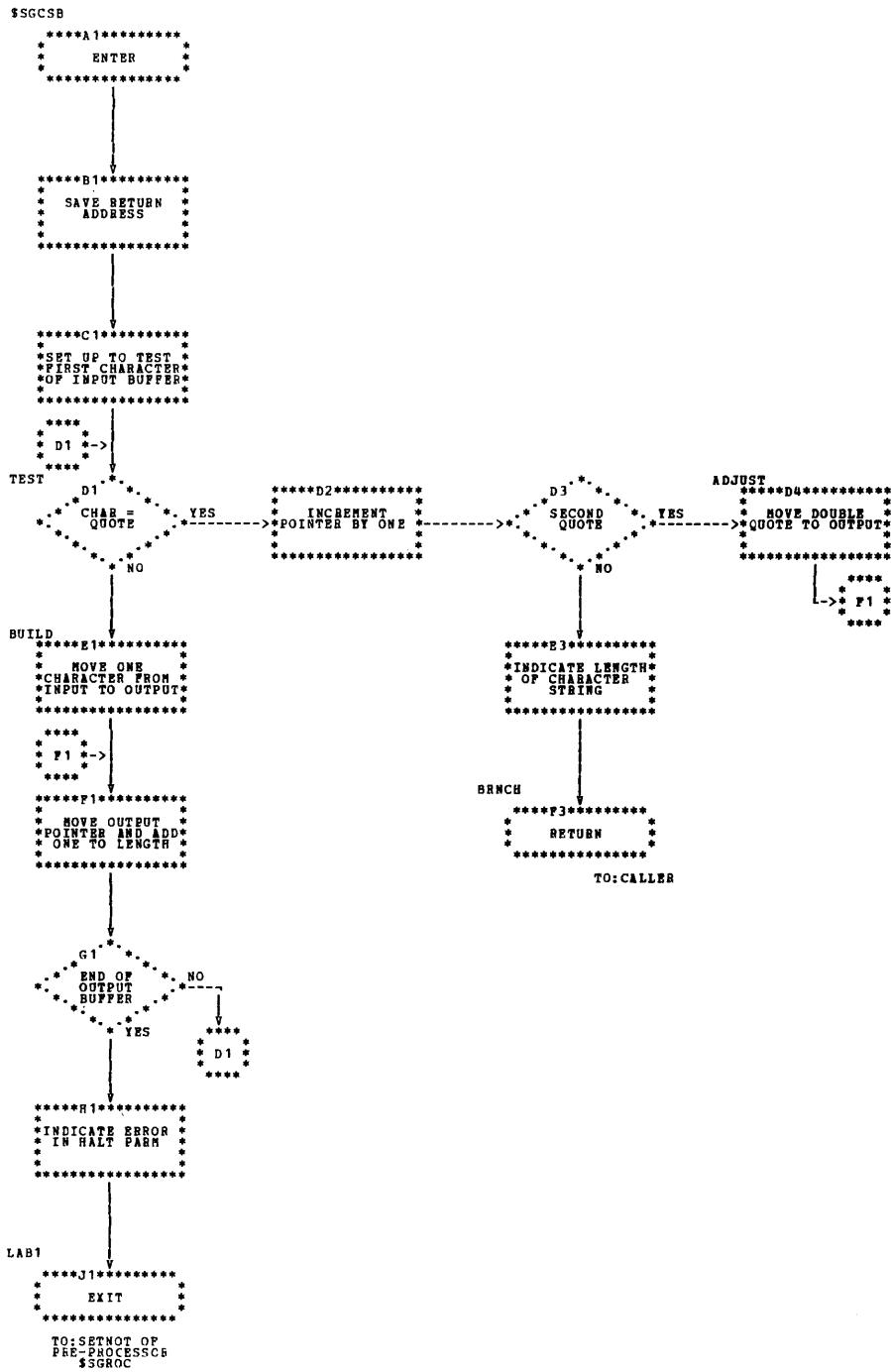


Chart NI. Character String-Check/Build Routine (\$SGCSB)

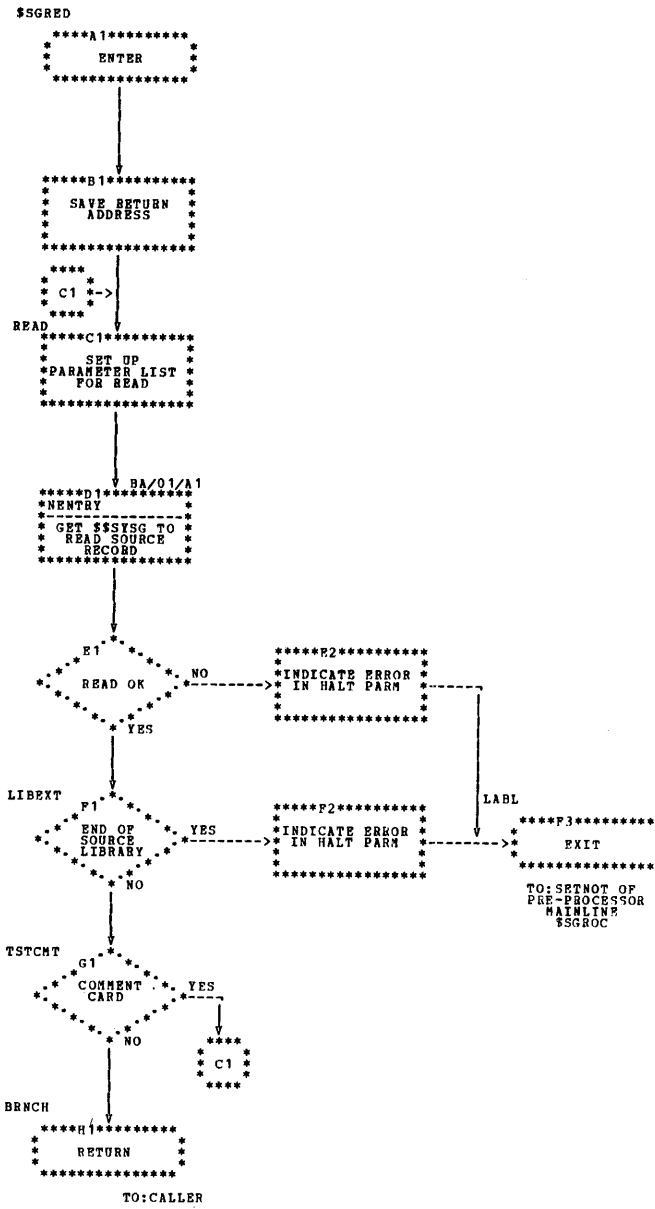


Chart NJ. Pre-Processor Statement-Read Routine (\$\$GRED)

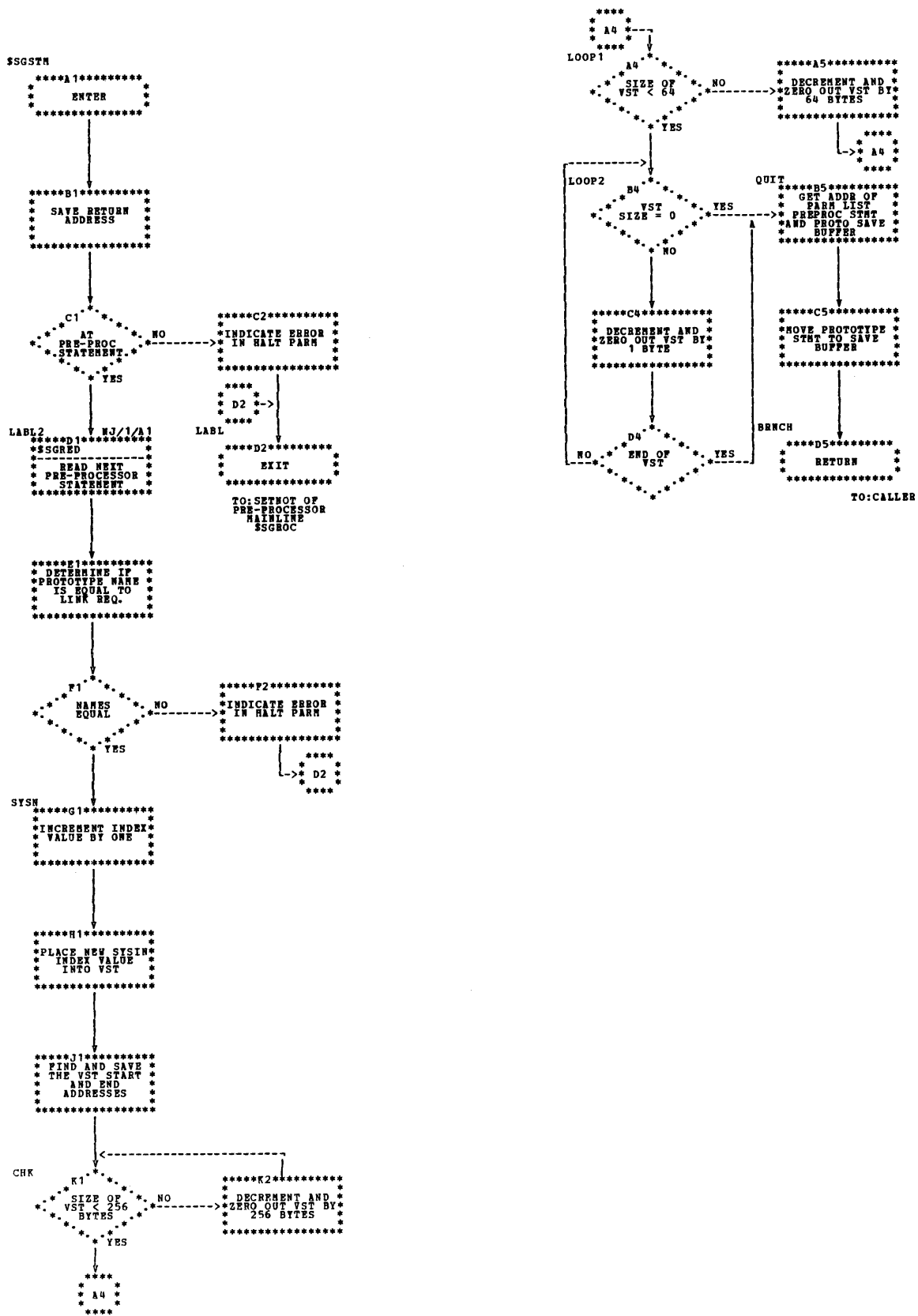


Chart NK. Link Statement Processor (\$SGSTM)

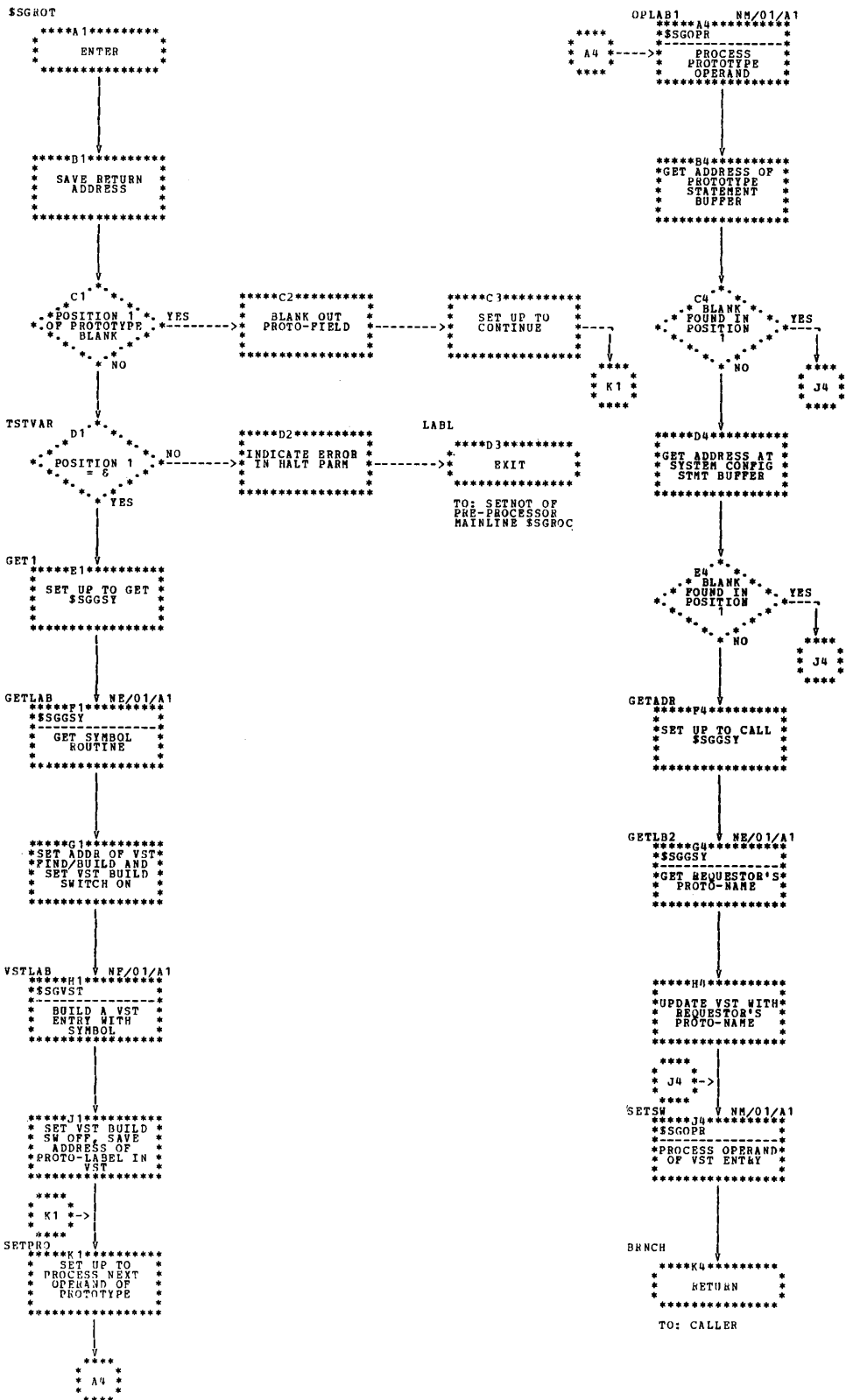


Chart NL. Prototype Statement Processor (\$SGROT)

SSGOPR

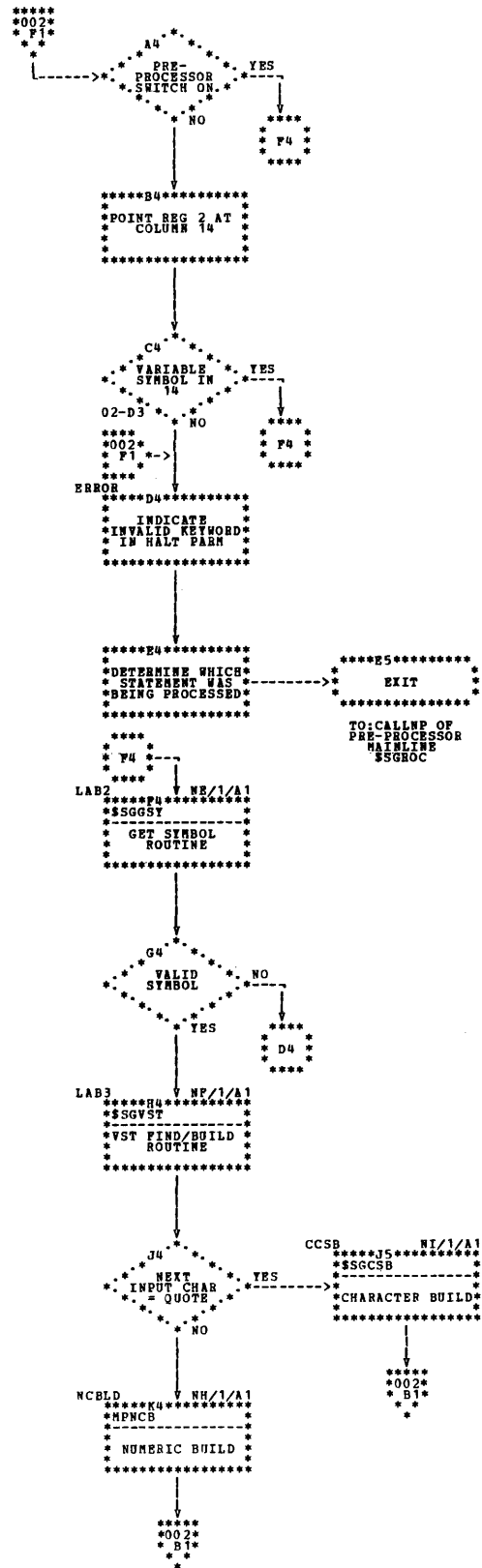
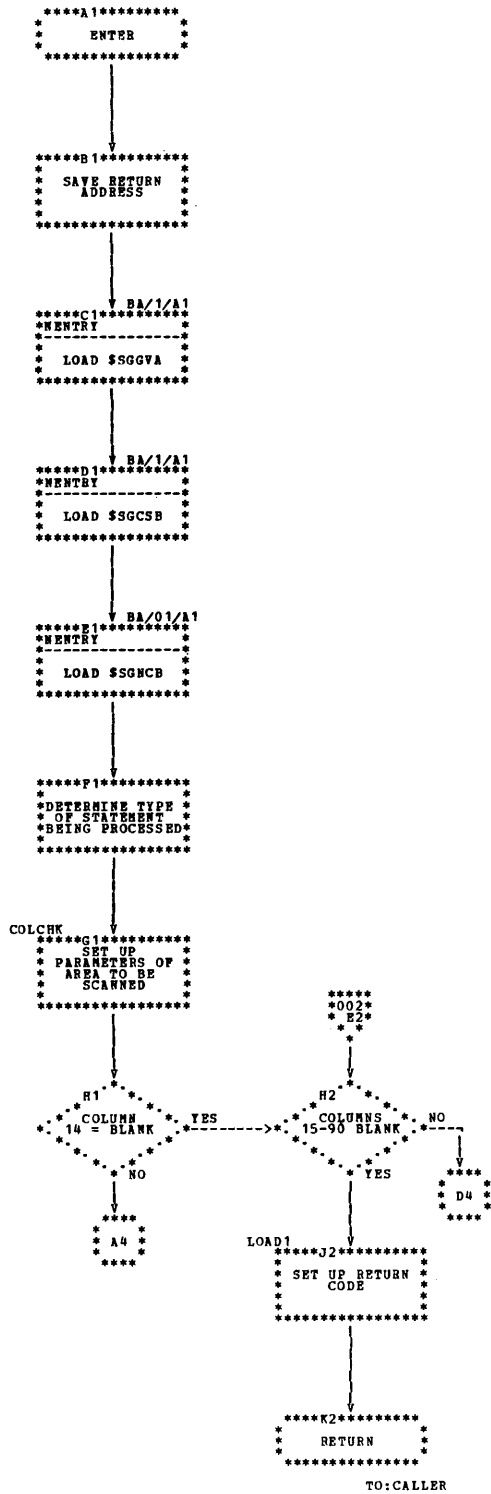


Chart NM (Part 1 of 2). Keyword Operand Processor (\$SGOPR)

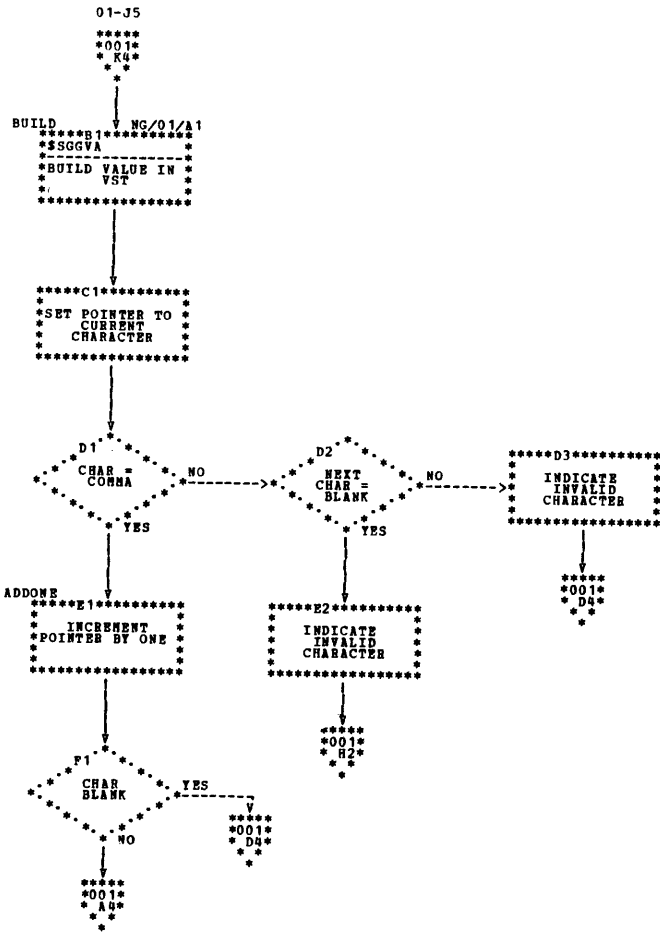


Chart NM (Part 2 of 2). Keyword Operand Processor (\$SGOPR)

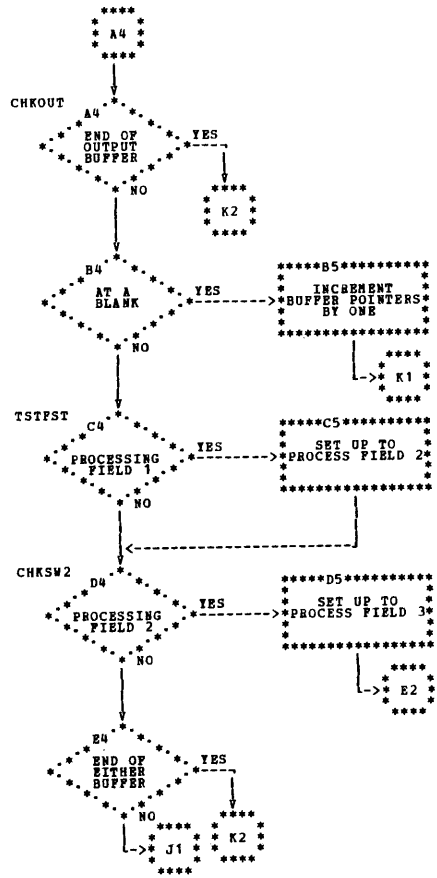
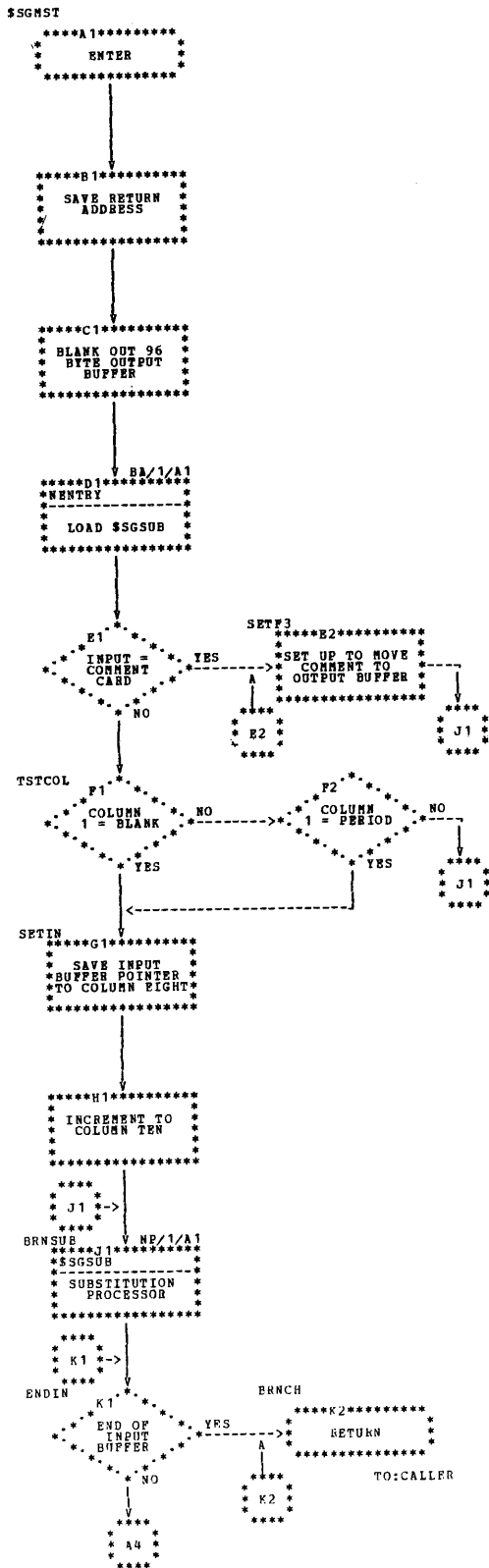


Chart NO. Model Statement Build Routine (\$SGMST)

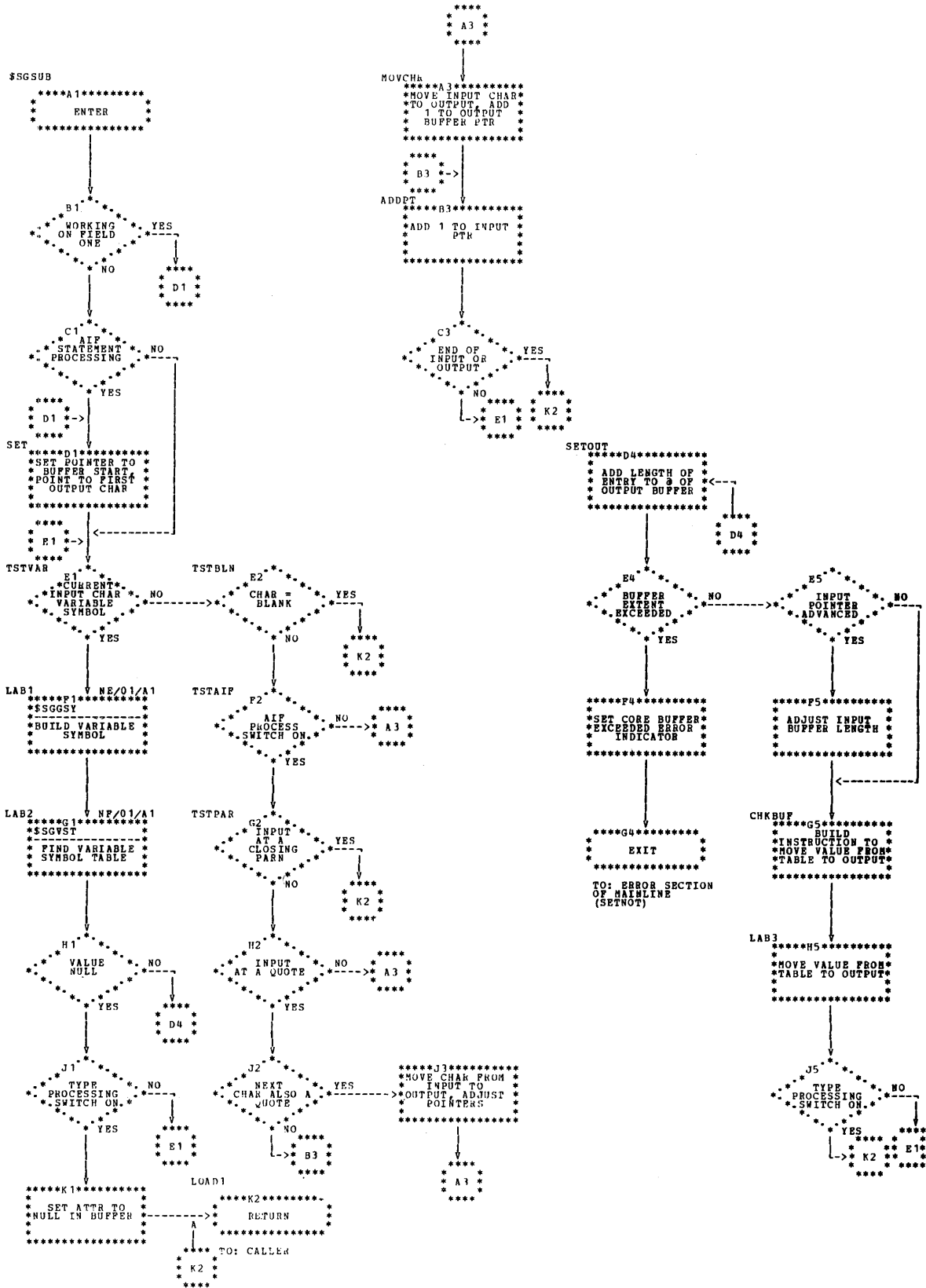


Chart NP. Subfield Analyzer (\$SGSUB)

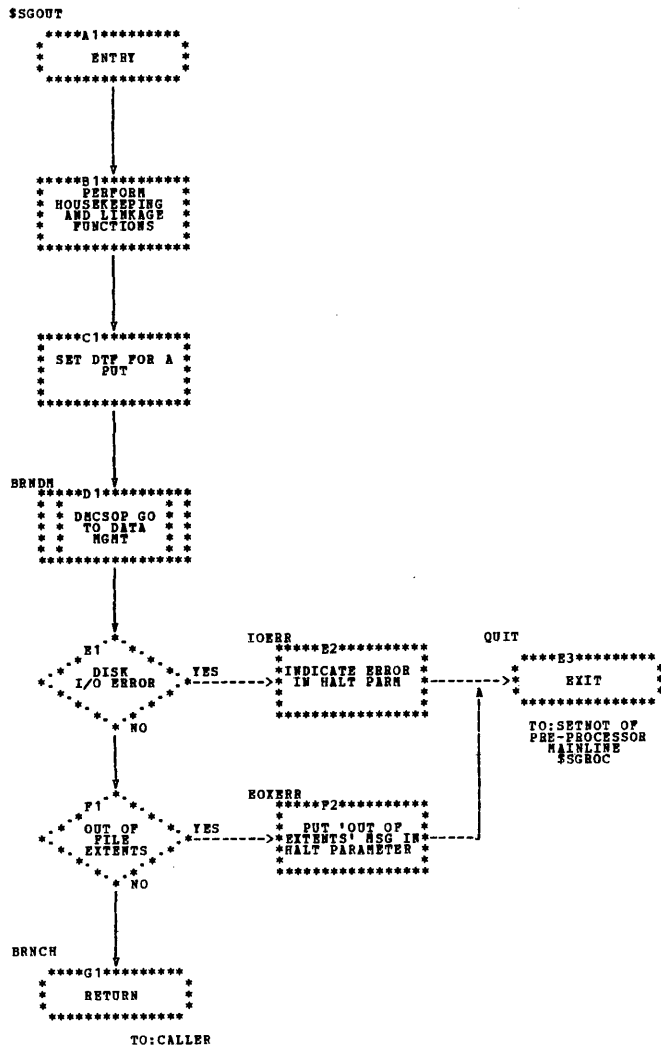


Chart NQ. Model Statement Output Routine (\$SGOUT)

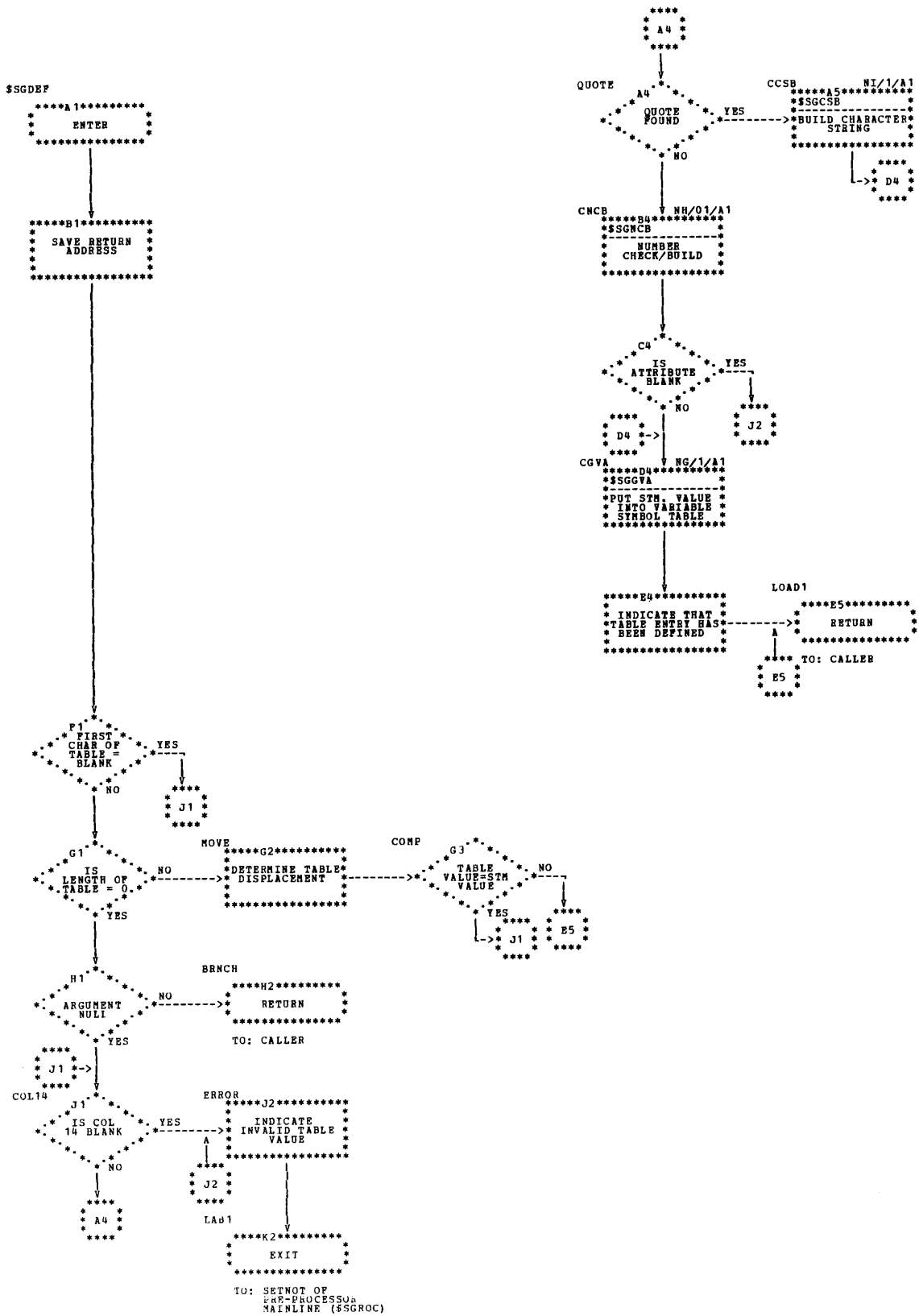


Chart NS. Table Definition Statement Processor (\$SGDEF)

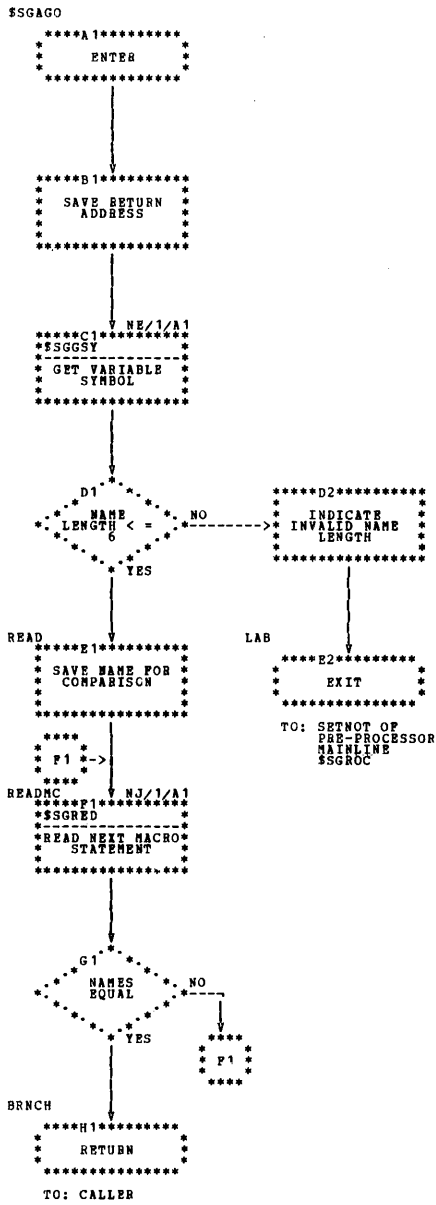


Chart NT. AGO Statement Processor (\$SGAGO)

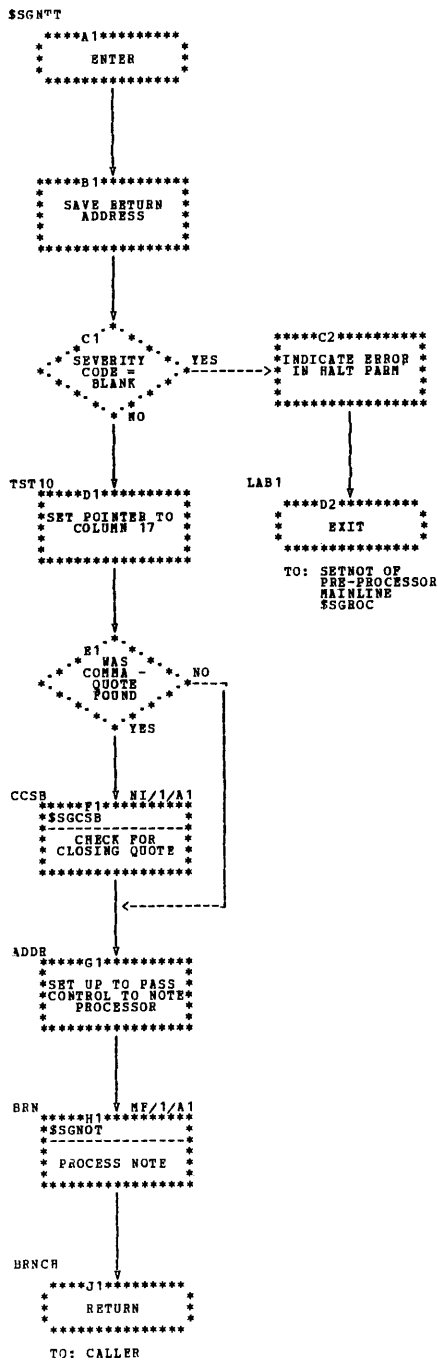


Chart NU. MNOTE Statement Processor (\$SGNTT)

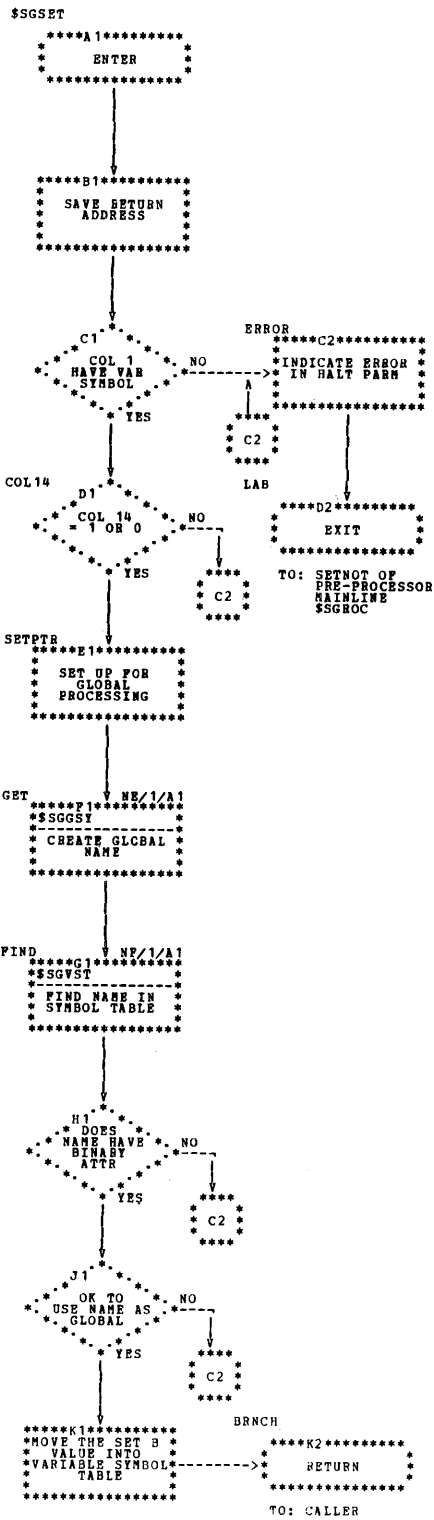


Chart NV. SETB Statement Processor (\$SGSET)

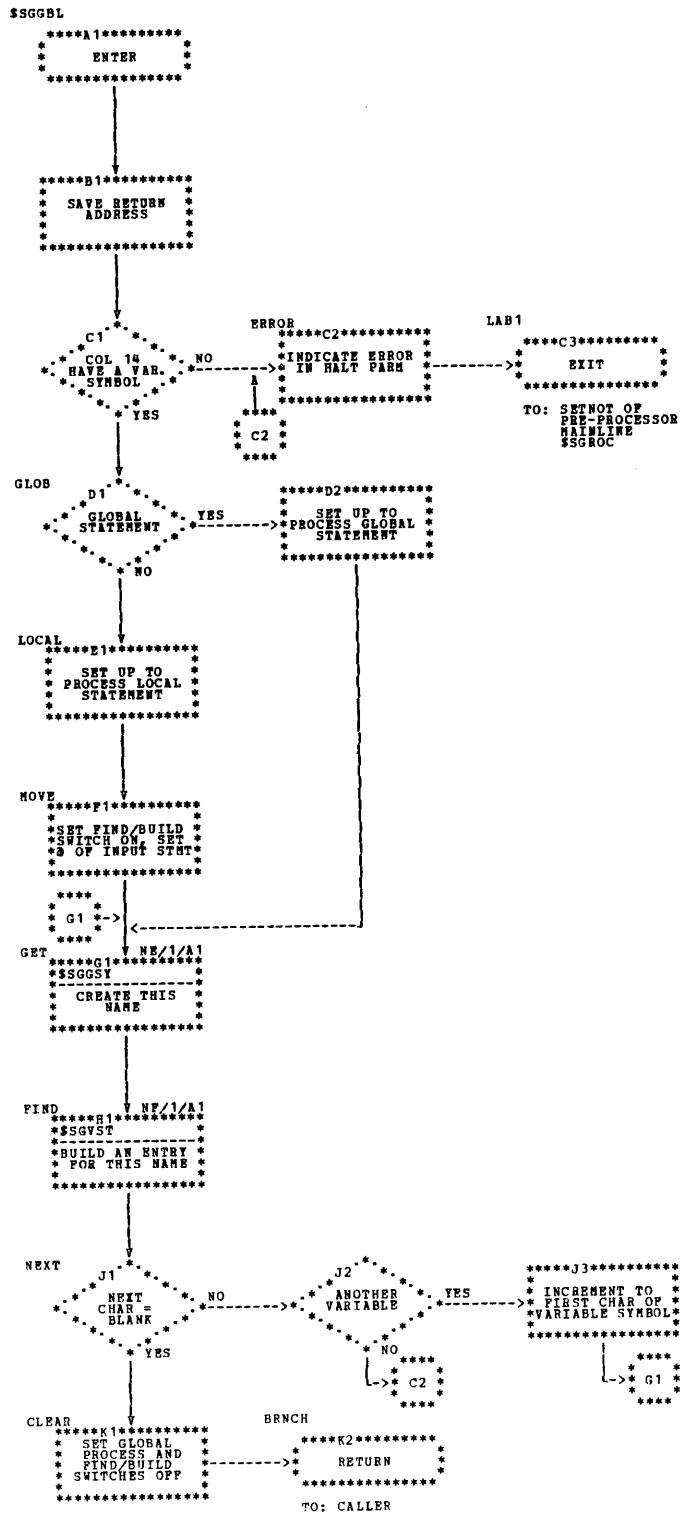


Chart NW. Global and Local Statement Processor (\$SGGBL)

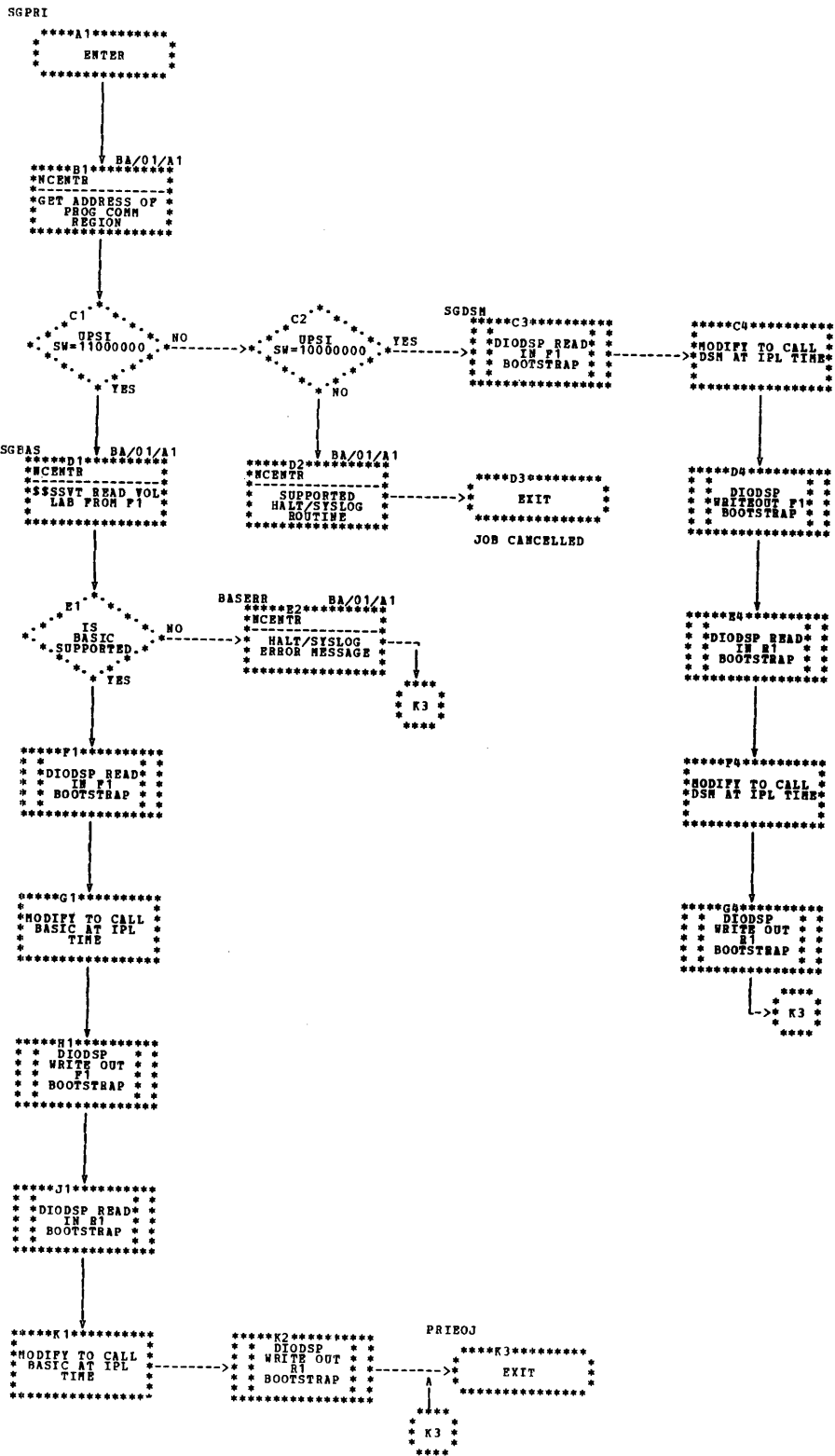


Chart NX. IPL Bootstrap Modify (\$SGPRI)—Model 6 Only

Figure 8-9 contains the Directory entries for System Generation.

Module/Phase Name	Chart ID	Descriptive Name	Entry Point	Function
\$SGKBD (Model 6 only)	MA	Keyboard Selection Routine	SBKBD	<ul style="list-style-type: none"> Allows user to select appropriate keyboard image table
\$SBBSV (Model 6 only)	MB	Co-resident System Analyzer Routine	SGBSV	<ul style="list-style-type: none"> Reserves space for BASIC system
\$GEN (Disk system)	MC	System Generation	SYSGEN	<ul style="list-style-type: none"> Obtains system configuration
\$GENB (Model 6 only)	MD			<ul style="list-style-type: none"> Passes responses to pre-processor
\$GEN1	ME	System Generation	GEN1	<ul style="list-style-type: none"> Processes pre-processor output file Creates new configuration record on F1 Builds procedures for SCP support generation Creates linkage editor input file (\$WORK)
\$SGNOT	MF	Note Processor	\$SGNOT	<ul style="list-style-type: none"> Loads halt transient parameter list Sets note error switch on
\$SGROC	NA	Mainline Phase One	\$SGROC	<ul style="list-style-type: none"> Acts as pre-processor mainline driver Calls \$SGMN1, \$SGMN2, and \$SGMN3, \$SGMST, \$SGAIF
\$SGMN1	NB	Mainline Phase Two	\$SGMN1	<ul style="list-style-type: none"> Directs processing of global, prototype, and pre-processor input statement keyword operands
\$SGMN2	NC	Mainline Phase Three	\$SGMN2	<ul style="list-style-type: none"> Directs the processing of TABLE and TABLE definition statements
\$SGMN3	ND	Mainline Phase Four	\$SGMN3	<ul style="list-style-type: none"> Controls processing of action (conditional processing) statements
\$SGGSY	NE	Get/Build Variable Symbol	\$SGGSY	<ul style="list-style-type: none"> Builds variable symbol in output buffer Computes symbol length and attribute
\$SGVST	NF	Variable Symbol Table—Find/Build	\$SGVST	<ul style="list-style-type: none"> Searches variable symbol table for variable symbol matches Adds symbol to table if no match is found and symbol meets conditions
\$SGGVA	NG	Variable Symbol Table—Value/Build	\$SGGVA	<ul style="list-style-type: none"> Places value of variable symbols in variable symbol table
\$SGNCB	NH	Number—Check/Build	\$SGNCB	<ul style="list-style-type: none"> Builds a number with length and attribute in output buffer

Figure 8-9. System Generation Directory (Part 1 of 2)

Module/Phase Name	Chart ID	Descriptive Name	Entry Point	Function
\$\$GCSB	NI	Character String— Check/Build	\$\$GCSB	<ul style="list-style-type: none"> Builds character string, with length and attribute in output buffer
\$\$GRED	NJ	Read Pre-processor Statement	\$\$GRED	<ul style="list-style-type: none"> Reads pre-processor statements from source library
\$\$GSTM	NK	LINK Statement Processor	\$\$GSTM	<ul style="list-style-type: none"> Processes LINK statement and puts prototype in save buffer
\$\$GROT	NL	Prototype Statement— Processor	\$\$GROT	<ul style="list-style-type: none"> Controls building of prototype statement entries in variable symbol table
\$\$GOPR	NM	Keyword Operand Processor	\$\$GOPR	<ul style="list-style-type: none"> Builds variable symbol table entries from keyword operands and values from prototype and pre-processor—call statements
\$\$GAIF	NN	AIF Statement Processor	\$\$GAIF	<ul style="list-style-type: none"> Translates variable symbols into values for AIF statements
\$\$GMST	NO	Model Statement Build	\$\$GMST	<ul style="list-style-type: none"> Analyzes model statement, rebuilds it in another buffer Replaces variable symbols with values
\$\$GSUB	NP	Subfield Analyzer	\$\$GSUB	<ul style="list-style-type: none"> Analyzes fields of data Translates variable symbols in the fields into values
\$\$GOUT	NQ	Model Statement Output File/Build Routine	\$\$GOUT	<ul style="list-style-type: none"> Writes analyzed model statement into caller's output file
\$\$GTBL	NR	TABLE Statement Processor	\$\$GTBL	<ul style="list-style-type: none"> Finds or builds entry for variable symbol on a table statement in variable symbol table
\$\$GDEF	NS	TABLE Definition Processor	\$\$GDEF	<ul style="list-style-type: none"> Matches value from table definition statement to argument and if equal, places value in variable symbol table
\$\$GAGO	NT	AGO Statement Processor	\$\$GAGO	<ul style="list-style-type: none"> Scans pre-processor statements until a match for variable symbol passed to this routine is found
\$\$GNNT	NU	MNOTE Statement	\$\$GNNT	<ul style="list-style-type: none"> Passes System Generation MNOTE statement to requestor's note processor
\$\$GSET	NV	SETB Statement Processor	\$\$GSET	<ul style="list-style-type: none"> Takes the value from the SETB statement and places the SETB value in variable symbol table entry
\$\$GGBL	NW	Global and Local Statement Processor	\$\$GGBL	<ul style="list-style-type: none"> Creates local or global entries in the variable symbol table
\$\$GPRI	NX	IPL Bootstrap Modify Routine	SGPRI	<ul style="list-style-type: none"> Allows user to select primary IPL system when he has both BASIC and DMS systems
\$\$GIVP	none	Installation Verification Message Routine	SGVIP	<ul style="list-style-type: none"> Prints out a message if a system has been generated properly

Figure 8-9. System Generation Directory (Part 2 of 2)



PART 9.

MAINTENANCE PROGRAMS



Section 1. Introduction

The following programs are supplied to the user for the maintenance of his IBM System/3 System Control Programs (SCP). The following sections have been omitted from this discussion: Method of Operation, Directory.

The following section describes, in detail, the organization of the Maintenance programs. Included in the following discussions are:

- Descriptions containing: name of the routine, input, output, function, and exits (both normal and error) from the routine.
- Main storage maps showing the routines that are found in main storage at the same time as the routine being described.
- Flowcharts showing the functional flow of the routine being described.

► **Field Engineering Maintenance Program (\$SGPTF)**

CHART: PA

FIGURE: 9-1

ENTRY POINT: \$SGPTF

FUNCTION:

- Temporarily patches other programs.
- Takes appropriate action with the relocation directory if the patch affects it.
- Keeps a record, on disk, of all PTFs (program temporary fixes) which have been applied to the system disk pack.
- Sets bit 6 of ATTRI in the directory entry for each module patched.

INPUT: Five control statements:

1. HDR (format—HDR PTFID, CKSUM, UNIT1 (,UNIT2))
2. PTF (format—PTF OMODULE NAME, LEVEL, CKSUM)
3. DATA (format—DATA CKSUM, DISP, HH, HHHH (R))
4. END (format—END CKSUM)
5. /*

OUTPUT:

- Patched modules, replaced in original positions on disk after PTF is applied
- The directory entry for each module that has been patched is updated.

EXIT:

- Normal: After the entire PTF has been applied and logged, control is passed to the End of Job transient (\$\$SPEJ).
- Error: In case of terminal error, control is passed to the supported Halt/Syslog routine.

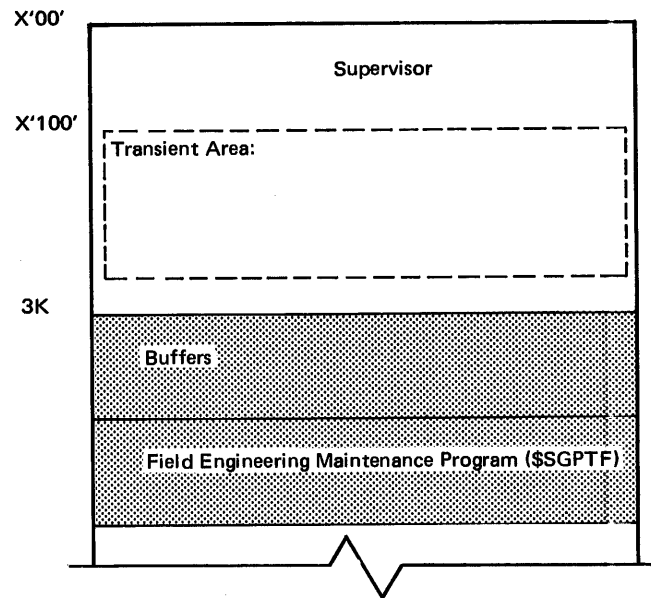


Figure 9-1. Main Storage Map for Field Engineering Maintenance Program (\$SGPTF)

► PTF Logging Program (\$SGPTG)

CHART: PB

FIGURE: 9-2

ENTRY POINT: \$SGPTG

FUNCTION: Logs the PTFID into the PTF log sector after the PTF has been successfully applied.

INPUT: Register 1 contains the address of a table containing the Q-code of the disk pack in which the PTF was applied, the PTF number, and the last five characters of each module that was patched.

OUTPUT: A 6-byte entry in the PTF log sector, recording that a PTF had been applied to the system disk pack

EXIT:

- Normal: Control is returned to the Field Engineering Maintenance Program (\$SGPTF).
- Error: Control is passed to the supported Halt/Syslog routine which in turn will issue an immediate cancel.

► System Maintenance (\$SGMNT)

CHART: None

FIGURE: None

ENTRY POINT: \$SGMNT

FUNCTION: Places \$MAINT (Library Maintenance) in the maintenance mode and then passes control to it.

INPUT: None

OUTPUT: None

EXIT:

- Normal: Control is passed to the Library Mainline (\$MAINT).
- Error: Control is passed to the supported Halt/Syslog routine.

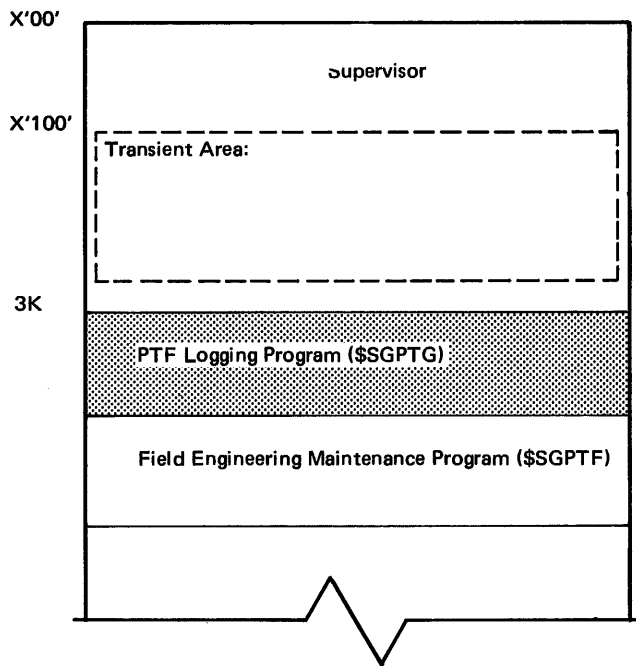


Figure 9-2. Main Storage Map for PTF Logging Program (\$SGPTG)

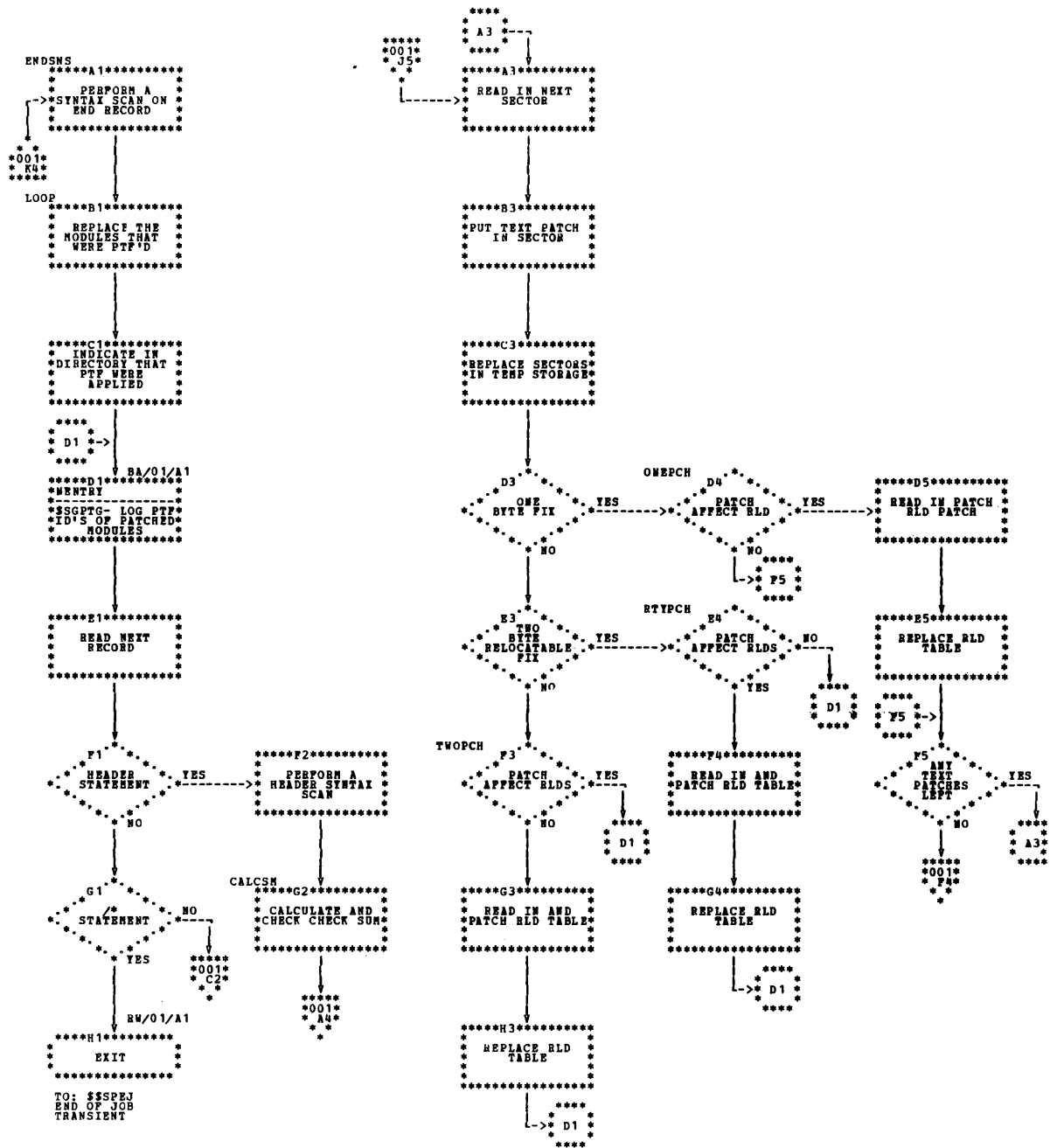


Chart PA. Field Engineering Maintenance Program (SSGPTF) (Part 2 of 2)

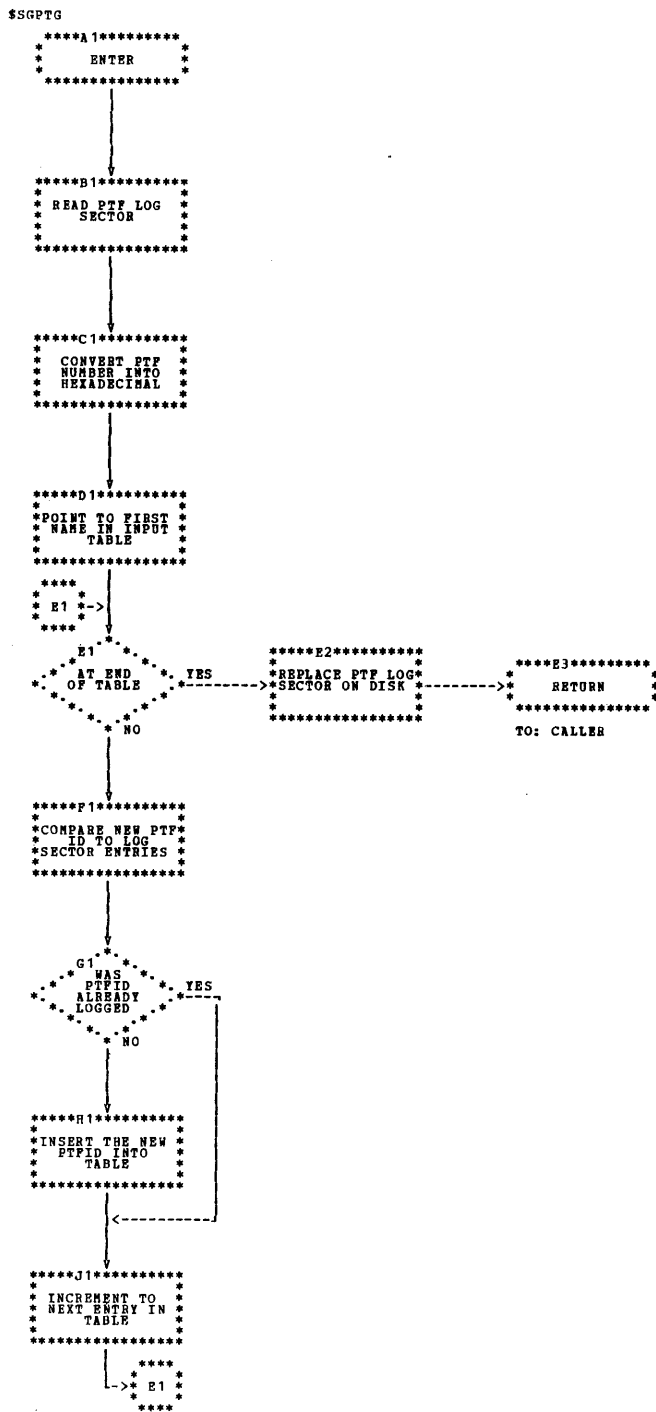


Chart PB. PTF Log Program (\$SGPTG)



APPENDIX A.
DATA AREAS

A

Disk System Dedicated Supervisor Data Area Formats

The following data areas are used by more than one program.

Supervisor Entry Points

This is an 18-byte area that is referenced by four LOAD IAR instructions located at absolute main storage addresses.

<i>Absolute Address</i>	<i>Number of Bytes</i>	<i>Contents</i>
X'00'	4	Load IAR for Dump Linkage program (CDUMPD)
X'04'	4	Load IAR for General Entry (NENTRY)
X'08'	4	Load IAR for IOS program (DIODSP)
X'0C'	4	Load IAR for the IOS Wait program (DIODWT)
X'11'	2	Address of the system communication area (SYS)

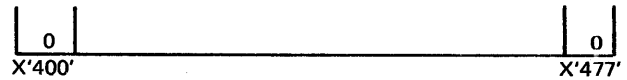
Transient Area

This 768-byte data area begins at X'100' and is used to contain the transient routines that are loaded into main storage.



Chain Image Area

This 120-byte data area is located at X'400' and is for the 5203 Line Printer. This area is maintained by the // IMAGE control card.



System Communication Region—NCPL1

This area is used to pass information between system programs. Figure A-1 shows the name of each entry, its displacement from the entry point (NCPL1), its byte count and its contents. The address of this communication region is located at X'11'.

Program Level Communication Region—N1COMM

This region is used to pass information between system programs within the same program level. Figure A-2 shows the name for each entry, its displacement from the entry point (NPIOB), its byte count, and its contents.

The address of this area is returned in XR2 after the user branches to General Entry with a request indicator byte (RIB) of X'00'.



Name	Hexadecimal Displacement	Number of Bytes	Contents
NCPL1	1	2	Address of program level 1 communication area
NCPL2	3	2	Address of program level 2 communication area (Disk System), X'0000' (Model 6)
NCXTAB	5	2	Address of transient scheduler tables
NCFCTR	6	1	Fetch trace I.D.
NCSGEN	6 }	1	Sysgen byte X'01' – Indicates to the Linkage Editor that Sysgen has control and wants the current chain image placed in the new supervisor X'80' – Indicates that \$SGNPP has read the first record from SWA.
NCPTZ	7	1	Printer size
NCLPSZ	8	1	Left tractor page size
NCRPSZ	9	1	Right tractor page size
NCSYSL	A	1	System List I.D.
NCSLOG	D	3	Syslog indicator C/S/device (Halt/Syslog) X'C0' – Console X'00' – Printer
NCSWRK	F	2	Cylinder/sector address of scheduler work area
NCSYSQ	10	1	Q byte for system pack from IPL routine X'A0' – R1 X'A8' – F1
NCOLIB	12	2	Cylinder/sector address of system object library
NCSCH1	13	1	Scheduler switches X'80' – Log device status X'40' – System date received X'20' – DPF system X'10' – Scheduler interlock for program level 1 X'08' – Scheduler interlock for program level 2 X'04' – Date format Indicator Off = MM/DD/YY Indicator On = DD/MM/YY X'02' } X'01' } Disk drive configuration 00 = F1, R1 01 = F1, R1, R2 11 = F1, R1, R2, F2
NCSMV1	14	1	'Data management/scheduler switches (Program level 1) X'80' – IPL successful X'04' – Consecutive multivolume files on R2 X'08' – Consecutive multivolume files on R1 X'01' – Other type of file on R2 X'02' – Other type of file on R1
NCSMV2	15	1	Data management/scheduler switches (Program level 2) X'04' – Consecutive multivolume files on R2 X'08' – Consecutive multivolume files on R1 X'01' – Other type of file on R2 X'02' – Other type of file on R1

Figure A-1. System Communication Region (Part 1 of 2)

Name	Hexadecimal Displacement	Number of Bytes	Contents
NCSCH	16	1	Scheduler byte X'80' — Printer interlock bit (printer has been opened) X'20' — Conversational Mode X'40' — Rollin now processing X'10' — Input for I-type program X'08' — Inquiry handled X'04' — PARTITION statement received X'02' — Rollout now processing X'01' — Inquiry pending
NCDSKO	18	2	Disk drive #1 I/O queue
NCSCH2	19	1	Partition value for level 2 (number of sectors)
NCMBSV	1A	1	Model 6 control byte
NCXTA	1B	1	Reserved
NCSTOR	2F	20	Transient storage area
NC@CIO	31	2	Main storage address of keyboard interrupt routine
NCRCSS	34	3	C/S/# of Rollout area
The following data areas are used only by a DPF system			
NCSXR1	36	2	XR1 } This is a storage area for interrupt level 0 XR2 } PSR }
NCSXR2	38	2	
NCSPSR	3A	2	
NCSNS	3B	1	
			Reader select switch sense information X'80' — On = level 1 Off = level 2 X'40' — Cancel X'20' — MFCU X'10' — AUX reader X'08' — Printer keyboard
NCHALT	3D	2	Address of resident halt routine (NPHALT)
NC@PIO	36	2	Address of printer IOB active
NCLPLC	37	1	Left page line count
NCRPLC	38	1	Right page line count
NCEPL	39	1	Print element location

Figure A-1. System Communication Region (Part 2 of 2)



Name	Hexadecimal Displacement	Number of Bytes	Contents
NPIOB	0	0	Entry for program level IOB
NPCHAN	1	2	IOS chain pointer (disk IOS)
NIOCMP	2	1	Completion code (disk IOS)
NIOQB	3	1	IOB Q byte
NIORB	4	1	IOB R byte
NIOCB	5	1	IOB cylinder byte
NIOSB	6	1	IOB sector byte
NIONB	7	1	Number of sectors to be read-1
NIODAT	9	2	Data buffer address
NIOSNS	B	2	Sense bytes (disk IOS)
NIOERR	C	1	IOS error retry counter
NIOFLG	D	1	Flag bits (disk IOS)
NIOOBR	11	4	OBR - SDR transient routine save areas for the caller's ARR, XR2 registers
NPDTF@	13	2	First DTF address
NPRIB	14	1	Program request indicator byte (RIB)
NPBEG	16	2	Program level beginning address
NPEND	18	2	Program level end address
NPQ	19	1	Program level Q byte (as in the // LOAD statement)
NPRLF	1B	2	Program relocation factor
NPCYL	1D	2	C/S of load name for this program level
NPOLIB	1F	2	C/S of program object library
NPXR1	21	2	Register 1 save area
NPXR2	23	2	Register 2 save area
NPNSI	25	2	Return address
NPORLF	27	2	Overlay relocation factor supplied by loader
NPTXT	29	2	Overlay text address supplied by relocation program
NPTEMP	2B	2	Temporary storage for resident loader (NLOADR)
Parameter List for the Resident Loader (NLOADR)			
NPCS	2D	2	C/S address of module to be loaded
NP#S	2E	1	Number of sectors to be read
NPLNK	30	2	Linkage edited address
NPRLD	31	1	Displacement of relocation directory (RLD)
NPENT	33	2	Address of entry point of loaded module
NPLOD	35	2	Load address
NPEOJ	36	1	End of job I.D.
			X'40' - Successful IPL
			X'20' - Library maintenance
			X'10' - Cancel pending
			X'0E' - RPG
			X'01' - Rollout bit
NPUPSI	37	1	UPSI switches
NPNAME	3D	6	Program name that is currently being executed
NPRELS	3E	1	Release level
NPPDATE	44	6	Program date
NPSYSI	48	4	SYSIN indicator (C/S/number of sectors/device info)
			'F0' - Hopper 1 MFCU
			'F8' - Hopper 2 MFCU
			'F4' - 1442
			'C0' - Console I/O

Figure A-2. Program Level Communication Region (Part 1 of 3)

Name	Hexadecimal Displacement	Number of Bytes	Contents
Scheduler parameters			
NPSCH1	49	1	Reader/Interpreter switches X'01' – IPL mode X'02' – INTER mode X'04' – INTRA mode X'08' – Override request X'10' – Procedure being processed X'20' – // SWITCH received X'40' – // COMPILE received X'80' – // DATE received (INTRA)
NPSCH2	4A	1	Scheduler switch X'01' – Controlled cancel X'02' – Immediate cancel X'04' – Clear Sysin device X'08' – End of job halt X'10' – A file statement received X'20' – // READ from Sysin device X'40' – Utility control cards in scheduler work area X'80' – Continuation of OCL statement
NPSCH3	4B	1	Scheduler switches X'80' – Tag sort required X'40' – Allocate transient required X'20' – Source required by RPG compiler X'10' – Logging (Halt/Syslog) Indicator Off – logging specified Indicator On – logging not specified Statement origin X'08' – MVF file allocated X'04' – Additional procedure statement X'02' – Override statement from Sysin device X'01' – Program level Indicator Off – level 1 Indicator On – level 2
NPOBJQ	4C	1	Object deck output Q code
NPBPSD	4D	1	Selected status of Sysin device X'80' – MFCU Disk System, Data Recorder Model 6 X'40' – Console I/O X'20' – 1442 X'10' – AUX on reader select switch X'08' – Console on reader select switch X'01' – Nested procedure
NPSCH4	4E	1	Allocation information X'80' – This program level is active X'40' – // IMAGE received X'20' – // FORMS received X'10' – F1 needed for allocation X'08' – R1 needed for allocation X'04' – F2 needed for allocation X'02' – R2 needed for allocation X'01' – EOJ scheduler bit On – Initial EOJ logic has been performed Off – Initial EOJ logic has not been performed

Figure A-2. Program Level Communication Region (Part 2 of 3)

Name	Hexadecimal Displacement	Number of Bytes	Contents
NPSCH8	4F	1	Type of files needed X'80' – Reader statement received X'40' – Console in use by Data Management X'20' – Valid FILE card received X'10' – Shared I/O program X'08' – 'I' or 'B' type program X'04' – Allocate unsuccessful X'02' – Maximum number of tracks for allocation are available X'01' – Minimum number of tracks for allocation are available
NPSCH5	50	1	Reader Select switch setting, DPF only X'80' – Level 1 X'10' – AUX position X'08' – P-KB position X'20' – MFCU position
NPSCH6	52	2	Save area for IAR pointer
NPSCH7	53	1	Save of last Sysin assignment
NPWKB	55	2	Disk IOS work area
NPSCH9	56	1	Disk log unit ID for EOJ X'80' – F2 X'40' – R2 X'20' – F1 X'10' – R1 } These units are logged only once X'08' – F2 X'04' – R2 X'02' – F1 X'01' – R1 } All these units are logged at end of job
The following data areas are used only by the Resident Halt Routine (NPHALT) in a DPF system			
NPHALT	57	1	Program level HPL
NPHLTQ	58	1	Q code for HPL
NPHLTR	59	1	R code for HPL
NPHBCH	5D	4	Branch for Resident Halt Routine (NPHALT)
NPHXR2	5F	2	Parameter save area
NPHTRN	67	8	Transient storage area
NPHQSV	84	29	Transient queue save area
NPUTIL	85	1	Utility interlock byte. The utilities related to this byte are: \$COPY, \$INIT, \$ALT, and \$BUILD X'80' – F2 allocated X'40' – R2 allocated X'20' – F1 allocated X'10' – R1 allocated X'08' – F2 in use X'04' – R2 in use X'02' – F1 in use X'01' – R1 in use
NPSAVE	87	2	IAR save area used on cancel condition
NPJUSV	8F	8	Used to save registers on a 'JU' Halt

Figure A-2. Program Level Communication Region (Part 3 of 3)

The following data areas are used by more than one Supervisor program.

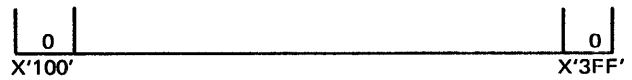
Supervisor Entry Points

This is an 18-byte area that is referenced by four LOAD IAR instructions located at absolute main storage addresses.

<i>Absolute Address</i>	<i>Number of Bytes</i>	<i>Contents</i>
X'00'	4	Load IAR for Dump Linkage program (CDUMPD)
X'04'	4	Load IAR for General Entry (NENTRY)
X'08'	4	Load IAR for IOS program (DIODSP)
X'0C'	4	Load IAR for the IOS Wait program (DIODWT)
X'11'	2	Address of the system communication area (SYS)

Transient Area

This 768-byte data area begins at X'100' and is used to contain the transient routines that are loaded into main storage.



Keyboard Translate Table

This 64-byte data area is located at X'400' and is for the Keyboard Printer. This area is initialized during system generation to one of the nine possible translate tables. See *System Generation*, Keyboard Selection Routine (\$SGKBD).

System Communication Region—NCPL1

This area is used to pass information between system programs. Figure A-3 shows the name of each entry, its displacement from the entry point (NCPL1), its byte count and its contents. The address of this communication region is located at X'11'.



Name	Hexadecimal Displacement	Number of Bytes	Contents
NCPL1	1	2	Address of program level 1 communication area
NCPL2	3	2	Address of program level 2 communication area (Disk System), X'0000' (Model 6)
NCXTAB	5	2	Address of transient scheduler tables
NCFCTR	6 }	1	Fetch trace I.D.
NCSGEN	6 }	1	Sysgen byte X'01' — Indicates to the Linkage Editor that Sysgen has control and wants the current chain image placed in the new supervisor. X'80' — Indicates that \$SGNPP has read the first record from SWA.
NCPRZ	7	1	Printer size
NCLPSZ	8	1	Left tractor page size
NCRPSZ	9	1	Right tractor page size
NCSYSL	A	1	System List I.D.
NCSLOG	D	3	Syslog indicator C/S/device (Halt/Syslog) X'C0' — Console X'00' — Printer
NCSWRK	F	2	Cylinder/sector address of scheduler work area
NCSYSQ	10	1	Q byte for system pack from IPL routine X'A0' — R1 X'A8' — F1
NCOLIB	12	2	Cylinder/sector address of system object library
NCSCH1	13	1	Scheduler switches X'80' — Log device status X'40' — System date received X'20' — DPF system X'10' — Scheduler interlock for program level 1 X'08' — Scheduler interlock for program level 2 X'04' — Date format Indicator Off = MM/DD/YY Indicator On = DD/MM/YY X'02' — } Disk drive configuration X'01' — } 00 = F1, R1 01 = F1, R1, R2 11 = F1, R1, R2, F2
NCSMV1	14	1	Data management/scheduler switches (Program level 1) X'80' — IPL successful X'04' — Consecutive multivolume files on R2 X'08' — Consecutive multivolume files on R1 X'01' — Other type of file on R2 X'02' — Other type of file on R1
NCSMV2	15	1	Data management/scheduler switches (Program level 2) X'04' — Consecutive multivolume files on R2 X'08' — Consecutive multivolume files on R1 X'01' — Other type of file on R2 X'02' — Other type of file on R1

Figure A-3. System Communication Region (Part 1 of 2)

Name	Hexadecimal Displacement	Number of Bytes	Contents
NCSCH	16	1	Scheduler byte X'80' – Printer interlock bit (printer has been opened) X'20' – Conversational mode X'40' – Rollin now processing X'10' – Input for I-type program X'08' – Inquiry handled X'04' – PARTITION statement received X'02' – Rollout now processing X'01' – Inquiry pending
NCDSKQ	18	2	Disk drive #1 I/O queue
NCSCH2	19	1	Partition value for level 2 (number of sectors)
NCMBSV	1A	1	Model 6 control byte
NCXTA	1B	1	Reserved
NCSTOR	2F	20	Transient storage area
NC@CIO	31	2	Main storage address of keyboard interrupt routine
NCRCSS	34	3	C/S/# of Rollout area
The following data areas are used only by a DPF system			
NCSXR1	36	2	XR1 } This is a storage area for interrupt level 0 XR2 } PSR }
NCSXR2	38	2	
NCSPSR	3A	2	
NCSNS	3B	1	
			Reader select switch sense information X'80' – On = level 1 Off = level 2 X'40' – Cancel X'20' – MFCU X'10' – AUX reader X'08' – Printer keyboard
NCHALT	3D	2	Address of resident halt routine (NPHALT)
NC@PIO	36	2	Address of printer IOB active
NCLPLC	37	1	Left page line count
NCRPLC	38	1	Right page line count
NCPEL	39	1	Print element location

Figure A-3. System Communication Region (Part 2 of 2)

Program Level Communication Region—N1COMN

This region is used to pass information between system programs. Figure A-4 shows the name for each entry, its displacement from the entry point (N1COMN), its byte count, and its contents.

The address of this area is returned in XR2 after the user branches to General Entry with a request indicator byte (RIB) of X'00'.



Name	Hexadecimal Displacement	Number of Bytes	Contents
NPIOB	0	0	Entry for program level IOB
NPCHAN	1	2	IOS chain pointer (disk IOS)
NIOCMP	2	1	Completion code (disk IOS)
NIOQB	3	1	IOB Q byte
NIORB	4	1	IOB R byte
NIOCB	5	1	IOB cylinder byte
NIO SB	6	1	IOB sector byte
NIONB	7	1	Number of sectors to be read -1
NIODAT	9	2	Data buffer address
NIOSNS	B	2	Sense bytes (disk IOS)
NIOERR	C	1	IOS error retry counter
NIOFLG	D	1	Flag bits (disk IOS)
NIOOBR	11	4	OBR - SDR transient routine save areas for the caller's ARR, XR2 registers
NPDTF@	13	2	First DTF address
NPRIB	14	1	Program request indicator byte (RIB)
NPBEG	16	2	Program level beginning address
NPEND	18	2	Program level end address
NPQ	19	1	Program level Q byte (as in the // LOAD statement)
NPRLF	1B	2	Program relocation factor
NPCYL	1D	2	C/S of load name for this program level
NPOLIB	1F	2	C/S of program object library
NPXR1	21	2	Register 1 save area
NPXR2	23	2	Register 2 save area
NPNSI	25	2	Return address
NPORLF	27	2	Overlay relocation factor supplied by loader
NPTXT	29	2	Overlay text address supplied by relocation program
NPTEMP	2B	2	Temporary storage for resident loader (NLOADR)
Parameter List for the Resident Loader (NLOADR)			
NPCS	2D	2	C/S address of module to be loaded
NP#S	2E	1	Number of sectors to be read
NPLNK	30	2	Linkage edited address
NPRLD	31	1	Displacement of relocation directory (RLD)
NPENT	33	2	Address of entry point of loaded module
NPLOD	35	2	Load address
NPEOJ	36	1	End of job I.D.
			X'40' - Successful IPL
			X'20' - Library maintenance
			X'10' - Cancel pending
			X'0E' - RPG
			X'01' - Rollout bit
NPUPSI	37	1	UPSI switches
NPNAME	3D	6	Program name that is currently being executed
NPRELS	3E	1	Release level
NPDATE	44	6	Program date
NPSYSI	48	4	SYSIN indicator (C/S/number of sectors/device info)
			'F0' - Hopper 1 MFCU
			'F8' - Hopper 2 MFCU
			'F4' - 1442
			'C0' - Console I/O

Figure A-4. Program Level Communication Region (Part 1 of 3)

Name	Hexadecimal Displacement	Number of Bytes	Contents
Scheduler parameters			
NPSCH1	49	1	<p>Reader/Interpreter switches</p> <p>X'01' – IPL mode X'02' – INTER mode X'04' – INTRA mode X'08' – Override request X'10' – Procedure being processed X'20' – // SWITCH received X'40' – // COMPILE received X'80' – // DATE received (INTRA)</p>
NPSCH2	4A	1	<p>Scheduler switch</p> <p>X'01' – Controlled cancel X'02' – Immediate cancel X'04' – Clear Sysin device X'08' – End of job halt X'10' – A file statement received X'20' – // READ from Sysin device X'40' – Utility control cards in scheduler work area X'80' – Continuation of OCL statement</p>
NPSCH3	4B	1	<p>Scheduler switches</p> <p>X'80' – Tag sort required X'40' – Allocate transient required X'20' – Source required by RPG compiler X'10' – Logging (Halt/Syslog) Indicator Off – logging specified Indicator On – logging not specified</p> <p>Statement origin</p> <p>X'08' – MVF file allocated X'04' – Additional procedure statement X'02' – Override statement from Sysin device X'01' – Program level Indicator Off – level 1 Indicator On – level 2</p>
NPOBJQ	4C	1	Object deck output Q code
NPBPSD	4D	1	<p>Selected status of Sysin device</p> <p>X'D0' – Data Recorder (Model 6) X'B0' – CRT (Model 6) X'A0' – Matrix Printer (Model 6) X'80' – MFCU X'40' – Console I/O X'20' – 1442 X'10' – AUX on reader select switch X'08' – Console on reader select switch X'01' – Nested procedure</p>
NPSCH4	4E	1	<p>Allocation information</p> <p>X'80' – This program level is active X'40' – // IMAGE received X'20' – // FORMS received X'10' – F1 needed for allocation X'08' – R1 needed for allocation X'04' – F2 needed for allocation X'02' – R2 needed for allocation X'01' – EOJ scheduler bit On – Initial EOJ logic has been performed Off – Initial EOJ logic has not been performed</p>



Figure A-4. Program Level Communication Region (Part 2 of 3)

Name	Hexadecimal Displacement	Number of Bytes	Contents
NPSCH8	4F	1	Type of files needed X'80' - Reader statement received X'40' - Console in use by Data Management X'20' - Valid FILE card received X'10' - Shared I/O program X'08' - 'I' or 'B' type program X'04' - Allocate unsuccessful X'02' - Maximum number of tracks for allocation are available X'01' - Minimum number of tracks for allocation are available
NPSCH5	50	1	Reader Select switch setting, DPF only X'80' - Level 1 X'10' - AUX position X'08' - P-KB position X'20' - MFCU position
NPSCH6	52	2	Save area for IAR pointer
NPSCH7	53	1	Save of last Sysin assignment
NPWKB	55	2	Disk IOS work area
NPSCH9	56	1	Disk log unit ID for EOJ X'80' - F2 X'40' - R2 X'20' - F1 X'10' - R1 } These units are logged only once X'08' - F2 X'04' - R2 X'02' - F1 X'01' - R1 } All these units are logged at end of job
The following data areas are used only by the Resident Halt Routine (NPHALT) in a DPF system			
NPHALT	57	1	Program level HPL
NPHLTQ	58	1	Q code for HPL
NPHLTR	59	1	R code for HPL
NPHBCH	5D	4	Branch for Resident Halt Routine (NPHALT)
NPHXR2	5F	2	Parameter save area
NPHTRN	67	8	Transient storage area
NPHQSV	84	29	Transient queue save area
NPUTIL	85	1	Utility interlock byte. The utilities related to this byte are: \$COPY, \$INIT, \$ALT, and \$BUILD X'80' - F2 allocated X'40' - R2 allocated X'20' - F1 allocated X'10' - R1 allocated X'08' - F2 in use X'04' - R2 in use X'02' - F1 in use X'01' - R1 in use
NPSAVE	87	2	IAR save area used on cancel condition
NPJUSV	8F	8	Used to save registers on a 'JU' Halt

Figure A-4. Program Level Communication Region (Part 3 of 3)

The following data areas are used by more than one Scheduler routine.

Configuration Record

The configuration record consists of 4-byte segments. Each segment is allocated to a different device that can be supported by the System/3 Disk Systems. The first byte of each segment indicates to the system if the device is supported and if it is currently being used. The remaining three bytes reflect unique characteristics of that device. Figure A-5 shows the devices that might be supported by the System/3 Disk Systems and their unique characteristics. Reference is made to the significance of bits in the configuration record. For example, X'80' indicates that the 0 bit is the indicator; X'40' indicates that bit 1 is the indicator, and so forth. The system configuration record is located at C/S address 00/04 on the system pack.



Label	Hex Displacement	Number of Bytes	Content and Possible Byte Settings
MFCU	00	4	MFCU X'80' - Device supported Three bytes not used
PRNTR	04	4	22LC Line Printer X'80' - Device supported X'04' - DFC X'02' } Print X'01' } Positions 00 - 96 PP 01 - 120 PP 11 - 132 PP Three bytes not used
#5471	08	4	Console I/O (Keyboard-Printer) X'80' - Device supported Three bytes not used
#1442	0C	4	1442 80-Column Card Reader X'80' - Device supported Three bytes not used
DISK	10	4	Disk ① { X'80' - Device supported X'40' - BASIC X'10' - Full capacity X'08' - 1/2 capacity X'02' } Disk drive X'01' } configuration ② { 00 - F1, R1 01 - F1, R1, R2 11 - F1, R1, R2, F2 ③ { X'40' - Overlap supported BASIC ④ { X'08' - Full capacity X'04' - 1/2 capacity X'10' - F1, R1, R2 X'01' - F1, R1, R2, F2
MTXPTR	14	4	Matrix Printer ① { X'80' - Device supported X'40' - BASIC X'04' - Line mode X'02' - 22 inch/200 print positions X'01' - 13 inch/132 print positions ② { X'80' - DFC X'40' - VFC X'20' - Pin BASIC ③ { X'08' - Matrix printer X'04' - Line printer X'02' - 22 inch (MP)/200 print positions X'01' - 13 inch (MP)/132 print positions One byte not used

Figure A-5. Configuration Record Formats (Part 1 of 3)

Label	Hex Displacement	Number of Bytes	Content and Possible Byte Settings
KEYBRD	18	4	Keyboard ① { X'80' – Device supported X'40' – BASIC ② { X'80' – 16 command keys X'40' – 8 command keys National Languages X'01' – Domestic (English) X'02' – Austria/Germany X'03' – Belgium/France X'04' – Denmark ③ { X'05' – Norway X'06' – Finland/Sweden X'07' – Spain X'08' – Brazil/Portugal X'09' – United Kingdom One byte not used
LCD	1C	4	Ledger Card Device X'80' – Device supported Three bytes not used
DATREC	20	4	Data Recorder X'80' – Device supported X'40' – BASIC Three bytes not used
BSCA	24	4	Binary Synchronous Communications Adapter ① { X'80' – Device supported X'80' – EBCDIC ② { X'40' – ASCII X'20' – RJE (Disk System Only) Two bytes not used
CRT	28	4	Cathode Ray Tube X'80' – Device supported X'40' – BASIC Three bytes not used
#5475	2C	4	Data Entry Keyboard X'80' – Device supported Three bytes not used
MICR	30	4	Magnetic Ink Character Reader X'80' – Device supported Three bytes not used
	38	1	End of devices indicator (X'FF')
CONSOL (DPF)	39	2	Console on reader select switch X'80' – Device supported Device selected 2nd Byte { X'00' – MFCU1 X'01' – MFCU2 X'02' – Console X'03' – 1442

Figure A-5. Configuration Record Formats (Part 2 of 3)



Label	Hex Displacement	Number of Bytes	Content and Possible Byte Settings
AUX (DPF)	3B	2	AUX on reader select switch X'80' – Device supported Device selected 2nd Byte { X'00' – MFCU1 X'01' – MFCU2 X'02' – Console X'03' – 1442
BISCOR	3D	1	BASIC Core Size X'01' – 8K X'02' – 12K X'04' – 16K
Available space in configuration record			
SIAM	F8	1	Shared I/O Access Method X'80' – Supported
TYPE	F9	1	System type X'00' – Disk System X'40' – BASIC, Model 6 X'80' – Model 6 (Commercial)
BITS	FA	1	Bit significant indicators X'80' – Reserved X'40' – Inquiry X'20' – DPF
DATFMT	FB	1	Date format X'04' – DDMMYY World Trade X'00' – MMDDYY Domestic
LINEPP	FC	1	Lines/page
CORSIZ	FF	3	Main storage size XL3'002000' – 8K XL3'003000' – 12K XL3'004000' – 16K XL3'006000' – 24K XL3'008000' – 32K

Figure A-5. Configuration Record Formats (Part 3 of 3)

Volume Label

This is a one sector (256-byte) area located in the third sector of cylinder 0 on any disk pack. Figure A-6 shows the contents of this sector. The system directory (bytes 11-255) reflects the status and location of the source and object libraries.

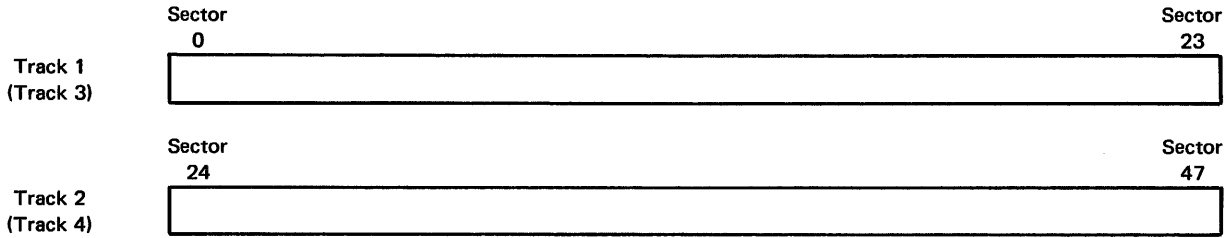
Scheduler Work Area (SWA)

This data area is 48 sectors (two tracks) if the system is dedicated, 96 sectors (four tracks) if operating within a DPF system. Figure A-7 shows the contents of the dedicated system's 48 sectors. When in a DPF system, two tracks are dedicated to each program level. Each set of tracks has the same format as the tracks for the dedicated system with the exception that sector 0 is not used on the second cylinder. An additional data area of 48 sectors (two tracks) is added if roll-in, roll-out is supported.

Decimal Displacement	Hexadecimal Displacement	Number of Bytes	Contents
0-2	0-2	3	Label identifier (VOL)
3-8	3-8	6	Volume identifier, 1-6 characters
9-10	9-A	2	Volume table of contents (VTOC) pointer C/S
11-12	B-C	2	Source directory pointer C/S
13-14	D-E	2	Next available library sector C/S
15-16	F-10	2	End of library C/S
17-18	11-12	2	Number of directory sectors
19-20	13-14	2	Number of permanent library sectors
21-22	15-16	2	Number of active library sectors
23-24	17-18	2	Number of available library sectors
25-36	19-24	12	Reserved
37-38	25-26	2	Object directory pointer C/S
39-40	27-28	2	End of directory C/S
41-42	29-2A	2	Start of library C/S
43-44	2B-2C	2	Allocated end of library C/S
45-46	2D-2E	2	Extended end of library C/S
47-48	2F-30	2	Number of available permanent directory entries
49-50	31-32	2	Number of available temporary directory entries
51-53	33-35	3	First temporary directory entry (C/S/D)
54-56	36-38	3	Next available temporary directory entry (C/S/D)
57-58	39-3A	2	Next available library sector for perms C/S
59-60	3B-3C	2	Next available library sector for temps C/S
61-62	3D-3E	2	Number of available library sectors for perms (after last perm)
63-64	3F-40	2	Number of available library sectors for temps
65-66	41-42	2	Number of active library sectors
67-68	43-44	2	Number of active O. type permanent sectors
69-70	45-46	2	Number of active R. type permanent sectors
71	47	1	Valid system indicator (X'80', commercial system)
72-73	48-49	2	Rollin/rollout pointer C/S
74	4A	1	Rollin/rollout size
75-76	4B-4C	2	Scheduler work area pointer C/S
77	4D	1	Scheduler work area size
78-81	4E-51	4	Start and end of libraries (F1) C/S C/S
82-91	52-5B	10	Owner identification
92-105	5C-69	14	Device constants
106-117	6A-75	12	Alternate track assignments
118-168	76-A8	51	Available tracks, format 5
169-180	A9-B3	11	Copypack save area
181-205	B4-CD	26	Unused
206-215	CE-D7	10	Scientific system file indicator
216-239	D8-EF	24	Suspected defective track indicators
240-255	F0-FF	16	Scientific system indicators

Figure A-6. Volume Label Format





Sector	Name	Displacement (Hex)	Bytes	Contents
0		0	6	System date, EBCDIC characters from the DATE control statement
		6	14	System temporary configuration record maintained by the IOS Open routines. Bytes relative to the start of the configuration area displacement: 00 – MFCU 01 – Printer 02 – 5471 Console I/O 03 – 1442 04 – Disk Drive 05 – Matrix Printer 06 – Keyboard 07 – Ledger Card Device 08 – Data Recorder 09 – BSCA 0A – Cathode Ray Tube 0B – 5475 Data Entry Keyboard 0C – Magnetic Ink Character Reader
		14	52	These bytes are not used
1	FORMT1	0	2	Cylinder/sector of the first sector that contains the format one images used by the Scheduler and Data Management
		2	2	Cylinder/sector address of the last sector containing the format one images
		4	2	Cylinder/sector address of the last sector that information was read from
		6	1	Displacement into the last format one sector from which information was read
		7	2	Cylinder/sector address of the last format one sector into which information was written
	UTILCC	9	1	Displacement into the last format one sector into which information was written
		A	2	Cylinder/sector address of the first sector that contains the utility control statements
		C	2	Cylinder/sector address of the last sector containing the utility control statements
		E	2	Cylinder/sector address of the last utility control card sector that information was read from
10	1	Displacement into the last utility control card sector from which information was read		

Figure A-7. Scheduler Work Area Format (Part 1 of 6)

Sector	Name	Displacement (Hex)	Bytes	Contents
1	INITAB	11	2	Cylinder/sector address of the last utility control statement sector into which information was written
		13	1	Displacement into the last utility control statement sector into which information was written
		14	2	Cylinder/sector address of the first sector that contains the initiator table
		16	2	Cylinder/sector address of the last sector containing the initiator table
		18	2	Cylinder/sector address of the last initiator table sector that information was read from
		1A	1	Displacement into the last initiator table sector from which information was read
		1B	2	Cylinder/sector address of the last initiator table sector into which information was written
	MISCEL	1D	1	Displacement into the last initiator table sector into which information was written
		1E	2	Cylinder/sector address of the first sector that contains miscellaneous work area for the Scheduler and Data Management
		20	2	Cylinder/sector address of the last sector containing the miscellaneous work area
		22	2	Cylinder/sector address of the last miscellaneous work area sector that information was read from
		24	1	Displacement into the last miscellaneous work area sector from which information was read
	MISCEL	25	2	Cylinder/sector address of the last miscellaneous work area sector into which information was written
		27	1	Displacement into the last miscellaneous work area sector into which information was written
	CONFIG	28	58	This area contains the program's temporary configuration record. This record is used with the system configuration record to maintain proper feature checking. (See sector 0, with the displacement of 6.)
	FORMAT5	68	2	Cylinder/sector address of the format 5's for the mounted packs
		6A	2	The attributes of the current program (these attributes are the same as those found in the directory)
		6C	6	Source name from the COMPILE control statement
		72	1	Source Unit from the COMPILE control statement
		73	1	Sector where last format seven was placed
74		1	Displacement into sector where last format seven was placed	
*Note: The entries in sector 2 are referenced by the index located in sector 1.				

Figure A-7. Scheduler Work Area Format (Part 2 of 6)

Sector	Name	Displacement (Hex)	Bytes	Contents
2		0	64	F5 image for fixed - one (F1) as follows:
	F5PACK	0	6	Pack ID
	F5ATTB	7	1	Bit 0 - 0 Not used - 1 Not used
				Bit 1 - 0 Not used - 1 This unit is dedicated to this level
				2 - 0 Not used - 1 This is this level's program or system disk pack
				3 - 0 Not used - 1 This program level is using this disk pack
				4 - 0 Not used - 1 Not used
				5 - 0 Not used - 1 Not used
				6 - 0 Not used - 1 Not used
				7 - 0 Not used - 1 Not used
	F5CNTR	8	1	This byte contains the number of OPEN files
		9	5	Not used
		E	51	Format 5 for this disk pack
		40	64	Format 5 image for removable 1 (R1) same format as bytes 0-63
		80	64	Format 5 image for fixed - 2 (F2) same format as bytes 0-63
		C0	64	Format 5 image for removable - 2 (R2) same format as bytes 0-63
3-15	F1TAG	0	1	Tag ID of index pointer
	F1CHAN	1	2	Chain address
	F1LABL	3	8	File label
	F1DATE	B	6	System date
	F1RTIN	11	1	Retain type

Figure A-7. Scheduler Work Area Format (Part 3 of 6)

Sector	Name	Displacement (Hex)	Bytes	Contents
3-15	F1TYPE	12	2	File type Left byte Hex value Meaning 80 ISAM 40 Consecutive 20 Direct 10 MVF 08 Input 04 Output 02 Update 01 Add Right byte Hex value Meaning 80 ISAM 40 Consecutive 20 Direct 10 MVF 08 Last pack for consecutive MVF 04 Sequential add 02 Random add 01 Unordered add
	F1RECL	14	2	Record length
	F1KEYL	16	1	Key length
	F1KEYO	17	2	Key location
	F1LSTR	19	3	C/S/D of last record
	F1LSTK	1C	3	C/S/D of last key
	F1STDA	1F	2	C/S of start key
	F1ENDA	21	2	C/S of end data
	F1STIX	23	2	C/S of start index
	F1ENIX	25	2	C/S of end index
	F1RECN	27	3	Number of records at creation
	F1SEQN	2A	1	Volume sequence number for MVF
	F1PACK	2B	6	Pack ID for this file
	F1NAME	31	8	File name
	F1UNIT	39	1	Unit address of this file

Figure A-7, Scheduler Work Area Format (Part 4 of 6)

Sector	Name	Displacement (Hex)	Bytes	Contents
3-15	F1ATT1	3A	1	Attribute byte 1 as follows: Bit 1 (X'80') 0 – Offline - Pack and units not equal 1 – Online - Pack and units are equal Bit 2 (X'40') 0 – File processed 1 – File not processed Bit 3 (X'20') 0 – Not deferred mount 1 – Deferred mount Bit 4 (X'10') 0 – Not multivolume file 1 – Multivolume file Bit 5 (X'08') 0 – Old 1 – New Bit 6 (X'04') 0 – No location given 1 – Location given Bit 7 (X'02') 0 – Space and location not equal 1 – Space and location given and equal Bit 8 (X'01') 0 – Unique labels 1 – Two labels are the same
	F1ATT2	3B	1	Attribute byte 2 as follows: Bit 1 (X'80') 0 – Not primary pack of a multivolume file 1 – Primary pack of MVF Bit 2 (X'40') 0 – 1 – RETAIN-A on FILE card Bit 3 (X'20') 0 – 1 – Remove F1 from VTOC at EOJ Bit 4 (X'10') 0 – Not open 1 – Open Bit 5 (X'08') 0 – Not closed 1 – Closed Bit 6 (X'04') 0 – Reassigns as usual at end of job 1 – Do not reassign at end of job Bit 7 (X'02') 0 – Not used 1 – Not used Bit 8 (X'01') 0 – File not allocated 1 – File allocated
	F1ATT3	3C	1	Attribute byte 3 as follows: Bit 1 (X'80') 0 – F1 not used 1 – F1 used Bit 2 (X'40') 0 – F2 not used 1 – F2 used Bit 3 (X'20') 0 – R1 not used 1 – R1 used Bit 4 (X'10') 0 – R2 not used 1 – R2 used Bit 5 (X'08') 0 – Not load to old file 1 – Load to old file Bit 6 (X'04') 0 – Not used 1 – F7 image given for this file Bit 7 (X'02') 0 – No DTF supplied for this file 1 – DTF supplied for this file Bit 8 (X'01') 0 – Special F1 image not used 1 – Special F1 image used
	F1INDX	3D	1	DTF scheduler work area index

Figure A-7. Scheduler Work Area Format (Part 5 of 6)

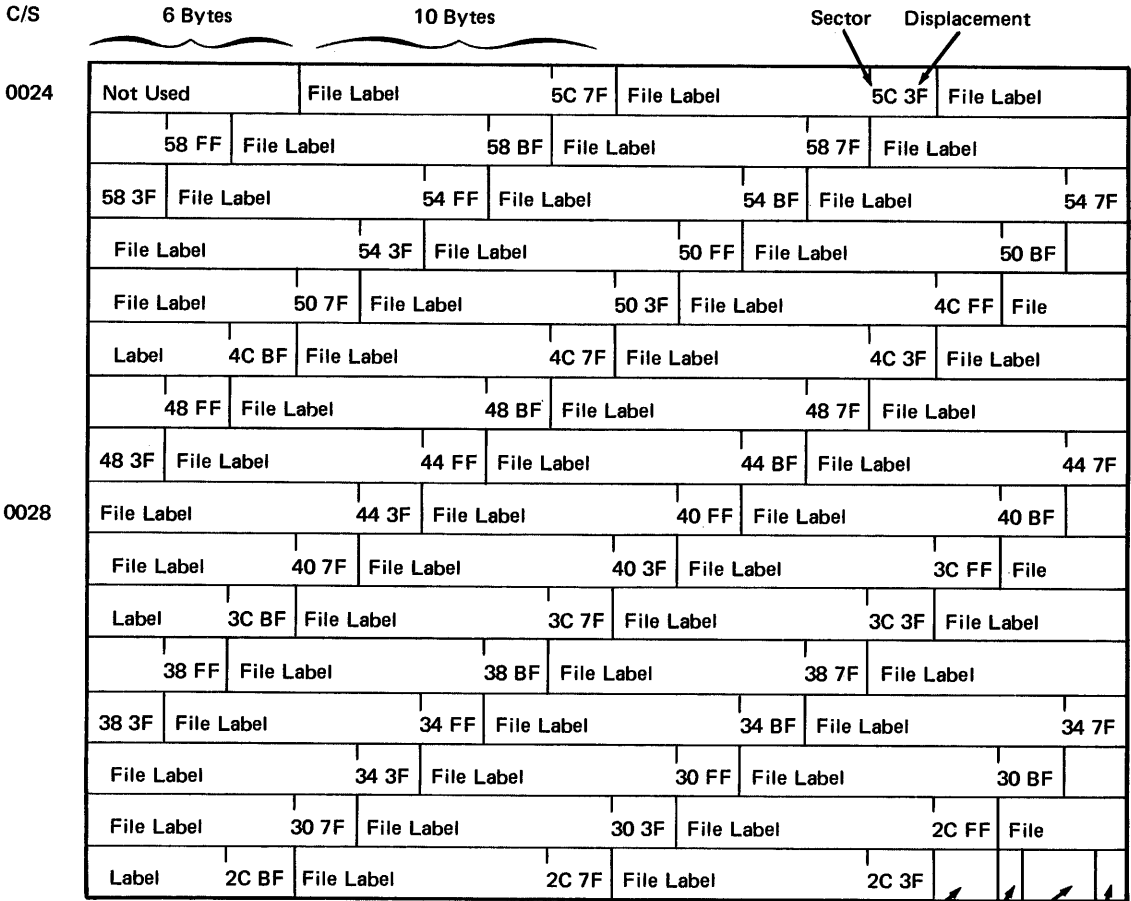
Sector	Name	Displacement (Hex)	Bytes	Contents
3-15	F1UPSI	3E	1	Save area for Open routine
	F1BYTE	3F	1	Save area for Open routine
<i>Note: Format sevens are loaded beginning at the end of sector 15.</i>				
	F7TAG	0	1	Format seven tag number
	F7CHAN	1	2	Sector/displacement of corresponding format one
	F7KEYL	3	1	Key length minus one
	F7INDX	4	1	Format one index number
	F7SEQN	5	1	Format one sequence number
	F7LOKE	6	29	Low key
	F7HIKE	35	29	High key
16-28				Initiator table
29-32				Work area for miscellaneous usage
33-47				Utility control statement storage area

Figure A-7. Scheduler Work Area Format (Part 6 of 6)

Volume Table of Contents—VTOC

A volume table of contents (VTOC) resides on every disk at location cylinder 0 / sectors X'24'-X'27'. This table is composed of *file labels* and an *index*. The index contains an

entry for each file label, and the file labels are used to describe each file maintained on the disk pack. Figure A-8 describes the format of the VTOC index. Figure A-9 describes the format of the VTOC file labels.



Note: The first byte of the file label is set to one of the following hexadecimal values:

- X'00' – F1 entry is not used.. (No file name is present within the tag.)*
- X'20' – F1 entry is used. The file name is contained within a name list chain or the system scratch list (S list). (No file name is present within the tag.)*
- X'40' or greater—F1 entry is used. The actual file name is contained within the tag.

*When a file name is not present, bytes 2-8 of the file label are zeros.

S/D address of the first member on the S list (two bytes)

Number of Format 1's on the S list (one byte)

S/D address of the last member on the S list (two bytes)

Number of free tags in the VTOC index (Note: Only P and T formats are considered not free) (one byte)

Figure A-8. Volume Table of Contents—Index Format (Part 1 of 2)

C/S	Tag ID	64-Byte Records	S/D		C/S	Tag ID	64-Byte Records	S/D	
002C	32		2C 3F	} Format 1's	0044	1A	44 3F	} Format 1's	
	31		2C 7F			19	44 7F		
	30		2C BF			18	44 BF		
	2F		2C FF			17	44 FF		
0030	2E		30 3F		0048	16	48 3F		
	2D		30 7F			15	48 7F		
	2C		30 BF			14	48 BF		
	2B		30 FF			13	48 FF		
0034	2A		34 3F		004C	12	4C 3F		
	29		34 7F			11	4C 7F		
	28		34 BF			10	4C BF		
	27		34 FF			F	4C FF		
0038	26		38 3F		0050	E	50 3F		
	25		38 7F			D	50 7F		
	24		38 BF			C	50 BF		
	23		38 FF			B	50 FF		
003C	22		3C 3F	0054	A	54 3F			
	21		3C 7F		9	54 7F			
	20		3C BF		8	54 BF			
	1F		3C FF		7	54 FF			
0040	1E		40 3F	0058	6	58 3F			
	1D		40 7F		5	58 7F			
	1C		40 BF		4	58 BF			
	1B		40 FF		3	58 FF			
				005C	2	5C 3F			
			1		5C 7F				
			First Format Seven		5C 80				
			Second Format Seven		5C C0				
								} Format 7's	

Note: See Figure A-9, Volume Table of Contents—File Label Format for a description of the 64-byte records.

Figure A-8. Volume Table of Contents—Index Format (Part 2 of 2)

Hexadecimal Displacement	Number of Bytes	Name	Contents
0	1	Tag ID	Tag identification
1	2	Chain address	Sector number/displacement into sector
3	8	File name	File name
B	6	Date	Date of file
11	1	Retain	Retain indicator for file
12	2	File type	Data Management File type*
14	2	Record length	The number of bytes within this record
16	1	Key length	The number of bytes within the record key
17	2	Key location	Displacement of the key within the record
19	3	Last record	Address (C/S/D) of the last record
1C	3	Last key	Address (C/S/D) of the last key
1F	2	Data start	Address (C/S) of the start of data
21	2	Data end	Address (C/S) of the end of the data
23	2	Index start	Address (C/S) of the start of the index
25	2	Index end	Address (C/S) of the end of the index
27	1	Rec/trks	X'80' On X'0' - # of tracks Off X'1' - # of records
28	2	Rec/trks #	Number of records/tracks created (See previous byte)
2A	1	Vol seq #	Volume sequence number
2B	2	Back ptr	Back S pointer
2D-3F	19	Not used	

*See F1 TYPE, Figure A-7.

Figure A-9. Volume Table of Contents—File Label Format

System Communication Region

This data area can be located in the *Dedicated Supervisor Data Area Formats*.

Program Level Communication Region

This data area can be located in *Dedicated Supervisor Data Area Formats*.

Reader/Interpreter Work Area (RDIWA)

This is a 108-byte data area that is loaded just before the Reader/Interpreter Mainline (\$\$RDML). The Reader/Interpreter uses this data area as a communication region for passing information between the Reader/Interpreter Mainline and the control statement processors. Figure A-10 contains the format for this work area.

Label	Hex Displacement	Contents
LDIWA	00	Beginning of the Reader/Interpreter work area
M1	01	A 2-byte constant of -1
P1	03	A 2-byte constant of 1
P3	05	A 2-byte constant of 3
FOUR	07	A 2-byte constant of 4
P10	09	A 2-byte constant of 10
INFO	0A	Logic switches
LOADRQ		X'80' - Indicates // LOAD required in procedure
CRIC		X'40' - Indicates invalid comment message
FNDQVR		X'20' - Indicates override being sought
*		X'10' - Not used
*		X'08' - Not used
PRNTEQ		X'04' - Print status of OCL statement in main storage
RETREQ		X'02' - Indicates do not return to caller from \$\$\$THB
SABORT		X'01' - Controlled cancel temporary indicator
CINFO	0B	Logic switches for the CCP routines
SOURCE		X'80' - Source parameter not specified for the COMPILE record
UNIT		X'40' - Unit not specified for the COMPILE record
SUBL		X'20' - Sublist being processed for the COMPILE record
CPROCS		X'10' - Current processing of COMPILE record ON - Object parameter being processed OFF - Unit parameter being processed
SPLTSL		X'08' - Split sublist on continue
*		X'04' - Not used
LAST		X'02' - Indicator for the last LOAD parameter
ASTISK		X'01' - // LOAD * specified
CBTABL	0C	Control byte table (DS CL10)
CKINFO	16	Logic switches for the \$\$\$RDML scan logic
SIGST		X'80' - Non-blank after // COMMENT record
BLKFND		X'40' - Blank found
COMENT		X'20' - Indicates comment found
CRCONT		X'10' - Continuation
ENDSCN		X'08' - Indicates end of scan
VRBFND		X'04' - Verb found on statement being scanned
NAMSCN		X'02' - Indicates searching for // FILE name
NOFIND		X'01' - Keyword or verb not valid
CKINF1	17	Logic switches for the \$\$\$RDML scan logic
KEYWRD		X'80' - Keyword found
*		X'40' - Not used
EQU		X'20' - Parameters matched
CKDASH		X'10' - Dash (-) found
CKCOMA		X'08' - Comma (,) found
SUBLST		X'04' - Sublist being encoded
CKQUOT		X'02' - Quote (') found
SUBLND		X'01' - End of sublist found
FINFO	18	Switches for // FILE processing
*		X'80' - Not used
*		X'40' - Not used
RECSP		X'20' - Records specified
TKSSP		X'10' - Tracks specified
PACKSP		X'08' - Pack specified
UNITSP		X'04' - Unit specified
NAMESP		X'02' - Name specified
LOCSP		X'01' - Location specified
INFO1	19	Logic Switches
NOPARM		X'80' - Null parameter being processed
WRTZRO		X'40' - Encoded // FILE statement loaded in format one area
NOPMER		X'20' - No parameters found on the statement
*		X'10' - Not used
*		X'08' - Not used
*		X'04' - Not used
*		X'02' - Not used
*		X'01' - Not used

Figure A-10. Reader/Interpreter Work Area (RDIWA) Format (Part 1 of 3)

Label	Hex Displacement	Contents
INFO2	1A	Logic switches for // FILE processing
FRSTPT		X'40' – First time put
UNITS	1B	Units specified on the // FILE statement
F1		X'80' – F1 specified in the unit parameter
*		X'40' – R1 specified in the unit parameter
*		X'20' – R2 specified in the unit parameter
F2		X'10' – F2 specified in the unit parameter
INFOID	1C	Input device indicator
CONSOL		X'00' – Indicates that the console is input device
INFOP1	1D	NPSCH1 (Taken from the program level communication region)
DATREC		X'80' – // DATE received (INTRA mode)
CPLREC		X'40' – // COMPILE received
SWREC		X'20' – // SWITCH received
PROCMG		X'10' – Procedure merge indicator
OVERRIDE		X'08' – Override statement in process
INTRA		X'04' – Mode indicator
INTER		X'02' – Mode indicator
IPL		X'01' – Mode indicator
INFOP2	1E	NPSCH2 (Taken from the program level communication region)
CONTNU		X'80' – Continuation
UTILCC		X'40' – Utility control cards in the scheduler work area
SLASHA		X'20' – A slash was read from Sysin device
FILREC		X'10' – // FILE statement received
EOJHLT		X'08' – End of job halt indicator
FLUSH		X'04' – Flush the Sysin device
HABORT		X'02' – Immediate cancel
SABRT		X'01' – Controlled cancel
INFOP3	1F	NPSCH3 (Taken from the program level communication region)
*		X'80' – Not used
*		X'40' – Not used
*		X'20' – Not used
LOG		X'10' – Logging specification
		ON – Do not log error
		OFF – Log error
*		X'08' – Not used
ADSTMT		X'04' – Add statement processing indicator
*		X'02' – Not used
PGMLVL		X'01' – Program level indicator
		ON – Level 2
		OFF – Level 1
PARMS3	20	Parameter list for \$\$RDS3 (function byte)
@BEGIN	22	Beginning address for the scan
@END	24	End address of the scan
@ENCOD	26	Address of the encoded parameters
@HS1	28	Address of the Halt/Syslog print parameter list
@BGREN	2A	Start address for encoding the statement
PARMHS	31	Halt/Syslog parameter list
@PLCR	33	Address of the program level communication region
@CBSAV	35	Address of the control byte being processed
NAMLEN	36	Length-1 of file name
FLNAME	3E	File name
#UNITS	3F	Number of unit parameters on the // FILE statement
#PACKS	40	Number of disk packs requested for this job
NUMPKS	41	Number of disk packs for the // FILE statement
ENDCNT	42	Counters used by the // DATE Control Card Processor (\$\$RDDT)
BTCNT	43	Counter used by the // DATE Control Card Processor (\$\$RDDT)
#LOC	44	Number of track parameters specified
#TKS	45	Number of track parameters specified
#REC	46	Number of record parameters specified

Figure A-10. Reader/Interpreter Work Area (RDIWA) Format (Part 2 of 3)

Label	Hex Displacement	Contents
RTRNAD	48	\$\$RDMK return address to \$\$RDML
ADPRC1	4A	Address of the procedure get parameter list
ER#SAV	4B	Temporary save area for the error number
ERROR#	4C	Error number to be logged
RTNID	4E	Two-byte ID for identifying the module calling \$\$STHB
VRBID	4F	Keyword scan information
OCL1	51	Prefix for OCL statements (//)
CKSEND	53	End of scan
CKSTRT	55	Start of scan
CS#SG	57	C/S/# of Source Library—Get (\$\$SYSG)
BUFEND	59	Address of blank past buffer one
ENCODE	5B	Address of present encoding
ERRRTN	5D	Address of the linkage to the error routine
SYSIN1	5E	Sysin parameter list (flag byte)
@BUF1	60	Address of the Sysin buffer
	62	Address of the Sysin buffer
@WORK	64	Address of the work area for Sysin
ADSTO1	65	Scheduler Work Area—Get (\$\$SSGT) parameter list
	65	Utility cards/first time/this level
	66	Length of get (96)
	67	Completion code
	69	Address of the I/O area
	6B	Transient save area
LOGERR		Go to ML1000 without changing the ARR
REINIT	80	Indicator to read after logging error
ARR	08	Address return register
IAR	10	Instruction address register

Figure A-10. Reader/Interpreter Work Area (RDIWA) Format (Part 3 of 3)

Initial Halt/Syslog Parameter Table

The initial Halt/Syslog parameter table is loaded with \$\$STHB and is located at the end of \$\$STHB. The table contains the following information:

- Name of the module currently processing the control statement.
- Routine cylinder/sector address of the correct table (\$\$STH0, \$\$STH1, \$\$STH2, \$\$STH3, or \$\$STH4) to be loaded.
- A value to modify the error number.
- Valid modes and a wrong mode error number.

Figure A-11 shows the format of the initial Halt/Syslog Parameter Table.

Table one entry format:						
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Each entry listed in table one contains the following 7-byte format:						
Bytes one and two: Name of module						
Bytes three and four: Routine cylinder/sector address						
Byte five: Value to modify error number						
Bytes six and seven: Valid modes and mode error number						
X'04' — INTRA mode						
X'02' — INTER mode						
X'01' — IPL mode						

Figure A-11. Initial Halt/Syslog Parameter Table



<p>Table Two Contains:</p> <ul style="list-style-type: none"> ● Modified error number ● Valid mode ● Halt, print, halt or print, or halt and print ● Length of Halt/Syslog parameter ● Options ● Displacements into tables three and four
<p>Table Three Contains:</p> <ul style="list-style-type: none"> ● Valid error displays for the console display lights and printer
<p>Table Four Contains:</p> <ul style="list-style-type: none"> ● Valid subidentifiers for the system printer

Table Two Entry Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
--------	--------	--------	--------	--------

Each entry listed in table two has the following 5-byte format:

- Byte one: Modified error number
- Byte two: Mode
 - X'04' – INTRA mode
 - X'02' – INTER mode
 - X'01' – IPL mode
- Byte three: Halt and/or print, and options
 - X'80' – Print
 - X'40' – Halt
 - X'20' – Halt or print
 - X'10' – Length of parameter list
 - On – 7 bytes
 - Off – 5 bytes
 - X'08' – Proceed
 - X'04' – Retry
 - X'02' – Controlled Cancel
 - X'01' – Immediate Cancel
- Byte four: Hexadecimal displacement into table three.
- Byte five: Hexadecimal displacement into table four if appropriate

Table Three Format:

The following table contains valid display lights and their associated meanings (display converted for Model 6):

Halt Light	Meaning
AH	IMAGE statement; syntax error
AJ	IMAGE statement; syntax error
A7	IMAGE statement; invalid parameter
A8	IMAGE statement; error in member
A9	IMAGE statement; error in member
AA	IMAGE statement; error in member
AC	IMAGE statement; invalid hex character
AE	IMAGE statement; invalid hex character
AF	IMAGE statement; invalid hex character
9A	IMAGE statement; previous OCL errors, statement ignored
9U	IMAGE statement; other level read an IMAGE statement
7'	I/O error when reading from the source library
7Y	I/O error when reading from the source library
IC	Invalid comment
75	OCL statement error
76	OCL statement error
78	OCL statement error
PE	IMAGE statement; chain check
PF	IMAGE statement; chain check

Table Four Format:

The following are valid subidentifiers:

01	0A
02	0B
03	0C
04	0D
05	0E
06	0F
07	0G
08	0H
09	

Figure A-12. Halt/Syslog Parameter Table One (\$\$STHO) Format

<p>Table Two Contains:</p> <ul style="list-style-type: none"> ● Modified error number ● Valid mode ● Halt, print, halt or print, or halt and print ● Length of Halt/Syslog parameter ● Options ● Displacements into tables three and four
<p>Table Three Contains:</p> <ul style="list-style-type: none"> ● Valid error displays for the console display lights and printer
<p>Table Four Contains:</p> <ul style="list-style-type: none"> ● Valid subidentifiers for the system printer

Table Three Format:

The following table contains the valid display lights and their associated meanings (display converted for Model 6):

<i>Halt Light</i>	<i>Meaning</i>
AL	PARTITION statement; parameter error
A	PARTITION statement; parameter error
AU	PARTITION statement; parameter error
70	PARTITION statement; system does not have DPF
73	PARTITION statement; a partition cannot be accepted
78	OCL statement error, invalid statement identifier
80	DATE statement; no system date provided
81	LOAD statement; parameter error
83	LOAD statement; LOAD * invalid for program level 2
7A	Second LOAD or CALL statement found after a LOAD
84	CALL statement; parameter error
88	CALL statement; procedure not found on specified unit
8H	CALL statement; too many levels of procedures called
97	LOG statement; parameter error in IPL mode
98	LOG statement; invalid parameter in INTRA mode
99	LOG statement; invalid parameter in INTER mode
9A	LOG statement; log device not changed due to previous OCL errors
9E	LOG statement; device not available in IPL mode
9F	LOG statement; device not available in INTER mode
9H	LOG statement; device not available in INTRA mode
9'	LOG statement; device not supported
9-	LOG statement; device not supported
9Y	LOG statement; device not supported
8Y	LOG statement; device turned off by other level
8-	LOG statement; device turned off by other level
7H	SWITCH statement; found between jobs
85	SWITCH statement; second statement found
86	SWITCH statement; invalid parameter

Table Two Entry Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
--------	--------	--------	--------	--------

Each entry listed in table two has the following 5-byte format:

- Byte one: Modified error number
- Byte two: Mode
 - X'04' – INTRA mode
 - X'02' – INTER mode
 - X'01' – IPL mode
- Byte three: Halt and/or print, and options
 - X'80' – Print
 - X'40' – Halt
 - X'20' – Halt or print
 - X'10' – Length of parameter list
 - On – 7 bytes
 - Off – 5 bytes
 - X'08' – Proceed
 - X'04' – Retry
 - X'02' – Controlled Cancel
 - X'01' – Immediate Cancel
- Byte four: Hexadecimal displacement into table three
- Byte five: Hexadecimal displacement into table four if appropriate

Table Four Format:

The following are valid subidentifiers:

01	0U	U1
02	2P	U2
03	IP	P2
04	N1	0D
05	N2	LR
0P	N3	0V
0N		

Figure A-13. Halt/Syslog Parameter Table Two (\$\$STH1) Format

HALT/SYSLOG Parameter Table Three (\$\$STH2)

This is a 512-byte area that is loaded into the transient area (X'100') by the Halt/Syslog Parameter Build routine (\$\$STHB). This data area is composed of three tables that

<p>Table Two Contains:</p> <ul style="list-style-type: none"> ● Modified error number ● Valid mode ● Halt, print, halt or print, or halt and print ● Length of Halt/Syslog parameter ● Options ● Displacements into tables three and four
<p>Table Three Contains:</p> <ul style="list-style-type: none"> ● Valid error displays for the console display lights and printer
<p>Table Four Contains:</p> <ul style="list-style-type: none"> ● Valid subidentifiers for the system printer

Table Two Entry Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
--------	--------	--------	--------	--------

Each entry listed in table two has the following 5-byte format:

Byte one: Modified error number

Byte two: Mode X'04' – INTRA mode
 X'02' – INTER mode
 X'01' – IPL mode

Byte three: Halt and/or print, and options

 X'80' – Print
 X'40' – Halt
 X'20' – Halt or print
 X'10' – Length of parameter list
 On – 7 bytes
 Off – 5 bytes
 X'08' – Proceed
 X'04' – Retry
 X'02' – Controlled Cancel
 X'01' – Immediate Cancel

Byte four: Hexadecimal displacement into table three

Byte five: Hexadecimal displacement into table four if appropriate

are used by \$\$STHB to build a parameter list for errors that are encountered in the COMPILE, FILE, and READER control statements. Figure A-14 contains the format for this data area. See *Appendix B. Diagnostic Aids* for CR96 halts.

Table Three Format:

The following table contains the valid display lights and their associated meanings (display converted to Model 6):

Halt Light	Meaning
92	COMPILE statement; second COMPILE statement found
93	COMPILE statement; syntax error
94	COMPILE statement; keyword error
95	COMPILE statement; parameter error
7C	COMPILE statement; COMPILE statement found between jobs
7F	FILE statement; statement found between jobs
A0	FILE statement; syntax error
A1	FILE statement; keyword error
A2	FILE statement; parameter error
A3	FILE statement; parameter missing
A4	FILE statement; both TRACKS and RECORDS specified
A5	FILE statement; UNITS and PACKS not in 1-1 correspondence
A6	FILE statement; multi-volume error
A-	FILE statement; more than 52 packs specified
A'	FILE statement; no space left in scheduler work area
8J	READER statement; invalid parameter

Table Four Format:

The following are valid subidentifiers:

01	08	1R	0P
02	09	1T	AE
03	0A	DK	AP
04	0B	IK	LO
05	0C	NP	KL
06	1L	NU	HK
07	1H	NN	SQ

Figure A-14. Halt/Syslog Parameter Table Three (\$\$STH2) Format

HALT/SYSLOG Parameter Table Four (\$\$STH3)

This is a 512-byte area that is loaded into the transient area (X'100') by the Halt/Syslog Parameter Build routine (\$\$STHB). This data area is composed of three tables that

are used by \$\$STHB to build a parameter list for errors that are encountered in the FORMS and DATE control statements. Figure A-15 contains the format for this data area. See Appendix B. *Diagnostic Aids* for CR96 halts.

<p>Table Two Contains:</p> <ul style="list-style-type: none"> ● Modified error number ● Valid mode ● Halt, print, halt or print, or halt and print ● Length of Halt/Syslog parameter ● Options ● Displacements into tables three and four
<p>Table Three Contains:</p> <ul style="list-style-type: none"> ● Valid error displays for the console display lights and printer
<p>Table Four Contains:</p> <ul style="list-style-type: none"> ● Valid subidentifiers for the system printer

Table Three Format:

The following table contains the valid halt lights and their associated meanings (display converted to Model 6).

<i>Halt Light</i>	<i>Meaning</i>
9A	Action on last OCL statement ignored due to previous OCL errors found for this job
9	FORMS statement; other level read a FORMS statement
9J	FORMS statement; keyword or parameter error
9L	FORMS statement; keyword or parameter error
9P	FORMS statement; keyword or parameter error
7E	DATE statement; found between jobs
8A	DATE statement; invalid statement
8C	DATE statement; second DATE statement found
8E	DATE statement; parameter missing
89	DATE statement; system date already read

Table Two Entry Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
--------	--------	--------	--------	--------

Each entry listed in table two has the following 5-byte format:

Byte one: Modified error number

Byte two: Mode X'04' — INTRA mode
 X'02' — INTER mode
 X'01' — IPL mode

Byte three: Halt and/or print, and options

 X'80' — Print
 X'40' — Halt
 X'20' — Halt or print
 X'10' — Length of parameter list
 On — 7 bytes
 Off — 5 bytes
 X'08' — Proceed
 X'04' — Retry
 X'02' — Controlled Cancel
 X'01' — Immediate Cancel

Byte four: Hexadecimal displacement into table three

Byte five: Hexadecimal displacement into table four if appropriate

Table Four Format:

The following are valid subidentifiers:

01	07
02	08
03	09
04	0A
05	0B
06	0C

Figure A-15. Halt/Syslog Parameter Table Four (\$\$STH3) Format



HALT/SYSLOG Parameter Table Five (\$\$STH4)

This is a 512-byte area that is loaded into the transient area (X'100') by the Halt/Syslog Parameter Build routine (\$\$STHB). This data area is composed of three tables that are used by \$\$STHB to build a parameter list for errors that are encountered in the FILE, RUN, DATE, READER, PAUSE, HALT, NOHALT control statements. Figure A-16 contains the format for this data area. See Appendix B. Diagnostic Aids for CR96 halts.

<p>Table Two Contains:</p> <ul style="list-style-type: none"> ● Modified error number ● Valid mode ● Halt, print, halt or print, or halt and print ● Length of Halt/Syslog parameter ● Options ● Displacements into tables three and four
<p>Table Three Contains:</p> <ul style="list-style-type: none"> ● Valid error displays for the console display lights and printer
<p>Table Four Contains:</p> <ul style="list-style-type: none"> ● Valid subidentifiers for the system printer

Table Two Entry Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
--------	--------	--------	--------	--------

Each entry listed in table two has the following 5-byte format:

- Byte one: Modified error number
- Byte two: Mode X'04' — INTRA mode
 X'02' — INTER mode
 X'01' — IPL mode
- Byte three: Halt and/or print, and options
 X'80' — Print
 X'40' — Halt
 X'20' — Halt or print
 X'10' — Length of parameter list
 On — 7 bytes
 Off — 5 bytes
 X'08' — Proceed
 X'04' — Retry
 X'02' — Controlled Cancel
 X'01' — Immediate Cancel
- Byte four: Hexadecimal displacement into table three
- Byte five: Hexadecimal displacement into table four if appropriate

Table Three Format:

The following table contains valid display lights and their associated meanings (display converted for Model 6):

Halt Light	Meaning
71	OCL statement error
72	
74	
75	
76	
77	
78	
79	Procedure not found on specified unit
88	
A'	FILE statement; there are too many file statements
A-	FILE statement; more than 52 packs specified
7U	RUN statement; no RUN statement found for job
'D	RUN statement error
/	RUN statement error
AP	RUN statement error
MN	RUN statement error
7E	DATE statement; found between jobs
7J	READER statement; found between LOAD or CALL and RUN statements
MB	HALT/NOHALT statement error
89	DATE statement; DATE statement already specified
8A	DATE statement; parameter error
8C	DATE statement; second statement found
8E	DATE statement; parameter missing or invalid
90	PAUSE statement
91	PAUSE statement
8J	READER statement; invalid parameter
8L	READER statement; input device not available
8P	READER statement; input device not part of system
MD	HALT/NOHALT statement error
7A	Second LOAD or CALL statement found for job
80	No system date provided
IC	Invalid comment
NH	NOHALT statement error

Table Four Format:

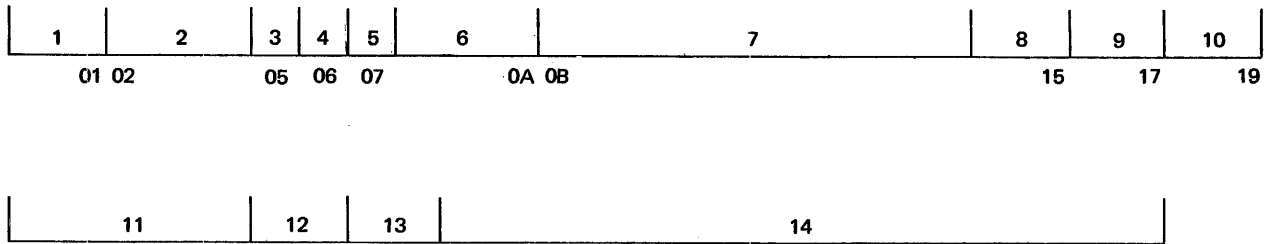
The following are valid subidentifiers:

- ID
- ND
- 01
- 02

Figure A-16. Halt/Syslog Parameter Table Five (\$\$STH4) Format

Call Common Area

The Call Common Area is the first 50 bytes of Call Cycle (\$\$RBCL).



Field	Disp	Name	# Bytes	Contents
1	01	BUFPTR	2	Pointer to Scheduler Common Area
2	02	PARLIB	3	Parameter List Passed to Librarian Interface
3	05	SWTCH1	1	Switch 1*
4	06	SWTCH2	1	Switch 2*
5	07	SWTCH3	1	Switch 3*
6	0A	SEQ#	3	Current Sequence Number
7	0B	PARMSX	9	Parameter List Passed to Syntax Checking Rtms
8	15	CS#ERR	2	Cylinder/Sector Address of \$\$SLSE
9	17	CS#SYG	2	Cylinder/Sector Address of Source Get Rtn
10	19	CS#SXI	2	Cylinder/Sector Address of Syntax Interface
11	1A	FLAG	3	Parameter List for Sysin Routine
12	1C	SYSWK	2	Work Area for Sysin
13	1E	COUNT	2	Counter
14		Unassigned	15	Unassigned

***Note:**

<i>Switch 1</i>	X'80' – Verb Switch X'40' – Need a Record X'20' – MVF Continuation X'10' – Delayed Resp Bit	X'08' – No-go Bit X'04' – File Type Stmt X'02' – Compile Type Stmt X'01' – Forms Type Stmt
<i>Switch 2</i>	X'80' – End of Response X'40' – Hikey Continuation X'20' – Unassigned X'10' – Unassigned	X'08' – Unassigned X'04' – Unassigned X'02' – Unassigned X'01' – Unassigned
<i>Switch 3</i>	Expansion	



Scheduler Control Table

The Scheduler Control Table is X'13F' bytes long and ends at the beginning address of Scheduler Control (\$\$RBCO).

				ATTRIBUTES Y					ATTRIBUTES Z												
		Position Code (P-Code)	Control Code (C-Code)	Last Parameter of Call	Last Parameter of Build	Last Parameter of Load	Last Parameter of Compile	Last Parameter of this OCL	A Required Response	An Allowable Delayed Response	Unassigned	A File Parameter	A File Name Parameter	A Records Parameter	A Tracks Parameter	MVF Parameters are Allowed	Unassigned	User Permitted to Term OCL Prompt (PIS-)	Unassigned		
				80	40	20	10	08	04	02	01	80	40	20	10	08	04	02	01		
CALL	NAME	80	80						1												
	UNIT	89	40	1				1	1												
BUILD	NAME	91	80						1												
	UNIT	9A	44		1*			1	1												
LOAD	NAME	A2	80						1											1***	
	UNIT	A8	40			1		1	1	1										1****	
COMPILE	OBJECT	BB	40							1											
	SOURCE	BC	80						1	1											
	UNIT	C4	40				1*	1	1	1											
FILE	NAME	CC	A0							1		1	1							1***	
	UNIT	D5	40						1	1		1				1					
	PACK	DD	90						1	1		1				1				1	
	LABEL	E5	A0							1		1									1
	RECORDS	ED	30							1		1		1		1					1
	TRACKS	F5	20							1		1			1	1					1
	LOCATION	FD	60							1		1				1					1
	RETAIN	85	10							1		1									1
DATE	8D	B0							1		1									1	
FORMS	DEVICE	95	20																		
	LINES	9E	10							1											

- *Set to 0 during scheduler execution
- **Program start (-) allowed to signify a compiler type program
- ***Program start (-) allowed to signify Hi-Key is to be prompted
- ****Program start (-) allowed to bypass prompting of DATE/SW/FL name

Resident loader code is included in the control table for \$\$RBCO. This code is used to interface between the supervisor and all scheduler phases for loading, and for passing control to phases used in the conversational mode. The loader code occupies approximately 150 bytes, and includes four bytes for each phase to be loaded or to receive control.

The format of each 4-byte entry is:



C/S/#—cylinder/sector and the number of object sectors
RLD—the RLD displacement

The 4-byte entries are initialized when \$\$RBCO is initialized.

When a scheduler phase uses the resident loader code, all phases are loaded via the supervisor without a find. The scheduler phase passes a 3-byte parameter list (addressed by XR2) to the resident loader code. The scheduler phase calls the loader code by executing the instruction B LDIAR, (XR1), where XR1 contains an address in the scheduler common area. At this address in the scheduler common area is the instruction L IAR, which causes control to be passed to the resident loader code.

The format of the 3-byte parameter list passed to the resident loader code is:



The following list gives the load code for each specific module required by conversational mode.

Load Code	Module Name
X'04'	\$\$RBBC
X'08'	\$\$RBM1
X'0C'	\$\$RBM2
X'10'	\$\$RBDS
X'14'	\$\$RBDT
X'18'	\$\$RBSW
X'1C'	\$\$RBHI
X'20'	\$\$RBFM
X'24'	\$\$RBLG
X'28'	\$\$RBRN
X'2C'	\$\$RBCC
X'30'	\$\$RBCL
X'34'	\$\$RBCD
X'38'	\$\$RBCK
X'3C'	\$\$RBCP
X'40'	\$\$RBM Y
X'44'	\$\$RBSX
X'48'	\$\$RBIN
X'4D'	\$\$RBLP
X'51'	\$\$SYSG

The 0 bit of the load code indicates whether or not the phase is to receive control or is just to be loaded. If X'80' is on, the phase will receive control; if it is off, the phase will just be loaded. The 7 bit of the load code indicates whether or not the phase to be loaded is a transient. If X'01' is on, the phase is a transient and X'0100' is assigned as the link edit address; if X'01' is off, the phase is not a transient, and a link edit address of X'0000' is assumed.



The following data areas are used by more than one Linkage-Editor routine or phase and can be located on the associated listing for each routine.

PASS1 Communication Region (COMMON)

This region is used to store and pass information between the PASS1 overlay phases: \$LINKB, \$LINKC, \$LINKD, \$LINKE, \$LINKF. The format of this area is shown in Figure A-17.

Name	Hexadecimal Displacement	Number of Bytes	Contents
OUTPTQ	00	2	Q-code for Linkage Editor output
ERRADD	02	2	Current error table address
WORK1	04	2	Work area 1
WORK2	06	2	Work area 2
LTWBEG	08	2	\$WORK start cylinder/sector
LTWEND	0A	2	\$WORK end cylinder/sector
LTINCS	0C	2	Current \$WORK input C/S address
MODICS	0E	2	Current module input C/S address
MODOCS	10	2	Current module output C/S address
REGSV1	12	2	Register 1 content save area
REGSV2	14	2	Register 2 content save area
RLDCNT	16	2	Phase output RLD counter
RELFAC	18	2	Phase relocation factor
MODREL	1A	2	Module relocation factor
EOCORE	1C	2	End of core address
EOSADD	1E	2	End of supervisor address
LENGTH	20	2	Phase output text length
CURESL	22	2	Current ESL table address
ESLADD	24	2	External symbol list table address
ROOTND	26	2	Address of last ESL table—root
PHSTRT	28	2	Address of first ESL table—phase
INPADD	2A	2	Input buffer address save area
IOBADD	2C	2	Address of the IOB
ERRCOD	2D	1	Error number hold area
CNTER	2E	1	Field byte counter
CCUNT	2F	1	Input record count
SWITCH0	30	1	Indicator switch 0
			<i>Bit Usage</i>
			0 Print error table only—MAPTBL
			1 AUTOLINK in process
			2 OPTNS record has been processed
			3 END record has been processed
			4 End of mainline has been read
			5 ENTRY record has been processed
			6 Module records in process
			7 First PHASE record has been processed

Figure A-17. PASS1 Communication Region (Part 1 of 2)



Name	Hexadecimal Displacement	Number of Bytes	Contents
SWITCH1	31	1	Indicator switch 1 <i>Bit Usage</i> 0 Object library directory found 1 LEVEL processed 2 TEMP-PERM processed 3 RPG overlay program 4 ESL processed 5 PERM output 6 \$LINKD in core 7 SIZE processed
RECSAV	32	1	Input record count save
EXTRES	34	2	AUTOLINK last table address
DIRNAM	3A	6	Phase directory name
DIRTYP	3B	1	Module ESL table type
DIRINO	3C	1	Number of text sectors
DIRLAD	3E	2	Module core load address
DIRLDD	3F	1	RLD displacement
DIREPT	41	2	Execution start address
DIRSIZ	42	1	Core size requirement
DIRATR	44	2	Program attributes
DIRLEV	45	1	Module level number
PHASE#	47	2	Phase number
SECTOT	49	2	Overlay total sector number
CONCNE	4B	2	Constant — Value X'0001'
ENTRY0	4D	2	DEDAB0
ENTRY1	4F	2	DEDAB1
ENTRY2	51	2	DEDAB2
ENTRY3	53	2	DEDAB3
ENTRY4	55	2	DEDAB4
ENTRY5	57	2	DEDAB5
ENTRY6	59	2	DEDAB6
ENTRY7	5B	2	DEDAB7
ENTRY8	5D	2	DEDAB8
OLDRCS	5F	2	Address of directory
LTINQ	60	1	\$WORK read input Q-code
LTOTQ	61	1	\$WORK write output Q-code
MODIQ	62	1	Module read input Q-code
OFTADD	65	3	Overlay fetch table address
ERRTAB	66	1	Error counter
RECADD	68	2	Current records buffer address
OPUNCH	6A	1	Punch output indicator
OFTBEG	6C	2	Overlay fetch table start @
INAME	72	6	INCLD name save area
NOPRNT	73	1	No-print indicator

Figure A-17. PASS1 Communication Region (Part 2 of 2)

PASS2 Communication Region (COMMON)

This region is used to store and pass information between the PASS2 overlay phases: \$LINKG, \$LINKH, \$LINKJ. The format of this area is shown in Figure A-18.

Name	Hexadecimal Displacement	Number of Bytes	Contents
OUTPTQ	00	1	Q-code for TEXT and RLD writes
ERRADD	02	2	Current error table address
LSTXT@	04	2	Address of last text byte
TXTCNT	06	2	Text output counter
RLDCNT	08	2	RLD output counter
NULCNT	0B	3	RLD counter for FE null RLD
WSWICH	0C	1	Text process phase switch
			<i>Bit Usage</i> 0 On = Write a TEXT sector, Off = Write an RLD sector 1 Not used 2 Not used 3 Not used 4 First TEXT sector of a module has not been written 5 Last TEXT and RLD sectors are to be written 6 Last TEXT and RLD have been written 7 Not used
ENTRY1	0E	2	DEDAG1
ENTRY2	10	2	DEDAG2
ENTRY3	12	2	DEDAG3
ENTRY4	14	2	DEDAG4
TXTWRK	16	2	Address of TEXT work area (TXTWRK)
RLDWRK	18	2	Address of RLD work area (RLDWRK)
EOCORE	1C	4	End of core address
OPRAND	1E	2	Operand save area—\$LINKH
WORK	20	2	Work area
WORK1	22	2	Work area
ESLADD	24	2	ESL table start address
REGSV1	26	2	Register 1 save area
REGSV2	28	2	Register 2 save area
LTINCS	2A	2	\$WORK input C/S address
IOBADD	2C	2	Address of the IOB
ERRCOD	2D	1	Error code hold area
NIXICS	2F	2	Next text C/S address
SWITCH0	30	1	Indicator switch 0
			<i>Bit Usage</i> 0 Terminal Linkage Editor error has occurred 1 End of job indicator 2 Permanent output indicator 3 Not used 4 Not used 5 RPG Root Phase has been processed 6 Not used 7 On = \$LINKH in core; Off = \$LINKJ in core
SWITCH1	31	1	Indicator switch 1
			<i>Bit Usage</i> 0 Not used 1 Not used 2 Not used 3 RPG overlay program indicator 4 Volume label I/O indicator 5 Overlay fetch table I/O indicator 6 On = read I/O; Off = write I/O 7 Not used

Figure A-18. PASS2 Communication Region (Part 1 of 2)



Name	Hexadecimal Displacement	Number of Bytes	Contents
DIRNAM	38	7	Load module name
DIRSCS	3A	2	Load module start C/S
DIRTNO	3B	1	Load module number of text sectors
DIRLAD	3D	2	Load module load address
DIRLDD	3E	1	Load module RLD displacement
DIREPT	40	2	Load module start control address
DIRSIZ	41	1	Load module core size
DIRATR	43	2	Load module attributes
DIRLEV	44	1	Load module release level
DIRSNO	46	2	Load module total number of sectors
PHASE#	47	1	Total number of phases in output
SECTOT	49	2	Total number of sectors in output
CONONE	4B	2	Constant — value X'0001'
OTSTRT	4D	2	Text output start C/S address
RLDSRT	4F	2	RLD output start C/S address
OUTCNT	50	1	Output phase control counter
EXSTRT	52	2	Current EXTRN start address
LBSTRT	55	3	Library start C/S/D address
TXADD	57	2	Text buffer current address
LSTRLD	59	2	Address value of last RLD
BUFADD	5B	2	Buffer start address
NRLDCS	5D	2	C/S for next RLD output
RECADD	5F	2	Address of the record to be processed
LTINQ	60	1	\$WORK read Q code
RLDTOT	61	1	RLD output sector count
MODIQ	62	1	Q-code to read object library
OFTADD	65	3	C/S/D start address of overlay fetch table
ERRTAB	66	1	Error table counter
OVCSAD	68	2	Overlay C/S Hold Area—\$LINKJ
OVTEXT	69	1	Overlay # Text Sectors—\$LINKJ
OPUNCH	6A	1	Punch Output Indicator
OFTCSD	69	3	Overlay fetch table start address of save area
OPTBEG	6C		Overlay fetch table start @
CPACKQ	6D		Level 1 Q-code save area
CLIBCS	6F		Level 1 object lib. start C/S
NLIBCS	71		// Compile object lib. start C/S

Figure A-18. PASS2 Communication Region (Part 2 of 2)

Input/Output Buffer (BUFFER)

This is a 256-byte region that is used to contain input/output data used by the print and punch routines of the Linkage Editor.

Error Table (ERRTAB)

This is a 256-byte area that is used to record Linkage Editor errors and the phase in which they occurred. See

the *IBM System/3 Model 6 Halt Procedure Guide*, GC21-7541, for the error codes and their corresponding meanings.

Print Buffer (PRTBUF)

This is a 132-byte area that is used to contain data that is to be printed by the Error and Overlay Fetch Table Print phase (\$LINKK) or the ESL—Process phase (\$LINKM).

External Symbol List (ESL) Table (ESLTAB)

This table has a minimum length of 3072 bytes (3K) and contains PHASE and ESL records. See Figure A-19 for the format of this area.

Text Work Area (TXTWRK)

This is a 256-byte work area used to store the TEXT generated by the TEXT-RLD Conversion phase (\$LINKH). The contents of this data area are moved into the object library.

Relocation Directory (RLD) Work Area (RLDWRK)

This is a 256-byte work area used to store the RLD bytes generated by the TEXT-RLD Conversion phase (\$LINKH). The TEXT-RLD Conversion phase (\$LINKH) later moves the contents of this data area into the object library.

Language Translator Output Work Area (\$WORK)

This work file contains:

- Linkage-Editor control records
- ESL records
- TEXT-RLD records
- END records
- /* record

See *Part 7. Linkage Editor, Section 1. Introduction* for the format of this data area.

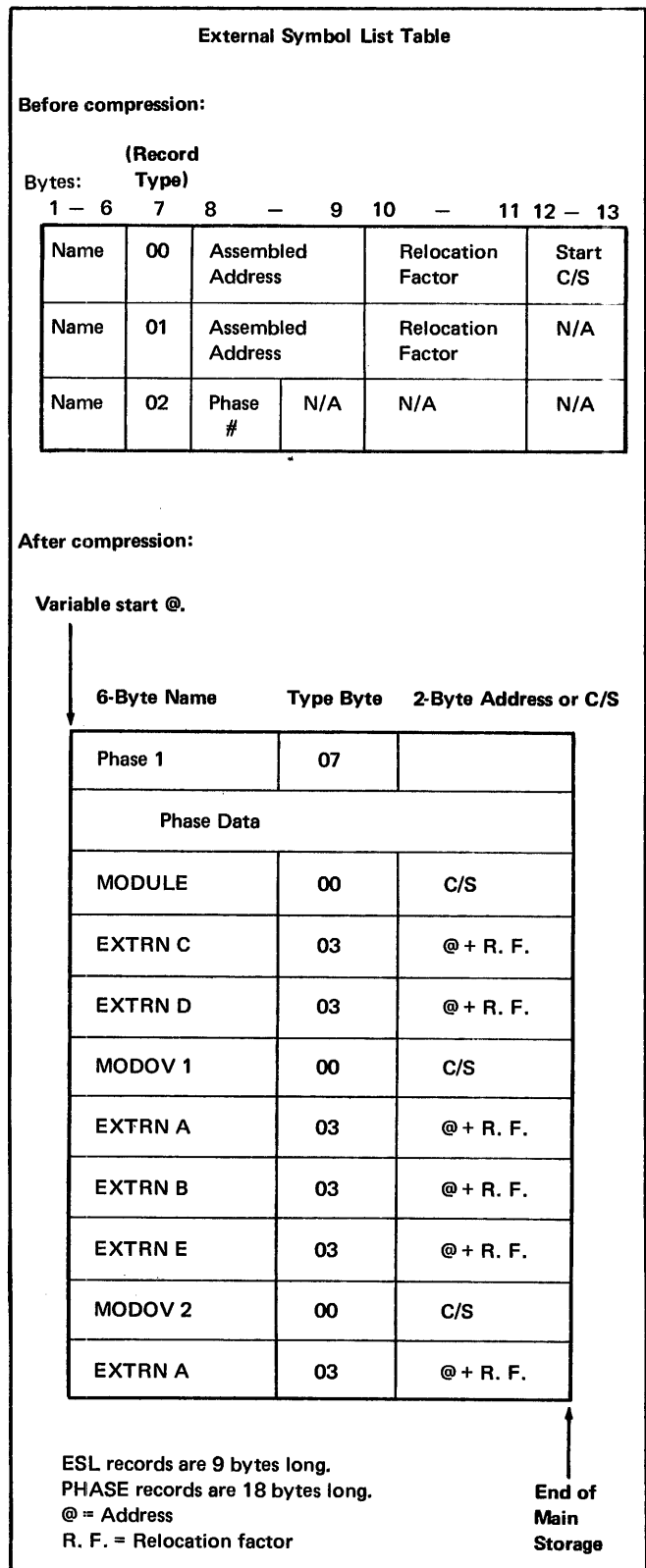


Figure A-19 ESL Table Format

Object Library Directory

This directory is located on the system disk pack and contains two types of entries: one entry for object (R.) modules and one entry for load (O.) modules. Each entry is 21 bytes in length (see Figure A-20).

Displacement Hex	Bytes	Contents	Description
0	1	Type	Entry type (O or R)
6	6	Name	Name of the object program or routine
8	2	Disk address	Cylinder/sector address of the library entry
9	1	Number of text sectors	Type O — number of text sectors in the library entry Type R — not used
B	2	Link edit address	Type O — storage address to which the library entry has — been link edited Type R — not used
C	1	Displacement of relocation directory	Type O — displacement (in bytes) of the relocation directory in the last sector of text Type R — not used
E	2	Starting control address	Type O — storage address at which program execution will begin Type R — not used
F	1	Storage size	Amount of storage (in sectors) required to run the program
11	2	Attributes	Byte 1 Bit 0 — 1 = Permanent entry 0 = Temporary entry Bit 1 — 1 = Inquiry Bit 2 — 1 = Inquiry evoking Bit 3 — 1 = Must run dedicated Bit 4 — 1 = Requires source information Bit 5 — 1 = Deferred mounting allowed Bit 6 — 1 = PTF applied Bit 7 — 1 = NOAUTO Byte 2 is reserved
12	1	Level	Version and modification level of this entry
14	2	Total number of sectors	Total number of sectors in the library entry

Figure A-20. Object Library Directory Entry Format

RPG Overlay Fetch Table

This fetch table is built by the Linkage Editor in a storage area supplied by the user's RPG program. The size of this table depends on the number of entries. See Figure A-21 for the entry format.

Relative C/S @	Number of Sectors TEXT	Core Load Address	RLD	Flag		
0	1	2	3	4	5	6
<i>Bytes</i>		<i>Contents</i>				
0-1	— Relative cylinder/start address of the load module. This is the number of cylinder/sectors past the C/S@ of the root load module of the overlay program as given in the object library directory entry for the program.					
2	— Number of sectors of core image text in the load module. (Does not include the number of related RLD sectors.)					
3-4	— Relative core start address of where the load module is to be placed in core by the system loader. (Relative to the end of Supervisor address.)					
5	— X'00' — RLD start displacement.					
6	— Flag byte — used at execution time by the root segments overlay call routine.					

Figure A-21. RPG Overlay Fetch Table Entry Format

The following data areas are used by more than one System Generation routine.

Pre-Processor Communication Region (RDPARM)

This is a 55-byte data area that is used to pass information between the Pre-Processor routines. This communication region is located in phase one of the Pre-Processor Mainline (\$SGROC). Figure A-22 contains the format for this communication region.



Name	Displacement Hex	Number of Bytes	Contents
RDPARM	0	0	
PA	0	2	Address of C/S of System Transient (\$\$SYSG)
PA1	2	2	Address of parameter list for reading the System Transient (\$\$SYSG)
BUFF	4	2	Address of pre-processor statement input buffer (BUFFR1)
HALT	6	2	Address of pre-processor halt parameter list
SNOT	8	2	Address of pre-processor error routine
PROTSV	A	2	Address of statement save area (SVPROT)
VSTST	C	2	Start address of variable symbol table
SYSEN	E	2	End address of SYNDX statements, start address of global statement
GLBEN	10	2	End address of global statement, start address of prototype statement
PROTMS	12	2	Address of prototype statement, start address of keyword parameters
TBLEN	14	2	Address of last entry built in table
VSTEN	16	2	Address of physical end of variable symbol table
VSTENT	18	2	Address of current VST table entry that the pre-processor is working with
VSTADR	1A	2	Address of where to put the variable symbol in the variable symbol table
COMON	1C	2	Address of the caller's communication area
MPRED	1E	2	Address of \$SGRED
MPGSY	20	2	Address of \$SGGSY
MPVST	22	2	Address of \$SGVST
SWADR	24	2	Address of switch area
SYSNDX	26	2	Address of SYSNDX in main storage
INBUFA	28	2	Address of work area—INBUF
OUTBUA	2A	2	Address of work area—OUTBUF
INGSY	2C	2	Address of 96-byte input work area containing input for the get symbol
INEND	2E	2	Address of last byte of the current work area that the pre-processor is working with
OUTGSY	30	2	Address pointer to the last character checked by \$SGGSY and the first character checked by \$SGCSB and \$SGNCB
SUBOBS	32	2	Address of current byte in work buffer that the pre-processor is using
SUBOBE	34	2	Address of the last byte of the work buffer that the pre-processor is using
CNOT	36	2	Address of the section in \$SGROC that handles the call to the user's error note processor (\$SGNOT)

Figure A-22. Pre-Processor Communication Region (RDPARM)

Pre-Processor Statement Input Buffer (BUFFR1)

This is a 96-byte data area that is initialized to blanks. This data area contains the pre-processor statements as they are read into main storage.

Statement Save Area (SVPROT)

This is a 96-byte data area that is initialized to blanks. This data area is used to store the prototype and model statements.

System Generation Communication Region (COMMON)

This is a 114-byte data area that is used to pass information between the System Generation routines and to the Pre-Processor routines. This data area resides in phase one of the System Generation Mainline (\$SGEN or \$SGENB). See Figure A-23 for the format of this data area.

Label	Hexadecimal Displacement	Number of Bytes	Contents
COMMON	0	96	Read buffer area
NOTESW	60	1	Note switch
GENTYP	61	1	Generation type (D)
HSPARM	62	2	Address of the Halt/Syslog parameter list
ADCOMN	64	2	Address of the COMMON area
ADNOTE	66	2	Address of the Error Note Processor
ADDDTF	68	2	Address of the disk DTF
ADGROC	6A	2	Address of the Pre-Processor
DTFIOA	6C	2	Disk DTF I/O area
NERMES	6E	2	Address of the note buffer containing error messages
ADDMGN	70	2	Address of Data Management entry point

Figure A-23. System Generation Communication Region (COMMON)



Pre-Processor Switch Area (SWADR)

This is a 12-byte data area that is used by the pre-processor to remember what processing is taking place. See Figure A-24 for the format of this area. This data area is located within the Pre-Processor Mainline (\$SGROC).

Input Work Area (INBUF)

This is a 62-byte data area that is initially filled with blanks. This data area is used as a subfield work area and is located in the Pre-Processor Mainline (\$SGROC).

Label	Hexadecimal Displacement	Number of Bytes	Contents
FTSW	0	1	First time switch (Mainline)
TEDSW	1	1	Table entry defined switch
GLOBSW	2	1	Global processing switch
PROTSW	3	1	Prototype processing switch
MACSW	4	1	System configuration statement processing switch
VSTBSW	5	1	Variable symbol table build switch
VSTFSW	6	1	Variable symbol table find/build switch
TPROSW	7	1	Type--T' & Processing Switch
FLDSW	8	1	Indicates fields 1, 2, or 3 processing
AIFSW	9	1	AIF processing switch
LASTSW	A	1	Constant value of zero
TDCNTR	B	1	Table definition statement counter

Figure A-24. Pre-Processor Switch Area

Output Work Area (OUTBUF)

This is a 62-byte data area that is initially filled with blanks. This data area is used as a subfield work area. OUTBUF is located within the Pre-Processor Mainline (\$SGROC).

Variable Symbol Table

This is a data area consisting of 0-512 bytes and is used to contain variable symbol entries. The format for a variable symbol entry is located in Figure A-25.

Source Library

The source library is an optional feature on any disk pack. It consists of any group of instructions performing a logical function (for example, a user's RPG program). The status of this library is contained in the system directory. This library, if existent, is a minimum of one cylinder on the system residence disk pack. If any additional source library space is required, the user must specify the required size during System Generation. Figure A-26 contains the format of the source library directory. This directory contains information about the source library entries (Figure A-27).

	A	B	C	D	E
<i>Chart ID</i>	<i>Number of Bytes</i>	<i>Contents</i>			
A	6	Variable symbol name			
B	1	Not used			
C	1	Attribute of value (located in 'E')			
D	1	Length of value (0-60 bytes) located in 'E'			
E	0-60	Variable symbol value			

Figure A-25. Variable Symbol Table Entry Format

	A	B	C	D	E
<i>Chart ID</i>	<i>Number of Bytes</i>	<i>Contents</i>	<i>Description</i>		
A	1	Type	Indicates whether entry is source program or procedure		
B	6	Name	Name of the source program or procedure		
C	2	Address of first sector in chain	Cylinder/sector address of first sector in chain		
D	2	Address of last sector in chain	Cylinder/sector address of last sector in chain		
E	2	Number of sectors (attribute)	Number of sectors in chain. The high order bit indicates:		
			0 = Permanent entry		
			1 = Temporary entry		

Figure A-26. Source Library Directory Format



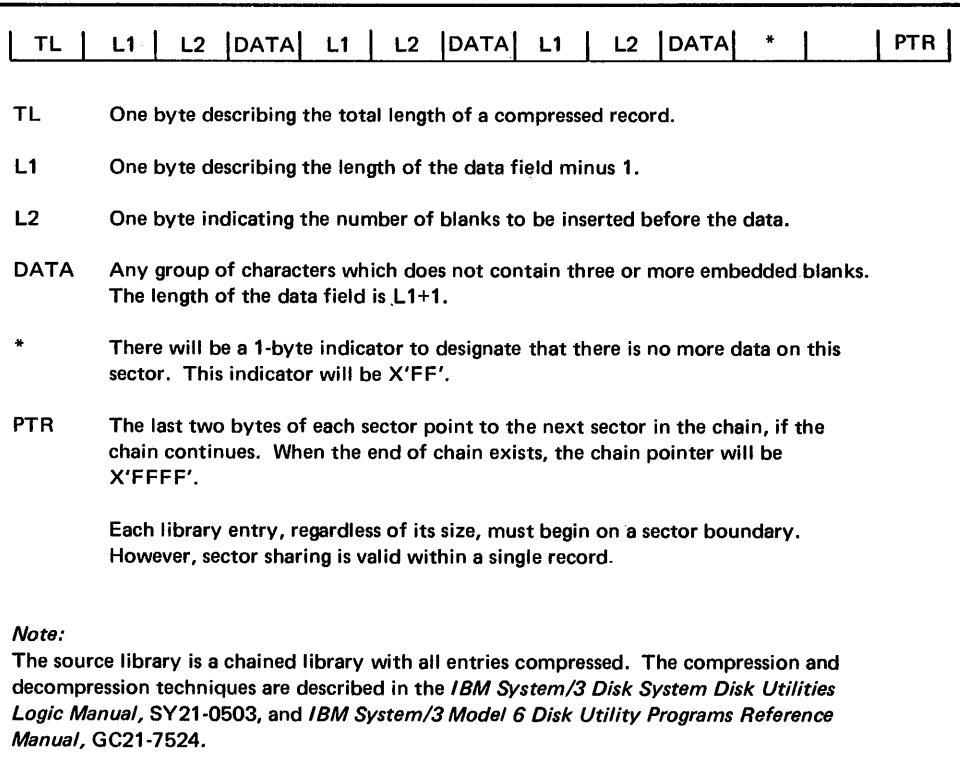


Figure A-27. Source Library Entry Format

Translate Table 1—Domestic (English)

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7C'	X'12'	CHARACTER @
XL1'7B'	X'13'	CHARACTER #
XL1'5B'	X'14'	CHARACTER \$
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'7D'	X'18'	CHARACTER '
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'7E'	X'2F'	CHARACTER =
XL1'4E'	X'30'	CHARACTER +
XL1'4B'	X'31'	CHARACTER .
XL1'5E'	X'32'	CHARACTER ;
XL1'5C'	X'33'	CHARACTER *
XL1'6B'	X'34'	CHARACTER ,
XL1'4B'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'6F'	X'37'	CHARACTER ?
XL1'4F'	X'38'	CHARACTER
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'7A'	X'3A'	CHARACTER :
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER -
XL1'5F'	X'3F'	CHARACTER -



Figure A-28. Translate Table 1—Domestic (English)

54030

Translate Table 2—Austria/Germany

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER }
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7A'	X'12'	CHARACTER :
XL1'5E'	X'13'	CHARACTER ;
XL1'7D'	X'14'	CHARACTER '
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER {
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E9'	X'2C'	CHARACTER Z
XL1'E8'	X'2D'	CHARACTER Y
XL1'60'	X'2E'	CHARACTER -
XL1'4E'	X'2F'	CHARACTER +
XL1'5B'	X'30'	WTC REPLACEMENT FOR US \$
XL1'4B'	X'31'	CHARACTER .
XL1'7C'	X'32'	WTC REPLACEMENT FOR US @
XL1'7B'	X'33'	WTC REPLACEMENT FOR US #
XL1'6B'	X'34'	CHARACTER ,
XL1'4B'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'5C'	X'37'	CHARACTER *
XL1'5F'	X'38'	CHARACTER ~
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'4F'	X'3A'	CHARACTER
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER _
XL1'7E'	X'3F'	CHARACTER =

54031

Figure A-29. Translate Table 2—Austria/Germany

Translate Table 3—Belgium/France

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'D8'	X'0A'	CHARACTER Q
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'7C'	X'10'	WTC REPLACEMENT FOR US @
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7D'	X'12'	CHARACTER ^
XL1'7B'	X'13'	CHARACTER #
XL1'5B'	X'14'	WTC REPLACEMENT FOR US \$
XL1'4D'	X'15'	CHARACTER (
XL1'6C'	X'16'	CHARACTER %
XL1'50'	X'17'	CHARACTER &
XL1'6D'	X'18'	CHARACTER -
XL1'4A'	X'19'	WTC REPLACEMENT FOR US 'CENT SIGN'
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'C1'	X'24'	CHARACTER A
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E9'	X'2A'	CHARACTER Z
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E6'	X'2D'	CHARACTER W
XL1'60'	X'2E'	CHARACTER -
XL1'7E'	X'2F'	CHARACTER =
XL1'4E'	X'30'	CHARACTER +
XL1'4B'	X'31'	CHARACTER .
XL1'5E'	X'32'	CHARACTER ;
XL1'5C'	X'33'	CHARACTER *
XL1'6B'	X'34'	CHARACTER ,
XL1'48'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'6F'	X'37'	CHARACTER ?
XL1'4F'	X'38'	CHARACTER
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'7A'	X'3A'	CHARACTER :
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'5D'	X'3E'	CHARACTER)
XL1'5F'	X'3F'	CHARACTER ~



54032

Figure A-30. Translate Table 3—Belgium/France

Translate Table 4—Denmark

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7A'	X'12'	CHARACTER :
XL1'5E'	X'13'	CHARACTER ;
XL1'7D'	X'14'	CHARACTER '
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'4E'	X'2F'	CHARACTER +
XL1'7C'	X'30'	WTC REPLACEMENT FOR US @
XL1'4B'	X'31'	CHARACTER .
XL1'7B'	X'32'	WTC REPLACEMENT FOR US #
XL1'5B'	X'33'	WTC REPLACEMENT FOR US \$
XL1'6B'	X'34'	CHARACTER '
XL1'4B'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'5C'	X'37'	CHARACTER *
XL1'5F'	X'38'	CHARACTER ~
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'4F'	X'3A'	CHARACTER
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER -
XL1'7E'	X'3F'	CHARACTER =

Figure A-31. Translate Table 4—Denmark

54034

Translate Table 5—Norway

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7A'	X'12'	CHARACTER ;
XL1'5E'	X'13'	CHARACTER ;
XL1'7D'	X'14'	CHARACTER '
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'4E'	X'2F'	CHARACTER +
XL1'61'	X'30'	CHARACTER /
XL1'4B'	X'31'	CHARACTER .
XL1'5B'	X'32'	WTC REPLACEMENT FOR US \$
XL1'6B'	X'33'	CHARACTER ,
XL1'7C'	X'34'	WTC REPLACEMENT FOR US @
XL1'7B'	X'35'	WTC REPLACEMENT FOR US #
XL1'4B'	X'36'	CHARACTER .
XL1'4F'	X'37'	CHARACTER
XL1'5C'	X'38'	CHARACTER *
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'5F'	X'3A'	CHARACTER ~
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER _
XL1'7E'	X'3F'	CHARACTER =



Figure A-32. Translate Table 5—Norway

54033

Translate Table 6—Finland/Sweden

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7A'	X'12'	CHARACTER :
XL1'5E'	X'13'	CHARACTER ;
XL1'7D'	X'14'	CHARACTER '
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'4E'	X'2F'	CHARACTER +
XL1'61'	X'30'	CHARACTER /
XL1'4B'	X'31'	CHARACTER .
XL1'4B'	X'32'	CHARACTER .
XL1'6B'	X'33'	CHARACTER ,
XL1'5B'	X'34'	WTC REPLACEMENT FOR US \$
XL1'7B'	X'35'	WTC REPLACEMENT FOR US #
XL1'7C'	X'36'	WTC REPLACEMENT FOR US @
XL1'7F'	X'37'	CHARACTER 'NOT EQUAL'
XL1'5C'	X'38'	CHARACTER *
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'4F'	X'3A'	CHARACTER
XL1'5F'	X'3B'	CHARACTER ~
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER _
XL1'7E'	X'3F'	CHARACTER =

54036

Figure A-33. Translate Table 6—Finland/Sweden

Translate Table 7—Spain

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER J
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7C'	X'12'	CHARACTER @
XL1'7D'	X'13'	CHARACTER '
XL1'5B'	X'14'	CHARACTER \$
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'7E'	X'2F'	CHARACTER =
XL1'4E'	X'30'	CHARACTER +
XL1'4B'	X'31'	CHARACTER .
XL1'7B'	X'32'	WTC REPLACEMENT FOR #
XL1'5C'	X'33'	CHARACTER *
XL1'6B'	X'34'	CHARACTER ,
XL1'4B'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'7F'	X'37'	CHARACTER 'NOT EQUAL'
XL1'4F'	X'38'	CHARACTER
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'4C'	X'3A'	CHARACTER <
XL1'6E'	X'3B'	CHARACTER >
XL1'5E'	X'3C'	CHARACTER ;
XL1'7A'	X'3D'	CHARACTER :
XL1'6D'	X'3E'	CHARACTER -
XL1'5F'	X'3F'	CHARACTER -

Figure A-34. Translate Table 7—Spain

54036



Translate Table 8—Brazil/Portugal

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7C'	X'12'	WTC REPLACEMENT FOR US @
XL1'7B'	X'13'	WTC REPLACEMENT FOR US #
XL1'7D'	X'14'	CHARACTER '
XL1'6C'	X'15'	CHARACTER %
XL1'4A'	X'16'	CHARACTER 'CENT SIGN'
XL1'50'	X'17'	CHARACTER &
XL1'6F'	X'18'	CHARACTER ?
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'5B'	X'20'	WTC REPLACEMENT FOR US \$
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'7E'	X'2F'	CHARACTER =
XL1'4E'	X'30'	CHARACTER +
XL1'4B'	X'31'	CHARACTER .
XL1'D4'	X'32'	CHARACTER M
XL1'5C'	X'33'	CHARACTER *
XL1'48'	X'34'	CHARACTER .
XL1'6B'	X'35'	CHARACTER ,
XL1'61'	X'36'	CHARACTER /
XL1'7F'	X'37'	CHARACTER 'NOT EQUAL'
XL1'4F'	X'38'	CHARACTER
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'7A'	X'3A'	CHARACTER :
XL1'5E'	X'3B'	CHARACTER ;
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER _
XL1'5F'	X'3F'	CHARACTER `

54037

Figure A-35. Translate Table 8—Brazil/Portugal

Translate Table 9—United Kingdom

KEY VALUE	TRANSLATED VALUE	CHARACTER
XL1'F0'	X'00'	CHARACTER 0
XL1'F1'	X'01'	CHARACTER 1
XL1'F2'	X'02'	CHARACTER 2
XL1'F3'	X'03'	CHARACTER 3
XL1'F4'	X'04'	CHARACTER 4
XL1'F5'	X'05'	CHARACTER 5
XL1'F6'	X'06'	CHARACTER 6
XL1'F7'	X'07'	CHARACTER 7
XL1'F8'	X'08'	CHARACTER 8
XL1'F9'	X'09'	CHARACTER 9
XL1'C1'	X'0A'	CHARACTER A
XL1'C2'	X'0B'	CHARACTER B
XL1'C3'	X'0C'	CHARACTER C
XL1'C4'	X'0D'	CHARACTER D
XL1'C5'	X'0E'	CHARACTER E
XL1'C6'	X'0F'	CHARACTER F
XL1'5D'	X'10'	CHARACTER)
XL1'5A'	X'11'	CHARACTER 'UP ARROW'
XL1'7C'	X'12'	CHARACTER @
XL1'7B'	X'13'	CHARACTER #
XL1'4A'	X'14'	ENGLISH POUND SIGN
XL1'6C'	X'15'	CHARACTER %
XL1'5B'	X'16'	ENGLISH CHAR. \$
XL1'50'	X'17'	CHARACTER &
XL1'7D'	X'18'	CHARACTER '
XL1'4D'	X'19'	CHARACTER (
XL1'C7'	X'1A'	CHARACTER G
XL1'C8'	X'1B'	CHARACTER H
XL1'C9'	X'1C'	CHARACTER I
XL1'D1'	X'1D'	CHARACTER J
XL1'D2'	X'1E'	CHARACTER K
XL1'D3'	X'1F'	CHARACTER L
XL1'D4'	X'20'	CHARACTER M
XL1'D5'	X'21'	CHARACTER N
XL1'D6'	X'22'	CHARACTER O
XL1'D7'	X'23'	CHARACTER P
XL1'D8'	X'24'	CHARACTER Q
XL1'D9'	X'25'	CHARACTER R
XL1'E2'	X'26'	CHARACTER S
XL1'E3'	X'27'	CHARACTER T
XL1'E4'	X'28'	CHARACTER U
XL1'E5'	X'29'	CHARACTER V
XL1'E6'	X'2A'	CHARACTER W
XL1'E7'	X'2B'	CHARACTER X
XL1'E8'	X'2C'	CHARACTER Y
XL1'E9'	X'2D'	CHARACTER Z
XL1'60'	X'2E'	CHARACTER -
XL1'7E'	X'2F'	CHARACTER =
XL1'4E'	X'30'	CHARACTER +
XL1'4B'	X'31'	CHARACTER .
XL1'5E'	X'32'	CHARACTER ;
XL1'5C'	X'33'	CHARACTER *
XL1'6B'	X'34'	CHARACTER ,
XL1'4B'	X'35'	CHARACTER .
XL1'61'	X'36'	CHARACTER /
XL1'6F'	X'37'	CHARACTER ?
XL1'4F'	X'38'	CHARACTER
XL1'40'	X'39'	CHARACTER 'BLANK'
XL1'7A'	X'3A'	CHARACTER :
XL1'7F'	X'3B'	CHARACTER 'NOT EQUAL'
XL1'4C'	X'3C'	CHARACTER <
XL1'6E'	X'3D'	CHARACTER >
XL1'6D'	X'3E'	CHARACTER -
XL1'5F'	X'3F'	CHARACTER -



Figure A-36. Translate Table 9—United Kingdom

54038



APPENDIX B.
DIAGNOSTIC AIDS

B

This section contains information that is designed to aid support personnel in:

1. Understanding System/3 linkages between programs.
2. Locating and retrieving user-specified storage areas.

Some routines and all data areas that are discussed in this section are described in other parts of this document.

CURRENT PHASE IDENTIFICATION PROCEDURE

The user can identify the program that is currently being executed by either taking a main storage dump or by using the console Address/Data switches.

Main Storage Dump (Phase Identification)

The name of the module that has control at the time of the main storage dump can be located as follows:

1. Determine the value at location X'11'.
2. Add to this value X'1' (if operating in program level one) or to X'3' (if operating within program level two). The result of this addition will be an address.
3. Determine the contents located at this address.
4. Add X'3D' to the contents (determined in step 3).
5. The result is the address of the rightmost byte of a 6-byte data area that contains the name of the current control module.

Console Address/Data Switches (Phase Identification)

The name of the module that is currently being executed can be determined as follows:

1. Press console STOP.
2. Set the roller-bar on display #3.
3. Set CE Mode Selector switch to ALTER SAR.
4. Set two rightmost Address/Data switches to 11.
5. Press console START.
6. Set CE Mode Selector switch to DISPLAY STOR.
7. Press console START to display address of system communication region.
8. Add 1 to displayed address (if in Program Level 1). Add 3 to displayed address (if in Program Level 2)—Disk System only.
9. Set two rightmost Address/Data switches to the resulting address.
10. Set CE Mode Selector switch to ALTER SAR.
11. Press console START.
12. Set CE Mode Selector switch to DISPLAY STOR.
13. Press console START button.
14. Add X'3D' to the displayed value.
15. Set two rightmost Address/Data switches to the resulting value.
16. Set CE Mode Selector switch to ALTER SAR.
17. Press console START.
18. Set CE Mode Selector to DISPLAY STOR.
19. Press console START.
20. The value displayed is the rightmost character of the module name.
21. Decrement rightmost Address/Data switch by 1.
22. Repeat steps 16 through 21 until the entire 6-byte data area has been displayed.

HALT/SYSLOG PROCEDURE

Setting Up Halt/Syslog Device for Logging

The function of Halt/Syslog is to provide a means of communication between the system programs and the operator. Halt/Syslog is used to indicate to the operator error conditions, error recovery procedures, and the validity of the operator's response to some halts.

When the user initially performs IPL, the system selects the printer as the system logging device. The user has the option of changing this device by inserting a LOG control statement containing the user-specified system logging device. System/3 currently supports the following devices to be used as Halt/Syslog logging devices:

Disk System

- 5203 Line Printer (Model A1)—supported by \$\$STOP
- 5475 Console Printer—supported by \$\$STOC

Model 6 System

- 2213 or 2222 Printer (Model A)—supported by \$\$STON
- CRT—supported by \$\$STOT

Calling Halt/Syslog Routine

Each Halt/Syslog Routine requires a parameter list as input (see Figure B-1). There are three types of parameter lists. Each parameter list supports at least one of the following Halt/Syslog functions:

<i>Function</i>	<i>Parameter Format</i>
Output only	A
Halt only	B
System message without halt	C
System message with halt	B

The Halt/Syslog calling routine must load register 2 with the address of the high order byte of the parameter list, and the ARR must contain the address of the caller's next sequential instruction (NSI). Register 1 will be stored by the Halt/Syslog routine and will be restored upon completion. Control is passed to the Supervisor with a Halt/Syslog RIB of 85 (normal) or C5 (refresh).

LA	HSPARM,XR2	Address of the halt parm is loaded into reg 2.
B	NCENTR	Control is passed to the Supervisor.
DC	XL1'85'	Halt/Syslog is requested.

B

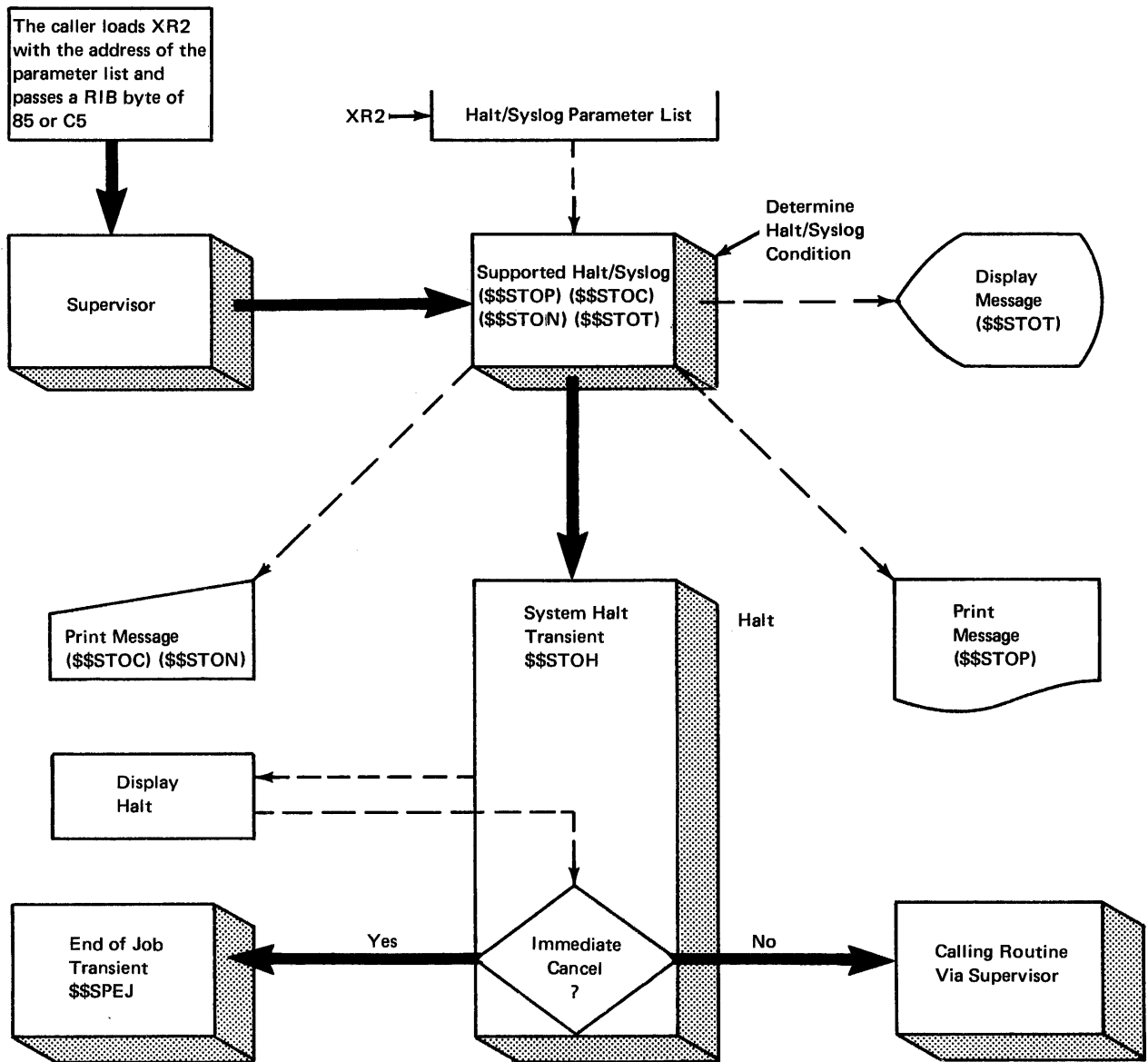


Figure B-1. Functional Flow of Halt/Syslog Logic

Halt/Syslog Parameter Formats

With each Halt/Syslog request, the calling routine must supply one of the parameters (A, B, or C) shown in Figure B-2.

<i>Format</i>	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
A	FF SS XXXX	PPPPPPPP	LLLLLLLL	AAAAAAAA	AAAAAAAA	Blank	Blank
B	FF SS CCCC	MMMMMMMM	HHHHHHHH	HHHHHHHH	RRRR OOOO	IIIIIII	IIIIIII
C	FF SS CCCC	MMMMMMMM	IIIIIII	IIIIIII			

<u>FF</u>	<i>Function Request</i>	<i>Format</i>
00	Output only	A (Use 5 or 7 bytes)
01	Halt only	B (Use 5 bytes only)
10	System message without halt	C (Use 4 bytes only)
11	System message and halt	B (Use 5 or 7 bytes)

<u>SS</u>	<i>Special Options</i>
S0	0 – halt normally 1 – do not release transient area if DPF for Disk System or console support, ignore log status for Model 6
OS	0 – 5-byte parameter list 1 – 7-byte parameter list

The 7-byte parameter list will cause two extra bytes of information to be logged for a system print and halt request. If output only, the carriage positions will be returned in these bytes, the left carriage given first. (CRT does not support this function.)

<u>I</u>	<i>Information Byte</i>
	Two printable EBCDIC characters will be printed out in addition to the halt code for function 11, or after the component identification for function 10.

<u>XXXX</u>	For output only – 22LC only	
X000	0 – 1 – print with no wait in process	} Disk System Only
0X00	0 – 1 –	
00X0	0 – print and wait 1 – print with no wait	
000X	0 – 1 – wait on last print	

The print with no wait in process bit X'08' will be turned on when the printing operation for a X'02' is started. It is up to the user to turn off the X'02' and turn on the X'01' to perform the wait. These two bits should never be on together.

0X00	0 – Call End of Job on a controlled cancel. 1 – Return control on a controlled cancel.	} Model 6 Only
------	---	----------------

<u>A</u>	<i>Output Address</i>
	This byte contains the address of the high order byte of the output message. There are no boundary restrictions.

Figure B-2. Halt/Syslog Parameter Formats (Part 1 of 3)



H

Halt Code

This halt code is displayed on the console lights. The meaning of the halt code can be found in the *IBM System/3 Disk System Operator's Guide*, GC21-7503, or *IBM System/3 Model 6 Halt Procedure Guide*, GC21-7541. The halt code consists of EBCDIC characters which initiate valid light characters.

R

Operators Reply

This corresponds to the options bits (see *Options*). One of these four bits is set on to indicate the operator's response to the halt. If all bits are on when halt/syslog is called, an either/or request exists: either halt if log is off, or system print if log is on, but not both.

O

Options

These four bits indicate the valid reply options.

1000	Proceed, bypass	(option 0)
0100	Retry, continue	(option 1)
0010	Controlled cancel	(option 2)
0001	Immediate cancel	(option 3)

The exact meaning of each option is dependent upon the purpose of the halt. If option three is allowed, and is selected by the operator, control will be passed to the End of Job transient (\$\$SPEJ), control will not be returned to the caller as in options 0, 1 and 2.

C

Component Identification

Character Displayed

0000	Scheduler	C	
0001	Data Management	D	
0010	Input/Output Supervisor (IOS)	I	
0011	RPG Program	R	
0100	Supervisor	V	
0101	Linkage Editor	E	
0110	Utilities Program	U	
0111	Sort Program	S	
1000	System Generation	G	
1001	Library Maintenance	L	
1010	Basic Assembler Program	A	} Disk System Only
1011	Card Utility Program Product	M	

M

Subcomponent identifier: One printable EBCDIC character supplied by the user.

P

Page Control Option

- | | | |
|---|---|------------------|
| <p>Console</p> <ul style="list-style-type: none"> — The four high order bits contain the number of lines (in hexadecimal) to be skipped before the line is printed. — The four low order bits contain the number of lines to be skipped after the page is printed. — (Default is space one line before printing.) <p>Printer</p> <ul style="list-style-type: none"> — Any bits on of the four high order bits will cause a skip to line one on the next page. — The four low order bits are not used. — (Default is space one line after printing.) | } | Disk System Only |
|---|---|------------------|

Figure B-2. Halt/Syslog Parameter Formats (Part 2 of 3)

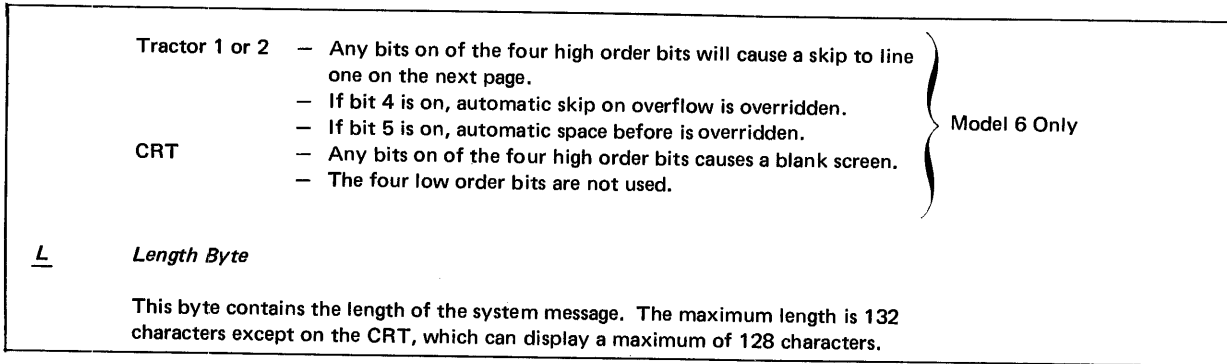


Figure B-2. Halt/Syslog Parameter Formats (Part 3 of 3)

Format of the Output from Halt/Syslog

The following four examples indicate the type of output that can be expected from the Halt/Syslog routines:

Example One: (Disk System)

Function: Output only

Parm Format: A

Input: XR2 →

00 00 0001	X'00'	X'08'	Addr. of AREA
------------	-------	-------	---------------

Light: None

Output on Logging Device: ALL DONE

Note: AREA = C'ALL DONE'

Example Three: (Disk System)

Function: System message without halt

Parm Format: C

Input: XR2 →

10 00 0000	X'C8'	X'D7'	X'D3'
------------	-------	-------	-------

Light: None

Output on Logging Device: CHPL

Note: C indicates Scheduler

Example Two: (Disk System)

Function: Halt only

Parm Format: B

Input: XR2 →

01 00 0000	00000000	X'85'	X'85'	0000 0100
------------	----------	-------	-------	-----------

Light: EJ

Output on Logging Device: None

Note: Subcomponent was not specified

Example Four: (Model 6)

Function: System message with halt

Parm Format: B

Input: XR2 →

11 00 0001	X'C5'	X'C5'	X'D1'	0000 0100
------------	-------	-------	-------	-----------

Light: D5

Output on Logging Device: DEEJ 1

Note: Controlled cancel is requested by Data Management



LOCAL STORAGE REGISTER DISPLAY PROCEDURE

This display procedure allows the user to display the contents of any local storage register without destroying the contents of that register. Displaying is accomplished by:

1. Set instruction address register (IAR) to X'E8'.
2. Set console Address/Data switches as follows:
 - a. The two leftmost switches to 34 (to store a register) or 30 (to sense a register).
 - b. The two rightmost switches to the register to be displayed.
3. Press console START. A halt condition is displayed on the console lights. Ignore the halt. The Q byte of this halt is the high order byte of the register being displayed. (This is displayed by the lights on roller three.)
4. Press console START again. Another halt condition will occur. The Q byte of this halt is the low order byte of the register being displayed. (This is displayed by the lights on roller three.)
5. Set two rightmost console Address/Data switches to the next register to be displayed.

Figure B-3 contains all of the local storage register numbers, contents, and acronyms.

TRACING THE TRANSIENT QUEUE

The transient queue contains from zero to three 9-byte entries. (See *Section 2. Method of Operation in Part 2. Disk Dedicated Supervisor* for the format of the transient queue entries.) The transient queue address can be located at LABEL NXQUE, a displacement of X'84' for Disk System or X'05' for Model 6, into the system communication region. The contents of this queue can be displayed at any time by the user, by following the main storage dump procedure outlined under *Storage Dump Selection Procedure*.

Base System For Disk System

LSR No.	High	Low	LSR Acronym
01	Program level 1 instruction address register		P1-IAR
02	Program level 1 address recall register		P1-ARR
03	Operand 2 address register		ARR
04	Spare		
05	Program level 1 index register 1		P1-XR1
06	Length count recall register	Condition recall register	P1-PSR
07	Operand 1 address register		BAR
08	MFCU print data address register		MPTAR
09	Program level 1 index register 2		P1-XR2
10	Line printer data address register		LPDAR
11	Line printer image address register		LPIAR
12	MFCU punch data address register		MPCAR
13	MFCU read address register		MRDAR
14	Length count registers	Data recall register	LCR DRR
15	Interrupt level 1 instruction address register		IAR-1
16	Interrupt level 1 address recall register		ARR-1

Base System For Model 6 System

LSR No.	High	Low	LSR Acronym
01	Instruction Address Register		IAR
02	Address recall register		ARR
03	Operand 2 address register		AAR
04	LCD locate line address register		LLAR
05	Index register 1		XR1
06	Program status register		PSR
07	Operand 1 address register		BAR
08	Disk file control address register		DFCR
09	Index register 2		XR2
10	Printer data address register		PDAR
11	Printer command address register		PCAR
12	Data recorder address register		DRAR
13	Disk file data address register		DFDR
14	Length count register	LCR/data recall register	DRR
15	Interrupt level 1 instruction address register		IAR-1
16	Interrupt level 1 address recall register		ARR-1

Figure B-3. Table of Local Storage Registers



STORAGE DUMP SELECTION PROCEDURE

Disk System

The user can elect to print out either:

- The entire contents of main storage
- The contents of a specified area of main storage

To display the entire Main Storage area:

1. Set Address/Data switches to any value other than CEFE.
2. Press SYSTEM RESET.
3. Press console START.

The entire System/3 main storage area will be printed out on the Syslog device. After the last byte has been displayed, the EJ halt code will be displayed on the console stick lights.

The user can selectively print the contents of a specified area of storage by performing the following steps:

1. Press console STOP.
2. Set Address/Data switches to CEFE.
3. Press SYSTEM RESET.
4. Press console START. A halt code of 50 will be displayed on the console stick lights.
5. Set rightmost Address/Data switch to:
 - 0—Indicating a request for a main storage dump.
 - 1—Indicating a request for a DTF dump.
 - 2—Indicating a request for a disk dump.

Any other setting will cause a halt code of 50.

6. Press console START.

The following steps are performed only for a main storage dump.

7. After console START has been pressed, a halt code of 5E will be displayed on the console stick lights. Set the two leftmost Address/Data switches to the 'start of dump address'. Set the two rightmost Address/Data switches to the 'end of dump address'.

8. Press console START again.
9. After the requested area has been displayed, a halt code of 50 will be displayed. The user can either continue displaying storage by going back to step 4 or he can perform the IPL procedure again.

It is recommended that a main storage dump be taken first, then a DTF dump, and finally a disk dump. System/3 does not support an automatic main storage dump at end-of-job or after an abnormal job termination.

Model 6

The user can selectively print the contents of a specified area of storage by performing the following steps:

1. Press SYSTEM RESET. (SYSTEM RESET can be pressed only once between IPL procedures.)
2. Press console START. A halt code of 'D' will be displayed on the console lights.
3. Set rightmost Data/Address switch to:
 - 0—Indicating a request for a main storage dump.
 - 1—Indicating a request for a DTF dump.
 - 2—Indicating a request for a disk dump.

Any other setting will cause a halt code of 'D'.

4. Press console START.

The following steps are performed only for a main storage dump.

5. After console START has been pressed, a halt code of 'D1' will be displayed on the console lights. Set the two leftmost Address/Data switches to the 'start of dump address'. Set the two rightmost Address/Data switches to the 'end of dump address'.
6. Press console START again.
7. After the requested area has been displayed, a halt code of 'D' will be displayed. The user can either continue displaying storage by going back to step 2 or he can perform the IPL procedure again.

It is recommended that a main storage dump be taken first, then a DTF dump, and finally a disk dump. System/3 does not support an automatic main storage dump at end-of-job or after an abnormal job termination.

LINKAGE FOR THE STORAGE DUMP ROUTINES

Disk System

The user effects a branch to location X'00' upon pressing the SYSTEM RESET and console START keys. This location contains the address of the Resident Dump Linkage routine (CDUMPD). This routine writes out the transient area and reads the Dump Selection routine (DMPFND) into the transient area (location X'100'). Control is then passed to the Dump Selection routine at location X'100'. DMPFND senses the Address/Data switches and, depending upon the settings, either:

1. Issues a halt code of 50 on the console stick lights (value of CEFE).
2. Passes control to the Main Storage Dump routine (value other than CEFE).

If the halt code of 50 is issued, DMPFND senses the Address/Data switches after the User has pressed the START key. Control is then passed to the supporting routine. Upon the completion of the requested dump, control is returned to the Dump Selection routine (DMPFND). DMPFND then re-issues a halt code of 50.

Model 6

The user effects a branch to location X'00' upon pressing the SYSTEM RESET and console START keys. This location contains the address of the Resident Dump Linkage routine (CDUMPD). This routine writes out the transient area and reads the Dump Selection routine (DMPFND) into the transient area (location X'100'). Control is then passed to the Dump Selection routine at location X'100'. DMPFND then issues a halt code of 'D' on the console lights. After 'D' is issued, DMPFND senses the Address/Data switches when the user has pressed START. Control is then passed to the supporting routine. Upon completion of the requested dump, control is returned to the Dump Selection routine (DMPFND). DMPFND then re-issues a halt code of 'D'.

► Dump Selection Routine (DMPFND)

CHART: QA
FIGURE: None
ENTRY POINT: DMPFND

FUNCTION:

- If the Address/Data switches are set to CEFE, a halt of 50 results. DMPFND then:
 1. Tests the rightmost Address/Data switch for the type of requested dump:
 - Main Storage – 0
 - DTF Dump – 1
 - Disk Dump – 2
 2. Passes control to the appropriate dump processor
- If the Address/Data switches are not set to CEFE, DMPFND passes control to the Main Storage Dump routine (DUMPPG).

INPUT: Console Address/Data switches

OUTPUT: None

EXIT: Control is passed to the appropriate dump processor. At end-of-job, a halt of 'EJ' on the Disk System, 'D5' on Model 6, is issued and displayed on the console lights. The operator must perform the IPL procedure again to load the next program.

► Main Storage Dump Routine (DUMPPG) (Option 0)

CHART: QB

FIGURE: None

ENTRY POINT: DUMPPG

FUNCTION:

- Prints out all main storage.
- Calls Format Dump (\$\$SPD1, Disk System, \$\$SPD0, Model 6) to print registers and transient area.

INPUT: Main Storage

OUTPUT: All of main storage and the transient area are printed using the 5203 or 5213 Printer. If duplicate lines are encountered the Printer will skip a few lines rather than print them out.

EXIT: Control is returned to the Dump Selection routine (DMPFND) if the console Data/Address switches were set to CEFE. Otherwise a halt of 'EJ' on the Disk System, 'D5' on Model 6, is issued.

► DTF Dump Routine (\$\$SPDF on the Disk System, \$\$SPD3 on Model 6) (Option 1)

CHART: QC

FIGURE: None

ENTRY POINT:

- \$\$SPDF, Disk System
- \$\$SPD3, Model 6

FUNCTION: Prints all chained DTFs and associated IOBs

INPUT: The DTFs and IOBs that are to be printed are located in main storage.

OUTPUT: The DTFs and associated IOBs are printed on the 5203 or 5213 Printer.

EXIT: Control is returned to the Dump Selection routine.

B

► **Disk Storage Dump Routine (\$\$SPD4 on the Disk System, \$\$SPD2, Model 6) (Option 2)**

CHART: QD
 FIGURE: B-5
 ENTRY POINT: DISKDP
 FUNCTION:

- Displays a halt of '55' on the Disk System, '02' on Model 6, on the console lights to indicate to the operator that the starting sector of the area to be displayed must be indicated on the two rightmost Address/Data switches (Figure B-4).

- Displays a halt of 'E5' on the Disk System, 'D3' on Model 6, on the console lights to indicate to the operator that end sector of the area to be displayed must be indicated on the two rightmost Address/Data switches (Figure B-4).
- Senses the Address/Data switch settings and performs the requested dump.

INPUT: Console Address/Data switches (Figure B-4)

OUTPUT: Listing containing the contents of the specified area on disk

EXIT: User must perform IPL procedure again to exit from this routine.

The following address/data switch settings are used to indicate the specified area on disk that is to be displayed.

SWITCH	SETTING	MEANING
The two leftmost address/data switches	00-CB*	Selected cylinder number on the specified disk *01, 02, and 03 are alternate tracks and can't be dumped. To display the information on these tracks, specify the primary tracks.
The two rightmost address/data switches	See the following table	Sector number (beginning or end) on the specified disk that is to be displayed.

Sector	R1	F1	R2	F2
0	00	01	02	03
1	04	05	06	07
2	08	09	0A	0B
3	0C	0D	0E	0F
4	10	11	12	13
5	14	15	16	17
6	18	19	1A	1B
7	1C	1D	1E	1F
8	20	21	22	23
9	24	25	26	27
10	28	29	2A	2B
11	2C	2D	2E	2F
12	30	31	32	33
13	34	35	36	37
14	38	39	3A	3B
15	3C	3D	3E	3F
16	40	41	42	43
17	44	45	46	47
18	48	49	4A	4B
19	4C	4D	4E	4F
20	50	51	52	53
21	54	55	56	57
22	58	59	5A	5B
23	5C	5D	5E	5F

Sector	R1	F1	R2	F2
24	80	81	82	83
25	94	95	96	97
26	88	89	8A	8B
27	8C	8D	8E	8F
28	90	91	92	93
29	94	95	96	97
30	98	99	9A	9B
31	9C	9D	9E	9F
32	A0	A1	A2	A3
33	A4	A5	A6	A7
34	A8	A9	AA	AB
35	AC	AD	AE	AF
36	B0	B1	B2	B3
37	B4	B5	B6	B7
38	B8	B9	BA	BB
39	BC	BD	BE	BF
40	C0	C1	C2	C3
41	C4	C5	C6	C7
42	C8	C9	CA	CB
43	CC	CD	CE	CF
44	D0	D1	D2	D3
45	D4	D5	D6	D7
46	D8	D9	DA	DB
47	DC	DD	DE	DF

Figure B-4. Table of Address/Data Switch Settings for the Disk Dump Routine

► Dump Format Routine (\$\$SPD1, Disk System, \$\$SPD0, Model 6)

CHART: QE

FIGURE: B-5

ENTRY POINT: START

FUNCTION:

- Prints the register headings.
- Prints the register contents.
- Prints the transient area heading.
- Prints the contents of the saved transient area.

INPUT: The address of the registers to be printed is passed to this routine by the supervisor.

OUTPUT: The formatted dump of the registers and the saved transient area are printed on the 5203 or 5213 Printer.

EXIT: Control is returned to the supervisor

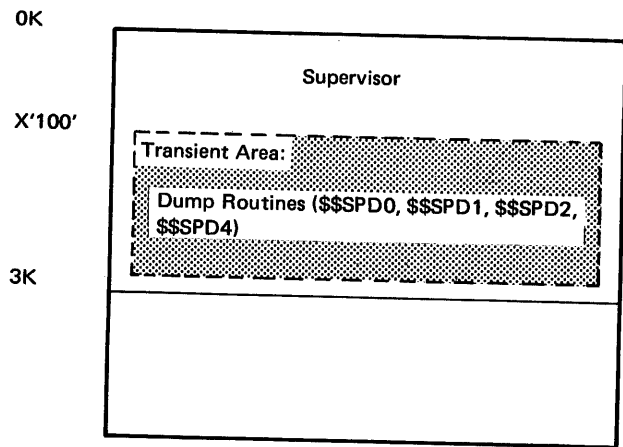


Figure B-5. Main Storage Map for Dump Routines (\$\$SPD0, \$\$SPD1, \$\$SPD2, \$\$SPD4)

B

REQUEST INDICATOR BYTE (RIB)

The RIB is used as a means of communication between calling routines and the Supervisor. The RIB can be used in one of four ways:

- To specify the type of load to be performed on a specified module when the location of the requested module is unknown (Figure B-6). For example:

LA	TABLE, XR2	The requested module table address is loaded into register 2. TABLE points to the leftmost byte of a parameter list for the module.
B	NCENTR	Control is passed to the Supervisor.
DC	XL1'71'	This RIB specifies a FIND, using the System IOB, and loading with a FETCH.

RIB Values for the Loader

Meaning	Bit	0	1	2	3	4	5	6	7
Use System IOB		0	1	1	0	0	0	0	0
Use Program IOB		0	1	0	0	0	0	0	0
Fetch		0	1	0	1	0	0	0	0
Fetch to Address		0	1	0	1	1	0	0	0
System Fetch		0	1	0	1	0	1	0	0
Load		0	1	0	0	1	0	0	0
RPG Load		0	1	0	0	0	0	0	0
Without Find		0	1	0	0	0	0	0	0
With Find		0	1	0	0	0	0	0	1

The above values can be combined to form multiple RIB meanings. The following examples indicate some of the possibilities. The hexadecimal value on the right is used as the RIB value.

Examples	0	1	2	3	4	5	6	7	HEX
Fetch, SYS IOB	0	1	1	1	0	0	0	0	70
Fetch, PGM IOB	0	1	0	1	0	0	0	0	50
Fetch, PGM IOB, Find	0	1	0	1	0	0	0	1	51
Fetch to @, SYS IOB	0	1	1	1	1	0	0	0	78
Fetch to @, PGM IOB	0	1	0	1	1	0	0	0	58
Fetch to @, SYS IOB, Find	0	1	1	1	1	0	0	1	79
System Fetch, SYS IOB	0	1	1	1	0	1	0	0	74
System Fetch, PGM IOB	0	1	0	1	0	1	0	0	54
Load, SYS IOB	0	1	1	0	1	0	0	0	68
Load, PGM IOB	0	1	0	0	1	0	0	0	48
Load, PGM IOB, Find	0	1	0	0	1	0	0	1	49
RPG Load, PGM IOB	0	1	0	0	0	0	0	0	40

Figure B-6. Request Indicator Byte Values for the Loader

2. To specify the type of load to be performed on a transient when the location of the requested transient module is known (Figure B-7). For example:

B NCENTR Control is passed to the Supervisor.

DC XL1'80' RIB to load a transient when the disk address is known.

DC CL3'###' Cylinder/sector/number of the requested module.

3. To specify a Supervisor transient (Figure B-7).

LA PARM,XR2 Parameter to be passed to the called routine.

B NCENTR Control is passed to the Supervisor.

DC XL1'85' RIB request for a Supervisor transient.

Note: The RIB value can be found under *Normal Value* on Figure B-7.

4. To specify that the caller is a transient and that it is to be restored after the requested routine has completed processing. For example:

LA PARM,XR2 Parameter to be passed to the called program.

B NCENTR Control is passed to the Supervisor.

DC XL1'C5' RIB request for a Supervisor transient.

Note: The RIB value can be found under *Refresh Value* on Figure B-7. If the caller A is a transient and a refresh value is used to load Supervisor Transient B, caller A is restored when transient B is finished processing. If caller A used a normal value for the RIB request, A would not be restored after B finished processing.

Transient	Normal Value	Refresh Value
Request Transient by C/S/#	80	C0
Find a transient	81	C1
Open	82	C2
Close	83	C3
End of Job (\$\$SPEJ)	84	C4
Supported Halt/Syslog	85	C5
Supported Sysin	86	C6
Scheduler Work Area—Get (\$\$SSGT)	87	C7
Scheduler Work Area—Put (\$\$SSPT)	88	C8
Scheduler Work Area—R/W (\$\$SSSC)	89	C9
VTOC Read/Write (\$\$SSVT)	8A	CA
Allocation Initiator (\$\$STAI)	8B	CB
Program Product Counter (\$\$SYPP)	8C	CC
Transient Free (\$\$STNQ)	8D	CD
Rollout (\$\$STRO)	8E	CE
RPG Halt Processor	8F	CF
OBR/SDR	90	D0

Figure B-7. Request Indicator Byte Values for the Supervisor Transients



THE RIB FETCH/TRACE PROCEDURE

1. Perform the IPL procedure. If running in a DPF environment, press HALT RESET for program level two. (The Fetch/Trace routine must always be initiated from program level two when using a DPF system.)
2. Load the Fetch/Trace routine (\$\$FTRC) using the following control statements:


```

// DATE 00/00/00
// LOAD $$FTRC,R1
// RUN
      
```
3. After the Fetch/Trace routine is loaded, control is passed to the End of Job transient (\$\$SPEJ) resulting in an EJ halt. The Fetch/Trace routine remains in main storage until the next IPL procedure is performed.
4. The user must set on bit 0 of NCFCTR (located in the system communication region) in order to use the Fetch/Trace routine. If you want the system to halt after the RIB information is printed, bit 7 of NCFCTR must be set on if the rightmost Address/Data switch will be set to an odd number. Otherwise it is set off if the rightmost Address/Data switch is set at an even number. Setting bit 7 on is done when address stop is being used. To set on bits 0 and 7, perform the following procedure:
 - a. Display and record the value of the IAR.
 - b. Display the address of the system communication region by displaying the value of main storage locations X'10' and X'11'.
 - c. Add 6 to the 2-byte system communication region address. (The resulting value is the address of the Fetch/Trace byte—NCFCTR.)
 - d. Alter the SAR to the address of NCFCTR.
 - e. Alter bit 0 of NCFCTR.
 - f. Alter the SAR to the address saved in step a.
5. Fetch/Trace can be terminated at any time by performing step 4, setting bit 0 off. The Fetch/Trace routine can be activated or de-activated as often as desired.

Figure B-8 shows the location of the Fetch/Trace routine within a dedicated system; Figure B-9 shows its location within a DPF system.

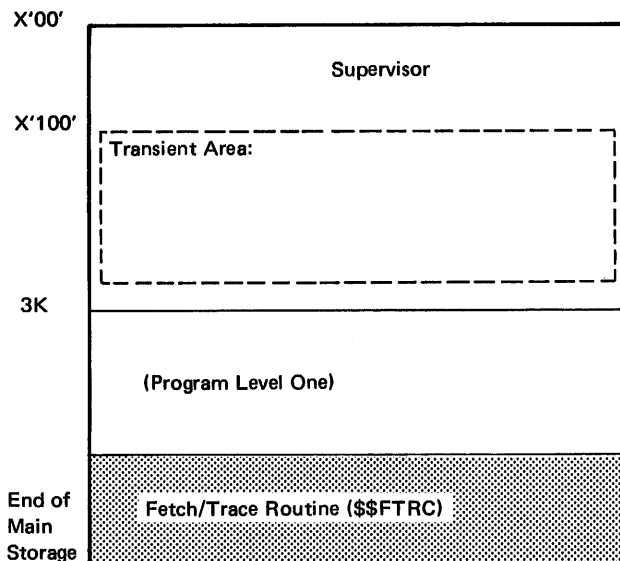


Figure B-8. Main Storage Map for Fetch/Trace Routine (\$\$FTRC) Within a Dedicated System

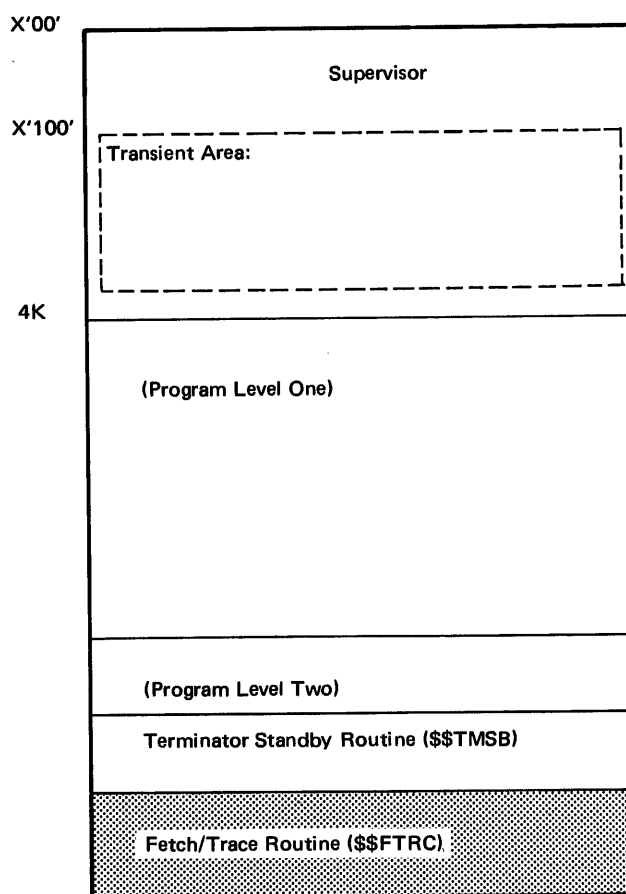


Figure B-9. Main Storage Map for Fetch/Trace Routine (\$\$FTRC) Within a DPF System

► RIB Fetch/Trace Routine (\$\$FTRC)

CHART: QF

FIGURE: None

ENTRY POINT: \$\$FTRC

FUNCTION:





- Loads FTRACE (located within \$\$FTRC) at the end of main storage.
- Sets a new upper limit for main storage.
- Prints out RIB trace information.

INPUT: Bit 0 of the fetch/trace byte (NCFCTR—located within the system communication region)

OUTPUT: The following printed RIB trace information:







- Program level the traced program is running in (LV-x)
- Request indicator byte (RIB-xx)
- Contents of register 1 (XR1-xxxx)
- Contents of register 2 (XR2-xxxx)
- Program status register (PSR-xxxx)
- Address recall register (ARR-xxxx)
- Loader request parameters. The information printed will depend on whether the requested module to be loaded for the calling program is requested by name or by address.

The format of the printed information, if requested by name, is:

LDR—type (O or R)	name	P=program pack find S or b=system find	load address
 1 byte	 6 bytes	 1 byte	 2 bytes

All other LDR requests print the parameter list needed for the function being performed. The various parameter list formats may be found by identifying the RIB and using the associated program listing.

The format of the printed information, if requested by address, is:

CS#—disk address	number of sectors	link edit address	RLD displacement	entry point	load address
 2 bytes	 1 byte	 2 bytes	 1 byte	 2 bytes	 2 bytes

A transient request of X'80' or X'C0' causes the following to be printed:

CS#—disk address (C/S/#)

EXIT: Control is passed to the End of Job Transient (\$\$SPEJ)

► Program Temporary Patch Program (\$\$SGFIX)

CHART: QG

FIGURE: None

ENTRY POINT: \$\$SGFIX

FUNCTION:

- Temporarily patches object modules.
- Logs the PTF with an X'FF' in the sixth byte of the log entry when a system module is patched.

INPUT: The following control statements:

- Header (HDR)
- PTF
- DATA
- END
- /*

OUTPUT: Patched modules replaced in their original position on disk.

EXIT:

- Normal: Control is passed to the End of Job Transient (\$\$SPEJ)
- Error: Control is passed to the supported Halt/Syslog routine.



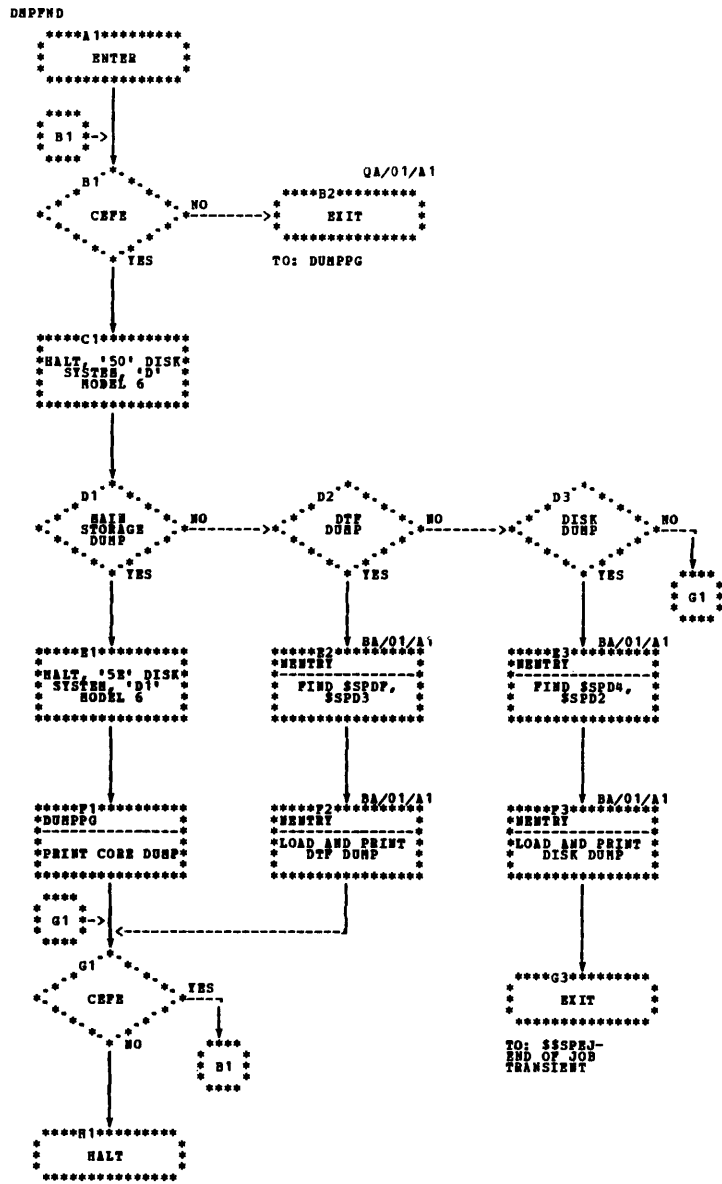


Chart QA. Dump Selection Routine (DMPFND)

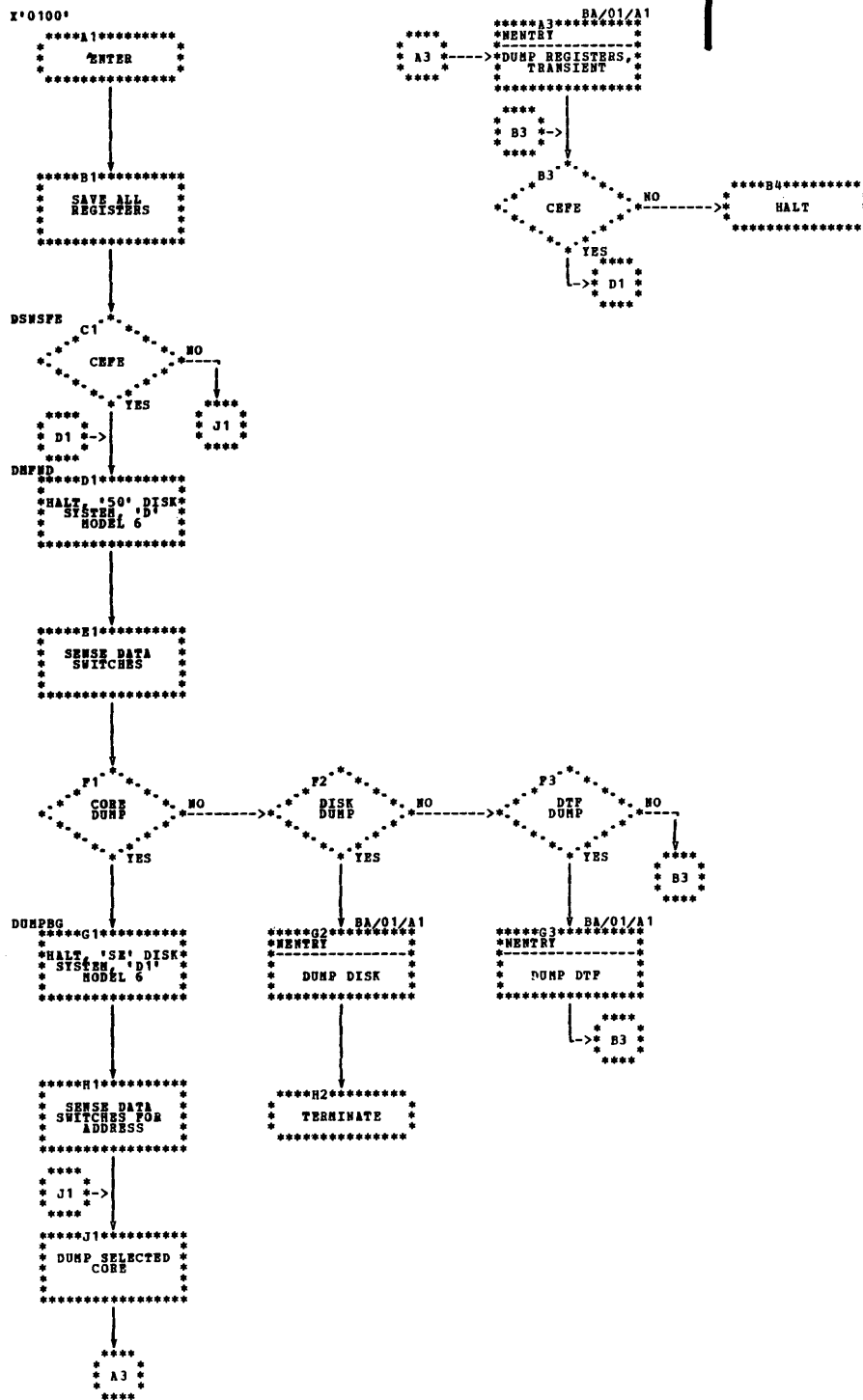


Chart QB. Main Storage Dump Routine (DUMPPG)



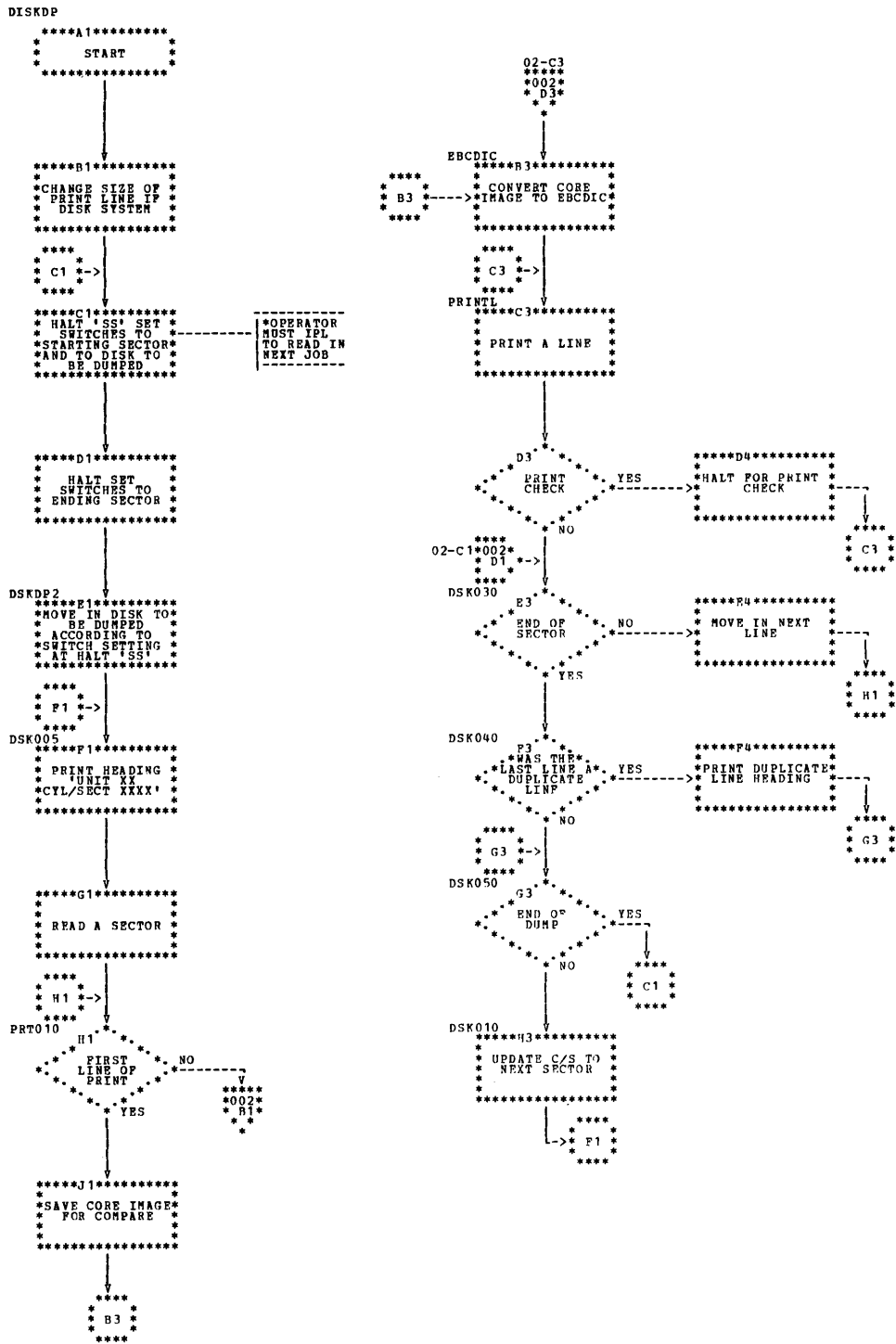
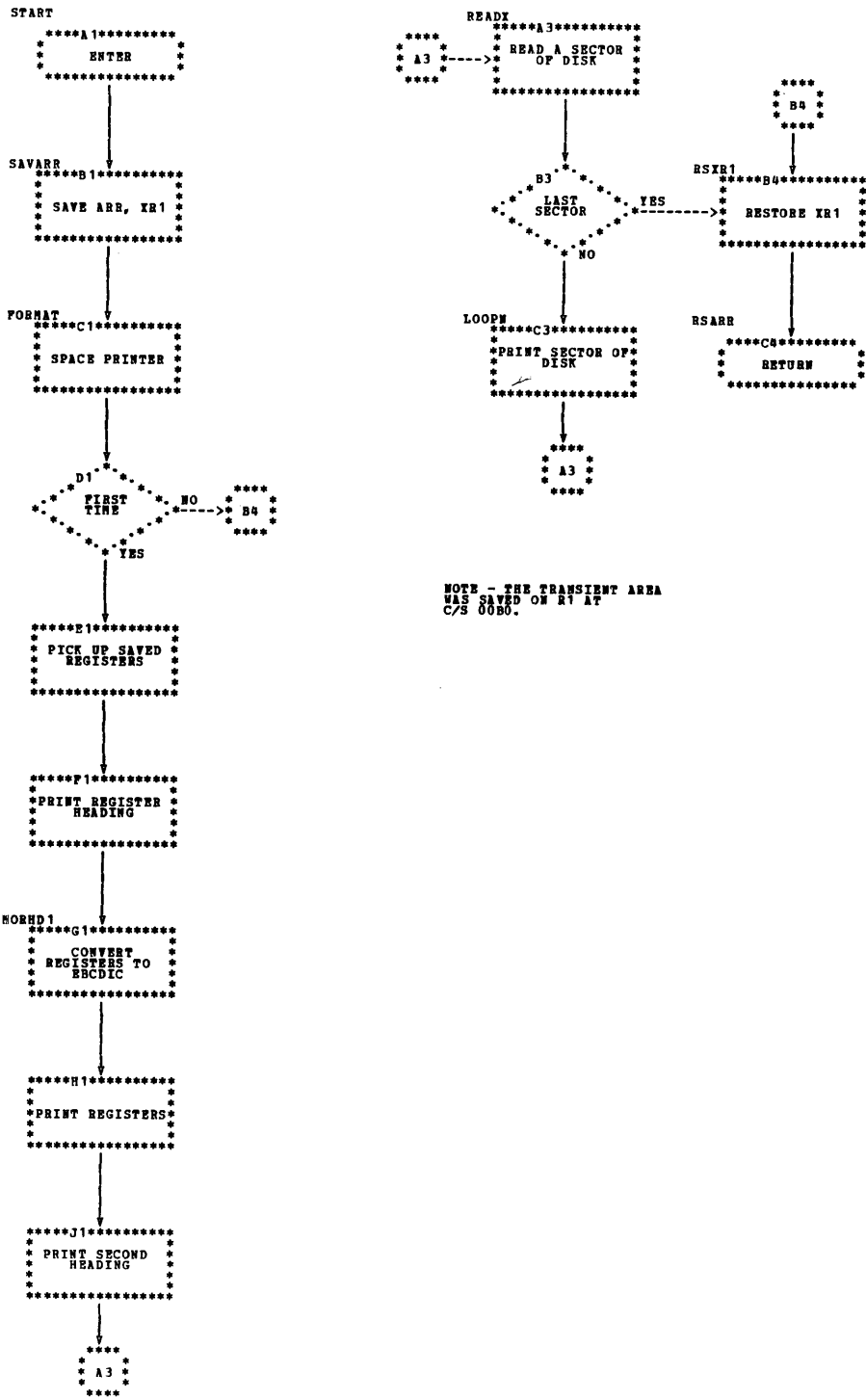


Chart QD (Part 1 of 2). Disk Storage Dump Routine (\$SPD4, Disk System; \$SPD2, Model 6)



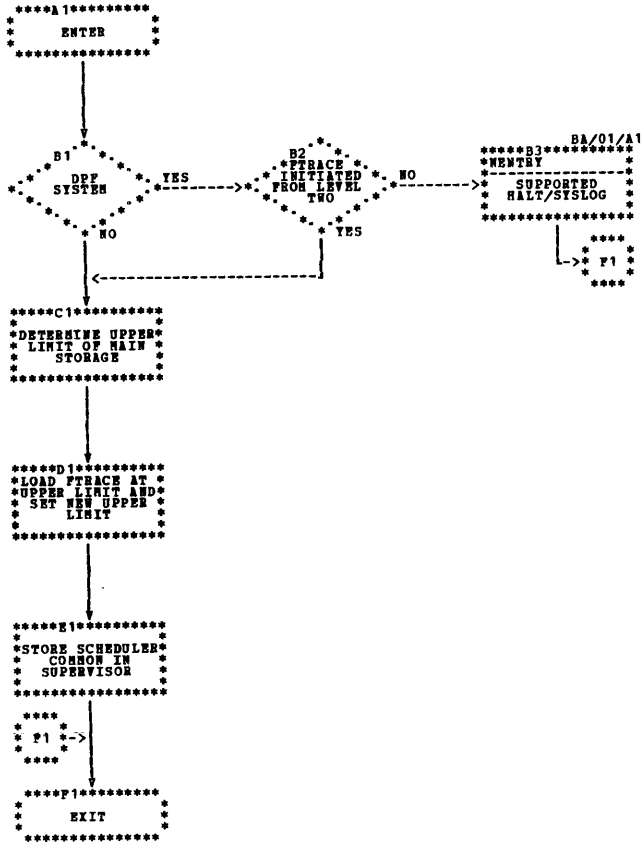


NOTE - THE TRANSIENT AREA
WAS SAVED ON R1 AT
C/S 00B0.

Chart QE. Dump Format Routine (\$\$SPD1, Disk System; \$\$SPD0, Model 6)

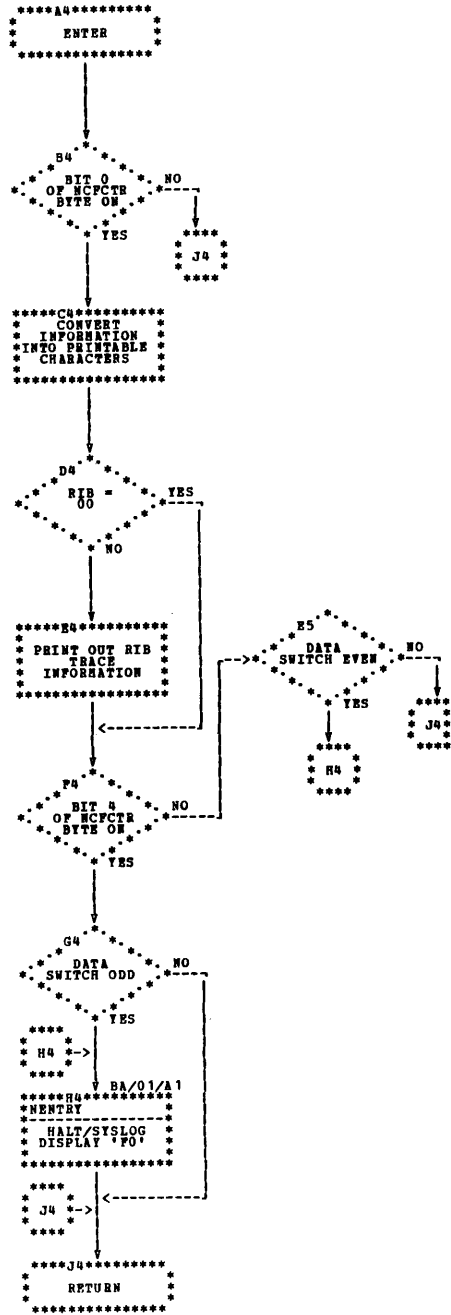


\$\$\$FTRC



TO END OF JOB
TRANSIENT (\$\$SPRJ)

FTRACE



TO: CALLER

Chart QF. RIB Fetch/Trace Routine (\$\$FTRC)

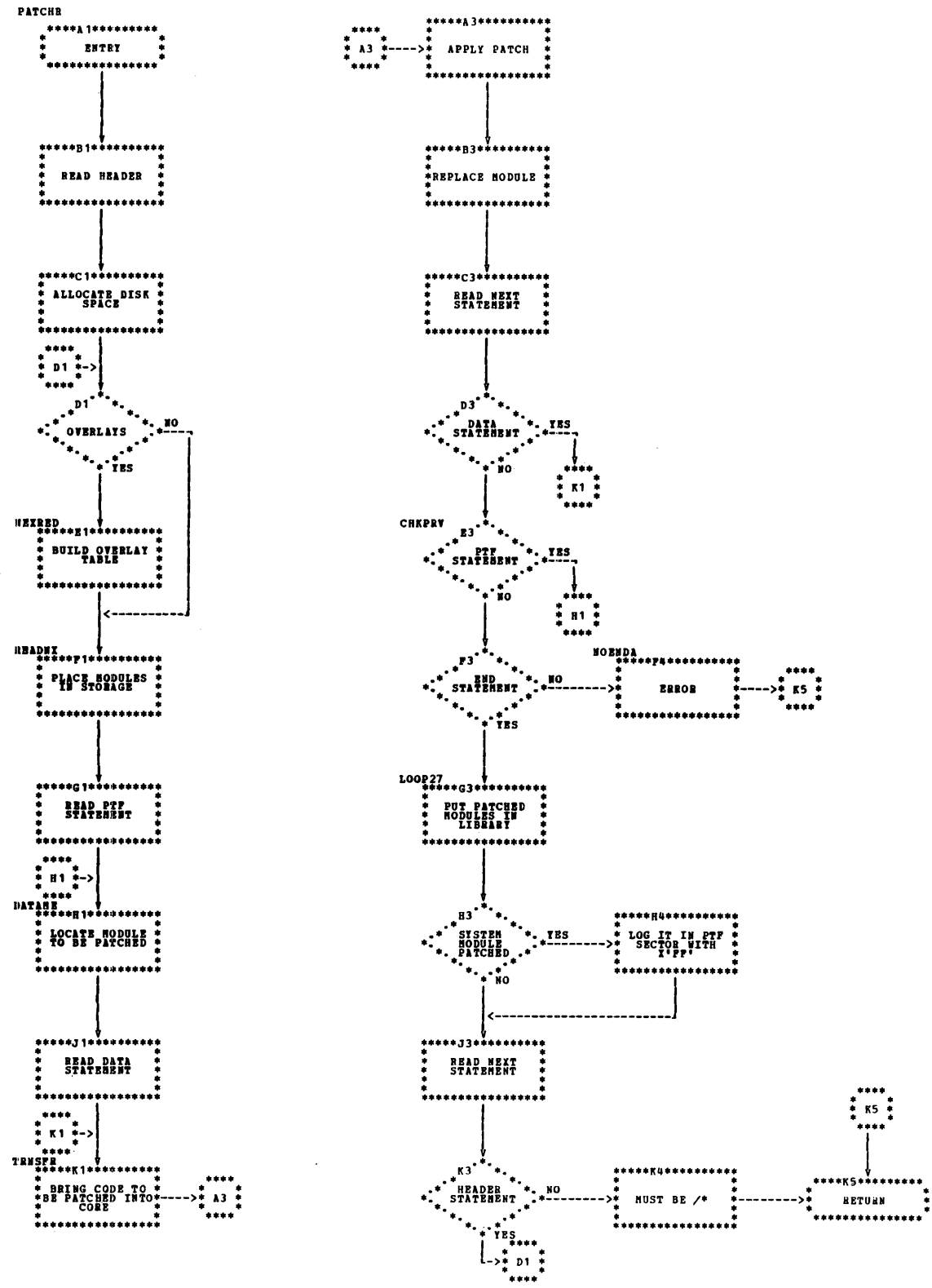


Chart QG. Program Temporary Patch Program (SSGFIX)



PROGRAM TEMPORARY PATCH (\$SGFIX) DESCRIPTION AND PROCEDURES

When an error in an O-type program in the disk resident object library is found and a PTF does not exist which covers the problem, the user can make a temporary repair using the Program Temporary Patch program while waiting for an updated module or PTF to be distributed. Once the user determines the patches necessary to correct the error, the following OCL statements must be prepared:

```
// DATE 00/00/00
// LOAD $SGFIX,unit (R1 or F1)
// RUN
```

The OCL statements are followed by these control statements:

```
HDR O program name,unit
PTF module identification
DATA disp,chkbyte,hh,hhhh,hhhhR
END
/*
```

The following is an explanation of the format and parameters of the control statements.

HDR O program name,unit

HDR—must start in column 1. HDR identifies the control statement and the start of the patch program.

O program name—must start in column 5. The program name must be six characters long. Use blanks if necessary. O program name identifies the object library type and the program name as it exists in the object library directory.

Unit—must start in column 13. Unit identifies the unit on which the patch is to be applied. When patching system modules (module names starting with \$\$), the patch should be applied to the program pack. If you apply the patch to the pack that you did the IPL procedure from and a system module is being patched, unexpected results could occur.

The HDR statement must follow the RUN statement.

PTF Module Identification

PTF—must start in column 1. PTF identifies the control statement.

Module identification—must start in column 5 and be six characters long. The module identification consists of the

first three characters of the module name and three overlay characters. In the case of patching a root phase or a program with no overlays, the overlay numbers will always be 000. If a program is being patched that has overlays, the overlay number will be from 001 (first overlay) to xxx (last overlay) depending on the overlay being patched.

As many PTF statements as necessary to apply the patch can be provided. However, each PTF statement must be followed by at least one DATA statement.

DATA Disp,chkbyte,hh,hhhh,hhhhR

DATA—must start in column 1. DATA identifies the control statement.

Disp—must start in column 6 and be four characters in length. Disp provides the displacement into the module where the patch is applied.

chkbyte—must start in column 11 and be two characters in length. Chkbyte represents the first byte being overlaid by the first byte of patch information on the data statement.

hh—one byte of text patch

hhhh—two bytes of text patch which are not relocatable

hhhhR—two bytes of text patch which are relocatable.

The bytes of text patches can be grouped in any order but must be separated by commas. The bytes of the text patches must start in column 14. The entire DATA statement can be filled.

The DATA statement can be followed by:

- Another DATA statement
- Another PTF statement
- An END statement

END

END—identifies the control statement. It means the module has been patched and may be placed back in the library. The END statement can be followed by an HDR statement or a /* statement.

/*

/*—indicates patching is complete and end-of-job occurs.

Procedures for Running \$SGFIX

1. Place statements in primary hopper of MFCU and press MFCU START.
2. Be sure correct pack is mounted and perform the IPL procedure from the desired pack. Initial program loading is performed and, when complete, begins reading cards from the MFCU. If the system has DPF, EJ is displayed in both message display units when initial program loading is complete. Press appropriate HALT/RESET key to continue.

When the modules have been patched, EJ is displayed in the message display unit.

MAINTENANCE PROGRAM PROCEDURE

The FE Maintenance Program (\$SGPTF) is used to apply program temporary fixes (PTFs) to programs residing in the object library. Before PTFs can be applied, the system to be updated must reside on disk unit F1 (this system must contain the Library Maintenance Program, \$MAINT), and the programs or modules to be fixed must have been copied to F1 if they are not there already. PTFs can then be applied as follows.

Disk System

1. Mount the user Distribution Disk Cartridge (PID pack) on R1. This pack contains the FE Maintenance Program (\$SGPTF).
2. Perform the IPL procedure from disk unit F1. Include the date statement at IPL time.
3. Load the PTF deck into the MFCU. The PTF deck contains the information to be inserted into or to replace a module. The PTF deck also includes the // LOAD and // RUN cards necessary to apply the information to the module(s) on F1.
4. Use the Library Maintenance Program (\$MAINT) to copy the updated module(s) from disk unit F1 to other user packs as required on R1.

Considerations

The operator should examine the comment cards in each PTF deck before he actually applies the PTF. This will ensure that the PTF applied is the one required to fix his program.

Model 6

1. Mount the PTF pack on R1. The PTF pack contains the FE Maintenance Program (\$SGPTF), PTFs, and the procedures necessary to apply each PTF on the pack. The PTF pack also contains a directory to the PTFs on the pack and a procedure to list the directory (PTFLST).
2. Perform the IPL procedure from disk unit F1.
3. Call the procedure that applies the required PTF:

<i>Prompt</i>	<i>Response</i>
READY	CALL
CALL NAME	PTFnnn
UNIT	R1
MODIFY	RUN

The 3-digit number nnn identifies the specific PTF requested and is obtained from the cover letter that arrives with each PTF pack. The procedure PTFnnn applies the PTF to disk unit F1.

4. Use the Library Maintenance Program (\$MAINT) to copy updated module(s) from disk unit F1 to user packs mounted on R1.

Considerations

Although a cover letter is distributed with each PTF pack to identify the contents of the pack, operators are advised to verify the contents of the pack and to examine requested PTFs before any PTF is actually applied.

A listing of the PTF pack directory can be obtained by performing steps 1 and 2 above and then calling the procedure PTFLST:

<i>Prompt</i>	<i>Response</i>
READY	CALL
CALL NAME	PTFLST
UNIT	R1
MODIFY	RUN

Each directory entry will include a PTF order number, related program number, the version and modification level of the related program, related Authorized Program Analysis Report (APAR) number, and the name of the procedure that applies the PTF (PTFnnn).



A printout of any PTF on the PTF pack can then be obtained:

<i>Prompt</i>	<i>Response</i>
READY	LOAD
LOAD NAME	\$MAINT
UNIT	F1 (ENTER- key)
MODIFY	RUN
ENTER '/' CONTROL STATEMENT	//COPY NAME-PTFnnn (taken from the directory) LIBRARY-P, TO-PRINT, UNIT-R1
ENTER '/' CONTROL STATEMENT	//END

The operator may now examine the comments included in the PTF to ensure that the PTF applied is the one required to fix his program.

SYSTEM GENERATION '1 HALT PROCEDURE

If a '1 Halt (converted to AB D1 345 on Model 6) appears on the console lights during System Generation, maintenance personnel can locate the specific cause of the problem by locating a 2-byte save area within the current System Generation phase. This 2-byte save area contains an error code that is saved but not displayed to the user. The meaning of the error code can be located under one of the following displayed subidentifiers:

GM'1 Halt

The maintenance personnel can locate the 2-byte save area at the label \$HALT located within the module \$SGROC. Because of the variety of routines that can cause this halt, the following data areas are being supplied at this point in an effort to aid in the understanding of the error description. See Figure B-10.

<i>Data Area</i>	<i>Contents</i>
OUTBUA	Contains the address of the output buffer in which the subfields are built.
INBUF	Contains the address of the input buffer in which the subfields are built.
COMON	Contains the address of the buffer that contains the system configuration statement.
BUFF	Contains the address of the buffer that contains the current source library LINK record that is being processed.
VSTST	Contains the address of the start of the variable symbol table.
PROTSV	Contains the address of the output buffer in which the MODEL record and the subfields are built.

Program	Description	Code	Error Description
\$SGROC	System Generation Pre-processor	AI	Invalid AGO statement (period missing)
		BE	The output buffer has overflowed while analyzing a model or AIF statement. The buffer length for the AIF statement is 60 bytes. The buffer length for the model statement is 96 bytes.
		BN	The AGO sequence name is longer than six bytes.
		EM	A MEND statement has been found in the middle of the source library entry.
		IA	Invalid AIF statement
		IG	Invalid GBLB/LCLB statement (variable symbol missing, comma missing).
		IM	An end-of-file condition was encountered, while reading the source library entry, before a MEND statement was found.
		IP	Invalid prototype statement
		IS	The first byte of a variable symbol name is not a character.
		IT	The keyword TABLE is misspelled or the variable symbol (columns 14-96) is missing on a TABLE statement.
		IV	The value being analyzed has overflowed the output buffer length of 60 bytes.
		MM	The source library entry statements are not in the expected logical order.
		MN	The severity code is missing on an MNOTE statement (columns 14-15).
		ND	There were no TABLE definition statements given for a previously read TABLE statement.
		NM	The LINK statement is missing for this source library entry.
		SB	The SETB statement has an invalid variable symbol name or the value in column 14 is not 0 or 1.
		SM	A request was made to find a variable symbol name that was not previously defined.
TF	The variable symbol table is full, this name/value cannot be added.		
TV	The TABLE definition statement has a character string that was not enclosed in quotes (columns 14-96).		

Figure B-10. Halt Codes for the System Generation Mainline (\$SGROC) GM'1 Halt

B

GG'1 Halt

The maintenance personnel can locate the two-byte save area at the label ERRSAV located within the module \$SGEN1. See Figure B-11.

Program	Description	Code	Error Description
\$SGEN1	System Generation Phase Two	BT	Column 14 of a configuration statement from the MACOUT file did not contain any of the following identifiers: 'N', 'M', 'F', or 'X'. These identifiers specify the type of operation to be performed in order to build data onto the configuration record. 'N' — SBN 'M' — MVI 'F' — SBF 'X' — Convert data to hex and 'SBN'
		IS	Column 15 or 18 of a configuration statement received from the MACOUT file does not contain commas.
		IE	The data entry, located in columns 19-22 on a configuration statement, is longer than eight hexadecimal digits (four bytes).
		NR	The librarian module \$\$SYSP has indicated that there is no room in the source library to build or continue to build the \$SGOPT entries.
		IF	The completion code received from \$\$SYSP is invalid.
		BF	\$\$SYSP indicates that it received an invalid flag byte from \$SGEN1.
		ID	The hex displacement, located in columns 16-17 on a configuration statement, is greater than decimal value of 255.
		IN	The data entry, located in columns 19-22 on a configuration statement, does not fall within the specified hexadecimal limits: F0-F9 inclusive C1-C6 inclusive
		BI	There was an IOS error indicated while writing the configuration record onto F1.
		EX	The end of extent was reached while writing the Linkage Editor input file \$WORK.

Figure B-11. Halt Codes for the System Generation—Phase Two (\$SGEN1) GG'1 Halt

GS'1 Halt

The maintenance personnel can locate the 2-byte save area at the label ERRSAV within \$SGENB. See Figure B-12.

Program	Description	Code	Error Description
\$SGENB	System Generation, Phase One	BC	An invalid completion code was received from the SWA Read/Write routine.
		PEM	The // END statement is missing from the procedure on the distribution disk cartridge.

Figure B-12. Halt Codes for System Generation—Phase One (\$SGENB) GS'1 Halt

SYSTEM HALTS AND THE MODULES THAT CAN ISSUE THEM

Figure B-13 is provided to help determine which modules issue which halts. The figure contains a list of halts, the modules that initiate the halts listed, and a brief description of the reason for the halt being issued. The halts listed are only those halts which can be issued by the modules discussed in this manual.

Note: The halts shown are converted by the Model 6 Halt/Syslog Transient (\$\$STOK) to display the Model 6 halt. See *Part 6. Transients and Scheduler Support* for a description of \$\$STOK.

Halt ID	Module Issuing Halt	Brief Description of Halt
00	\$\$STOH—System Halt Transient	Invalid option selected for a Halt
50	\$\$SPVR—Supervisor	A bad track exists which cannot be assigned
70	\$\$RDRN—// RUN Control Card Processor \$\$RDCL—// CALL Control Card Processor	Partition statement read but not a DPF system
71	\$\$RDML—Reader/Interpreter	Partition statement read but not a DPF system
72	\$\$RDML—Reader/Interpreter	Statement Identifier missing
73	\$\$RDPN—// PARTITION Control Card Processor	Partition statement given in Program Level 2 or Level 2 was active when Partition statement given in Level 1
74	\$\$RDML—Reader Interpreter	/& between a // LOAD or // CALL and a // RUN
75	\$\$RDML—Reader Interpreter \$\$RDIM—// IMAGE Control Card Processor	Extraneous statement in columns 1 and 2 or * in column 1
76	\$\$RDML—Reader/Interpreter \$\$RDIM—// IMAGE Control Card Processor	Extraneous statement in columns 1 and 2 or * in column 1
77	\$\$RDML—Reader/Interpreter	Invalid statement identifier
78	\$\$RDML—Reader/Interpreter \$\$RDIM—// IMAGE Control Card Processor \$\$RDPN—// PARTITION Control Card Processor	Invalid statement identifier
79	\$\$RDML—Reader/Interpreter	Continuation of OCL statement expected but not received
7A	\$\$RDCL—// CALL Control Card Processor \$\$RDLD—// LOAD Control Card Processor	A second load or call statement has been read prior to reading a run statement
7C	\$\$RDCM—// COMPILE Control Card Processor	Compile statement not found between jobs (between // LOAD or // CALL and // RUN)
7E	\$\$RDDT—// DATE Control Card Processor	Date statement found between jobs
7F	\$\$RDFL—// FILE Control Card Processor \$\$RDML—Reader/Interpreter \$\$STHB—Halt Syslog Parameter Build Routine	File statement found between jobs
7H	\$\$RDSW—// SWITCH Control Card Processor	Switch statement found between jobs
7J	\$\$RDRR—// READER Control Card Processor	Reader statement found between LOAD or CALL and RUN statement
7P	\$\$RDIM—// IMAGE Control Card Processor	New print chain expected
7U	\$\$RDRN—// RUN Control Card Processor	Run statement not found after Load or Call statement

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 1 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
7-	\$\$RDPN--// PARTITION Control Card Processor	Too many override statements for procedure
7b	\$\$RDRN--// RUN Control Card Processor	Disk I/O error while reading from source library
80	\$\$RDCL--// CALL Control Card Processor \$\$RDLD--// LOAD Control Card Processor	No system date given at IPL time
81	\$\$RDLD--// LOAD Control Card Processor	Error in Load statement coding (no program given, no unit given, etc.)
83	\$\$RDLD--// LOAD Control Card Processor	Load * statement invalid in program level 2
84	\$\$RDCL--// CALL Control Card Processor	Error in Call statement coding (no parameters found, no unit specified)
85	\$\$RDSW--// SWITCH Statement Control Card Processor	A second Switch statement found
86	\$\$RDSW--// SWITCH Statement Control Card Processor	Invalid parameter in Switch statement
88	\$\$RDML--Reader/Interpreter \$\$SPVR--Supervisor	Procedure not found in source library on specified unit
89	\$\$RDDT--// DATE Control Card Processor	System date already specified
8A	\$\$RDDT--// DATE Control Card Processor	Invalid date or date specified incorrectly on Date statement
8C	\$\$RDDT--// DATE Control Card Processor	Second date statement found
8E	\$\$RDDT--// DATE Control Card Processor	Date parameter missing or invalid
8H	\$\$RDCL--// CALL Control Card Processor	First statement in procedure is not a load statement
8J	\$\$RDRR--// READER Control Card Processor	Invalid parameter in Reader statement
8L	\$\$RDRR--// READER Control Card Processor	Desired system input device not available
8P	\$\$RDRR--// READER Control Card Processor	Desired system input device was not defined as part of system at system generation
8U	\$\$RDFL--// FILE Control Card Processor	
8Y	\$\$RDLG--// LOG Control Card Processor	No logging can be done, Log turned off in other program level
8-	\$\$RDLG--// LOG Control Card Processor	Logging is requested but cannot be done because log was turned off by other program level, DPF system only
90	\$\$RDPS--// PAUSE Control Card Processor	Pause statement read
91	\$\$RDPS--// PAUSE Control Card Processor	Pause statement read
92	\$\$RDCM--// COMPILE Control Card Processor	Compile statement already received for this job
93	\$\$RDML--Reader/Interpreter	Error in Compile statement, abnormal end of statement or format or punctuation error

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 2 of 10)



Halt ID	Module Issuing Halt	Brief Description of Halt
94	\$\$RDML—Reader/Interpreter	Error in Compile statement, duplicate or invalid keyword
95	\$\$RDML—Reader/Interpreter \$\$RDCM—// COMPILE Control Card Processor	Error in Compile statement
96	\$\$STHB—Halt/Syslog parameter Build \$\$RDCL-\$\$RDS3	An OCL error but the system can not resolve it
97	\$\$RDLG—// LOG Control Card Processor	Error in Log statement
98	\$\$RDLG—// LOG Control Card Processor	Error in Log statement
99	\$\$RDLG—// LOG Control Card Processor	Error in Log statement
9A	\$\$RDIM—// IMAGE Control Card Processor \$\$RDLG—// LOG Control Card Processor	Indicated action on the last OCL statement read will be ignored due to previous OCL statement errors found
9E	\$\$RDLG—// LOG Control Card Processor	Desired logging device is allocated to other program level
9F	\$\$RDLG—// LOG Control Card Processor	Desired logging device allocated to other program level
9H	\$\$RDLG—// LOG Control Card Processor	Desired logging device allocated to other program level
9J	\$\$RDFM—// FORMS Control Card Processor \$\$RDML—Reader/Interpreter Mainline	Error in Forms statement
9L	\$\$RDFM—// FORMS Control Card \$\$RDML—Reader/Interpreter Mainline	Error in Forms statement
9P	\$\$RDFM—//FORMS Control Card Processor \$\$RDML—Reader/Interpreter Mainline	Error in Forms statement
9U	\$\$RDIM—// IMAGE Control Card Processor	Image statement already accepted by other program level
9Y	\$\$RDLG—// LOG Control Card Processor	Desired logging device not defined as part of the system at System Generation
9	\$\$RDLG—// LOG Control Card Processor	Other program level has already read in the Forms statement, DPF only
9-	\$\$RDFM—// FORMS Control Card Processor	Desired logging device was not defined as part of the system at System Generation
9'	\$\$RDLG—// LOG Control Card Processor	Desired logging device was not defined as part of the system at System Generation
A0	\$\$RDML—Reader/Interpreter Mainline \$\$RDHK—HIKEY Parameter Scan Routine	Format or punctuation error in File statement
A1	\$\$RDML—Reader/Interpreter Mainline	Keyword error in File statement

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 3 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
A2	\$\$\$RDM L—Reader/Interpreter Mainline \$\$\$RDFL—// FILE Control Card Processor	Parameter error in File statement
A3	\$\$\$RDM K—File Diagnostics and Merge Keyword Parameters	Missing parameter on File statement
A4	\$\$\$RDM K—File Diagnostics and Merge Keyword Parameters	Both tracks and records given in File statement
A5	\$\$\$RDM K—File Diagnostics and Merge Keyword Parameters	File statement for Multivolume file which must be on-line doesn't have an equal number of units and packs specified
A6	\$\$\$RDM K—File Diagnostics and Merge Keyword Parameters	Error in File statement for Multivolume files
A7	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statement
A8	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statements on disk
A9	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statements on disk
AA	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statements on disk
AC	\$\$\$RDM I—// IMAGE Control Card Processor	Invalid hexadecimal character in chain image
AE	\$\$\$RDM I—// IMAGE Control Card Processor	Invalid hexadecimal character in chain image
AF	\$\$\$RDM I—// IMAGE Control Card Processor	Invalid hexadecimal character in chain image
AH	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statement
AJ	\$\$\$RDM I—// IMAGE Control Card Processor	Error in Image statement
AL	\$\$\$RDP N—// PARTITION Control Card Statement	Error in Partition statement
AP	\$\$\$RDR N—// RUN Control Card Processor	Errors have been found in the OCL for job
AU	\$\$\$RDP N—// PARTITION Control Card Processor	Error in Partition statement
A	\$\$\$RDP N—// PARTITION Control Card Processor	Error in Partition statement
A-	\$\$\$RDP N—// PARTITION Control Card Processor	Number of packs specified in File statements exceeds 52
A'	\$\$\$RDR N—// RUN Control Card Processor \$\$\$RDFL—// FILE Control Card Processor	No space left in system work area; too many file statements in this job
EJ	\$\$\$SPVR—Supervisor	End of Job

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 4 of 10)



Halt ID	Module Issuing Halt	Brief Description of Halt
FL	\$\$STR2—Resource Allocation (Phase Two)	Printer keyboard or data entry keyboard is not supported or data entry keyboard is allocated to the other program level
FP	\$\$STR2—Resource Allocation (Phase Two)	Program is attempting to allocate the printer and it has been allocated for the other program level
FU	\$\$STR2—Resource Allocation (Phase Two)	Program attempting to allocate the MFCU and it has been allocated to other level
FY	\$\$STR2—Resource Allocation (Phase Two)	Program is attempting to allocate device that is not supported by the system
F	\$\$INPS—Initiator Program Setup (Phase One Routine) \$\$INA1—Initiator Allocate (Phase One Routine)	Cannot execute a utility program on unit because unit is being used on other level, a file has been allocated to this unit on other level or file has been allocated to this unit prior to unity request
HA	\$\$INP2—Initiator Program Setup (Phase Two Routine)	An attempt is being made to compile a source program and a // FILE statement with name \$SOURCE has been read, space specified too small to contain the source program
HC	\$\$INPS—Initiator Program Setup (Phase One Routine)	Requested program not found on specified fixed disk
HE	\$\$STAI—Allocate Initiator \$\$INAT—Allocate Terminator \$\$INA3—Initiator Allocate (Phase Three Routine) \$\$SPFN—FIND Transient \$\$SPEJ—End of Job Transient \$\$INDS—Initiator Disk File Processor (Phase Four) \$\$TMEO—Terminator End of Job (Phase Four) \$\$SSSC—Scheduler Work Area (Read/Write) \$\$SSPT—Scheduler Work Area (Put) \$\$SSGT—Scheduler Work Area (GET) \$\$STR1—Resource Allocation (Phase One) \$\$STRO—Rollout (Phase One) \$\$SSVT—Volume Table of Contents (Read/Write) \$\$STRU—Rollout (Phase Two) \$\$STRT—Rollout (Phase Three) \$\$STRI—Rollout (Phase 1) \$\$STRN—Rollin (Phase Two)	A permanent disk error has occurred and the system may not be able to continue proper functioning

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 5 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
	\$LINKB—PASS1 Root and Initialization Phase	
	\$LINKG—PASS2 Root Phase	
	\$LINKJ—PASS2 Directory Build Phase	
	\$LINKK—Error and Overlay Fetch Table— Print Phase	
	\$LINKN—Punch Output Processor	
HF	\$\$INPS—Initiator Program Setup (Phase One Routine)	A // COMPILE statement was read and is not required
HH	\$\$INPS—Initiator Program Setup (Phase One Routine)	Requested program cannot be found on specified removable pack, the removable pack is being used for some other function and cannot be removed at this time
HJ	\$\$INPS—Initiator Program Setup (Phase One Routine)	Requested program not found on specified removable pack
HL	\$\$INPS—Initiator Program Setup (Phase One Routine)	An inquiry request has been allowed, but the program to be executed is the wrong type and cannot be executed
HP	\$\$INPS—Initiator Program Setup (Phase One Routine)	Insufficient storage to run requested program
HU	\$\$INP2—Initiator Program Setup (Phase Two Routine)	Requested source program not found on fixed disk specified by Compile statement
H	\$\$INP2—Initiator Program Setup (Phase Two Routine)	Requested source program not found on removable pack specified by the // COMPILE statement
JA	\$\$INPS—Initiator Program Setup (Phase One Routine)	Attempting to load a program that requires or allows inquiry while a program that also requires or allows inquiry is in execution in the other program level
JC	\$\$INPS—Initiator Program Setup (Phase One Routine)	The program cannot be run because the program must be run in the dedicated mode and the other program level is being used or the program in the other level must run dedicated
JE	\$\$INPS—Initiator Program Setup (Phase One Routine)	A dedicated program being loaded into Level 1 requires more core storage than is available
JF	\$\$\$RRT—Rollout (Phase Three) Setup (Phase One Routine)	An attempt is being made to load an inquiry program but the keyboard is being used by other program level as system input device
JH	\$\$INPS—Initiator Program Setup (Phase One Routine)	An attempt is being made to start a program which allows interrupts, in program level 2; DPF only
JJ	\$\$INPS—Initiator Program Setup (Phase One)	Attempt to load an object program failed, no object library on specified unit
JL	\$\$TMSB—Terminator Standby Routine	There is not enough available storage to start the selected program, DPF only
JP	\$\$TMSB—Terminator Standby Routine	A system input device has been selected to load a program, by the system input device already assigned to other program level
JU	\$\$TMIH—Terminator Interrupt Handler	A cancel request has been made from the interrupt key, DPF only
	@\$SPV2—Dedicated Supervisor	

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 6 of 10)



Halt ID	Module Issuing Halt	Brief Description of Halt
JY	\$\$STRT—Rollout (Phase Three)	An inquiry request has been received and accepted
J	\$\$SMSI—Terminator SYSIN Initialization	Selected dual program control switch position does not have the device desired assigned to that position or device has not been requested, DPF only
J'	\$\$STRN—Rollin (Phase Two)	Inquiry program complete but device required by the interrupted program is not ready for use, pack that was being used by the interrupted program is not mounted or an MFCU error could have occurred when reading the first card from either hopper
LA	\$\$STAI—Allocate Initiator	Number of files is too large to be allocated in the available storage
LC	\$\$STAI—Allocate Initiator	File allocation hampered by file information defined outside the program levels partition or in last 1024 bytes of storage
LE	\$\$INA1—Initiator Allocate (Phase One Routine)	No // FILE statement was read for a file used by the current program
LH	\$\$INA1—Initiator Allocate (Phase One Routine)	No location and space was specified by the file statement for an output file
LJ	\$\$INA1—Initiator Allocate (Phase One Routine)	Existing permanent file is referenced as output file, instead of temporary or scratch
LL	\$\$INA1—Initiator Allocate (Phase One Routine)	An existing temporary file is being allocated as an output file
LP	\$\$INA1—Initiator Allocate (Phase One Routine)	Allocation of new file hampered since same file exists on the referenced volume in a different location
LU	\$\$INA1—Initiator Allocate (Phase One Routine)	File use hampered by pack name not agreeing with // FILE 'Pack-' specification
LY	\$\$INA3—Initiator Allocate (Phase Three Routine)	Allocation of new disk file hampered by insufficient space on pack or specified track location not available
P1	\$\$STEP—System Output Printer Error Recovery	Printer carriage check
P3	\$\$STEP—System Output Printer Error Recovery	Paper forms on the printer have jammed in print area
P5	\$\$STEP—System Output Printer Error Recovery	Printer synchronous check
P6	\$\$STEP—System Output Printer Error Recovery	Printer incrementer failure check
P7	\$\$STEP—System Output Printer Error Recovery	Printer thermal check
P8	\$\$STEP—System Output Printer Error Recovery	Printer print check
PC	\$\$SPD4—Disk Storage Dump Routine \$\$SPVR—Supervisor	One or more characters in print line cannot be printed. They are not part of the chain image in the communications area
PE	\$\$RDIM—// IMAGE Control Card Processor	Chain check

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 7 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
PF	\$\$RDIM—// IMAGE Control Card Processor	Chain check
PH	\$\$INA1—Initiator Allocate (Phase One Routine)	Allocating a new file has been hampered because of insufficient space on the specified unit
PJ	\$\$TMEJ—Terminator End of Job (Phase Two)	A pack is to be remounted
PU	\$\$INDC—Initiator Disk File Processor (Phase Three)	Duplicate file names have been found in the // FILE statements
PY	\$\$INA1—Initiator Allocate (Phase One Routine)	// FILE statement specifying only one track when trying to allocate a disk file as an indexed file
P	\$\$INDC—Initiator Disk File Processor (Phase Three)	The other program level is creating a file with the same label and date
P'	\$\$INA1—Initiator Allocate (Phase One Routine)	There have been more than four requests to allocate temporary disk space during the job
UA	\$\$INPD—Initiator Disk File Processor (Phase One) \$\$INDF—Initiator Disk File Processor (Phase Two)	Halt because trying to create an off-line multivolume scratch file or trying to access a scratch file as an off-line multivolume file or allowing RETAIN-S off-line multivolume files
UE	\$\$INDF—Initiator Disk File Processor (Phase Two)	Pack name does not match // FILE statement, pack specification or the pack may not be initialized when referencing a disk file
UF	\$\$INDF—Initiator Disk File Processor (Phase Two)	A disk file has been referenced by name and date, but file name cannot be found
UL	\$\$INDF—Initiator Disk File Processor (Phase Two)	File cannot be found and no tracks or records parameters are given
UJ	\$\$INDF—Initiator Disk File Processor (Phase Two)	A reference disk file has been found on the specified unit but the date given is not the same and no space was specified
UP	\$\$INDF—Initiator Disk File Processor (Phase Two)	An S file is being referenced with Retain-P on the file statement or a P file is being referenced with Retain-S
UU	\$\$INDF—Initiator Disk File Processor (Phase Two)	A disk file has been referenced but the requested pack is not available
U	\$\$INDF—Initiator Disk File Processor (Phase Two)	An existing file has been referenced with the same date specified in the // FILE statement but the location and/or tracks/records specified do not match file specifications
U'	\$\$INDF—Initiator Disk File Processor (Phase Two)	VTOC is full
UC	\$\$INDF—Initiator Disk File Processor (Phase Two)	The specified active file cannot be found in the list of scratch files
UH	\$\$INDF—Initiator Disk File Processor (Phase Two)	An attempt is being made to create a new multivolume file that will have the same label on disk as an existing multivolume file
0	\$\$SPVR—Supervisor \$\$IPLNIP—Nucleus Initializations Program IPLBOOT—Initial Program Load Bootstrap Program	The disk file write switch is off, a disk error occurred during IPL or IPL from a pack that does not contain a system



Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 8 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
1	\$\$SPVR—Supervisor	Permanent disk error when the system was trying to read, loading a program outside of core or system encountered an unexpected condition
8	\$\$STIC—System Input Device (Console) \$\$STIM—System Input Device (MFCU/1442)	Device designated as system input for this job is allocated to the other program level
9	\$\$STIM—System Input Device (MFCU/1442)	Device designated as system input device has an error condition
A	\$\$STIM—System Input Device (MFCU/1442)	Number of characters entered from the printer keyboard is incorrect
'0	\$\$GEN—System Generation \$\$GROC—Mainline—Phase One \$\$GNOT—Note Processor \$\$GMN1—Mainline—Phase Two \$\$GOUT—Model Statement Output File/Build Routine	System generation errors
'1	\$\$GEN1—System Generation \$\$GROC—Mainline—Phase One \$\$GRED—Read Pre-processor Statement \$\$GGSY—Get/Build Variable Symbol \$\$GVST—Variable Symbol Table—Find/Build \$\$GAIF—AIF Statement Processor \$\$GGBL—Global and Local Statement Processor \$\$GROT—Prototype Statement Processor \$\$GSTM—LINK Statement Processor \$\$GMN2—Mainline—Phase Three \$\$GTBL—TABLE Statement Processor \$\$GSUB—Subfield Analyzer \$\$GDEF—TABLE Definition Processor \$\$GCSB—Character String Check/Build \$\$GNCB—Number—Check/Build \$\$GBVA—Variable Symbol Table—Find/Build \$\$GMN3—Mainline—Phase Four \$\$GAGO—AGO Statement Processor \$\$GSET—SETB Statement Processor	System generation errors

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 9 of 10)

Halt ID	Module Issuing Halt	Brief Description of Halt
	\$SGNTT—MNOTE Statement	
	\$SGOUT—Model Statement Output File/Build Routine	
	\$SGOPR—Keyword Operand Processor	
'4	\$SGEN1—System Generation	End of extent reached while building the output file on F1 during system generation
'L	\$LINKB—PASS1 Root and Initialization Phase	System module missing
	\$LINKG—PASS2 Root Phase	
	\$LINKK—Error and Overlay Fetch Table— Print Phase	
'P	\$LINKG—PASS2 Root Phase	No object library
'U	\$LINKN—Punch Output Processor	Compiler output error
	\$LINKJ—PASS2 Directory Build Phase	
	\$LINKG—PASS2 Root Phase	
	\$LINKK—Error and Overlay Fetch Table Print Phase	System or program errors

Figure B-13. Table of System Halts, Including Disk System Scheduler (Part 10 of 10)

ERROR MESSAGES

The following charts identify the error messages which may be used by phases in the conversational Reader/Interpreter.

Phase Message No.	\$\$RBIN	\$\$RBCO	\$\$RBMV	\$\$RBIP	\$\$RBRN	\$\$RBM1	\$\$RBM2	\$\$RBLG	\$\$RBFM	\$\$RBBC	\$\$RBCC	\$\$RBHI
00	•											
01	•				•					•		
02	•											
03	•									•		
04		•										
05	•									•		
06		•										
07	•									•		
08					•							
09											•	
10												
11	•				•							
12	•											
13	•											
14		•										•
15					•							
16	•				•							
17	•											
18	•											
19	•											
20	•		•									
21	•											
22		•										
23					•							
24	•											
25	•											
26				•			•					
27				•				•				
28								•				

Phase Message No.	\$\$RBIN	\$\$RBCO	\$\$RBMV	\$\$RBIP	\$\$RBRN	\$\$RBM1	\$\$RBM2	\$\$RBLG	\$\$RBFM	\$\$RBBC	\$\$RBCC	\$\$RBHI
29			•									
30							•					
31							•					
32			•	•								
33												•
34					•							
35	•											•
36	•		•		•		•	•		•		•
37	•											
38	•											
39												
40												
41			•					•				
42		•										
43						•						
44				•			•					
45							•					
46												
47	•						•					
48				•								
49					•							
50				•								
51					•							
52					•							
53				•								
54												•
55												•
56					•							•

Phase Message No.	\$\$RBCL	\$\$RBCP	\$\$RBCD	\$\$RBCK
00				
01				
02				
03				
04				
05				
06				
07				
08				
09				
10			•	
11				
12				
13			•	
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				

Phase Message No.	\$\$RBCL	\$\$RBCP	\$\$RBCD	\$\$RBCK
29				
30				
31				
32				
33		•		•
34				
35				
36		•	•	•
37				
38				
39	•			
40	•	•		•
41				•
42				
43				
44				
45				
46	•			•
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				

B

Message 00 – No Program Name Given
Message 01 – No Unit Given
Message 02 – Invalid Program Name Specified
Message 03 – Invalid Unit Specified
Message 04 – Program Not Found on Specified Unit
Message 05 – No Procedure Name Given
Message 06 – Source Not Found on Specified Unit
Message 07 – Invalid Procedure Name Specified
Message 08 – Multi-Volume File Responses Not in 1-1 Ratio
Message 09 – Procedure Not Found on Specified Unit
Message 10 – Invalid Switch Settings
Message 11 – No Source Name Given
Message 12 – Invalid Source Name Specified
Message 13 – Invalid Date Specified
Message 14 – Too Many Responses to a Multi-Volume File Keyword
Message 15 – No File Name Given
Message 16 – No Pack Given
Message 17 – Invalid File Name Specified
Message 18 – Invalid Label Specified
Message 19 – Invalid Pack Specified
Message 20 – Invalid Retain Designation Specified
Message 21 – Invalid Tracks Specified
Message 22 – Maximum File Statements Entered
Message 23 – Both Tracks and Records Specified
Message 24 – Invalid Records Specified
Message 25 – Invalid Location Specified
Message 26 – Device Not Supported
Message 27 – Invalid Device
Message 28 – Invalid Number of Lines Specified
Message 29 – Invalid Request
Message 30 – Invalid Statement Number
Message 31 – Too Many Utility Control Statements in Procedure—Job Canceled
Message 32 – Run Out of Space in Scheduler Work Area
Message 33 – Response Required—Delayed Response in Called Procedure
Message 34 – Too Many Multi-Volume File Units Specified
Message 35 – Delayed Response (?) Not Allowed
Message 36 – Job Canceled
Message 37 – Multi-Volume File Not Valid This Statement
Message 38 – Enter Minus (-) Not Allowed
Message 39 – Errors in Procedure—Job Canceled
Message 40 – Errors in OCL Statement
Message 41 – Errors in Response
Message 42 – Duplicate Procedure Name in Library
Message 43 – Duplicate Procedure Deleted
Message 44 – Invalid Keyword
Message 45 – Too Many Utility Control Statements Entered
Message 46 – Permanent Disk Error
Message 47 – Run Out of Space in Procedure Library—Job Canceled
Message 48 – Invalid System Date Specified
Message 49 – Duplicate Keyword
Message 50 – Response Required
Message 51 – Too Many Packs/Hikeys Specified
Message 52 – Duplicate Multi-Volume File Unit Specified
Message 53 – Invalid Response During Inquiry
Message 54 – Invalid Hikey Specified
Message 55 – Invalid Hikey Length Specified
Message 56 – Hikeys Out of Sequence

TABLE OF LIGHT IDENTIFIERS

Model 6

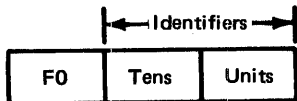
Disk System

Character Displayed	Hex Value
0	X'6F'
1	X'03'
2	X'76'
3	X'57'
4	X'1B'
5	X'5D'
6	X'7D'
7	X'07'
8	X'7F'
9	X'5F'
A	X'3F'

1	X'6C'
E	X'7C'
F	X'3C'
H	X'3B'
U	X'63'
L	X'68'
P	X'3E'^1
U	X'6B'
U	X'5B' DASH
	X'02' QUOTE
	X'00' BLANK
-	X'10' DASH

Character Displayed Hex Value

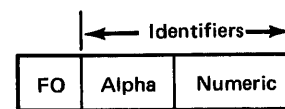
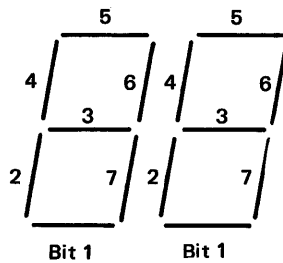
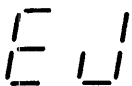
A	X'40'
B	X'20'
C	X'10'
D	X'08'
1	X'40'
2	X'20'
3	X'10'
4	X'08'
5	X'04'



Example:

X'F0'	X'7C'	X'63'
-------	-------	-------

Displays



Example

X'F0'	X'78'	X'7C'
-------	-------	-------

Displays

ABCD12345

Figure B-14. Table of Light Identifiers

B

READER/INTERPRETER CR96 ERROR PROCEDURE

The user will receive a CR96 error code if the system cannot diagnose an error during the processing of an OCL statement. The user can locate the invalid error number at the label ERROR# located within the Reader/Interpreter work area (RDIWA). Just below ERROR# is the label RTNID that contains a 2-byte module identifier. This identifier indicates the module that passed the invalid error number. The label ML1090 located within the Reader/Interpreter Mainline (\$\$RDML) contains the address of the next sequential instruction (NSI) of the caller.



APPENDIX C.

ABBREVIATIONS AND TERMINOLOGY



C

The following abbreviations and terms are used throughout this publication.

ARR	Address Recall Register	IAR	Instruction Address Register
BSAREA	BASIC Area	INBUF	Input Buffer
BSBUFF	BASIC Buffer	INTER Mode	The mode the scheduler is in after a // RUN statement is processed and before the next // LOAD statement is processed.
BSFILE	BASIC File	INTRA Mode	The mode the scheduler is in after a // LOAD statement and before the next // RUN statement is processed.
BUFFR1	Buffer One	IOB	Input/Output Block
BUFR1	Buffer One	IOS	Input/Output Supervisor
BUFR2	Buffer Two	IPL	Initial Program Load
BUFR3	Buffer Three	IPL Mode	The mode the scheduler is in when it receives control from IPLNIP and before the first // LOAD statement is processed.
C/S/D	Cylinder/Sector/Displacement	IPLNIP	Initial Program Load Nucleus Initialization Program
CALCM	Call Communication Area	LSR	Local Storage Register
COMMON	Communication Area	MACOUT	Phase One Output Work File
CONFIG	Configuration Record	O. Module	Load Module
DISPLIO	Display Input/Output Area	OBJLIB	Object Library
DPF	Dual Programming Feature	OBR	Outboard Recording
DTF	Define The File	OUTBUF	Output Buffer
ESL	External Symbol List	OVLAYT	Overlay Table
ESLTAB	External Symbol List Table	PLCA	Program Level Communication Area (synonymous with program level communication region)
ERRTAB	Error Table		
F1, F2	Fixed Disk—One, Fixed Disk—Two		
GBLTAB	Global Table		

Pre-processor statements	Source input statements to the System Generation Pre-processor.	SCT	Scheduler Control Table
PTF	Program Temporary Fix	SDR	Statistical Data Recording
R1, R2	Removable disk one, Removable disk two	SPVROT	Save Buffer
R. Module	Object Module	SWA	Scheduler Work Area
RDPARM	Pre-processor common area	TXTWRK	Text Work Area
RIB	Request Indicator Byte	VARTAB	Variable Symbol Table
RLD	Relocation Directory	VOLLAB	Volume Label
RLDWRK	RLD Work Area	VTOC	Volume Table Of Contents
SCA	System Communication Area (synonymous with system communication region)	XR1, XR2	Index Register One, Index Register Two





APPENDIX D.

FLOWCHARTING TECHNIQUES

D

Flowcharts are identified in this publication in the following manner:

- A flowchart consisting of a single page is identified by a unique pair of letters.

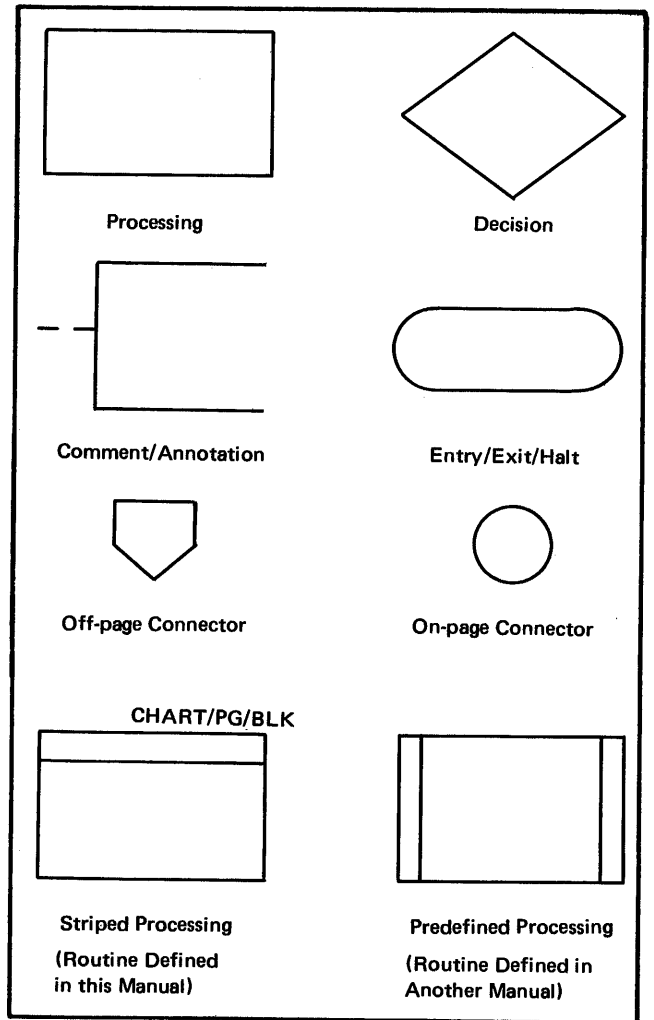
Example: AA, AB, AC

- If a flowchart consists of multiple pages, each page is identified by the same pair of letters, but each page has a unique number.

Example: First page, CA-01
 Second page, CA-02
 Third page, CA-03, etc.

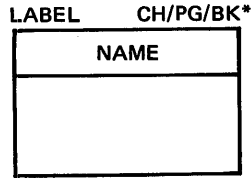
- Each part has been assigned two sets of flowchart identifying letters. Only after the first set has been completely used, i.e., AA-AZ, will the second set be used.

The flowchart symbols used in this PLM are:



1. The striped processing block indicates entry to a module or routine which is flowcharted and/or described in this logic manual.

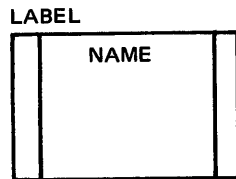
Example:



*CH/PG/BK indicates the flowchart, page, and block identification where the module or routine is flowcharted. If it is not flowcharted, a note gives the location of the description of that routine.

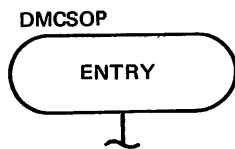
2. Predefined processing indicates a module or routine flowcharted in the *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual, SY21-0512*.

Example:



3. The label in the upper lefthand corner, just above a flowcharting symbol, is the entry point in the listing for that part of the program.

Example:



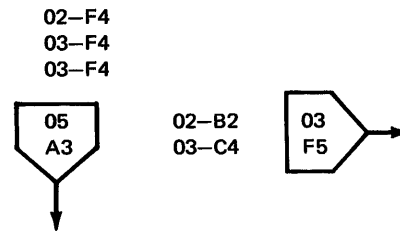
4. Off-page connectors are used to reference between different page of the same chart ID. Off-page connectors leaving a page contain the page number and block number of their destination.

Example:



Off-page connectors contain the page and block number of their origin. If the entry point referenced by the off-page connector is referenced from more than one origin, all origins are given. The origins are listed in alphameric order with the last reference contained within the block.

Example:



5. On-page connectors contain the location of a block on the same page. On-page connectors always contain the location of the destination block.

- \$\$FTRC (RIB Fetch/Trace routine)
 - description B-17
 - flowchart B-24
- \$\$INAT (Allocate Terminator)
 - description 6-24
 - directory 6-111
 - flowchart 6-96
 - main storage map 6-24
 - method of operation diagram 5-9
- \$\$INAI (Initiator Allocate–Phase One Routine)
 - description 5-46
 - directory 5-183
 - flowchart 5-157
 - main storage map 5-46
 - method of operation diagram 5-8
- \$\$INA2 (Initiator Allocate–Phase Two Routine)
 - description 5-47
 - directory 5-183
 - flowchart 5-160
 - main storage map 5-47
 - method of operation diagram 5-8
- \$\$INA3 (Initiator Allocate–Phase Three Routine)
 - description 5-48
 - directory 5-183
 - flowchart 5-161
 - main storage map 5-48
 - method of operation diagram 5-8
- \$\$INA4 (Initiator Allocate–Phase Four Routine)
 - description 5-49
 - directory 5-183
 - flowchart 5-162
 - main storage map 5-49
 - method of operation diagram 5-8
- \$\$INDC (Initiator Disk File Processor–Phase Three)
 - description 5-51
 - directory 5-183
 - flowchart 5-170
 - main storage map 5-51
 - method of operation diagram 5-8
- \$\$INDF (Initiator Disk File Processor–Phase Two)
 - description 5-50
 - directory 5-183
 - flowchart 5-167
 - main storage map 5-50
 - method of operation diagram 5-8
- \$\$INDS (Initiator Disk File Processor–Phase Four)
 - description 5-51
 - directory 5-183
 - flowchart 5-171
 - main storage map 5-51
 - method of operation diagram 5-8
- \$\$INMS (Initiator Allocate–Support Routines)
 - description 5-53
 - directory 5-183
 - main storage map 5-53
 - method of operation diagram 5-8
- \$\$INPD (Initiator Disk File Processor–Phase One)
 - description 5-50
 - directory 5-183
 - flowchart 5-163
 - main storage map 5-50
 - method of operation diagram 5-8
- \$\$INPS (Initiator Program Setup–Phase One Routine)
 - description 5-52
 - directory 5-183
 - flowchart 5-172
 - main storage map 5-52
 - method of operation diagram 5-8
- \$\$INP2 (Initiator Program Setup–Phase Two Routine)
 - description 5-52
 - directory 5-183
 - flowchart 5-175
 - main storage map 5-52
 - method of operation diagram 5-8
- \$\$ODNP (Transient Resolver No-op Routine)
 - description 6-26
 - main storage map 6-5
- \$\$OXRF (Transient Resolver)
 - description 6-23
 - directory 6-111
 - flowchart 6-95
 - main storage map 6-23
- \$\$RBBC (Build–Chained Cycle)
 - description 5-20
 - directory 5-183
 - flowchart 5-79
 - main storage map 5-20
 - method of operation diagram 5-8
- \$\$RBCC (Call Control)
 - description 5-21
 - directory 5-183
 - flowchart 5-82
 - main storage map 5-21
 - method of operation diagram 5-8
- \$\$RBCD (Data Type Statement Processor)
 - description 5-24
 - directory 5-183
 - flowchart 5-89
 - main storage map 5-24
 - method of operation diagram 5-8
- \$\$RBCK (Keyword Type Statement Processor)
 - description 5-25
 - directory 5-183
 - flowchart 5-90
 - main storage map 5-25
 - method of operation diagram 5-8
- \$\$RBCL (Call Cycle)
 - description 5-22
 - directory 5-183
 - flowchart 5-84
 - main storage map 5-22
 - method of operation diagram 5-8

\$\$RBCO (Scheduler Control)
 description 5-19
 directory 5-183
 flowchart 5-74
 main storage map 5-19
 method of operation diagram 5-8

\$\$RBCP (Positional Statement Processor)
 description 5-23
 directory 5-183
 flowchart 5-86
 main storage map 5-23
 method of operation diagram 5-8

\$\$RBDS (Data/Switch)
 description 5-29
 directory 5-183
 flowchart 5-113
 main storage map 5-29
 method of operation diagram 5-8

\$\$RBDT (Data Scan)
 description 5-30
 directory 5-183
 flowchart 5-115
 main storage map 5-31
 method of operation diagram 5-8

\$\$RBFM (FORMS)
 description 5-31
 directory 5-183
 flowchart 5-117
 main storage map 5-31
 method of operation diagram 5-8

\$\$RBHI (HIKEY)
 description 5-32
 directory 5-183
 flowchart 5-120
 main storage map 5-32
 method of operation diagram 5-8

\$\$RBIN (Interaction Routine)
 description 5-29
 directory 5-183
 flowchart 5-111
 main storage map 5-29
 method of operation 5-8

\$\$RBIP (Conversational Scheduler Initializer)
 description 5-18
 directory 5-183
 flowchart 5-73
 main storage map 5-18
 method of operation 5-8

\$\$RBLG (Log)
 description 5-30
 directory 5-183
 flowchart 5-116
 main storage map 5-31
 method of operation 5-8

\$\$RBLI (Model 6 Library Interaction Routine)
 description 6-29
 directory 6-111
 flowchart 6-110
 main storage map 6-5

\$\$RBLP (Model 6 Logical Put Routine)
 description 6-27
 directory 6-111
 flowchart 6-103
 main storage map 6-5

\$\$RBM1 (Procedure Library Scheduler Interface)
 description 5-27
 directory 5-183
 flowchart 5-98
 main storage map 5-27
 method of operation 5-8

\$\$RBM2 (Utility OCL Processor)
 description 5-27
 directory 5-183
 flowchart 5-104
 main storage map 5-27
 method of operation diagram 5-8

\$\$RBRN (Run Processor)
 description 5-28
 directory 5-183
 flowchart 5-106
 main storage map 5-28
 method of operation diagram 5-8

\$\$RBSI (Model 6 Syntax Checker Interface)
 description 6-29
 directory 6-111
 flowchart 6-108
 main storage map 6-5

\$\$RBSW (Switch Control Statement Processor)
 description 5-30
 directory 5-183
 flowchart 5-114
 main storage map 5-30
 method of operation diagram 5-8

\$\$RBSX (Model 6 Syntax Checker)
 description 6-28
 directory 6-111
 flowchart 6-104
 main storage map 6-28

\$\$RDCL (// CALL Control Card Processor)
 description 5-43
 directory 5-183
 flowchart 5-153
 main storage map 5-43
 method of operation diagram 5-8

\$\$RDCM (// COMPILE Control Card Processor)
 description 5-38
 directory 5-183
 flowchart 5-143
 main storage map 5-38
 method of operation diagram 5-8

\$\$RDDT (// DATE Control Card Processor)
 description 5-36
 directory 5-183
 flowchart 5-139
 main storage map 5-36
 method of operation diagram 5-8

\$\$RDFL (// FILE Control Card Processor)
 description 5-35
 directory 5-183
 flowchart 5-136
 main storage map 5-35
 method of operation diagram 5-8

\$\$RDFM (// FORMS Control Card Processor)
 description 5-38
 directory 5-183
 flowchart 5-146
 main storage map 5-38
 method of operation diagram 5-8

\$\$RDHK (Hikey Parameter Scan Routine)
 description 6-27
 directory 6-111
 flowchart 6-101
 main storage map 6-5

\$\$RDHN (// HALT/NOHALT Control Card Processor)
 description 5-40
 directory 5-183
 flowchart 5-150
 main storage map 5-40
 method of operation diagram 5-8

\$\$RDIM (// IMAGE Control Card Processor)
 description 5-37
 directory 5-183
 flowchart 5-141
 main storage map 5-37
 method of operation diagram 5-8

\$\$RDLD (// LOAD Control Card Processor)
 description 5-33
 directory 5-183
 flowchart 5-133
 main storage map 5-33
 method of operation diagram 5-8

\$\$RDLG (// LOG Control Card Processor)
 description 5-39
 directory 5-183
 flowchart 5-147
 main storage map 5-39
 method of operation diagram 5-8

\$\$RDML (Reader/Interpreter Mainline)
 description 5-33
 directory 5-183
 flowchart 5-125
 main storage map 5-33
 method of operation diagram 5-8

\$\$RDMK (File Diagnostics and Merge Keyword Parameters)
 description 5-45
 directory 5-183
 flowchart 5-156
 main storage map 5-45
 method of operation diagram 5-8

\$\$RDPN (// PARTITION Control Card Processor)
 description 5-42
 directory 5-183
 flowchart 5-152
 main storage map 5-42
 method of operation diagram 5-8

\$\$RDPS (// PAUSE Control Card Processor)
 description 5-40
 directory 5-183
 flowchart 5-149
 main storage map 5-40
 method of operation diagram 5-8

\$\$RDRN (// RUN Control Card Processor)
 description 5-34
 directory 5-183
 flowchart 5-135
 main storage map 5-34
 method of operation diagram 5-8

\$\$RDRR (// READER Control Card Processor)
 description 5-41
 directory 5-183
 flowchart 5-151
 main storage map 5-41
 method of operation diagram 5-8

\$\$RDSW (// SWITCH Control Card Processor)
 description 5-37
 directory 5-183
 flowchart 5-140
 main storage map 5-37
 method of operation diagram 5-8

\$\$RDS1 (Keyword Syntax Scan Routine)
 description 6-25
 directory 6-111
 flowchart 6-98
 main storage map 6-25

\$\$RDS3 (Positional Parameter Syntax Scan Routine)
 description 5-44
 directory 5-183
 flowchart 5-154
 main storage map 5-44
 method of operation diagram 5-8

\$\$SLSE (Model 6 Halt/Syslog Error Message Routine)
 description 6-10
 directory 6-111
 flowchart 6-58
 main storage map 6-5

\$\$SLS1 (Model 6 Halt/Syslog Error Message Routine)
 description 6-10
 directory 6-111
 flowchart 6-59
 main storage map 6-5

\$\$SPC1 (System List—Print/Punch, Disk System)
 description 6-17
 directory 6-111
 flowchart 6-79
 main storage map 6-17

\$\$SPC2 (System List—Punch Routine, Model 6)
 description 6-18
 directory 6-111
 flowchart 6-80
 main storage map 6-18

\$\$SPDF (DTF Dump Routine, Disk System)
 description B-11
 flowchart B-20

\$\$SPD0 (Dump Format Routine, Model 6)
 description B-13
 flowchart B-23
 main storage map B-13

\$\$SPD1 (Dump Format Routine, Disk System)
 description B-13
 flowchart B-23
 main storage map B-13

\$\$SPD2 (Disk Storage Dump Routine, Model 6)
 address/data switch settings B-12
 description B-12
 flowchart B-21
 main storage map B-13

\$\$SPD3 (DTF Dump Routine, Model 6)
 description B-11
 flowchart B-20

\$\$\$PD4 (Disk Storage Dump Routine, Disk System)
 address/data switch settings B-12
 description B-12
 flowchart B-21
 main storage map B-13

\$\$SPEJ (End of Job Transient)
 description 6-21
 directory 6-111
 main storage map 6-5

\$\$\$PFN (FIND Transient)
 description 6-19
 directory 6-111
 flowchart 6-83
 main storage map 6-5

\$\$\$PVR (see supervisor)

\$\$\$SGT (Scheduler Work Area-GET)
 description 6-11
 directory 6-111
 flowchart 6-60
 main storage map 6-5
 parameter list 6-11

\$\$\$SPT (Scheduler Work Area-PUT)
 description 6-12
 directory 6-111
 flowchart 6-61
 main storage map 6-5
 parameter list 6-12

\$\$\$SSC (Scheduler Work Area-Read/Write)
 description 6-13
 directory 6-111
 flowchart 6-62
 main storage map 6-5
 parameter list 6-13

\$\$\$SVT (Volume Table of Contents-Read/Write)
 description 6-14
 directory 6-111
 flowchart 6-63
 main storage map 6-5
 parameter list 6-14

\$\$STAI (Allocate Initiator)
 description 6-20
 directory 6-111
 flowchart 6-84
 main storage map 6-20
 method of operation diagram 5-9

\$\$STEP (System Output-Printer Error Recovery)
 description 6-7
 directory 6-111
 flowchart 6-41
 main storage map 6-5

\$\$STF7 (HIKEY Transient)
 description 6-26
 directory 6-111
 flowchart 6-99
 main storage map 6-5
 parameter list 6-21

\$\$STHB (Halt/Syslog Parameter Build Routine)
 description 6-25
 directory 6-111
 flowchart 6-97
 main storage amp 6-25

\$\$STH0 (Halt/Syslog Parameter Table One) A-32
\$\$STH1 (Halt/Syslog Parameter Table Two) A-33
\$\$STH2 (Halt/Syslog Parameter Table Three) A-34
\$\$STH3 (Halt/Syslog Parameter Table Four) A-35
\$\$STH4 (Halt/Syslog Parameter Table Five) A-36

\$\$STIC (System Input Device-Console)
 description 6-5
 directory 6-111
 flowchart 6-30
 main storage map 6-5
 parameter list 6-6

\$\$STID (Model 6 System Input Device-Data Recorder)
 description 6-6
 directory 6-111
 flowchart 6-35
 main storage map 6-5
 parameter list 6-6

\$\$STIM (System Input Device-MFCU/1442)
 description 6-5
 directory 6-111
 flowchart 6-31
 main storage map 6-5
 parameter list 6-6

\$\$STIP (Model 6 System Input Device-Keyboard/Printer)
 description 6-6
 directory 6-111
 flowchart 6-33
 main storage map 6-5
 parameter list 6-6

\$\$STIS (Model 6 System Input Device-Keyboard/Printer)
 description 6-6
 directory 6-111
 flowchart 6-33
 main storage map 6-5
 parameter list 6-6

\$\$STIT (Model 6 System Input Device-CRT)
 description 6-7
 directory 6-111
 flowchart 6-37
 main storage map 6-5
 parameter list 6-6

\$\$STIX (Model 6 System Input Device-CRT)
 description 6-7
 directory 6-111
 flowchart 6-37
 main storage map 6-5
 parameter list 6-6

\$\$STNQ (System Queue Routine)
 description 6-24
 directory 6-111
 main storage map 6-5

\$\$STOC (Halt/Syslog-Console)
 description 6-7
 directory 6-111
 flowchart 6-43
 main storage map 6-5

\$\$STOH (System HatI Transient)
 description 6-8
 directory 6-111
 flowchart 6-46
 main storage map 6-5

\$\$STOK (Halt/Syslog—Halt Conversion Routine)
 description 6-8
 directory 6-111
 flowchart 6-49
 main storage map 6-5
 tables 6-9

\$\$STOM (Model 6 Halt/Syslog Routine—Matrix Printer)
 description 6-10
 directory 6-111
 flowchart 6-51
 main storage map 6-5

\$\$STON (Model 6 Halt/Syslog Routines—Output Only, Matrix Printer)
 description 6-10
 directory 6-111
 flowchart 6-53
 main storage map 6-5

\$\$STOP (Halt/Syslog—Printer)
 description 6-7
 directory 6-111
 flowchart 6-44
 main storage map 6-5

\$\$STOS (Model 6 Halt/Syslog Hatl Routine—Keyboard)
 description 6-10
 directory 6-111
 flowchart 6-50
 main storage map 6-5

\$\$STOT (Model 6 Halt/Syslog Halt Routine—CRT)
 description 6-10
 directory 6-111
 flowchart 6-55
 main storage map 6-5

\$\$STOX (Model 6 Halt/Syslog Routine—CRT On and Off Transient)
 description 6-10
 directory 6-111
 flowchart 6-57
 main storage map 6-5

\$\$STP6 (Model 6 Printer Syslog Error Recovery Procedures Routine)
 description 6-8
 directory 6-111
 flowchart 6-47
 main storage map 6-5

\$\$STRI (Rollin—Phase One)
 description 6-15
 directory 6-111
 flowchart 6-66
 main storage map 6-5

\$\$STRN (Rollin—Phase Two)
 description 6-15
 directory 6-111
 flowchart 6-68
 main storage map 6-5

\$\$STRO (Rollout—Phase One)
 description 6-15
 directory 6-111
 flowchart 6-70
 main storage map 6-5

\$\$STRT (Rollout—Phase Three)
 description 6-15
 directory 6-111
 flowchart 6-74
 main storage map 6-5

\$\$STRU (Rollout—Phase Two)
 description 6-15
 directory 6-111
 flowchart 6-72
 main storage map 6-5

\$\$STR1 (Resource Allocation—Phase One)
 description 6-23
 directory 6-111
 flowchart 6-92
 main storage map 6-5

\$\$STR2 (Resource Allocation—Phase Two)
 description 6-23
 directory 6-111
 flowchart
 Disk System 6-93
 Model 6 6-94
 main storage map 6-5

\$\$SYLA (LOAD * Mainline)
 description 6-21
 directory 6-111
 flowchart 6-85
 main storage map 6-21

\$\$SYP1 (System List—Printer, Disk System)
 description 6-16
 directory 6-111
 flowchart 6-76
 main storage amp 6-16

\$\$SYP2 (System List—Print, Model 6)
 description 6-16
 directory 6-111
 flowchart 6-77
 main storage map 6-16

\$\$SYSG (Source Library—GET)
 description 6-19
 directory 6-111
 flowchart 6-81
 main storage map 6-5

\$\$SYSP (Copy Main Storage to Source Library)
 description 6-22
 directory 6-111
 flowchart 6-89
 main storage map 6-22

\$\$TMDS (Terminator Duplicate Key Scan)
 description 5-17
 directory 5-183
 flowchart 5-71
 main storage map 5-17
 method of operation diagram 5-8

\$\$TMEJ (Terminator End of Job—Phase Two)
 description 5-13
 directory 5-183
 flowchart 5-58
 main storage map 5-13
 method of operation diagram 5-8

\$\$TME0 (Terminator End of Job—Phase One)
 description 5-13
 directory 5-183
 flowchart 5-57
 main storage map 5-13
 method of operation diagram 5-8

\$\$TMIP (Terminator Initial Program Load)
 description 5-12
 directory 5-183
 flowchart 5-55
 main storage map 5-12
 method of operaton diagram 5-8

- \$\$TMRI (Terminator Re-Initialization)**
 - description 5-14
 - directory 5-183
 - flowchart 5-60
 - main storage map 5-14
 - method of operation diagram 5-8
- \$\$TMSB (Terminator Standby Routine)**
 - description 5-15
 - directory 5-183
 - flowchart 5-63
 - main storage map 5-15
 - method of operation diagram 5-8
- \$\$TMSI (Terminator SYSIN Initialization)**
 - description 5-16
 - directory 5-183
 - flowchart 5-64
 - main storage map 5-16
 - method of operation diagram 5-8
- \$\$TMSK (Terminator Key Sort)**
 - description 5-17
 - directory 5-183
 - flowchart 5-65
 - main storage map 5-17
 - method of operation diagram 5-8
- \$\$TMST (End of Job—Disk Status)**
 - description 6-22
 - directory 6-111
 - flowchart 6-88
 - main storage map 6-5
 - parameter list 6-22
- \$\$@SPV1 (see Dedicated Supervisor)**
- \$\$@SPV2 (see DPF Supervisor)**
- \$LINKB (PASS1 Root and Initialization Phase)**
 - description 7-13
 - directory 7-55
 - flowchart 7-19
 - main storage map 7-15
 - method of operation description 7-8
 - method of operation diagram 7-9
- \$LINKC (PASS1 INCLD and ENTRY Control Record and AUTOLINK Processor)**
 - description 7-14
 - directory 7-55
 - flowchart 7-24
 - main storage map 7-15
 - method of operation description 7-8
 - method of operation diagram 7-9
- \$LINKD (PASS1 TEXT-RLD and END Record Processor)**
 - description 7-14
 - directory 7-55
 - flowchart 7-27
 - main storage map 7-15
 - method of operation description 7-10
 - method of operation diagram 7-9
- \$LINKE (PASS1 External Symbol List Control Record Processor)**
 - description 7-15
 - directory 7-55
 - flowchart 7-29
 - main storage map 7-15
 - method of operation description 7-11
 - method of operation diagram 7-9
- \$LINKF (PASS1 PHASE Record Processor)**
 - description 7-15
 - directory 7-55
 - flowchart 7-32
 - main storage map 7-15
 - method of operation description 7-11
 - method of operation diagram 7-9
- \$LINKG (PASS2 Root Phase)**
 - description 7-16
 - directory 7-55
 - flowchart 7-35
 - main storage map 7-16
 - method of operation description 7-12
 - method of operation diagram 7-9
- \$LINKH (PASS2 TEXT-RLD Conversion Phase)**
 - description 7-17
 - directory 7-55
 - flowchart 7-38
 - main storage map 7-17
 - method of operation diagram 7-9
- \$LINKJ (PASS2 Directory Build Phase)**
 - description 7-17
 - directory 7-55
 - flowchart 7-42
 - main storage map 7-17
 - method of operation diagram 7-9
- \$LINKK (Error and Overlay Fetch Table—Print Phase)**
 - description 7-18
 - directory 7-55
 - flowchart 7-46
 - main storage map 7-18
 - method of operation diagram 7-9
- \$LINKM (External Symbol List Table—Print Phase)**
 - description 7-18
 - directory 7-55
 - flowchart 7-48
 - main storage map 7-18
 - method of operation description 7-12
 - method of operation diagram 7-9
- \$LINKN (Punch Output Processor)**
 - description 7-18
 - directory 7-55
 - flowchart 7-50
 - main storage map 7-18
 - method of operation description 7-12
 - method of operation diagram 7-9
- \$\$SGAGO (AGO Statement Processor)**
 - description 8-23
 - directory 8-59
 - flowchart 8-53
 - main storage map 8-19
 - method of operation diagram 8-12
- \$\$SGAIF (AIF Statement Processor)**
 - description 8-21
 - directory 8-59
 - flowchart 8-46
 - main storage map 8-23
 - method of operation diagram 8-12
- \$\$GBSV (Co-Resident System Analyzer, Model 6)**
 - description 8-13
 - direction 8-59
 - flowchart 8-26
 - main storage map 8-13
 - method of operation diagram 8-12

\$SGCSB (Character String—Check/Build Routine)
 description 8-20
 directory 8-59
 flowchart 8-40
 main storage map 8-16, 8-17
 method of operation diagram 1-14 through 8-12

\$SGDEF (Table Definition Processor)
 description 8-22
 directory 8-59
 flowchart 8-52
 main storage map 8-23
 method of operation diagram 8-12

\$SGEN (System Generation—Phase One, Disk System)
 description 8-14
 directory 8-59
 flowchart 8-27
 main storage map 8-14
 method of operation diagram 8-12

\$SGENB (System Generation—Phase One, Model 6)
 description 8-14
 directory 8-59
 flowchart 8-28
 main storage map 8-14
 method of operation diagram 8-12

\$SGEN1 (System Generation—Phase Two)
 description 8-15
 directory 8-59
 flowchart 8-29
 main storage map 8-13
 method of operation diagram 8-12

\$SGFIX (Program Temporary Patch Program)
 description B-17
 flowchart B-25
 procedures B-27

\$SGGBL (GBLB/LCLB Statement Processor)
 description 8-24
 directory 8-59
 flowchart 8-56
 main storage map 8-16
 method of operation diagram 8-12

\$SGGSY (GET Variable Symbol Routine)
 description 8-18
 directory 8-59
 flowchart 8-36
 main storage map
 phase two 8-16
 phase three 8-17
 phase four 8-19
 method of operation diagram 8-12

\$SGGVA (Variable Symbol Table—Value Build Routine)
 description 8-19
 directory 8-59
 flowchart 8-38
 main storage map
 phase two 8-16
 phase three 8-17
 phase four 8-19
 method of operation diagram 8-12

\$SGIVP (System Verification Message Routine)
 description 8-24
 directory 8-59

\$SGKBD (Keyboard Selection Routine, Model 6)
 description 8-13
 directory 8-59
 flowchart 8-25
 main storage map 8-13
 method of operation diagram 8-12

\$SGMNT (System Maintenance)
 description 9-6

\$SGMN1 (Pre-Processor Mainline—Phase Two)
 description 8-16
 directory 8-59
 flowchart 8-33
 main storage map 8-16
 method of operation diagram 8-12

\$SGMN2 (Pre-Processor Mainline—Phase Three)
 description 8-17
 directory 8-59
 flowchart 8-34
 main storage map 8-17
 method of operation diagram 8-12

\$SGMN3 (Pre-Processor Mainline—Phase Four)
 description 8-18
 directory 8-59
 flowchart 8-35
 main storage map 8-19
 method of operation diagram 8-12

\$SGMST (MODEL Statement Build Routine)
 description 8-21
 directory 8-59
 flowchart 8-48
 main storage map 8-23
 method of operation diagram 8-12

\$SGNCB (Number—Check/Build Routine)
 description 8-19
 directory 8-59
 flowchart 8-39
 main storage map 8-16, 8-17
 method of operation diagram 8-12

\$SGNTT (MNOTE Statement Processor)
 description 8-23
 directory 8-59
 flowchart 8-54
 main storage map 8-19
 method of operation diagram 8-12

\$SGNOT (System Generation Error Note Processor)
 description 8-15
 directory 8-59
 flowchart 8-31
 main storage map 8-14
 method of operation diagram 8-12

\$SGOPR (Keyword Operand Processor)
 description 8-21
 directory 8-59
 flowchart 8-44
 main storage map 8-16
 method of operation diagram 8-12

\$SGOUT (MODEL Statement Output Processor)
 description 8-22
 directory 8-59
 flowchart 8-50
 main storage map 8-23
 method of operation diagram 8-12

\$SGPRI (IPL Bootstrap Modify, Model 6)
 description 8-24
 directory 8-59
 flowchart 8-57

\$SGPTF (FE Maintenance Program)
 description 9-5
 flowchart 9-7
 main storage map 9-5

\$SGPTG (PTF Logging Program)
 description 9-6
 flowchart 9-9
 main storage map 9-6

\$SGRED (Read Pre-Processor Statement Routine)
 description 8-20
 directory 8-59
 flowchart 8-41
 main storage map
 phase two 8-16
 phase three 8-17
 phase four 8-18
 method of operation diagram 8-12

\$SGROC (Pre-Processor Mainline—Phase One)
 description 8-16
 directory 8-59
 flowchart 8-32
 main storage map 8-16, 8-17
 method of operation diagram 8-12

\$SGROT (PROTOTYPE Statement Processor)
 description 8-20
 directory 8-59
 flowchart 8-43
 main storage map 8-16
 method of operation diagram 8-12

\$SGSET (SETB Statement Processor)
 description 8-24
 directory 8-59
 flowchart 8-55
 main storage map 8-19
 method of operation diagram 8-12

\$SGSTM (LINK Statement Processor)
 description 8-20
 directory 8-59
 flowchart 8-42
 main storage map 8-16
 method of operation diagram 8-12

\$SGSUB (Subfield Analyzer)
 description 8-22
 directory 8-59
 flowchart 8-49
 main storage map 8-19
 method of operation diagram 8-12

\$SGTBL (TABLE Statement Processor)
 description 8-22
 directory 8-59
 flowchart 8-51
 main storage map 8-16
 method of operation diagram 8-12

\$SGVST (Variable Symbol Table—Find/Build Routine)
 description 8-18
 directory 8-59
 flowchart 8-37
 main storage map
 phase two 8-16
 phase three 8-17
 phase four 8-19
 method of operation diagram 8-12

\$WORK Format (Linkage Editor language translator output work area) A-45
 * (Comment Record) 8-8
 /* Control Card Processor (*see* \$LINKC)
 // COMPILE Control Card Processor (*see* \$\$RDCM)
 // DATE Control Card Processor (*see* \$\$RDDT)
 // FILE Control Card Processor (*see* \$\$RDFL)
 // FORMS Control Card Processor (*see* \$\$RDFM)
 // HALT Control Card Processor (*see* \$\$RDHN)
 // IMAGE Control Card Processor (*see* \$\$RDIM)
 // LOAD Control Card Processor (*see* \$\$RDLG)
 // LOG Control Card Processor (*see* \$\$RDLG)
 // NOHALT Control Card Processor (*see* \$\$RDHN)
 // PARTITION Control Card Processor (*see* \$\$RDPN)
 // PAUSE Control Card Processor (*see* \$\$RDPS)
 // READER Control Card Processor (*see* \$\$RDRR)
 // RUN Control Card Processor (*see* \$\$RDRN)
 // SWITCH Control Card Processor (*see* \$\$RDSW)
 '1 HALT Procedure (System Generation) B-28

Abbreviations and Terminology C-1
 AGO record format 8-6
 AGO statement processor (*see* \$SGAGO)
 AIF record format 8-6
 AIF statement processor (*see* \$SGAIF)
 Allocate Initiator (*see* \$\$STAI)
 Allocate Terminator (*see* \$\$INAT)
 ANOP record format 8-6
 AUTOLINK Processor (*see* \$LINKC)

BUFFER (Linkage Editor input/output buffer) A-44
 BUFRF1 Format (Pre-Processor statement input buffer) A-51
 Build-Chained Cycle (*see* \$\$RBBC)

Call Control (*see* \$\$RBCC)
 Call Cycle (*see* \$\$RBCL)
 CDUMPD (Resident Dump Linkage)
 Disk System
 description 2-10
 directory 2-17
 flowchart 2-15
 main storage map 2-9
 procedure B-11
 Model 6
 description 4-8
 directory 4-9
 flowchart 2-15
 main storage map 4-7
 procedure B-11

Chain Image Area A-3
Character String—Check/Build Routine (*see* \$SGCSB)
Comment Record format (*) 8-8
COMMON
 Linkage Editor PASS1 Communication Region A-41
 Linkage Editor PASS2 Communication Region A-42
 System Generation communication region A-51
Communication Regions
 COMMON (Linkage Editor PASS1) A-41
 COMMON (Linkage Editor PASS2) A-42
 COMMON (System Generation) A-51
 NCPL1 (Supervisor System) A-3, A-9
 N1COMN (Supervisor Program Level) A-3, A-11
 RDIWA (Reader/Interpreter Work Area) A-28
 RDPARM (Pre-Processor) A-49
 SWA (Scheduler Work Area) A-19
 SWADR (Pre-Processor Switch Area) A-52
 Volume Label A-19
 VTOC (Volume Table of Contents) A-26
Configurattn Record Format A-16
Control Records for Linkage Editor 7-3
Conversational Reader/Interpreter 5-6
Conversational Scheduler Initializor (*see* \$RBIP)
Convert Records to Tracks (*see* PAITRK)
Copy Main Storage to Source Library (*see* \$SYSP)
Co-Resident System Analyzer (*see* \$GBSV)
current phase identification procedure using:
 console address/data switches B-2
 main storage dump B-2

data area formats (*see* Appendix A)
Data Type Statement Processor (*see* \$RBCD)
Date Scan (*see* \$RBDT)
Date/Switch (*see* \$RBDS)
DCLPAF (Possible Allocation Fit)
 (*see also* \$INMS)
 description 5-54
 directory 5-183
 flowchart 5-182
 main storage map 5-53
Dedicated Supervisor (\$@SPV1), Disk System 2-1
Dedicated Supervisor (\$@SPV1), Model 6 4-1
Dedicated Supervisor General ENTRY/EXIT and Transient
 Area Scheduler (*see* NENTRY)
Determine Size of Index (*see* PAINDX)
determining entrance to Supervisor 2-5
Determining Supervisor Function (*see* RIB)
determining transient or support routine in storage (*see* current
 phase identification procedure)
DFSBN (Set Bits Routine)
 (*see also* \$INMS)
 description 5-54
 directory 5-183
 flowchart 5-181
 main storage map 5-53
Diagnostic Aids (*see* Appendix B)
Disk Storage Dump Routine, Disk System (*see* \$SPD4)
Disk Storage Dump Routine, Model 6 (*see* \$SPD2)
Disk Storage Dump Switch Settings A-11
display light identifiers B-45
display light mask (*see* display light identifiers)

DMPFND (Dump Selection Routine)
 description B-11
 flowchart B-18
DPF Supervisor (\$@SPV2) 3-1
 introduction 3-3
 method of operation 3-5
 program organization 3-7
 directory 3-17
DPF Supervisor General ENTRY/EXIT and Transient Area
 Scheduler (*see* NENTRY)
DTF compressed format 6-20
 (*see also* \$STAI)
DTF Dump Routine, Disk System (*see* \$SPDF)
DTF Dump Routine, Model 6 (*see* \$SPD3)
Dump Format Routine, Disk System (*see* \$SPD1)
Dump Format Routine, Model 6 (*see* \$SPD0)
Dump Selection Routine (*see* DMPFND) A-10
DUMPPG (Main Storage Dump Routine)
 description B-11
 flowchart B-19

END Control Record Processor (*see* \$LINKD)
END input record format (Linkage Editor) 7-5
End of Job—Disk Status (*see* \$TMST)
End of Job Transient (*see* \$SPEJ)
ENTRY Control Record Processor (*see* \$LINKC)
ENTRY control record format 7-4
Error and Overlay Fetch Table—Print Phase (*see* \$LINKK)
Error Table (ERRTAB) A-44
ERRTAB (Linkage Editor Error Table) A-44
ESL (external symbol list input record) 7-5
ESLTAB (External Symbol List Table) A-45
External Symbol List Control Record Processor (*see* \$LINKE)
external symbol list input record format 7-5
external symbol list table (ESLTAB) A-45
External Symbol List Table—Print Phase (*see* \$LINKM)

FE Maintenance Program (*see* \$GPTF)
FE maintenance program procedure B-27
FETCH only 2-5, 4-5
FETCH with FIND 2-5, 4-5
Fetch Table (RPG Overlay Fetch Table) A-47
Fetch/Trace Procedure, RIB B-16
Fetch/Trace Routine (*see* \$FTRC)
Field Engineering Maintenance Program (*see* \$SGPTF)
File Diagnostics (*see* \$RDMK)
FIND (*see* FETCH with FIND LOAD with FIND)
FIND Transient (*see* \$SPFN)
Flowcharting Techniques C-1
Format One (Scheduler Work Area) A-20
Format Five (Scheduler Work Area) A-21
FORMS (*see* \$RBFM)
Functional Flow Diagram for:
 Dedicated Supervisor, Disk System 2-7
 Dedicated Supervisor, Model 6 4-1
 Linkage Editor 7-9
 Scheduler 5-8
 System Generation 8-12

GBLB Record (Global Record) 8-8
 GET Variable Symbol Routine (*see* \$SGGSY)
 GG'1 Halt B-30
 Global Record Format 8-8
 Global Statement Processor (*see* \$GGBL)
 GM'1 Halt B-28
 GS'1 Halt B-31

HALT/SYSLOG
 Console (*see* \$\$STOC)
 Printer (*see* \$\$STOP)
 Procedures B-3
 Calling Halt/Syslog Routine B-3
 Output Format B-7
 Parameter Formats B-4
 Setting up Halt/Syslog Device for Logging B-3
 System Generation '1 Halt Procedure B-28
 Transient (*see Part 6 in Table of Contents*)
 Halt/Syslog—Console (*see* \$\$STOC)
 Halt/Syslog—Halt Conversion Routine (*see* \$\$STOK)
 Halt/Syslog Parameter Build Routine (Scheduler) (*see* \$\$STHB)
 Halt/Syslog Parameter Table One (*see* \$\$STH0)
 Halt/Syslog Parameter Table Two (*see* \$\$STH1)
 Halt/Syslog Parameter Table Three (*see* \$\$STH2)
 Halt/Syslog Parameter Table Four (*see* \$\$STH3)
 Halt/Syslog Parameter Table Five (*see* \$\$STH4)
 Halt/Syslog—Printer (*see* \$\$STOP)
 HIKEY (*see* \$\$RBHI)
 Hikey Parameter Scan Routine (*see* \$\$RDHK)
 HIKEY Transient (*see* \$\$STF7)
 HPLNUC (Resident Halt Display Routine)
 description 3-8
 directory 3-17
 flowchart 3-13
 main storage map 3-3

Image Area (Chain) A-3
 INBUF Format (Linkage Editor input work area) A-52
 INCLD Control record format 7-4
 INCLD Control Record Processor (*see* \$LINKC)
 Initial Halt/Syslog Parameter Table A-31
 Initial Program Load Bootstrap Program (*see* IPL)
Initiator
 Directory 5-183
 Method of Operation Diagram 5-8
 Method of Operation Narrative 5-7
 Program Organization Section 5-46
 Initiator Allocate—Phase One Routine (*see* \$\$INA1)
 Initiator Allocate—Phase Two Routine (*see* \$\$INA2)
 Initiator Allocate—Phase Three Routine (*see* \$\$INA3)
 Initiator Allocate—Phase Four Routine (*see* \$\$INA4)
 Initiator Allocate—Support Routines (*see* \$\$INMS)
 Initiator Disk File Processor—Phase One (*see* \$\$INPD)
 Initiator Disk File Processor—Phase Two (*see* \$\$INDF)
 Initiator Disk File Processor—Phase Three (*see* \$\$INDC)
 Initiator Disk File Processor—Phase Four (*see* \$\$INDS)
 Initiator Program Setup—Phase One Routine (*see* \$\$INPS)
 Initiator Program Setup—Phase Two Routine (*see* \$\$INP2)
 input control records to the Linkage Editor 7-3
 ENTRY Control Record Format 7-4
 INCLD Control Record Format 7-4
 OPTNS Control Record Format 7-4
 PHASE Control Record Format 7-3

Input/Output Buffer (Linkage Editor (BUFFER) A-44
 input source records to the Linkage Editor 7-5
 END Input Record Format 7-5
 External Symbol List Input Record Format 7-5
 TEXT-RLD Input Record Format 7-5
 input work area (INBUF) A-52
 Interaction Routine (*see* \$\$RBIN)
INTLO
 description 3-8
 directory 3-17
 flowchart 3-14
 main storage map 3-3
IPL (Initial Program Load Bootstrap Program)
 Disk System
 description 1-5
 directory 1-9
 Model 6
 description 1-5
 directory 1-9
IPL Bootstrap Modify, Model 6 (*see* \$SGPRI)
IPLNIP (Nucleus Initialization Program)
 Disk System
 description 1-5
 directory 1-9
 flowchart 1-7
 Model 6
 description 1-6
 directory 1-9
 flowchart 1-8

Keyboard Selection Routine, Model 6 (*see* \$SGKBD)
 Keyword Operand Processor (*see* \$SGOPR)
 Keyword prototype record 8-4
 Keyword Type Statement Processor (*see* \$\$RBCK)

Language Translator Output Work Area (*see* \$WORK)
 LCLB Record (Local Record) 8-8
 LINK definition MEND record format 8-4
 LINK definition exit record (MEXIT record) 8-8
 LINK definition header record format 8-4
 LINK Statement Processor (*see* \$GSTM)
Linkage Editor 7-1
 Introduction 7-3
 Method of Operation 7-7
 Program Organization 7-13
 Directory 7-55
 Data Area Format A-41
Linkage Editor Communication Regions
 PASS1 Communication Region A-41
 PASS2 Communication Region A-42
Linkage Editor Error Table (ERRTAB) A-44
Linkage Editor Input/Output Buffer (BUFFER) A-44
Linkage Editor Input Records
 (*see* Input Control Records)
 (*see* Input Source Records)
Linkage Editor Language Translator Output Work Area (\$WORK) A-45
Linkage Editor Output 7-6
Linkage Editor Print Buffer (*see* PRTBUF)
Linkage Editor Relocation Directory (RLD) Work Area (*see* RLDWRK)
Linkage Editor Text Work Area (*see* TXTWRK)
 linkage for storage dump routines B-11
 linkage to and from supervisor (*see* RIB)

LOAD (*see* **LOAD** with **FIND LOAD** only)
LOAD * Mainline (*see* **\$\$SYLA**)
LOAD (O.) Module Release Number (*see* **\$LINKF**)
LOAD Module Size (*see* **\$LINKF**)
LOAD only 2-5, 4-5
LOAD with FIND 2-5, 4-5
Local Record Format (LCLB) 8-8
Local Statement Processor (*see* **\$\$GGBL**)
Local Storage Register Display Procedure B-8
Local Storage Register Display Routine (*see* **LSR**)
Log (*see* **\$\$RBLG**)
LSR (Local Storage Register)
 Disk System
 description 2-10
 display procedure B-8
 directory 2-17
 main storage map 2-9
 Model 6
 description 4-8
 display procedure B-8
 directory 4-9
 main storage map 4-7

Main Storage Dump Routine (*see* **DUMPBG**)
Maintenance 9-1
 FE Maintenance Program (\$SGPTF) 9-5
 FE Maintenance Program Procedure B-27
 PTF Maintenance (\$SGPTG) 9-6
 System Maintenance (\$SGMNT) 9-6
MEND Record Format 8-4
Merge Keyword Parameters Routine (*see* **RDMK**)
Method of Operation for:
 Dedicated Supervisor 2-7
 Linkage Editor 7-9
 Scheduler 5-8
 System Generation 8-12
MEXIT Record Format 8-8
MNOTE Record Format 8-8
MNOTE Statement Processor (*see* **\$\$GNNT**)
MODEL Record Format 8-8
MODEL Statement Output Processor (*see* **\$\$GOUT**)
MODEL Statement Processor (*see* **\$\$GMST**)
Model 6 Halt/Syslog Error Message Routine (*see* **\$\$SLSE** and **\$\$SLS1**)
Model 6 Halt/Syslog Halt Routine—CRT (*see* **\$\$STOT**)
Model 6 Halt/Syslog Halt Routine—Keyboard (*see* **\$\$STOS**)
Model 6 Halt/Syslog Routine—CRT On and Off Transient (*see* **\$\$STOX**)
Model 6 Halt/Syslog Routine—Matrix Printer (*see* **\$\$STOM**)
Model 6 Halt/Syslog Routine—Output Only, Matrix Printer (*see* **\$\$STON**)
Model 6 Library Interaction Routine (*see* **\$\$RBLI**)
Model 6 Logical Put Routine (*see* **\$\$RBLP**)
Model 6 Printer Syslog Error Recovery Procedures Routine (*see* **\$\$STP6**)
Model 6 Syntax Checker (*see* **\$\$RBSX**)
Model 6 Syntax Checker Interface (*see* **\$\$RBSI**)
Model 6 System Input Device—Data Recorder (*see* **\$\$STID**)
Model 6 System Input Device—CRT (*see* **\$\$STIT** and **\$\$STIX**)
Model 6 System Input Device—Keyboard/Printer (*see* **\$\$STIP** and **\$\$STIS**)
Modify (*see* **\$\$RBMV**)

NCENTR (Supervisor Entry Point) A-9
NCPL1 (System Communication Region) A-3, A-9
NENTRY (Dedicated Supervisor General ENTRY/EXIT and Transient Area Scheduler, Disk System)
 description 2-9
 directory 2-17
 flowchart 2-11
 main storage map 2-9
 method of operation 2-7
NENTRY (Dedicated Supervisor General ENTRY/EXIT and Transient Area Scheduler, Model 6)
 description 4-7
 directory 4-9
 flowchart 2-11
 main storage map 4-7
 method of operation diagram 2-7
NENTRY (DPF Supervisor General ENTRY/EXIT and Transient Area Scheduler)
 description 3-7
 directory 3-17
 flowchart 3-9
 main storage map 3-3
 method of operation 3-5
NLOADR (Resident Loader and Relocation Program)
 Disk Dedicated System
 description 2-10
 directory 2-17
 flowchart 2-13
 main storage map 2-9
 method of operation diagram 2-7
 Disk DPF System
 description 3-7
 directory 3-17
 flowchart 3-11
 main storage map 3-3
 Model 6
 description 4-8
 directory 4-9
 flowchart 2-13
 main storage map 4-7
 method of operation diagram 2-7
NREAD (System Disk Read Routine)
 Disk System
 description 2-10
 directory 2-17
 main storage map 2-7
 Model 6
 description 4-8
 directory 4-9
 main storage map 4-7
Nucleus Initialization Program (*see* **IPLNIP**)
Number—Check/Build Routine (*see* **\$\$GNCB**)
N1COMN (Program Level Communication Region) A-3, A-11

 object library directory entry format A-46
 operational flow diagram (*see* **Functional Flow Diagram for:**)
 OPTNS control record format 7-4
 OPTNS Control Record Processor (*see* **\$LINKE**)
 OUTBUF format (Linkage Editor output work area) A-53
 output from the Linkage Editor 7-6
 output work area (OUTBUF) A-53
 Overview of System/3 Disk and Model 6 Programming Systems xi

PAGTRK (Track Conversion)
(see also \$\$INMS)
description 5-54
directory 5-183
flowchart 5-180
main storage map 5-53

PAINDX (Determine Size of Index)
(see also \$\$INMS)
description 5-54
directory 5-183
flowchart 5-178
main storage map 5-53

PAITRK (Convert Records to Tracks)
(see also \$\$INMS)
description 5-53
directory 5-183
flowchart 5-177
main storage map 5-53

PARTIN (Track Apportioning)
(see also \$\$INMS)
description 5-54
directory 5-183
flowchart 5-179
main storage map 5-53

PASS1 communication region A-41
PASS2 communication region A-42
PASS1 Root Phase *(see \$LINKB)*
PASS2 Root Phase *(see \$LINKG)*
PHASE control record format 7-3
PHASE identification procedure using:
console address/data switches B-2
main storage dump B-2
PHASE name *(see \$LINKF)*
PHASE origin address *(see \$LINKF)*
Positional Parameter Syntax Scan Routine *(see \$\$RDS3)*
Positional Statement Processor *(see \$\$RBCP)*
Possible Allocation Fit *(see DCLPAF)*
Pre-Processor Communication Region (RDPARM) A-49
Pre-Processor Mainline—Phase One *(see \$SGROC)*
Pre-Processor Mainline—Phase Two *(see \$SGMN1)*
Pre-Processor Mainline—Phase Three *(see \$SGMN2)*
Pre-Processor Mainline—Phase Four *(see \$SGMN3)*
Pre-Processor Routine *(see Method of Operation for System Generation)*
Pre-Processor Statement Input Buffer (BUFR1) A-51
pre-processor switch area (SWADR) A-52
print buffer for Linkage Editor (PRTBUF) A-44
Procedure Library Scheduler Interface *(see \$\$RBM1)*
program level communication region (NICOMN) A-3, A-11
Program Temporary Fix (PTF) *see:*
Field Engineering Maintenance Program (\$SGPTF)
Program Temporary Patch Program (\$SGFIX)
PTF Logging Program (\$SGPTG)
System Maintenance (\$SGMNT)
Program Temporary Patch Program *(see \$SGFIX)*
PROTOTYPE Statement Processor *(see \$SGROT)*
PRTBUF (Linkage Editor Print Buffer) A-44
PTF Logging Program *(see \$SGPTG)*
Punch Processor *(see \$LINKN)*

RDIWA (reader/interpreter work area) A-28
RDPARM format (pre-processor communication region) A-49
Read Pre-Processor Statement Routine *(see \$SGRED)*
Reader/Interpreter
directory 5-184
method of operation description 5-6
method of operation diagram 5-8
program organization section 5-18, 5-33
Reader/Interpreter Error CR96 procedure B-46
Reader/Interpreter Mainline *(see \$\$RDML)*
Reader/Interpreter Work Area (RDIWA) A-28
Relocation Directory *(see RLD)*
Request Indicator Byte *(see RIB)*
Resident Dump Linkage *(see CDUMPD)*
Resident Halt Display Routine *(see NLOADR)*
Resident Loader and Relocation Program *(see NLOADR)*
Resource Allocation—Phase One *(see \$\$STR1)*
Resource Allocation—Phase Two *(see \$\$STR2)*
RIB (request indicator byte) B-14
RIB fetch/trace procedure B-16
RIB Fetch/Trace Routine (\$\$FTRC) B-17
RLD (relocation directory)
format 7-5
location 7-5
RLDWRK (Linkage Editor relocation directory (RLD) work area) A-45
Rollin—Phase One *(see \$\$STRI)*
Rollin—Phase Two *(see \$\$STRN)*
Rollout—Phase One *(see \$\$STRO)*
Rollout—Phase Two *(see \$\$STRU)*
Rollout—Phase Three *(see \$\$STRT)*
RPG overlay fetch table (entry format) A-47
Run Processor *(see \$\$RBRN)*

Scheduler 5-1
(see also Terminator, Reader/Interpreter, Initiator)
Introduction 5-3
Method of Operation 5-5
Program Organization 5-11
Directory 5-183
Data Area Format A-15
Scheduler Control *(see \$\$RBCO)*
Scheduler Support *(see Transient and Scheduler Support)* 6-1
Scheduler work area format A-19
Scheduler Work Area—Get *(see \$\$\$SGT)*
Scheduler Work Area—Put *(see \$\$\$SPT)*
Scheduler Work Area—Read/Write *(see \$\$\$SSC)*
Set Bits Routine *(see DFSBN)*
SETB record format 8-8
SETB Statement Processor *(see \$SGSET)*
Source Library Director Format A-53
Source Library—Get *(see \$\$\$SYG)*
Source Records for Linkage Editor 7-5
~START *(see Initial Program Load Bootstrap Program)*
statement save area (SVPROT) A-51
storage dump linkage B-11
storage dump routines
(see \$\$\$SPOF, \$\$\$SPD0, \$\$\$SPD1, \$\$\$SPD2, \$\$\$SPD3, \$\$\$SPD4, CDUMPD, DMPFND, DUMPPG)

storage dump selection procedure B-10
 Subfield Analyzer (see \$SGSUB)
 Supervisor *see*:
 Dedicated Supervisor 2-1
 DPF Supervisor 3-1
 Model Supervisor 4-1
 SVPROT format (Statement Save Area) A-51
 SWA (scheduler work area) A-19
 SWADR format (pre-processor switch area) A-52
 Switch Control Statement Processor (see \$RBSW)
 SYSIN Routine
 (see \$STIC, \$STID, \$STIM, \$STIP, \$STIS, \$STIT, \$STIX)
 System Residence Disk Pack 8-9
 system communication region (NCPL1) A-3, A-9
 system configuration statement format 8-5
 System Generation 8-1
 Introduction 8-3
 Method of Operation 8-11
 Program Organization 8-13
 Directory 8-59
 Data Area Format A-49
 system generation communication region (COMMON) A-51
 system generation control records 8-4
 System Generation Error Note Processor (see \$SGNOT)
 System Generation '1 halt procedure B-28
 GM'1
 GM'1
 System Generation—Phase One, Disk System (see \$SGEN)
 System Generation—Phase One, Model 6 (see \$GENB)
 System Generation—Phase Two (see \$SGEN1)
 System Halt Transient (see \$STOH)
 System Input Device—Console (see \$STIC)
 System Input Device—MFCU/1442 (see \$STIM)
 System List—Printer, Disk System (see \$SYP1)
 System List—Print, Model 6 (see \$SYP2)
 System List—Print/Punch, Disk System (see \$SPC1)
 System List—Punch Routine, Model 6 (see \$SPC2)
 System Maintenance (see \$SGMNT)
 System Output—Printer Error Recovery (see \$STEP)
 System Queue Routine (see \$STNQ)
 system requirements for:
 Dedicated Supervisor 2-3
 Linkage Editor 7-6
 Scheduler 5-3
 System Generation 8-10
 System Residence Disk Pack (SYSRES) organization 8-9
 System Verification Message Routine (see \$SGIVP)

 TABLE definition record format 8-6
 TABLE Definition Processor (see \$GDEF)
 TABLE record format 8-6
 TABLE Statement Processor (see \$GTBL)

Terminator
 directory 5-184
 method of operation description 5-6
 method of operation diagram 5-8
 program organization section 5-12
 Terminator Duplicate Key Scan (see \$TMDS)
 Terminator End of Job—Phase One (see \$TMEO)
 Terminator End of Job—Phase Two (see \$TMEJ)
 Terminator Interrupt Handler (see INTLO)
 Terminator Initial Program Load (see \$TMIP)
 Terminator Key Sort (see \$TMSK)
 Terminator Re-Initialization (see \$TMRI)
 Terminator Standby Routine (see \$TMSB)
 Terminator SYSIN Initialization (see \$TMSI)
 TEXT record format (system generation) 8-6
 TEXT-RLD input record format (Linkage Editor) 7-5
 TEXT-RLD Record Processor (see \$LINKD)
 Text work area (TXTWRK) A-45
 Tracing the Transient Queue B-8
 Track Apportioning (see PARTIN)
 Track Conversion (see PAGTRK)
 Transient and Scheduler Support 6-1
 Introduction 6-3
 Program Organization 6-5
 Directory 6-111
 Transient Area Scheduler
 (see Dedicated Supervisor Generatio ENTRY/EXIT and
 Transient Area Scheduler (NENTRY)
 (see DPF Supervisor General ENTRY/EXIT and Transient Area
 Scheduler (NENTRY)
 transient queue
 format 2-6, 4-6
 tracing procedure B-8
 transient requests 2-6, 4-6
 Transient Resolver (see \$OXRF)
 Transient Resolver No-op (see \$ODNP)
 Translate Table (1-9) A-9, A-55 through A-63
 TXTWRK (Linkage Editor Test work area) A-45

 Utility OCL Processor (see \$RBM2)

 variable symbol table (VST) format A-53
 Variable Symbol Table—Find/Build Routine (see \$SGVST)
 Variable Symbol Table—Value Build Routine (see \$SGGVA)
 Volume Label Format A-19
 VST (variable symbol table) A-53
 VTOC (volume table of contents) A-26
 volume table of contents (VTOC) A-26
 Volume Table of Contents—Read/Write (see \$SSVT)

 work area (see data area format)