IBM SYSTEM/3

IBM System/3 Overlay Linkage Editor and Checkpoint/Restart Programs Logic Manual

Program Numbers:

5702-SC1 Model 10 Disk System
Features 6026 and 6027
5705-SC1 Model 12

SY21-0530-1

Second Edition (September 1972)

This is a major revision of, and obsoletes, the previous edition SY21-0530-0 and Technical Newsletter SN21-7643. This new edition applies to revision 07, modification 00 of the IBM System/3 Model 10 Disk System, Program Number 5702-SC1 and to all subsequent revisions and modifications unless otherwise indicated in new editions or Technical Newsletters.

Changes to text and illustrations are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by a bullet at the left of the figure title.

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, GN20-2228, for the editions that are applicable and current.

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

A Reader's Comment Form is at the back of this publication. If there isn't a form, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

PREFACE

This manual is a guide to program listings for people who maintain the Overlay Linkage Editor or Checkpoint/Restart features for the IBM System/3 Model 10 Disk System, the Overlay Linkage Editor feature for the IBM System/3 Model 6, and the Overlay Linkage Editor or Checkpoint/Restart functions for the IBM System/3 Model 12.

Before using this manual, the reader should be familiar with the operating procedures contained in the *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561, and the *IBM System/3 Model 10 Disk System Control Programming Reference Manual,* GC21-7512.

SYSTEM/3 MODEL 8

The System/3 Model 8 is supported by System/3 Model 10 Disk System control programming and program products. The facilities described in this publication for the Model 10 are also applicable to the Model 8, although the Model 8 is not referenced. It should be noted that not all devices and features which are available on the Model 10 are available on the Model 8. Therefore, Model 8 users should be familiar with the contents of *IBM System/3 Model 8 Introduction,* GC21-5114.

RELATED PUBLICATIONS

- *IBM System/3 Card and Disk System Components Reference Manual,* GA21-9103
- *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502
- *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Program Logic Manual,* SY21-0512
- *IBM System/3 Model 10 Disk System Operator's Guide,* GC21-7508
- *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046
- *IBM System/3 Model 12 Data Areas and Diagnostic Aids,* SY21-0045

HOW THIS MANUAL IS ORGANIZED

This manual has two parts: Overlay Linkage Editor and Checkpoint/Restart.

Part I includes:

Introduction: general information about the characteristics of the program, functions of the program, and input descriptions.

Program Organization: an operational diagram, storage maps, and flowcharts.

Data Areas: contents and formats of data areas used by two or more routines.

Diagnostic Aids: various aids that help diagnose problems.

Part II includes:

Introduction: general information about the program.

Program Organization: functional considerations, operational diagrams, storage maps, linkage considerations, and flowcharts for both the Checkpoint and Restart programs.

Data Areas: contents and formats of data areas used by Checkpoint and Restart.

Diagnostic Aids: various aids that help diagnose problems.

CONTENTS

PART I

OVERLAY LINKAGE EDITOR

SECTION 1. INTRODUCTION

The Overlay Linkage Editor enables the user to influence the determination of overlays for his programs. An automatic determination of overlays is also provided.

OPERATING CHARACTERISTICS

The Overlay Linkage Editor can be entered two ways: directly from a language processor (compiler), or as a user-called program. The functions and method of operation are different depending on whether the entry is via the compiler entry or the user entry.

Compiler Entry

When entered directly from a language processor (Figure 1), the Overlay Linkage Editor can perform any or all of the following functions:

1. Catalog an R module into an object library on disk.
2. Punch an R module into cards.
3. Link R modules into an object program and catalog the program into an object library on disk and/or punch it into cards.

Input for Compiler Entry

Input to the Overlay Linkage Editor is in the $WORK file on disk. Each record in $WORK is 64 bytes long (Figure 2). The first record must be the options record; R modules follow the options record.

Each R module consists of External Symbol List (ESL) fields (packed five to a 64-byte S-type record) and text records. An END record follows the R modules. A /* record must be the last record in the compiler output.

Options Record: The options record tells the Overlay Linkage Editor what functions to perform. The options record must be the first record in $WORK. Figure 3 shows the format of the options record.

R module: The R module consists of ESL fields packed into S-type records, text records, and an END record. Each 64-byte S-type record can contain up to five, 12-byte ESL fields. The S-type record must be hex '00's after the ESL fields.

R modules are described in the *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561.

Output From Compiler Entry

Output from the Overlay Linkage Editor is specified by the options record in $WORK. The R module in $WORK can be punched into cards and/or cataloged into the object library. If link edit is specified, an O module is built from the input R module. The O module is then punched into cards and/or cataloged into the object library.

A storage map and cross reference list is printed unless the options card specifies otherwise.

User Entry

The Overlay Linkage Editor can be loaded by using a // LOAD $OLINK OCL statement. The user must supply control statements.

Input for User Entry

Input for the user entry is described in the *IBM System/3 Overlay Linkage Editor Reference Manual,* GC21-7561.

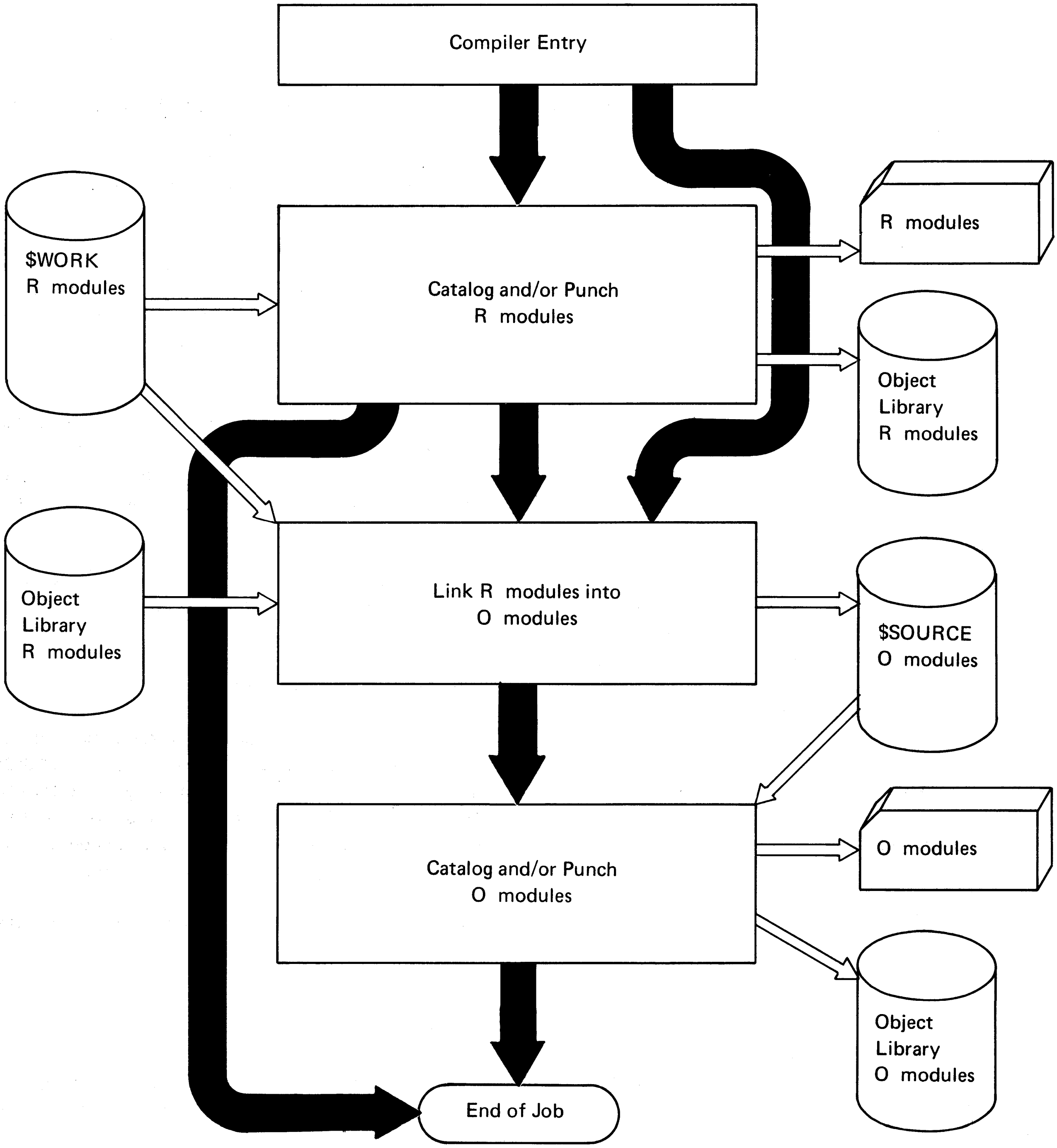


Figure 1. Overview of Overlay Linkage Editor Compiler Entry

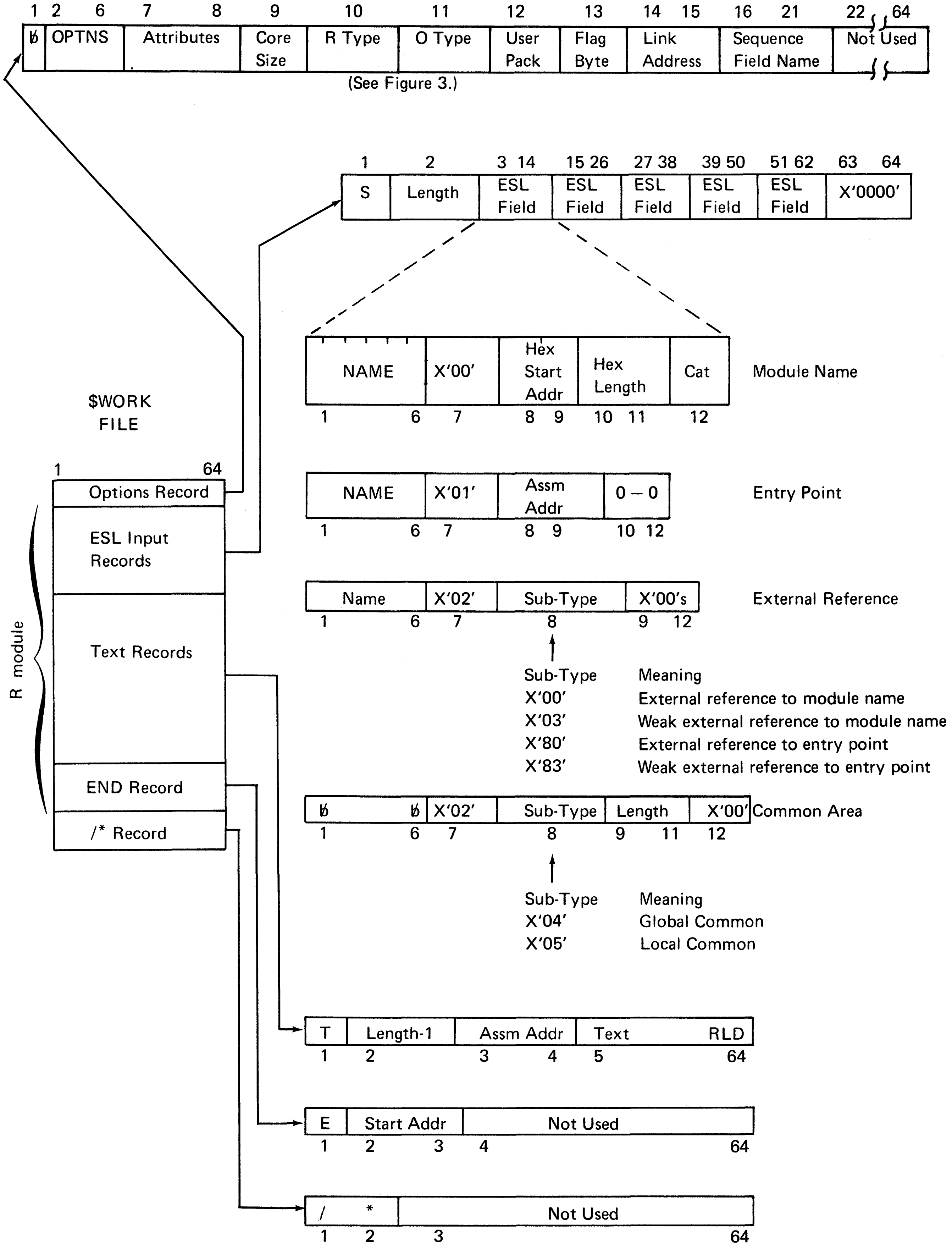

1 2 6 7 8 9 10 11 12 13 14 15 16 21 22 64
b̸ OPTNS Attributes Core Size R Type O Type User Pack Flag Byte Link Address Sequence Field Name Not Used
(See Figure 3.)
1 2 3 14 15 26 27 38 39 50 51 62 63 64
S Length ESL Field ESL Field ESL Field ESL Field ESL Field X'0000'
NAME X'00' Hex Start Addr Hex Length Cat
1 6 7 8 9 10 11 12
Module Name
NAME X'01' Assm Addr 0 – 0
1 6 7 8 9 10 12
Entry Point
Name X'02' Sub-Type X'00's
1 6 7 8 9 12
External Reference
Sub-Type Meaning
X'00' External reference to module name
X'03' Weak external reference to module name
X'80' External reference to entry point
X'83' Weak external reference to entry point
b̸ b̸ X'02' Sub-Type Length X'00'
1 6 7 8 9 11 12
Common Area
Sub-Type Meaning
X'04' Global Common
X'05' Local Common
$WORK FILE
1 64
Options Record
ESL Input Records
Text Records
END Record
/* Record
R module
T Length-1 Assm Addr Text RLD
1 2 3 4 5 64
E Start Addr Not Used
1 2 3 4 64
/ * Not Used
1 2 3 64

Figure 2. Input to Compiler Entry

1	6	7 8	9	10	11	12	13	14 15	16 21	22 64
b̸	OPTNS	Attributes	Core Size	R Type	O Type	User Pack	Flag byte	Link Address	Sequence Field Name	Not Used

Attributes: This two-byte field describes the linked object program built by the Overlay Linkage Editor.

	Byte 1	Byte 2	
Bit	01234567	01234567	
	1xxxxxxx	xxxxxxxx	Permanent entry
	0xxxxxxx	xxxxxxxx	Temporary entry
	x1xxxxxx	xxxxxxxx	Inquiry
	xx1xxxxx	xxxxxxxx	Inquiry Evoking
	xxx1xxxx	xxxxxxxx	Must run dedicated
	xxxx1xxx	xxxxxxxx	Requires source
	xxxxx1xx	xxxxxxxx	Deferred mounting
	xxxxxx1x	xxxxxxxx	PTF applied (not used by Overlay Linkage Editor)
	xxxxxxx1	xxxxxxxx	Overlay program
	xxxxxxxx	1xxxxxxx	System input dedication
	xxxxxxxx	x1xxxxxx	Checkpoint program
	xxxxxxxx	xx1xxxxx	Direct source read
	xxxxxxxx	xxx1xxxx	Macro processor allowed
	xxxxxxxx	xxxx1xxx	Reserved
	xxxxxxxx	xxxxx1xx	FORTRAN Common
	xxxxxxxx	xxxxxx1x	Reserved
	xxxxxxxx	xxxxxxx1	Reserved

Main Storage Size: This is the amount of storage (in 1/4K increments) necessary for execution of the object program.

Example = Hex'12' = 18(Hex'12') X 256 (1/4K) = 4608 Bytes

R-Type: This byte specifies the disposition of the R. module in $WORK.

Bit	01234567	
	1xxxxxxx	Punch into cards
	x1xxxxxx	Catalog into object library on R1
	xx1xxxxx	Catalog into object library on R2
	xxx1xxxx	Catalog into object library on F1
	xxxx1xxx	Catalog into object library on F2
	xxxxx1xx	Catalog as Retain-R in library
	xxxxxx1x	Catalog as permanent entry in library
	xxxxxxx1	Catalog into object library on program pack
	00000000	No R module

Figure 3 (Part 1 of 2). Options Record

O-Type: This byte specifies the disposition of the linked object program and the type of printed output from the Overlay Linkage Editor.

Bit 01234567	
1xxxxxxx	Punch object program into cards
x1xxxxxx	Catalog object program into object library on R1
xx1xxxxx	Catalog object program into object library on R2
xxx1xxxx	Catalog object program into object library on F1
xxxx1xxx	Catalog object program into object library on F2
xxxxx1xx	Do not print core map
xxxxxx1x	Do not print cross-reference list
xxxxxxx1	Catalog object program into object library on program pack
00000000	No linked output. (If neither an R or an O module is specified, an O is cataloged on the program pack.)

User Pack: This byte specifies the pack where user routines are stored. The Overlay Linkage Editor will search this pack first when resolving EXTRNs to modules whose names do not begin with $. If the EXTRN name is not found on this pack, the program pack is searched.

Bit 01234567	
1xxxxxxx	Reserved
x1xxxxxx	Search R1
xx1xxxxx	Search R2
xxx1xxxx	Search F1
xxxx1xxx	Search F2
xxxxx111	Reserved

Flag Byte: This byte passes information to the Overlay Linkage Editor.

Bit 01234567	
11110000	Reserved
000000x1	Link edit address in bytes 14 and 15
0000001x	Catalog Load Module as Retain-R in·library
000001xx	Sequence field name given in bytes 16 through 21
00001xxx	Print messages

Link Address: These two bytes specify a link edit address. If bit 7 of the flag byte is not on,the Overlay Linkage Editor links the load module to the end of the supervisor.

Sequence Field Name: Name put in sequence field. If not specified, ESL will be used.

Figure 3 (Part 2 of 2). Options Record

Output From User Entry

Output of the Overlay Linkage Editor for user entry is an object program cataloged into an object library and/or punched into cards (Figure 4).

A storage map and cross reference list are printed depending on the MAP parameter of the // OPTIONS card.

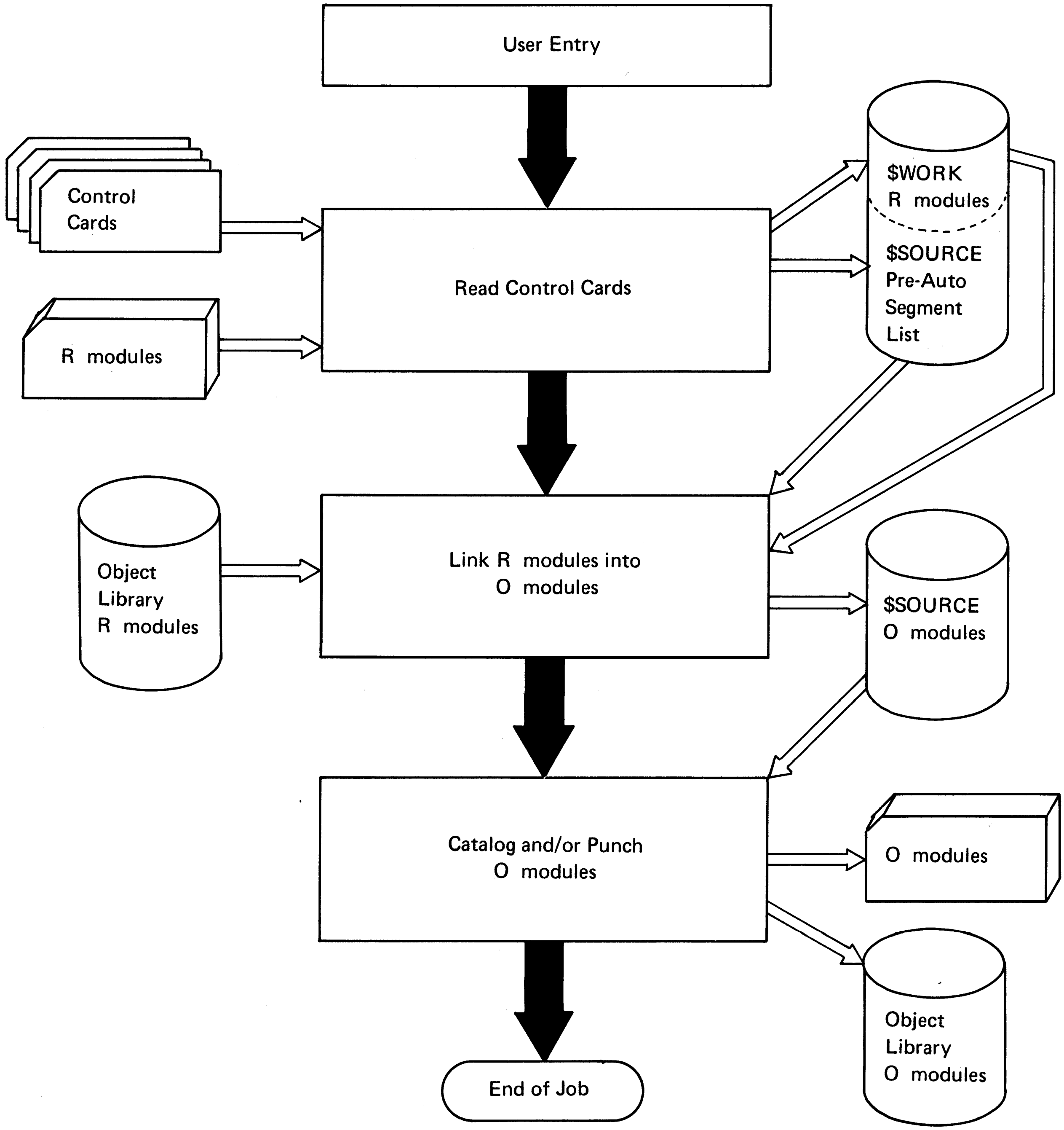
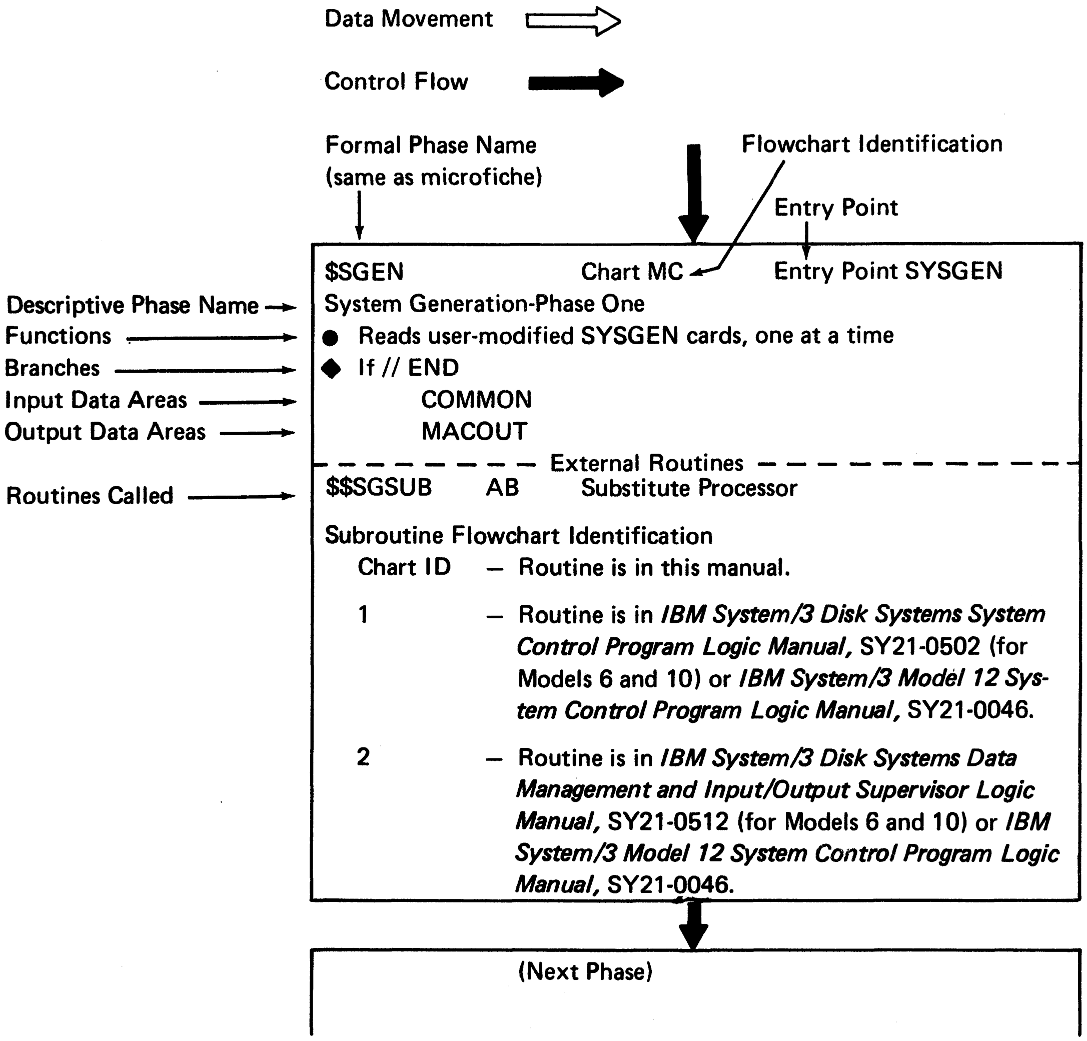


Figure 4. Overview of Overlay Linkage Editor User Entry

SECTION 2. PROGRAM ORGANIZATION

The Overlay Linkage Editor is divided into self-overlaying routines. The sequence in which routines are loaded and which routines are used depends on whether the compiler entry or the user entry is used and which functions are requested. Figure 5 shows an operational diagram of the Overlay Linkage Editor program. Storage maps of the compiler interface and the user interface are shown in Figures 6 and 7.



Subroutine Flowchart Identification

Chart ID – Routine is in this manual.

1 – Routine is in *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502 (for Models 6 and 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

2 – Routine is in *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual,* SY21-0512 (for Models 6 and 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

Figure 5 (Part 1 of 3). Operational Diagram Legend

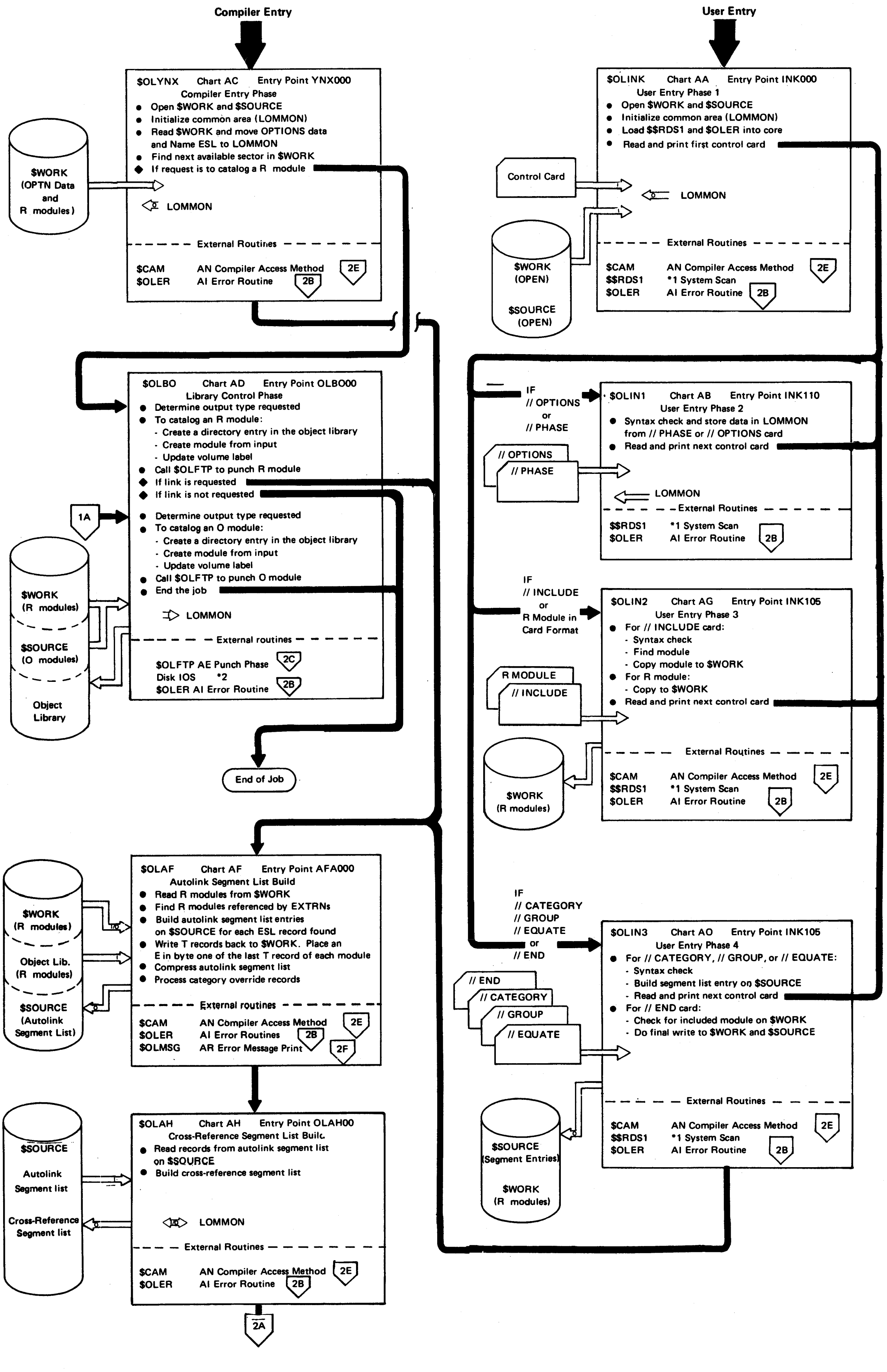
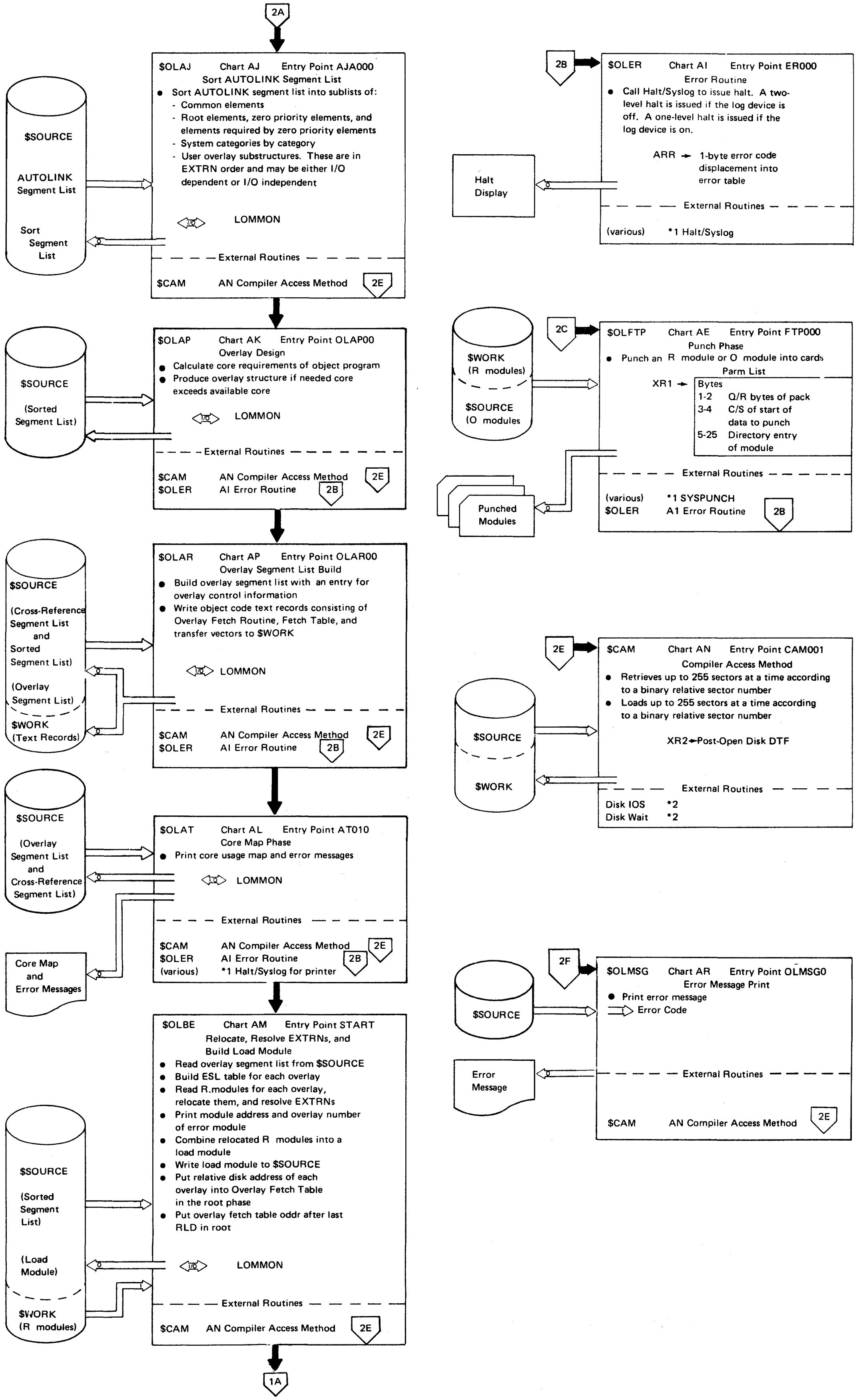

Compiler Entry
$OLYNX Chart AC Entry Point YNX000
Compiler Entry Phase
● Open $WORK and $SOURCE
● Initialize common area (LOMMON)
● Read $WORK and move OPTIONS data and Name ESL to LOMMON
● Find next available sector in $WORK
◆ If request is to catalog a R module
$WORK (OPTN Data and R modules)
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routine 2B
User Entry
$OLINK Chart AA Entry Point INK000
User Entry Phase 1
● Open $WORK and $SOURCE
● Initialize common area (LOMMON)
● Load $$RDS1 and $OLER into core
● Read and print first control card
Control Card
LOMMON
$WORK (OPEN)
$SOURCE (OPEN)
External Routines
$CAM AN Compiler Access Method 2E
$$RDS1 *1 System Scan
$OLER AI Error Routine 2B
$OLBO Chart AD Entry Point OLBO00
Library Control Phase
● Determine output type requested
● To catalog an R module:
- Create a directory entry in the object library
- Create module from input
- Update volume label
● Call $OLFTP to punch R module
◆ If link is requested
◆ If link is not requested
1A
● Determine output type requested
● To catalog an O module:
- Create a directory entry in the object library
- Create module from input
- Update volume label
● Call $OLFTP to punch O module
● End the job
LOMMON
External routines
$OLFTP AE Punch Phase 2C
Disk IOS *2
$OLER AI Error Routine 2B
$WORK (R modules)
$SOURCE (O modules)
Object Library
End of Job
IF
// OPTIONS
or
// PHASE
$OLIN1 Chart AB Entry Point INK110
User Entry Phase 2
● Syntax check and store data in LOMMON from // PHASE or // OPTIONS card
● Read and print next control card
// OPTIONS
// PHASE
LOMMON
External Routines
$$RDS1 *1 System Scan
$OLER AI Error Routine 2B
IF
// INCLUDE
or
R Module in Card Format
$OLIN2 Chart AG Entry Point INK105
User Entry Phase 3
● For // INCLUDE card:
- Syntax check
- Find module
- Copy module to $WORK
● For R module:
- Copy to $WORK
● Read and print next control card
R MODULE
// INCLUDE
$WORK (R modules)
External Routines
$CAM AN Compiler Access Method 2E
$$RDS1 *1 System Scan
$OLER AI Error Routine 2B
$OLAF Chart AF Entry Point AFA000
Autolink Segment List Build
● Read R modules from $WORK
● Find R modules referenced by EXTRNs
● Build autolink segment list entries on $SOURCE for each ESL record found
● Write T records back to $WORK. Place an E in byte one of the last T record of each module
● Compress autolink segment list
● Process category override records
$WORK (R modules)
Object Lib. (R modules)
$SOURCE (Autolink Segment List)
External routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routines 2B
$OLMSG AR Error Message Print 2F
IF
// CATEGORY
// GROUP
// EQUATE
or
// END
$OLIN3 Chart AO Entry Point INK105
User Entry Phase 4
● For // CATEGORY, // GROUP, or // EQUATE:
- Syntax check
- Build segment list entry on $SOURCE
- Read and print next control card
● For // END card:
- Check for included module on $WORK
- Do final write to $WORK and $SOURCE
// END
// CATEGORY
// GROUP
// EQUATE
$SOURCE (Segment Entries)
$WORK (R modules)
External Routines
$CAM AN Compiler Access Method 2E
$$RDS1 *1 System Scan
$OLER AI Error Routine 2B
$OLAH Chart AH Entry Point OLAH00
Cross-Reference Segment List Built
● Read records from autolink segment list on $SOURCE
● Build cross-reference segment list
$SOURCE
Autolink Segment list
Cross-Reference Segment list
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routine 2B
2A

Figure 5 (Part 2 of 3). Operational Diagram


2A
$OLAJ Chart AJ Entry Point AJA000
Sort AUTOLINK Segment List
● Sort AUTOLINK segment list into sublists of:
- Common elements
- Root elements, zero priority elements, and elements required by zero priority elements
- System categories by category
- User overlay substructures. These are in EXTRN order and may be either I/O dependent or I/O independent
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$SOURCE
AUTOLINK Segment List
Sort Segment List
$OLAP Chart AK Entry Point OLAP00
Overlay Design
● Calculate core requirements of object program
● Produce overlay structure if needed core exceeds available core
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routine 2B
$SOURCE
(Sorted Segment List)
$OLAR Chart AP Entry Point OLAR00
Overlay Segment List Build
● Build overlay segment list with an entry for overlay control information
● Write object code text records consisting of Overlay Fetch Routine, Fetch Table, and transfer vectors to $WORK
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routine 2B
$SOURCE
(Cross-Reference Segment List and Sorted Segment List)
(Overlay Segment List)
$WORK (Text Records)
$OLAT Chart AL Entry Point AT010
Core Map Phase
● Print core usage map and error messages
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$OLER AI Error Routine 2B
(various) *1 Halt/Syslog for printer
$SOURCE
(Overlay Segment List and Cross-Reference Segment List)
Core Map and Error Messages
$OLBE Chart AM Entry Point START
Relocate, Resolve EXTRNs, and Build Load Module
● Read overlay segment list from $SOURCE
● Build ESL table for each overlay
● Read R.modules for each overlay, relocate them, and resolve EXTRNs
● Print module address and overlay number of error module
● Combine relocated R modules into a load module
● Write load module to $SOURCE
● Put relative disk address of each overlay into Overlay Fetch Table in the root phase
● Put overlay fetch table oddr after last RLD in root
LOMMON
External Routines
$CAM AN Compiler Access Method 2E
$SOURCE
(Sorted Segment List)
(Load Module)
$WORK (R modules)
1A
2B
$OLER Chart AI Entry Point ER000
Error Routine
● Call Halt/Syslog to issue halt. A two-level halt is issued if the log device is off. A one-level halt is issued if the log device is on.
ARR → 1-byte error code displacement into error table
Halt Display
External Routines
(various) *1 Halt/Syslog
2C
$OLFTP Chart AE Entry Point FTP000
Punch Phase
● Punch an R module or O module into cards
Parm List
XR1 →
Bytes
1-2 Q/R bytes of pack
3-4 C/S of start of data to punch
5-25 Directory entry of module
External Routines
(various) *1 SYSPUNCH
$OLER A1 Error Routine 2B
$WORK (R modules)
$SOURCE (O modules
Punched Modules
2E
$CAM Chart AN Entry Point CAM001
Compiler Access Method
● Retrieves up to 255 sectors at a time according to a binary relative sector number
● Loads up to 255 sectors at a time according to a binary relative sector number
XR2→Post-Open Disk DTF
External Routines
Disk IOS *2
Disk Wait *2
$SOURCE
$WORK
2F
$OLMSG Chart AR Entry Point OLMSG0
Error Message Print
● Print error message
Error Code
External Routines
$CAM AN Compiler Access Method 2E
$SOURCE
Error Message

● Figure 5 (Part 3 of 3). Operational Diagram

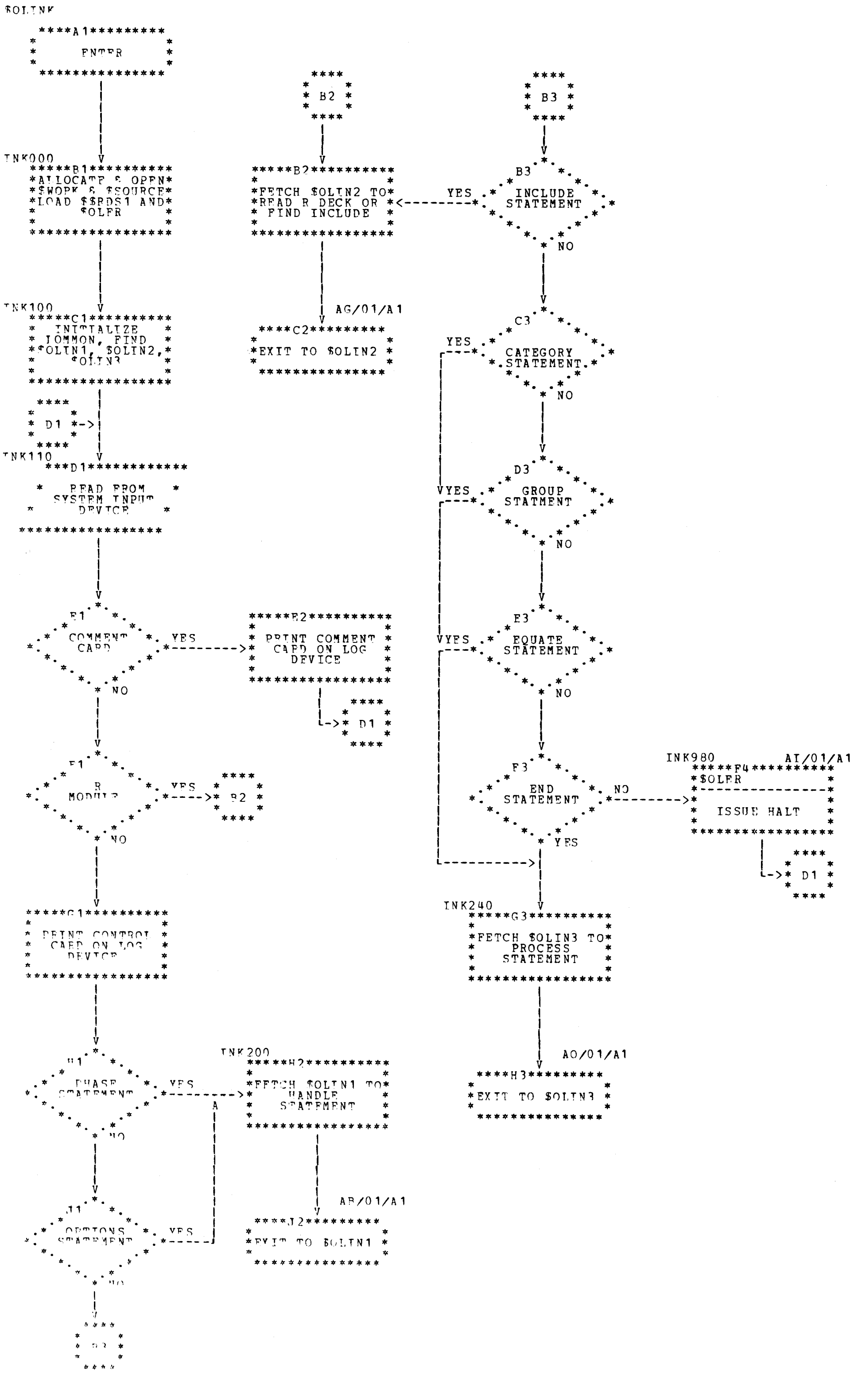


Chart AA. User Entry Phase 1 ($OLINK)

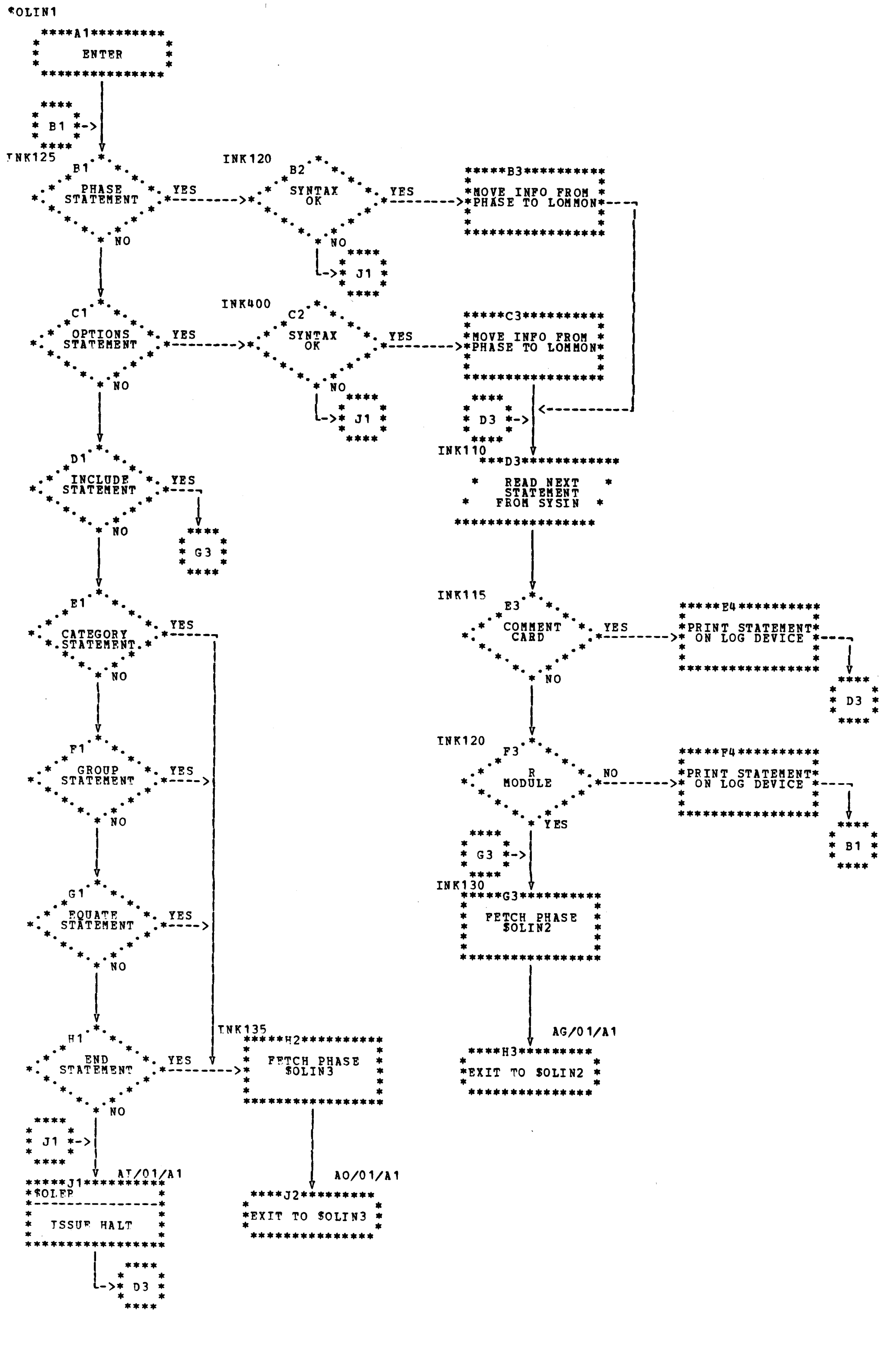


Chart AB. User Entry Phase 2 ($OLIN1)

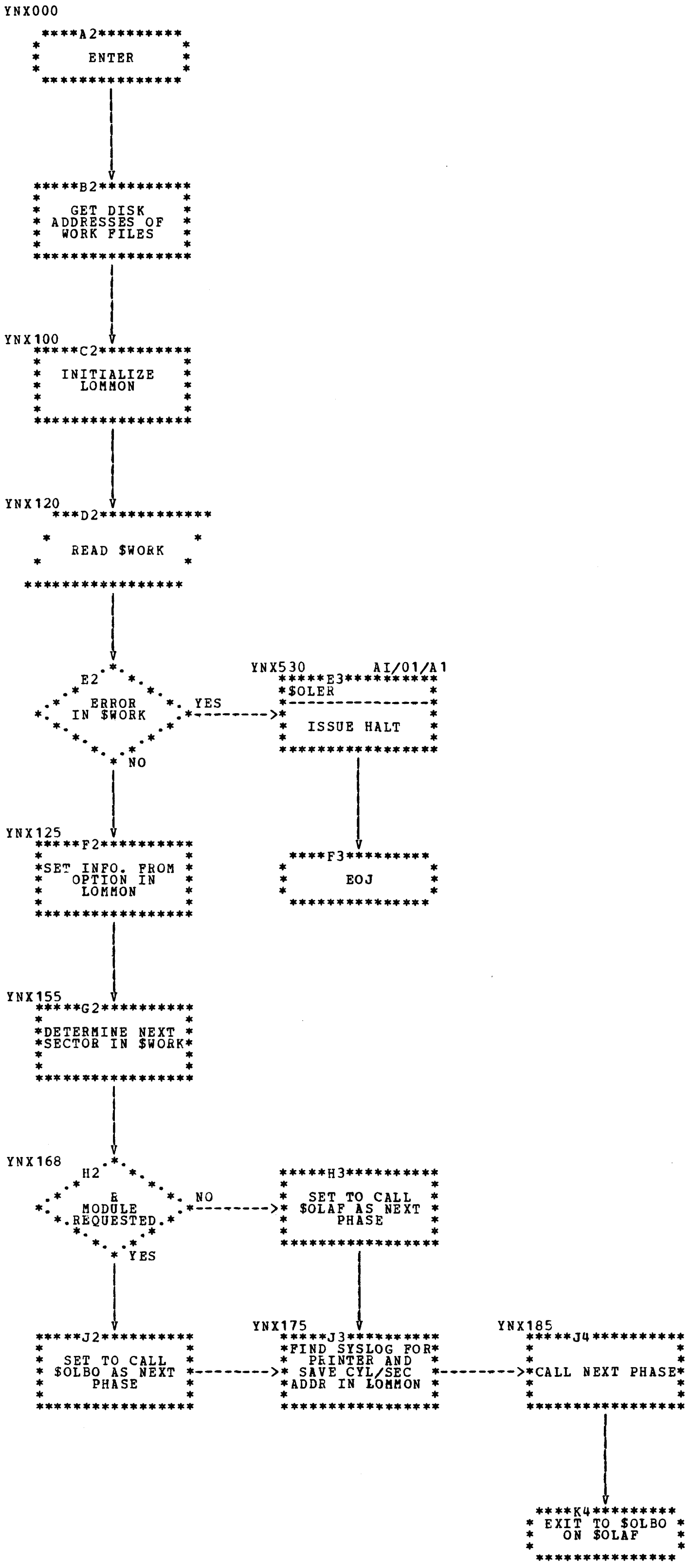
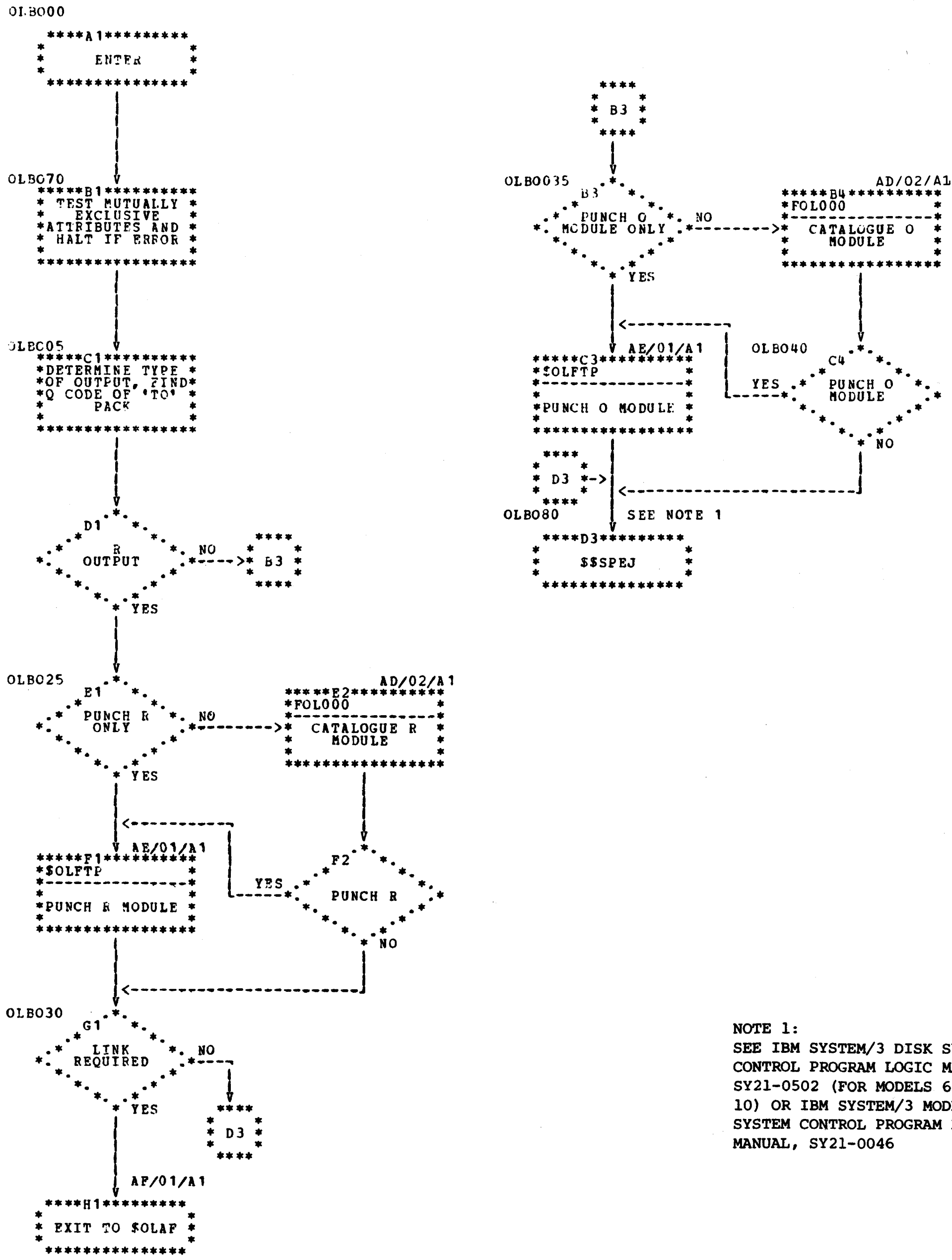


Chart AC. Compiler Entry Phase ($OLYNX)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart AD (Part 1 of 2). Library Control Phase ($OLBO)

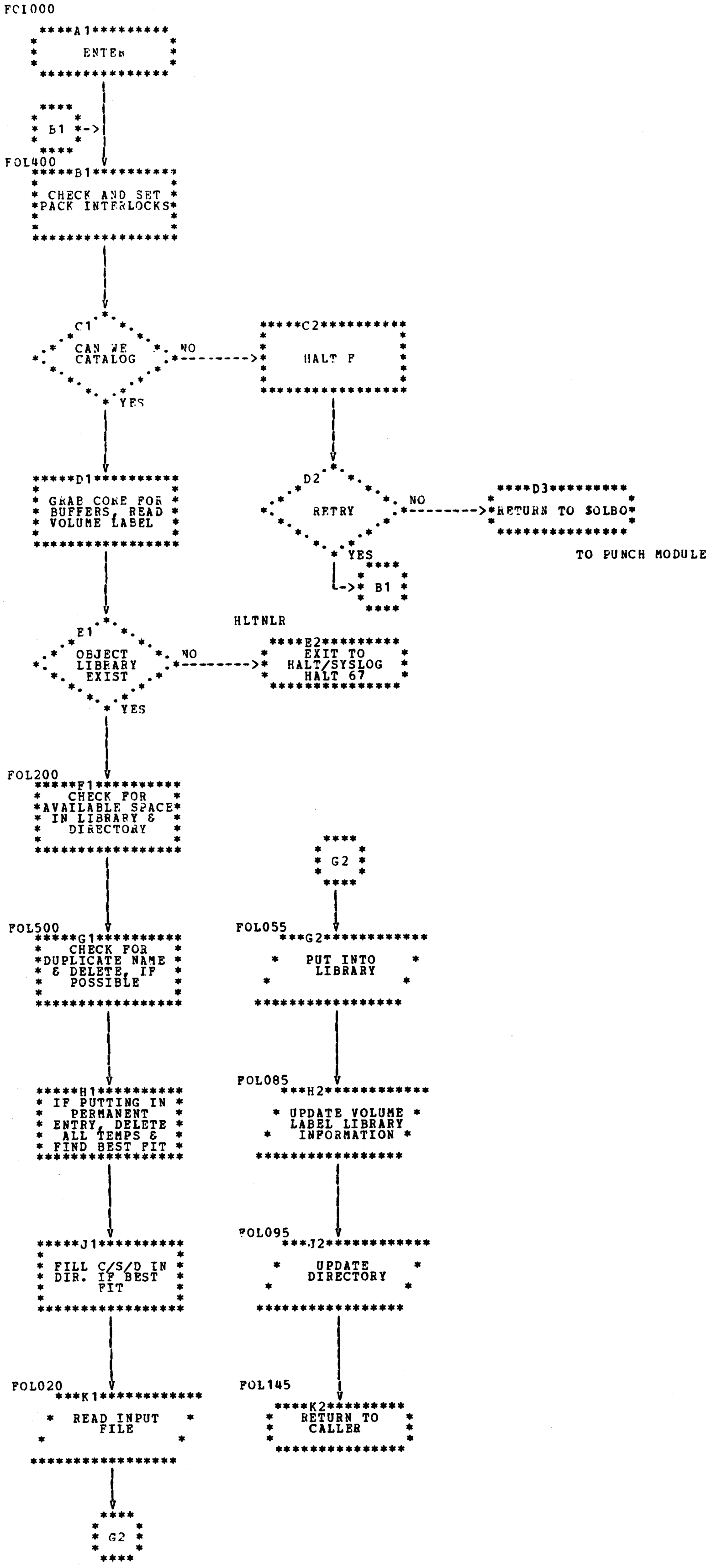

FCI000
A1
ENTER
B1
FOL400
B1
CHECK AND SET PACK INTERLOCKS
C1
CAN WE CATALOG
NO
YES
C2
HALT F
D1
GRAB CORE FOR BUFFERS, READ VOLUME LABEL
D2
RETRY
NO
YES
B1
D3
RETURN TO $OLBO
TO PUNCH MODULE
E1
OBJECT LIBRARY EXIST
NO
YES
HLTNLR
E2
EXIT TO HALT/SYSLOG HALT 67
FOL200
F1
CHECK FOR AVAILABLE SPACE IN LIBRARY & DIRECTORY
G2
FOL500
G1
CHECK FOR DUPLICATE NAME & DELETE, IF POSSIBLE
FOL055
G2
PUT INTO LIBRARY
H1
IF PUTTING IN PERMANENT ENTRY, DELETE ALL TEMPS & FIND BEST FIT
FOL085
H2
UPDATE VOLUME LABEL LIBRARY INFORMATION
J1
FILL C/S/D IN DIR. IF BEST FIT
FOL095
J2
UPDATE DIRECTORY
FOL020
K1
READ INPUT FILE
FOL145
K2
RETURN TO CALLER
G2

Chart AD (Part 2 of 2). Library Control Phase ($OLBO)

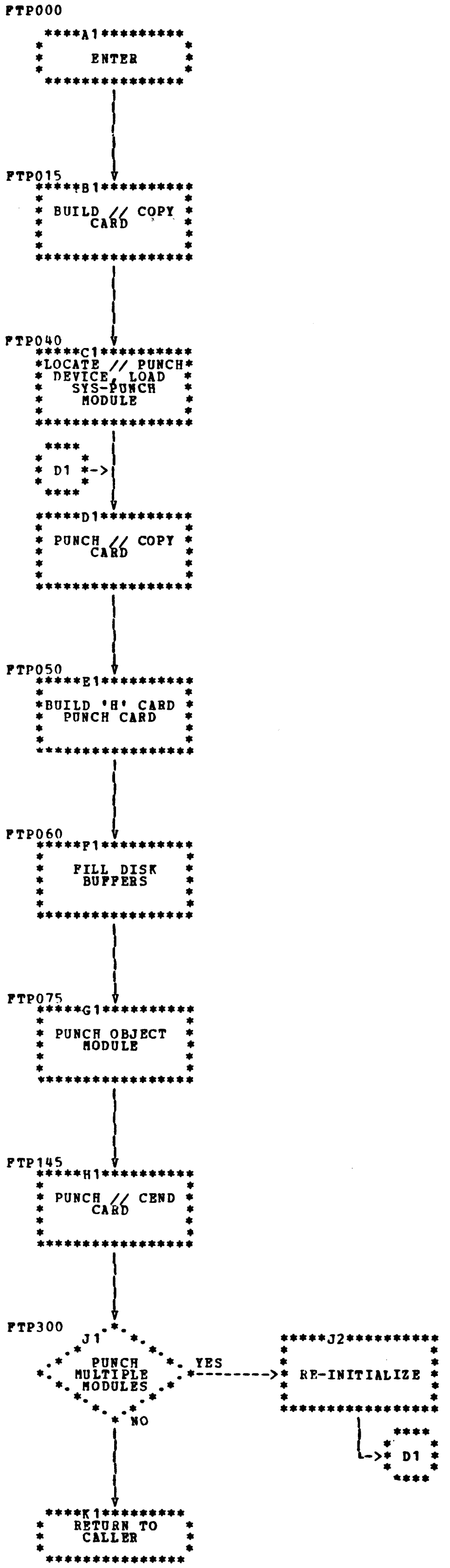
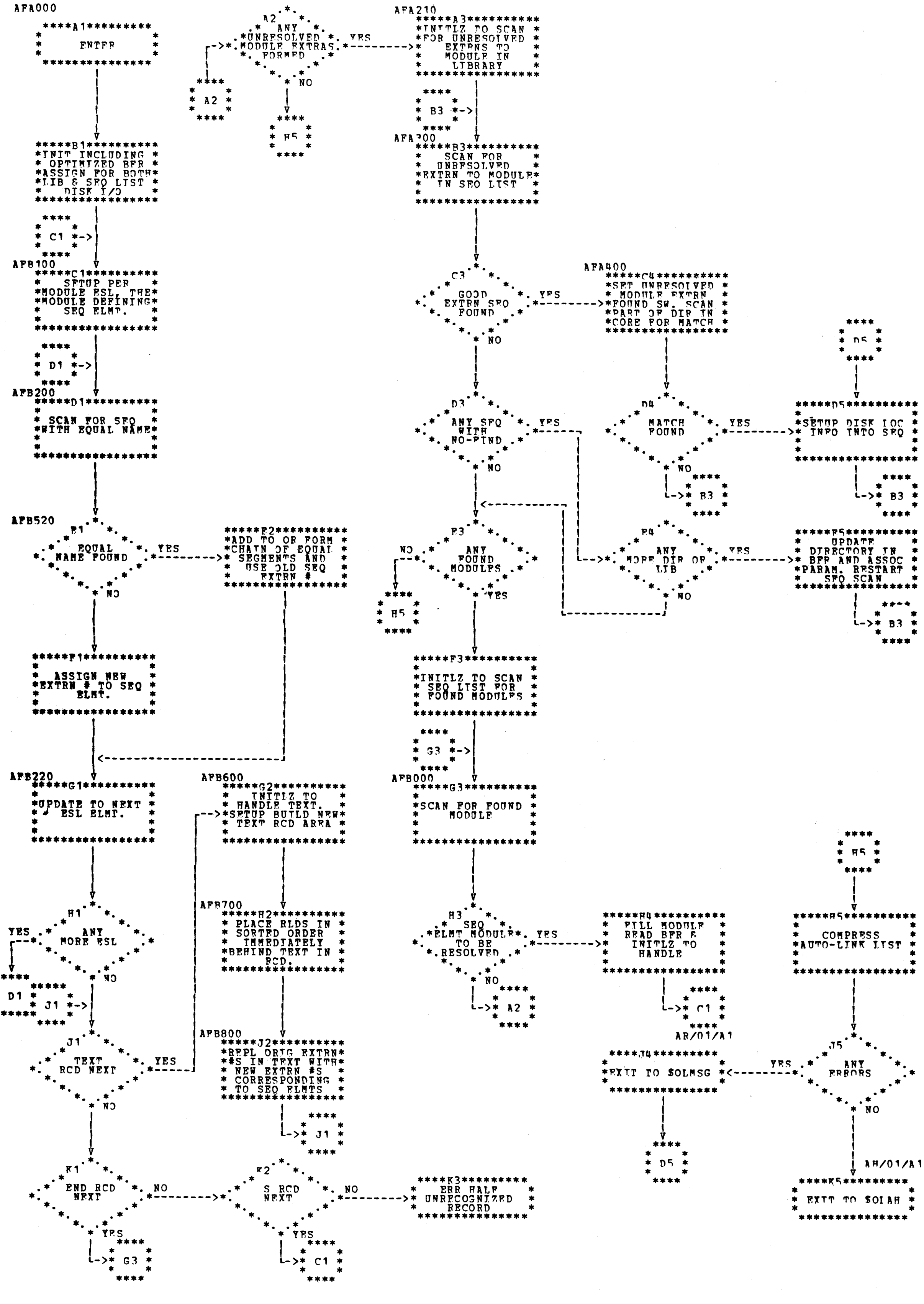
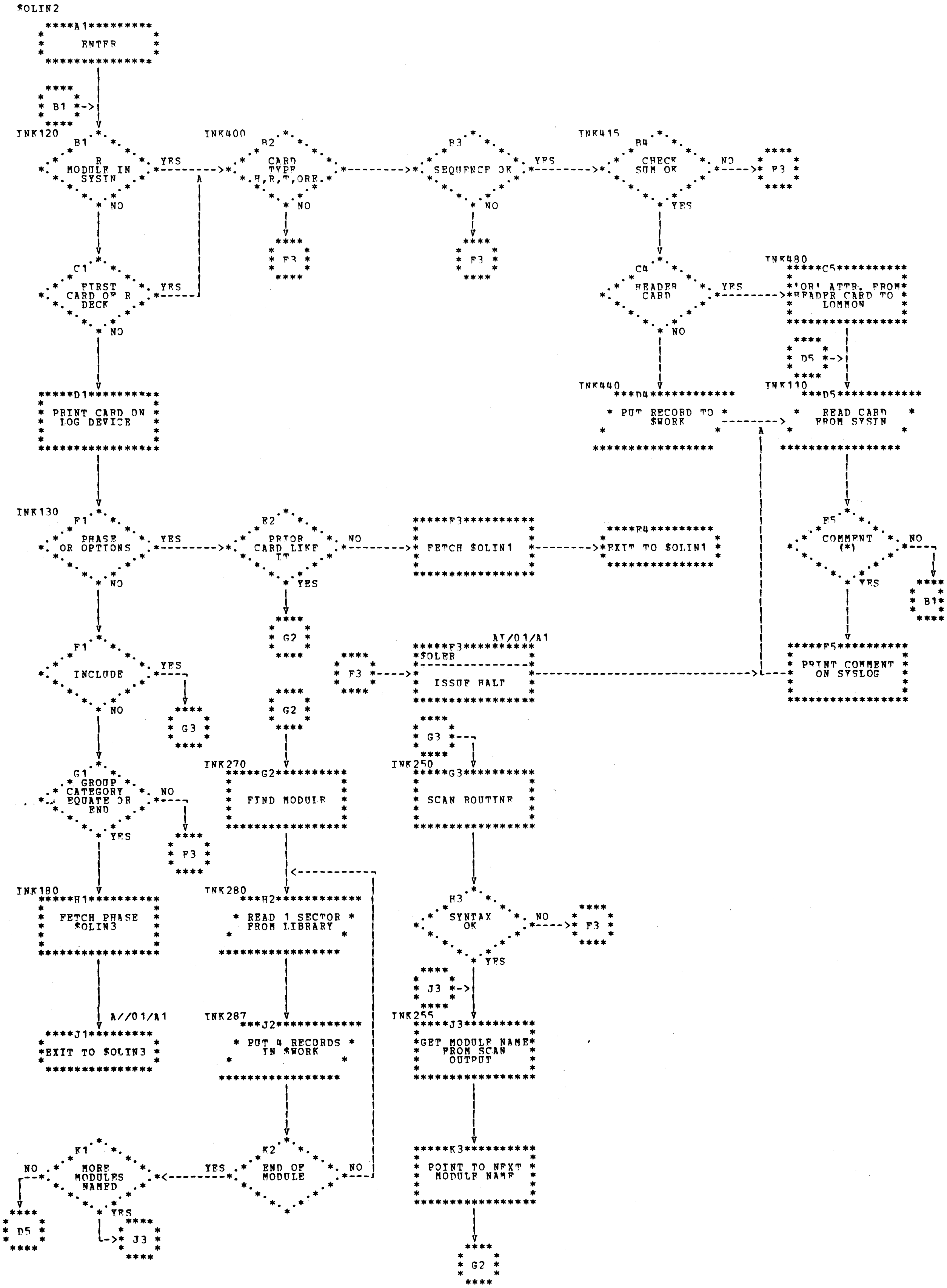


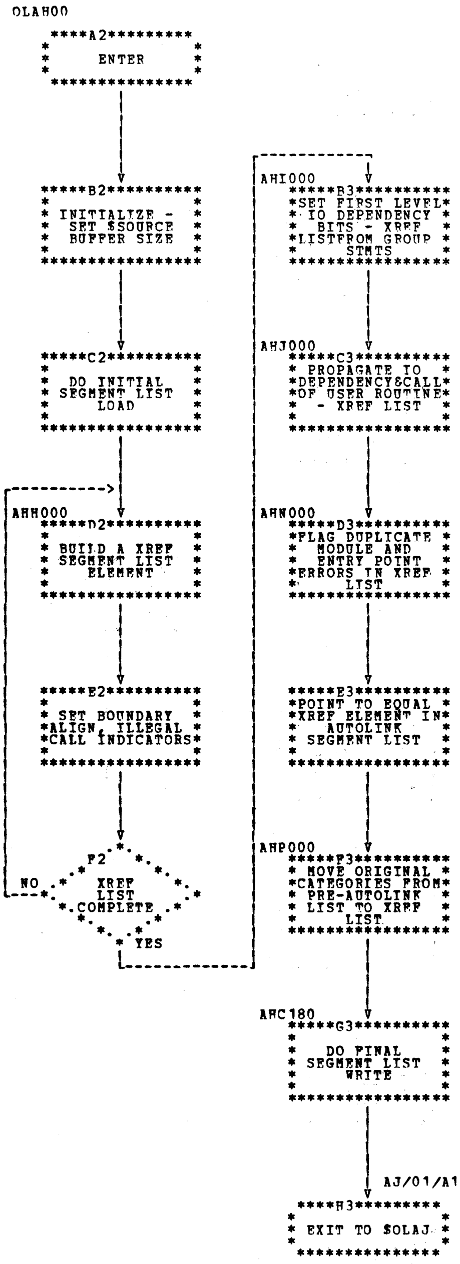
Chart AE. Punch Phase ($OLFTP)


AFA000
A1
ENTER
A2
ANY UNRESOLVED MODULE EXTRNS FORMED
YES
NO
A2
H5
AFA210
A3
INITLZ TO SCAN FOR UNRESOLVED EXTRNS TO MODULE IN LIBRARY
B3
AFA200
B3
SCAN FOR UNRESOLVED EXTRN TO MODULE IN SEQ LIST
B1
INIT INCLUDING OPTIMIZED BFR ASSIGN FOR BOTH LIB & SEQ LIST DISK I/O
C1
AFB100
C1
SETUP PER MODULE ESL, THE MODULE DEFINING SEQ ELMT.
D1
AFB200
D1
SCAN FOR SEQ WITH EQUAL NAME
AFB520
E1
EQUAL NAME FOUND
YES
NO
E2
ADD TO OR FORM CHAIN OF EQUAL SEGMENTS AND USE OLD SEQ EXTRN #
F1
ASSIGN NEW EXTRN # TO SEQ ELMT.
AFB220
G1
UPDATE TO NEXT ESL ELMT.
H1
ANY MORE ESL
YES
NO
D1
J1
J1
TEXT RCD NEXT
YES
NO
K1
END RCD NEXT
NO
YES
G3
AFB600
G2
INITLZ TO HANDLE TEXT. SETUP BUILD NEW TEXT RCD AREA
AFB700
H2
PLACE RLDS IN SORTED ORDER IMMEDIATELY BEHIND TEXT IN RCD.
AFB800
J2
REPL ORIG EXTRN #S IN TEXT WITH NEW EXTRN #S CORRESPONDING TO SEQ ELMTS
J1
K2
S RCD NEXT
NO
YES
C1
K3
ERR HALT UNRECOGNIZED RECORD
C3
GOOD EXTRN SEQ FOUND
YES
NO
AFA400
C4
SET UNRESOLVED MODULE EXTRN FOUND SW. SCAN PART OF DIR IN CORE FOR MATCH
D5
D3
ANY SEQ WITH NO-FIND
YES
NO
D4
MATCH FOUND
YES
NO
B3
D5
SETUP DISK LOC INFO INTO SEQ
B3
E3
ANY FOUND MODULES
NO
YES
H5
E4
ANY MORE DIR OF LIB
YES
NO
E5
UPDATE DIRECTORY IN BFR AND ASSOC PARAM. RESTART SEQ SCAN
B3
F3
INITLZ TO SCAN SEQ LIST FOR FOUND MODULES
G3
AFB000
G3
SCAN FOR FOUND MODULE
H3
SEQ ELMT MODULE TO BE RESOLVED
YES
NO
A2
H4
FILL MODULE READ BFR & INITLZ TO HANDLE
C1
AB/01/A1
H5
H5
COMPRESS AUTO-LINK LIST
J5
ANY ERRORS
YES
NO
J4
EXIT TO $OLMSG
D5
AH/01/A1
K5
EXIT TO $OLAH

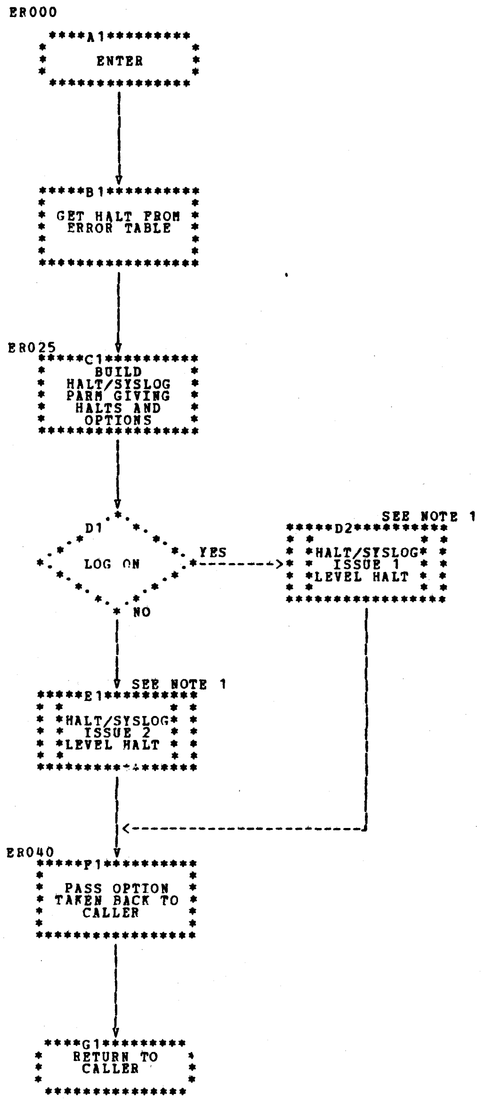
● Chart AF. Auto-Link Segment List Build ($OLAF)



● Chart AG. User Entry Phase 3 ($OLIN2)

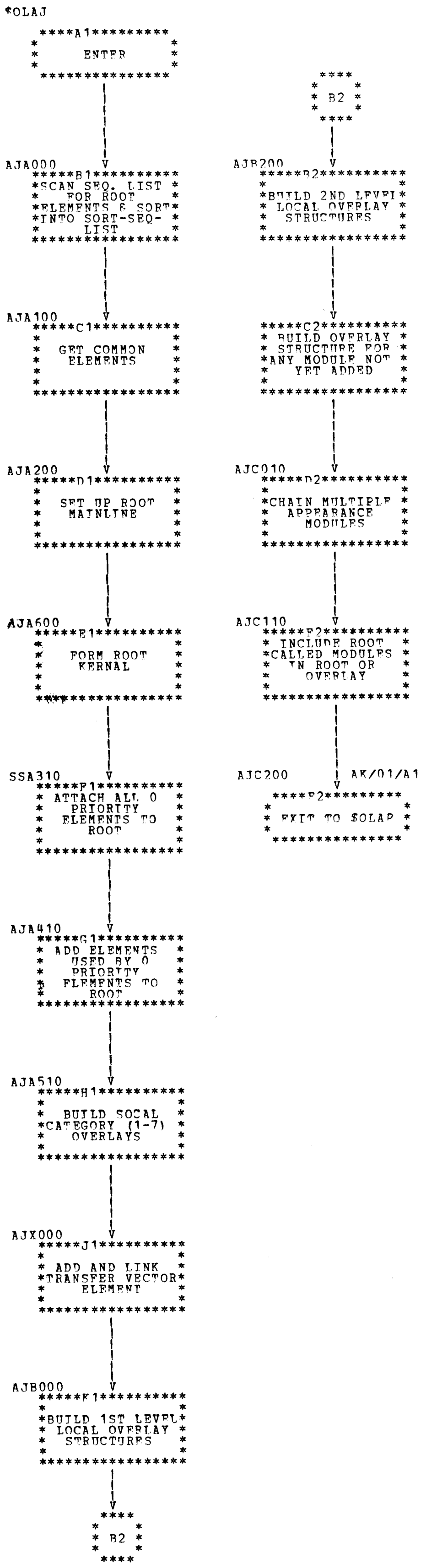

OLAH00
A2
ENTER
B2
INITIALIZE - SET $SOURCE BUFFER SIZE
C2
DO INITIAL SEGMENT LIST LOAD
AHH000
D2
BUILD A XREF SEGMENT LIST ELEMENT
E2
SET BOUNDARY ALIGN, ILLEGAL CALL INDICATORS
F2
XREF LIST COMPLETE
NO
YES
AHI000
B3
SET FIRST LEVEL IO DEPENDENCY BITS - XREF LISTFROM GROUP STMTS
AHJ000
C3
PROPAGATE IO DEPENDENCY&CALL OF USER ROUTINE - XREF LIST
AHN000
D3
FLAG DUPLICATE MODULE AND ENTRY POINT ERRORS IN XREF LIST
E3
POINT TO EQUAL XREF ELEMENT IN AUTOLINK SEGMENT LIST
AHP000
F3
MOVE ORIGINAL CATEGORIES FROM PRE-AUTOLINK LIST TO XREF LIST
AHC180
G3
DO FINAL SEGMENT LIST WRITE
AJ/01/A1
H3
EXIT TO $OLAJ

● Chart AH. Cross-Reference Segment List Build ($OLAH)


ER000
A1
ENTER
B1
GET HALT FROM ERROR TABLE
ER025
C1
BUILD HALT/SYSLOG PARM GIVING HALTS AND OPTIONS
D1
LOG ON
YES
NO
SEE NOTE 1
D2
HALT/SYSLOG ISSUE 1 LEVEL HALT
SEE NOTE 1
E1
HALT/SYSLOG ISSUE 2 LEVEL HALT
ER040
F1
PASS OPTION TAKEN BACK TO CALLER
G1
RETURN TO CALLER

NOTE 1:
SEE IBM SYSTEM/3
DISK SYSTEMS SYSTEM
CONTROL PROGRAM
LOGIC MANUAL
SY21-0502

Chart AI. Error Routing ($OLER)


$OLAJ
A1
ENTER
B2
AJA000
B1
SCAN SEQ. LIST FOR ROOT ELEMENTS & SORT INTO SORT-SEQ-LIST
AJB200
B2
BUILD 2ND LEVEL LOCAL OVERLAY STRUCTURES
AJA100
C1
GET COMMON ELEMENTS
C2
BUILD OVERLAY STRUCTURE FOR ANY MODULE NOT YET ADDED
AJA200
D1
SET UP ROOT MAINLINE
AJC010
D2
CHAIN MULTIPLE APPEARANCE MODULES
AJA600
E1
FORM ROOT KERNAL
AJC110
E2
INCLUDE ROOT CALLED MODULES IN ROOT OR OVERLAY
SSA310
F1
ATTACH ALL 0 PRIORITY ELEMENTS TO ROOT
AJC200
AK/01/A1
F2
EXIT TO $OLAP
AJA410
G1
ADD ELEMENTS USED BY 0 PRIORITY ELEMENTS TO ROOT
AJA510
H1
BUILD SOCAL CATEGORY (1-7) OVERLAYS
AJX000
J1
ADD AND LINK TRANSFER VECTOR ELEMENT
AJB000
K1
BUILD 1ST LEVEL LOCAL OVERLAY STRUCTURES
B2

● Chart AJ. Sort Auto-Link Segment List ($OLAJ)

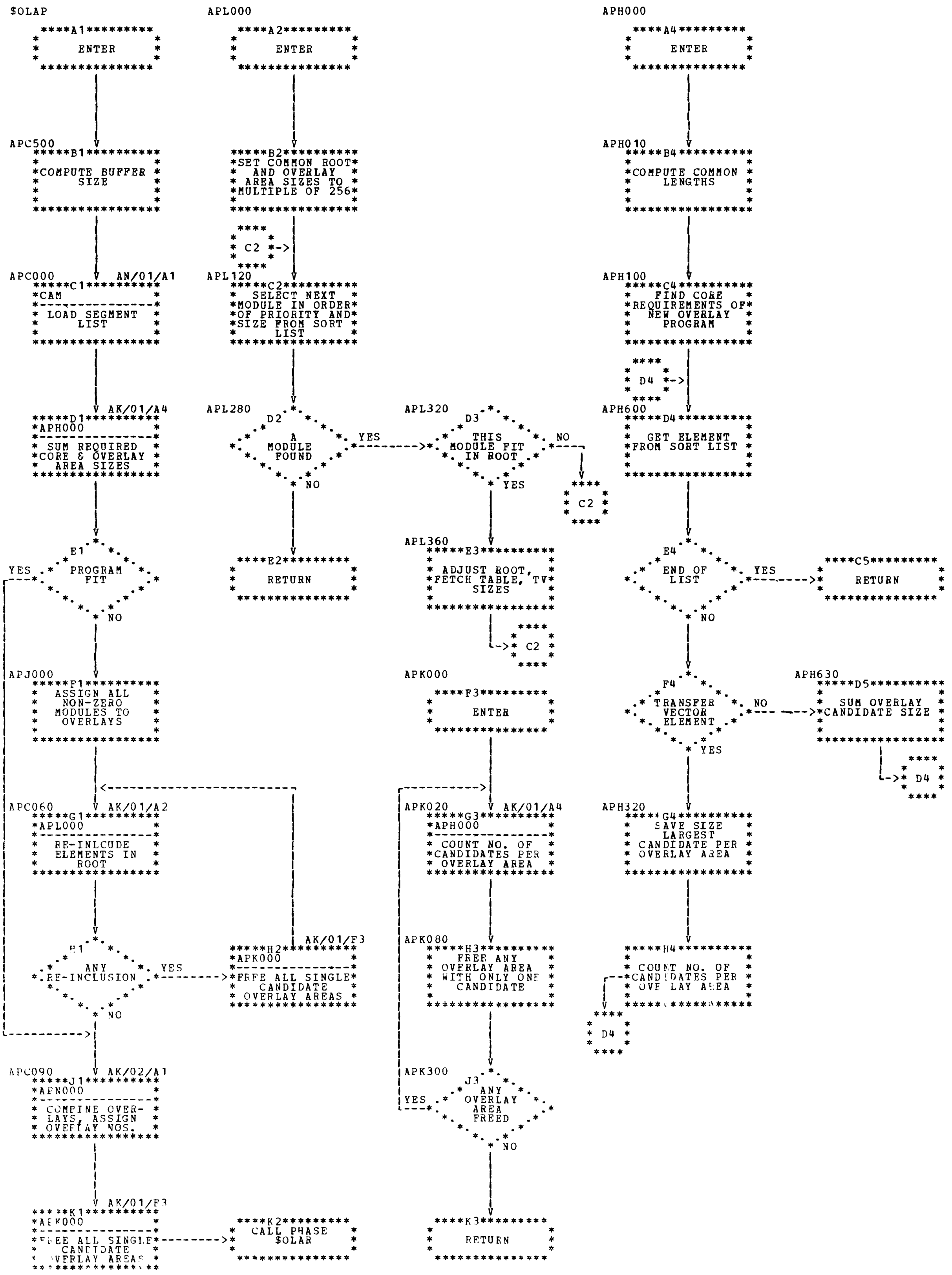

$OLAP
A1 ENTER
APC500
B1 COMPUTE BUFFER SIZE
APC000 AN/01/A1
C1 CAM
LOAD SEGMENT LIST
AK/01/A4
D1 APH000
SUM REQUIRED CORE & OVERLAY AREA SIZES
E1 PROGRAM FIT
YES
NO
APJ000
F1 ASSIGN ALL NON-ZERO MODULES TO OVERLAYS
APC060 AK/01/A2
G1 APL000
RE-INLCUDE ELEMENTS IN ROOT
H1 ANY RE-INCLUSION
YES
NO
APC090 AK/02/A1
J1 APN000
COMBINE OVER-LAYS, ASSIGN OVERLAY NOS.
AK/01/F3
K1 APK000
FREE ALL SINGLE CANDIDATE OVERLAY AREAS
APL000
A2 ENTER
B2 SET COMMON ROOT AND OVERLAY AREA SIZES TO MULTIPLE OF 256
C2
APL120
C2 SELECT NEXT MODULE IN ORDER OF PRIORITY AND SIZE FROM SORT LIST
APL280
D2 A MODULE FOUND
YES
NO
E2 RETURN
AK/01/F3
H2 APK000
FREE ALL SINGLE CANDIDATE OVERLAY AREAS
K2 CALL PHASE $OLAR
APL320
D3 THIS MODULE FIT IN ROOT
NO
YES
C2
APL360
E3 ADJUST ROOT, FETCH TABLE, TV SIZES
C2
APK000
F3 ENTER
APK020 AK/01/A4
G3 APH000
COUNT NO. OF CANDIDATES PER OVERLAY AREA
APK080
H3 FREE ANY OVERLAY AREA WITH ONLY ONE CANDIDATE
APK300
J3 ANY OVERLAY AREA FREED
YES
NO
K3 RETURN
APH000
A4 ENTER
APH010
B4 COMPUTE COMMON LENGTHS
APH100
C4 FIND CORE REQUIREMENTS OF NEW OVERLAY PROGRAM
D4
APH600
D4 GET ELEMENT FROM SORT LIST
E4 END OF LIST
YES
NO
F4 TRANSFER VECTOR ELEMENT
NO
YES
APH320
G4 SAVE SIZE LARGEST CANDIDATE PER OVERLAY AREA
H4 COUNT NO. OF CANDIDATES PER OVERLAY AREA
D4
C5 RETURN
APH630
D5 SUM OVERLAY CANDIDATE SIZE
D4

Chart AK (Part 1 of 2). Overlay Design ($OLAP)

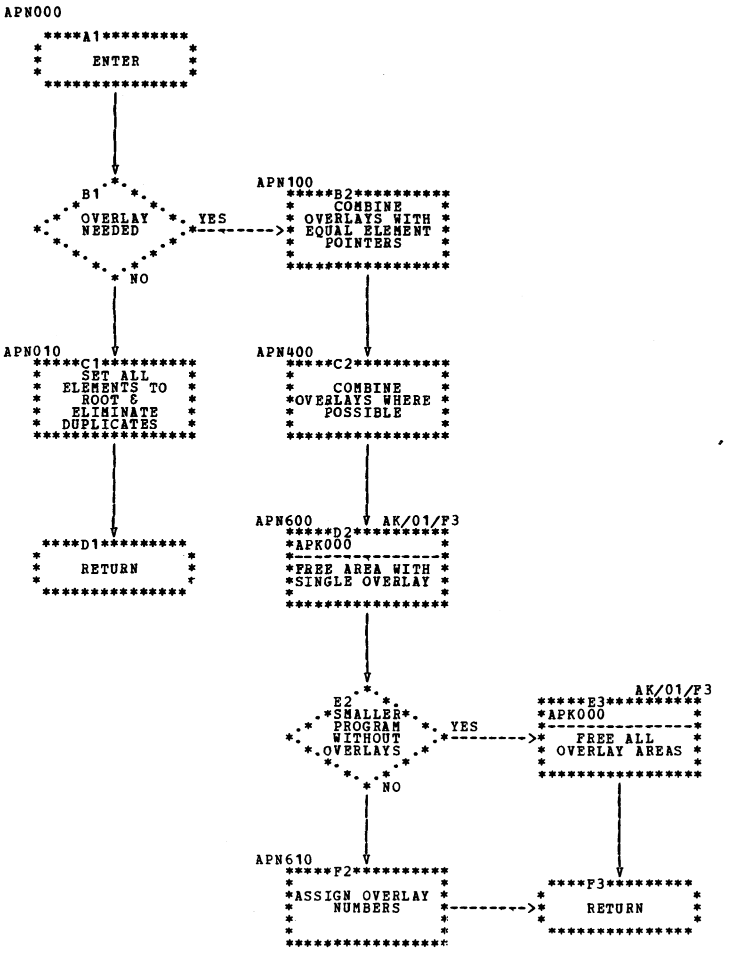

APN000
A1
ENTER
B1
OVERLAY NEEDED
YES
NO
APN100
B2
COMBINE OVERLAYS WITH EQUAL ELEMENT POINTERS
APN010
C1
SET ALL ELEMENTS TO ROOT & ELIMINATE DUPLICATES
APN400
C2
COMBINE OVERLAYS WHERE POSSIBLE
D1
RETURN
APN600
AK/01/F3
D2
APK000
FREE AREA WITH SINGLE OVERLAY
E2
SMALLER PROGRAM WITHOUT OVERLAYS
YES
NO
AK/01/F3
E3
APK000
FREE ALL OVERLAY AREAS
APN610
F2
ASSIGN OVERLAY NUMBERS
F3
RETURN

Chart AK (Part 2 of 2). Overlay Design ($OLAP)

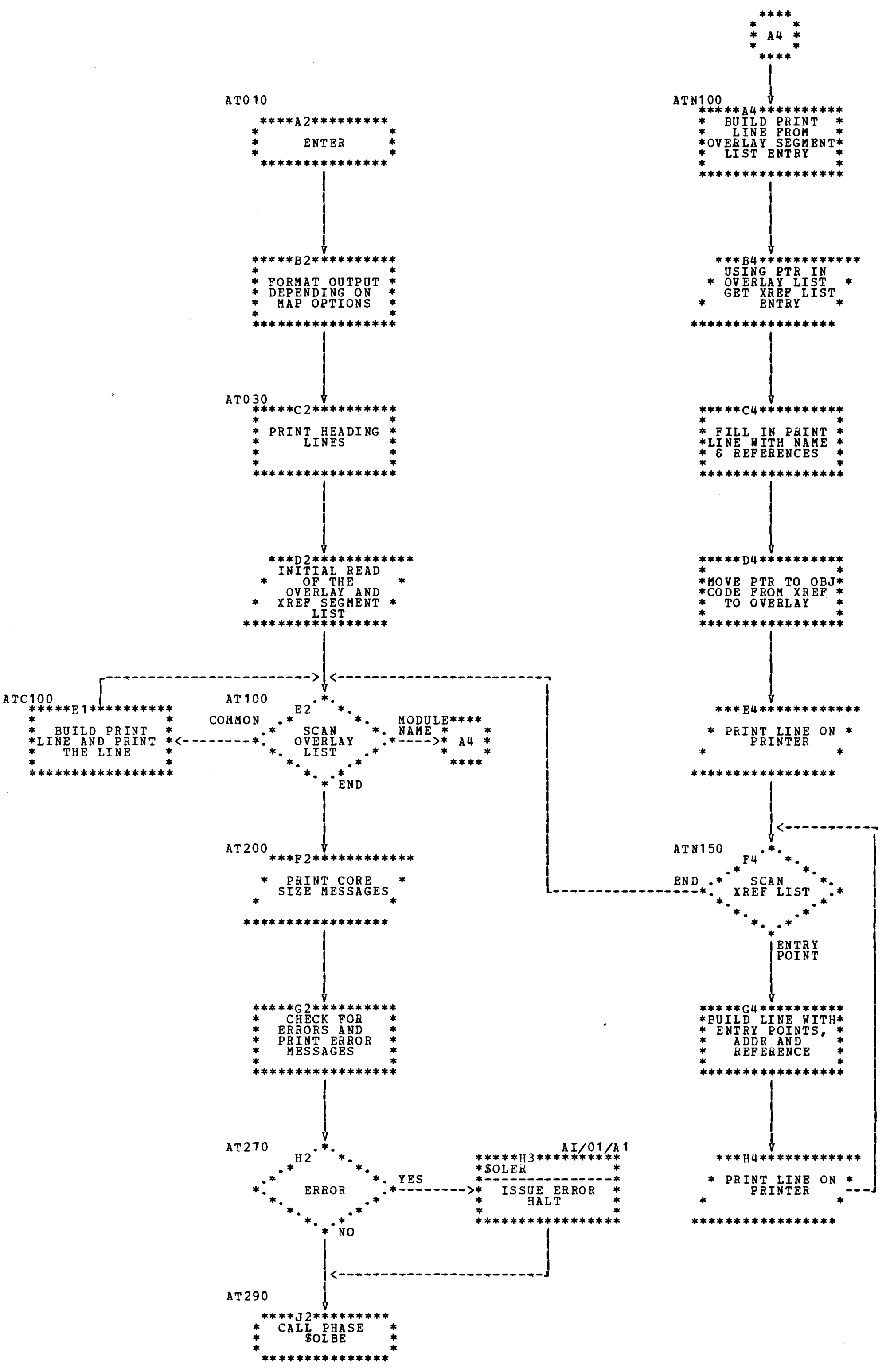
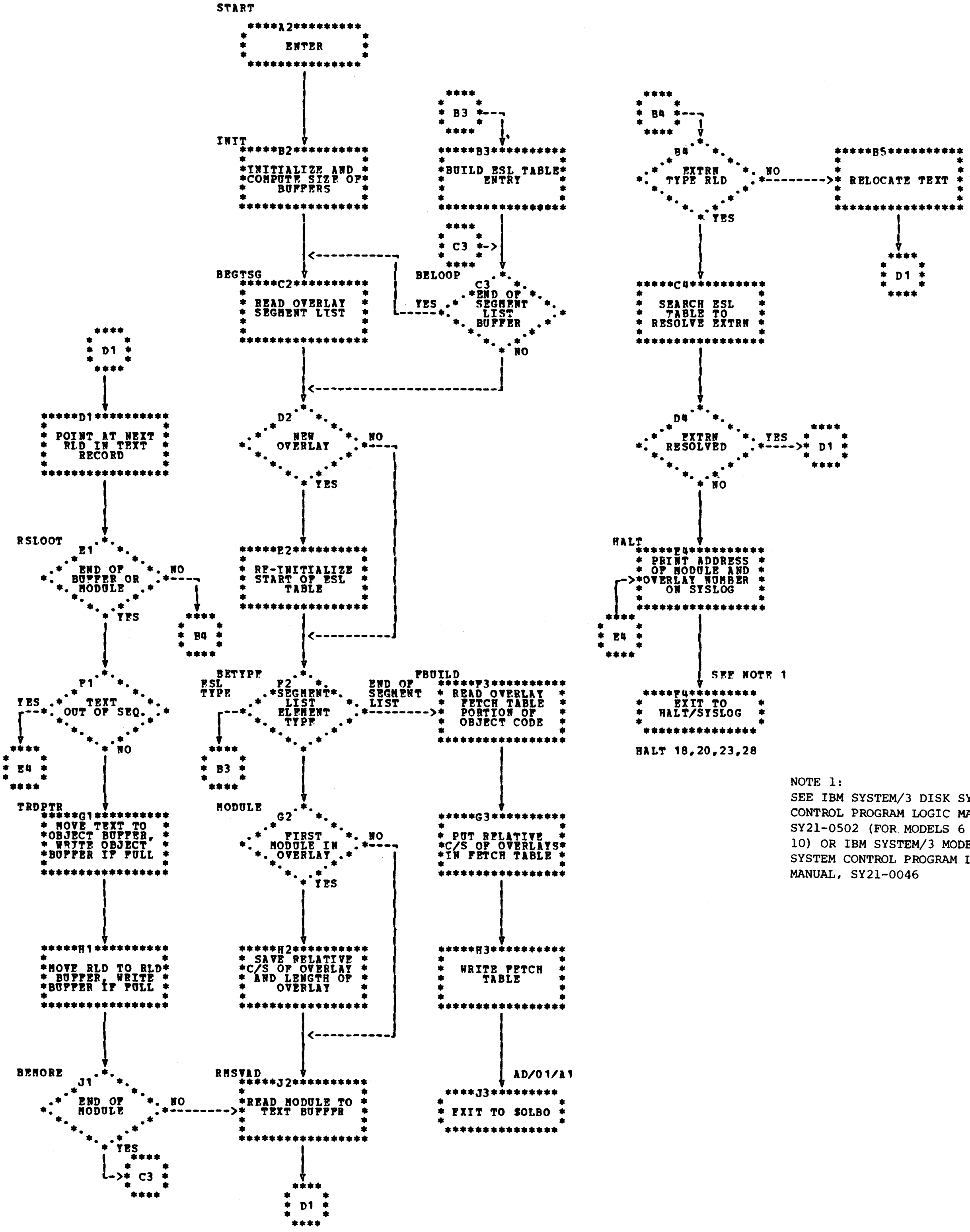
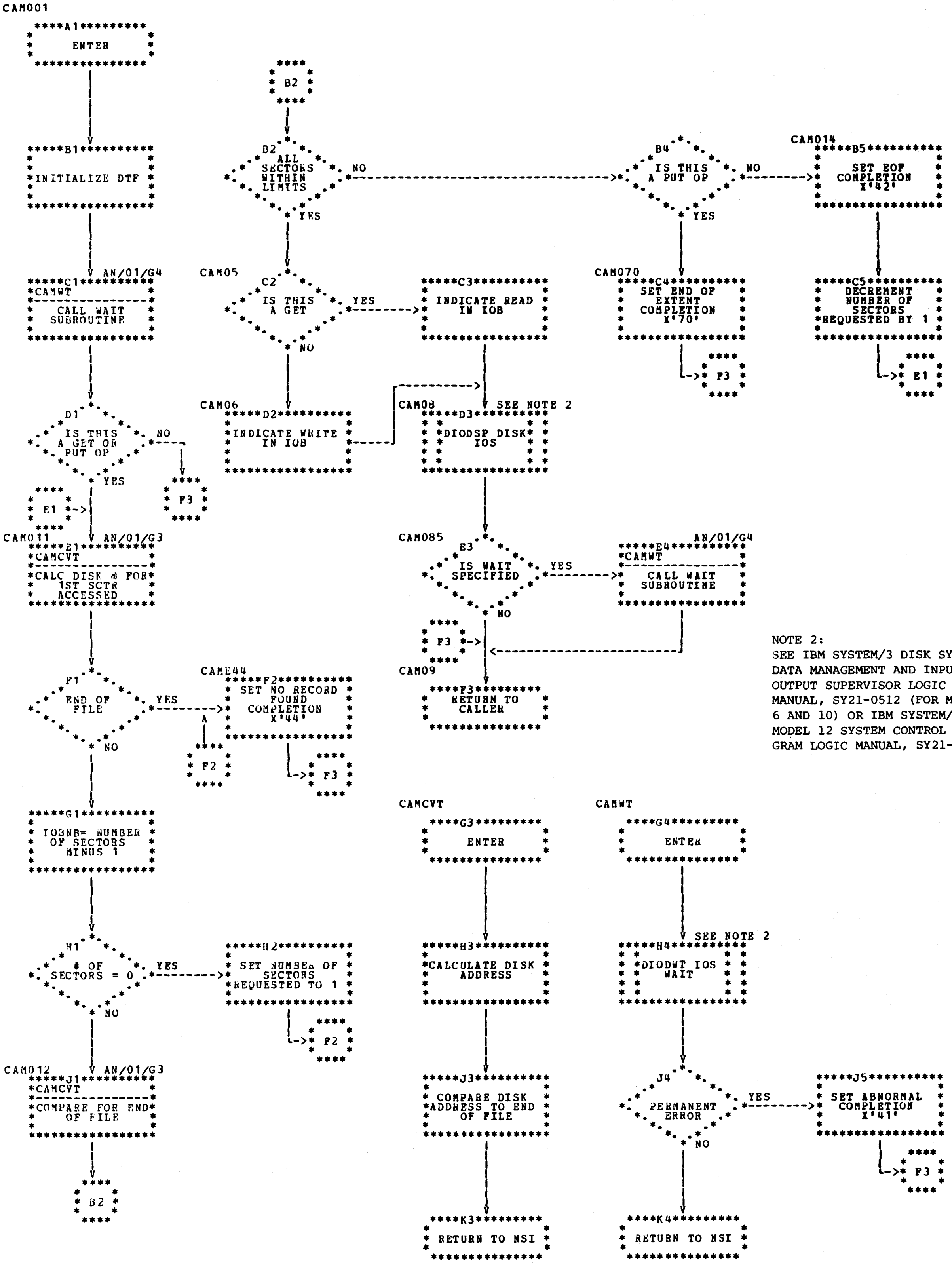

AT010
A2
ENTER
B2
FORMAT OUTPUT DEPENDING ON MAP OPTIONS
AT030
C2
PRINT HEADING LINES
D2
INITIAL READ OF THE OVERLAY AND XREF SEGMENT LIST
ATC100
E1
BUILD PRINT LINE AND PRINT THE LINE
AT100
E2
COMMON
SCAN OVERLAY LIST
MODULE NAME
A4
END
AT200
F2
PRINT CORE SIZE MESSAGES
G2
CHECK FOR ERRORS AND PRINT ERROR MESSAGES
AT270
H2
ERROR
YES
NO
AI/01/A1
H3
$OLER
ISSUE ERROR HALT
AT290
J2
CALL PHASE $OLBE
A4
ATN100
A4
BUILD PRINT LINE FROM OVERLAY SEGMENT LIST ENTRY
B4
USING PTR IN OVERLAY LIST GET XREF LIST ENTRY
C4
FILL IN PRINT LINE WITH NAME & REFERENCES
D4
MOVE PTR TO OBJ CODE FROM XREF TO OVERLAY
E4
PRINT LINE ON PRINTER
ATN150
F4
END
SCAN XREF LIST
ENTRY POINT
G4
BUILD LINE WITH ENTRY POINTS, ADDR AND REFERENCE
H4
PRINT LINE ON PRINTER

Chart AL. Core Map Phase ($OLAT)


START
A2
ENTER
INIT
B2
INITIALIZE AND COMPUTE SIZE OF BUFFERS
BEGTSG
C2
READ OVERLAY SEGMENT LIST
B3
B3
BUILD ESL TABLE ENTRY
C3
BELOOP
C3
END OF SEGMENT LIST BUFFER
YES
NO
D2
NEW OVERLAY
NO
YES
E2
RE-INITIALIZE START OF ESL TABLE
BETYPE
ESL TYPE
F2
SEGMENT LIST ELEMENT TYPE
END OF SEGMENT LIST
B3
FBUILD
F3
READ OVERLAY FETCH TABLE PORTION OF OBJECT CODE
MODULE
G2
FIRST MODULE IN OVERLAY
NO
YES
H2
SAVE RELATIVE C/S OF OVERLAY AND LENGTH OF OVERLAY
RMSVAD
J2
READ MODULE TO TEXT BUFFER
D1
G3
PUT RELATIVE C/S OF OVERLAYS IN FETCH TABLE
H3
WRITE FETCH TABLE
AD/01/A1
J3
EXIT TO $OLBO
D1
D1
POINT AT NEXT RLD IN TEXT RECORD
RSLOOT
E1
END OF BUFFER OR MODULE
NO
YES
B4
F1
TEXT OUT OF SEQ.
YES
NO
E4
TRDPTR
G1
MOVE TEXT TO OBJECT BUFFER, WRITE OBJECT BUFFER IF FULL
H1
MOVE RLD TO RLD BUFFER, WRITE BUFFER IF FULL
BEMORE
J1
END OF MODULE
NO
YES
C3
B4
B4
EXTRN TYPE RLD
NO
YES
B5
RELOCATE TEXT
D1
C4
SEARCH ESL TABLE TO RESOLVE EXTRN
D4
EXTRN RESOLVED
YES
D1
NO
HALT
E4
PRINT ADDRESS OF MODULE AND OVERLAY NUMBER ON SYSLOG
E4
SEE NOTE 1
F4
EXIT TO HALT/SYSLOG
HALT 18,20,23,28

NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart AM. Relocate, Resolve EXTRNs, and Build Load Module ($OLBE)



NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart AN. Complier Access Method ($CAM)

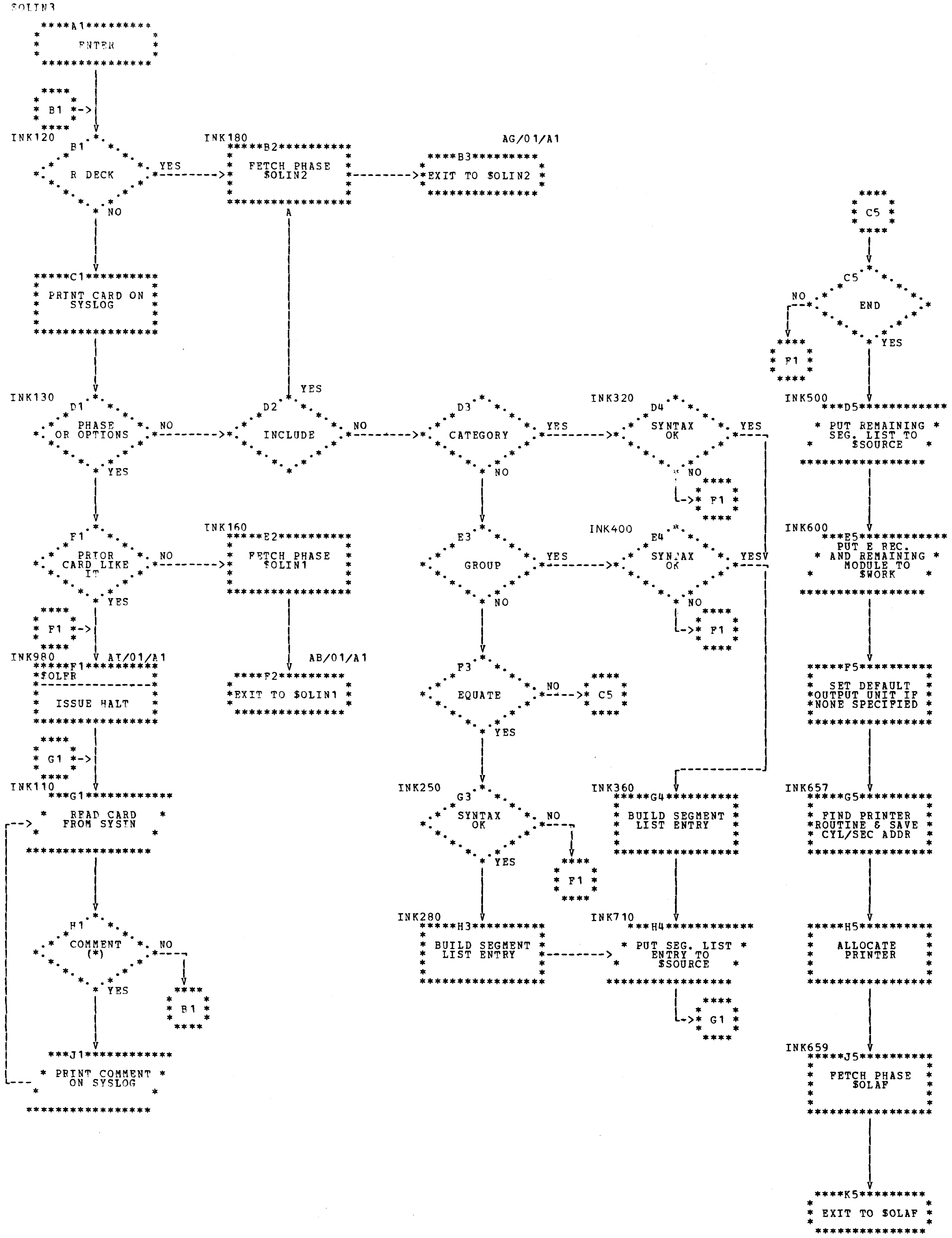

$OLIN3
A1 ENTER
B1
INK120 B1 R DECK
YES
NO
INK180 B2 FETCH PHASE $OLIN2
A
AG/01/A1
B3 EXIT TO $OLIN2
C1 PRINT CARD ON SYSLOG
INK130 D1 PHASE OR OPTIONS
NO
YES
D2 INCLUDE
YES
NO
D3 CATEGORY
YES
NO
INK320 D4 SYNTAX OK
YES
NO
F1
C5
C5 END
NO
YES
F1
INK500 D5 PUT REMAINING SEG. LIST TO $SOURCE
F1 PRIOR CARD LIKE IT
NO
YES
F1
INK160 E2 FETCH PHASE $OLIN1
E3 GROUP
YES
NO
INK400 E4 SYNTAX OK
YES
NO
F1
INK600 E5 PUT E REC. AND REMAINING MODULE TO $WORK
INK980 AT/01/A1
F1 $OLFR
ISSUE HALT
AB/01/A1
F2 EXIT TO $OLIN1
F3 EQUATE
NO
C5
YES
F5 SET DEFAULT OUTPUT UNIT IF NONE SPECIFIED
G1
INK110 G1 READ CARD FROM SYSIN
INK250 G3 SYNTAX OK
NO
YES
F1
INK360 G4 BUILD SEGMENT LIST ENTRY
INK657 G5 FIND PRINTER ROUTINE & SAVE CYL/SEC ADDR
H1 COMMENT (*)
NO
YES
B1
INK280 H3 BUILD SEGMENT LIST ENTRY
INK710 H4 PUT SEG. LIST ENTRY TO $SOURCE
G1
H5 ALLOCATE PRINTER
J1 PRINT COMMENT ON SYSLOG
INK659 J5 FETCH PHASE $OLAF
K5 EXIT TO $OLAF

Chart AO. User Entry Phase 4 ($OLIN3)

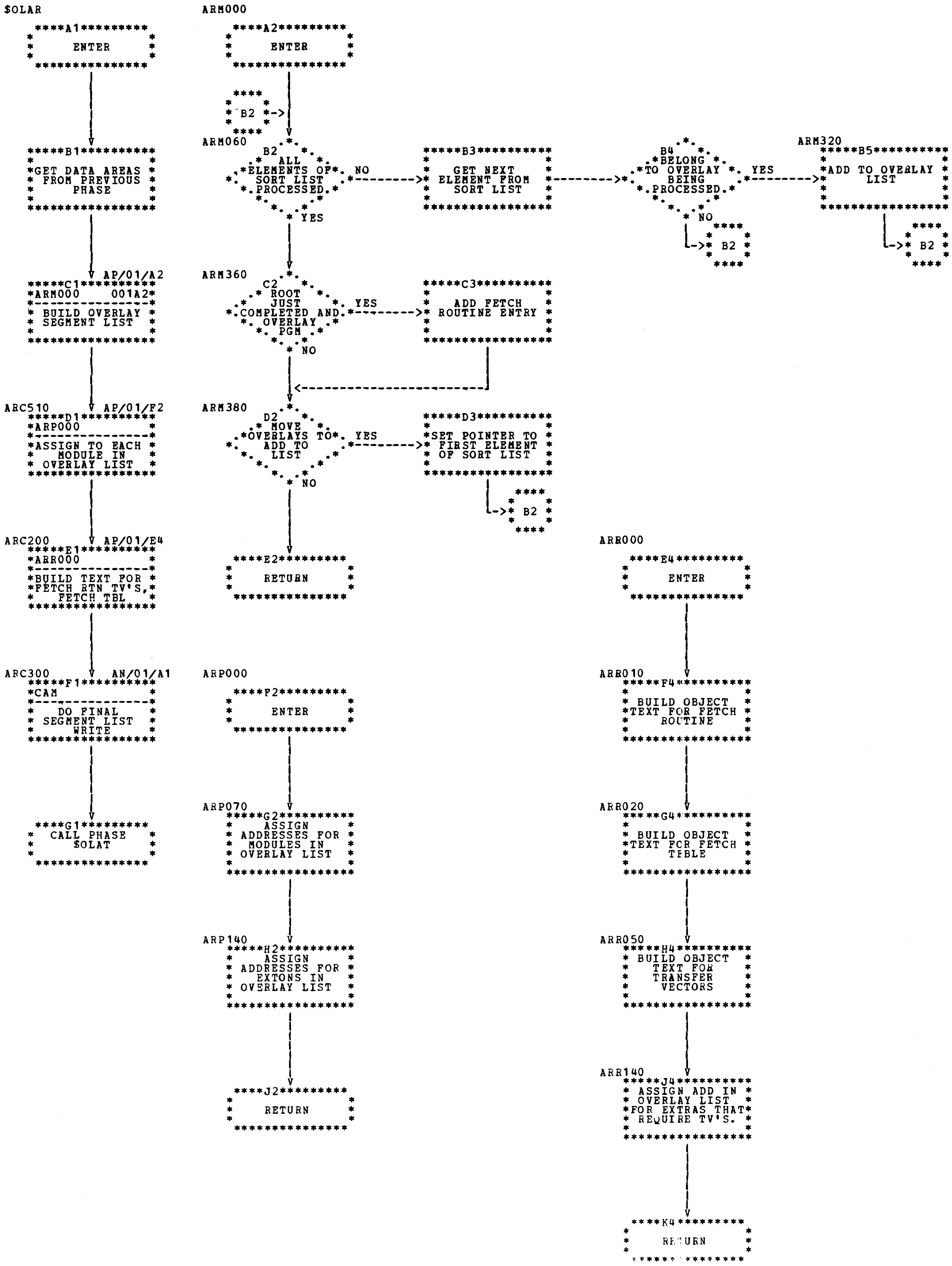
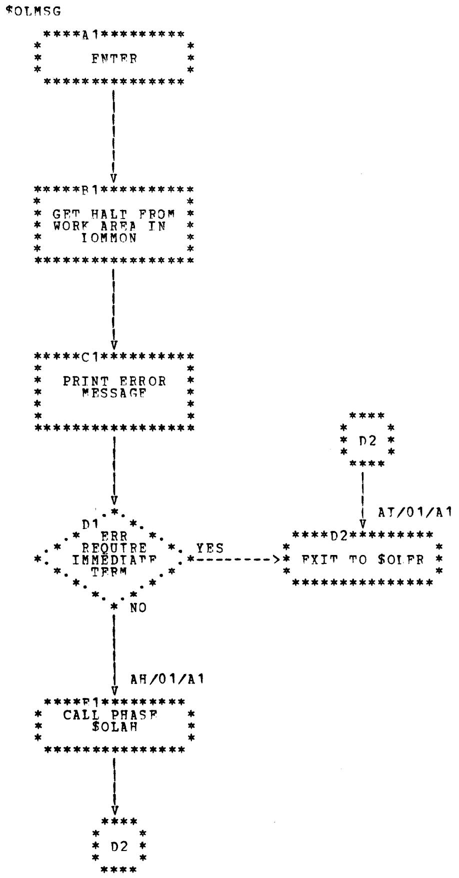

$OLAR
A1
ENTER
B1
GET DATA AREAS FROM PREVIOUS PHASE
AP/01/A2
C1
ARM000 001A2
BUILD OVERLAY SEGMENT LIST
ARC510
AP/01/F2
D1
ARP000
ASSIGN TO EACH MODULE IN OVERLAY LIST
ARC200
AP/01/E4
E1
ARR000
BUILD TEXT FOR FETCH RTN TV'S, FETCH TBL
ARC300
AN/01/A1
F1
CAM
DO FINAL SEGMENT LIST WRITE
G1
CALL PHASE $OLAT
ARM000
A2
ENTER
B2
ARM060
B2
ALL ELEMENTS OF SORT LIST PROCESSED
NO
YES
B3
GET NEXT ELEMENT FROM SORT LIST
B4
BELONG TO OVERLAY BEING PROCESSED
YES
NO
B2
ARM320
B5
ADD TO OVERLAY LIST
B2
ARM360
C2
ROOT JUST COMPLETED AND OVERLAY PGM
YES
NO
C3
ADD FETCH ROUTINE ENTRY
ARM380
D2
MOVE OVERLAYS TO ADD TO LIST
YES
NO
D3
SET POINTER TO FIRST ELEMENT OF SORT LIST
B2
E2
RETURN
ARP000
F2
ENTER
ARP070
G2
ASSIGN ADDRESSES FOR MODULES IN OVERLAY LIST
ARP140
H2
ASSIGN ADDRESSES FOR EXTONS IN OVERLAY LIST
J2
RETURN
ARR000
E4
ENTER
ARR010
F4
BUILD OBJECT TEXT FOR FETCH ROUTINE
ARR020
G4
BUILD OBJECT TEXT FOR FETCH TABLE
ARR050
H4
BUILD OBJECT TEXT FOR TRANSFER VECTORS
ARR140
J4
ASSIGN ADD IN OVERLAY LIST FOR EXTRAS THAT REQUIRE TV'S.
K4
RETURN

Chart AP. Overlay Segment List Build ($OLAR)



● Chart AR. Error Message Print Phase ($OLMSG)

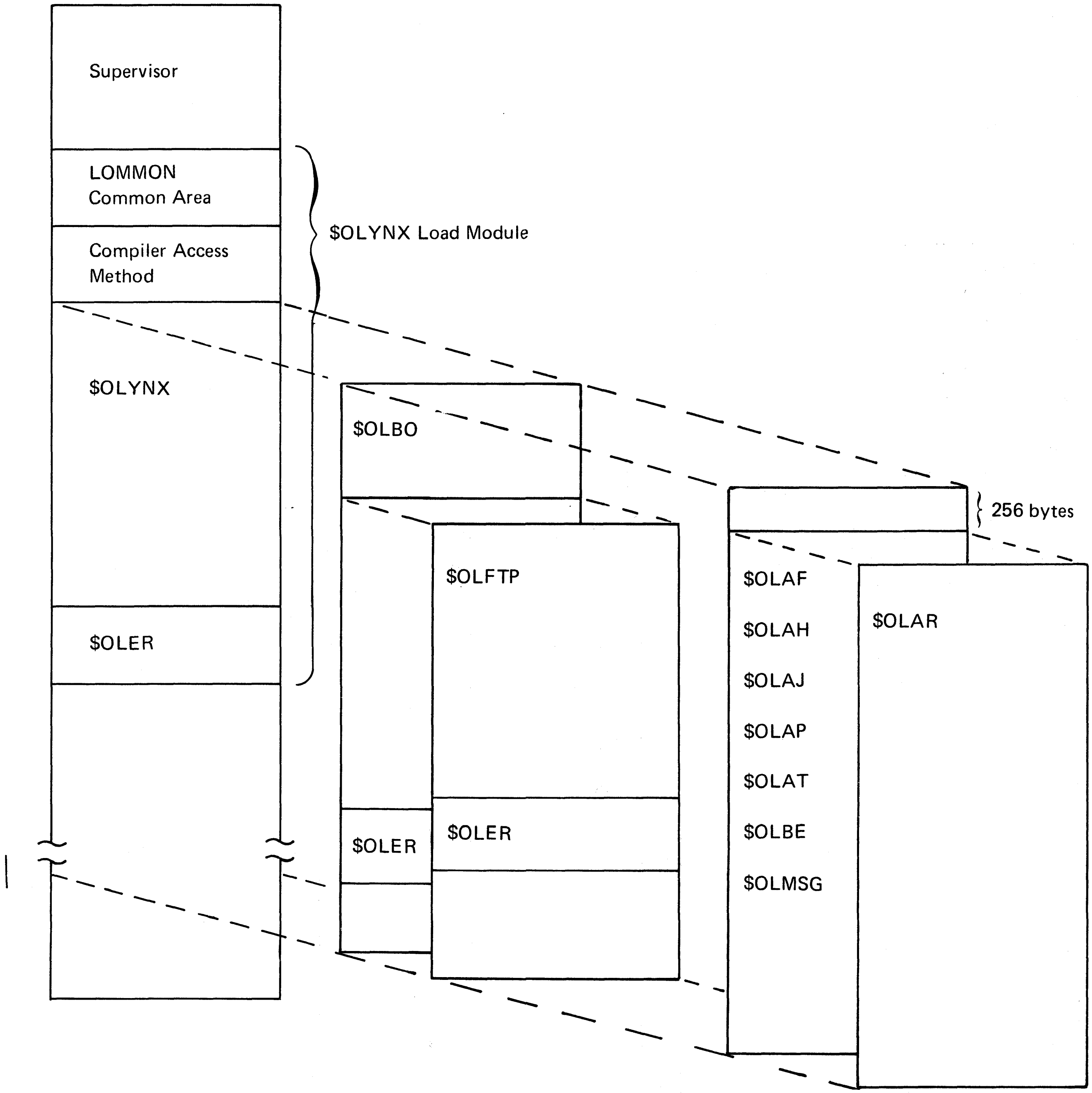


Figure 6. Compiler Entry Storage Map

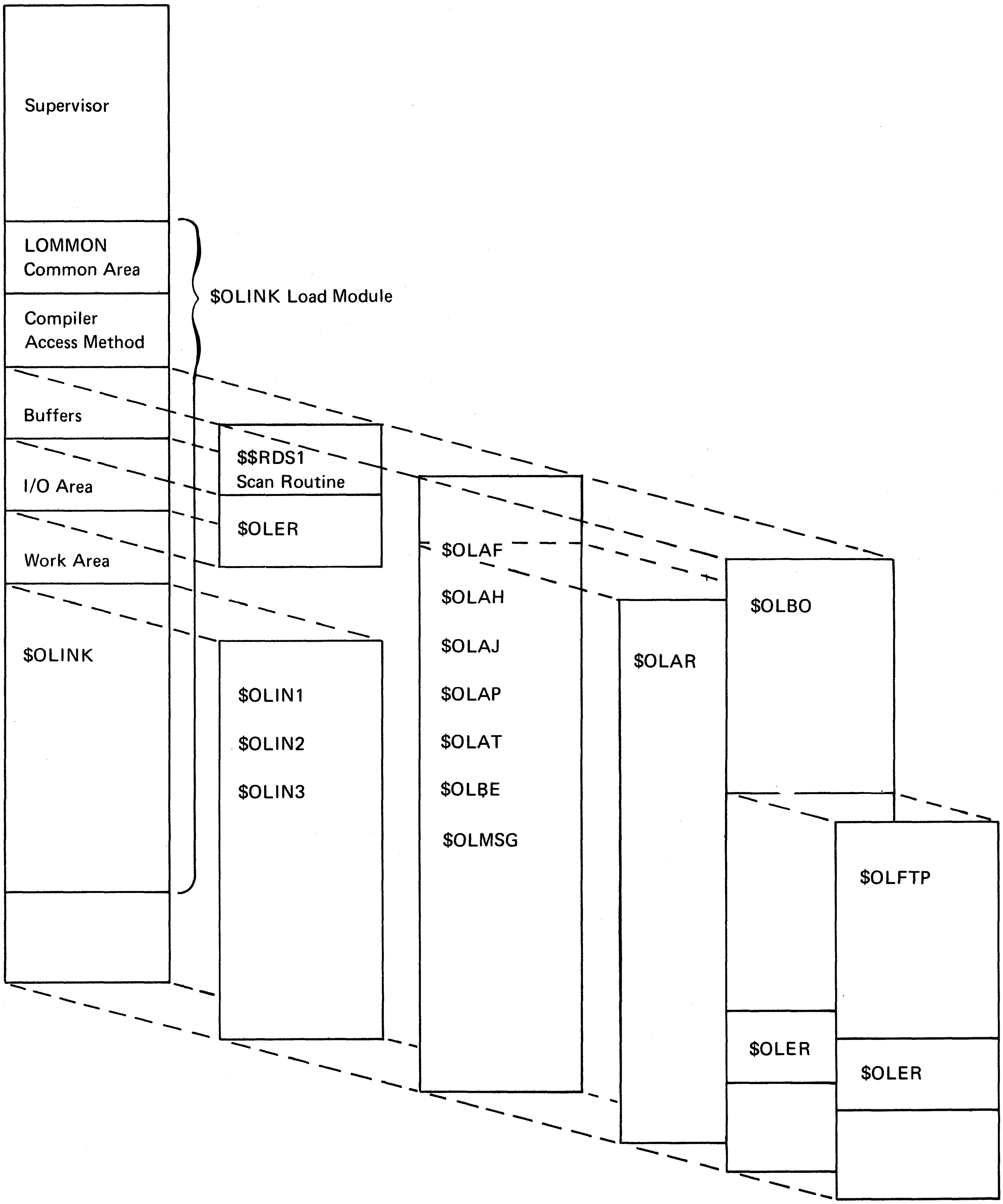


Figure 7. User Entry Storage Map

This section describes the data areas that pass information between routines of the Overlay Linkage Editor.

OVERLAY LINKAGE EDITOR COMMON (LOMMON)

The Overlay Linkage Editor common area (Figure 8) passes control information between the various routines. All of LOMMON, except for DTFs and IOBs, is initially set to zero by $OLINK or $OLYNX.

SEGMENT LIST ENTRIES

The various routines of the Overlay Linkage Edition build a series of segment lists. These segment lists are built in the $SOURCE work file (Figure 9). Each entry is 16 bytes long. The format of entries varies between and within segment lists depending on the type of entry. See Figures 10 through 14. Because all data fields in the segment list entries are not used for all types of entries, the column headed 'Applies to Segment Type' indicates which types of segment list entry will contain the data. Figure 11 lists all the segment types.

Displacement		Label	Length in Bytes	Routines that Change Data ($OLxxx)	Description
Hex	Decimal				
0	0	LODTFW	52	All	DTF for $WORK (see note)
34	52	LOIOBW	23	All	IOB for $WORK
4B	75	LODTFS	52	All	DTF for $SOURCE (see note)
7F	127	LOIOBS	23	All	IOB for $SOURCE
96	150	LORTYP	1	YNX, B0	R module information *Value* *Description* X'80' Punch R module X'40' R module to R1 X'20' R module to R2 X'10' R module to F1 X'08' R module to F2 X'04' Replace as a permanent entry X'02' Permanent entry X'01' R to program pack X'00' No R module
97	151	LOOTYP	1	INK, YNX, IN1, IN3, AT	O module information *Value* *Description* X'80' Punch O module X'40' O module to R1 X'20' O module to R2 X'10' O module to F1 X'08' O module to F2 X'04' No core map X'02' No cross-reference entry X'01' O module to program pack X'00' No O module
98	152	LOSWT1	1	INK, YNX, AF, IN1, IN2, IN3	Overlay Link Switch 1 *Value* *Description* X'80' Segment list in $SOURCE X'40' User call X'20' User specified overlays X'10' Start control addr set X'08' Groups in segment list X'04' Reserved X'02' Print messages X'01' Override 6E halt, Retain-R specified

Figure 8 (Part 1 of 3). Common Area (LOMMON)

Displacement		Label	Length in Bytes	Routines that Change Data ($OLxxx)	Description
Hex	Decimal				
99	153	LOSWT2	1	AF, AH, AJ, AR, AT, IN1, YNX	Overlay Link Error Switch *Value* *Description* X'80' System Cat called by User X'40' Not used X'20' An element in group is category 0 through 7 X'10' Not used X'08' Entry point not 0 in program with common X'04' No find on start control entry label X'02' Core size in OPTIONS record or program will not fit X'01' Terminal error
9A	154	LOUSPK	1	INK, YNX, IN1	User pack Q byte
9B	155	LONOVL	1	AP	Number of overlays
9C	156	LOPRPK	1	INK, YNX	Program pack Q byte
9D	157	LOEND@	2	INK, YNX	Addr of partition end
9F	159	LOCRSZ	2	AP, YNX, B0	Actual exec core size
A1	161	LORSV1	2		Reserved
A3	163	LOFTBL	2	AR	Displacement of OLFT
A5	165	LOAUTO	2	IN3	Relative entry number of auto segment list
A7	167	LOXREF	2	AF, BE	Relative entry number of X-ref segment list
A9	169	LOSORT	2	AH, BE	Relative entry number of sort segment list
AB	171	LOPRDA	2	INK, YNX	Directory addr on prog pack
AB	171	LOOVER	2	AJ, AP, BE	Relative entry number of overlay segment list
AD	173	LOUSDA	2	INK, YNX, IN1	Directory addr on user pack
AD	173	LOLIMT	2	AR, BE	Relative entry number of last delimiter
AF	175	LOWKCS	2	IN3, IN2, INK YNX, AF, AR	Relative sector number of next sector in $WORK
B1	177	LOLQBT	1	INK, YNX, BE	Q-byte of data pack
B2	178	LOLCSB	3	INK, YNX, BE	C/S addr of data start
B5	181	LOLHDR	1	INK, YNX, B0	Library type R or O
B6	182	LOLNAM	6	IN1, YNX, IN2, FTP, BO	Module name
BC	188	LOLLCS	2	BO	C/S of library entry
BE	190	LOLCAT	1	YNX	Overlay category of R
BE	190	LOLTXS	1	BE, B0	Number of text records in O
BF	191	LOLLEA	2	INK, YNX, AR, IN1	Link edit address
C1	193	LOLRLD	1	BE	RLD displacement
C2	194	LOLSCA	2	YNX, IN2, AJ, AR	Start control address
C4	196	LOLCSZ	1	INK, YNX, IN1, BO	Execution core size in 1/4K incr

Figure 8 (Part 2 of 3). Common Area (LOMMON)

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Routines that Change Data ($OLxxx)	Description
C5	197	LOATB1	2	IN1, IN2, AF, BE, B0, YNX, AJ	Attributes *Value Description* X'8000' Permanent entry X'4000' Inquiry X'2000' Inquiry invoking X'1000' Must run dedicated X'0800' Requires source X'0400' Deferred mounting allowed X'0200' Reserved, must be OFF X'0100' Reserved, must be OFF X'0080' System input dedication X'0040' Checkpoint restart program X'0020' Direct source read X'0010' Reserved X'0008' Reserved X'0004' Common X'0002' Reserved X'0001' Reserved
C7	199	LOLLVL	1	IN1, INK, YNX	Release level
C8	200	LOLTSC	2	YNX, BE	Total sector count
CA	202	LOWORK	31	Any	Phase work area
E9	233	LOCZER	2	INK, YNX	Constant of zero
EB	235	LOCONE	1	INK, YNX	Constant of one
EC	236	LOCHFF	2	INK, YNX	Constant X'FFFF'
EC	236	LOCM1	2	INK, YNX	Constant of minus one
EE	238	LOSCAT	1	AH	System category
EF	239	LOPTCS	2	YNX, IN3	C/S addr of printer routine
F1	241	LOERCD	1	AT, AF	Error code
F2	242	LOENTR	6	INK, IN1, AH, YNX	Entry point name save area

Note: Unused portions of the DTFs are used to store load lists of the linkage editor modules. These areas are:

Displacement Hex	Displacement Decimal	Label	Length in Bytes	Routines that Change Data ($OLxxx)	Description
02	02	LOAJ	6	AF	Load list $OLAJ
23	35	LOAP	6	AF	Load list $OLAP
29	41	LOAR	6	AF	Load list $OLAR
40	64	LOAT	6	AF	Load list $OLAT
6E	110	LOBE	6	AF	Load list $OLBE
74	116	LOBO	6	AF	Load list $OLBO
					Load List Format 2-byte Cyl/Sec Addr 1-byte No. of Sectors 1-byte RLD Displacement 2-byte Start Control Address

Figure 8 (Part 3 of 3). Common Area (LOMMON)

$SOURCE
Pre-Auto Segment List
Auto-link Segment List
Cross-Reference Segment List
Sort Segment List
Overlay Segment List

Segment List Entry Types

00	Module name
01	Entry point
02	EXTRN
03	Weak EXTRN
04	Global common
05	Local common
0B	EQUATE entry
0C	Transfer vector
0D	Reference a previous name or entry point
0E	GROUP entry
0F	CATEGORY entry
FE	Nulled entry
FF	End of segment list

Figure 9. Segment Lists in $SOURCE

Byte(s)	Bit(s)	Routine that Sets Data ($OLxxx)	Applies to Segment Type	Description
0	0-3	IN3	0E, 0F	Reserved
0	4-7	IN3	0E, 0F, 0B	Segment type (see Figure 9)
1		IN3	0E	Group number
1		IN3	0E	Category override number
2		AH	0E	Work area—original category
2-5		IN3	0E, 0F	Reserved
3	7	IN3	0E	User area specified for module
6-7		AF	0E, 0F	Reference number—pointer to module element in Auto-Link Segment List*
6-7		AJ	0E	Reference number—pointer to lead element in last overlay*
8-9		AF	0E, 0F	ESL Sequence Number
A-F		IN3	0E, 0F, 0B	Module name
A-B		AF	0E	Reserved
C-D		AJ	0E	Module element pointer (moved from bytes 6 through 7)
E-F		AF	0E	Reserved

* Segment displacement within $SOURCE.

Figure 10. Pre-Auto Segment List

Byte(s)	Bit(s)	Routine that Sets Data ($OLxxx)	Applies to Segment Type	Description
0	0	AF	00	COBOL entry
0	0	AF	01, 02, 03, 04, 05	Entry point or references an entry point
0	1	AF	02, 03	Resolved to module and/or entry
0	1	AF	00, 01	This module or entry point has an EXTRN referencing it
0	2	AF	00, 01, 02, 03	Work area must be OFF at phase end
0	2	AJ	00	Used—do not place in tree
0	3	AF	00	Calls a user routine or requires a transfer vector
0	3	AF	02, 04, 05	Delete this element when compressing list.
0	3	AJ	00	Module already placed in root
0	4-7	AF	All	Segment type (see Figure 9)
1		AF, AH	00, 01, 02, 03	Category
2-3		AF	00	$WORK address of object code
2-3		AF	01	Entry displacement from start of module
2		AH	00	Number of entry points
3	0	AH	00	Module requires boundary alignment
3	1	AH	00	Module calls a user routine
3	2	AH	00	Module has I/O dependency
3	3	AJ	00	Module already in an overlay
3	3	AJ	00	Substructure pointer already built
4-5		AF	00	Object code length
6-7		AF	00, 01, 02, 03, 04	Reference number—pointer to equal ESL number in Auto-Link Segment List
8-9		AF	00, 01, 02, 03, 04, 05	ESL number
A-F		AF, INK	00, 01, 02, 03, 04, 05	ESL name
A-B		AH	00	Reference number—pointer to equal O type in X-ref segment list
C-D		AJ	00	Work area
E-F		AH	00	Reserved

The Pre-Auto Segment List is included in this list and will appear as the first elements in the list.

Figure 11. Auto-Link Segment List

Byte(s)	Bit(s)	Routine that sets data ($OLxxx)	Applies to Segment Type	Description
0	0	AH	00, 01, 0D	Reserved
0	1	AH	00, 01	Reserved
0	2	AH	00, 01, 0D	Reserved
0	3	AH	00, 01,0D	Reserved
0	4-7	AH	00,01,0D	Segment type (see Figure 9)
1		AH	00,01,0D	Category
2-3		AH	00,01	Entry point displacement from start of module
4		AH	00	Work area = number of entry points on original category
5	0	AH	00	Module requires boundary alignment
5	0	AH	0D	Categories make this call a potential program failure
5	1	AH	00	Module calls a user routine
5	2	AH	00	Module has I/O dependency
5	3	AH	00	Work area - OFF at end of phase
5	4	AH	00	Work area - OFF at end of phase
5	5	AH	00	No reference made to this module
5	5	AH	01	Same name as module name
5	6	AH	00,01	Duplicate name
5	7	AH	00,01	Start control label
6-7		AH	00	Location of object text in $WORK (X'FFFF'=null text)
8-9		AH	00,01,0D	ESL number
A-F		AH	00,01,0D	Module or entry point name

Figure 12. Cross-Reference Segment List

Byte(s)	Bit(s)	Routine that Sets Data ($OLxxx)	Applies to Segment Type	Description
0	0-3	AP	0C	Set of modules already summed
0	1	AP	0C	Set of modules contains a boundry alignment module
0	0-3	AJ	00,02,04,05,0C	Reserved
0	4-7	AJ	00,02,04,05,0C	Segment type (see Figure 9)
1		AJ	00,02	Category
1		AP	00,0C	Overlay number
2		AJ	00	Number of entry points this module
2		AP	0C	Number of entry points this overlay
3	0	AJ	00	Module requires boundary alignment
3	1	AJ	00	Module calls a user routine
3	2	AJ	00	Module has I/O dependency
3	3	AP	00	Work area
3	4-7	AJ	00	Reserved
3		AJ	0C	Overlay area used by this set of modules at execution time
4-5		AJ	00,04,05	Length of object area associated with this ESL
4-5		AP	0C	Length of object area this overlay candidate
6-7		AJ	00	Reference number - pointer to equal module
6-7		AJ	02	Pointer to module
8-9		AJ	00,02,04,05	ESL number
A-B		AJ	00	Pointer to module name element in X-ref list
A-B		AP	0C	Pointer to next set of modules in same overlay
C-D		AJ	00	Chain to substructure referencing this module
C-D		AJ	02	Chain to other substructure and module
C-D		AJ	0C	Chain to last previous transfer vector element
E-F		AJ	02,04,05,0C	Reserved
E-F		AP	00	Boundry alignment adjustment factor

Figure 13. Sort Segment List

Byte(s)	Bit(s)	Routine that Sets Data ($OLxxx)	Applies to Segment Type	Description
0	0	AR	02	Work area = resolve to transfer vector
0	4-7	AR	00,02,04,05	Segment type (see Figure 9)
1		AR	00,02,04,05	Overlay Number
2-3		AR	00,02,04,05	Object time address for this ESL
4-5		AR	00,04,05	Object time length for this ESL
4-5		AR	02	Corresponding module type ESL number
6-7		AR	00	Address of module's first transfer vector
6-7		AT	00	$WORK location of object text
6-7		AR	02	3-byte RLD object time addr for this ESL
8-9		AR	00,02,04,05	ESL number
A-B		AR	00	Pointer to equal 0-type in X-ref segment list. FFFF designates overlay fetch routine
B		AR	02	Relative entry point position
C-D		AR	00	Overlay size - first 0 type of overlay only
C-D		AR	02	Pointer to 0 type entry in sort list
E-F		AR	00,02,04,05	Reserved

Figure 14. Overlay Segment List

APAR SUBMISSION

For APAR submission, the following information is necessary:

- Disk dumps of $WORK and $SOURCE
- Core dump
- Library directory dump of all routines used by the object program

OVERLAY FETCH ROUTINE

The Overlay Fetch routine is added to the root segment of every program that has overlays. It is built by routine $OLAR. When an overlay segment is needed during program execution, the Overlay Fetch routine is called. It fetches overlay segments from access devices and places them in the overlay regions in main storage. Bits are set in the overlay fetch table (Figure 15) telling which overlay region is used.

The Overlay Fetch routine requires three parameters as input:

1. Overlay number (one byte)
2. Entry address of the overlay (two bytes)
3. Return address from the overlay (two bytes)

A transfer vector is built for each overlay in an object program. Transfer vectors provide input parameters for the Overlay Fetch routine. Overlay Linkage Editor routine $OLAR builds transfer vectors. Figure 16 shows the format of transfer vectors.

Relative C/S @	Number of Sectors TEXT	Core Load Address	RLD	Flag
0 1	2	3 4	5	6

Bytes *Contents*

0-1 – Relative cylinder/start address of the overlay segment.
This is the number of cylinder/sectors past the C/S @ of the root segment of the overlay program as given in the object library directory entry for the program.

2 – Number of sectors of text in the load module. (Does not include the number of related RLD sectors.)

3-4 – Relative main storage start address of where the overlay segment is to be placed in main storage by the system loader. (Relative to the end of the Supervisor address.)

5 – RLD start displacement.

6 – Flag byte – used at execution time by the root segments Overlay Fetch routine.

X'80'	Overlay in core
X'40'	Non-I/O calling area
X'20'	System area
X'10'	I/O calling area
X'OF'	reserved

Figure 15. Overlay Fetch Table Entry Format

```
ST    OVFRS1,ARR      Save the return address
B     OVFR            Call the Overlay Fetch routine
DC    XL1 'NN'        One byte containing the overlay
                      number
DC    AL2 (entry)     Two-byte entry address
```

Figure 16. Transfer Vector Format

The Overlay Fetch routine checks to see if the requested overlay segment is already in main storage. If it is, the routine branches to the entry address of the overlay; if not, the overlay fetch table entries are checked to see if they use the same main storage. If they do, the overlay is flagged as not being in main storage.

After the Overlay Fetch routine checks all entries in the overlay fetch table, it sets the 'overlay in core' bit in the overlay fetch table entry for the requested overlay. The Overlay Fetch routine then loads the overlay segment and branches to its entry address. Figure 17 describes the Overlay Fetch routine.

Overlay Fetch Table

The overlay fetch table is built by routine $OLAR. It contains one 7-byte entry for each overlay in the program. Figure 15 shows the format of an overlay fetch table entry.

How to Find an Overlay

When a process check occurs, the following steps can determine which overlays are in main storage and where to find them.

1. Locate the address of the Overlay Fetch routine on the core usage map of the source listing (Figure 18).

2. Locate the overlay fetch table in the dump. The overlay fetch table is 115 bytes past the start address of the Overlay Fetch routine. It can be obtained by this hex formula: Address of Overlay Fetch routine +X'73'=overlay fetch table (Figure 19).

3. Mark off every 7-byte entry in the overlay fetch table until the last entry is reached. The last entry is X'FF' (Figure 19).

4. Number each entry left to right, starting with number 1. Each entry refers to an overlay (Figure 18).

5. Look at the seventh byte in each entry. This is the flag byte. The first bit will be on for every overlay in storage at the time of the dump (Figure 19).

6. Compare the numbers you gave the overlays in storage at the time of the dump with the number of the overlays in the core usage map (Figure 18). This gives the names and addresses of the segments within the overlays which were in storage at the time of the dump (Figure 19).

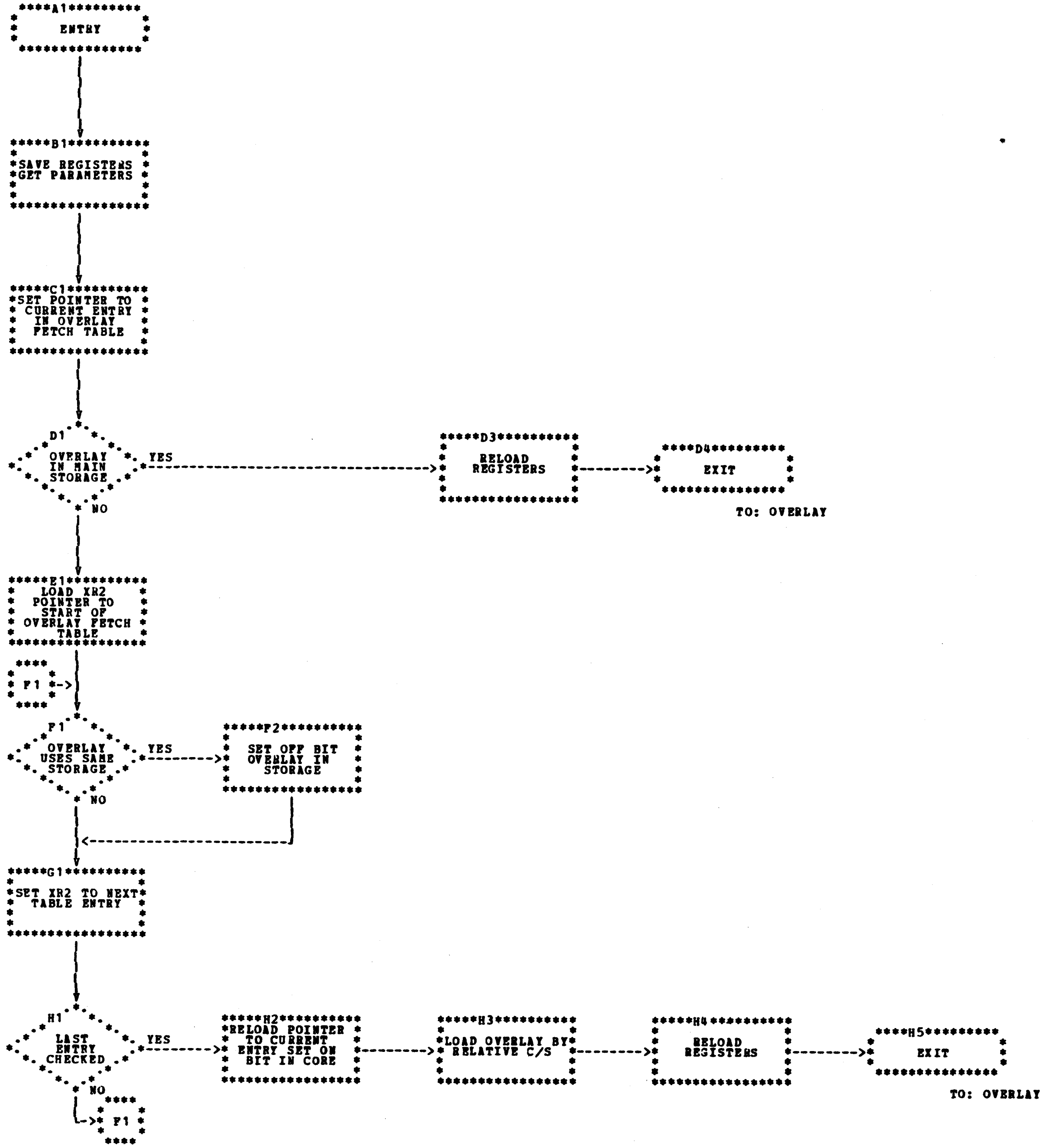
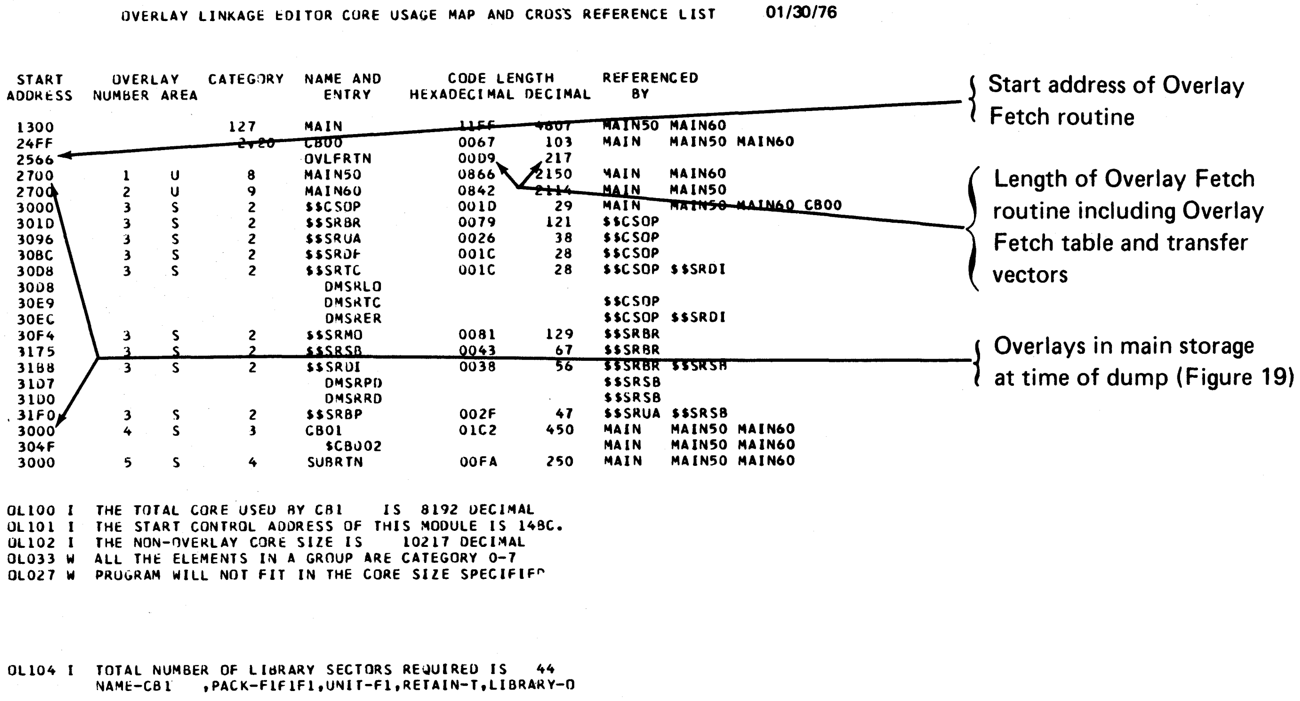


Figure 17. Overlay Fetch Routine



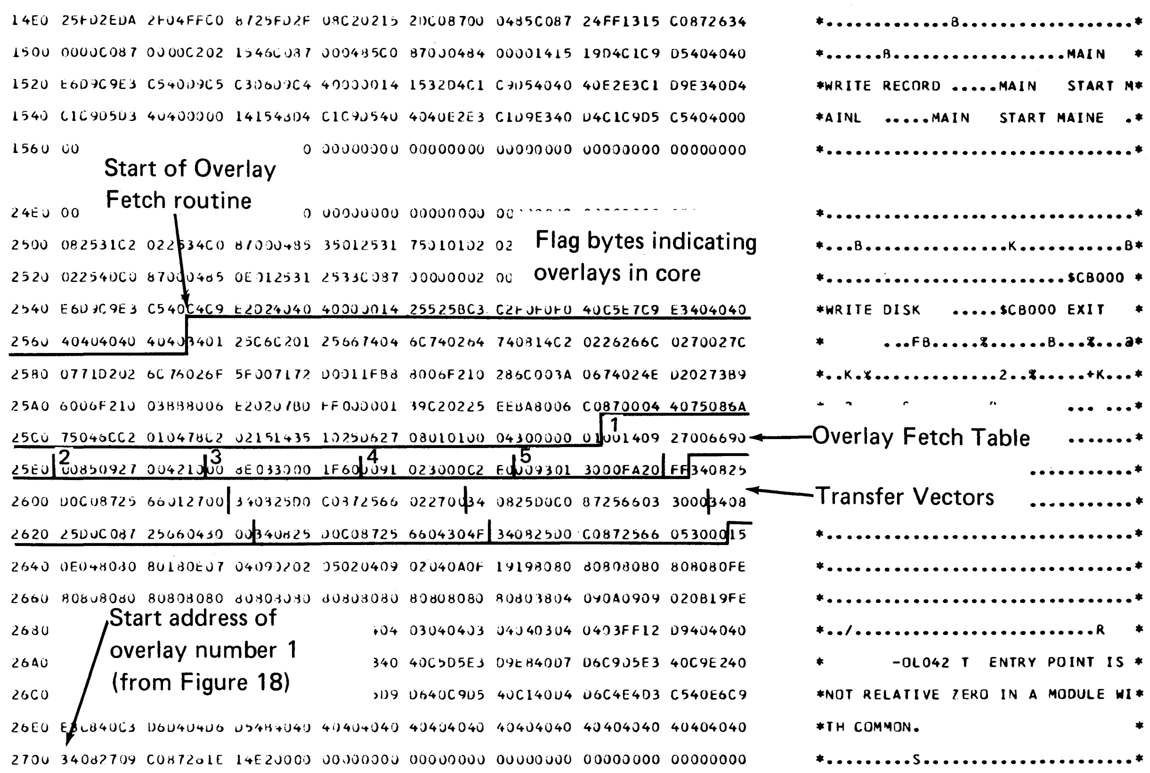
```
          OVERLAY LINKAGE EDITOR CORE USAGE MAP AND CROSS REFERENCE LIST      01/30/76


START     OVERLAY    CATEGORY  NAME AND      CODE LENGTH       REFERENCED
ADDRESS  NUMBER AREA             ENTRY   HEXADECIMAL DECIMAL     BY

1300                   127     MAIN          11FF      4607   MAIN50 MAIN60
24FF                   2,20    CB00          0067       103   MAIN    MAIN50 MAIN60
2566                           OVLFRTN       00D9       217
2700      1     U        8     MAIN50        0866      2150   MAIN    MAIN60
2700      2     U        9     MAIN60        0842      2114   MAIN    MAIN50
3000      3     S        2     $$CSOP        001D        29   MAIN    MAIN50 MAIN60 CB00
301D      3     S        2     $$SRBR        0079       121   $$CSOP
3096      3     S        2     $$SRUA        0026        38   $$CSOP
30BC      3     S        2     $$SRDF        001C        28   $$CSOP
30D8      3     S        2     $$SRTC        001C        28   $$CSOP $$SRDI
30D8                             DMSRLO
30E9                             DMSRTC                       $$CSOP
30EC                             DMSRER                       $$CSOP $$SRDI
30F4      3     S        2     $$SRMO        0081       129   $$SRBR
3175      3     S        2     $$SRSB        0043        67   $$SRBR
31B8      3     S        2     $$SRDI        0038        56   $$SRBR $$SRSB
31D7                             DMSRPD                       $$SRSB
31D0                             DMSRRD                       $$SRSB
31F0      3     S        2     $$SRBP        002F        47   $$SRUA $$SRSB
3000      4     S        3     CB01          01C2       450   MAIN    MAIN50 MAIN60
304F                             $CB002                       MAIN    MAIN50 MAIN60
3000      5     S        4     SUBRTN        00FA       250   MAIN    MAIN50 MAIN60

OL100 I   THE TOTAL CORE USED BY CB1    IS  8192 DECIMAL
OL101 I   THE START CONTROL ADDRESS OF THIS MODULE IS 148C.
OL102 I   THE NON-OVERLAY CORE SIZE IS    10217 DECIMAL
OL033 W   ALL THE ELEMENTS IN A GROUP ARE CATEGORY 0-7
OL027 W   PROGRAM WILL NOT FIT IN THE CORE SIZE SPECIFIE^


OL104 I   TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS   44
          NAME-CB1   ,PACK-F1F1F1,UNIT-F1,RETAIN-T,LIBRARY-0
```

Start address of Overlay Fetch routine

Length of Overlay Fetch routine including Overlay Fetch table and transfer vectors

Overlays in main storage at time of dump (Figure 19)

Figure 18. Core Usage Map and Cross-Reference List



```
14E0 25F02EDA 2F04FFC0 8725F02F 08C20215 20C08700 0485C087 24FF1315 C0872634     *.............B.................*
1500 0000C087 0000C202 1546C087 000485C0 87000484 00001415 19D4C1C9 D5404040     *......B..................MAIN  *
1520 E6D9C9E3 C540D9C5 C306D9C4 40000014 1532D4C1 C9D54040 40E2E3C1 D9E340D4     *WRITE RECORD .....MAIN   START M*
1540 C1C9D5D3 40400000 14154304 C1C9D540 4040E2E3 C1D9E340 D4C1C9D5 C5404000     *AINL  .....MAIN   START MAINE  .*
1560 00                         0 00000000 00000000 00000000 00000000 00000000     *...............................*

24E0 00                         0 00000000 00000000 00                             *...............................*
2500 082531C2 022534C0 87000485 35012531 75010102 02                               *...B...............K...........B*
2520 022540C0 87000485 0E012531 2533C087 00000002 00                               *.........................$CB000 *
2540 E6D9C9E3 C540C4C9 E2D24040 40000014 25525BC3 C2F0F0F0 40C5E7C9 E3404040     *WRITE DISK   .....$CB000 EXIT   *
2560 40404040 40403401 25C6C201 25667404 6C740264 740814C2 0226266C 0270027C     *      ...FB.....%......B...%...@*
2580 0771D202 6C76026F 5F007172 D0011FB8 8006F210 286C003A 0674024E D20273B9     *..K.%.............2..%.....+K...*
25A0 6006F210 03B88006 E20207BD FF000001 39C20225 EEBA8006 C0870004 4075086A     *                           ... ...*
25C0 75046CC2 010478C2 02151435 10250627 08010100 04300000 01001409 27006690                                ......*
25E0 00850927 00421000 8E033000 1F600091 023000C2 E0009301 3000FA20 FF340825                                ..........*
2600 D0C08725 66012700 340325D0 C0372566 02270034 0825D0C0 87256603 30003408                                ..........*
2620 25D0C087 25660430 00340825 D0C08725 6604304F 340825D0 C0872566 05300015     *...............................*
2640 0E048030 80180E07 04090202 05020409 02040A0F 19198080 80808080 808080FE     *...............................*
2660 80808080 80808080 80808080 80808080 80808080 80803804 090A0909 020B19FE     *...............................*
2680                            404 03040403 04040304 0403FF12 D9404040     *../.......................R    *
26A0                            340 40C5D5E3 D9E840D7 D6C9D5E3 40C9E240     *        -OL042 T  ENTRY POINT IS *
26C0                            5D9 D640C9D5 40C140D4 D6C4E4D3 C540E6C9     *NOT RELATIVE ZERO IN A MODULE WI*
26E0 E3C840C3 D6D4D4D6 D54B4040 40404040 40404040 40404040 40404040 40404040     *TH COMMON.                      *
2700 34082709 C08726 1E 14E20000 00000000 00000000 00000000 00000000 00000000     *.........S......................*
```

Start of Overlay Fetch routine
Flag bytes indicating overlays in core
1
2
3
4
5
Overlay Fetch Table
Transfer Vectors
Start address of overlay number 1 (from Figure 18)

Figure 19 (Part 1 of 2). Sample Core Dump

```
2EC0 C0000000 00000000 00000000 00000000 00000000 00000000 0000C202 2F1BC087    *..........................B.....*
2FE0 000485C2 012EEF34 012905C0 8728E0C0 8724FF27 15C08726 342F1AC2 022F34C0    *...B.......................B....*
2F00 87000485 C08714E7 C2022F40 C0870004 85C08726 2926082E DA000000 00142F20    *.......XB.......................*
2F20 D4C1C9D5 F5F040E2 E3C1D9E3 40C1F5F0 40404040 0000142F 39D4C1C9 D5F5F040    *MAIN50 START A50    .....MAIN50 *
2F40 C5D5C440 C1F5F040 40404040 40000014 2F52D4C1 C9D5F5F0 40C5D5E3 C5D940C3    *END A50      .....MAIN50 ENTER C*
2F60 F5F04040 40400304 808080FE 80808080 80808080 80808080 56090404 04020402    *50    ..........................*
2F80 04C                       04 04030404 03040403 04040304 0403FF12 D9404040    *............................R   *
2FA0 404                       F4 F240E340 40C5D5E3 D9E84007 D6C9D5E3 40C9E240    *       -OL042 T  ENTRY POINT IS *
2FC0 D5                        C5 40E9C5D9 D640C9D5                   40E6C9    *NOT RELATIVE ZERO IN A MODULE WI*
2FE0 E3C                       40 40404040 4040404                    404040    *TH COMMON.                      *
3000 F2870658 C3C2F0F0 F1340830 76C20230 77C0870C                      50200    *2..$CB001....B............K.....*
3020 85020134 02304E75 020080FF 04F20112 750100D2 01037502 00E20203 020102B4    *......+......2.....K.....S..K...*
3040 01C0C202 30A9C087 000485C0 872EDA34 083076C2 023090C0 87000485 35013076    *..B................B............*
3060 D2020375 01C13401 30722C01 307400CC 87000000 00270800 0014307C 5BC3C2F0    *K..........................@$CB0*
3080 F0F140C5 D5E3C5D9 40404040 40404040 00001430 955BC3C2 F0F0F140 C5D5E3C5    *01 ENTER        .....$CB001 ENTE*
30A0 D9405BC3 C2F0F0F2 40000014 30AE5BC3 C2F0F0F1 4040C5E7 C9E34040 40404040    *R $CB002 .....$CB001  EXIT      *
30C0 40400000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    *  ..............................*
30E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    *................................*

XR1   XR2   PSR   LPIAR LPDAR MPTAR MRDAR MPCAR DFCR  DFDR  IAR2  ARR2  XR1-2 XR2-2 PSR2
2EEF  2F19  0001  0428  037F  1E60  13A0  1E20  0F73  3200  5F04  5F1A  5F10  0563  0004
TRANSIENT AREA
0100 F28705E2 E3D6D706 340102CB 340802CF 34020358 C0870004          *2..STOP.................*
0118 00880458 35020358 B8400089 8000C010 028889C0 00380100          *......... ..............*
0130 F210C235 010011B9 C0008820 00F2100A 78801379 8016C090          *2.B..........2..........*
0148 02837A80 163C4003 FF0C8203 FE03FFC0 8702D035 0100117D          *...... ................'*
```

Start address of overlay number 4 (from Figure 18)

Entry point of overlay number 4 (from Figure 18)

Figure 19 (Part 2 of 2). Sample Core Dump

MESSAGES

The Overlay Linkage Editor issues informational messages and error messages. Figure 20 lists these message numbers with the routines that issue them. For a full explanation of these messages, see *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561.

Message Number	Issued by Routine
OL016 W	$OLAT
OL021 T	$OLAF
OL022 T	$OLAF
OL025 T	$OLAT
OL026 T	$OLAT
OL027 W	$OLAT
OL029 W	$OLAT
OL031 W	$OLAT
OL032 W	$OLAF
OL033 W	$OLAT
OL034 W	$OLAF
OL035 T	$OLAT
OL036 W	$OLAT
OL038 T	$OLAF
OL042 T	$OLAT
OL100 I	$OLAT
OL101 I	$OLAT
OL102 I	$OLAT
OL103 I	$OLBO
OL104 I	$OLBO

Figure 20. Message Numbers with Issuing Routine

PART II

CHECKPOINT/RESTART

SECTION 1. INTRODUCTION

Checkpoint/Restart enables the user to restart a checkpointed program from the last checkpoint taken, provided no intervening program executions take place.

OPERATING CHARACTERISTICS

Checkpoint/Restart depends on the System/3 SCP Programming support and the Overlay Linkage Editor support of the OPTIONS statement.

Checkpoint/Restart requires that the system IPL pack scheduler workarea contain the additional tracks for a Roll-in/Roll-out area. This area is available if either the inquiry capability or the Checkpoint/Restart feature is included at system generation time.

SECTION 2. PROGRAM ORGANIZATION

Checkpoint/Restart enables the user to restart a checkpointed program. Checkpoint is a means of recording the status of a problem program at certain intervals (checkpoints). After an error occurs, Restart can resume execution of a checkpointed program from the last checkpoint before the error. Restart is not allowed after a controlled cancel or normal end of job.

CHECKPOINT

Figure 21 shows an overview of Checkpoint. (An operational diagram legend is included.) Figure 22 is the storage map.

$$STKP (Checkpoint - Main Load) is entered from the supervisor as a result of a find and fetch request. Before a checkpoint can be made, $$STKP awaits completion of all pending non-tape I/O operations. Upon completion, $$STKP then gives control to $$STKQ (Checkpoint-Quiesce Magnetic Tape I/O) if tapes are used; otherwise, control goes to $$STKR (Checkpoint-Problem Program and SWA Load) to de-activate the checkpoint area and save the SWA and checkpointed program on disk. $$STKT (Checkpoint-Final Load) can be called to store the checkpoint information, then restore the checkpointed program and re-activate the checkpoint area. At this time the checkpointed program is given control to continue processing.

If errors occur, control returns to the checkpointed program with a completion code of X'41'. If an unrecoverable disk error occurs, control passes to the end-of-job transient, $$SPEJ.

The position of tapes can be saved only if the tapes were opened and allocated by SCP support. Checkpoint modules use fields NPSCHA (set by ALLOCATE) and NPDTF@ (set by OPEN) in the program level 1 communication region (N1COMN) to determine if tapes are used by the program and to save the following information from the DTFs:

- Q-code

- SWA Format 1 (F1) numbers

- Current block count

This information is used by RESTART to reposition the tapes. Checkpoint/Restart cannot reposition BTAM tape files or files accessed directly via tape IOS.

CHECKPOINT LINKAGE

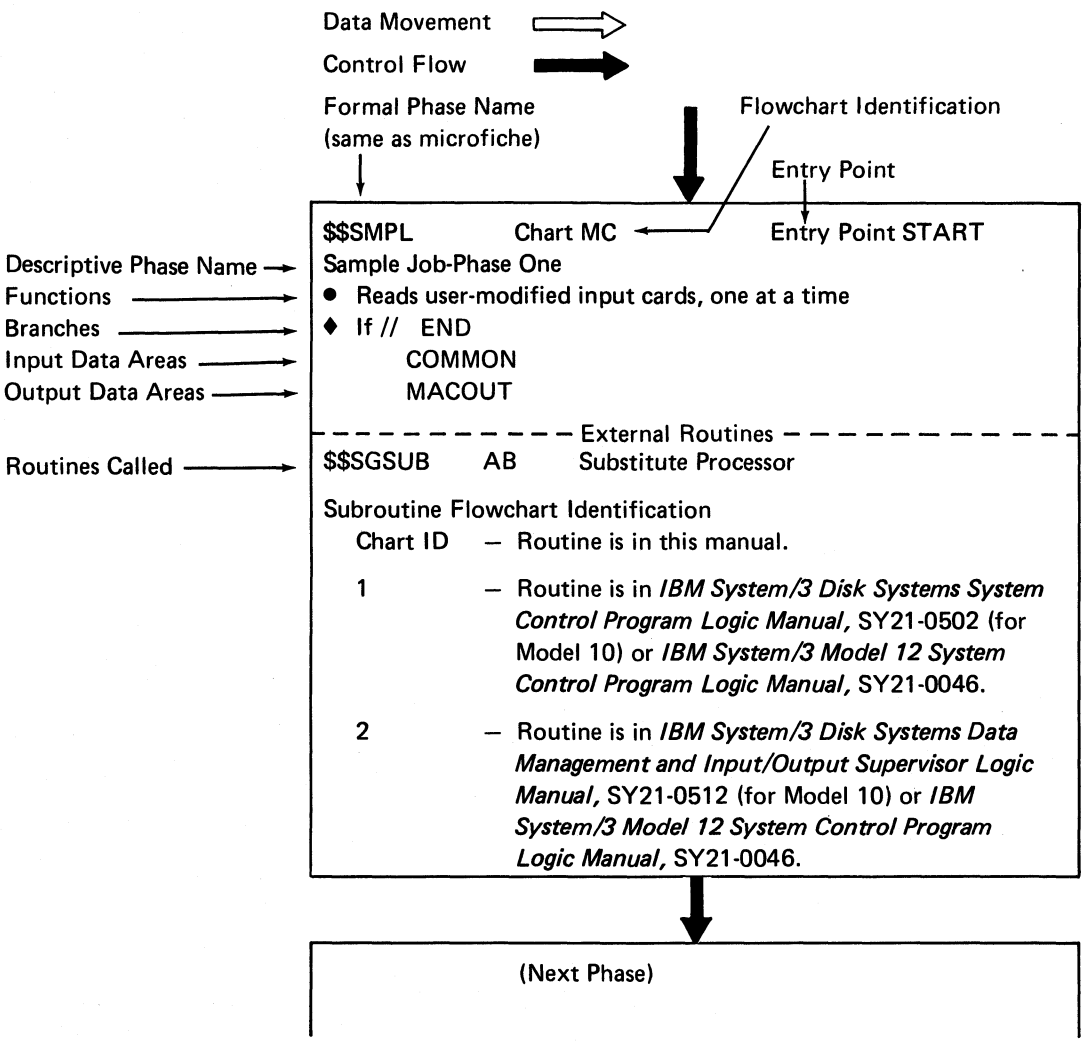
To use Checkpoint, the checkpoint attribute (bit 1 of the second attribute byte) must be set in the object library directory entry for the program to be checkpointed. This attribute is set from the information in the object deck header card if the program is put in the object library by the Library Maintenance program. The checkpoint attribute is mutually exclusive with the inquiry invoking attribute.

To call Checkpoint, the following linkage convention is required:

```
         LA   FDPARM,2       Pointer to find para-
                             meter list
         B    NCENTR         General entry
         DC   XL1'81'        Find
         CLI  0(,2),C'O'     $$STKP FOUND?
         BE   ERRTN          NO, GO TO USER
                             ERROR RTN
         MVC  CS,1(2,2)      Set up C/S for fetch
         B    NCENTR         General entry
         DC   XL1'80'        Fetch
CS       DC   CL2'00'        C/S of $$STKP from
                             object library into
                             transient area
         DC   XL1'02'        (three sectors)
         B    CONTIN         Return from $$STKP
         DS   XL1            Completion code
RESTRT   EQU  *              Restart entry point
          .                  (Any setup/reposi-
          .                  tioning necessary to
          .                  continue after
                             RESTART)
FDPARM   EQU  *              Find parameter list
         DC   CL7'0$$STKP'   Object module $$STKP
         DC   CL1'S'         On system pack
         DS   CL4
```

Linkage to Checkpoint is achieved by branching to the resident general entry routine, NENTRY, through the resident communication vector NCENTR. (NENTRY is documented in the *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502, for Models 6 and 10, or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.) This branch must be followed by a constant value, called the request indicator byte (RIB), identifying the Find transient, $$SPFN. $$SPFN finds Checkpoint, using the find parameter list (FDPARM). The C/S address is set up for a fetch, and another branch is made to NENTRY with a RIB for requesting $$STKP. This is followed by the C/S address of the $$STKP and the module size (three sectors).

The completion code should be checked to ensure that a valid checkpoint occurred. The checkpoint is ignored if an I/O error is outstanding on any of the supported and allocated devices. Valid return codes are X'40' (checkpoint taken) and X'41' (checkpoint ignored).



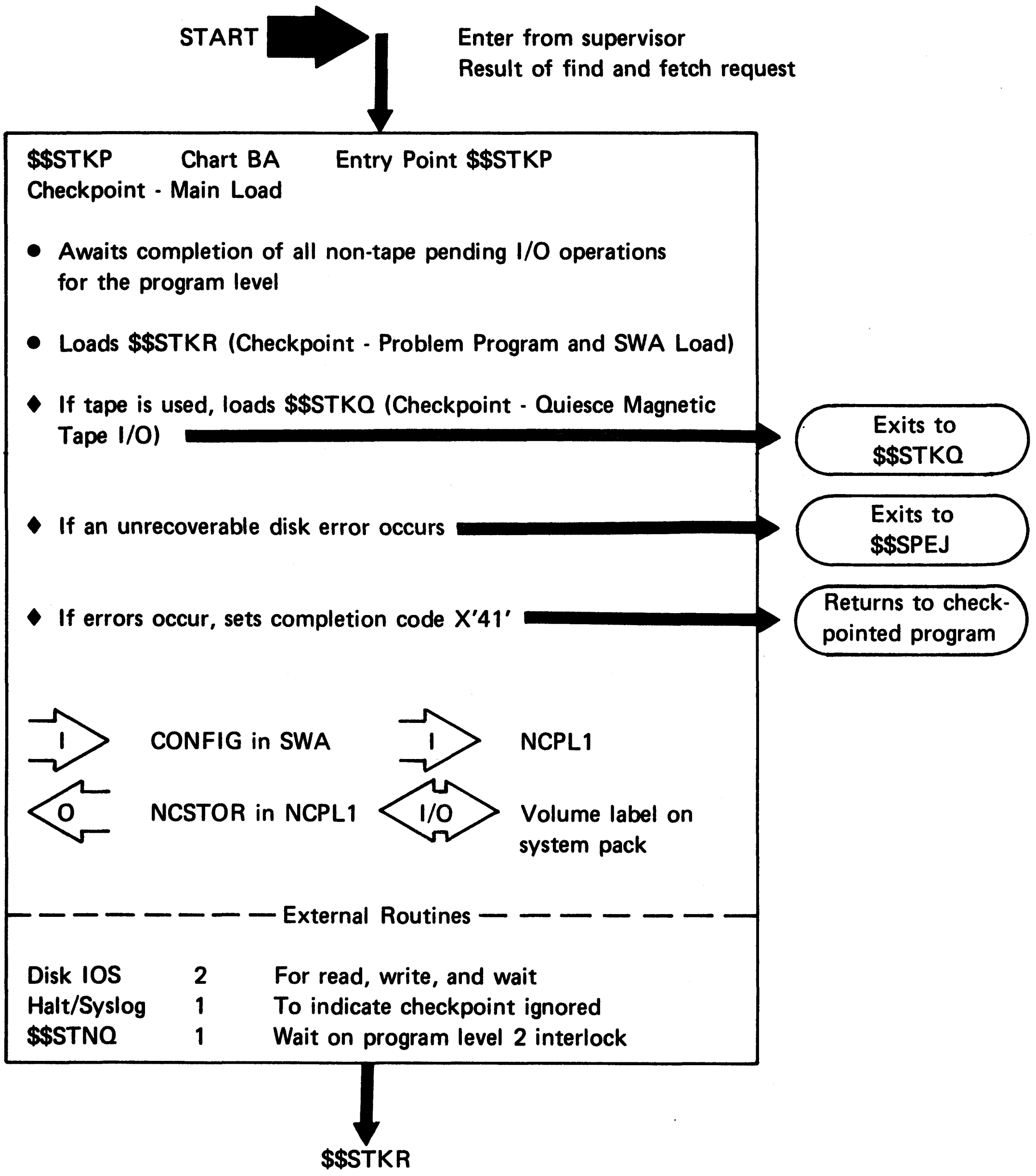
Subroutine Flowchart Identification

Chart ID – Routine is in this manual.

1 – Routine is in *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502 (for Model 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

2 – Routine is in *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual,* SY21-0512 (for Model 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

Figure 21 (Part 1 of 5). Checkpoint Operational Diagram Legend



START

Enter from supervisor
Result of find and fetch request

$$STKP Chart BA Entry Point $$STKP
Checkpoint - Main Load

- Awaits completion of all non-tape pending I/O operations for the program level
- Loads $$STKR (Checkpoint - Problem Program and SWA Load)

♦ If tape is used, loads $$STKQ (Checkpoint - Quiesce Magnetic Tape I/O) → Exits to $$STKQ

♦ If an unrecoverable disk error occurs → Exits to $$SPEJ

♦ If errors occur, sets completion code X'41' → Returns to checkpointed program

CONFIG in SWA

NCPL1

NCSTOR in NCPL1

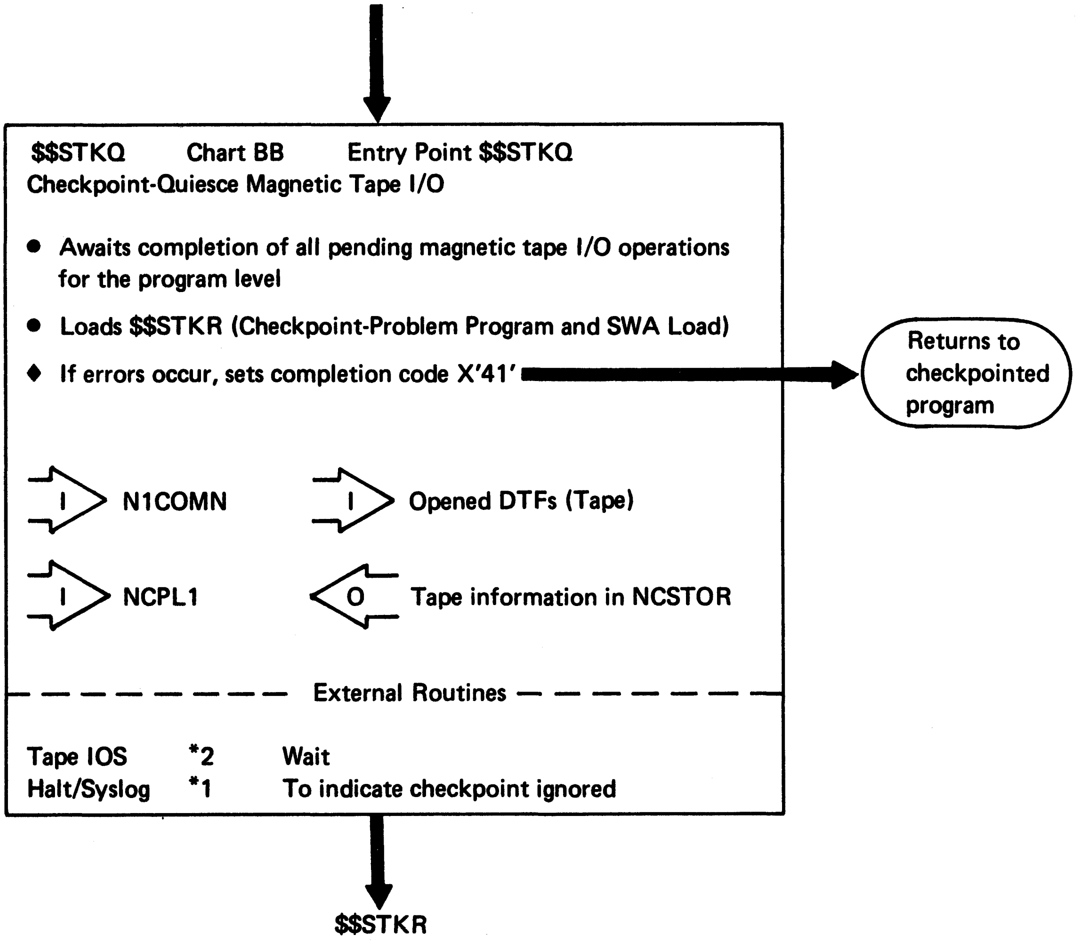
Volume label on system pack

External Routines

Disk IOS	2	For read, write, and wait
Halt/Syslog	1	To indicate checkpoint ignored
$$STNQ	1	Wait on program level 2 interlock

$$STKR

Figure 21 (Part 2 of 5). Checkpoint Operational Diagram Legend



$$STKQ Chart BB Entry Point $$STKQ

Checkpoint-Quiesce Magnetic Tape I/O

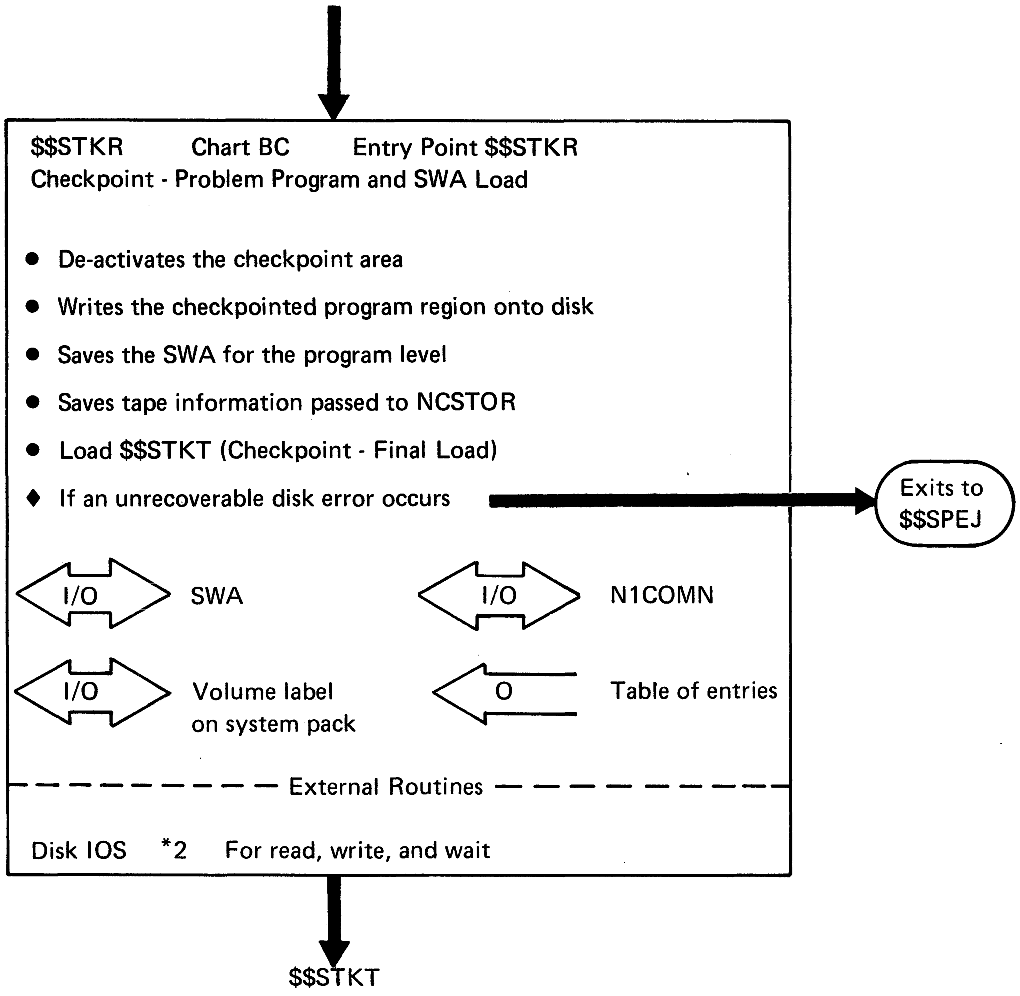
- Awaits completion of all pending magnetic tape I/O operations for the program level
- Loads $$STKR (Checkpoint-Problem Program and SWA Load)

◆ If errors occur, sets completion code X'41'

External Routines

Tape IOS	*2	Wait
Halt/Syslog	*1	To indicate checkpoint ignored

Figure 21 (Part 3 of 5). Checkpoint Operational Diagram Legend



$$STKR Chart BC Entry Point $$STKR

Checkpoint - Problem Program and SWA Load

- De-activates the checkpoint area
- Writes the checkpointed program region onto disk
- Saves the SWA for the program level
- Saves tape information passed to NCSTOR
- Load $$STKT (Checkpoint - Final Load)

◆ If an unrecoverable disk error occurs

Exits to $$SPEJ

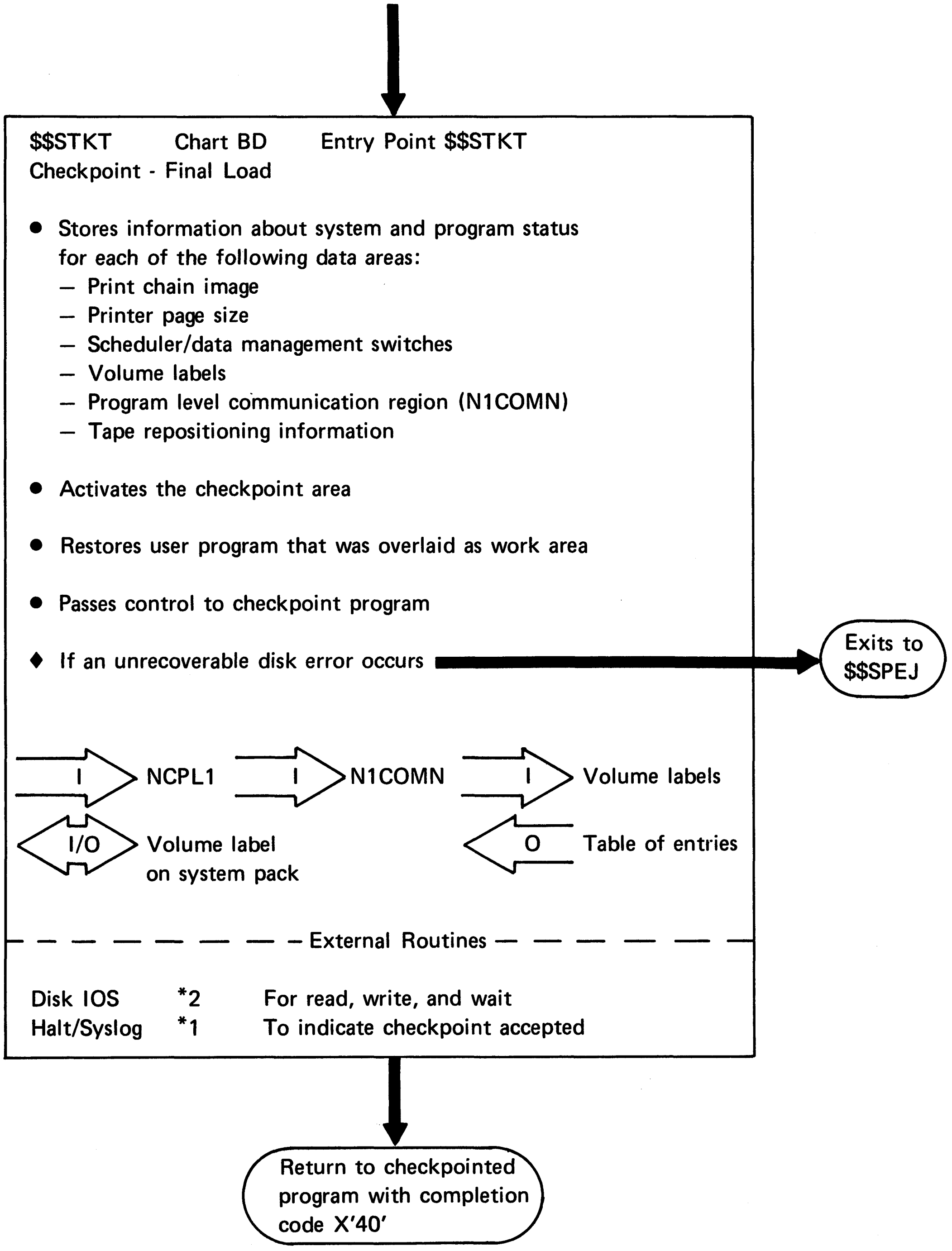
I/O	SWA	I/O	N1COMN
I/O	Volume label on system pack	O	Table of entries

External Routines

Disk IOS *2 For read, write, and wait

$$STKT

Figure 21 (Part 4 of 5). Checkpoint Operational Diagram Legend



$$STKT Chart BD Entry Point $$STKT
Checkpoint - Final Load

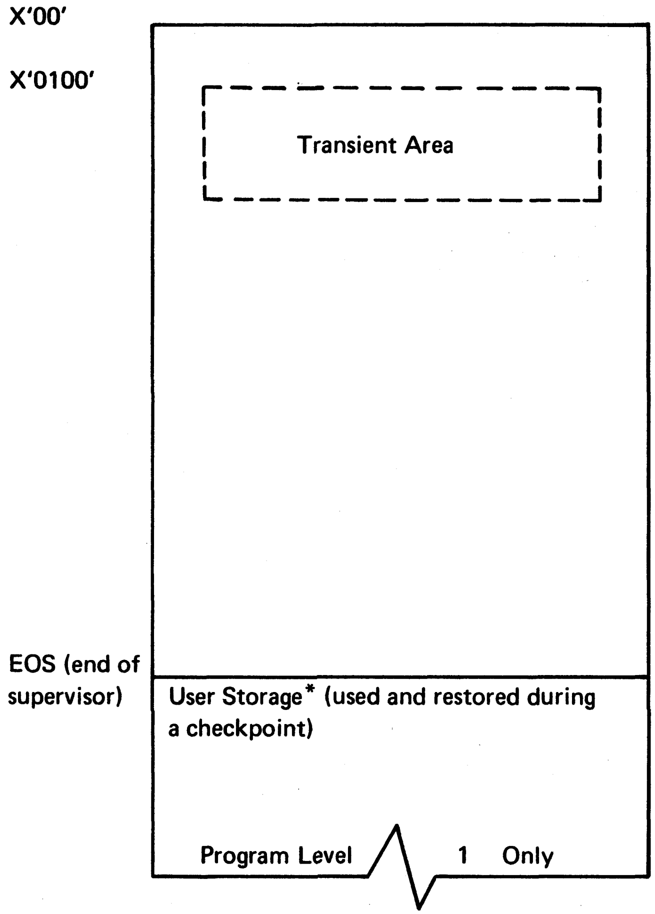
- Stores information about system and program status for each of the following data areas:
  - Print chain image
  - Printer page size
  - Scheduler/data management switches
  - Volume labels
  - Program level communication region (N1COMN)
  - Tape repositioning information
- Activates the checkpoint area
- Restores user program that was overlaid as work area
- Passes control to checkpoint program

♦ If an unrecoverable disk error occurs

External Routines

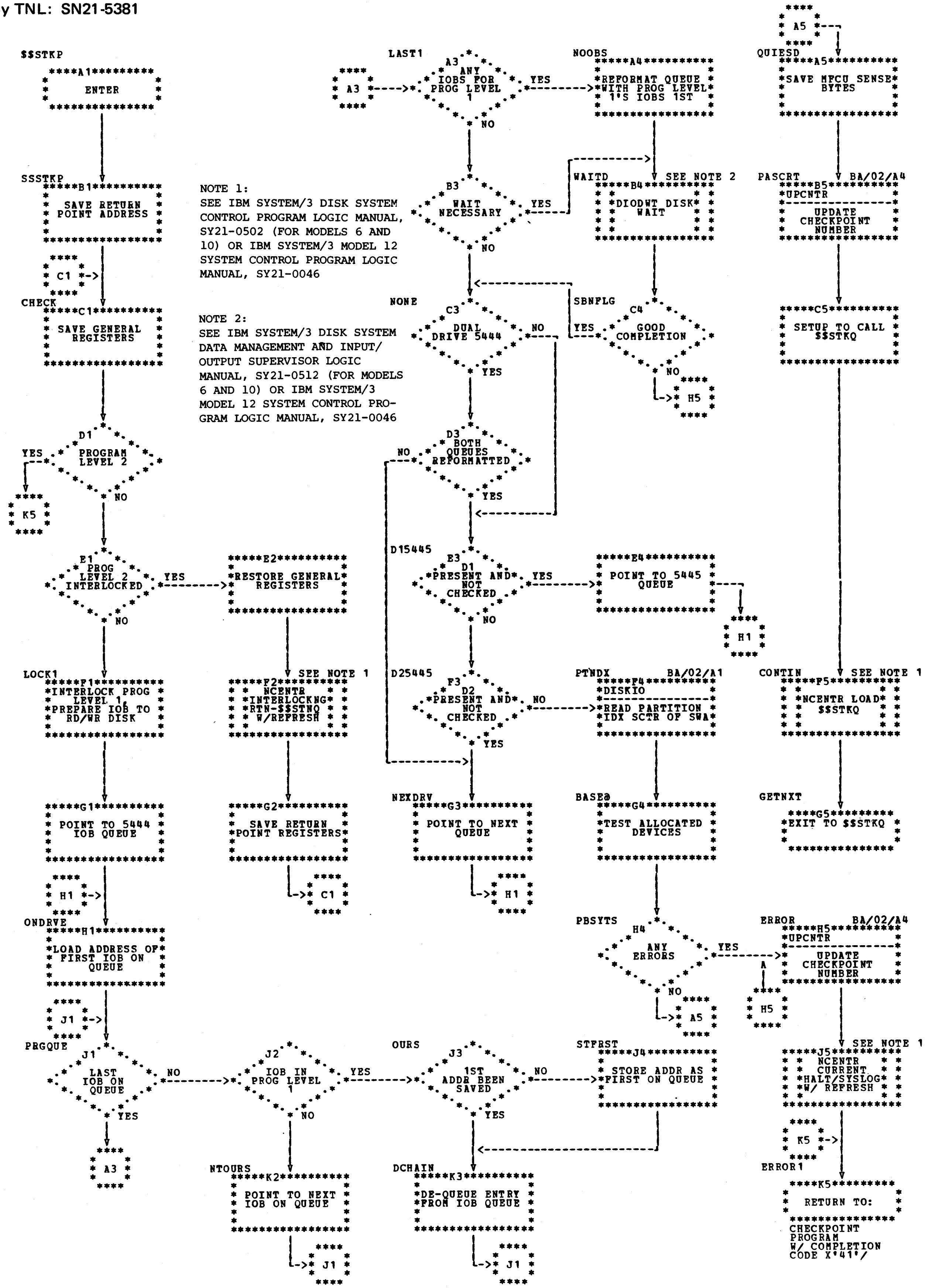
Disk IOS	*2	For read, write, and wait
Halt/Syslog	*1	To indicate checkpoint accepted

Figure 21 (Part 5 of 5). Checkpoint Operational Diagram Legend



*Note: User's program must be at least 1K since this is the minimum work area required.

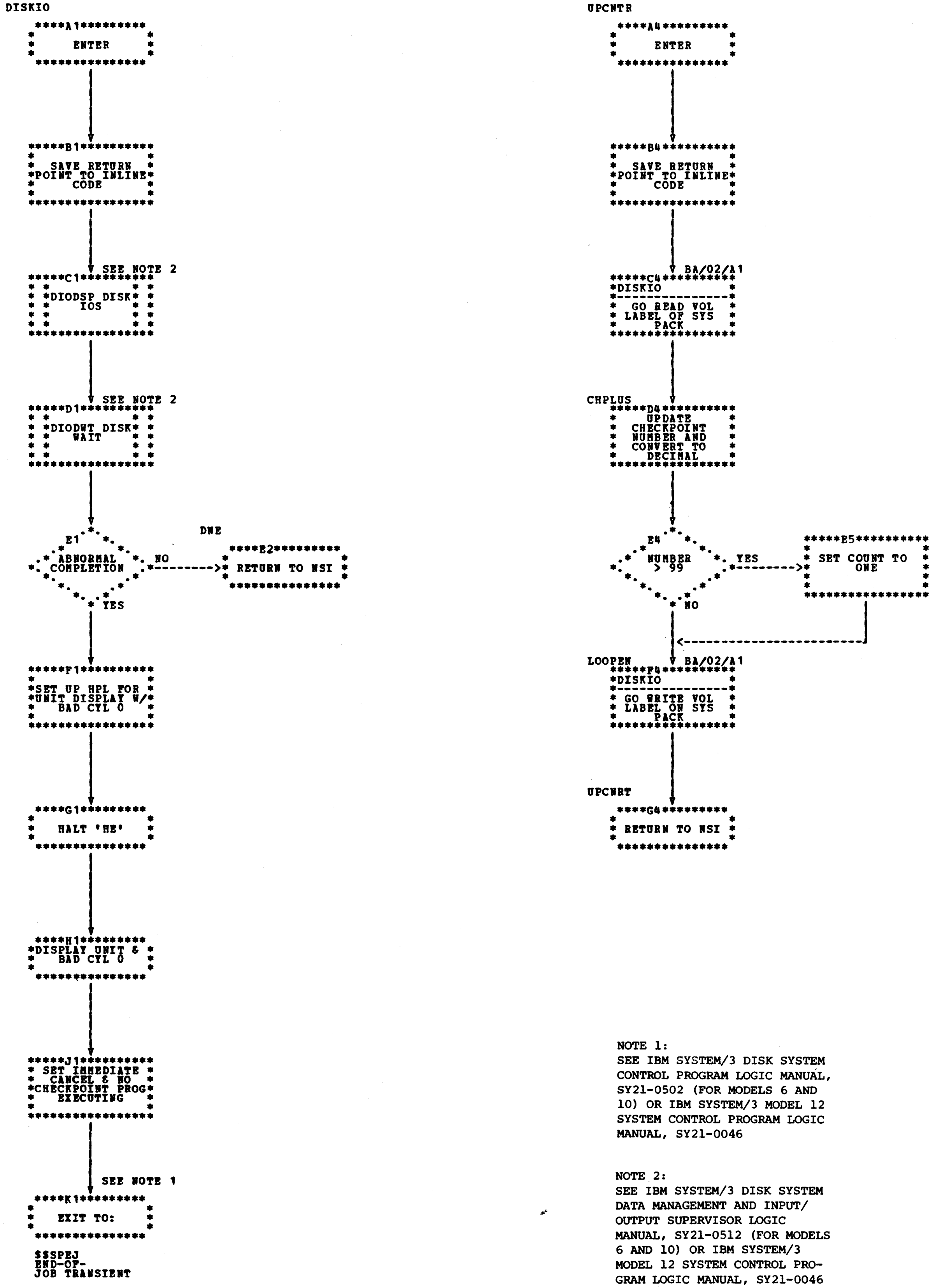
Figure 22. Main Storage Map Showing Transient Area and User Storage Needed by Checkpoint



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

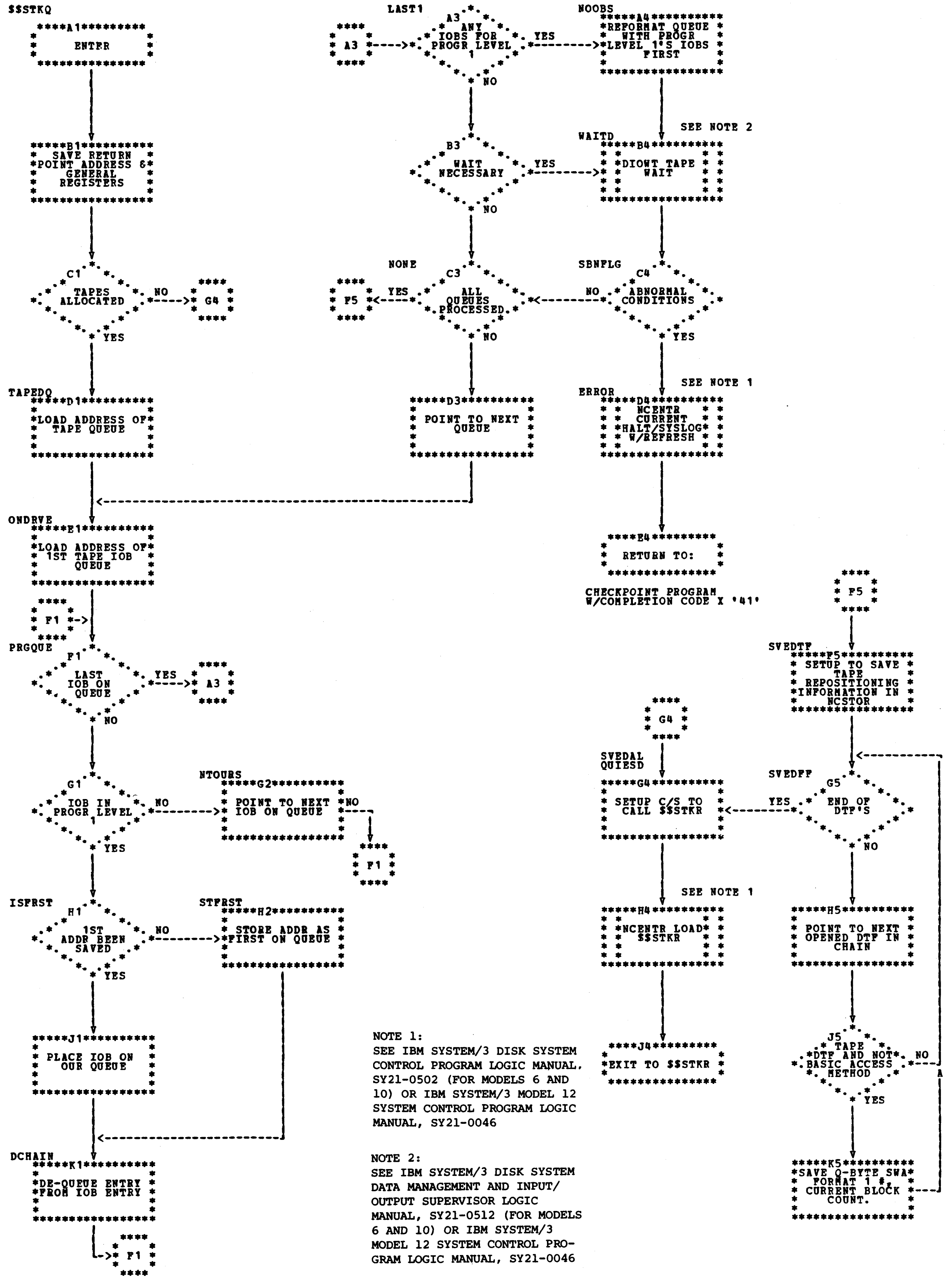
Chart BA (Part 1 of 2). Checkpoint – Main Load ($$STKP)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

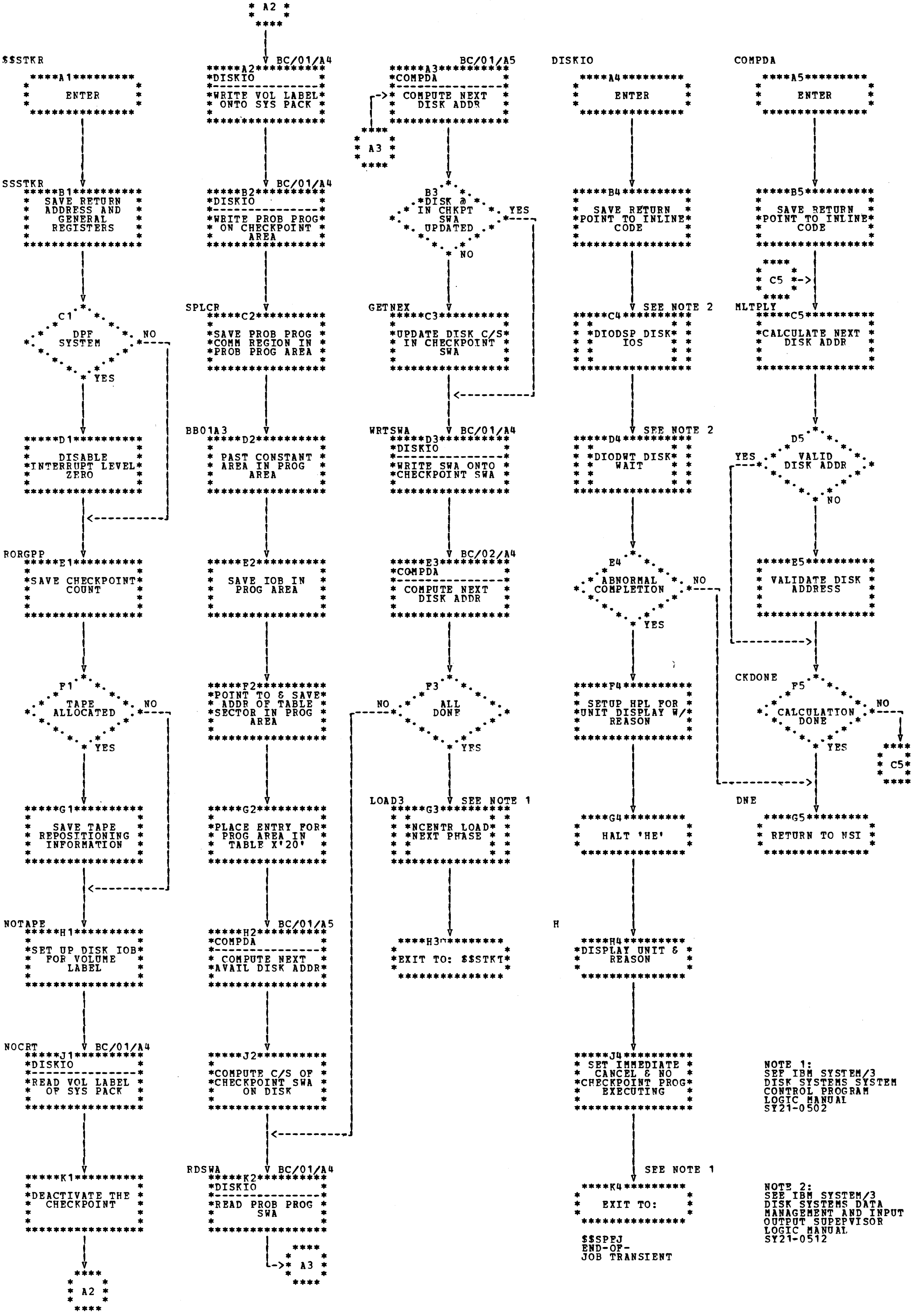
Chart BA (Part 2 of 2). Checkpoint – Main Load ($$STKP)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/ OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart BB. Checkpoint — Quiesce Magnetic Tape I/O ($$STKQ)


A2
$$STKR
A1
ENTER
SSSTKR
B1
SAVE RETURN ADDRESS AND GENERAL REGISTERS
C1
DPF SYSTEM
NO
YES
D1
DISABLE INTERRUPT LEVEL ZERO
RORGPP
E1
SAVE CHECKPOINT COUNT
F1
TAPE ALLOCATED
NO
YES
G1
SAVE TAPE REPOSITIONING INFORMATION
NOTAPE
H1
SET UP DISK IOB FOR VOLUME LABEL
NOCRT
BC/01/A4
J1
DISKIO
READ VOL LABEL OF SYS PACK
K1
DEACTIVATE THE CHECKPOINT
A2
BC/01/A4
A2
DISKIO
WRITE VOL LABEL ONTO SYS PACK
BC/01/A4
B2
DISKIO
WRITE PROB PROG ON CHECKPOINT AREA
SPLCR
C2
SAVE PROB PROG COMM REGION IN PROB PROG AREA
BB01A3
D2
PAST CONSTANT AREA IN PROG AREA
E2
SAVE IOB IN PROG AREA
F2
POINT TO & SAVE ADDR OF TABLE SECTOR IN PROG AREA
G2
PLACE ENTRY FOR PROG AREA IN TABLE X'20'
BC/01/A5
H2
COMPDA
COMPUTE NEXT AVAIL DISK ADDR
J2
COMPUTE C/S OF CHECKPOINT SWA ON DISK
RDSWA
BC/01/A4
K2
DISKIO
READ PROB PROG SWA
A3
BC/01/A5
A3
COMPDA
COMPUTE NEXT DISK ADDR
A3
B3
DISK @ IN CHKPT SWA UPDATED
YES
NO
GETNEX
C3
UPDATE DISK C/S IN CHECKPOINT SWA
WRTSWA
BC/01/A4
D3
DISKIO
WRITE SWA ONTO CHECKPOINT SWA
BC/02/A4
E3
COMPDA
COMPUTE NEXT DISK ADDR
F3
ALL DONE
NO
YES
LOAD3
SEE NOTE 1
G3
NCENTR LOAD NEXT PHASE
H3
EXIT TO: $$STKT
DISKIO
A4
ENTER
B4
SAVE RETURN POINT TO INLINE CODE
SEE NOTE 2
C4
DIODSP DISK IOS
SEE NOTE 2
D4
DIODWT DISK WAIT
E4
ABNORMAL COMPLETION
NO
YES
F4
SETUP HPL FOR UNIT DISPLAY W/ REASON
G4
HALT 'HE'
H
H4
DISPLAY UNIT & REASON
J4
SET IMMEDIATE CANCEL & NO CHECKPOINT PROG EXECUTING
SEE NOTE 1
K4
EXIT TO:
$$SPEJ END-OF-JOB TRANSIENT
COMPDA
A5
ENTER
B5
SAVE RETURN POINT TO INLINE CODE
C5
MLTPLY
C5
CALCULATE NEXT DISK ADDR
D5
VALID DISK ADDR
YES
NO
E5
VALIDATE DISK ADDRESS
CKDONE
F5
CALCULATION DONE
NO
YES
C5
DNE
G5
RETURN TO NSI

NOTE 1:
SEE IBM SYSTEM/3
DISK SYSTEMS SYSTEM
CONTROL PROGRAM
LOGIC MANUAL
SY21-0502

NOTE 2:
SEE IBM SYSTEM/3
DISK SYSTEMS DATA
MANAGEMENT AND INPUT
OUTPUT SUPERVISOR
LOGIC MANUAL
SY21-0512

Chart BC. Checkpoint — Problem Program and SWA Load ($$STKR)

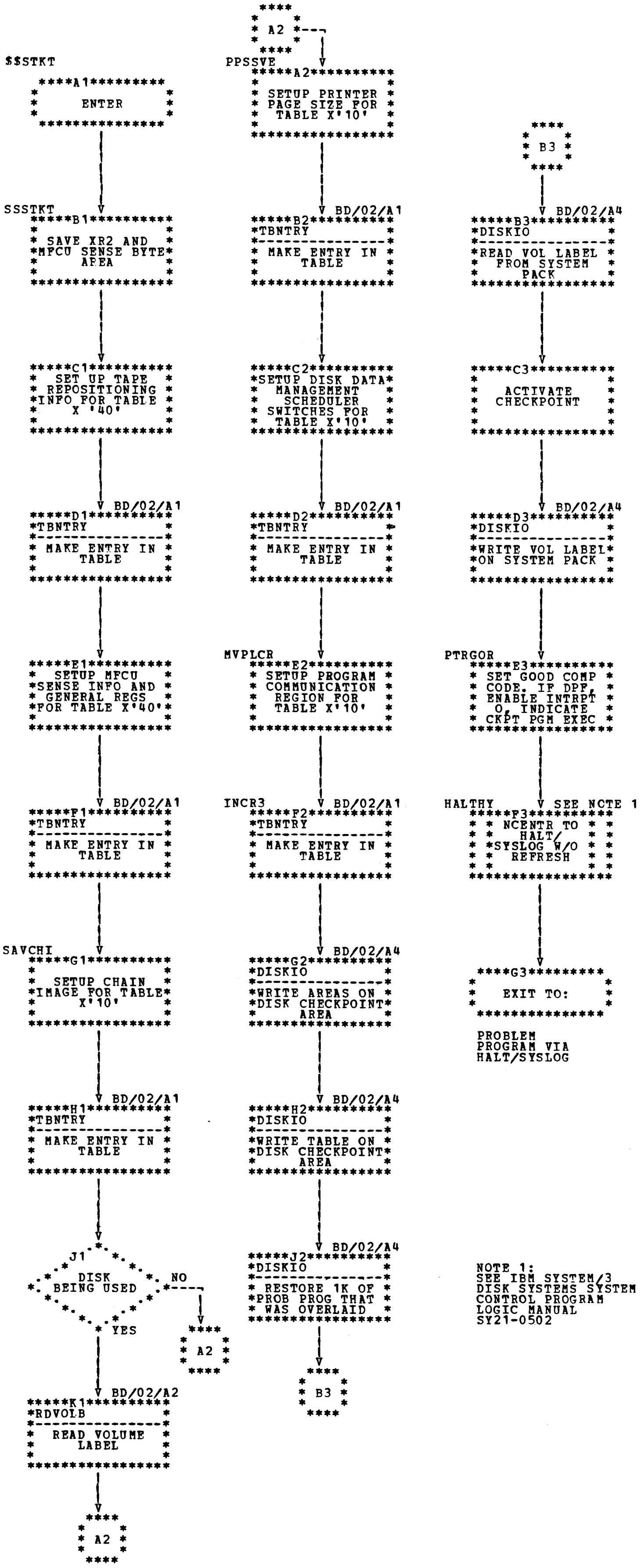
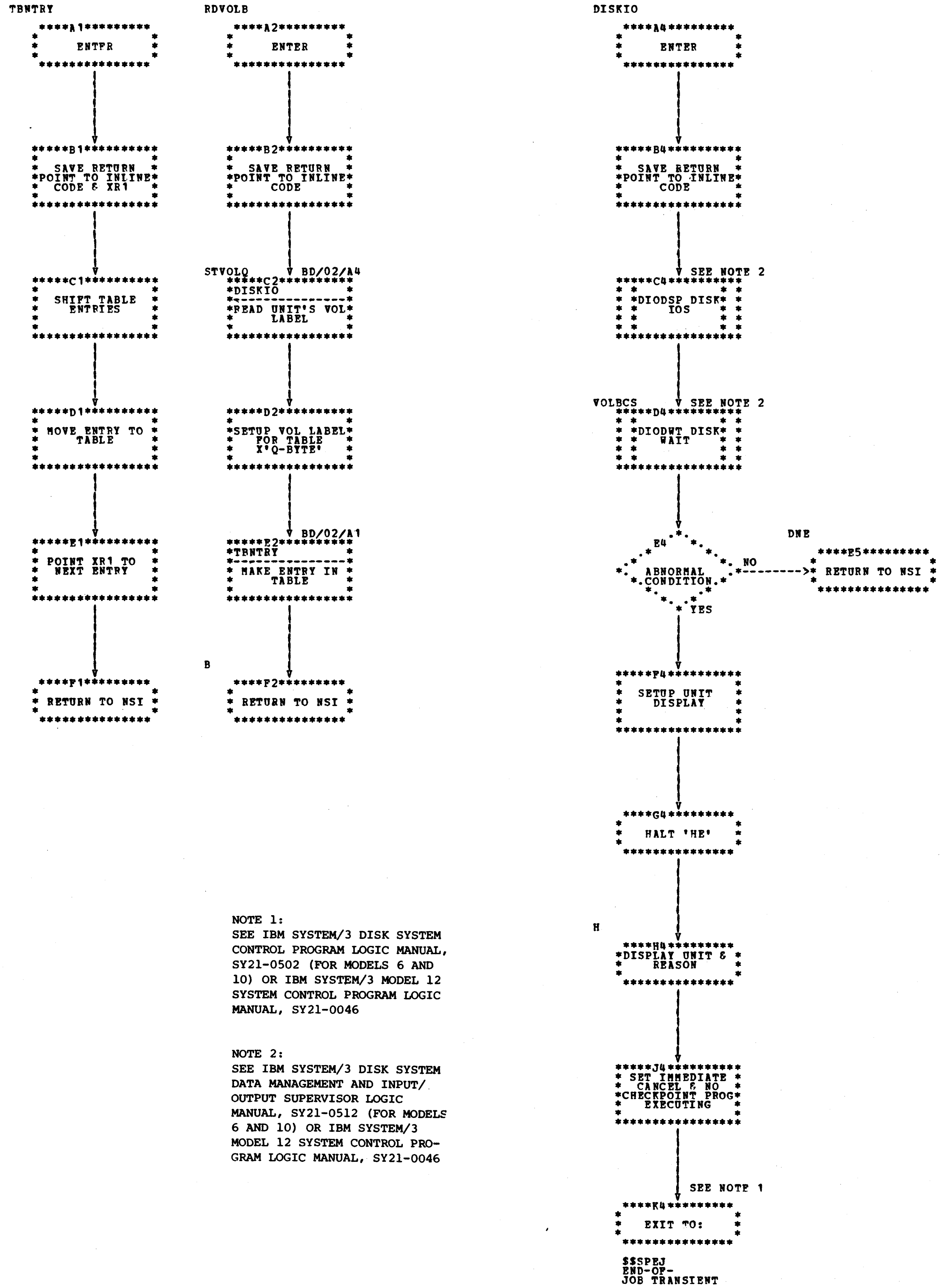

A2
$$STKT
A1
ENTER
SSSTKT
B1
SAVE XR2 AND MPCU SENSE BYTE AREA
C1
SET UP TAPE REPOSITIONING INFO FOR TABLE X '40'
BD/02/A1
D1
TBNTRY
MAKE ENTRY IN TABLE
E1
SETUP MFCU SENSE INFO AND GENERAL REGS FOR TABLE X'40'
BD/02/A1
F1
TBNTRY
MAKE ENTRY IN TABLE
SAVCHI
G1
SETUP CHAIN IMAGE FOR TABLE X'10'
BD/02/A1
H1
TBNTRY
MAKE ENTRY IN TABLE
J1
DISK BEING USED
NO
YES
A2
BD/02/A2
K1
RDVOLB
READ VOLUME LABEL
A2
PPSSVE
A2
SETUP PRINTER PAGE SIZE FOR TABLE X'10'
BD/02/A1
B2
TBNTRY
MAKE ENTRY IN TABLE
C2
SETUP DISK DATA MANAGEMENT SCHEDULER SWITCHES FOR TABLE X'10'
BD/02/A1
D2
TBNTRY
MAKE ENTRY IN TABLE
MVPLCR
E2
SETUP PROGRAM COMMUNICATION REGION FOR TABLE X'10'
INCR3
BD/02/A1
F2
TBNTRY
MAKE ENTRY IN TABLE
BD/02/A4
G2
DISKIO
WRITE AREAS ON DISK CHECKPOINT AREA
BD/02/A4
H2
DISKIO
WRITE TABLE ON DISK CHECKPOINT AREA
BD/02/A4
J2
DISKIO
RESTORE 1K OF PROB PROG THAT WAS OVERLAID
B3
B3
BD/02/A4
B3
DISKIO
READ VOL LABEL FROM SYSTEM PACK
C3
ACTIVATE CHECKPOINT
BD/02/A4
D3
DISKIO
WRITE VOL LABEL ON SYSTEM PACK
PTRGOR
E3
SET GOOD COMP CODE. IF DPF, ENABLE INTRPT O, INDICATE CKPT PGM EXEC
HALTHY
SEE NCTE 1
F3
NCENTR TO HALT/ SYSLOG W/O REFRESH
G3
EXIT TO:
PROBLEM PROGRAM VIA HALT/SYSLOG
NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEMS SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502

Chart BD (Part 1 of 2). Checkpoint — Final Load ($$STKT)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart BD (Part 2 of 2). Checkpoint — Final Load ($$STKT)

RESTART

Figure 23 shows an overview of Restart. (An operational diagram legend is shown in Figure 21.) A storage map is given in Figure 24.

$$RSTR (Restart - Main Load) is entered from the Supervisor as a result of // LOAD $$RSTR OCL statements. It checks for an active checkpoint and available storage. $$RSTR restores the SWA from the checkpoint disk area back to the system disk area. It also assures that the correct disks or tapes are mounted and cards or tapes are repositioned for any allocated and supported card devices (except 1442) or tape devices.

If tapes are used, the files are repositioned to the block following the last block processed at the last checkpoint, provided the same reel is mounted. The filename and the tape drive are logged for each tape drive being processed to give the operator the opportunity to verify that the correct reel is mounted. Standard labeled tapes are checked to verify that the file label and volume sequence number match the reels being processed at the last accepted checkpoint. The nonstandard or unlabeled tapes are not verified.

Basic access method files or direct calls to tape IOS are the responsibility of the user and no repositioning or label checking is done.

Once this is done, $$STKV (Restart - Problem Program and Final Load) is loaded.

$$STKV restores N1COMN and passes control to the checkpointed program. The restart entry point is at the last checkpoint taken.

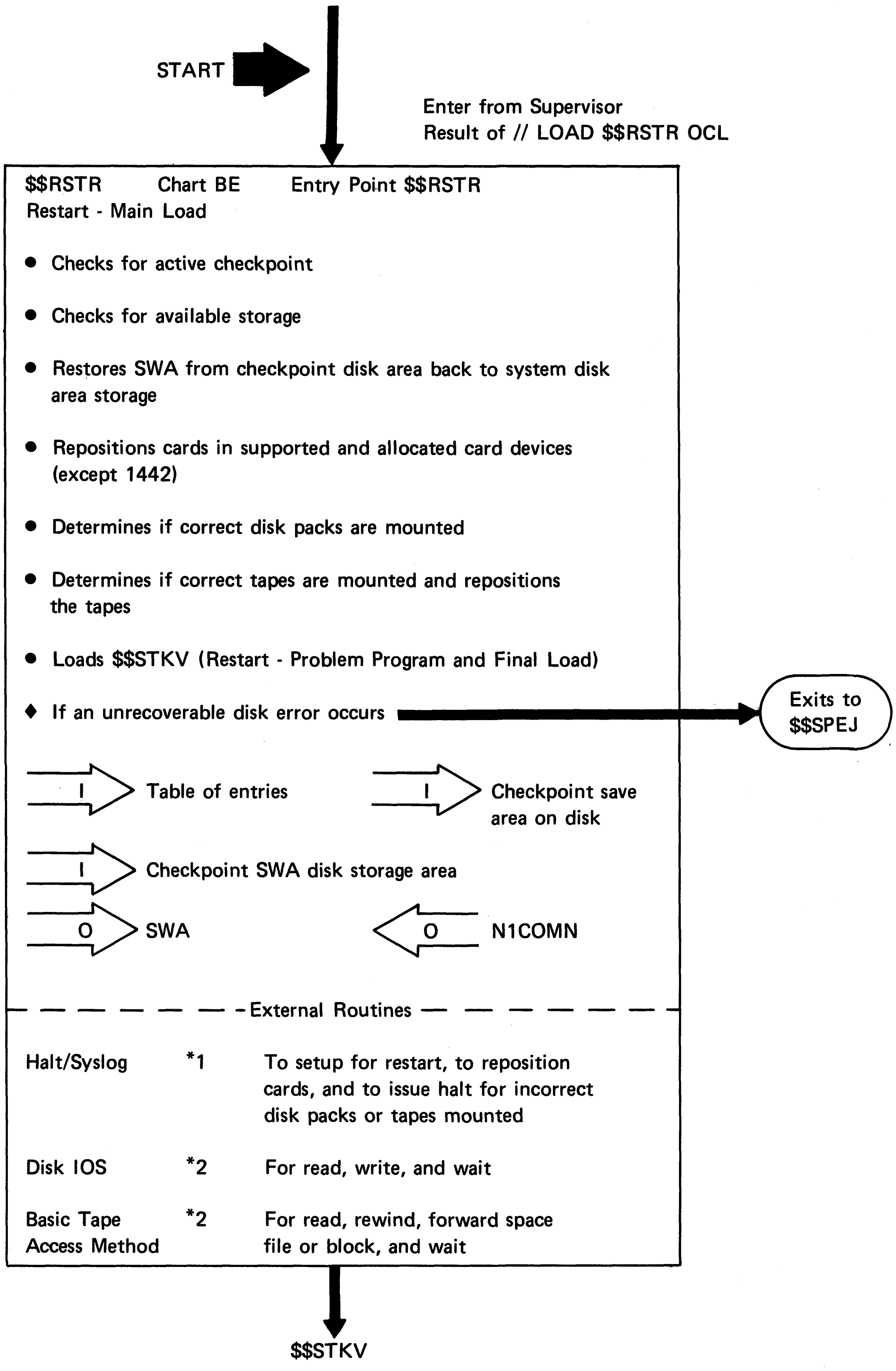
If an immediate cancel or an unrecoverable disk error should occur, control is passed to the end-of-job transient, $$SPEJ.

RESTART LINKAGE

To continue execution of the interrupted job at the last checkpoint the user must submit the following OCL statements to load Restart:

```
// LOAD $$RSTR,unit
// RUN
```

The LOAD statement identifies the program to be run and indicates the disk on which it is located. For Restart, the unit must be the system IPL pack containing the checkpoint file to be restarted. The RUN statement indicates the end of the OCL statements, and the system runs the program. To guarantee the required minimum size for program level 2 (which must allow 5K for Restart in program level 1), a PARTITION statement may be required. Also, to reestablish the log device, a LOG statement may be required. For more information on these OCL statements, see *IBM System/3 Model 10 Disk System Operation Control Language and Disk Utilities Reference Manual,* GC21-7512, or *IBM System/3 Model 12 User's Guide,* GC21-5142.



START

Enter from Supervisor
Result of // LOAD $$RSTR OCL

$$RSTR Chart BE Entry Point $$RSTR
Restart - Main Load

- Checks for active checkpoint
- Checks for available storage
- Restores SWA from checkpoint disk area back to system disk area storage
- Repositions cards in supported and allocated card devices (except 1442)
- Determines if correct disk packs are mounted
- Determines if correct tapes are mounted and repositions the tapes
- Loads $$STKV (Restart - Problem Program and Final Load)

♦ If an unrecoverable disk error occurs

Exits to $$SPEJ

I Table of entries

I Checkpoint save area on disk

I Checkpoint SWA disk storage area

O SWA

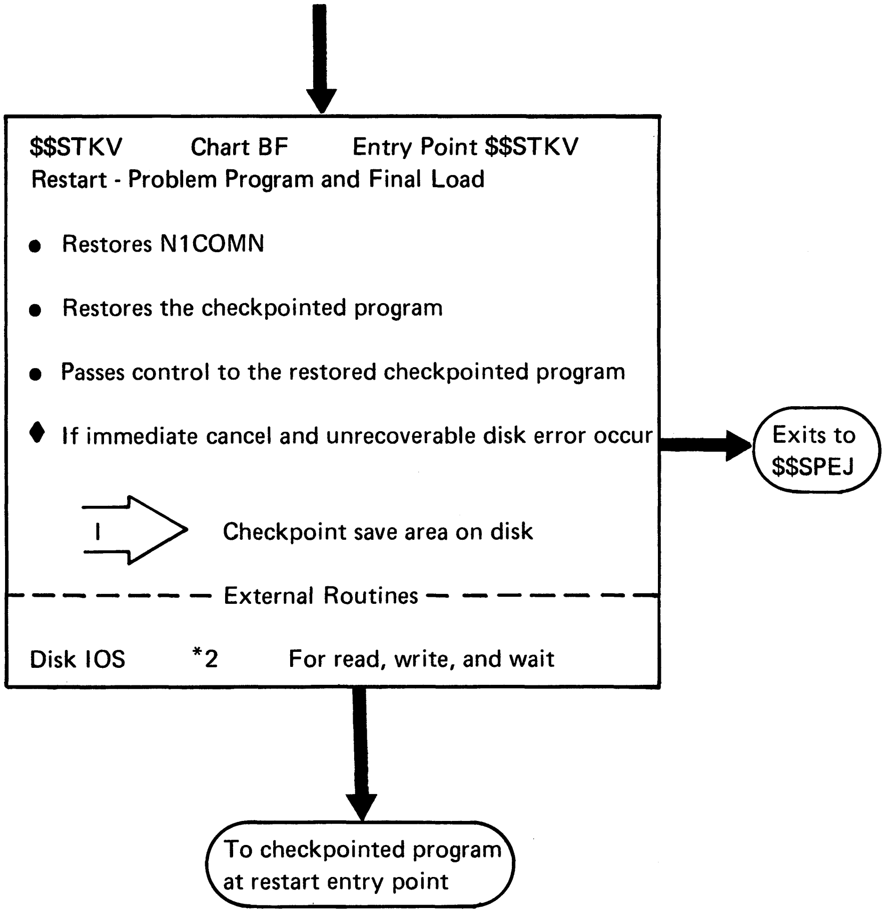
O N1COMN

External Routines

Halt/Syslog	*1	To setup for restart, to reposition cards, and to issue halt for incorrect disk packs or tapes mounted
Disk IOS	*2	For read, write, and wait
Basic Tape Access Method	*2	For read, rewind, forward space file or block, and wait

$$STKV

Figure 23 (Part 1 of 2). Restart Operational Diagram



$$STKV Chart BF Entry Point $$STKV
Restart - Problem Program and Final Load

- Restores N1COMN
- Restores the checkpointed program
- Passes control to the restored checkpointed program

♦ If immediate cancel and unrecoverable disk error occur

Exits to $$SPEJ

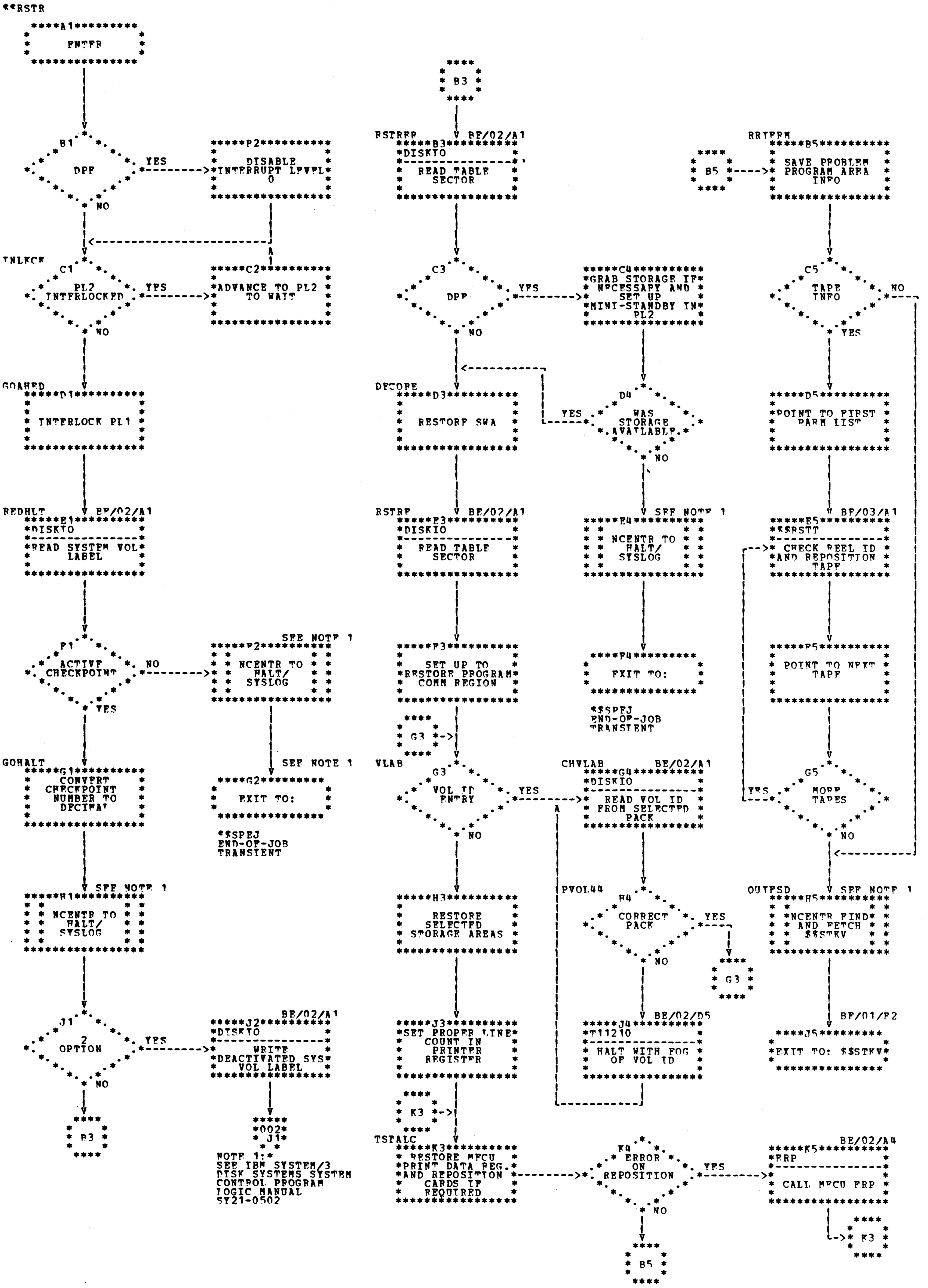
I Checkpoint save area on disk

External Routines

Disk IOS *2 For read, write, and wait

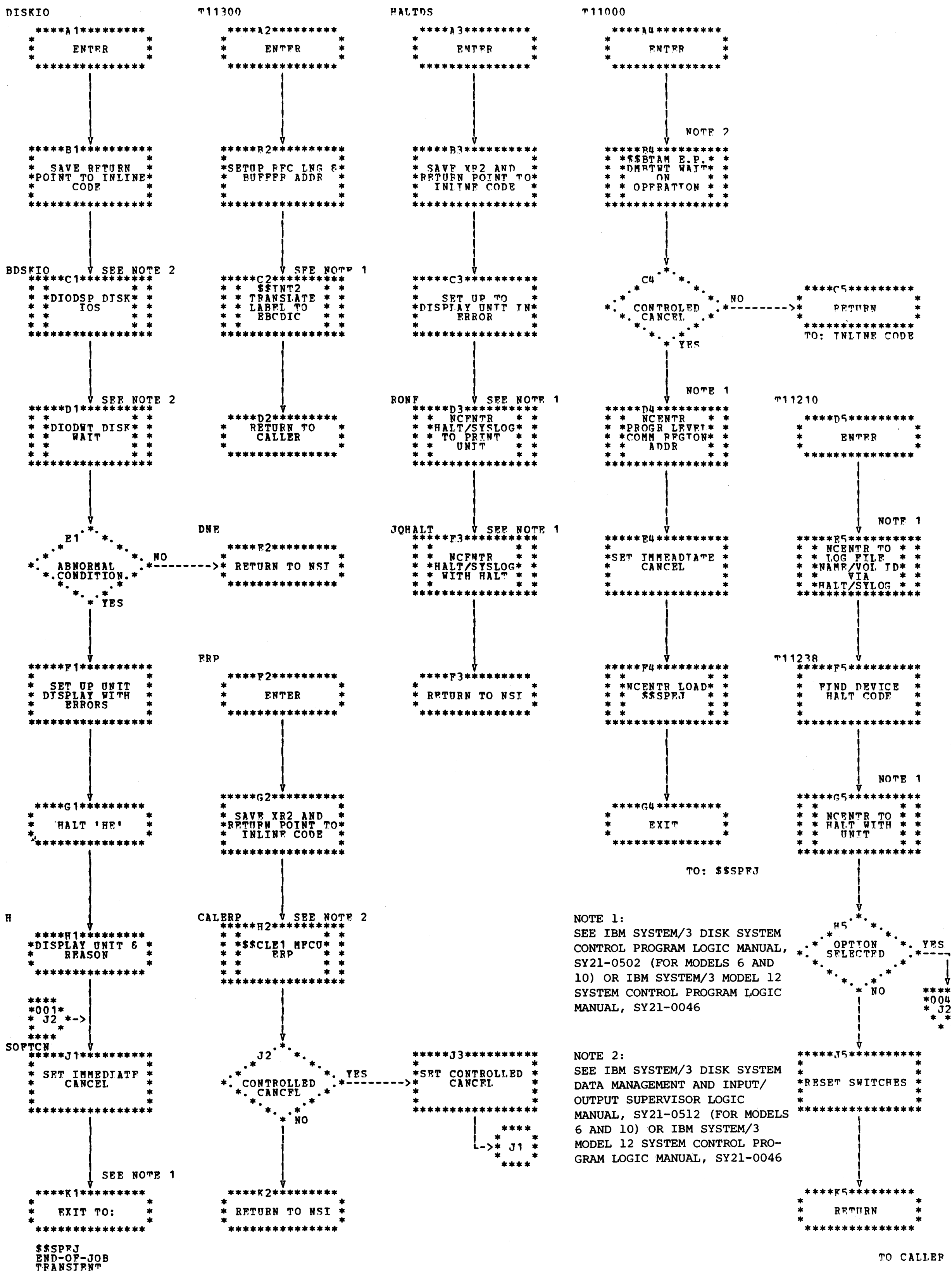
To checkpointed program at restart entry point

Figure 23 (Part 2 of 2). Restart Operational Diagram



NOTE 1: SEE IBM SYSTEM/3 DISK SYSTEMS SYSTEM CONTROL PROGRAM LOGIC MANUAL SY21-0502

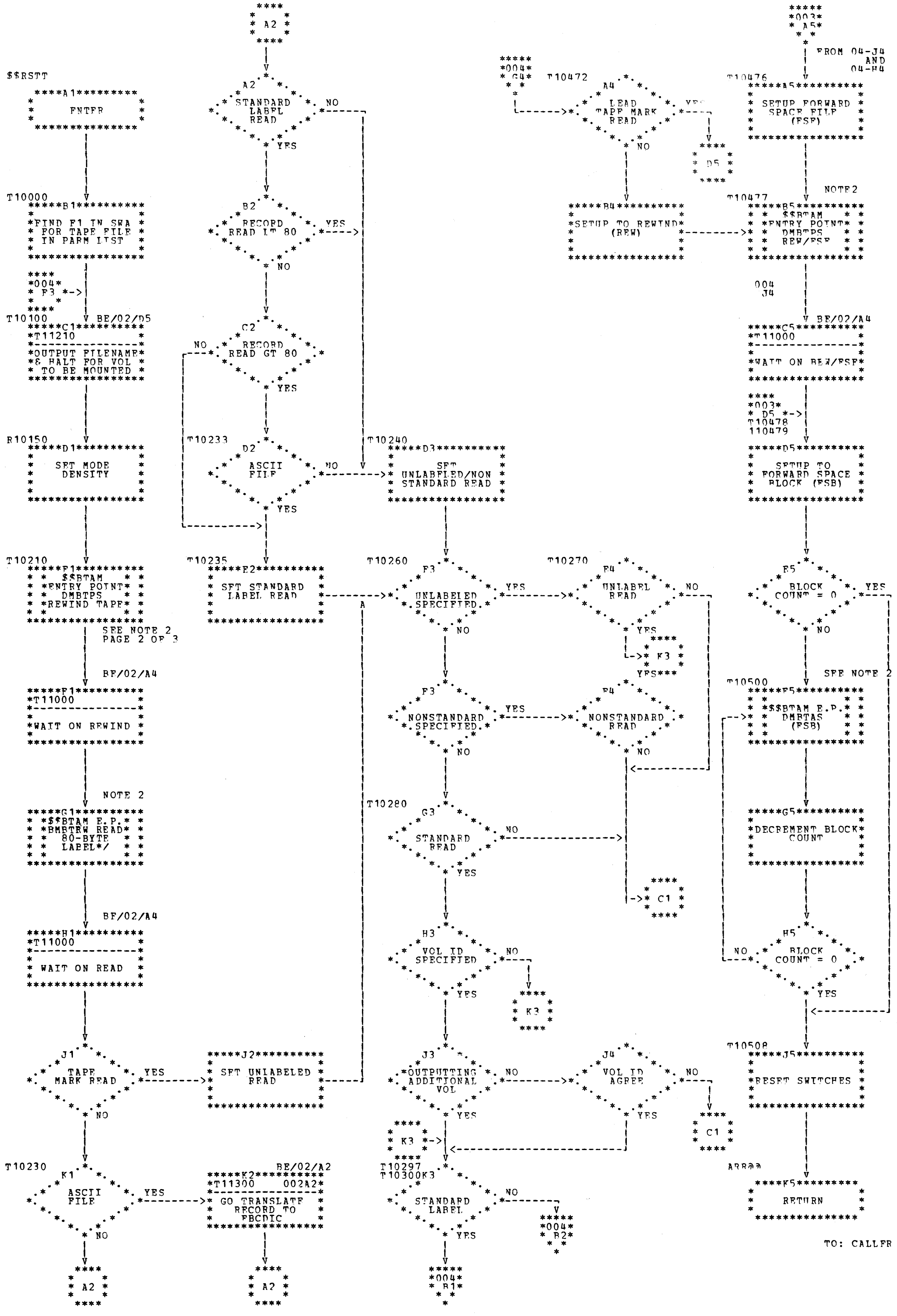
Chart BE (Part 1 of 4). Restart — Main Load ($$RSTR)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/ OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart BE (Part 2 of 4). Restart — Main Load ($$RSTR)



● Chart BE (Part 3 of 4). Restart — Main Load ($$RSTR)

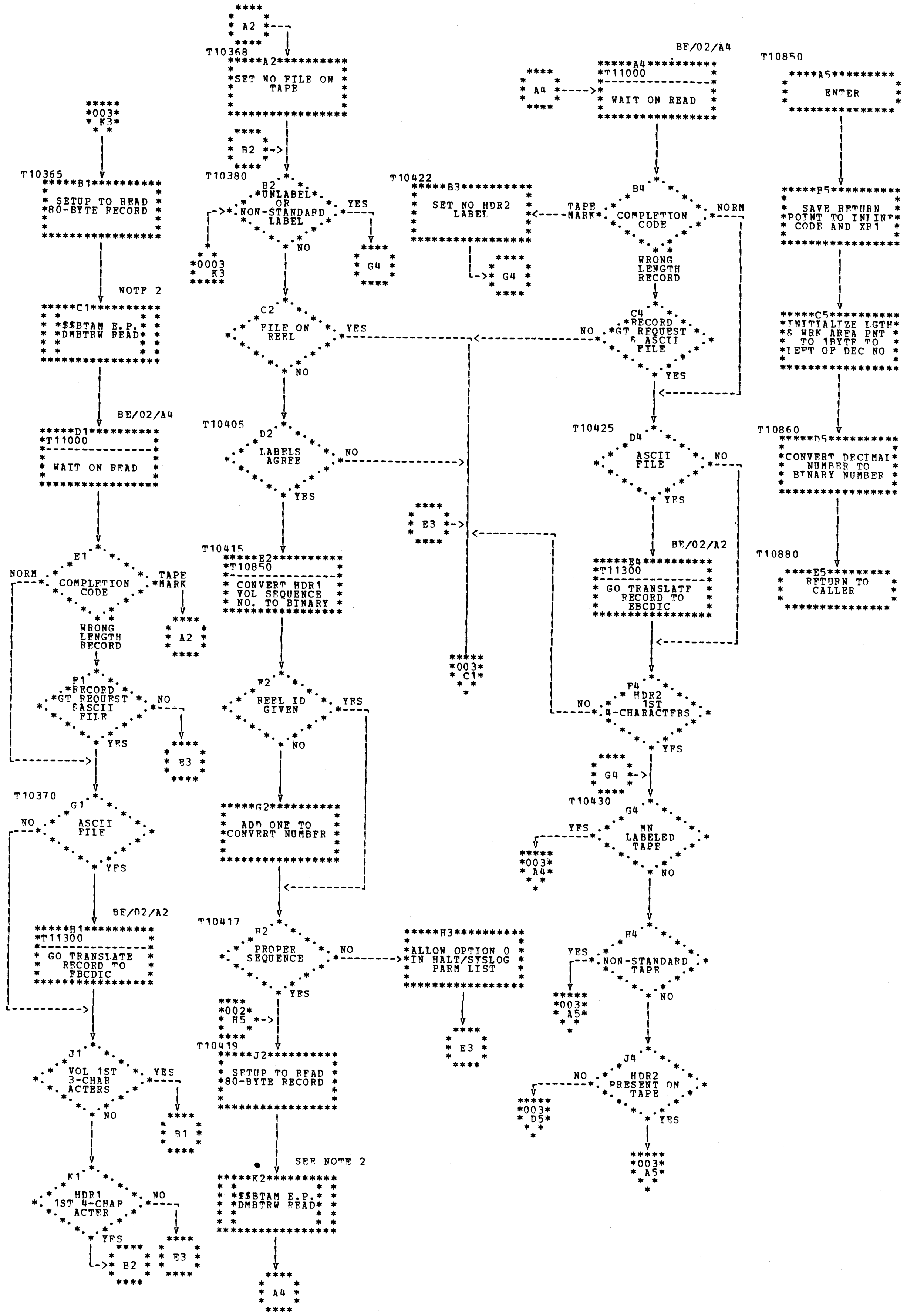
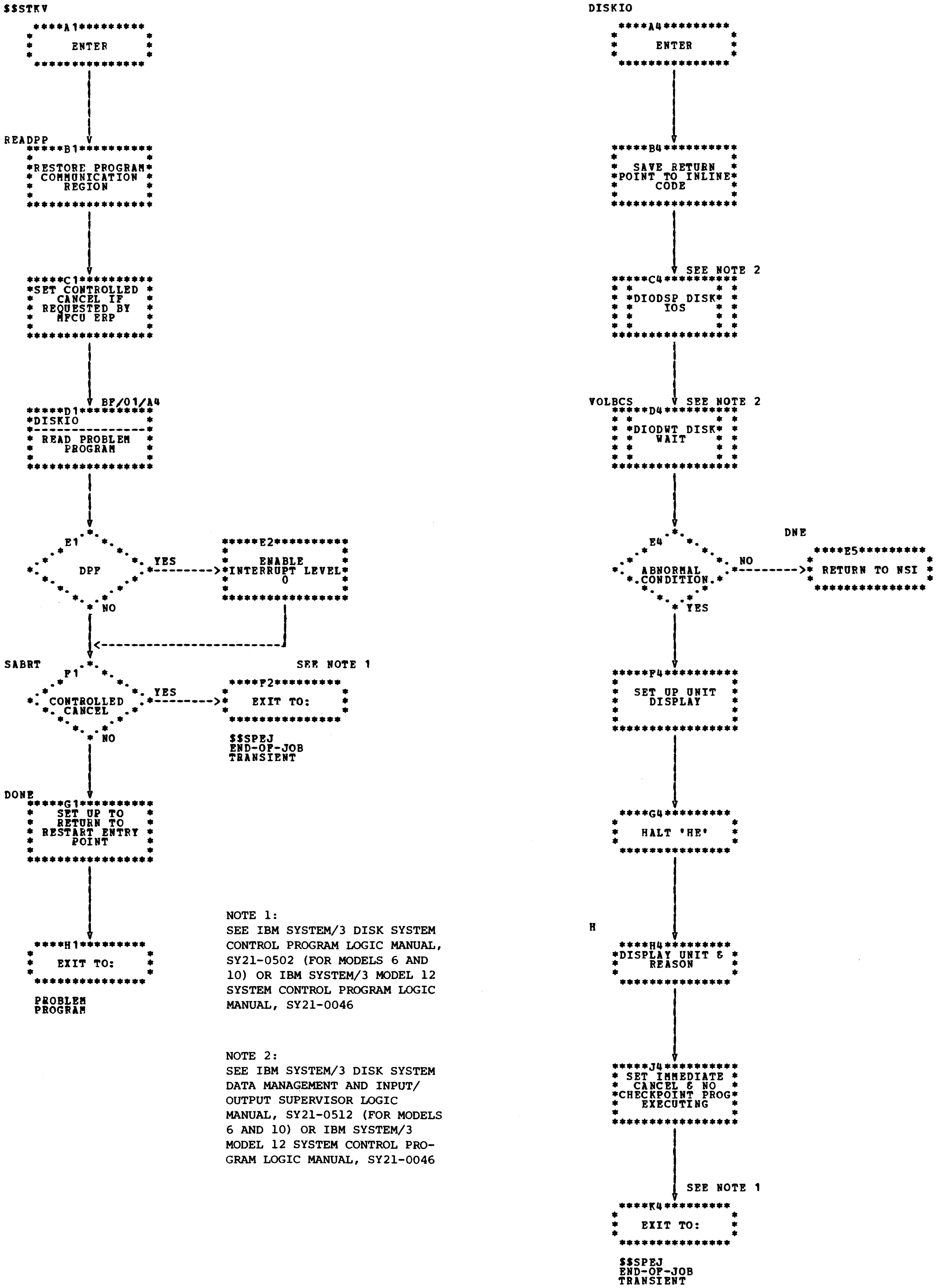


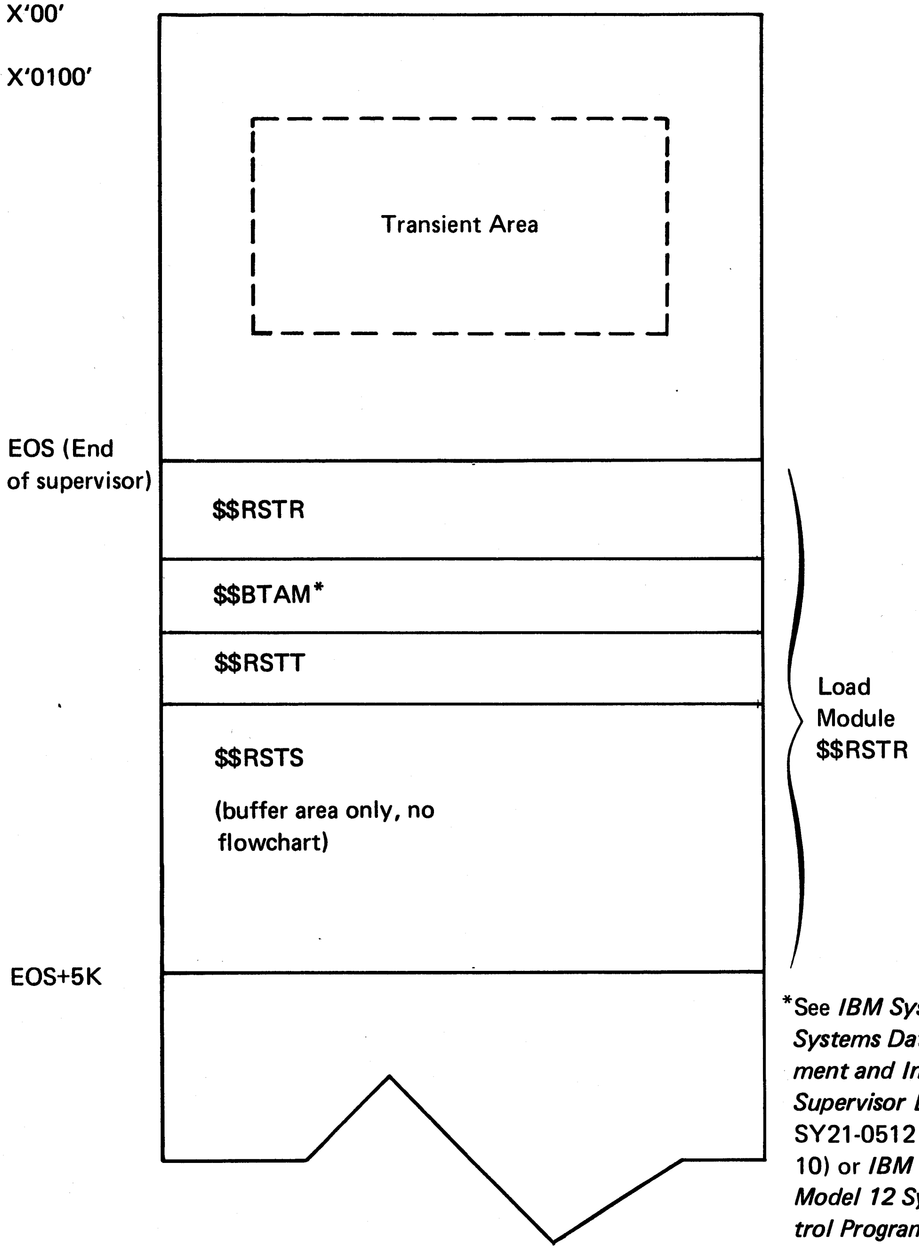
Chart BE (Part 4 of 4). Restart — Main Load ($$RSTR)



NOTE 1:
SEE IBM SYSTEM/3 DISK SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0502 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

NOTE 2:
SEE IBM SYSTEM/3 DISK SYSTEM DATA MANAGEMENT AND INPUT/ OUTPUT SUPERVISOR LOGIC MANUAL, SY21-0512 (FOR MODELS 6 AND 10) OR IBM SYSTEM/3 MODEL 12 SYSTEM CONTROL PROGRAM LOGIC MANUAL, SY21-0046

Chart BF. Restart — Problem Program and Final Load ($$STKV)



*See *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual,* SY21-0512 (for Model 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

Figure 24. Main Storage Map Showing Transient Area and User Storage Needed by Restart

SECTION 3. DATA AREA FORMATS

For a description of the following data areas, refer to *IBM System/3 Disk Systems System Control Program Logic Manual,* SY21-0502 (for Model 10) or *IBM System/3 Model 12 System Control Program Logic Manual,* SY21-0046.

- Configuration record (CONFIG) in the scheduler work area
- Program level communications region (N1COMN)
- System communication region (NCPL1)
- Volume labels

TABLE OF ENTRIES

Checkpoint saves pointers to the saved data at each checkpoint taken in a table of entries. Restart uses this saved data to resume execution of a program at the last checkpoint. The address of the disk area in which the table of entries is located is contained in field NCRCSS in the system communication region. The format of the table of entries is shown in Figure 25. The possible entries are shown in Figure 26.

	Entry ID*	Disk Address Where Stored (C/S/#)	Displacement into Sector	Length -1 of Data	Storage Location for Restore
Number of Bytes	1	3	2	1	2

*The following entry IDs may be used:

X'10'	Storage information
X'20'	Object program
X'40'	Checkpoint parameter list
X'C1'	5445 or 3340 main data area drive 1
X'C9'	5445 or 3340 main data area drive 2
X'A1'	5444 removable 1
X'B1'	5444 removable 2 or 5444 simulation area — D2B
X'B9'	5444 simulation area — D2A

Figure 25. Table of Entries

Volume Label

The volume label of the system pack contains information about the checkpoint stored on the pack. The data is saved at:

Decimal Disp	*Meaning*
185	Bit 0 = 1 Active checkpoint
186	Number of last checkpoint request (01-99)

The entries *not* required *if missing* are not in sector and other entries are shifted left and appear in order given.

Required	1	3	2	1	2	
YES	10	CS#	Displacement into sector	LNG-1 of Prog Comm Region	Program Communication @ (NCCOMN)	LNG depends on dedicated or DPF System
YES	10	CS#	Displacement into sector	0	SCH/D.M. 5445 Switches @ (NCSMV3)	
YES	10	CS#	Displacement into sector	0	SCH/D.M. 5444 Switches @ (NCSMV1)	
YES	10	CS#	Displacement into sector	1	Printer Page Size @ (NCRPSZ)	
NO	C1	CS#	Displacement into sector	5	N/A	
NO	C9	CS#	Displacement into sector	5	N/A	
NO	A1	CS#	Displacement into sector	5	N/A	
NO	B9	CS#	Displacement into sector	5	N/A	
NO	B1	CS#	Displacement into sector	5	N/A	
NO	10	CS#	Displacement into sector	LNG-1 of Chain Image (119)	Chain Image @ (NCHIMG+119)	
YES	40	CS#	Displacement into sector	09	N/A	MFCU status, ARR, XR1, XR2, MFCU Print Data Register.
YES	40	CS#	Displacement into sector	15	N/A	Tape Repositioning Information *
YES	20	CS#	N/A	N/A	Program Level 1 Start @	

*The saved data for tape is 16 bytes, as follows:

Number of Bytes	1	1	2	1	1	2	1	1	2	1	1	2
	60 T1	SWA F1 Number	Current Block #	68 T2	SWA F1 Number	Current Block #	70 T3	SWA F1 Number	Current Block #	78 T4	SWA F1 Number	Current Block #

The entries are left adjusted and any entries not required are X'00'. The entries are *not* necessarily in this order.

Figure 26. Possible Table Entries

APAR SUBMISSION

For APAR submission, the following information is necessary:

- Contents of IAR and ARR.
- Full core dump including the transient area.
- Disk dump of the checkpoint area. The C/S/No. of sectors can be obtained from the core dump (Field NCRCSS of the System Communications region).
- Disk dump of the program level 1 scheduler workarea. The C/S can be obtained from the core dump (Field NCSWRK of the System Communications region). The scheduler workarea is 48 sectors (2 tracks).
- Disk dump of the volume label of the system pack.

DATA SAVED AT CHECKPOINT

The table of entries (described in Section 3) can be used to determine what information was saved at the last checkpoint. This information is used by RESTART.

At each checkpoint, $$STKR and $$STKT build three sectors of data in the first three sectors of program level 1. The first sector contains data to be saved and pointers to buffers and workareas. The second sector contains the table of entries (see Section 3. Data Areas) which points to the table information in the third sector.

Index

IBM System/3 Printed in U.S.A. SY21-0530-1

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

IBM Technical Newsletter

This Newsletter No. SN21-7668
Date March 15, 1973

Base Publication No. SY21-0530-1
File No. System

Previous Newsletters None

36

IBM System/3 Overlay Linkage Editor and Checkpoint/Restart Program Logic Manual

This Technical Newsletter, a part of version, 08 modification 00 of the IBM System/3 Model 10 and Model 6 Disk Systems, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

5, 6
9 through 12
19 through 24
30.1 (added)
31, 32
35 through 40
71 through 74

Changes to text and illustrations are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by the symbol ● at the left of the caption.

Summary of Amendments

An error message phase has been added to the Overlay Linkage Editor.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

IBM Technical Newsletter

This Newsletter No.	SN21-5289
Date	April 1975
Base Publication No.	SY21-0530-1
File No.	S3-31
Previous Newsletters	SN21-7668

IBM System/3 Overlay Linkage Editor and Checkpoint/Restart Programs Logic Manual

This Technical Newsletter, a part of version 12, modification 00 of IBM System/3 Model 10 Disk System System Control Programming, Program Number 5702-SC1, and version 12, modification 00 of IBM System/3 Model 6 System Control Programming, Program Number 5703-SC1, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

i, ii
3, 4
61, 62

Changes to text and illustrations are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by the symbol ● at the left of the caption.

Summary of Amendments

- Add Model 8 reference to Preface

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

IBM **Technical Newsletter**

This Newsletter No.	SN21-5381
Date	17 March 1976
Base Publication No.	SY21-0530-1
File No.	S3-31
Previous Newsletters	SN21-5289 SN21-7668

IBM System/3
Overlay Linkage Editor and
Checkpoint/Restart Programs
Logic Manual

This technical newsletter is a part of version 13, modification 00 of IBM System/3 Model 10 Disk System Control Programming (Program Number 5702-SC1) and also applies to the IBM System/3 Model 6 System Control Programming (Program Number 5703-SC1) and IBM System/3 Model 12 System Control Programming (Program Number 5705-SC1). This technical newsletter provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

Cover, edition notice	53 through 58
i, ii	61 through 64
9, 10	67, 68
15, 16	71, 72
27, 28	75 through 78
45, 46	

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

Model 12 reference has been added to titles of associated publications.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

IBM® Technical Newsletter

This Newsletter No. SN21-5574
Date 24 June 1977

Base Publication No. SY21-0530-1
File No. S3-31

Previous Newsletters SN21-5289
SN21-7668
SN21-5381

IBM System/3 Overlay Linkage Editor and Checkpoint/Restart Program Logic Manual

This technical newsletter is a part of version 03, modification 00 of the System Control Program for IBM System/3 Model 12 (Program Number 5705-SC1). It also applies to the following IBM System/3 models: Models 4 and 6 (Program Number 5703-SC1) and Models 8 and 10 (Program Number 5702-SC1). This newsletter provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

77, 78

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

Includes information on the 3340 disk under *Table of Entries* in Section 3.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

READER'S COMMENT FORM

IBM System/3
Overlay Linkage Editor and
Checkpoint/Restart Programs
Logic Manual

SY21-0530-1

YOUR COMMENTS, PLEASE . . .

Your comments concerning this publication will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Publications, Dept 245

Fold

Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)