

Program Logic

IBM 1130 Disk Monitor Programming System, Version 2 Program Logic Manual

Program Numbers

1130-OS-005

1130-OS-006

This publication describes the internal logic of the IBM 1130 Disk Monitor Programming System, Version 2. The contents are intended for use by persons involved in program maintenance, and for system programmers who are altering the program design. Program logic information is not necessary for the use and operation of the program; therefore, distribution of this manual is limited to those who are performing the aforementioned functions.

Restricted Distribution

PREFACE

This publication is composed of four parts. Part 1 is a description of each of the components of the monitor system. Sections of Part 1 are devoted to:

- System Communication areas
- System Loader
- Cold Start Programs
- Resident Monitor
- Supervisor
- Core Image Loader
- Core Load Builder
- Disk Utility Program (DUP)
- Assembler Program
- FORTRAN Compiler
- System Library
- Stand-alone Utilities

Each description includes a discussion of the logical structure and functional operation of the component, table formats, and core storage layouts.

Part 2 is a description of the techniques and procedures for use by personnel involved in system maintenance and/or modification during error diagnosis and program analysis.

Part 3 is the flowcharts for the monitor system components described in Part 1.

Part 4 is the appendices provided to support Parts 1 through 3.

RESTRICTED DISTRIBUTION: This publication is intended for use by IBM personnel only and may not be made available to others without the approval of local IBM management.

First Edition (November 1967)

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

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 232, San Jose, California 95114.

CONVENTIONS OBSERVED

The following conventions have been observed in this publication:

1. Numbers written in the form /XXXX are hexadecimal numbers; numbers written without a preceding slash (/) are decimal numbers.
2. The diagrams showing the layouts of core storage are intended to illustrate the contents of core storage and their relative locations; no exact representation of size or proportion is intended.

PREREQUISITE PUBLICATIONS

Effective use of this publication requires that the reader be familiar with the following publications:

IBM 1130 Functional Characteristics (Form A26-5881)

IBM 1130 Input/Output Units (Form A26-5890)

IBM 1130 Assembler Language (Form C26-5927)

IBM 1130 Subroutine Library (Form C26-5929)

IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (Form C26-3717)

SECTION 1. INTRODUCTION	1	Supervisor Control Record Area (SCRA)	22
SECTION 2. COMMUNICATIONS AREAS	3	System Core Dump Program	23
In-core Communications Area (COMMA)	3	Auxiliary Supervisor	23
Disk-resident Communications Area (DCOM)	3	Core Layout	24
Drive- and Cartridge-dependent Parameters	3		
SECTION 3. SYSTEM LOADER	9	SECTION 7. CORE IMAGE LOADER	27
Flowcharts	9	Flowcharts	27
Phase 1	9	Phase 1	27
Functions	9	Phase 2	27
Buffers and I/O Areas	9	Core Layout	28
Communication From Phase 1 to Phase 2	9	Debugging/Analysis Aids	30
Phase 2	10		
Functions, Initial Load and Reload	10	SECTION 8. CORE LOAD BUILDER	33
Functions, Initial Load Only	10	Flowcharts	33
Functions, Reload Only	10	General Comments	33
Buffers and I/O Areas	10	Overlay Scheme and Core Layout	33
Subphases	11	Disk Buffers	34
Subphase 1	11	Core Image Buffer (CIB)	34
Subphase 2	11	Load Table	35
Subphase 3	12	LOCAL, NOCAL, and FILES Information	35
Core Layout	12	ISS Table	35
Cartridge Identification Sector	13	Interrupt Branch Table (IBT)	35
System Location Equivalence Table (SLET)	13	Incorporating Programs Into the Core Load	36
Reload Table	14	Pass 1	36
		Pass 2	36
SECTION 4. COLD START PROGRAMS	15	LOCALs and SOCALs	37
Flowcharts	15	Interrupt Level Subroutines (ILSs)	37
Cold Start Loader	15	Transfer Vector (TV)	37
Cold Start Program	15	Linkage to LOCALs	37
Core Layout	15	Linkage to the System Overlays (SOCALs)	38
		DEFINE FILE Table	39
SECTION 5. RESIDENT MONITOR	17	Phase Descriptions	40
Flowcharts	17	Phase 0	40
COMMA	17	Phase 1	40
Skeleton Supervisor	17	Phase 2	40
CALL LINK, CALL DUMP, CALL EXIT Processor	17	Phase 3	41
Error Traps	17	Phase 4	41
ILSs	17	Phase 5	41
Disk I/O Subroutine	17	Phase 6	42
		Phase 7	42
SECTION 6. SUPERVISOR	19	Phase 8	42
Flowcharts	19	Phase 9	42
Monitor Control Record Analyzer - Phase 1	19	Phase 10	42
JOB Control Record Processing	19	Phase 11	42
System Update Program	20	Phase 12	42
Other Control Record Processing	20	Debugging/Analysis Aids	42
XEQ Control Record Processor - Phase 2	21		
XEQ Control Record Processing	21	SECTION 9. DISK UTILITY PROGRAM (DUP)	43
Supervisor Control Record Processing	21	Flowcharts	43
LOCAL/NOCAL Control Record Processing	22	DUP Operation	43
FILES Control Record Processing	22	Core Storage Layout	43
		DUP Control Records	44

Location Equivalence Table (LET)/Fixed Location		SECTION 11. FORTRAN COMPILER	73
Equivalence Table (FLET)	44	Flowcharts	73
DUP Concatenated Communications Area (CATCO)	45	General Compiler Description	73
DUP Phase Descriptions	49	Phase Objectives	73
DUP COMMON (DUPCO)	49	Core Layout	75
DUP CONTROL (DCTL)	51	FORTRAN Communications Area	76
STORE	52	Phase Area	77
FILEQ	54	String Area	77
DDUMP	55	Symbol Table	78
DUMPLET/DUMPFLET	56	Statement String	80
DELETE	57	Compilation Errors	81
DEFINE	57	Compiler I/O	81
DEXIT	58	Fetching Compiler Phases	81
2501/1442 Card Interface (CFACE)	59	Phase Descriptions	82
Keyboard Interface (KFACE)	60	Phase 1	82
1134/1055 Paper Tape Interface (PFACE)	60	Phase 2	82
PRECI	62	Phase 3	83
DUP Diagnostic Aids	63	Phase 4	83
General	63	Phase 5	84
PRECI	63	Phase 6	84
STORE	63	Phase 7	85
		Phase 8	85
		Phase 9	86
		Phase 10	86
SECTION 10. ASSEMBLER PROGRAM	65	Phase 11	87
Flowcharts	65	Phase 12	88
Introduction	65	Phase 13	89
Program Operation	65	Phase 14	89
Assembler Communications Area	66	Phase 15	90
Overlay Area	66	Phase 16	91
Symbol Table	66	Phase 17	92
Intermediate I/O	67	Phase 18	93
Double-Buffering	67	Phase 19	94
Phase Descriptions	67	Phase 20	94
Phase 0	67	Phase 21	95
Phase 1	67	Phase 22	95
Phase 1A	68	Phase 23	95
Phase 2	68	Phase 24	96
Phase 2A	68	Phase 25	96
Phase 3	68	Phase 26	96
Phase 4	68	Phase 27	96
Phase 5	68		
Phase 6	69	SECTION 12. SYSTEM LIBRARY	99
Phase 7	69	Flowcharts	99
Phase 7A	69	Contents List	99
Phase 8	69	Interrupt Level Subroutines	105
Phase 8A	69	ILS02	105
Phase 9	69	ILS04	105
Phase 10	70	Mainline Programs	105
Phase 10A	71	Disk Initialization Program (DISC)	105
Phase 11	71	Print Cartridge ID (IDENT)	106
Phase 12	71	Change Cartridge ID (ID)	106
ERMSG	71	Disk Copy (COPY)	106
@ARCV	71	Delete CIB (DLCIB)	107
@APCV	71	Dump SLET Table (DSLET)	107
Core Layout	71		

SECTION 13. SYSTEM DEVICE SUBROUTINES	109	WRTYZ	146
SECTION 14. STAND-ALONE UTILITIES	111	PRNZ	147
Disk Cartridge Initialization Program (DCIP)	111	PRNTZ	148
Disk Initialization	111	PAPTZ	150
Disk Dump	111	CARD0	152
Disk Copy	112	CARD1	154
UCART	112	READ0	156
PROGRAM ANALYSIS PROCEDURES	113	READ1	157
Introduction	113	PNCHO	158
Program Analysis Procedures Summary	113	PNCH1	160
Identification of the Failing Component or Function	113	TYPE0	162
Subroutine Error Number/Error Stop Lists	115	PRNT1	164
Core Dump Procedure	115	PRNT3	166
Core Block Diagrams	115	PAPT1	168
Core Location Procedures	115	PAPTn	170
FAC	115	PLOT1	172
Arithmetic and Function Subprogram Error Indicators	118	OMPR1	174
LIBF TV	118	WRTY0	176
LIBF TV SOCAL Linkage	119	DISK1	177
CALL TV	119	DISKN	180
Disk I/O Subroutine	119	FLOWCHARTS	185
DFT (DEFINE FILE Table)	119	System Overview	185
Arrays	119	System Loader	186
Constants and Integers	119	Cold Start Programs	191
COMMON	119	Supervisor	193
In-core Subroutines	119	Core Image Loader	203
LOCAL/SOCAL Flipper (FLIPR)	120	Core Load Builder	205
LOCAL Area	120	Disk Utility Program	207
SOCAL Area	120	Assembler Program	220
Generalized Subroutine Maintenance/Analysis Procedure	120	FORTRAN Compiler	244
Trace Back Procedures	120	System Library	274
Subroutine Looping Capabilities	120	APPENDIX A. EXAMPLES OF FORTRAN OBJECT CODING	307
System Device Subroutines	120	APPENDIX B. LISTINGS	323
Library Subroutines	120	DCOM	323
Subroutine Data Charts	125	Resident Image	324
System Device Subroutine for Keyboard/Console		Resident Monitor	324
Printer	125	DISKZ	329
System Device Subroutine for 1442/1442	126	Equivalences	333
System Device Subroutine for 2501/1442	128	Cold Start Program	334
System Device Subroutine for Console Printer	130	Cross-Reference	335
System Device Subroutine for 1132	132	APPENDIX C. ABBREVIATIONS	339
System Device Subroutine for 1403	134	APPENDIX D. MICROFICHE REFERENCE TABLE	345
System Device Subroutine for 1134/1055	136	INDEX	349
System Device Subroutine for Disk - DISKZ	138		
CARDZ	140		
PNCHZ	142		
READZ	144		
TYPEZ	145		

Figures

1. Core Layout During System Loader Operation	13	9. Core Layout During Core Load Builder Operation . . .	34
2. Core Layout During Cold Start	16	10. Layout of the Transfer Vector	38
3. Core Layout During Supervisor Operation	25	11. SOCAL Linkage in the LIBF Transfer Vector	39
4. Core Layout on Supervisor Entry at \$EXIT (DISKZ in Core)	28	12. CALL Transfer Vector for SOCALs	39
5. Core Layout on Supervisor Entry at \$EXIT (DISKZ Not in Core)	29	13. Core Layout during Disk Utility Program Operation . .	43
6. Core Layout on Supervisor Entry at \$DUMP	29	14. Core Layout During Assembler Program Operation	72
7. Core Layout on Supervisor Entry at \$LINK (Link in Disk System Format)	30	15. Core Layout During FORTRAN Compiler Operation	76
8. Core Layout on Supervisor Entry at \$LINK (Link in Core Image Format)	31	16. FORTRAN Scan Example	93
		17. Core Layout During User Core Load Execution	119

Tables

1. The Contents of COMMA	4	5. FORTRAN Statement ID Word Type Codes	80
2. The Contents of DCOM	6	6. Conversion of FORTRAN FORMAT Specifications	87
3. The Contents of the FORTRAN Communications Area	77	7. FORTRAN Forcing Table	91
4. The Contents of the FORTRAN Symbol Table ID Word	79	8. Error Number List.	116
		9. Error Stop List	118

Flowdiagrams

1. General Procedure for Program Analysis	113	4. Generalized Subroutine Maintenance/Analysis Procedure	121
2. Procedure for Identification of the Failing Component or Function	114	5. Trace Back Procedures	122
3. Core Dump Procedure	115		

Flowcharts

DMS01. Disk Monitor System, System Overview	185	CST01. Cold Start Loader	191
SYL01. System Loader, General Flow	186	CST02. Cold Start Program	192
SYL02. System Loader, Phase 1	187	SUP01. Supervisor, Skeleton Supervisor	193
SYL03. System Loader, Phase 2	188	SUP02. Supervisor, Monitor Control Record Analyzer . .	194
SYL04. System Loader, Phase 2	189	SUP03. Supervisor, JOB Control Record Processor	195
SYL05. System Loader, Phase 2	190	SUP04. Supervisor, System Update Program	196

SUP05.	Supervisor, XEQ Control Record Analyzer	197
SUP06.	Supervisor, Supervisor Control Record Analyzer	198
SUP07.	Supervisor, Supervisor Control Record Analyzer	199
SUP08.	Supervisor, Supervisor Control Record Analyzer	200
SUP09.	Supervisor, System Core Dump Program	201
SUP10.	Supervisor, Auxiliary Supervisor	202
CIL01.	Core Image Loader, Phase 1	203
CIL02.	Core Image Loader, Phase 2	204
CLB01.	Core Load Builder, Initialization	205
CLB02.	Core Load Builder, Master Control	206
DUP01.	Disk Utility Program, CCAT	207
DUP02.	Disk Utility Program, DCTL	208
DUP03.	Disk Utility Program, STORE.	209
DUP04.	Disk Utility Program, FILEQ	210
DUP05.	Disk Utility Program, FILEQ	211
DUP06.	Disk Utility Program, FILEQ	212
DUP07.	Disk Utility Program, DDUMP	213
DUP08.	Disk Utility Program, DDUMP	214
DUP09.	Disk Utility Program, DUMPLET/DUMPFLET	215
DUP10.	Disk Utility Program, DELETE	216
DUP11.	Disk Utility Program, DEFINE	217
DUP12.	Disk Utility Program, DEXIT	218
DUP13.	Disk Utility Program, PRECI	219
ASM01.	Assembler Program, General Flow	220
ASM02.	Assembler Program, Phase 0	221
ASM03.	Assembler Program, Phase 1	222
ASM04.	Assembler Program, Phase 1A	223
ASM05.	Assembler Program, Phase 2	224
ASM06.	Assembler Program, Phase 2A	225
ASM07.	Assembler Program, Phase 3	226
ASM08.	Assembler Program, Phase 4	227
ASM09.	Assembler Program, Phase 5	228
ASM10.	Assembler Program, Phase 6	229
ASM11.	Assembler Program, Phase 7	230
ASM12.	Assembler Program, Phase 7A	231
ASM13.	Assembler Program, Phase 8	232
ASM14.	Assembler Program, Phase 8A	233
ASM15.	Assembler Program, Phase 9	234
ASM16.	Assembler Program, Phase 9	235
ASM17.	Assembler Program, Phase 9	236
ASM18.	Assembler Program, Phase 10	237
ASM19.	Assembler Program, Phase 10A	238
ASM20.	Assembler Program, Phase 11	239
ASM21.	Assembler Program, Phase 12	240
ASM22.	Assembler Program, Error Message Phase	241
ASM23.	Assembler Program, Read Conversion Phase	242
ASM24.	Assembler Program, Punch Conversion Phase	243
FOR01.	FORTRAN Compiler, General Flow	244
FOR02.	FORTRAN Compiler, Phase 1	245
FOR03.	FORTRAN Compiler, Phase 2	246
FOR04.	FORTRAN Compiler, Phase 3	247
FOR05.	FORTRAN Compiler, Phase 4	248
FOR06.	FORTRAN Compiler, Phase 4	249
FOR07.	FORTRAN Compiler, Phase 5	250
FOR08.	FORTRAN Compiler, Phase 5	251
FOR09.	FORTRAN Compiler, Phase 6	252
FOR10.	FORTRAN Compiler, Phase 7	253
FOR11.	FORTRAN Compiler, Phase 8	254
FOR12.	FORTRAN Compiler, Phase 9	255
FOR13.	FORTRAN Compiler, Phase 10	256
FOR14.	FORTRAN Compiler, Phase 11	257
FOR15.	FORTRAN Compiler, Phase 12	258
FOR16.	FORTRAN Compiler, Phase 13	259
FOR17.	FORTRAN Compiler, Phase 14	260
FOR18.	FORTRAN Compiler, Phase 15	261
FOR19.	FORTRAN Compiler, Phase 16	262
FOR20.	FORTRAN Compiler, Phase 17	263
FOR21.	FORTRAN Compiler, Phase 18	264
FOR22.	FORTRAN Compiler, Phase 19	265
FOR23.	FORTRAN Compiler, Phase 20	266
FOR24.	FORTRAN Compiler, Phase 21.	267
FOR25.	FORTRAN Compiler, Phase 22.	268
FOR26.	FORTRAN Compiler, Phase 23.	269
FOR27.	FORTRAN Compiler, Phase 24.	270
FOR28.	FORTRAN Compiler, Phase 25.	271
FOR29.	FORTRAN Compiler, Phase 26.	272
FOR30.	FORTRAN Compiler, Phase 27.	273
UTL01.	System Library, ID	274
UTL02.	System Library, FSLEN/FSYSU	275
UTL03.	System Library, ADRWS	276
UTL04.	System Library, DISC	277
UTL05.	System Library, RDREC	278
UTL06.	System Library, IDENT	279
UTL07.	System Library, CALPR	280
UTL08.	System Library, COPY	281
UTL09.	System Library, DLCIB	282
UTL10.	System Library, DSLET	283
UTL11.	System Library, MODIF	284
UTL12.	System Library, MODIF	285
UTL13.	System Library, MODIF	286
SCA01.	System Library, SCAT2 Call Processing	287
SCA02.	System Library, SCAT2 Interrupt Processing	288
SCA03.	System Library, SCAT2 Interrupt Processing	289
SCA04.	System Library, SCAT3 Call Processing	290
SCA05.	System Library, SCAT3 Interrupt Processing	291
SCA06.	System Library, SCAT3 Interrupt Processing	292
SCA07.	System Library, SCAT3 Interrupt Processing	293
FIO01.	System Library, FORTRAN Non-disk I/O	294
FIO02.	System Library, FORTRAN Non-disk I/O	295
FIO03.	System Library, FORTRAN Non-disk I/O	296
FIO04.	System Library, CARDZ	297
FIO05.	System Library, CARDZ	298
FIO06.	System Library, PRNTZ	299
FIO07.	System Library, PAPTZ	300
FIO08.	System Library, READZ	301
FIO09.	System Library, WRTYZ	302
FIO10.	System Library, PRNZ	303
FIO11.	System Library, PNCHZ	304
FIO12.	System Library, TYPEZ	305
FIO13.	System Library, HOLEZ	306

The 1130 Disk Monitor System, Version 2, consists of the following components:

Communication Areas

This component consists of the in-core communication area (COMMA) and the disk-resident communication area (DCOM).

Generally speaking, COMMA contains only those parameters required by the monitor system to fetch a program stored on disk in disk core image format (DCI).

DCOM contains all the parameters required by the monitor system that are not found in COMMA.

System Loader

This component provides the means for loading all, or reloading a part of, the monitor system onto disk. In other words, the System Loader generates the monitor system on disk.

Cold Start Programs

This component consists of the Cold Start Loader and the Cold Start Program.

The Cold Start Loader is the bootstrap loader used in the IPL procedure to initiate the operation of the Cold Start Program.

The Cold Start Program reads the monitor system, i. e., the Resident Monitor, into core storage and transfers control to it.

Resident Monitor

This component consists of three intermixed parts: (1) COMMA, (2) the Skeleton Supervisor, and (3) one of the three disk I/O subroutines -- DISKZ, DISK1, or DISKN.

COMMA is defined above, under Communication Areas.

The Skeleton Supervisor consists of the core-resident coding necessary to process CALL DUMP,

CALL LINK, and CALL EXIT statements, and various I/O traps.

One of the three disk I/O subroutines is present in the Resident Monitor at all times. The disk I/O subroutine in the Resident Monitor is the only such subroutine in core storage at any one time. Any of the three disk I/O subroutines can be used by the user. The DISKZ subroutine is used by the monitor system programs; DISKZ is initially loaded when a cold start is performed.

Supervisor

This component consists of the Monitor Control Record Analyzer (MCRA), the Supervisor Control Record Analyzer, the Auxiliary Supervisor, and the System Core Dump program.

The MCRA is the program that reads and analyzes the monitor control records, initiating the actions indicated on those control records.

The Supervisor Control Record Analyzer is the program that reads and analyzes the Supervisor control records, passing the information on these control records to the Core Load Builder.

The Auxiliary Supervisor is the program called to perform specialized supervisory functions for the monitor system.

The System Core Dump program is the program used to print all or selected portions of the contents of core storage on the principal print device. The dump can be dynamic (execution of the calling core load is resumed after the completion of the dump) or terminal (a CALL EXIT is executed after the completion of the dump).

Core Image Loader

This component consists of two parts, the first being an intermediate supervisor for the monitor system, the second being a loader for user and system programs in core image format.

Phase 1 of the Core Image Loader is fetched into core storage as the result of an entry to the Skeleton Supervisor. Phase 1 is the program that determines the type of entry made and the program(s)

to be fetched as a result.

Phase 2 of the Core Image Loader is the program that fetches into core storage and, if indicated, transfers control to the program(s) indicated by phase 1.

Core Load Builder

This component is the program that converts a mainline program from disk system format (DSF) to a core load, a program in disk core image format (DCI); that is, the Core Load Builder relocates the mainline program and all the subroutines required and constructs the other necessary parts of the core load, e. g., the transfer vector, LOCALs, and SOCALs.

Disk Utility Program (DUP)

This component provides the means for performing the following functions, largely through the use of control records only:

1. Make available the contents of disk storage in punched or printed format -- DUMP, DUMPPDATA.
2. Print a map of the contents of the variable portions of disk storage -- DUMPLET, DUMPFLET.
3. Store information on the disk in disk system format (DSF), disk data format (DDF), or disk core image format (DCI) -- STORE, STOREDATA, STOREDATA CI, STORECI, STOREMOD.
4. Remove information from the User/Fixed Area -- DELETE.
5. Alter the allocation of the Fixed Area on the disk or delete the Assembler Program and/or the FORTRAN Compiler from the monitor system -- DEFINE.
6. Initialize the Working Storage area on disk -- DWADR.
7. Provide file protection for the contents of disk storage.

Assembler Program

This component is the program that translates the statements of a source program written in the IBM 1130 Assembler Language into a program in disk system format (DSF).

FORTRAN Compiler

This component is the program that translates the statements of a source program written in the IBM 1130 Basic FORTRAN IV Language into a program in disk system format (DSF).

System Library

This component consists of (1) a complete library of input/output (except disk I/O), data conversion, arithmetic, and function subroutines, (2) selective dump subroutines, and (3) special programs for disk maintenance.

System Device Subroutines

This component consists of a library of special subroutines, one for each device (except the disk) used by the monitor system programs. These subroutines and DISKZ are the only device subroutines used by the monitor system programs.

Utilities

This component consists of the following stand-alone, self-loading utility programs:

1. The 2501/1442 Card Loader
2. The Disk Cartridge Initialization Program (DCIP)
3. The Core-Dump-to-Printer Program

In general the organization of and flow of control through the 1130 Disk Monitor System, Version 2, is shown in Flowchart DMS01.

THE IN-CORE COMMUNICATIONS AREA (COMMA)

COMMA includes, for the most part, only those system parameters that are required to link from one core load to another that is stored on disk in disk core image format (DCI). The exceptions are those parameters that would create awkward communication between monitor system programs if they resided in DCOM.

COMMA is not a single block of locations in the Resident Monitor; the system parameters that constitute COMMA are intermixed with the various parts of the Skeleton Supervisor.

Table 1 is a description of COMMA by parameter. The entries are arranged in alphabetic sequence for easy reference. See the listing of the Resident Monitor in Appendix B. Listings for the absolute addresses associated with the parameters in this table.

THE DISK-RESIDENT COMMUNICATIONS AREA (DCOM)

DCOM contains those parameters that must be passed from one monitor system program to another but are not found in COMMA.

Table 2 is a description of DCOM by parameter. The entries are arranged in alphabetic sequence for easy reference. See the listing of DCOM in Appendix B. Listings for the relative addresses associated with the parameters in this table.

DRIVE- AND CARTRIDGE-DEPENDENT PARAMETERS

Whenever a parameter that is associated with a disk cartridge is required for system use during a job, a table of five such parameters (a quintuple), one for each of the five possible drives, is reserved in COMMA or DCOM. The first of the five parameters is assigned a label. Such a parameter is said to be a drive- or cartridge-dependent parameter, whichever term is applicable.

The position in the quintuple indicates the logical drive number of the drive on which the associated cartridge is mounted. Thus, the first parameter in a quintuple is associated with logical drive zero, the second with logical drive one, etc. The assignment of logical drive numbers is done during JOB processing; that is, the logical drive numbers are assigned in the sequence specified on the JOB monitor control record. Thus, the first cartridge specified is assigned to logical drive zero, the second to logical drive one, etc. If no cartridges at all are specified, then the current logical drive zero is defined as logical drive zero for the job being defined. The drive- and cartridge-dependent parameters for all unspecified cartridges are cleared to zero, except for logical drive zero as noted above.

JOB processing includes the reading of DCOM and the ID sector from each specified cartridge and the setting up of the drive- and cartridge-dependent quintuples in DCOM on the master cartridge.

Initialization of the quintuples is done during cold start processing, which defines logical drive zero (and all associated drive- and cartridge-dependent parameters for logical drive zero) as the physical drive selected in the Console Entry switches (see Section 4. Cold Start Programs). All other values in the drive- and cartridge-dependent quintuples are cleared to zero.

Table 1. The Contents of COMMA

LABEL	DESCRIPTION										
\$ACDE through \$ACDE+4	\$ACDE contains the device code for the physical disk drive assigned as logical disk drive 0. \$ACDE+1 through \$ACDE+4 contain the device codes for logical disk drives 1, 2, 3, and 4, respectively. The device code is contained in bits 11-15.										
\$ACEX and \$ACEX+1	\$ACEX and \$ACEX+1 are the locations in which the contents of the accumulator and extension, respectively, are saved by the Supervisor when entered at the \$DUMP entry point.										
\$CCAD	\$CCAD contains the address of the lowest-addressed word of COMMON to be saved on the Core Image Buffer (CIB) by the Core Image Loader.										
\$CH12	\$CH12 contains the address of \$CPTR, \$1132, or \$1403 depending upon the device defined as the principal print device--the Console Printer, 1132 Printer, or 1403 Printer, respectively.										
\$CIBA	\$CIBA contains the sector address of the first sector of the Core Image Buffer (CIB) in use by the monitor system programs during the current job. The logical disk drive number is contained in bits 0-3.										
\$CIBA-1	\$CIBA-1 contains 4095 minus the location of \$CIBA. This value is used as the word count (in conjunction with \$CIBA, which contains the sector address of the CIB) in saving the first 4K of core storage following an entry at \$DUMP.										
\$CILA	\$CILA contains the address of the end of the disk I/O subroutine currently in core storage, minus 4. \$CILA always points to the word count (followed by the sector address) of phase 1 of the Core Image Loader.										
\$CLSW	\$CLSW is a switch indicating to phase 2 of the Core Image Loader the function it is to perform. The switch settings are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>Positive</td> <td>Load the indicated disk I/O subroutine</td> </tr> <tr> <td>Zero or Negative</td> <td>Load the indicated core load and its required disk I/O subroutine. Zero indicates that the core load has just been built by the Core Load Builder; negative indicates that the core load is stored in the User or Fixed Area in core image format.</td> </tr> </tbody> </table>	Setting	Meaning	Positive	Load the indicated disk I/O subroutine	Zero or Negative	Load the indicated core load and its required disk I/O subroutine. Zero indicates that the core load has just been built by the Core Load Builder; negative indicates that the core load is stored in the User or Fixed Area in core image format.				
Setting	Meaning										
Positive	Load the indicated disk I/O subroutine										
Zero or Negative	Load the indicated core load and its required disk I/O subroutine. Zero indicates that the core load has just been built by the Core Load Builder; negative indicates that the core load is stored in the User or Fixed Area in core image format.										
\$COMN	\$COMN contains the number of words of COMMON defined for the core load currently in execution.										
\$CORE	\$CORE contains a code indicating the number of words of core storage within which the monitor system programs are to operate. The codes are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Code</th> <th>Size of Core Storage</th> </tr> </thead> <tbody> <tr> <td>/1000</td> <td>4096 words</td> </tr> <tr> <td>/2000</td> <td>8192 words</td> </tr> <tr> <td>/4000</td> <td>16384 words</td> </tr> <tr> <td>/8000</td> <td>32768 words</td> </tr> </tbody> </table>	Code	Size of Core Storage	/1000	4096 words	/2000	8192 words	/4000	16384 words	/8000	32768 words
Code	Size of Core Storage										
/1000	4096 words										
/2000	8192 words										
/4000	16384 words										
/8000	32768 words										
\$CPTR	\$CPTR is a dummy channel 12 indicator for the Console Printer.										
\$CTSW	\$CTSW is a switch indicating that a monitor control record has been detected by a monitor system program other than the Supervisor. The switch settings are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>Positive</td> <td>Monitor control record detected</td> </tr> <tr> <td>Zero</td> <td>Monitor control record not detected</td> </tr> </tbody> </table>	Setting	Meaning	Positive	Monitor control record detected	Zero	Monitor control record not detected				
Setting	Meaning										
Positive	Monitor control record detected										
Zero	Monitor control record not detected										

LABEL	DESCRIPTION								
\$CWCT	\$CWCT contains the number of words of COMMON to be saved on the Core Image Buffer (CIB) by the Core Image Loader.								
\$CWCT+1	\$CWCT+1 contains the sector address of the first sector of the Core Image Buffer (CIB) to be used for the saving of COMMON by the Core Image Loader. The logical disk drive number is contained in bits 0-3.								
\$CXRI	\$CXRI is the location in which the contents of index register 1 are saved by the Skeleton Supervisor.								
\$CYLN through \$CYLN+4	\$CYLN contains the sector address of sector 0 on the cylinder over which the access arm on logical disk drive 0 is currently positioned. \$CYLN+1 through \$CYLN+4 contain analogous sector addresses for logical disk drives 1, 2, 3, and 4, respectively.								
\$DABL	\$DABL contains the second word of the IOCC used to reset the Synchronous Communications Adapter. \$DABL is, therefore, aligned on an odd word boundary. \$DABL contains /5540, the bit configuration of an Initiate Write with modifier bit 9 on.								
\$DADR	\$DADR contains the disk block address of the first sector of the program or core load to be fetched into core storage and executed.								
\$DBSY	\$DBSY is a switch indicating whether or not a disk I/O operation is in progress. The switch settings are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Disk I/O not in progress</td> </tr> <tr> <td>non-zero</td> <td>Disk I/O in progress</td> </tr> </tbody> </table> <p>\$DBSY is simultaneously used as a retry counter by DISKZ and DISK1.</p>	Setting	Meaning	zero	Disk I/O not in progress	non-zero	Disk I/O in progress		
Setting	Meaning								
zero	Disk I/O not in progress								
non-zero	Disk I/O in progress								
\$DCDE	\$DCDE contains the number of the logical disk drive on which the program to be fetched by the Core Image Loader is to be found. The drive number is contained in bits 0-3.								
\$DCYL through \$DCYL+14	\$DCYL through \$DCYL+2 contain the defective cylinder addresses (the contents of words 1, 2, and 3 of sector 0, cylinder 0) for the cartridge mounted on logical disk drive 0. \$DCYL+3 through \$DCYL+14 contain analogous addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively.								
\$DDSW	\$DDSW contains the device status word (DSW) sensed during the last disk I/O operation performed.								
\$DMPF	\$DMPF contains the contents of the word following the branch to the \$DUMP entry point, i.e., the dump format code.								
\$DREQ	\$DREQ is a switch indicating the disk I/O subroutine that has been requested. The switch settings are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>positive</td> <td>DISKN</td> </tr> <tr> <td>zero</td> <td>DISK1</td> </tr> <tr> <td>negative</td> <td>DISKZ</td> </tr> </tbody> </table>	Setting	Meaning	positive	DISKN	zero	DISK1	negative	DISKZ
Setting	Meaning								
positive	DISKN								
zero	DISK1								
negative	DISKZ								
\$DZIN	\$DZIN is a switch indicating the disk I/O subroutine presently in core storage. The switch settings are as follows: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>negative</td> <td>DISKZ is in core</td> </tr> <tr> <td>zero</td> <td>DISK1 is in core</td> </tr> <tr> <td>positive</td> <td>DISKN is in core</td> </tr> </tbody> </table>	Setting	Meaning	negative	DISKZ is in core	zero	DISK1 is in core	positive	DISKN is in core
Setting	Meaning								
negative	DISKZ is in core								
zero	DISK1 is in core								
positive	DISKN is in core								

Table 1. The Contents of COMMA (Continued)

LABEL	DESCRIPTION								
\$FPAD through \$FPAD+4	\$FPAD contains the sector address of the first sector of Working Storage on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. \$FPAD+1 through \$FPAD+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively. \$FPAD through \$FPAD+4 are effectively the file-protection addresses for the cartridges in use. These addresses are adjusted in non-temporary mode only.								
\$HASH through \$HASH+11	\$HASH through \$HASH+11 are a work area used variously by the monitor system programs.								
\$IBSY	\$IBSY is a switch indicating whether or not an I/O operation involving the principal I/O device is in progress. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Principal I/O device not busy</td> </tr> <tr> <td>non-zero</td> <td>Principal I/O device busy</td> </tr> </tbody> </table>	Setting	Meaning	zero	Principal I/O device not busy	non-zero	Principal I/O device busy		
Setting	Meaning								
zero	Principal I/O device not busy								
non-zero	Principal I/O device busy								
\$IBT2	\$IBT2 contains the address of the interrupt branch table (IBT) for interrupt level 2. Since the disk is the only device on interrupt level 2, \$IBT2 contains the address of the interrupt entry point in the disk I/O subroutine currently in core storage.								
\$IBT4	\$IBT4 contains the address of the interrupt branch table (IBT) for interrupt level 4 used by the program currently in control.								
\$IOCT	\$IOCT is the IOCS counter for I/O operations. \$IOCT is incremented by 1 for each I/O operation initiated. \$IOCT is decremented by 1 for each I/O operation completed or terminated. \$IOCT equals zero when all I/O operations have been completed.								
\$IREQ	\$IREQ contains the address of the subroutine servicing the INTERRUPT REQUEST key on the Keyboard (interrupt level 4). This address is supplied by the user core load using the INTERRUPT REQUEST key. Unless an address is supplied \$IREQ contains the address of the \$DUMP entry point.								
\$KCSW	\$KCSW is a switch indicating whether or not (1) the Keyboard has been defined as the principal input device and/or (2) the Console Printer has been defined as the principal print device. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Settings</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>negative</td> <td>Either the Console Printer is the principal print device or the Keyboard is the principal input device, but not both</td> </tr> <tr> <td>zero</td> <td>Neither is the Console Printer the principal print device nor is the Keyboard the principal input device</td> </tr> <tr> <td>positive</td> <td>The Console Printer is the principal print device and the Keyboard is the principal input device</td> </tr> </tbody> </table> Depending on the setting on \$KCSW, the system device subroutine for the Keyboard/Console Printer either permits or inhibits the overlapping of input and output.	Settings	Meaning	negative	Either the Console Printer is the principal print device or the Keyboard is the principal input device, but not both	zero	Neither is the Console Printer the principal print device nor is the Keyboard the principal input device	positive	The Console Printer is the principal print device and the Keyboard is the principal input device
Settings	Meaning								
negative	Either the Console Printer is the principal print device or the Keyboard is the principal input device, but not both								
zero	Neither is the Console Printer the principal print device nor is the Keyboard the principal input device								
positive	The Console Printer is the principal print device and the Keyboard is the principal input device								
\$LAST	\$LAST is a switch indicating whether or not the last card has been read by the system device subroutine servicing the card input device. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Settings</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Last card has not been read</td> </tr> <tr> <td>non-zero</td> <td>Last card has been read</td> </tr> </tbody> </table>	Settings	Meaning	zero	Last card has not been read	non-zero	Last card has been read		
Settings	Meaning								
zero	Last card has not been read								
non-zero	Last card has been read								
\$LKNM and \$LKNM+1	\$LKNM and \$LKNM+1 contain the name, in name code, of the program or core load to be executed next. \$LKNM is aligned on an even word boundary.								

LABEL	DESCRIPTION								
\$LSAD	\$LSAD contains the absolute sector address of the first sector of the first LOCAL (or SOCAL if there are no LOCALs) for the core load currently in core. The logical disk drive number is contained in bits 0-3.								
\$NDUP	\$NDUP is a switch indicating whether or not DUP operations may be performed. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Permit DUP operations</td> </tr> <tr> <td>non-zero</td> <td>Inhibit DUP operations</td> </tr> </tbody> </table>	Setting	Meaning	zero	Permit DUP operations	non-zero	Inhibit DUP operations		
Setting	Meaning								
zero	Permit DUP operations								
non-zero	Inhibit DUP operations								
\$NEND	\$NEND is equivalent to the address of the end of DISKN plus 1.								
\$NXEQ	\$NXEQ is a switch indicating whether or not execution of a user core load may be performed. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Permit core load execution</td> </tr> <tr> <td>non-zero</td> <td>Inhibit core load execution</td> </tr> </tbody> </table>	Setting	Meaning	zero	Permit core load execution	non-zero	Inhibit core load execution		
Setting	Meaning								
zero	Permit core load execution								
non-zero	Inhibit core load execution								
\$PAUS	\$PAUS is a switch set by every ISS that does not set \$IOCT when initiating an I/O operation, e.g., SCAT1. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Exit from the PAUS subroutine</td> </tr> <tr> <td>non-zero</td> <td>Branch back to the WAIT in the PAUS subroutine</td> </tr> </tbody> </table>	Setting	Meaning	zero	Exit from the PAUS subroutine	non-zero	Branch back to the WAIT in the PAUS subroutine		
Setting	Meaning								
zero	Exit from the PAUS subroutine								
non-zero	Branch back to the WAIT in the PAUS subroutine								
\$PGCT	\$PGBSY is a switch indicating whether or not an I/O operation involving the principal print device is in progress. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Principal print device not busy</td> </tr> <tr> <td>non-zero</td> <td>Principal print device busy</td> </tr> </tbody> </table>	Setting	Meaning	zero	Principal print device not busy	non-zero	Principal print device busy		
Setting	Meaning								
zero	Principal print device not busy								
non-zero	Principal print device busy								
\$PGCT	\$PGCT contains the number, in binary, of the page of the job listing currently being printed.								
\$PHSE	\$PHSE contains the SLET ID number (in bits 8-15) of the phase of the monitor system program currently in control, excepting the Cold Start Program and the Skeleton Supervisor. \$PHSE always contains zero when a user core load is in control. Bits 0-7 of \$PHSE sometimes contain a subphase ID number.								
\$RMSW	\$RMSW is a switch indicating the entry point at which the Skeleton Supervisor was entered and, hence, the type of CALL causing the Skeleton Supervisor to be entered. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>positive</td> <td>Entry at \$DUMP</td> </tr> <tr> <td>zero</td> <td>Entry at \$LINK</td> </tr> <tr> <td>negative</td> <td>Entry at \$EXIT</td> </tr> </tbody> </table>	Setting	Meaning	positive	Entry at \$DUMP	zero	Entry at \$LINK	negative	Entry at \$EXIT
Setting	Meaning								
positive	Entry at \$DUMP								
zero	Entry at \$LINK								
negative	Entry at \$EXIT								
\$RW CZ	\$RW CZ is a switch indicating the type of operation last performed by the CARDZ subroutine. The switch settings are as follows: <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Last operation a Read</td> </tr> <tr> <td>non-zero</td> <td>Last operation a Punch</td> </tr> </tbody> </table>	Setting	Meaning	zero	Last operation a Read	non-zero	Last operation a Punch		
Setting	Meaning								
zero	Last operation a Read								
non-zero	Last operation a Punch								
\$SCAN through \$SCAN+7	\$SCAN through \$SCAN+7 are an area used by the 1132 Printer when printing a line. This area is also used as a work area by the monitor system programs.								

Table 1. The Contents of COMMA (Concluded)

LABEL	DESCRIPTION										
\$SNLT	<p>\$SNLT is the location used for sense light simulation by FORTRAN programs. The bits are used as follows:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Sense Light</th> </tr> </thead> <tbody> <tr> <td>14</td> <td>1</td> </tr> <tr> <td>13</td> <td>2</td> </tr> <tr> <td>12</td> <td>3</td> </tr> <tr> <td>11</td> <td>4</td> </tr> </tbody> </table>	Bit	Sense Light	14	1	13	2	12	3	11	4
Bit	Sense Light										
14	1										
13	2										
12	3										
11	4										
\$SYSC	\$SYSC contains the version and modification level numbers identifying the 1130 Disk Monitor System. Bits 0-7 contain the version number; bits 8-15 contain the modification level number.										
\$UFDR	\$UFDR contains the number of the logical disk drive on which the unformatted I/O area in use by the monitor system programs during the current job is to be found. The drive number is contained in bits 0-3.										
\$UFIO	\$UFIO contains the displacement, in sectors, from the start of the unformatted I/O area to the sector at which the writing or reading of the next logical record to or from the unformatted I/O area is to begin.										
\$ULET through \$ULET+4	\$ULET contains the sector address of the first sector of LET on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. \$ULET+1 through \$ULET+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively.										
\$WRD1	\$WRD1 contains the address at which the first word of the core image header of the core load to be/being executed will/does reside.										
\$WSDR	\$WSDR contains the number of the logical disk drive on which the Working Storage in use by the monitor system programs during the current job is to be found. The drive number is contained in bits 0-3.										
\$ZEND	\$ZEND is equivalent to the address of the end of DISKZ plus 1.										
\$1END	\$1END is equivalent to the address of the end of DISK1 plus 1.										
\$1132	<p>\$1132 is a switch indicating whether or not channel 12 has been detected on the 1132 Printer. The switch settings are as follows:</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Channel 12 not detected or a skip to channel 1 executed</td> </tr> <tr> <td>non-zero</td> <td>Channel 12 detected</td> </tr> </tbody> </table>	Setting	Meaning	zero	Channel 12 not detected or a skip to channel 1 executed	non-zero	Channel 12 detected				
Setting	Meaning										
zero	Channel 12 not detected or a skip to channel 1 executed										
non-zero	Channel 12 detected										
\$1403	<p>\$1403 is a switch indicating whether or not channel 12 has been detected on the 1403 Printer. The switch settings are as follows:</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Channel 12 not detected or a skip to channel 1 executed</td> </tr> <tr> <td>non-zero</td> <td>Channel 12 detected</td> </tr> </tbody> </table>	Setting	Meaning	zero	Channel 12 not detected or a skip to channel 1 executed	non-zero	Channel 12 detected				
Setting	Meaning										
zero	Channel 12 not detected or a skip to channel 1 executed										
non-zero	Channel 12 detected										

Table 2. The Contents of DCOM

LABEL	DESCRIPTION						
#ANDU through #ANDU+4	<p>#ANDU contains the displacement, in disk blocks, from word 0, sector 0, cylinder 0 on the cartridge mounted on logical disk drive 0 to the last disk block of the User Area on that cartridge, plus 1 disk block. #ANDU+1 through #ANDU+4 contain analogous displacements for the cartridges on logical disk drives 1, 2, 3, and 4, respectively. #ANDU through #ANDU+4 are effectively the adjusted addresses of the ends of the User Areas on the cartridges in use. These addresses are adjusted instead of #BNDU through #BNDU+4 during temporary mode, in parallel with #BNDU through #BNDU+4 during non-temporary mode.</p>						
#BNDU through #BNDU+4	<p>#BNDU contains the displacement, in disk blocks, from word 0, sector 0, cylinder 0 on the cartridge mounted on logical disk drive 0 to the last disk block of the User Area on that cartridge, plus 1 disk block. #BNDU+1 through #BNDU+4 contain analogous displacements for the cartridges on logical disk drives 1, 2, 3, and 4, respectively. #BNDU through #BNDU+4 are effectively the base addresses of the ends of the User Areas on the cartridges in use. These addresses are adjusted only during non-temporary mode, in parallel with #ANDU through #ANDU+4.</p>						
#CBSW	<p>#CBSW is a switch indicating to the Core Load Builder the type of exit to be made. The switch settings are as follows:</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Return to Core Image Loader</td> </tr> <tr> <td>non-zero</td> <td>Return to DUP</td> </tr> </tbody> </table>	Setting	Meaning	zero	Return to Core Image Loader	non-zero	Return to DUP
Setting	Meaning						
zero	Return to Core Image Loader						
non-zero	Return to DUP						
#CIAD	#CIAD contains the relative location in sector @IAD (within DISKZ) where the address of the Core Image Loader is to be found.						
#CIBA through #CIBA+4	#CIBA contains the sector address of the Core Image Buffer (CIB) on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. #CIBA+1 through #CIBA+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively.						
#CIDN through #CIDN+4	#CIDN contains the ID (the contents of word 4, sector 0, cylinder 0) of the cartridge mounted on logical disk drive 0. #CIDN+1 through #CIDN+4 contain the IDs of the cartridges on logical disk drives 1, 2, 3, and 4, respectively.						
#CSHN through #CSHN+4	#CSHN contains the number of sectors available for expansion of the monitor system programs on the system cartridge mounted on logical disk drive 0. #CSHN+1 through #CSHN+4 contain analogous numbers for any system cartridges mounted on logical disk drives 1, 2, 3, and 4, respectively.						
#DBCT	#DBCT contains the number of disk blocks occupied by the program, core load, or data file named on a DUP control record.						
#DCSW	<p>#DCSW is a switch indicating to the ADRWS program the type of exit to be made. The switch settings are as follows:</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Branch to \$EXIT in Skeleton Supervisor</td> </tr> <tr> <td>non-zero</td> <td>Return to DUP</td> </tr> </tbody> </table>	Setting	Meaning	zero	Branch to \$EXIT in Skeleton Supervisor	non-zero	Return to DUP
Setting	Meaning						
zero	Branch to \$EXIT in Skeleton Supervisor						
non-zero	Return to DUP						
#ENTY	#ENTY contains the address of entry point 1 in the program placed into system Working Storage by the Assembler or FORTRAN Compiler. This address is the address to be placed into word 12 of the DSF program header by DUP or the address from which word 1 of the core image header is generated by the Core Load Builder.						

Table 2. The Contents of DCOM (Continued)

LABEL	DESCRIPTION								
#FCNT	#FCNT contains the number of files defined for the core load being built or the execution to be initiated.								
#FHOL	#FHOL contains the displacement, in disk blocks, from word 0, sector 0, cylinder 0 to the first disk block of the largest unused (IDUMY) area in the Fixed Area on the cartridge to which a STORE operation is to be made.								
#FLET through #FLET+4	#FLET contains the sector address of the first sector of FLET on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. #FLET+1 through #FLET+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively.								
#FMAT through #FMAT+4	#FMAT is a switch indicating the format of the contents, if any, of Working Storage on the cartridge mounted on logical disk drive 0. #FMAT+1 through #FMAT+4 are analogous switches for Working Storage on logical disk drives 1, 2, 3 and 4, respectively. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>negative</td> <td>disk core image format (DCI)</td> </tr> <tr> <td>zero</td> <td>disk system format (DSF)</td> </tr> <tr> <td>positive</td> <td>disk data format (DDF)</td> </tr> </tbody> </table>	Setting	Meaning	negative	disk core image format (DCI)	zero	disk system format (DSF)	positive	disk data format (DDF)
Setting	Meaning								
negative	disk core image format (DCI)								
zero	disk system format (DSF)								
positive	disk data format (DDF)								
#FPAD through #FPAD+4	#FPAD contains the sector address of the first sector of Working Storage on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. #FPAD+1 through #FPAD+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively. #FPAD through #FPAD+4 are effectively the file-protection addresses for the cartridges in use. These addresses are adjusted in non-temporary mode only.								
#FRDR	#FRDR contains the number of the logical disk drive on which the cartridge specified by the "FROM" cartridge ID (in columns 31-34 of the DUP control record) is mounted. The drive number is contained in bits 12-15. A negative number indicates that no ID was specified.								
#FSZE	#FSZE contains the number of disk blocks contained in the largest unused (IDUMY) area in the Fixed Area on the cartridge to which a STORE operation is to be made.								
#JBSW	#JBSW is a switch indicating the mode of operation established by the last JOB monitor control record. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>temporary mode</td> </tr> <tr> <td>non-zero</td> <td>non-temporary mode.</td> </tr> </tbody> </table>	Setting	Meaning	zero	temporary mode	non-zero	non-temporary mode.		
Setting	Meaning								
zero	temporary mode								
non-zero	non-temporary mode.								
#LCNT	#LCNT contains the number of LOCALs specified for the execution to be initiated or the core load being built.								
#MDF1	#MDF1 contains, in bits 8-15, the number of DUP control records to be processed by DUP when called by the MODIF program. #MDF1 also contains, in bits 0-7, the number of errors detected by DUP during the processing of the DUP control records for the MODIF program.								
#MDF2	#MDF2 is a switch indicating to DUP that control must be returned to the MODIF program. The switch setting are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Do not return to MODIF</td> </tr> <tr> <td>non-zero</td> <td>Return to MODIF</td> </tr> </tbody> </table>	Setting	Meaning	zero	Do not return to MODIF	non-zero	Return to MODIF		
Setting	Meaning								
zero	Do not return to MODIF								
non-zero	Return to MODIF								

LABEL	DESCRIPTION								
#MPSW	#MPSW is a switch indicating to the Core Load Builder whether or not a core map is to be printed for each core load built during the current execution. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>Do not print a core map</td> </tr> <tr> <td>non-zero</td> <td>Print a core map</td> </tr> </tbody> </table>	Setting	Meaning	zero	Do not print a core map	non-zero	Print a core map		
Setting	Meaning								
zero	Do not print a core map								
non-zero	Print a core map								
#NAME and #NAME+1	#NAME and #NAME+1 contain the name, in name code, of the program, core load, or data file currently being processed by the Supervisor, DUP, Core Load Builder, or Core Image Loader. The name is obtained from a control record or from a LET/FLET search. #NAME is aligned on an even word boundary.								
#NCNT	#NCNT contains the number of NOCALs specified for the execution to be initiated or the core load being built.								
#PCID through #PCID+4	#PCID contains the ID (the contents of word 4, sector 0, cylinder 0) of the cartridge mounted on physical disk drive 0, if that drive is "ready". #PCID+1 through #PCID+4 contain the IDs of the cartridges on physical disk drives 1, 2, 3, and 4, respectively, if the corresponding drives are "ready". The entries for "not ready" drives contain zeroes.								
#PIOD	#PIOD is a switch indicating the device defined as the principal I/O device for the system. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Principal I/O Device</th> </tr> </thead> <tbody> <tr> <td>positive</td> <td>2501 with 1442, any model</td> </tr> <tr> <td>zero</td> <td>1442, Model 6 or 7</td> </tr> <tr> <td>negative</td> <td>1134 with 1055</td> </tr> </tbody> </table>	Setting	Principal I/O Device	positive	2501 with 1442, any model	zero	1442, Model 6 or 7	negative	1134 with 1055
Setting	Principal I/O Device								
positive	2501 with 1442, any model								
zero	1442, Model 6 or 7								
negative	1134 with 1055								
#PPTR	#PPTR is a switch indicating the device defined as the principal print device for the system. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Principal Print Device</th> </tr> </thead> <tbody> <tr> <td>positive</td> <td>1403 Printer</td> </tr> <tr> <td>zero</td> <td>1132 Printer</td> </tr> <tr> <td>negative</td> <td>Console Printer</td> </tr> </tbody> </table>	Setting	Principal Print Device	positive	1403 Printer	zero	1132 Printer	negative	Console Printer
Setting	Principal Print Device								
positive	1403 Printer								
zero	1132 Printer								
negative	Console Printer								
#RP67	#RP67 is a switch indicating the type of card I/O device present on the system. The switch settings are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Card I/O Device</th> </tr> </thead> <tbody> <tr> <td>zero</td> <td>2501 with 1442, Model 5</td> </tr> <tr> <td>positive</td> <td>1442, Model 6 or 7</td> </tr> </tbody> </table>	Setting	Card I/O Device	zero	2501 with 1442, Model 5	positive	1442, Model 6 or 7		
Setting	Card I/O Device								
zero	2501 with 1442, Model 5								
positive	1442, Model 6 or 7								
#SCRA through #SCRA+4	#SCRA contains the sector address of the first sector of the Supervisor Control Record Area (SCRA) on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. #SCRA+1 through #SCRA+4 contain analogous sector addresses for any system cartridges on logical disk drives 1, 2, 3, and 4, respectively.								
#SYSC	#SYSC, on a system cartridge, contains the version and modification level numbers for the currently resident 1130 Disk Monitor System. Bits 0-7 contain the version number; bits 8-15 contain the modification level number. #SYSC on a non-system cartridge contains zeroes.								
#TODR	#TODR contains the number of the logical disk drive on which the cartridge specified by the "TO" cartridge ID (in columns 36-40 of the DUP control record) is mounted. The drive number is contained in bits 12-15. A negative number indicates that no ID was specified.								

Table 2. The Contents of DCOM (Concluded)

LABEL	DESCRIPTION
#UHOL	#UHOL contains the displacement, in disk blocks, from word 0, sector 0, cylinder 0 to the first disk block of the largest unused (IDUMMY) area in the User Area on the cartridge to which a STORE operation is to be made.
#ULET through #ULET+4	#ULET contains the sector address of the first sector of LET on the cartridge mounted on logical disk drive 0. The logical disk drive number is contained in bits 0-3. #ULET+1 through #ULET+4 contain analogous sector addresses for the cartridges on logical disk drives 1, 2, 3, and 4, respectively.
#USZE	#USZE contains the number of disk blocks contained in the largest unused (IDUMMY) area in the User Area on the cartridge to which a STORE operation is to be made.
#WSCT through #WSCT+4	#WSCT contains the number of disk blocks occupied by a program, core load, or data file placed into Working Storage on the cartridge mounted on logical disk drive 0. #WSCT+1 through #WSCT+4 contain analogous disk block counts for the contents of Working Storage on logical disk drives 1, 2, 3, and 4, respectively.

FLOWCHARTS

General: SYL01

p186

Phase 1: SYL02

p187

Phase 2: SYL03 - SYL05

pp188-190

PHASE 1

FUNCTIONS

Phase 1 determines which card I/O subroutine is to be used, i. e., 2501 or 1442. If 2501, a naturally relocatable 2501 subroutine overlays the 1442 subroutine that is part of this phase.

Phase 1 also reads the Console Entry switches to get the physical drive number (0-4) for the cartridge to be loaded. It sets up the PROGRAM STOP key interrupt trap. The Keyboard INTERRUPT REQUEST key is made non-effective until the monitor system programs have been loaded.

Phase 1 reads the Resident Monitor and DISKZ object decks into core storage and initializes them preparatory to loading phase 2 of the System Loader to disk.

Phase 1 picks up defective track data from sector @IDAD on the cartridge to be used to initialize the disk I/O subroutine.

Phase 1 loads phase 2 to cylinders 198 and 199 (sectors /0630 through /063F) on disk. (These sector addresses are absolute; they are assembled as part of the phase 2 deck.) If an initial load is being performed, subphase 1 is loaded to sector /0634, overlaying a part of the reload processing subroutine. Subphases 2 and 3 are loaded to sectors /063B and /063C.

If an initial load is being performed, phase 1 stores the Resident Image on sector @RIAD. (DISKZ is stored on disk by phase 2.)

Phase 1 processes the System Configuration records, saving the data obtained in the phase 1 communications area for use by phase 2. At the end of the Configuration records processing, the accumulated data is checked for errors and consolidated for phase 2. If a CORE control record is present, its contents are processed and replace the calculated core size.

Phase 1 processes the PHID control record, checks it for errors, and saves the data obtained from it for use by phase 2. The version and modification level number is taken from the PHID control record and saved.

Phase 1 fetches phase 2 from disk into core storage and transfers control to it at MAING.

BUFFERS AND I/O AREAS

Card Input

A: an 80-word buffer that contains card images in left-justified 12 bit/word format, as read by the 1442 or 2501 card I/O subroutine.

B: a 60-word buffer that contains the 12-bit/word data from buffer A after it has been compressed to 16 bits/word.

Paper Tape Input

A: an 80-word buffer for PTTC/8 records.

B: a 60-word buffer into which binary data from buffer BIGCB is compressed.

C: a 1-word buffer used for the Delete character test when reading binary paper tape records.

BIGCB: a 108-word buffer for binary paper tape records, 108 frames, left-justified.

Disk Input and Output

BUFF1: a 320-word buffer used in all disk I/O operations.

COMMUNICATION FROM PHASE 1 TO PHASE 2

The card I/O subroutine (2501 or 1442) that was selected by phase 1 for use during System Loader operation is written onto sector /0638 to be used by phase 2 also.

Information acquired by phase 1 is accumulated in the area in core storage bounded by EDIT and EDIT+/0140. This 320-word area, which also contains a number of error messages, is written to sector /0632 before phase 2 is entered.

PHASE 2

FUNCTIONS, INITIAL LOAD AND RELOAD

Phase 2 checks the phase ID number sequence for ascending order during initial loads only.

Phase 2 performs a checksum test on all type A (data) records.

Phase 2 builds the Reload Table in core storage as the monitor system program phases are loaded. Each 3-word entry in the table consists of the ID number of a phase requesting SLET data, the relative location within the requesting phase where the SLET data is to be stored, and the number of SLET items to be supplied by the System Loader. On an initial load, this Reload Table is written to disk in sector @RTBL.

If so indicated by the Load Mode control record, phase 2 bypasses the FORTRAN Compiler and/or the Assembler Program.

Phase 2 updates the version and modification level numbers in the parameter #SYSC in sector @DCOM of the cartridge. These numbers are taken directly from the PHID control record. No comparison with previous version and modification level numbers is made.

Phase 2 determines from the data obtained from the System Configuration records which devices are the principal I/O and principal print devices. Phase 2 builds five special sets of SLET entries for the specified devices as well as for the principal I/O and print device conversion subroutines.

Phase 2 steps through the Reload Table and searches out every phase requesting SLET data. It then searches out the SLET data that is requested, places it in the requesting phase, and writes that phase back to disk. This continues until the end of the Reload Table (/FFFF) is reached.

Phase 2 substitutes zeros for the SLET data requested by a phase if that phase requested a FORTRAN Compiler and/or Assembler Program SLET entry and the FORTRAN Compiler and/or the Assembler Program were bypassed or never loaded.

Phase 2 displays appropriate error messages, as necessary, using the WRTY0 subroutine in core storage.

FUNCTIONS, INITIAL LOAD ONLY

Phase 2 clears to zero the sectors that will become the SLET table. The SLET entries are filled in as each monitor system program is stored.

Phase 2 checks for missing phases. All phases specified in the PHID control record must be present, except when the FORTRAN Compiler and/or the Assembler Program are bypassed.

Phase 2 keeps a record of the highest sector loaded so that the sector addresses for the Supervisor Control Record Area (SCRA), Core Image Buffer (CIB), Location Equivalence Table (LET), and User Area (UA) on disk may be correctly established.

Phase 2 checks the data obtained from the Load Mode control record and, if the FORTRAN Compiler and/or the Assembler Program are to be bypassed, a gap is forced in SLET and filled with zeros. These zero entries in SLET occupy as much space as the bypassed program(s) would ordinarily require.

FUNCTIONS, RELOAD ONLY

Phase 2 verifies that the file-protection address is not greater than /062F. (Otherwise, phase 2 cannot be temporarily stored on disk.)

Phase 2 updates the SLET entries for each phase as that phase is reloaded.

Phase 2 provides for expansion of the system into the Cushion Area when required. If a phase grows by more than one sector, it is expanded accordingly. All subsequent SLET entries are updated each time an expansion occurs. A check is made to see that the Supervisor Control Record Area (SCRA) is not overlaid.

Phase 2 constructs an in-core Reload Table. At the end of the monitor system program reload, the data that was accumulated as one or more phases were reloaded is first compared with the existing Reload Table on disk. Entries are replaced or added to the Reload Table as necessary. Then the updated Reload Table is processed as described above.

BUFFERS AND I/O AREAS

Card Input

A: an 80-word buffer that contains card images in left-justified 12 bit/word format, as read by the 1442 or 2501 card I/O subroutine.

A2: an 80-word buffer used in conjunction with buffer A for double buffering capability. This buffer is used only by the 2501 card I/O subroutine.

B: a 60-word buffer into which 12 bits/word data is compressed from buffer A when a 1442 is used as the input device, or from buffers A and A2 when a 2501 is used.

Paper Tape Input

B: a 60-word buffer into which binary data from buffer BIGCB is compressed.

C: a 1-word buffer used for the Delete character test when reading binary paper tape records.

BIGCB: a 108-word buffer for binary paper tape records, 108 frames, left-justified.

Disk Input and Output

BUFF1: a 320-word buffer used in disk I/O operations.

FSLET: a 320-word buffer used for reading or writing a SLET sector. During a reload operation that requires expansion into the Cushion Area of the cartridge, a sector of SLET from FSLET is saved by writing it on the first sector of the CIB. The CIB is not cleared afterward.

RTBFR: a buffer that is used to gather the data that will become the Reload Table. Its length is variable, up to 320 words. The word count (never over /0140) may be found in location RTBFR.

SUBPHASES

Subphase 1

Subphase 1 contains the subroutines that are used only during an initial load. If an initial load is being performed, subphase 1 overlays a portion of phase 2 that contains only reload subroutines. Phase 1 determines from the Load Mode control record the type of load being performed, and either overlays the described area or bypasses the subphase 1 portion of the phase 2 deck.

Most of the functions of this subphase involve the checking of the ID number of each phase as it is encountered. An error message (error 12, 14, or 15) is displayed whenever a phase ID is encountered that is out of sequence or was not specified on the PHID control record.

Subphase 2

Subphase 2 contains the procedures for system initialization prior to the loading of the System Library. After the type 81 (end-of-system) record has been read, phase 2 fetches this subphase and overlays the area in core storage bounded by /0682 through /07C1.

Subphase 2 modifies the coding for the processing of those types of records that are no longer expected, so that if one of these records is encountered, an error message is printed.

Subphase 2 looks up DISKZ in SLET and saves the word count and sector address so that DISKZ may be fetched during processing in subphase 3.

Subphase 2 reads DCOM into the buffer FSLET. During an initial load DCOM is cleared to zeros, after which the following entries are initialized:

<u>Location</u>	<u>Value Inserted</u>
#SYSC	version and modification level from PHID control record
#RP67	a positive value if a 1442, Model 6 or 7 is present
#PIOD	the value indicating the principal I/O device, as determined from REQ control records
#PPTR	the value indicating the principal print device, as determined from REQ control records
#CIAD	relative location in sector @IDAD where CIL word count and sector address is maintained
#ANDU	file-protection disk block address
#BNDU	file-protection disk block address
#FPAD	file-protection or Working Storage address

#CIDN	cartridge ID from word 4 of sector @IDAD
#CIBA	sector address of CIB
#SCRA	sector address of Supervisor Control Record Area
#ULET	sector address of LET
#CSHN	number of unused sectors between last system program and Supervisor Control Record Area.

During a reload operation, only the following are updated: #SYSC, #RP67, #PIOD, #PPTR, #CIAD, and #CSHN, as described above.

Subphase 2 reads the Resident Image from disk into the buffer FSLET. During an initial load the Resident Image is initialized as follows:

<u>Location</u>	<u>Value Inserted</u>
\$CH12	address of channel 12 indicator for the principal print device, as determined from REQ control records
\$CORE	core size (may be actual, or set by CORE control record)
\$DREQ	a negative value indicating DISKZ
\$HASH and \$HASH+1	word count and sector address of CIB
\$IREQ	address of \$DUMP entry point
\$ULET	sector address of LET for logical drive 0
\$CILA	address of the end of DISKZ minus 1 (location in DISKZ where the word count and sector address of the Core Image Loader is to be placed)
\$DZIN	a negative value indicating DISKZ
\$FPAD	file-protection sector address for logical drive 0
\$DCYL	table of defective cylinders (from sector @IDAD)
\$IBT2	address of level 2 interrupt branch table

During a reload operation, only the following are updated: \$CH12, \$CORE, \$IREQ, \$CILA, and \$IBT2, as described above.

The first sector of LET is established and initialized with a 1DUMMY entry, Working Storage disk block address, User Area sector address, and the number of unused words in the first sector of LET. All other words in LET are set to zero until the System Library is loaded.

Subphase 2, on an initial load, goes to the Reload Table processing subroutine, and upon return, fetches and transfers control to subphase 3.

Subphase 3

Subphase 3 clears the sign bits from all sector addresses in SLET and resets them according to the data obtained from the System Configuration records. (The sign bits are used to indicate which, if any, I/O devices are not present on the system.)

Subphase 3, on a reload operation, compares all entries in the Reload Table built in core storage with the Reload Table on disk. Phase ID numbers in the Reload Table on disk that match phase ID numbers in the Reload Table in core storage are replaced by those from the table in core storage. Any additional phase ID numbers from the table in core storage that are not present in the table on disk are added to the table on disk. At the conclusion of this update of the Reload Table on disk, it is completely re-processed by the W200 subroutine in phase 2.

Subphase 3 places the word count and sector address of the Core Image Loader, obtained from SLET, into DISKZ in core storage and into DISKZ on cylinder 0. If a reload operation is being performed, subphase 3 displays "END RELOAD" and halts.

On an initial load, subphase 3 branches to the Auxiliary Supervisor with a parameter of minus 5, causing a dummy DUP monitor control record to be placed in the Supervisor buffer and the Monitor Control Record Analyzer to be called via the \$EXIT entry point in the Skeleton Supervisor. The Monitor Control Record Analyzer then calls DUP to load the System Library.

CORE LAYOUT

Figure 1 shows the layout of the contents of core storage during System Loader operation.

①	②	③
1442/2501 Bootstrap Loader	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
	DISKZ	DISKZ
Phase 1	Reload Table	Reload Table
Communications Area	Communications Area	Communications Area
	Phase 2	Phase 2
		Subphase 1, 2, or 3
Card or Paper Tape Subroutine	Card or Paper Tape Subroutine	Card or Paper Tape Subroutine

Figure 1. Core Layout During System Loader Operation

CARTRIDGE IDENTIFICATION SECTOR

On an initial load, the System Loader uses the contents of the cartridge ID sector (sector @IDAD) as follows:

The first three words of the sector (the defective cylinder table for the cartridge, initialized by the DCIP program) are placed into its disk I/O subroutine

prior to performing any disk operations, and into locations \$DCYL through \$DCYL+2 in the Resident Image.

The fourth word of the sector (the cartridge identification word, initialized by the DCIP or DISC program) is placed into location #CIDN in DCOM.

SYSTEM LOCATION EQUIVALENCE TABLE (SLET)

SLET occupies two adjacent sectors on the system cartridge. Its functions are:

- To provide a convenient means for locating each monitor system program that has been stored on the system cartridge.
- To indicate which are the principal I/O devices for the system.
- To indicate which devices, if any, are not present on the system.

When the System Loader initially stores a phase of a monitor system program on the disk, it makes a 4-word entry in SLET for that program consisting of:

1. The phase identification (phase ID) number.
2. The core loading address of the phase. This is the address in which the word count is to be stored prior to fetching the phase from the disk.
3. The word count of the phase, not including the two words occupied by the word count and sector address used in fetching the phase from disk.
4. The sector address of the phase.

During an initial load, the SLET sectors are cleared to zero. The 4-word entries describing each phase are built into the table, phase by phase, as the monitor system programs are loaded. When a program, such as the Assembler Program, is to be bypassed, the words in SLET that it would normally need (the number of phases in the program times 4) are left at zero and are bypassed. The SLET entry for the next program phase that is loaded follows this gap. (A program or phase not included during an initial load cannot later be included in a reload operation; a new initial load must be performed.)

All phase ID numbers in SLET are in ascending order. No duplications exist. The only jumps in sequence are between programs, not between phases within a program.

The five 4-word entries describing the principal I/O devices and the corresponding conversion sub-routines are built by the System Loader and do not come directly from program decks. These special entries, as well as all I/O phases, are located on the second of the two SLET sectors.

The contents of SLET can be obtained at any time by an execution of the DSLET program (see the description of DSLET under Mainline Programs in Section 12. System Library).

RELOAD TABLE

The Reload Table occupies one sector on the system cartridge. It contains a 3-word entry for each phase that requests SLET information. Word 1 of each entry contains the phase ID number of the requesting phase. Word 2 of each entry contains the number of the location, relative to the beginning of the phase, where the SLET entries are to be inserted into the

requesting phase. Word 3 of each entry contains the number of 4-word SLET entries to be inserted into the requesting phase. The phase ID number of the requesting phase itself is in 2's complement form to indicate to the System Loader that SLET data is requested by that phase.

When completed at the end of an initial load or reload operation, the Reload Table consists of a string of 3-word entries, as described above, except that the phase ID numbers have been recomplemented by the System Loader. At the end of the string is /FFFF. It may be at an odd or an even address, depending upon the length of the string. At the end of an initial load, the phase ID numbers are in ascending order. After one or more reload operations the phase ID numbers may or may not be in ascending order.

When a DEFINE VOID ASSEMBLER or DEFINE VOID FORTRAN operation is performed by DUP, all phase ID numbers belonging to the voided program(s) are removed from the Reload Table. The remaining 3-word entries in the Reload Table are packed together and terminated with /FFFF.

FLOWCHARTS

Cold Start Loader: CST01
Cold Start Program: CST02

COLD START LOADER

The Cold Start Loader is the one-card bootstrap used to initiate the operation of the Cold Start Program, which in turn initiates the operation of the monitor system.

Since it is loaded by the IPL procedure, all instructions in the Cold Start Loader are in Load Mode format. Hence, the program must construct all IOCCs as well as any long instructions required by it.

The first word set up after entering the program is the second word of the IOCC for reading the Console Entry switches. After this is done, the number of the physical drive to be assigned as logical drive 0 is obtained from the Console Entry switches. The program then checks to see if the number obtained is valid (0-4, inclusive). If it is not, the program comes to a WAIT from which the user may restart by entering a valid number and pressing the START key. Once a valid number has been obtained, the device code for the drive specified is constructed and saved (for use by the Cold Start Program as well as the Cold Start Loader).

After setting up the second word of the IOCC for sensing the disk (with reset), the program senses the disk. All bits except the not-ready bit (bit 2) are masked out. If the drive is not ready, the program comes to the same WAIT mentioned above.

Four long instructions are built, and the final steps in the setting up of the IOCC for seeking are performed. The word count of the Cold Start Program plus the word count of DISKZ plus 27 is stored in DZ000-29, the 27 being the number of words reserved in sector @IDAD for parameters.

After setting up the second word of the IOCC for reading the disk, the program initiates a seek toward the home position, one cylinder at a time. When the seek is complete, sector zero on the cylinder currently under the read/write heads is read from the disk. If no disk error occurs

during this read and if the sector address is that of sector @IDAD, then a branch is made to \$ZEND, which is the address of the first word of the Cold Start Program.

If the sector address is not that of the Cold Start Program, another seek toward the home position is initiated. This seek-and-read process is repeated until the proper sector address is found. (Therefore, a cartridge with invalid sector addresses causes the program to function improperly.) Any disk error results in the program coming to a WAIT with a /3028 in the storage address register.

COLD START PROGRAM

The Cold Start Program is fetched from the disk by and receives control from the Cold Start Loader.

Using the same device code that was set up by the Cold Start Loader for the physical drive assigned as logical drive 0, the Cold Start Program reads the Resident Image into its normal location in core storage. Once this operation is performed, location zero is initialized with an MDX to \$DUMP+1 and the Auxiliary Supervisor is entered with a parameter of minus 1, causing it to place a dummy JOB monitor control record into the Supervisor buffer and execute a CALL EXIT.

CORE LAYOUT

Figure 2, panel 1, shows the layout of the contents of core storage after the Cold Start Loader has been loaded and, in turn, has fetched sector @IDAD from logical drive 0 into core storage. This sector is read into core such that the DISKZ subroutine resides at DZ000, followed by the Cold Start Program.

Figure 2, panel 2, shows the layout of the contents of core storage after the Cold Start Program has fetched the Resident Image from sector @RIAD from logical drive 0 into core storage. The Resident Image, like DISKZ, is fetched so that all locations occupy their permanent positions in core storage.

①	②
Cold Start Loader	Resident Image
Parameters	
DISKZ	DISKZ
Cold Start Program	Cold Start Program

Figure 2. Core Layout During Cold Start

FLOWCHARTS

CALL LINK, CALL EXIT, CALL DUMP Processor:
SUP01

The Resident Monitor consists of COMMA, the Skeleton Supervisor, and a disk I/O subroutine.

COMMA

For a description of COMMA and its contents, see Section 2. Communications Areas.

SKELETON SUPERVISOR

The Skeleton Supervisor is composed of: the CALL LINK, CALL DUMP, CALL EXIT processor; error traps for interrupt levels 1, 2, 3, and 4, the PROGRAM STOP key, and preoperative errors; and the ILSs for interrupt levels 2 and 4.

CALL LINK, CALL DUMP, CALL EXIT PROCESSOR

This portion of the Skeleton Supervisor indicates the type of entry made to it -- LINK, DUMP, or EXIT -- then fetches and transfers control to phase 1 of the Core Image Loader.

On any entry at \$DUMP locations \$CIBA+1 through 4095 are saved on the Core Image Buffer before they are overlaid by the monitor system programs required for the core dump.

On all entries to the Skeleton Supervisor, the two words following the branch instruction are saved to prevent their being overlaid and lost. They are needed only if the entry was at \$LINK, at which time these two words contain the name of the link to be executed next.

ERROR TRAPS

There are six error traps in the Skeleton Supervisor: the preoperative error trap, the PROGRAM STOP key trap, and one postoperative error trap each for interrupt levels 1, 2, 3, and 4.

Each error trap except for the PROGRAM STOP key trap consists of a link word, a WAIT instruction, and a BSC indirect to the link word. The PROGRAM STOP key trap employs a BOSC indirect instead of a BSC indirect.

ILSs

The Skeleton Supervisor contains the ILSs for interrupt level 2 (ILS02) and interrupt level 4 (ILS04). These are used in all cases by all monitor system programs and by user programs in lieu of user-written ILS02 and/or ILS04 subroutines.

DISK I/O SUBROUTINE

The Resident Monitor contains one of the three disk I/O subroutines at all times. The disk I/O subroutine in the Resident Monitor is the disk I/O subroutine currently in use.

For a description of the DISKZ subroutine, see Section 13. System Device Subroutines. A listing of the DISKZ subroutine is contained in Appendix B. Listings.

For descriptions of the DISK1 and DISKN subroutines, see Section 12. System Library.

FLOWCHARTS

Monitor Control Record Analyzer: SUP02 - SUP05
 Supervisor Control Record Analyzer: SUP06-SUP08
 Auxiliary Supervisor: SUP10
 System Core Dump Program: SUP09

MONITOR CONTROL RECORD ANALYZER -
PHASE 1

The Monitor Control Record Analyzer is the program that decodes monitor control records and takes the specified action.

The Monitor Control Record Analyzer is entered via the \$EXIT entry point in the Skeleton Supervisor. This entry causes the Core Image Loader to fetch the Monitor Control Record Analyzer and transfer control to it.

The Monitor Control Record Analyzer utilizes the system I/O device subroutines. Three of these subroutines (an input, an output, and the appropriate conversion subroutine) are fetched into core storage by the Monitor Control Record Analyzer itself, using SLET information provided by the System Loader.

The Monitor Control Record Analyzer reads monitor control records from the principal input device into the Supervisor buffer, which occupies locations @SBFR through @SBFR+79 and contains a monitor control record in unpacked, right-justified EBCDIC format.

The principal conversion subroutine checks for monitor control records. If the principal conversion subroutine detects a monitor control record during the execution of a monitor system program other than the Monitor Control Record Analyzer, \$CTSW in COMMA is set to a positive non-zero value, the monitor control record is converted to unpacked, right-justified EBCDIC format, and the record is passed to the Monitor Control Record Analyzer in the locations assigned as the Supervisor buffer.

JOB CONTROL RECORD PROCESSING

Upon detecting a JOB control record, the Monitor Control Record Analyzer initializes the SLET information for the principal input device, COMMA, DCOM, and LET.

The entire heading sector (sector @HDNG) is cleared, "PAGEbbb1" is stored in words 0-3, and the contents of columns 51-58 of the JOB control record are stored in words 6-9.

The SLET information for the device other than the Keyboard normally assigned as the principal input device replaces the SLET information for the current principal input device and its conversion subroutine.

The following parameters in COMMA are initialized to zero:

\$COMN,	\$LAST,	\$PST1,
\$CTSW,	\$LINK,	\$PST2,
\$DADR,	\$NDUP,	\$PST3,
\$DUMP, /003F	\$NXEQ,	\$PST4,
\$IBSY,	\$PBSY,	\$SNLT,
\$IOCT,	\$PRET,	\$STOP,
		\$WRD1

In addition, \$PGCT and \$UFIO are set to one, and the address of \$DUMP is stored in \$IREQ.

The cartridge IDs specified on the JOB control record are unpacked to one EBCDIC character per word, converted to binary, and stored in a five-word ID table to be used by the System Update program to update the cartridge-dependent tables in COMMA and DCOM (see System Update, below).

\$CIBA in COMMA is set by comparing the ID of the cartridge on which the CIB for the current job is to be found, as specified on the JOB control record, to the list of specified IDs and selecting the entry in #CIBA that corresponds to the matching ID. Similarly, \$WSDR is set equivalent to the logical drive number of the ID that matches the Working Storage cartridge ID, and \$UFDR is set equivalent

to the drive number that matches the unformatted I/O cartridge ID.

The address of the adjusted end of the User Area (#ANDU) is compared to the address of the base end of the User Area (#BNDU) for each cartridge. If the two items are unequal for any cartridge, the temporary items in LET on that cartridge are deleted. A sector of LET is read and the disk block counts are accumulated until the sum is equivalent to #BNDU. The sector address of that sector and any following LET sectors are stored in a table (JB936) until a continuation of the disk block accumulation equals #ANDU. The next entry consists of a dummy name (IDUMMY) and the size of Working Storage. The size of Working Storage is saved (JB934) and the entry is cleared. The preceding disk block counts are added one at a time to JB934, subtracted from #ANDU, and deleted until #ANDU equals #BNDU. If an entire sector is deleted before #ANDU equals #BNDU, it is necessary to fetch the sector address of the preceding LET sector from the table JB936 and read in that sector so the deletion process may be continued. When all the temporary entries have been deleted (#ANDU equals #BNDU) IDUMMY is stored as the name, and the disk block count of the last entry is stored as the size of Working Storage.

The following parameters in DCOM are set to zero:

#CBSW,	#FRDR,	#MPSW,
#DBCT,	#FRMT,	#NAME,
#DCSW,	#FSZE,	#NCNT,
#ENTY,	#LCNT,	#TODR,
#FCNT,	#MDF1,	#UHOL,
#FHOL,	#MDF2,	#USZE,
		#WSCT

#JBSW is set to zero unless a 'T' appears in column 8 of the JOB control record, in which case #JBSW is set to a positive, non-zero value.

SYSTEM UPDATE PROGRAM

The purpose of the System Update program is to update the drive- and cartridge-dependent tables of the system cartridge DCOM to reflect the logical definition and contents of the cartridge in use during the current job, as specified by the JOB control record.

The specified cartridge IDs are communicated to the System Update program in logical order and in tabular form from core storage or from disk. This table is referred to as ID LIST.

A table of available cartridge IDs in physical order is formed by the System Update program by executing a command to read one word from each drive successively. If the drive is available and ready, the DSW, when immediately tested, shows a busy condition. DISKZ is then called to fetch the cartridge ID and defective cylinder addresses from sector @IDAD and the ID is stored in CART LIST. If a drive is not available or not ready, the appropriate position of CART LIST is set to zero.

CART LIST is searched for each ID in ID LIST. When matching IDs are found, DCOM is fetched from the physical drive represented by the matching ID from CART LIST. The ID being processed is located in the #CIDN parameter in the DCOM just fetched. The entries from the drive- and cartridge-dependent parameters in DCOM in the position where the ID is found are moved to the corresponding drive- and cartridge-dependent parameters in the master DCOM in the position specified by the logical drive assignment for that ID. The entries for unspecified cartridges are set to zero.

At the same time, the device code (\$ACDE) and defective cylinder addresses (\$DCYL) in COMMA are set. \$ACDE is the only link between the logical and physical definition of a drive. The position of an entry in the \$ACDE parameter defines the logical drive, and the device code defines the physical drive. The defective cylinder addresses are those fetched from words 0-2 of sector @IDAD of the cartridge. These addresses and cartridge IDs were set by the DCIP program. \$FPAD and \$ULET in COMMA are updated from the newly updated DCOM on the master cartridge.

OTHER CONTROL RECORD PROCESSING

Upon detecting an XEQ monitor control record, the Monitor Control Record Analyzer tests \$NXEQ, the non-execute switch, in COMMA. If \$NXEQ is zero, the Monitor Control Record Analyzer fetches and transfers control to the XEQ Control Record Processor (see below). Otherwise, the Monitor Control Record Analyzer prints an error message and reads the next control record for processing.

Upon detecting an ASM or FOR monitor control

record the Monitor Control Record Analyzer fetches the first phase of the specified program using SLET information provided by the System Loader, and transfers control to it.

Upon detecting a DUP monitor control record, the Monitor Control Record Analyzer tests \$NDUP, the non-DUP switch, in COMMA. If \$NDUP is zero, the Monitor Control Record Analyzer fetches and transfers control to the first phase of DUP. Otherwise, an error message is printed and the next control record is read for processing.

Upon detecting a PAUS monitor control record, the Monitor Control Record Analyzer comes to a WAIT at the PROGRAM STOP key trap in the Resident Monitor. When the PROGRAM START key is pressed, the Monitor Control Record Analyzer reads and processes the next control record.

Upon detecting a TYP monitor control record, the Monitor Control Record Analyzer replaces the SLET information used to fetch the principal input device subroutine and its associated conversion subroutine with the SLET information for the Keyboard input subroutine and its associated conversion subroutine. These subroutines are then fetched and used for the reading and converting of subsequent input records from the Keyboard.

Upon detecting a TEND monitor control record, the Monitor Control Record Analyzer replaces the SLET information used to fetch the principal input device (the Keyboard) subroutine and its associated conversion subroutine with the SLET information for the device subroutine and conversion subroutine used with the device normally assigned as the principal input device, i. e., not the Keyboard. These subroutines are then fetched and used for the reading and converting of subsequent input records.

Upon detecting a CPRNT monitor control record, the Monitor Control Record Analyzer replaces the SLET information used to fetch the principal print device subroutine with the SLET information for the Console Printer output subroutine. (This replacement is permanent and can be changed only by a new initial load of the system.) This subroutine is then fetched and used for the printing of subsequent output records on the Console Printer.

Upon detecting an EJECT monitor control record, the Monitor Control Record Analyzer ejects the page on the principal print device, prints the current page heading, and reads and processes the next monitor control record.

XEQ CONTROL RECORD PROCESSOR - PHASE 2

The XEQ Control Record Processor processes the XEQ monitor control record and the Supervisor control records - LOCAL, NOCAL, and FILES.

XEQ CONTROL RECORD PROCESSING

The presence of Supervisor control records following the XEQ control record is indicated by the contents of columns 16 and 17. If the number in these columns is not zero or blank, the Supervisor Control Record Analyzer, the subroutine that processes LOCAL, NOCAL, and FILES control records, is called.

#MPSW in DCOM is set non-zero if an 'L' appears in column 14; otherwise, #MPSW is set to zero.

\$DREQ in COMMA is set according to the disk I/O subroutine indicated in column 19. If column 19 is blank, \$DREQ is set to indicate DISKZ.

If a name appears in columns 8 through 12 of the XEQ control record, the name is converted to name code and the Core Image Loader is called via the \$LINK entry point in the Skeleton Supervisor. (The 2-word name of the program to be linked to follows the branch to \$LINK.)

If no name appears in the XEQ control record, phase 0/1 of the Core Load Builder is fetched into core storage and control is transferred to phase 0. The cartridge ID in columns 21-24 is located in #CIDN to determine the drive on which the program is to be found. This drive number is stored in \$DCDE in COMMA.

SUPERVISOR CONTROL RECORD PROCESSING

LOCAL/NOCAL Control Record Processing

The mainline program name(s) and subroutine name(s) found on the LOCAL/NOCAL control records are converted to name code and stored on the Supervisor Control Record Area (SCRA). If a mainline program name is not specified, two words of blanks replace it in the SCRA.

LOCAL control record information is stored in sectors 0 and 1 of the SCRA. NOCAL control record information is stored in sectors 2 and 3.

FILES Control Record Processing

Each file number is converted to binary and stored in the SCRA. Following each file number in the SCRA is the file name in name code. If no name was specified, two words of zeros replace the name. Following each file name in the SCRA is the cartridge ID converted to binary. If no cartridge ID was specified for a file, one word of zeros replaces the cartridge ID. Both the file name and cartridge ID may not be unspecified.

FILES control record information is stored in sectors 4 and 5 of the SCRA.

SUPERVISOR CONTROL RECORD AREA (SCRA)

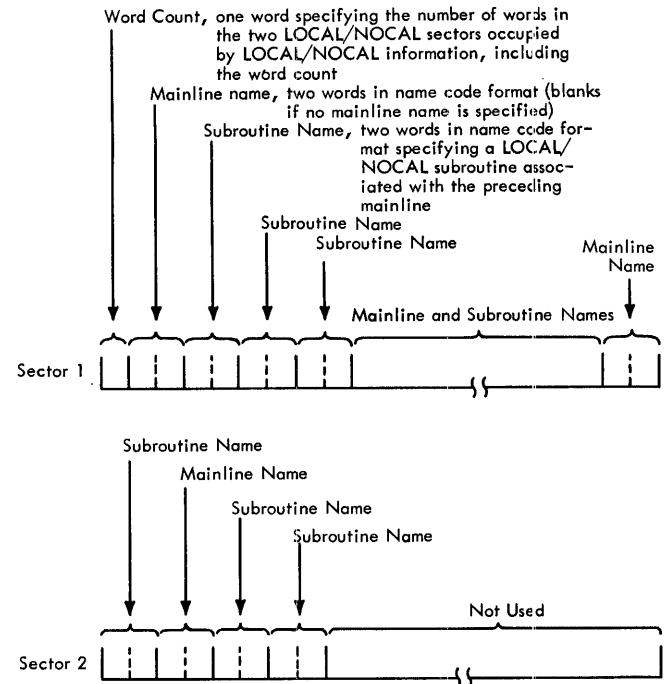
Sectors 0 and 1 of the SCRA are occupied by the LOCAL information for the core load or execution currently in progress (see diagram, below). The first word of sector 0 contains the word count of the information stored in the two LOCAL sectors.

Sectors 2 and 3 of the SCRA are occupied by the NOCAL information for the core load or execution currently in progress (see diagram, below). The first word of sector 2 contains the word count of the information stored in the two NOCAL sectors.

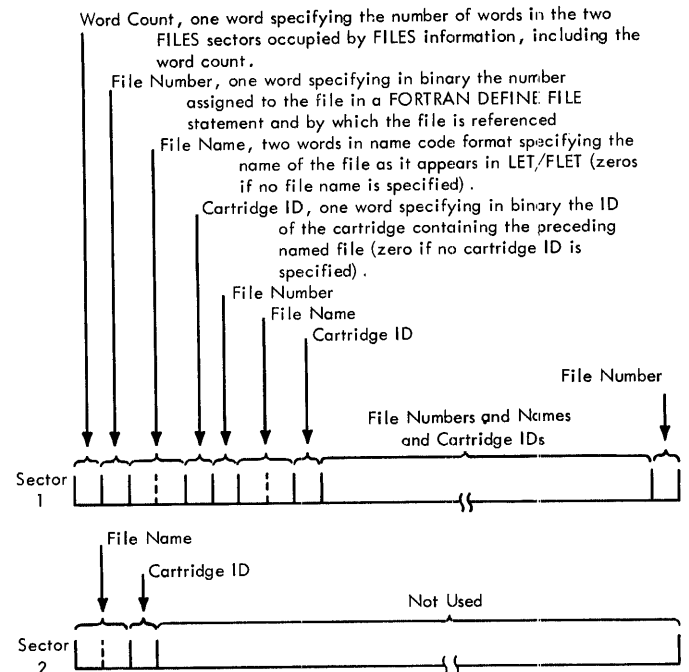
Sectors 4 and 5 of the SCRA are occupied by the FILES information for the core load or execution currently in progress (see diagram, below). The first word of sector 4 contains the word count of the information stored in the two FILES sectors.

Sectors 6 and 7 of the SCRA are not used.

The format of information in the LOCAL/NOCAL sectors is as follows:



The format of information in the FILES sectors is as follows:



1003F

SYSTEM CORE DUMP PROGRAM

If an entry was made to the Skeleton Supervisor at the \$DUMP entry point, the Skeleton Supervisor saves the contents of location \$CIBA+1 through 4095 on the CIB, then fetches and transfers control to the Core Image Loader. If \$DMPF (the dump format indicator) is zero or positive, the Core Image Loader fetches into core storage and transfers control to the System Core Dump program.

The Dump program requires the principal print device subroutine. This subroutine is fetched into core storage by the Dump program itself utilizing SLET information provided by the System Loader.

If dump limits are specified, three checks are made:

1. If both limits are zero, the lower limit is left zero and the upper limit is set to core size.
2. If a limit is larger than core size, the limit is subtracted from the core size and the difference is used as the limit.
3. If the lower limit is greater than the upper limit, a wrap-around dump is given. That is, core storage between the lower limit and the end of core storage is dumped, then core storage between location 0 and the upper limit is dumped.

The lower dump limit is checked to determine which, if any, sections of the CIB must be read into the dump buffer. If any or all of the contents of the CIB are to be dumped, the CIB is read into core storage in sections; sectors 0-3 constituting section 1, sectors 4-7 constituting section 2, and sectors 8-12 constituting section 3. Since the first six words of core storage were not stored to the CIB, the contents of the dump buffer are off-set by six words. These six words are filled in from words 0-5 in the case of section 1 and are saved from the end of the previous section in the cases of sections 2 and 3. Locations greater than 4095 were not stored to the CIB and are dumped from their original locations.

If the \$DUMP entry point contains no return address (i. e., is zero), the Dump program calls the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

If the \$DUMP entry point contains a return address (i. e., is non-zero), the Dump program restores the contents of core storage in three stages. First, the locations between \$CIBA+1 and the beginning of the disk I/O subroutine are restored from the CIB.

Second, the locations between the beginning of the disk I/O subroutine and the beginning of the principal print device subroutine are restored. Third, the locations between the beginning of the principal print device subroutine and location 4095 are restored from the CIB using the disk read subroutine in COMMA. Control is then returned to the restored core load at the location following the dump parameters.

AUXILIARY SUPERVISOR

If an entry was made to the Skeleton Supervisor at the \$DUMP entry point and \$DMPF (the dump format indicator) is negative, the Core Image Loader fetches into core storage and transfers control to the Auxiliary Supervisor.

The Auxiliary Supervisor has three functions:

1. It stores dummy monitor control records to the Supervisor buffer for processing by the Monitor Control Record Analyzer.
2. It prints error messages for errors detected by the Core Image Loader.
3. It aborts a JOB.

The Cold Start Program calls the Auxiliary Supervisor with a parameter of minus one (-1). This parameter causes the Auxiliary Supervisor to place a dummy JOB monitor control record in the Supervisor buffer, convert from binary to EBCDIC the cartridge ID of the cartridge from which the cold start was made and store it in the Supervisor Buffer, set \$CTSW non-zero, and call the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

The ILS04 subroutine calls the Auxiliary Supervisor with a parameter of minus two (-2) if an interrupt occurs from the Keyboard INTERRUPT REQUEST key and the user has not provided a servicing subroutine for that interrupt. This parameter causes the Auxiliary Supervisor to set \$IOCT, \$IBSY and \$PBSY in COMMA to zero, set \$FLSH in COMMA non-zero, and call the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

The Core Image Loader calls the Auxiliary Supervisor with a parameter of minus three (-3) or minus four (-4). If a program name cannot be found in LET/FLET, the parameter used is minus

three. If a name specified as the program or core load for execution is found to be a data file in LET/FLET, the parameter used is minus four. In either case, the Auxiliary Supervisor prints an error message, sets \$NXEQ in COMMA non-zero, and calls the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

The System Loader calls the Auxiliary Supervisor with a parameter of minus five (-5). This parameter causes the Auxiliary Supervisor to place a dummy DUP monitor control record in the Supervisor buffer and call the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

A negative parameter other than minus 1 through minus 5 causes the Auxiliary Supervisor to print an appropriate error message and call the Monitor Control Record Analyzer via the \$EXIT entry point in the Skeleton Supervisor.

CORE LAYOUT

Figure 3, panel 1 shows the layout of the contents of core storage at the time the Skeleton Supervisor is entered at the \$LINK, \$EXIT, or \$DUMP entry point.

Figure 3, panel 2 shows the layout of the contents of core storage after phase 1 of the Core Image Loader has been fetched into core storage by the Skeleton Supervisor. If the Skeleton Supervisor was entered at the \$DUMP entry point, the contents of locations \$CIBA+1 through 4095 are saved on the CIB prior to the fetching of the Core Image Loader.

Figure 3, panel 3 shows the layout of the contents of core storage after phase 1 of the Core Image Loader, using whichever disk I/O subroutine is in the Resident Monitor, has fetched phase 2 of the Core Image Loader into core storage.

Figure 3, panel 4 shows the layout of the contents of core storage after the Core Image Loader, as the result of an entry to the Skeleton Supervisor at the \$EXIT entry point, has fetched the DISKZ disk I/O subroutine into the Resident Monitor for use by the Monitor Control Record Analyzer and has fetched into core storage and transferred control to the Monitor Control Record Analyzer. The principal print device and principal input device subroutines have been fetched into core storage by the Monitor Control Record Analyzer. The areas for disk I/O and control record buffers have been allocated by the Monitor Control Record Analyzer. If a monitor control record had been detected by a monitor system program other than the Monitor Control Record Analyzer, the record would have been passed to the Monitor Control Record Analyzer in that area to be allocated by the Monitor Control Record Analyzer as the Supervisor buffer.

Figure 3, panel 5 shows the layout of the contents of core storage after the Monitor Control Record Analyzer has fetched the XEQ Control Record Processor into core storage.

Figure 3, panel 6 shows the layout of the contents of core storage after the Core Image Loader, as the result of an entry to the Skeleton Supervisor at the \$DUMP entry point with a non-negative parameter, has fetched into core storage and transferred control to the System Core Dump program. The Dump program, using whichever disk I/O subroutine is in the Resident Monitor, has fetched the principal print device subroutine into core storage. The areas for disk I/O and print buffers have been allocated by the Dump program.

Figure 3, panel 7 shows the layout of the contents of core storage after the Core Image Loader, as the result of an entry to the Skeleton Supervisor at the \$DUMP entry point with a negative parameter, has fetched the DISKZ disk I/O subroutine into core storage and has fetched and transferred control to the Auxiliary Supervisor. The Auxiliary Supervisor has fetched the principal print device subroutine into core storage.

①	②	③	④	⑤	⑥	⑦
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
Disk I/O Subroutine	Disk I/O Subroutine	Disk I/O Subroutine	DISKZ	DISKZ	Disk I/O Subroutine	DISKZ
User Core Load or Monitor System Program	Core Image Loader, Phase 1	Core Image Loader, Phase 1	Principal Print Device Subroutine	Principal Print Device Subroutine	Principal Print Device Subroutine	Auxiliary Supervisor
			Principal Input Device Subroutine	Principal Input Device Subroutine	Dump Program	Principal Print Device Subroutine
			MCRA	MCRA		Disk I/O Buffer
			JOB Processor, System Update	XEQ Processor, SCRA		
			DCOM Buffer	Disk I/O Buffer		
			C. R. Buffer	C. R. Buffer		
					Portion of User Core Load Above 4095	

Figure 3. Core Layout During Supervisor Operation

FLOWCHARTS

Phase 1: CIL01

Phase 2: CIL02

PHASE 1

Phase 1 of the Core Image Loader handles the three entries to the Skeleton Supervisor - LINK, DUMP, and EXIT. The Core Image Loader is assigned this task in order to minimize transfer time (via CALL LINK) from one link to another.

Phase 1 of the Core Image Loader is naturally relocatable. It is read into core storage by the Skeleton Supervisor immediately following whichever disk I/O subroutine is currently in the Resident Monitor. (This can be done by the Skeleton Supervisor with minimal core requirement because the word count and sector address of this phase permanently reside at the end of each disk I/O subroutine.)

If the Skeleton Supervisor was entered at the \$DUMP entry point (\$RMSW is positive), phase 1 tests \$DMPF, the dump format code indicator. If \$DMPF is negative, phase 1 fetches and transfers control to the Auxiliary Supervisor. If \$DMPF is not negative, phase 1 fetches and transfers control to the System Core Dump program.

If the Skeleton Supervisor was entered at the \$EXIT entry point (\$RMSW is negative), phase 1 tests \$DZ1N to determine whether DISKZ is in the Resident Monitor. If DISKZ is in the Resident Monitor, phase 1 fetches and transfers control to the Monitor Control Record Analyzer. If DISKZ is not in the Resident Monitor, phase 1 fetches phase 2 of the Core Image Loader. Using phase 2 as a subroutine, phase 1 overlays DISK1 or DISKN with DISKZ. Phase 1 then fetches and transfers control to the Monitor Control Record Analyzer.

If the Skeleton Supervisor was entered at the \$LINK entry point (\$RMSW is zero), phase 1 tests \$COMN in COMMA to determine if COMMON was defined by the core load just terminated. If \$COMN is non-zero, phase 1 saves Low COMMON on the CIB. (Low COMMON is the lowest 320 words that could have been defined as COMMON by the core load just terminated.) Depending on the disk I/O subroutine currently in the Resident Monitor, Low COMMON is defined as follows:

Low COMMON

<u>Disk I/O Subroutine</u>	<u>Decimal</u>	<u>Hexadecimal</u>
DISKZ	896 - 1215	/0380 - /04BF
DISK1	1216 - 1535	/04C0 - /05FF
DISKN	1536 - 1855	/0600 - /073F

The area occupied by Low COMMON is used by phase 1 as a disk I/O buffer during the LET/FLET search and/or as the area into which phase 2 is fetched when phase 2 is to be used to fetch DISKZ.

Once Low COMMON has been saved, or if no COMMON was defined by the core load just terminated, phase 1 searches LET/FLET for the name of the program or core load to be executed next. The name of the link has been saved in \$LKNM by the Skeleton Supervisor.

If the link is in disk system format (DSF), phase 1 saves any COMMON defined below 4096 on the CIB. It then fetches phase 2, uses phase 2 as a subroutine to overlay DISK1 or DISKN with DISKZ, fetches phase 0/1 of the Core Load Builder, and transfers control to phase 1 of the Core Load Builder.

If the link is in disk core image format (DCI), phase 1 fetches and transfers control to phase 2 to fetch the link and the required disk I/O subroutine, if necessary, and to transfer control to that link.

Special Techniques. Phase 1 of the Core Image Loader places a disk call subroutine in COMMA at \$HASH+8 through \$HASH+19. Using this disk call subroutine, phase 1 is able to overlay itself when fetching phase 2, the Monitor Control Record Analyzer, etc.

PHASE 2

Phase 2 of the Core Image Loader is naturally relocatable. It is read into core storage (by phase 1) immediately following the end of phase 1 if it is to be used by phase 1 to fetch DISKZ, or (by either phase 1 or the Core Load Builder) following the end of the disk I/O subroutine currently in the Resident Monitor if it is to fetch and transfer control to a core load. Phase 2 provides two functions: (1) to overlay the disk I/O subroutine currently in the Resident Monitor with the requested disk I/O sub-

routine and (2) to fetch a core load. It may be used for either or both of these functions on any single entry.

If called to fetch and transfer control to a core load, phase 2 first fetches and processes the core image header and then the disk I/O subroutine required by the core load to be fetched if it is not currently present in the Resident Monitor. The last thing it does is to fetch and transfer control to the core load itself.

Special Techniques. Upon entry, phase 2 requires the disk call subroutine placed at \$HASH+8 through \$HASH+19 by phase 1. Using this disk call subroutine, phase 2 is able to overlay itself when fetching a core load. Included in this subroutine, at \$HASH+13 through \$HASH+19, is the coding that moves the transfer vector from its position at the end of the core load (as it exists on disk) to the end of COMMON in core storage. Once the transfer vector has been moved, control passes to the core load being fetched.

CORE LAYOUT

Figure 4 shows the layout of the contents of core storage following an entry to the Skeleton Supervisor at \$EXIT when DISKZ is present in the Resident Monitor. In panel 1, phase 1 of the Core Image Loader has been fetched by the Skeleton Supervisor. In panel 2, the Monitor Control Record Analyzer has been fetched by phase 1.

Figure 5 shows the layout of the contents of core storage following an entry to the Skeleton Supervisor at \$EXIT when DISKZ is not present in the Resident Monitor. In panel 1, phase 1 of the Core Image Loader has been fetched by the Skeleton Supervisor. In panel 2, phase 2 of the Core Image Loader has been fetched by phase 1. In panel 3, DISKZ has been fetched by phase 2. In panel 4, the Monitor Control Record Analyzer has been fetched by phase 1.

Figure 6 shows the layout of the contents of core storage following an entry to the Skeleton Supervisor

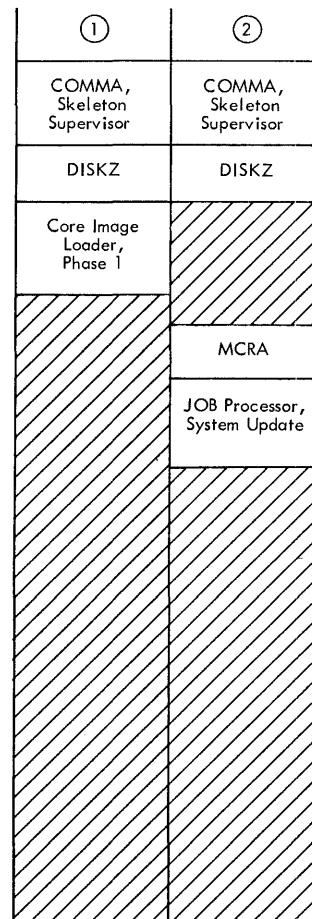


Figure 4. Core Layout on Supervisor Entry at \$EXIT (DISKZ in Core)

at \$DUMP. In panel 1, locations \$CIBA+1 through 4095 have been saved on the CIB and phase 1 of the Core Image Loader has been fetched by the Skeleton Supervisor. In panel 2, the System Core Dump program has been fetched by phase 1 as the result of a non-negative parameter following the branch to \$DUMP. In panel 3, the Auxiliary Supervisor has been fetched by phase 1 as the result of a negative parameter following the branch to \$DUMP.

①	②	③	④
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISK1 or DISKN	DISK1 or DISKN	DISKZ	DISKZ
Core Image Loader, Phase 1	Core Image Loader, Phase 1	Core Image Loader, Phase 1	Core Load Builder, Phase 0/1
LET/FLET Buffer	LET/FLET Buffer	Core Image Loader, Phase 2	
Low COMMON	Low COMMON		
	COMMON Below 4096, Saved		
	COMMON Above 4096, Not Saved		

Figure 7. Core Layout on Supervisor at \$LINK (Link in Disk System Format)

has been saved by phase 1, and the LET/FLET search buffer has been allocated. In panel 2, phase 2 of the Core Image Loader has been fetched by phase 1; the core image header buffer has been allocated by phase 2. In panel 3, the disk I/O subroutine required by the program being linked to has been fetched into the Resident Monitor. In panel 4, the program being linked to has been fetched by phase 2. In panel 5, COMMON defined by the previous core load below location 4096, previously saved on the CIB by phase 1 of the Core Image Loader, as well as the program being linked to, has been fetched by phase 2. In panel 6, the portion of the program being linked to that is contained in the CIB (the portion below location 4096, placed in the CIB by the Core Load Builder) has been fetched by phase 2.

DEBUGGING/ANALYSIS AIDS

To facilitate the finding of errors in and associated with the Core Image Loader, NOP instructions have been placed at critical locations in the Core Image Loader; they are: CM000+1, CM118-5, CM180, LD000+1, GETCL, and LD100+8. These NOPs can be replaced by WAIT instructions so that core dumps can be taken at various stages during Core Image Loader execution. An analysis of the core dump(s) may provide enough information to locate the problem.

Bear in mind that the Core Image Loader is naturally relocatable. Thus, all modifications made to it must be executable irrespective of core location.

①	②	③	④	⑤	⑥
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISKZ	DISK1 or DISKN	Required Disk I/O Subroutine	Required Disk I/O Subroutine	Required Disk I/O Subroutine	Required Disk I/O Subroutine
Core Image Loader, Phase 1	Core Image Loader, Phase 2	Core Image Loader, Phase 2	Link Core Load	Link Core Load	That Portion of Core Load Loaded From CIB
LET/FLET Buffer					
Low COMMON				COMMON Below 4096, Restored	
				COMMON Above 4095, Not Saved But Not Overlaid	That Portion of Core Load Above 4095, Placed In Core By Core Load Builder

Figure 8. Core Layout on Supervisor Entry at \$LINK (Link in Core Image Format)

FLOWCHARTS

Phase 1 (IN): CLB01

Phase 2 (MC): CLB02

The Core Load Builder builds a specified mainline program into an executable core load. The mainline program, with its required subroutines (LOCALs and SOCALs included), is converted from disk system format (DSF) to a format suitable for execution. During the conversion, the Core Load Builder also builds the core image header record and the transfer vector. The resulting core load is suitable for immediate execution or for storing on the disk in disk core image format (DCI) for future execution.

GENERAL COMMENTS

Each phase of the Core Load Builder has been broken up into a series of relatively small, self-contained subroutines. After initialization (phase 1) control remains in the Master Control subroutine, which is a part of phase 2. (The labels in this subroutine all start with "MC".) In other words, the basic control logic is found in the Master Control subroutine.

The labels assigned to constants and work areas within subroutines are in the range 900-999. Whenever noted, even-numbered labels are on even boundaries, and odd-numbered labels are on odd boundaries. Constants and work areas in RCOM (phase 0) are mnemonic and are arranged in four groups, each ordered alphabetically. Double-word cells are in one group, indexed cells are in a second; constants are in a third; and switches and work areas are in a fourth. The labels of switches are of the form "LSWx", where "x" is a number. The labels of constants are of the form "Kx", where "x" is either the number, in decimal, defined in the constant or the four hexadecimal digits defined in the constant.

Patch areas are usually found at the end of a phase. Each one is defined by a BSS followed by a DC.

OVERLAY SCHEME AND CORE LAYOUT

The overlays (phases) of the Core Load Builder have been organized to allow maximum core storage for the Load Table while minimizing the flip-flopping of phases. "Minimizing" here means that, during a one-pass building process (no LOCALs or SOCALs), the phases are executed serially from 1 through 6 (excluding 5). During a two-pass building process (LOCALs and/or SOCALs required), there is some flip-flopping of phases 3 and 5.

Phase 0 is never overlaid. It contains the subroutines that must never be overlaid, as well as work areas and constants required by more than one subroutine.

Phase 1 is fetched along with phase 0. The only difference is that phase 2 overlays phase 1 but not, of course, phase 0. Phases 3, 4, 5, 6, and 12 overlay the last part of phase 2.

Phases 7-10 contain messages. They all require that the principal print subroutine be in the data buffer; these phases themselves are executed from the LET/FLET search buffer.

Phase 11 prints the file map and phase 12 the core map. Both of these phases require that the principal print device subroutine be in the LET/FLET search buffer. Phase 11 is executed from the data buffer.

Figure 9, panel 1 shows the layout of the contents of core storage after phases 0 and 1 of the Core Load Builder have been fetched into core storage by phase 1 of the Core Image Loader or the STORE function of DUP.

Figure 9, panel 2 shows the layout of the contents of core storage after phase 1 has fetched phase 2, overlaying itself. Phase 2 has allocated the areas for the Load Table and the disk I/O buffers.

Figure 9, panel 3 shows the layout of the contents of core storage after any one of the overlay phases has been fetched by phase 2.

Phase 1 includes the subroutines called by the initialization subroutine. In this way, phase 2 can overlay phase 1 completely. Phase 2 includes the subroutines called by the relocation subroutine, RL. The order of the subroutines in this phase is important. Those that are required only during the relocation of the mainline (MV, ML, CK, DC, DF, and FM) come

①	②	③
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISKZ	DISKZ	DISKZ
Phase 0	Phase 0	Phase 0
Phase 1	Phase 2	Phase 2
		Phases 3,4,5,6,12
	Disk I/O Buffers	Phases 7,8,9,10,11
	Load Table	Load Table
	That Portion of Core Load Above 40%	That Portion of Core Load Above 40%

Figure 9. Core Layout During Core Load Builder Operation

last so that they may be overlaid by phase 3. Phase 3 includes the subroutines required to choose a subroutine (as opposed to a mainline) from the Load Table and relocate it. Phases 4 and 6 round out the one-pass core load building process. Phase 4 determines whether or not SOCALs are required, and, if so, whether or not they can be employed to make the core load fit into core storage. It also processes ILSs. Phase 6 performs the miscellaneous jobs, such as creating the transfer vector, that can be done only at the end of the process of building a core load. Phase 5 is executed only during pass 2 in a two-pass building process. It organizes the LOCALs and SOCALs for relocation, including their special linkages.

DISK BUFFERS

There are three buffers used by the Core Load Builder. Each is 320 words long, not counting the word count and sector address, and each has a primary use, although it may be used temporarily for something else. For example, the LET search buffer is used primarily to hold a sector of LET/FLET when searching that table. However, it contains one of the message phases (phases 7-10) whenever a message is printed.

The data buffer is a buffer for the User Area. The program currently being incorporated into the core load is read into this buffer, one sector at a time. For example, after a sector of the mainline is read into this buffer from the User Area or Working Storage, the relocation of the mainline can begin. When this sector of the mainline has been relocated, another sector (if any) is fetched, and so on until the entire mainline is relocated.

The main use of the CIB buffer is to contain the CIB, one sector at a time. For example, if a core load is to occupy locations 1000 - 1639, then the first sector of the CIB contains the part of the core load that is to occupy 1000 - 1319 and the second sector 1320 - 1639. As the core load is built, the Location Assignment Counter (LAC) reflects the ultimate core address of the data word currently being relocated. In this example, the LAC would start at 1000, thus causing sector 1 of the CIB to be read into the CIB buffer. This first word of the core load would be placed in the first word of the CIB buffer and the LAC advanced by 1. Assuming no data breaks, the LAC will eventually be incremented to 1320. Then the contents of the CIB buffer will be written out on sector 1 of the CIB, and sector 2 will replace sector 1 in the CIB buffer. In short, each word of a core load is always transferred to the CIB via the CIB buffer.

The data and CIB buffers are combined into a single 640-word buffer for the purpose of fetching the LOCAL, NOCAL, and FILES information from the SCRA.

CORE IMAGE BUFFER (CIB)

The Core Image Buffer is used by the Core Load Builder, the Core Image Loader, and the Skeleton Supervisor. The Core Load Builder uses it to store any part of the core load that is to reside (when the

core load is executed) below location 4096. The first word of the mainline is stored in the first word of the CIB, and subsequent words follow similarly. Thus, the mainline must be relocated first, and a subsequent ORG that would set the Location Assignment Counter below its first value is not allowed.

LOAD TABLE

The Load Table is used by the Core Load Builder primarily to tell what subprograms to include in the core load as well as at what time during the process of core load building to include a given subprogram. Other uses of the table are discussed below. It exists only during the building of a core load.

There is an entry in the Load Table for (1) every LOCAL/NOCAL entry point specified for a given mainline and (2) for each subprogram entry point referenced in a core load, via CALL or LIBF. For example, even though subprogram A is called five times, there is only one entry in the Load Table for A. On the other hand, if A and B are different entry points to the same subprogram and both are referenced, then there will be an entry for A and another for B.

Each of the Load Table entries is four words in length. The first entry occupies locations 4086 - 4089, the second 4082 - 4085, etc. The first two words of each entry contain the name (in name code) of the subprogram that caused the entry to be made. Bit zero of the first word is set if the entry is that of a LOCAL, bit one is set if the entry is that of a CALL. Mainlines and interrupt level subroutines never appear in the Load Table.

Words three and four of each entry are zero when the entry is first made. As the relocation of a given subprogram begins, word three is set with the entry point, i.e., the absolute, address. In this way, the Core Load Builder can tell by looking at its Load Table entry whether or not a subprogram has been relocated and where it has been relocated to.

Word four is put to several uses, most of which involve LOCAL processing. The use of this word at a given time is dependent upon the pass (1 or 2) and/or whether the subprogram associated with the Load Table entry is a LOCAL that was specified in the LOCAL information in the SCRA. For example, A and B are entry points to the same subprogram, and A (but not B) appears in the LOCAL information in the SCRA. Both A and B can be called in the core load; in such a case A is said to be specified and B unspecified.

The values stored in word four at various times are: (1) the class code (for a non-LOCAL), (2) the address of the Flipper Table entry, (2) /0008 for every subroutine called by a LOCAL, (4) the word count of a LOCAL, (5) the address of the Load Table entry for the specified entry point for the LOCAL currently being relocated, and (6) the address of the LIBF TV that corresponds to the entry in the Load Table.

LOCAL, NOCAL, AND FILES INFORMATION

LOCAL, NOCAL, and FILES information is obtained from the Supervisor Control Record Area (SCRA). This information is supplied by the Supervisor Control Record Analyzer (see Section 6: Supervisor) or the STORECI function of DUP (see Section 9: Disk Utility Program). For the format of LOCAL, NOCAL, and FILES information in the SCRA, see Section 6: Supervisor or Section 9. Disk Utility Program.

ISS TABLE

The ISS Table is used by the Core Load Builder as it constructs Interrupt Branch Tables for ILSs. When the address to which an ISS is to be relocated becomes available, that address is stored in the appropriate entry in the ISS Table. For example, if an ISS for the 1132 Printer (ISS number 6) is being relocated to location /1000, then /1000 is stored in the fourth word of the ISS. This address will later be used during the construction of the Interrupt Branch Table (IBT) for ILS01.

INTERRUPT BRANCH TABLE (IBT)

Each ILS in the System Library that is filled in by the Core Load Builder (ILS00, ILS01, and ILS03) begins with an Interrupt Branch Table (IBT). ILS02 and ILS04, which are a part of the Resident Monitor, require special treatment to construct their IBTs. A given IBT consists of the addresses of the interrupt service entry points for the devices on the corresponding interrupt level. Thus, if there are two devices on a level, there are two entries in the IBT; and each entry consists of one word, that word being the address of the interrupt service entry point to the ISS for the device represented by the IBT entry.

For user-written core loads (as opposed to monitor system programs), the IBT for ILS04 is

constructed by the Core Load Builder and stored in the core image header record. When the Core Image Loader fetches a core load (including the core image header), the address of the IBT for level 4 is stored in location \$IBT4, from which it is accessed by ILS04. The IBT for ILS02 is a single word, \$IBT2, which contains the address of DZ000+4, regardless of the disk I/O subroutine present in the Resident Monitor. This address is stored in \$IBT2 by the Core Image Loader as it fetches the requested disk I/O subroutine. The IBTs for the remaining ILSs are constructed and stored in the ILSs themselves by the Core Load Builder.

After all subprograms have been relocated, the Core Load Builder constructs the IBTs. The IBTs for all ILSs except ILS02 are constructed as described below. Bear in mind that these ILSs are written with special constants stored in each IBT entry. These constants, which will be overlaid by the Core Load Builder, are as follows: The first eight bits of each constant represent the increment to be added to the loading address of the corresponding ISS to get the address of the interrupt service entry point; for IBM-supplied ISSs, this value is four, except for the "operation complete" entry point for the 1442 subroutines, for which the value is seven. The second eight bits are @ISTV plus the ISS number; thus, each IBM entry has an identifier to relate it to a specific ISS.

The Core Load Builder fetches one word at a time from the IBT, i. e., one of the special constants occupying the IBT locations in the ILS. The second eight bits of the word fetched are used to compute the address of the ISS Table entry for the ISS number indicated. If the ISS Table entry is non-zero, it contains the loading address of the ISS itself, which is then incremented by the value stored in the first eight bits of the word fetched. The resulting address, which is the address of the interrupt service entry point, replaces the special constant that supplied the two eight-bit values. If the ISS Table entry is zero, the special constant is replaced with the address of \$STOP, the PROGRAM STOP key trap in the Skeleton Supervisor. This process of replacing special constants continues until a zero is fetched from the IBT, indicating that the entire IBT has been processed. Except for ILS04, this zero is the entry point to the ILS itself.

INCORPORATING PROGRAMS INTO THE CORE LOAD

PASS 1

The mainline is relocated first. Any calls found during this relocation are put in the Load Table. After the mainline has been converted, each subprogram represented in the Load Table is relocated, generally in the order found in the table itself. An entry is flagged as having been relocated by storing the address of the entry point in the third word of the entry. Before an entry is relocated, the names of all the entry points to the entry being considered for relocation are compared with the name of each entry preceding it in the Load Table. A match indicates that the current entry is simply another entry point to a previously relocated subprogram. Thus, the current entry is not relocated; instead, the absolute address of the entry point is determined and stored in the third word of the entry to signify that it has already been relocated.

Furthermore, the names of all the entry points to the entry being considered for relocation are compared with the name in each entry in the Load Table following it. If a match occurs, the third words of both entries are filled in with the absolute address of the entry point. Thus, the Load Table is scanned forward and backward for other entry points to the subprogram currently being considered for relocation.

PASS 2

The Load Table is scanned during pass 2 in the same way as during pass 1. The only difference is that subprograms are relocated in a certain order during pass 2, thus necessitating multiple passes through the Load Table; in fact, one pass is required for each class of subprograms. Thus, all the in-cores (class 0) are relocated first, followed by LOCALs, subprograms in SOCAL 1 (class 1), subroutines in SOCAL 2 (class 2), and subroutines in SOCAL 3 (class 3), in that order.

LOCALs AND SOCALs

If during the first pass the Core Load Builder (phase 4) determines that an Assembler core load will not fit into core storage even with any LOCALs that have been specified, the core load building process is terminated. However, for a FORTRAN core load special overlays (SOCALs) of parts of the core load will be created during a second pass if this will make the core load fit. The decision of whether to proceed with a second pass is made after phase 4 accounts for the sizes of the LOCAL area, if any, the flipper and its table, and each of the SOCALs. If the check shows that SOCAL option 1 (SOCAL 1 and SOCAL 2) will be insufficient, then a further check is made for option 2 (all three SOCALs). If option 2 is still insufficient, processing is terminated; otherwise, a second pass is made.

During pass 2, the entire core load is built again, but, unlike during pass 1, subprograms are relocated in a special order. First, the mainline and the in-core (class 0) subprograms are relocated, followed by: the flipper; the LOCALs, if any; the arithmetic and function (class 1) subprograms; the non-disk FIO (class 2) subroutines; and, if necessary, the disk FIO (class 3) subroutines.

The same procedure described above is necessary if LOCALs are employed without SOCALs. In other words, LOCALs, as well as SOCALs, require two passes.

INTERRUPT LEVEL SUBROUTINES (ILSs)

After all other subprograms have been relocated, the Interrupt Transfer Vector (ITV) is scanned. Except for the entries for interrupt levels 2 and 4, a non-zero entry causes the corresponding ILS to be incorporated into the core load. (ILS02 and ILS04, unless supplied by the user, are a part of the Resident Monitor.) See Interrupt Branch Table, above, for a description of the processing of that part of an ILS.

TRANSFER VECTOR (TV)

The transfer vector consists of two parts: the LIBF TV and the CALL TV. The former provides the linkage to LIBF subprograms, the latter to CALL subprograms. The LIBF TV was created to

enable the LIBF statement to require only one storage location during execution. This is desirable because 1130 FORTRAN object code contains a very high percentage of calls to subprograms. Long branches to those subprograms would greatly increase core requirements for core loads over a method that employs short branches. By replacing the LIBF statement with a short BSI, tag 3, to a transfer vector entry, which could then supply the long branch to the desired subprogram, this problem is solved. The cost, of course, is that XR3 is taken away from the user and the transfer vector is limited to 255 words. This means the LIBF TV has a maximum of 85 3-word entries, two of which become the floating accumulator (FAC) and an indicator for certain arithmetic subroutines. Thus, the user is limited to LIBFs to not more than 83 separate subprogram entry points per core load.

There is no theoretical limit on the number of CALL entry points per core load, for the CALL statement is replaced by an indirect BSI to the desired subprograms. However, the number of CALL and LIBF references combined must not exceed the capacity of the Load Table, which is approximately 150 entries.

The CALL TV entry is one word only, the address of the subprogram entry point. This makes it possible to replace a CALL statement with an indirect BSI to the corresponding CALL TV entry, even though the address of the subprogram itself may not be known at the time the CALL is processed.

When stored on disk in disk core image format (DCI), the LIBF TV follows the last word of the last subprogram in the core load. It may leave one word vacant in order to make the floating accumulator (FAC) begin on an odd boundary. The CALL TV immediately follows the indicator entry in the LIBF TV. During execution the TV extends downward in core storage from the lowest-addressed word in COMMON.

Whereas the CALL TV entry consists of only one word (the address of the subprogram), the LIBF TV entry consists of three words. The first is a link word (initially zero), and the second and third are a long BSC to the subprogram entry point.

Figure 10 shows the layout of the transfer vector in core storage.

Linkage to LOCALs

The LOCAL/SOCAL flipper (FLIPR) is included in a core load if that core load requires LOCALs and/or SOCALs. The flipper transfers control to a LOCAL,

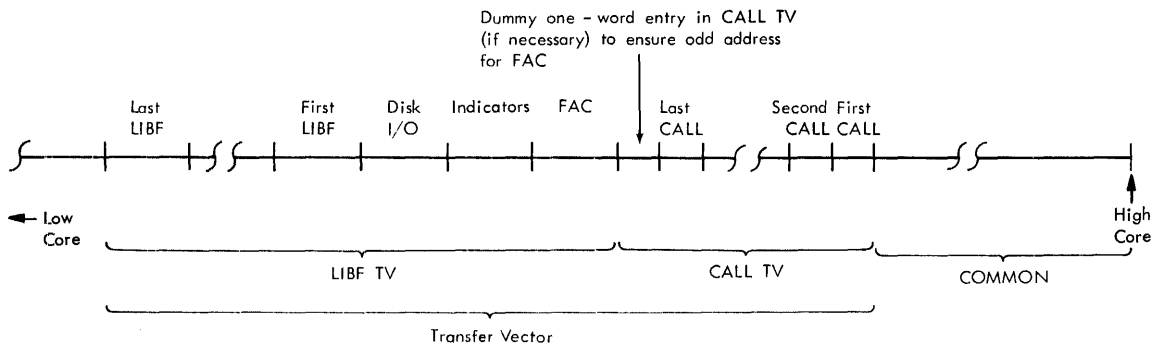


Figure 10. Layout of the Transfer Vector

fetching it first, if necessary. It does likewise for a SOCAL, except that it is never entered if a subprogram is called that is a part of the SOCAL currently in the SOCAL area (see Linkage to the System Overlays).

The Flipper Table immediately precedes the flipper. It consists of a 6-word entry for each entry point specified in the LOCAL information in the SCRA (for a given mainline) that is referenced by a CALL and a 5-word entry for each entry point referenced by a LIBF. If a subprogram has more than one entry point but only one is specified in the LOCAL information (a specified LOCAL), there is a Flipper Table entry for each entry point referenced in the core load.

The format of a 5-word (LIBF) entry in the Flipper Table is as follows:

<u>Word</u>	<u>Description</u>
1-2	BSI L FL000
3	Word count of the subprogram
4	Sector address of the subprogram
5	Entry point address in the subprogram

The format of a 6-word (CALL) entry in the Flipper Table is as follows:

<u>Word</u>	<u>Description</u>
1	Link word
2-3	BSI L FL010
4	Word count of the subprogram
5	Sector address of the subprogram
6	Entry point address in the subprogram

Linkage to the System Overlays (SOCALs)

In order to assure very fast transfer to a subprogram that is a part of a SOCAL that is in core storage at a given time, special transfer vector entries are made for SOCAL subprograms. They are different from the standard LIBF and CALL linkages, and they are different from the linkage to a LOCAL. The SOCAL transfer time is approximately 20 microseconds, compared to 150-180 microseconds to a LOCAL. (Both timings assume a 3.6 microsecond storage cycle.)

Figure 11 shows an entry in the LIBF TV for an in-core subprogram (entry 2) and the special linkage in the LIBF TV for SOCAL subprograms (entries 3-8). Entry 1 is the LIBF TV entry for a SOCAL subprogram. The "disp" is a displacement to the second word of the linkage for the SOCAL in which the subprogram is found.

The example represented in Figure 11 is one that requires SOCAL option 2; TV entries 5 and 8 would not appear if option 1 were used. Entry 1 is the last entry in the LIBF TV, i.e., the highest-addressed word of the transfer vector. Suppose that (1) a LIBF to FADD were made and (2) SOCAL 1 were not in core. The LIBF would be a BSI to the first word of entry 1, which would then BSI to the second word of entry 3. Entry 3 would MDX to the first word of entry 6, which would transfer control to the LOCAL/SOCAL flipper subroutine (FLIPR) at FL230, the entry point in FLIPR for fetching the arithmetic subprograms. The flipper would fetch SOCAL 1, change the third word of entry 3 to MDX to *-3, and BSC to the first word of entry 3, which then transfers control to FADD. The flipper would also ensure that the third words of entries 4 and 5 were both MDX to *-12.

LOW
CORE

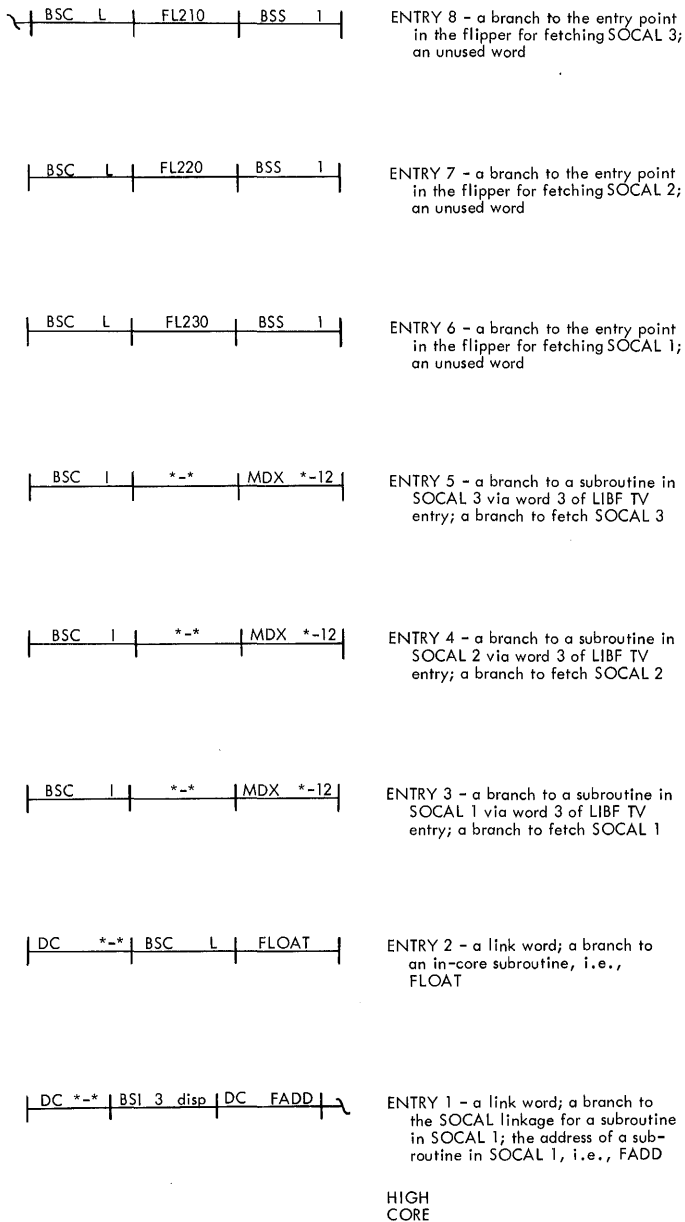


Figure 11. SOCAL Linkage in the LIBF Transfer Vector

Suppose now that FADD were called again before some subprogram in either SOCAL 2 or SOCAL 3 were called. This time the LIBF would cause a BSI to the first word of entry 1 and then to the second word of entry 3. The MDX would then be to

the first word of entry 3, followed by a transfer of control to FADD. The transfer has required only 2 short BSIs, a short MDX, and an indirect BSC.

The linkage for a CALL to a function is somewhat different from that just described. Suppose that (1) SOCAL option 2 was used and (2) each SOCAL consists of two subprograms, FABS and FSQR being the functions in SOCAL 1.

Figure 12 shows SOCAL 1, SOCAL 2, and SOCAL 3 as they are stored on the disk. The first 2 words of each of these SOCALs are the CALL TV for the subprograms in that SOCAL.

A CALL to FSQR, for example, would be an indirect BSI to the second word of whichever SOCAL happened to be in the SOCAL area. If this were SOCAL 1, control would be immediately transferred to FSQR. Otherwise, control would first be given to the LOCAL/SOCAL flipper at FL200, the entry point in FLIPR for fetching the function subprograms. The flipper would fetch SOCAL 1 and re-execute the original CALL to FSQR.

DEFINE FILE TABLE

The processing of the DEFINE FILE Table normally consists of filling in word 5 (the sector address) for each entry in the DEFINE FILE Table preceding the mainline program.

However, additional processing is required when a file must be truncated, i.e., the space available on the disk is insufficient to store the number of records defined in the file. If the file is in the User/Fixed Area, or if it is the only file in a particular Working Storage, then the Core Load Builder attempts to truncate it enough to fit.

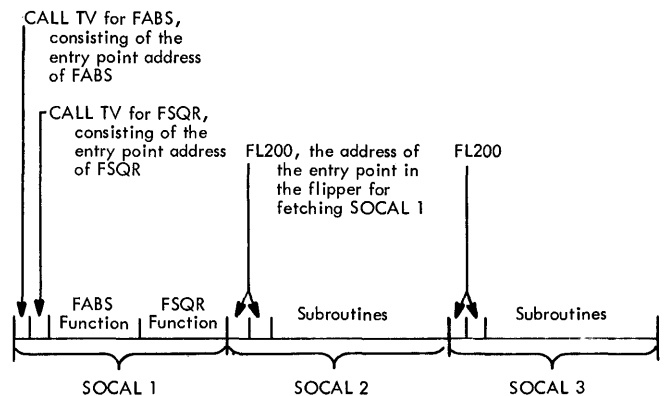


Figure 12. CALL Transfer Vector for SOCALs

First, the entire DEFINE FILE Table is fetched and stored in the unoccupied area allocated to the Load Table. If the Core Load Builder determines that a file can be truncated, the number of records and the disk block count in the appropriate DEFINE FILE Table entry are modified accordingly. As each entry is completed, all seven words are relocated in the same manner as the other words of the core load.

The processing consists of comparing the file number of a DEFINE FILE Table entry with each of the file numbers in the FILES information in the SCRA, if any. If a match occurs, the name of the disk area associated with the file number obtained from the FILES information is found in LET/FLET, and the sector address of that disk area is placed in word 5 of the DEFINE FILE Table entry. If none of the file numbers from the FILES information match the file number in the DEFINE FILE Table, the file is set up in Working Storage. In either case, the system cartridge is assumed unless a cartridge ID has been specified in the FILES information.

The format of a DEFINE FILE Table entry is as follows:

<u>Word</u>	<u>Description</u>	<u>Symbolic Reference</u>
1	File number	@FLNR
2	Number of records in the file	@RCCT
3	Number of words per record	@WDRC
4	Address of the associated variable	@ASOC
5	Sector address of the file; initially zero, filled in by the Core Load Builder	@SCAD
6	Number of records per sector	@RCSC
7	Disk block count of the file	@BCNT

PHASE DESCRIPTIONS

PHASE 0

Phase 0 always remains in core. It consists of two main sections, (1) the basic subroutines required by all other phases and (2) the constants and work areas shared by two or more subroutines. The latter section is known as RCOM.

The following is a list of the subroutines that comprise phase 0 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
LS000	Search LET/FLET
RH000	Fetch the mainline program header record
NW000	Fetch the next word in sequence from the data buffer, reading a new sector when necessary
CN000	Test a subroutine name for disk I/O
BT000	Add an entry to the Load Table
LK000	Fetch a phase
EX000	Exit to DUP, Supervisor, or Core Image Loader
TL000	Print a message and transfer to EX000
PM000	Fetch one of the message phases and transfer control to it
GP000	Read from or write to the disk

PHASE 1

Phase 1 performs the initialization functions that must be done prior to the relocation of the mainline. Initialization consists of, principally, fetching DCOM and extracting the parameters that are needed by the Core Load Builder, and fetching the mainline header record and saving the information therein.

In addition, phase 1 makes an entry in the Load Table for each LOCAL and NOCAL specified in the LOCAL and NOCAL information in the SCRA, if any.

The following is a list of the subroutines that comprise phase 1 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
IN000	Initialize the Core Load Builder, process the mainline header
LN000	Enter LOCAL and/or NOCAL names in the Load Table

PHASE 2

After the execution of phase 1, part of phase 2 remains in core until the core load is completed. Phase 2 contains the Master Control subroutine, the relocation subroutine, and the transfer subroutine, among others (see below). Master Control supplies the basic logic for the Core Load Builder. The relocation subroutine supplies the logic for relocating a program, i. e., incorporating it into the core load.

The transfer subroutine provides the logic for transferring a relocated word of the core load to the CIB, the CIB buffer, or Working Storage, whichever is appropriate. These three subroutines are basic to the process of building a core load.

The following is a list of the subroutines that comprise phase 2 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
MC000	Master control for the Core Load Builder
RL000	Relocate a program; convert it from disk system format to disk core image format
TS000	Process the IBT
TR000	Output one data word to core or disk
XC000	Fill in exit control cells during pass 2
DC000	Process DSA statements
MV000	Output the DEFINE FILE Table
ML000	Check mainline loading address for validity
CK000	Check for overlay of core load and COMMON
DF000	Process the DEFINE FILE Table entries
FM000	Print a map of the DEFINE FILE Table

PHASE 3

Phase 3 performs four functions. It checks the information in subroutine header records (except ILSs) and stores it in RCOM. It also ensures that during pass 2 the subprograms are relocated by class, i.e., class 0 first, LOCALs second, class 1 third, class 2 fourth, and class 3 fifth. It compares the reference to each subprogram, i.e., CALL or LIBF, with the type (from the program header record). Lastly, as a subprogram is chosen for relocation, phase 3 checks whether or not it has already been relocated under a different name, i.e., another entry point.

The following is a list of the subroutines that comprise phase 3 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
HR000	Process the program header record for subroutines
CC000	Control the loading of subroutines by type
TY000	Verify subroutine references
SV000	Scan Load Table for multiple references

PHASE 4

Phase 4 performs two functions. It incorporates ILSs into the core load and it determines whether or not a core load fits in core storage or can be made to fit with SOCALs by computing the core that can be saved by employing SOCALs.

The following is a list of the subroutines that comprise phase 4 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
IL000	Fetch and relocate an ILS
ET000	Calculate core load size

PHASE 5

Phase 5 creates the Flipper Table if LOCALs have been specified, sees to it that the flipper is relocated, and provides the logic for building each SOCAL. This phase is flip-flopped with phase 3. It is brought into core storage once if there are LOCALs and once for each SOCAL, which implies a maximum of four times.

The following is a list of the subroutines that comprise phase 5 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
PL000	Process LOCAL subprograms
PS000	Process SOCAL subprograms
FF000	Relocate the LOCAL/SOCAL flipper, FLIPR

PHASE 6

Phase 6 performs several miscellaneous functions that must follow the actual building of most of the core load. The most important of these is the construction of the transfer vector from the Load Table. Other functions performed in phase 6 are filling in exit control cells and completing the core image header.

The following is a list of the subroutines that comprise phase 6 and the functions they perform.

<u>Subroutine</u>	<u>Function</u>
TV000	Build the transfer vector
TP000	Complete core image header, fill in exit control cells, etc.

PHASE 7

Phase 7 formats and prints (via the principal print device subroutine) all messages from R00-R10. These messages contain no variables.

PHASE 8

Phase 8 formats and prints (via the principal print device subroutine) all messages from R16-R23. These messages contain a 5-character name following "R XX".

PHASE 9

Phase 9 formats and prints (via the principal print device subroutine) all messages from R39-R47. These messages contain a hexadecimal address following "R XX".

PHASE 10

Phase 10 formats and prints (via the principal print device subroutine) all messages from R64-R68. These messages contain a 5-character name following "R XX".

PHASE 11

Phase 11 formats and prints (via the principal print device subroutine) the files portion of the map. It is entered only if (1) a map is requested and (2) there are files defined.

PHASE 12

Phase 12 formats and prints (via the principal print device subroutine) the allocations of core storage. It is entered only if a map is requested.

DEBUGGING/ANALYSIS AIDS

Stopping the Core Load Builder at the proper time is often the key to pinpointing problems in monitor system and, in some cases, user programs. There are NOP instructions in several critical locations in the Core Load Builder; they are: LK000+1, PM000+1, IN000+1, MC000, E1000+1, E2000+1, E3000+1, and E4000+1. These NOPs can be replaced by WAIT instructions so that core dumps can be taken at various stages of the core load building process. A WAIT replacing the NOP at PM000+1 is often the most useful, for it can be used to stop the Core Load Builder just before an error message is printed.

Bear in mind that, even though an error is detected by the Core Load Builder, it may well be due to a failure somewhere else in the monitor system. The message printed may not be a very good indication of the error; many checks are present in the Core Load Builder simply to keep it from destroying itself. For example, a common message is R16, and the name given in the message may well be something that makes no sense or was not referenced in the core load. The problem may well be erroneous output from the FORTRAN Compiler or Assembler Program or a destroyed User Area. In such a case an analysis of the contents of the data buffer BUFLO usually provides the clue to the error.

To facilitate path tracing through the Core Load Builder, all subroutines in the Core Load Builder are entered with BSI instructions.

FLOWCHARTS

CCAT: DUP01
 DCTL: DUP02
 STORE: DUP03
 FILEQ: DUP04-06
 DDUMP: DUP07-08
 DUMPLET: DUP09
 DELETE: DUP10
 DEFINE: DUP11
 DEXIT: DUP12
 PRECI: DUP13

The Disk Utility Program (DUP) is actually a group of programs provided by IBM to perform certain frequently required operations involving the disk such as storing, moving, deleting, and dumping data and/or programs. These operations are called, for the most part, by user-supplied DUP control records.

DUP OPERATION

When DUP is called, the phases CCAT and DUPCO are brought into core storage. CCAT forms the required DUP I/O subroutine sets (phases 14, 15, 16) and records them. CCAT also forms the balance of UPCOR, including CATCO and the principal print device subroutine, and is completely overlaid by part of UPCOR, leaving only the DUPCO part of phase 1 in core storage as part of UPCOR.

Control is passed to REST (of DUPCO) and REST in turn calls DCTL into core storage.

In general, DCTL reads, prints, decodes and checks the control records, and then calls in the required phase to continue processing as the function requires.

The called phase completes the function, including the printing of the terminal message. Control is then passed to REST (of DUPCO), which restores CATCO areas to zero as required for initialization, fetches DCTL if it is not already in core (4K system), and branches to DCTL to read the next record.

When a monitor control record is read, a CALL EXIT is executed by DEXIT.

CORE STORAGE LAYOUT

Figure 13 shows the layout of core storage during DUP operation. Panel 1 shows the overlay scheme used for 4K systems, panel 2 for 8K systems, panel 3 for 16K systems, and panel 4 for 32K systems.

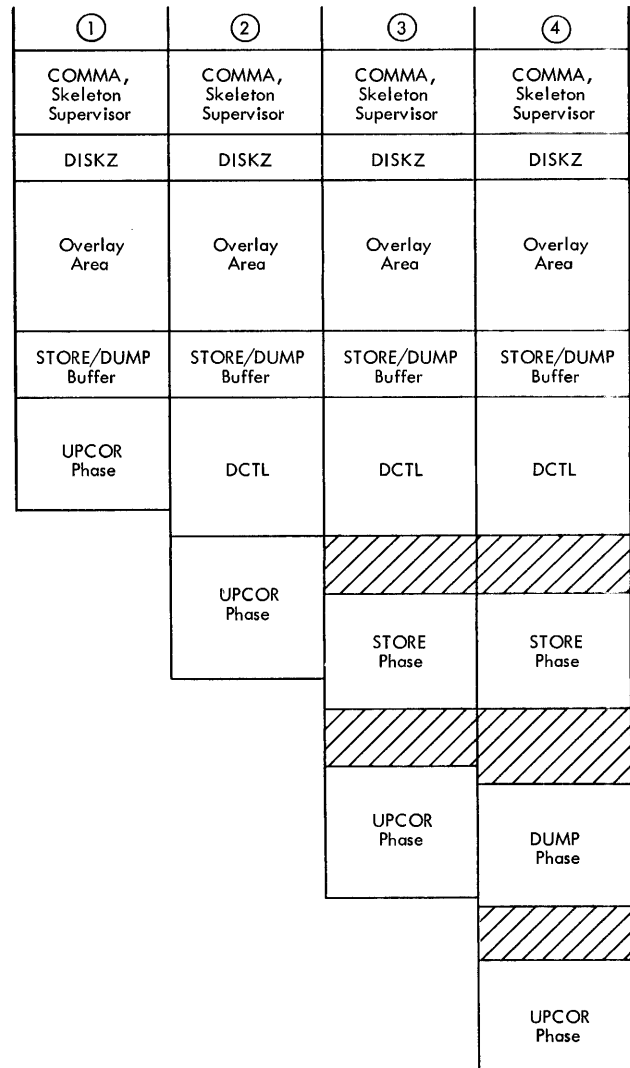


Figure 13. Core Layout During Disk Utility Program Operation

DUP CONTROL RECORDS

In the table below, DUP control records are classified by type according to the phases required to complete their processing.

Type	Phases Required
STORE STOREMOD STOREDATA STOREDATA CI	DCTL, STORE, DUPCO
STORECI	DCTL, FILEQ, STORE, DEXIT, Core Load Builder, PRECI, DCTL, STORE, DUPCO
DUMP DUMPDATA	DCTL, DDUMP, DUPCO
DELETE	DCTL, DLETE, DUPCO
DEFINE FIXED AREA DEFINE VOID ASSEMBLER DEFINE VOID FORTRAN	DCTL, DFINE, DUPCO
DUMPLET DUMPFLET	DCTL, DMPLT, DUPCO
DWADR	DCTL, DEXIT, ADRWS Program, DUPCO

LOCATION EQUIVALENCE TABLE (LET)/FIXED LOCATION EQUIVALENCE TABLE (FLET)

LET is the table through which the sector addresses of programs and data files stored in the User Area may be found. Each entry in this table consists of three words, the first two of which are the program or file name in name code. The third word is the disk block count of the program or file. Bits 0 and 1 of the first word denote the format of the entry, i. e., DSF, DCI, or DDF. The corresponding bit patterns are 00, 10, and 11. The 01 pattern is reserved for

future use. For a DSF subroutine having multiple entry points, the disk block count is zero for all entry points except the first.

Padding, wherever required to ensure that DCI programs and data files are sectorized, is reflected in LET as if a program called "1DUMMY" were stored. That is, each instance of padding generates a 1DUMMY entry in LET, and the block count for each of these entries is the number of disk blocks to the nearest sector boundary. The last entry in LET is always a 1DUMMY entry that reflects the number of disk blocks from the end of the last program stored in the User Area to the end of the disk.

Each sector of LET contains a header, which occupies the first five words of the sector. The first word contains the sector number, which is 0, 1, . . . , or 7. The second contains the sector address of the User Area for this cartridge. The third is reserved for future use. The fourth contains 315 minus three times the number of LET entries found in this sector, i. e., the number of words unused (available) in this sector. If this is not the last sector of LET on this cartridge, then the fifth word contains the address of the next sector of LET. If it is the last sector of LET and if there is no FLET on this cartridge, this word contains zero; otherwise, it contains the address of the first sector of FLET. In other words, this fifth word (chain address) is used to chain from LET through FLET, sector by sector. Bits 0-3 of the fifth word are always zeros. Note that, when referring to a dump of LET, the above header words are expressed in hexadecimal.

FLET is the table through which the sector addresses of programs and data files stored in the Fixed Area may be found. FLET is analogous to LET in the format of its entries and its use by the monitor system programs.

LET/FLET is searched by LETSR of DCTL for the name decoded from DUP control records of the STORE, DUMP, and DELETE types. The information required by other DUP phases is recorded in CATCO. If a DSF program is being stored, then LETSR also searches LET/FLET for the secondary entry point names as well.

STORE inserts the required entries into LET/FLET (one entry for each entry point). If a DCI program or data file is being stored and padding is required, then a 1DUMMY LET/FLET entry is inserted prior to the named LET/FLET entry. All secondary entry points have their entries on the same sector as the LET/FLET entry for the primary entry point.

DUP CONCATENATED COMMUNICATIONS
AREA (CATCO)

CATCO contains the following elements:

- DCOM values that are read from DCOM and placed in CATCO by CCAT of DUPCO.
- IOAR headers (word counts and sector addresses) required by DUP, furnished to CCAT by the System Loader, converted by CCAT to two-word entries each, and placed in DCOM by CCAT of DUPCO.
- Words used only by DUP for switches, small work areas, and communications between various DUP phases.
- I/O addresses used by DUP, initialized by CCAT of DUPCO.

DCOM VALUES

DCOM is read from the master cartridge by CCAT of DUPCO whenever DUP is called by the Monitor Control Record Analyzer. The following parameters in DCOM are used by DUP:

<u>Location</u>	<u>Relative Address</u> <u>(decimal)</u>
#ANDU	35
#BNDU	40
#CBSW	10
#CIAD	27
#CIBA	60
#CIDN	55
#CSHN	90
#DBCT	6
#DCSW	24
#ENTY	16
#FCNT	7
#FHOL	20
#FLET	75
#FMAT	70
#FPAD	45
#FRDR	19
#FSZE	21
#JBSW	9
#LCNT	11

<u>Location</u>	<u>Relative Address</u> <u>(decimal)</u>
#MDF1	13
#MDF2	14
#MPSW	12
#NAME	4
#NCNT	15
#PCID	50
#PIOD	25
#PPTR	26
#RP67	17
#SCRA	65
#TODR	18
#UHOL	22
#ULET	80
#USZE	23
#WSCT	85

These parameters are referred to by an index register that contains the address of the first word of DCOM plus the displacement given above.

Whenever a parameter in DCOM has been changed by DUP and control is being relinquished to another monitor system program, DUP writes the DCOM values in CATCO to DCOM on the master cartridge before exiting. If a change has been made that refers to a satellite cartridge, the DCOM values are also written to DCOM on the affected cartridge.

See the description of DCOM in Section 2. Communication Areas for details regarding the above parameters.

IOAR HEADERS

CATCO contains the IOAR header for each phase of DUP required by other phases during the execution of the various DUP functions; they are:

<u>Location</u>	<u>Phase Name</u>	<u>Phase Number</u>
DCHDR	DCTL	2
STHDR	STORE	3
FLHDR	FILEQ	4
DMHDR	DDUMP	5
DLHDR	DMPLT	6
DTHDR	DLETE	7
DFHDR	DFINE	8
DXHDR	DEXIT	9
UCHDR	UPCOR	10

<u>Location</u>	<u>Phase Name</u>	<u>Phase Number</u>
PIHDR	KFACE	11
SIHDR	CFACE	12
PTHDR	PFACE	13
CIHDR	PRECI	17

These headers are initialized by CCAT of DUPCO whenever the Monitor Control Record Analyzer calls DUP. The contents of these headers are not altered by any phase of DUP.

Each IOAR header consists of two words, word 1 containing the word count and word 2 containing the sector address of a phase. Each pair is aligned on an even boundary.

SWITCHES

The following DUP switches are initialized by CCAT of DUPCO and not altered by any function of DUP.

ADDR2 -- Keyboard interrupt address, to be put in location /0045 by MASK of DUPCO so that during a masked operation, the Keyboard interrupt is delayed by DUP until a critical operation is completed.

KBREQ -- Contents of location /0045, saved by CCAT of DUPCO when DUP is given control, and restored by DEXIT of DUP when leaving DUP.

INOUT -- Indicator for the principal I/O device when DUP was called by the Monitor Control Record Analyzer.
 Negative = Paper Tape.
 Zero = Cards.
 Positive = Keyboard.

PTPON -- A non-zero value if paper tape devices are present on the system.

IBT -- Nine locations containing the interrupt branch table for level 4, initialized by CCAT of DUPCO and the card and paper tape interface phases.

The following locations are used by DUP for internal communication, and are initialized to zero by REST before each DUP control record is processed.

ASMSW -- Set non-zero by DFCTL of DCTL when a DEFINE control record indicates that the Assembler Program is to be deleted from the master cartridge. Used by DFINE for functional flow control.

BITSW -- Set non-zero by RE015 of DCTL to allow the MDUMP subroutine in DUPCO to call the System Core Dump program while executing various DUP phases. It is set on the basis of the contents of column 35 of the DUP control records. This column is not normally used, but it may be used to obtain snap-shot core dumps while performing DUP operations. A zero punched in this column causes all possible DUP dumps to occur. Other numbers cause core dumps to be taken when the phase with the same phase ID is in control (See DUP Diagnostic Aids.).

BLKSW -- Set by the DUP I/O interface subroutine (in IOBLK) when reading control records if the record is neither a monitor control record (//) or a DUP control record (*D or *S). If turned on, DCTL turns it off and returns to the GETHO entry of the DUP I/O interface subroutine. This permits DUP to pass non-control records, including blanks, at the maximum rate of 1000 per minute with a single buffer.

CIERR -- Set to a DUP error code for an error detected by PRECI during a STORECI operation. DCTL checks CIERR when entered from PRECI (CISW is non-zero) and goes to DEXIT thru LEAVE of DUPCO with the specified error code. PRECI cannot go directly to LEAVE because DUP UPCOR may not be in core storage at this time due to the possibility of being overlaid by the core load being built.

CISW -- Set by DCTL when *STORECI is the function specified on the DUP control record. Used by DCTL to detect an entry from PRECI (during a *STORECI function). Used by STORE to determine the functional path to be used.

CLBSW -- Set non-zero by PRECI. Used by STORE to indicate an entry from PRECI after the Core Load Builder has built the core load for the STORECI function.

- CL1, CL2 -- The addresses of the lower and upper limits, respectively, of parameters in CATCO to be cleared to zero by REST of DUPCO.
- CNTNO -- Used by GETBI of the DUP I/O interface subroutine (in IOBLK) to record the count of binary records being read or punched. Permits checksum and sequence check operations.
- DATSW -- The binary equivalent of the decimal value in the count field of the DUP control record. Entered by DACNT of DCTL. A non-zero value represents either STOREDATA, DUMPDATA, DEFINE FIXED AREA count, or STORECI with *FILES, *LOCAL, and *NOCAL control records following. Contents are in disk blocks if the input is from disk, records if from an I/O device. Used by DUMP, STORE, DEFINE and FILEQ as a count; also used to control functional flow. FILEQ clears DATSW before calling STORE.
- DBADR -- Set by LETSR of DCTL to the disk block address of the program represented by the last LET/FLET entry searched. Used by DUMP and DELETE to indicate the disk block address of the desired program or data file.
- DELSW -- Set by LETSR of DCTL to point to the required entry in LET/FLET minus one word. Actually contains a value somewhere in the buffer LETAR. Used by DMPLT when dumping the entry point(s) or name of a single program. Used by DELETE to point to an entry in LET/FLET that is to be deleted. Used by STORE to point to an entry in LET/FLET where the entry point(s) is to be inserted.
- DFNSW -- Set by DFCTL of DCTL to indicate a DEFINE FIXED AREA operation. Used by FRLAB of DCTL to bypass the decoding of the FROM field.
- DKSAD -- Set by DUP30 and DUP34 of DUPCO to indicate the sector address (without a logical drive code) of the current GET or PUT operation.
- DUMPP -- Two words located on even boundary, set by all DUP phases requiring special monitoring dumps. Used by MDUMP of DUPCO to specify lower and upper limits to be dumped to the printer.
- FRWS -- Set non-zero by SC130 if the FROM field is Working Storage. Used by DCTL for functional flow control and error checking.
- FXSW -- Set non-zero by SC130 or SC170 of DCTL when either the FROM or the TO field, respectively, of the DUP control record specifies the Fixed Area or when the control record specifies DEFINE FIXED AREA. Used by DCTL for error checking and functional flow control. Used by DFINE, STORE and DUMP for functional flow control.
- FORSW -- Set non-zero by DFCTL of DCTL when a DEFINE control record indicates that the FORTRAN Compiler is to be deleted from the master cartridge. Used by DFINE for functional flow control.
- HOLSW -- Set non-zero by DCTL when a STOREDATA or a DUMPDATA control record contains an H in column 11. Used by STORE and DUMP to allow data movement without any conversion for the I/O devices, i. e., 1 word to 1 column or 1 frame.
- IOSW -- Set non-zero by either SC130 or SC170 of DCTL when any I/O device is specified in the FROM or TO field of the DUP control record. Used by DCTL for error checking and functional flow control. Used by DUMP and STORE for functional flow control.
- LETSW -- Set positive by LECTL and negative by FLCTL of DCTL. Used by DUMPLET to indicate, respectively, a full LET/FLET dump or a FLET dump only.
- LSTLF -- Set by LETSR of DCTL to the sector address (with a logical drive code) of the last LET/FLET sector searched. If only one sector was searched, then the address of that sector is entered in LSTLF. Used by DUMP and DELETE to identify the logical drive required.
- MODSW -- Set non-zero if STCTL of DCTL detected a STOREMOD function specified by the DUP control record. Used by STORE for functional flow control.
- NAMSW -- Set non-zero by LETSR of DCTL when a name is found in LET/FLET that matches the name

- specified on the DUP control record. Used by DCTL for error detection and functional flow control. Used by DUMPLET to indicate that only the specified LET/FLET entry is to be dumped.
- NEGSW -- Set non-zero by DFCTL of DCTL when a minus sign is detected in column 31 of a DEFINE FIXED AREA control record. Used by DEFINE to indicate expansion (zero) or contraction (non-zero) of the Fixed Area.
- PGMHL -- Word count (length) of the program header in DSF programs. Set by RDHDR of DCTL to the actual program header length. Used by STORE to start the placement of the first data header in the DSF output. Set by DUMP from the program header. Used by STORE to update LET with the required number of entries.
- P1442 -- Set by CCAT of DUPCO to contain the word count and sector address of the System 1442 subroutine. Used by DDUMP to read the System 1442 subroutine into core when dumping to cards.
- PHDUP -- Duplicate of \$PHSE to permit printed identification of the DUP phase requesting a core dump.
- PRPAR -- Two words specifying the default limits to be dumped by MDUMP. Set by any module of DUP desiring to use MDUMP for monitoring DUP's status. Usually set to point at key parameters and work areas.
- PRSW -- Set non-zero by SC170 of DCTL when printing is specified as the desired output on the DUP control record. Used by DCTL for error detection and by DUMP for functional flow control.
- PTSW -- Set non-zero by SC130 or SC170 when paper tape is specified in the FROM or TO field of the DUP control record. Used by DCTL for error detection and functional flow control. Used by DUMP for functional flow control.
- SDWDS -- The number of words yet to search in the current LET/FLET sector. Set by LETSR of DCTL. Used by LETSR of DCTL to test for the sector search complete condition.
- STCSW -- Set non-zero by ST400 of DCTL when the CI is detected in columns 11 and 12 of the STOREDATA CI control record. Used by STORE for functional flow control.
- STSW -- Set non-zero by STCTL of DCTL when a STORE control record is found. Set by LETSR of DCTL to the sector address (with a logical drive number) of the LET/FLET sector that contains the IDUMMY entry that may be replaced by the entry for the program to be stored. Used by DCTL for functional flow control. Used by STORE to hold the LET/FLET sector address and drive code prior to inserting the LET/FLET entry for the program or data file to be stored.
- TEMP1, TEMP2 -- Two words, on an even boundary, used by various phases of DUP for miscellaneous purposes; i. e., DUP10 of DUPCO returns four EBCDIC characters in TEMP1 and TEMP2 as the result of converting from binary to hexadecimal for purposes of printing.
- TOWS -- Set non-zero by SC170 if the TO field is Working Storage. Used by DCTL for error checking and functional flow control.
- T3MSW -- Set non-zero by STCTL of DCTL when a type 3 or 4 subroutine contains a SOCAL level number specified on the DUP control record. Used by STORE to modify the type field in the program header before storing the subroutine to disk.
- UASW -- Set non-zero by SC130 or SC170 of DCTL when either the FROM or TO field of the DUP control record specifies the User Area. Used by DCTL for error checking and functional flow control. Used by STORE for functional flow control.
- WSSW -- Set non-zero by SC130 or SC170 of DCTL if the FROM or TO field of the DUP control record specifies Working Storage. Used by DCTL for error detection. Used by DUMP and STORE for functional flow control.
- XEQSW -- Set non-zero by PLUS2 of DCTL when calling in the required DUP phase to indicate that execution of that phase is desired rather than returning to PLUS2. Set non-zero by any other DUP phase using GET of DUPCO to fetch other phases from the disk that are to be executed immediately. Used by GET of DUPCO to determine whether to return to the link address (zero) or to execute the phase just fetched (non-zero).

The following switches are initialized to zero by CCAT of DUPCO, set by PLUS2 of DCTL, and not reset by REST of DUPCO. These are cleared by DEXIT before UPCOR is saved in preparation to calling the Core Load Builder. This forces DCTL on return from the Core Load Builder via PRECI to fetch STORE again as it may have been overlaid by the core load being built.

PH2 -- Set non-zero by PLUS2 of DCTL when fetching another DUP phase. Used by REST of DUPCO: if zero, DCTL must be fetched from disk; if non-zero, DCTL has already been fetched.

PH3 -- Set non-zero by PLUS2 of DCTL when fetching STORE. Used by PLUS2 of DCTL: if zero, STORE must be fetched from disk; if non-zero, STORE has already been fetched.

PH4 -- Set non-zero by PLUS2 of DCTL when fetching DDUMP. Used by PLUS2 of DCTL: if zero, DDUMP must be fetched from disk; if non-zero, DDUMP has already been fetched.

IOREQ -- Set non-zero by PLUS2 of DCTL in case I/O other than from the specified I/O device is required (i. e., Keyboard input when the DUP operation is a STORE from cards). Checked by READ of DCTL, and, if still non-zero, the principal I/O section (DUP phase 14) is brought back into core storage.

I/O ADDRESSES

The following are the I/O addresses required by the various DUP phases. They are initialized by CCAT of DUPCO when DUP is given control. All except THIS and NEXT remain as initially set. All (except SDBUF) contain the address of an I/O buffer in UPCOR. Thus, the locations of the referenced buffers are dependent on core size; in any case, they always reside in the upper 4K of core storage. SDBUF always resides in the first 4K of core storage.

CRBUF -- Set to the address of an 81-word buffer used for reading DUP control records in unpacked EBCDIC.

HDBUF -- Set to the address of the buffer used for printing the page heading after each page restore

performed during any DUP operation. The contents of this buffer are in packed EBCDIC.

IOBLK -- Set to the starting address for the I/O block portion of UPCOR. The I/O block contains one of phases 14, 15, or 16 of DUP.

SDBUF -- Set to the address of the 322-word buffer used by the STORE, DDUMP, and DELETE functions of DUP. This buffer always resides in the first 4K of core storage.

LETAR -- Set to the address of the 322-word buffer used for the LET/FLET search of DUP. This buffer is also a one-sector buffer, or the second half of a two-sector buffer used in disk I/O operations.

PEBUF -- Set to the address of a buffer used for printing the DUP control records (41 words in packed EBCDIC) or for printing a LET dump, a FLET dump, a program dump, or a data dump (61 words in packed EBCDIC).

THIS -- Set to the address of one of two buffers used for double buffering of binary input (see NEXT). The buffer is 81 words long.

NEXT -- Set to the address of one of two buffers used for double buffering of binary input (see THIS). The buffer is 81 words long.

DUP PHASE DESCRIPTIONS

DUP COMMON (DUPCO)

- Initializes the I/O phases required by DUP and builds the DUP Communications Area (CATCO).
- Performs functions commonly required by other DUP phases.

The initialization function of DUPCO is performed by a subroutine known as CCAT. CCAT resides in an area reserved for the System print subroutine, but is not overlaid by it until all other initialization has been completed. This initialization includes:

- Construction of the DUP paper tape I/O phase if paper tape is attached. This phase is written to

the disk area reserved for DUP phase 16 at System generation time.

- Construction of the DUP principal I/O phase (without Keyboard). This phase is written to the disk area reserved for DUP phase 15 at System generation time.
- Construction of the DUP principal I/O phase. This phase is written to the disk area reserved for DUP phase 14. This phase is left in core storage at IOADR.
- Initialization of all I/O-dependent switches in CATCO.
- Incorporation of DCOM from the master cartridge into CATCO.
- Incorporation of IOAR headers (word counts and sector addresses) of other DUP phases into CATCO. This information is supplied to CCAT by the System Loader.
- Initialization of DUP's page heading buffer with the heading contained in sector @HDNG.
- Fetching the System device subroutine for the principal print device. This subroutine overlays all but a few words of CCAT. These last words are cleared to zero just before branching to REST.

The functions that are common to all DUP phases are included in the non-overlaid section of DUPCO. These functions are provided by the following subroutines:

WRTDC -- This subroutine is used by STORE, DELETE, and DEFINE when it is necessary to update DCOM. This includes the updating of DCOM on any affected satellite cartridge as well as on the master cartridge.

PHIDM -- This subroutine is used to modify the next-to-high-order hexadecimal digit of \$PHSE in COMMA. It is used primarily by DUP's I/O functions to illustrate in a core dump the I/O operation last performed. The modifications are:

- | | |
|---|----------------|
| 1 | Read from disk |
| 2 | Write to disk |

- | | |
|---|--------------------------|
| 4 | Convert binary to EBCDIC |
| 5 | Print terminal messages |
| 8 | Read cards |
| 9 | Read paper tape |
| A | Read Keyboard |

PHID -- This subroutine is used to record the phase ID of the phase in execution in \$PHSE of COMMA. It is also used by some DUP phases to illustrate the progress of execution from one section of the phase to the next. When used for this purpose, the high-order digit of \$PHSE is changed to the appropriate phase section modifier. A core dump indicates the last section of the phase that was executed.

MASK -- This subroutine is used to prevent recognition of the INTERRUPT REQUEST key. The function of this key is to terminate the current job, but DUP must not allow this termination to take place while in a critical operation. Therefore, functions that affect LET/FLET, the User or Fixed Area, the CIB, and DCOM delay its recognition (STORE, DELETE, DEFINE).

LEAVE -- This subroutine is used to fetch DUP's exit phase (DEXIT) to print an error message or service a special exit, such as an exit to the Core Load Builder (STORECI function), an exit to the ADRWS program (DWADR function), or an exit to the Supervisor following the trapping of a monitor control record.

MDUMP -- This subroutine makes selective calls to the System Core Dump program. See DUP Diagnostic Aids.

BINEB -- This subroutine is used to convert binary numbers to EBCDIC hexadecimal characters. It is used primarily to convert a number for insertion into a phase termination message, e.g., a cartridge ID, a disk block count.

PRINT -- This subroutine is used to print a line on the principal print device. It interfaces with the System principal print device subroutine.

PAGE -- This subroutine is used to skip to channel 1 and print a page heading if the principal print device is the 1132 or 1403 Printer. If the Console Printer is the principal print device, five carriage returns are executed before the page heading is printed.

LINE -- This subroutine is used to single-space the principal print device.

REST -- This subroutine is used to chain from one DUP function to the next. When a function is completed without errors, the DUP phase in control prints its termination message and exits to REST. REST determines whether or not it is necessary to bring DCTL into core storage (i. e. , core size is 4K, or DCTL has not yet been loaded), and, if necessary, fetches DCTL. REST then exits to DCTL.

ENTER -- This subroutine is used to save the accumulator and extension, the overflow and status indicators, as well as XR1, XR2, and XR3. XR1 is then loaded with the address of the CATCO pointer.

RTURN -- This subroutine is used to restore the contents of the accumulator and extension, the overflow and carry indicators, as well as XR1, XR2, and XR3, as saved by the ENTER subroutine.

GET -- This subroutine is used to read the disk using DISKZ. XR3 points to the IOAR header for this read when GET is entered, and XR1 contains the address of the CATCO pointer.

PUT -- This subroutine is used to write to the disk using DISKZ. XR1 and XR3 are initialized in the same manner as described for the GET subroutine. Some error checking is done of the word count and sector address in the case of GET and PUT. No check is made in GET or PUT as to the validity of the logical drive code associated with the sector address. GET and PUT assume the proper logical drive code has been included with the sector address.

Errors Detected

The following DUP errors are detected in DUPCO by the GET and PUT subroutines: D92 and D93.

DUP CONTROL (DCTL)

- Reads, prints and decodes DUP control records.
- Sets switches in CATCO to reflect the parameters specified on the DUP control record.

- Searches LET for the name specified on STORE, DUMP, DUMPLET and DELETE type control records.
- Detects errors in the fields of the DUP control records.
- Calls into core storage the required DUP phase and exits to it.

DCTL remains in core storage during DUP operations for configurations of 8K or larger, except during STORECI. DCTL is executed after the REST function of DUPCO for each DUP control record as well as after PRECI during the processing of a STORECI control record.

DUP Control (DCTL) may be considered in three logical parts: the READ subroutine, the DCTL subroutines, and the PLUS2 subroutine.

READ Subroutine

This is the entry point into DCTL. It performs the following functions:

- Reads and prints DUP control records.
- Flushes invalid DUP subjobs.
- Checks for a monitor control record and exits when one is detected.
- Ensures that the required DUP I/O subroutine set is in core storage.
- Decodes the function field of the DUP control record into the various DUP types.
- Goes to one of the subroutines in the second logical part of DCTL to continue the decoding and processing of the specified type of DUP control record as follows:

<u>Control Record</u>	<u>Subroutine Called</u>
*STORE	STCTL
*DUMP	DUCTL
*DUMPDATA	DACTL
*DUMPLET	LECTL
*DUMPFLET	FLCTL
*DELETE	DLCTL
*DEFINE	DFCTL
*DWADR	WACTL

DCTL Subroutines

These subroutines make up the second logical part of DCTL. These subroutines perform the following functions for their particular control record type.

- Decode the balance of the Function field as required.
- Decode the FROM and TO device or area fields.
- Decode the Name field and perform a LET/FLET search if the name is required.
- Decode the Disk I/O subroutine required with STORECI.
- Decode and record the Count field as required. If the operation is a STORECI, FILEQ is brought into core storage using the PLUS2 subroutine.
- Decode the FROM and TO Cartridge ID fields.
- Check the validity of all fields and go to the DEXIT phase if any errors are detected.
- Prevent restricted phases from being called in and executed during Temporary mode of operation.
- Record all required data in CATCO for use by the required phase.
- Go to the appropriate entry point in the PLUS2 subroutine to fetch and transfer control to the DUP phase required to finish the processing of the specified DUP subjob.

PLUS2 Subroutine

This subroutine is the third logical part of DCTL and is the normal-exit subroutine. The various entry points to PLUS2 set up respective IOAR headers that cause the desired DUP phase to be called into core storage and executed. This subroutine performs the following functions:

- Sets up the IOAR header if the required DUP phase is not already in core storage.
- Fetches and/or executes the required DUP phase.

Buffers Used By DCTL

Control records are read into the area defined by CRBUF and converted to packed EBCDIC in the area defined by PEBUF. It is from PEBUF that the control record is printed by the principal print device subroutine and from which the various fields are decoded.

Binary records read for STORE are placed into the two buffers specified by THIS and NEXT in order to process secondary entry points that are on the header record.

LET/FLET sectors are read one at a time into the area specified by LETAR to be searched for the name specified on the control record.

Errors Detected

The following DUP errors are detected in DCTL: D01, D02, D03, D05, D06, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25, D26, and D27.

STORE

The STORE phase of DUP resides in the DUP phase overlay area if the core size is 4K or 8K, or in the overlay area plus 8K if the core size is 16K or 32K. This phase is read into core storage the first time DCTL recognizes a STORE control record. It remains in core storage on a 16K or 32K machine as long as DUP has control of the system (i. e., must be brought back into core storage when performing a STORECI operation).

STORE may be considered in seven logical parts, each of the following subroutines constituting one logical part.

ST000 Subroutine

This is the entry point to STORE. It serves as a master control for STORE, causing various other sections of STORE to be executed as they are required for various STORE operations.

IOWS Subroutine

This subroutine is executed whenever a STORE operation is from card or paper tape. It provides the following functions:

- Reads card or paper tape records punched in system, data, or core image format.
- Moves data from card or paper tape buffer(s) to a disk buffer.
- Writes the disk buffer to Working Storage (or to the User Area or Fixed Area if the operation is a STOREDATA or STOREDATA CI to the User Area or Fixed Area). If the operation is to Working Storage, Working Storage of the cartridge defined as the TO cartridge is used. By default, this is the System Working Storage.
- Updates the Working Storage disk block count and format when operation is to, or through, Working Storage.

WD000 Subroutine

This subroutine is executed whenever a STORE operation (except STORECI, STOREDATA, or STOREDATA CI from cards or paper tape) is to the User Area or the Fixed Area. It performs the following functions:

- Adjusts the destination User Area or Fixed Area disk block address to the next sector when not at a sector boundary, if the operation is a STOREDATA or STOREDATA CI.
- Makes the required disk block adjustment when moving a system format program to the User Area.
- Moves data or a program in Working Storage to the User Area or the Fixed Area. If the operation is from Working Storage, Working Storage of the cartridge defined as the FROM cartridge is used. By default, this is the System Working Storage. If the operation is a STORE from cards or paper tape, System Working Storage is used.

DOLET Subroutine

This subroutine is executed whenever a STORE operation is to the User Area or Fixed Area (except STOREMOD). It performs the following functions:

- Reads the LET or FLET sector to which the entry point name (or names) is to be added.
- Checks if a 1DUMY padding entry is required before the name is entered when storing data or core image programs to the User Area.
- If a LET sector cannot contain the entry, updates the header words of the LET sector and writes the completed LET sector to disk.
- Enters a name (or names) in a LET or FLET sector and updates the "words available" entry in the header. In the case of a LET sector, the terminal 1DUMY disk block size is decremented by the number of disk blocks stored.
- Decrements the 1DUMY size in the FLET sector by the number of disk blocks stored. All entries that follow this FLET entry are pushed to the right by the size of one entry (3 words).

UPDCM Subroutine

This subroutine is executed whenever the STORE operation is to the User Area (except STOREMOD). It updates DCOM as follows:

- Clears the Working Storage disk block count (#WSCT) of the TO drive in DCOM to zero.
- Puts the disk block address of the end of the User Area (plus one disk block) into #ANDU in DCOM (in the entry in the drive-dependent parameter for the cartridge affected by the STORE operation).
- Determines if the STORE operation is in Temporary mode; does not put the disk block address of the end of the User Area into #BNDU of the TO drive during Temporary mode.
- Updates the base file-protection address in DCOM (#FPAD) of the TO drive if the STORE operation is not during Temporary mode.

SNOFF Subroutine

This subroutine is executed by all STORE operations. It terminates the STORE function as follows.

- Moves the cartridge ID of the TO drive into the STORE terminal message.
- Moves the disk block address where the data or program was stored into the STORE terminal message.
- Moves the disk block count of the program or data into the STORE terminal message.
- Uses the PRINT subroutine in DUPCO to print the terminal message.
- Exits to the REST subroutine in DUPCO to clear the CATCO switches and restore DCTL if it is not in core storage.

ST700 Subroutine

This subroutine is executed when performing a STOREMOD operation. It performs the following functions:

- Computes the location within the User Area or Fixed Area sector at which the old version of the program begins.
- Moves the new version of the program into the buffer to replace the old version, one word at a time.

When an output sector is completed, it is written to the User Area or Fixed Area. The next User or Fixed Area sector is then read into the buffer to allow the word by word replacement to continue.

The entire STOREMOD process is under control of the disk block count. The number of disk blocks replaced by the new version is determined by the disk block count of the old version, as found in LET/FLET. STOREMOD does not alter this count.

Buffers Used By STORE

The disk buffer used for moving data or programs between Working Storage and the User Area or the

Fixed Area is the buffer specified by SDBUF. It is one sector long if the core size is 4K; otherwise, it is seven sectors long.

LET/FLET sectors are read and written one at a time, using the buffer specified by LETAR.

When STORE reads binary records, it reads them into the area specified by THIS and NEXT. If storing from cards, double buffering is used; THIS and NEXT are each considered to be 80 words long. If storing from paper tape, double buffering is not used; THIS and NEXT are considered to be an extended buffer, large enough to contain the maximum length record (108 words).

Errors Detected

The following DUP errors are detected in STORE: D30, D31, D33, D90, and D93.

FILEQ

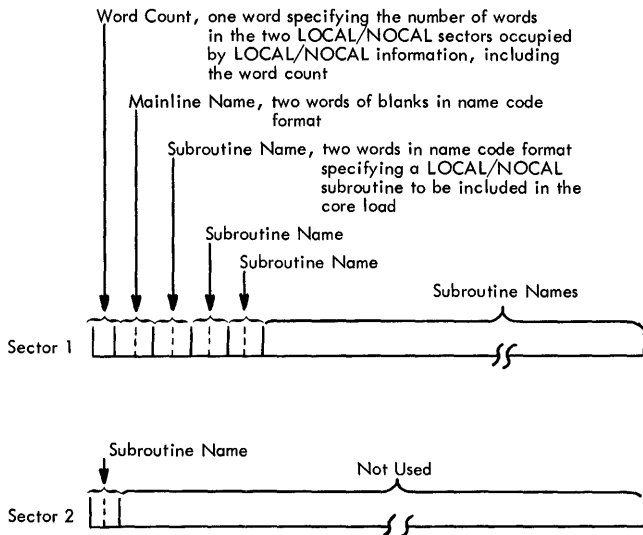
This phase of DUP is read into core storage by DCTL when the Count field (27-30) of the STORECI control record is non-zero. The function of FILEQ is to process the records following the STORECI control record and place the processed records into the SCRA for use by the Core Load Builder. Three types of STORECI control records are processed by this phase; *LOCAL, *NOCAL, and *FILES. FILEQ consists of the subroutines LC000, FR000, and LF200.

LC000

This subroutine processes *LOCAL and *NOCAL control records. A mainline name is not specified on these records when they follow the STORECI control record; an error is indicated if one is specified. Thus, the mainline name is set to blanks. All subroutine names specified in *LOCAL or *NOCAL control records are converted to name code.

LOCAL information for the core load to be built is stored in sectors 0 and 1 of the SCRA, NOCAL information in sectors 2 and 3. The first word of the LOCAL information is the number of words in the LOCAL sectors used to store the LOCAL information. A like word count precedes the NOCAL information in the NOCAL sectors.

The format of LOCAL/NOCAL information in the SCRA is shown below.

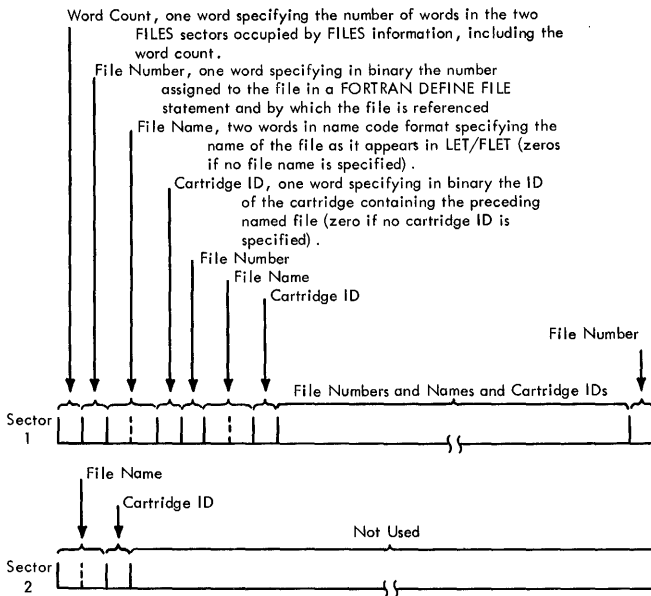


FR000

This subroutine processes *FILES control records. File numbers are converted to binary. File names, if specified, are converted to name code; if unspecified, the name is set to blanks. Cartridge IDs, if specified, are converted to binary; if unspecified, the ID is set to zeros.

FILES information for the core load to be built is stored in sectors 4 and 5 of the SCRA. The first word of the FILES information is the number of words in the FILES sectors used to store the FILES information.

The format of FILES information in the SCRA is shown below.



LF200

This subroutine provides the exit for FILEQ. When all *LOCAL, *NOCAL, and *FILES control records have been processed, DCTL is read into core storage using the GET subroutine in DUPCO with the execute switch (XEQSW) set. DCTL begins the processing of the header of the mainline program that is to be stored in core image format.

Buffers Used By FILEQ

The buffer used for writing the processed control records to the SCRA is referred to as SCRAB. SCRAB is another name for BUF7, known indirectly through CATCO as SDBUF.

Errors Detected

The following DUP errors are detected in FILEQ: D41, D42, D43, D44, D45, D46, D47, and D48.

DDUMP

The DDUMP phase of DUP resides in part of the first or fifth 4K block of core storage, depending on whether the core size is 4K, 8K, 16K, or 32K. Only on a 32K machine does the DDUMP phase reside in the fifth 4K block of core storage.

This phase is read into core storage the first time DCTL recognizes a DUMP control record and remains there on a 32K system as long as DUP is in control. This phase uses all the subroutines in DUPCO that are needed, e.g., disk reading and writing.

DD000

This is the mainline of the DDUMP phase. It initializes the parameters of the subroutines, sets up the IOAR headers for the areas used for input/output, and directs the execution of the subroutines.

XG000

This subroutine gets the words from the disk and, if the program is in DSF, the words are typed as to data, header indicator, or last data word.

XW000

This subroutine places the data or program in Working Storage.

XF000

This subroutine formats the data into a system or data record to be punched.

XP000

This subroutine checksums, unpacks, and punches the record formatted by XF000 on either cards or paper tape as specified.

XL000

This subroutine prints the data or program on the principal print device.

XC000

This subroutine clears the print area as directed by XL000.

XI000

This subroutine inserts the data or program words into the print area as directed by XL000.

Buffers Used By DDUMP

PEBUF -- 61-word buffer to hold printed output.

THIS -- 81-word buffer to read in cards to check to see if they are blank.

NEXT -- 111-word buffer from which the output is punched.

LETAR -- 322-word buffer to be used to get data from disk.

SDBUF -- 320-word buffer to be used to place data in Working Storage.

Errors Detected

The following DUP error is detected in DDUMP: D50.

DUMPLET/DUMPFLET

This phase of DUP prints the contents of the Location Equivalence Table (LET) and/or the Fixed Area Location Equivalence Table (FLET) in an easily readable format on the principal print device. The extent of the dump depends on the setting of the following three DUPCO switches:

LETSW. When this switch is positive, both LET and FLET are to be dumped; when negative, only FLET is dumped.

DRIVE. When this switch is negative, LET and/or FLET from all cartridges are dumped; when not negative, LET and/or FLET is dumped from the cartridge specified only.

NAMSW. If this switch is on, only the LET or FLET entry corresponding to the name in #NAME is printed.

One sector of LET/FLET is printed per page. Each sector of LET/FLET dumped is preceded by two lines of header information. The first header line contains the contents of the following locations from COMMA/DCOM: #CIDN, \$FPAD, #FPAD, #CIBA, #ULET, and #FLET.

Following this line is a second header line that reflects information concerning the LET/FLET sector being dumped, i. e., the sector number (SCTR NO.), User Area/Fixed Area (UA/FXA), words available (WORDS AVAIL), and chain address (CHAIN ADR).

Following these two header lines are the LET/FLET entries. Twenty-one lines of entries are printed, made up of five entries per line but sequenced by column.

Once LET and/or FLET have been dumped according to LETSW, DRIVE, and NAMSW, DUMPLET prints a terminal message and exits to the REST subroutine in DUPCO.

Buffers Used By DUMPLET/DUMPFLET

PRNTA -- 61-word buffer in UPCOR used for printing a line.

LETAR -- 322-word buffer in UPCOR used for reading a sector of LET/FLET.

DELETE

The DELETE phase removes programs and data files from either the User or Fixed Area, along with their corresponding LET/FLET entries.

DCTL passes control to DELETE after having read a DELETE control record and having found the specified entry in LET/FLET. This sector of LET/FLET is left in the buffer LETAR with DELSW pointing to the location previous to the specified entry. DBADR is set with the User or Fixed Area disk block address of the program to be deleted.

DELETE compresses LET/FLET by the number of words made available by the deleted entry. If the deletion is to be from the Fixed Area, the specified entry is replaced by a 1DUMMY entry of the same size. If there are adjacent 1DUMMY entries, they are combined to form a single 1DUMMY entry and FLET is compressed by 0, 3, or 6 words depending upon whether there are 0, 1, or 2 adjacent 1DUMMY entries.

If the deletion is to be from the User Area, the amount of the compression of LET is dependent upon the number of words made available by the specified LET entry. This number varies since DSF programs with multiple entries may be stored in the User Area. As in the case of FLET, adjacent 1DUMMY entries may cause additional compression of 3 or 6 words.

The packing of LET/FLET begins in the sector containing the entry to be deleted and continues throughout the remainder of LET/FLET. Since multiple entries must reside in a single LET sector, they are moved across a LET sector boundary only when room exists in the previous sector for all the entry points. LET is packed until a DCI program or data file is encountered. If a 1DUMMY entry precedes this entry, the 1DUMMY entry is updated to reflect the number of disk blocks required to make the DCI program or data file start at a sector boundary after it is moved the appropriate number of sectors. If a 1DUMMY entry did not precede the DCI program or data file, one of the appropriate size is inserted, if required, to sectorize the DCI program or data file. The shrinkage or packing continues until the last 1DUMMY entry of LET is found.

Packing of the User Area begins with the sector containing the program to be deleted. Subsequent DSF programs are shifted by disk blocks into the

area made available until a DCI program or data file is encountered. If the User Area is being packed by an amount equal to or greater than one sector, the remaining programs are moved by whole sectors.

After all required disk movement of the specified DELETE operation is complete, DELETE prints a terminal message to signify completion.

Buffers Used By DELETE

LETAR -- used for storage of LET/FLET sectors. Up to 2 sectors may reside in core storage with the addresses of the first word of each saved in DE918 and DE919.

SDBUF -- used to process the User Area. Two or eight sectors of core storage are used depending upon core size.

Errors Detected

The following DUP errors are detected in DELETE: D70, D71, and D72.

DEFINE

To initially provide a Fixed Area on a system cartridge or to increase its size, the Core Image Buffer (CIB), LET, and the User Area are moved toward and partly into Working Storage. Working Storage is reduced by the increased size of the Fixed Area. The sector address of the CIB is updated in DCOM and the Resident Image on the updated system cartridge. If a Fixed Area is defined on a non-system cartridge, LET is not shifted because it precedes the Fixed Area sector address.

To decrease the size of the Fixed Area on a system cartridge, the Core Image Buffer (CIB), LET, and the User Area are moved away from Working Storage and into a part of the existing Fixed Area. Working Storage is increased by the amount of the decrease in the size of the Fixed Area. DCOM and the Resident Image on the updated system cartridge are updated with the new sector address of the CIB. If the size of the Fixed Area is decreased on a non-system cartridge, LET is not shifted, as it precedes the Fixed Area sector address.

To delete the FORTRAN Compiler and/or the Assembler Program from a system cartridge, all succeeding programs and special purpose areas on the disk are moved away from Working Storage toward the voided area. Working Storage is increased by the size of the voided program(s).

Once the FORTRAN Compiler and/or the Assembler Program have been eliminated, neither can be restored without an initial load of the disk cartridge, including all the programs in the User or Fixed Area.

DEFINE determines if a system or non-system cartridge is being processed by testing the cartridge ID sector for the presence of the version and modification level number, which is zero on non-system cartridges.

If the FORTRAN Compiler and/or the Assembler Program are deleted, all SLET entries for that program(s) are cleared to zero. SLET is also revised to reflect the new sector addresses of those programs that are shifted.

All entries in the Reload Table indicating SLET lookups requested by the deleted program(s) are removed and the remaining Reload Table entries are packed together. The revised Reload Table is then reprocessed to generate new sector addresses where necessary in the monitor system programs.

Buffers Used By DEFINE

LETAR -- a disk I/O buffer.

SDBUF -- a disk I/O buffer.

Switches and Indicators

FORSW -- non-zero when the FORTRAN Compiler is to be deleted.

ASMSW -- non-zero when the Assembler Program is to be deleted.

FXSW -- non-zero when the Fixed Area is to be defined or modified.

DATSW -- indicates the disk block adjustment of the Fixed Area.

NEGSW -- non-zero when the Fixed Area is to be decreased.

Errors Detected

The following DUP errors are detected in DEFINE: D80, D81, D82, D83, D84, D85, and D86.

DEXIT

This phase is brought into core storage and executed by the LEAVE subroutine in DUPCO. DEXIT performs the following functions:

- Prints DUP error messages
- Traps monitor control records and exits to the Supervisor
- Links to the System Library program ADRWS for DWADR
- Passes control to the Core Load Builder for STORECI
- Returns control to MODIF when a modification includes changes to the System Library.
- Exits to the Supervisor upon recognition of the INTERRUPT REQUEST key interrupt.

DEXIT is called with an indirect branch via LEAVE. Following the branch instruction is a parameter that specifies the function to be performed by DEXIT.

If the parameter is a positive integer, DEXIT prints a DUP error message. The message printed corresponds to the integer parameter.

If the parameter is zero, DEXIT moves a trapped monitor control record from the buffer CRBUF to the Supervisor buffer @SBFR. \$CTSW in COMMA is set to minus one (/FFFF) to indicate to the Supervisor that the next monitor control record has already been read.

DEXIT checks to see if control should be returned to the System Maintenance program, MODIF. If #MDF2 in DCOM is non-zero, DEXIT reads DUP phase 18 into core storage and transfers to it. This phase contains part of MODIF, written on this sector when MODIF was last in control. In this manner MODIF is able to use DUP to delete an old version of a program or subroutine from the System Library, and then use DUP to store the new version.

If control is not given to MODIF, DEXIT transfers to \$EXIT in the Skeleton Supervisor.

If the parameter is minus two (-2), the interrupt caused by the INTERRUPT REQUEST key is recognized, causing the Supervisor to read records from the principal input device until the next JOB monitor control record is encountered. DEXIT exits via the \$DUMP entry point in the Skeleton Supervisor with a dump format code of minus two (-2).

If the parameter is minus three (-3), DEXIT transfers control to the Core Load Builder so that the DSF program currently in Working Storage can be converted to core image format for the STORECI operation. Before DEXIT transfers control to phase 0/1 of the Core Load Builder, the area in core storage bounded by IOADR and the end of core storage is written to DUP phase 13 (UPCOR). DUP phase 17 (PRECI) restores this area to core storage after the DCI program has been moved to the User or Fixed Area.

If the parameter is minus four (-4), DEXIT initiates a CALL LINK to the System Library program ADRWS to complete the DUP DWADR operation. DEXIT enters the Skeleton Supervisor at the \$LINK entry point with the name ADRWS specified in name code. The ADRWS program initializes the Working Storage sector addresses on the cartridge whose logical drive code is found in #TODR in DCOM. ADRWS causes control to be returned to DUP by placing a dummy DUP monitor control record into the Supervisor buffer @SBFR before returning control to the Skeleton Supervisor via the \$EXIT entry point.

All DEXIT functions write DUP's DCOM buffer to sector @DCOM on the master cartridge before exiting.

Buffers Used By DEXIT

CATCO -- used to write DCOM on sector @DCOM of the master cartridge. The buffer size is 112 words.

IOADR -- used to write UPCOR on DUP phase 13. The buffer size is approximately 1528 words.

B -- used to read phase 0/1 of the Core Load Builder into core storage. This read is executed from core locations 32-39. Approximately 480 words are read.

@SBFR - /0140 -- used to read the MODIF phase (DUP phase 18) into core storage. This read is executed from core locations 32-39. The number of words read is 320.

2501/1442 CARD INTERFACE (CFACE)

This phase serves as an interface between DUP programs and the system device subroutine for card I/O. CFACE consists of four subroutines; they are listed below along with their primary functions:

GETHO -- Reads a card and converts from IBM Card Code to unpacked EBCDIC.

GETBI -- Reads a binary card.

PACKB -- Converts from card binary (1 column per word) to packed binary (4 columns per 3 words), with checksumming.

PCHBI -- Punches a card from an 80-word buffer.

The phase has four entry points, one corresponding to each of the functions listed above. Each subroutine is entered by an indirect BSI instruction to the symbolic name listed above. Upon entry to each subroutine, the registers and status conditions of the calling program are saved using the ENTER subroutine in DUPCO. The PHIDM subroutine in DUPCO records the phase identification of CFACE.

Upon completion of the required function and prior to returning to the calling program, the original conditions of the calling program are restored using the RTURN subroutine in DUPCO.

GETHO

This subroutine is used by DCTL to read DUP control records. The system device routine for the 2501 or the 1442 is used to read the card. An 81-word buffer specified by CRBUF in CATCO is used for card input. GETHO examines each card for either //, *S, or *D in columns 1 and 2. If this information is not found in the first two columns of the card, the subroutine returns immediately to DCTL. In this way non-DUP control records can be bypassed at maximum card speed without using a double buffering technique.

If //, *S, or *D is found in columns 1 and 2, the subroutine loops till the whole card is read, then converts it from IBM Card Code to unpacked EBCDIC using the system conversion subroutine CDCNV.

GETHO packs the unpacked EBCDIC data from the 81-word input buffer to a 41-word packed buffer specified by PEBUF.

The subroutine then exits to DCTL.

GETBI

This subroutine reads a binary card into an 81-word buffer specified by THIS. The system device subroutine for the 2501 or 1442 is used to read the card.

PACKB

This subroutine converts card binary (one word per column) in an 81-word input buffer (NEXT) to packed binary (four columns per three words) in a 55-word output buffer (NEXT). The packed data overlays the unpacked data.

After packing, the checksum of the 54 words is verified. If the checksum is correct, the subroutine returns to the calling program. A checksum error causes an exit to LEAVE in DUPCO with an error parameter.

PCHBI

This subroutine, using the system device subroutine for the 1442, punches a card from an 81-word buffer specified by NEXT.

Buffers Used By CFACE

THIS -- 81-word input buffer used by GETBI for reading binary records.

NEXT -- 81-word buffer used by PACKB for packing binary records (4 columns per 3 words). The packed data overlays the unpacked data. Also used by PCHBI for outputting to the punch.

CRBUF -- 81-word input buffer used by GETHO for reading control records in unpacked EBCDIC.

PEBUF -- 41-word buffer used to hold the packed EBCDIC control record, converted from the unpacked EBCDIC in CRBUF.

KEYBOARD INTERFACE (KFACE)

KFACE serves as an interface for DUP programs when the principal input device is the Keyboard. KFACE is used by DCTL to read DUP control records from the Keyboard.

The PHIDM subroutine in DUPCO is used to record the KFACE phase identification. The ENTER subroutine in DUPCO is used to save the conditions of the calling program. A control record of up to 80 characters is read and converted from Keyboard code to unpacked EBCDIC by the system device subroutine for the Keyboard.

An EOF character causes the termination of input and the filling of the remainder of the buffer with blanks or a word count of 80, whichever is first.

The record is read into the 81-word buffer specified by CRBUF in CATCO in unpacked EBCDIC, then converted to packed EBCDIC and stored in the 41-word buffer specified by PEBUF in CATCO. The RETURN subroutine in DUPCO is used to restore the conditions of the calling program.

Buffers Used By KFACE

CRBUF -- 81-word input buffer used to hold a control record in unpacked EBCDIC.

PEBUF -- 41-word buffer used to hold the packed EBCDIC control record, converted from the unpacked EBCDIC in CRBUF.

1134/1055 PAPER TAPE INTERFACE (PFACE)

This phase serves as an interface between DUP programs and the system device subroutine for paper tape I/O. PFACE consists of four subroutines; they are listed below along with their primary functions:

GETHO -- Reads a paper tape record punched in PTTC/8 code and converts it to both unpacked and packed EBCDIC.

GETBI -- Reads a binary paper tape record.

PACKB -- Converts from unpacked binary (two frames per word) to packed binary, with checksumming.

PCHBI -- Punches a binary paper tape record.

The phase has four entry points, one corresponding to each of the functions listed above. Each subroutine is entered by an indirect BSI instruction to the symbolic name listed above. Upon entry to each subroutine, the registers and status conditions are saved using the ENTER subroutine in DUPCO. Another DUPCO subroutine, PHIDM, modifies the phase identification with the PFACE identification.

Upon completion of the requested function and prior to exiting to the calling program, the original conditions of the calling program are restored using the RTURN subroutine in DUPCO.

GETHO

This subroutine is used by DCTL to read DUP control records. The system device subroutine for the 1134/1055 is used to read the record. An 81-word buffer specified by CRBUF in CATCO is used to contain each record read. All conversion from PTTC/8 code to EBCDIC is performed by the system device subroutine. When the control record has been read, it is converted to packed EBCDIC within the 41-word buffer specified by PEBUF in CATCO. The subroutine then exits to DCTL.

GETBI

This subroutine reads a binary paper tape record into a 109-word buffer specified by THIS in CATCO. The system device subroutine for the 1134/1055 is used to read the record.

Note that THIS and NEXT are reversed for each record read when reading binary cards. When reading paper tape DCTL places the address of the buffer having the lowest core address into THIS, since the reading of binary paper tape records requires an extended buffer. The system device subroutine reads each frame (eight bits of data) into one word of the buffer.

The word count preceding each binary paper tape record is punched into a single frame, and is read separately from the body of the record. After the word count is read, it is checked for validity. If it is valid, it is used to read the record that follows. If it is not valid, the next frame is read to determine if it is a word count. In this way, delete characters and other special codes may appear between binary records or between a control record and the first binary record.

PACKB

This subroutine packs the binary record as read by GETBI into normal binary data. The packed data overlays the unpacked data.

After packing, the checksum of the words read is verified. If the checksum is correct, the subroutine returns to the calling program. A checksum error causes an exit to LEAVE in DUPCO with an error parameter.

PCHBI

This subroutine uses the system device subroutine for the 1134/1055 to punch a binary paper tape record from a 109-word buffer specified by NEXT.

Buffers Used By PFACE

THIS -- 109-word buffer used by GETBI for reading binary paper tape records.

The two buffers THIS and NEXT, used for double buffering when reading binary cards, constitute one extended buffer when reading binary paper tape records. Double buffering is not used for binary paper tape records.

NEXT -- 109-word buffer used by PCHBI for punching binary paper tape records, consisting of THIS and NEXT taken as consecutive buffers.

CRBUF -- 81-word input buffer used to hold a control record in unpacked EBCDIC.

PEBUF -- 41-word buffer used to hold the packed EBCDIC control record, converted from the unpacked EBCDIC in CRBUF.

PREC1

This phase is read into core storage by the Core Load Builder after a core load has been built for a STORECI function. This phase resides in the DUP overlay area. It moves the core load to the User Area or Fixed Area before restoring DUP phase 13 (UPCOR). Since UPCOR is not available to PRECI, copies of the DUPCO subroutines PHID, PHIDM, GET, and PUT have been incorporated into PRECI.

PREC1 is composed of five sections or logical parts -- PC000, PC040, PC100, PC180, and PC240.

PC000

This is the entry point to PRECI from the Core Load Builder. DCOM from the master cartridge is read into PRECI's work area, as is the core image header from the CIB. If the inhibit DUP function switch (\$NDUP) has not been set by the Core Load Builder, PRECI proceeds. Otherwise, the PC240 section is used to exit to DCTL after indicating an error has occurred.

DCOM is used to determine the logical drive code of the destination drive, as well as the starting sector address to which the program is to be moved. From the core image header the total length of the core load in sectors is determined. A check is then made to determine if the program can be contained in the User Area or Fixed Area. If the core load is too large, the PC240 section is used to exit to DCTL after indicating an error has occurred.

PC040

This section moves LOCAL/SOCAL sectors (if any) from Working Storage to their position at the end of the core load after it has been moved to the User Area or Fixed Area. The Working Storage sector address from which the LOCALs/SOCALs are moved includes an adjustment for Working Storage files when they are present. LOCALs/SOCALs are moved to the User or Fixed Area one sector at a time. The number of sectors required for Working Storage files and the number of LOCAL/SOCAL sectors are obtained from the core image header.

PC100

This section moves that part of the core load (including the core image header) that resides in the CIB to its destination in the User Area or Fixed Area. The length of the program (in words) and the starting address of the core load are used to control the number of sectors moved from the CIB to the User Area or Fixed Area. The core image header also indicates if the core load exceeds the 4K boundary; i. e., the core load's load address plus the number of words in the core load produces a number greater than 4095.

The core load residing in the CIB is moved to the User Area or Fixed Area four sectors at a time. This move continues until the CIB sector containing that part of the core load that resides at location 4095 has been moved.

PC180

This section moves that part of the core load that resides in core storage above location 4095 to the User or Fixed Area. The last sector of the core load written to the User or Fixed Area from the CIB is read into core storage so as to be contiguous with the part of the core load residing in core, i. e., above location 4095. The remainder of the core load, starting with the first word read into core storage from the User or Fixed Area, is then written to the User or Fixed Area, starting at the sector read into core storage.

PC240

This section restores DUP phase 13 (UPCOR), DCTL (phase 2), and then exits to DCTL. Before exiting to DCTL, however, termination data is placed in the DCOM buffer in CATCO, and in CATCO itself. The interrupt locations in low core storage are restored for DUP operation, and phase switches are cleared in order to ensure that DDUMP and STORE are reloaded from disk. If any errors have been detected during PRECI operation, the DUP error message number is communicated back to DCTL through CIERR in CATCO.

Buffers Used By PRECI

The disk I/O buffer used to move all parts of the core load to the User Area or Fixed Area is located

at BUF7. All references are direct, rather than indirect through SDBUF in CATCO, since UPCOR is not in core storage during the operation of PRECI.

DUP DIAGNOSTIC AIDS

GENERAL

DUP has provided a selective, dynamic dump of various core storage areas to facilitate problem analysis. The core dumps are under the control of column 35 of individual DUP control records. In general, to obtain a dump of a particular DUP phase, column 35 should contain the phase ID number of that phase. If column 35 contains zero, all DUP phases associated with the function named on the control record yield a core dump, starting with the execution of DCTL. The column 35 codes and the corresponding DUP phase yielding core dumps are shown below.

<u>Code in column 35</u>	<u>Phase(s) dumped</u>
0	All associated phases, starting with DCTL
2	DCTL
3	STORE
4	FILEQ
5	DDUMP
6	DUMPLET
7	DELETE
9	DEXIT

The core dumps include the following:

Disk I/O Areas -- The buffer area used for the disk I/O operation, including the area into which DUP phases are read, is dumped. The number of words dumped is dependent on the number of words read or written.

Buffer Areas -- The buffers between BUF4 and PRPNT are dumped in DCTL execution. This dump occurs when the LET search begins, when the LET search on one cartridge is complete and the next cartridge LET search begins, and before the required DUP phase (STORE, DUMP, or DELETE) is read.

CATCO -- This area of core storage is dumped in conjunction with any dump mentioned above.

If a dump (or dumps) is desired during CCAT execution, the next to the last card (i. e. , the last type A card) of DUPCO should be removed before reloading this phase with a System Loader reload function. This change in DUPCO causes all possible core dumps within DUP to occur regardless of the contents of column 35 in any DUP control record.

Core dumps are not allowed when the DEFINE or PRECI phase is in execution. The System Core Dump program uses the CIB to save the contents of core storage, and, since the CIB moves during a DEFINE operation and the core load to be stored by PRECI is in the CIB, serious errors occur if core dumps are taken when executing either DEFINE or PRECI.

PRECI

When PRECI is entered from the Core Load Builder, the phase identification word in COMMA (\$PHSE) is initially set to /0011 by the PHIDP subroutine in PRECI. As each subroutine of PRECI is executed, \$PHSE is modified by PHIDP as follows:

<u>Subroutine in execution</u>	<u>Contents of \$PHSE</u>
PC040	/1011
PC100	/2011
PC180	/3011
PC240	/4011

In addition, the second hexadecimal digit of \$PHSE is modified by the IDMP subroutine in PRECI each time a disk I/O operation is performed.

<u>Digit 2 of \$PHSE</u>	<u>Disk I/O Operation</u>
1	Read from Disk
2	Write to Disk

STORE

When STORE is entered from DUP control, the phase identification word in COMMA (\$PHSE) is initially set to /0003 by the PHID subroutine in DUPCO. As each subroutine of STORE is executed, \$PHSE is modified by PHID as follows:

Subroutine in execution Contents of \$PHSE

IOWS	/1003
WD000	/2003
DOLET	/3003
UPDCM	/4003
ST700	/5003
SNOFF	/6003

In addition, the second hexadecimal digit of the phase identification is modified by the PHIDM

subroutine in DUPCO each time an I/O operation is performed.

Digit 2 of \$PHSE

I/O operation

1	Read from Disk
2	Write to Disk
4	Convert Binary to EBCDIC
5	Print Terminal Message
8	Read Binary Cards
9	Read Binary Paper Tape

FLOWCHARTS

General:	ASM01
Phase 0:	ASM02
Phase 1:	ASM03
Phase 1A:	ASM04
Phase 2:	ASM05
Phase 2A:	ASM06
Phase 3:	ASM07
Phase 4:	ASM08
Phase 5:	ASM09
Phase 6:	ASM10
Phase 7:	ASM11
Phase 7A:	ASM12
Phase 8:	ASM13
Phase 8A:	ASM14
Phase 9:	ASM15-ASM17
Phase 10:	ASM18
Phase 10A:	ASM19
Phase 11:	ASM20
Phase 12:	ASM21
ERMSG:	ASM22
@ARCV	ASM23
@APCV:	ASM24

The Assembler Program is designed to translate the statements of a source program written in 1130 Assembler Language into a format that may be dumped and/or stored by DUP or executed directly from Working Storage.

Basically, the functions of the Assembler are:

1. Convert the mnemonic to machine language (except for Assembler control records).
2. Assign addresses to statement labels.
3. Insert the format and index register bits into the instruction, if applicable.
4. Convert the instruction operands to addresses or data.

INTRODUCTION

The Assembler Program is structurally divided into two parts, the resident portion and the overlay portion. The resident portion consists of the Assembler Communications Area (ASCOM), part of phase 0,

phase 9, and the phase 9 Communications Area (PHSCO). All of the other phases are called into core storage as overlays.

The Assembler Program is functionally divided into two parts, pass 1 and pass 2. The source program is read and processed, one statement at a time, during each of the two passes. During pass 1, the source program is read into core storage from the principal I/O device. Unless the user specifies by control record that two-pass mode is in effect, the source program is stored on the disk, from which it is reentered for pass 2 processing. If two-pass mode is specified or required, the source program is reentered via the principal I/O device for pass 2 processing. (If a list deck or paper tape object program is desired, the assembly must be made in two-pass mode.)

PROGRAM OPERATION

When the Monitor Control Record Analyzer detects an ASM monitor control record, it reads the first sector of the Assembler Program (phase 0) into core storage and transfers control to it.

Phase 0 reads into core storage all the subroutines required during assembly processing for I/O, and phase 9. The word counts and sector addresses of all the buffers and major overlay phases are initialized in ASCOM and in phase 9, and the boundary conditions are set for the Symbol Table. Phase 1 is then fetched and control is passed to it.

Phase 1 reads and analyzes control records, setting the appropriate switches in ASCOM for the options specified. Upon detection of the first non-control record, phase 1A is fetched and given control.

Phase 1A initializes the core addresses for the buffers, then fetches and transfers control to phase 2 to start statement processing.

Statement processing is performed by an op code search in phase 9 and a transfer through a branch table (that precedes every major overlay) to the overlay phase currently in core storage. If the required overlay phase is in core storage, execution of the phase proceeds. If it is not in core storage, the branch table causes a return to phase 9 to fetch the required overlay phase. The op code search is performed again and control is passed to the overlay

phase. When the overlay phase completes the necessary processing, another record is read and the entire process is repeated.

All overlays exit by branching to either the LDLBL or the PALBL subroutines within phase 9 and then to STRT9 (the op code search). A branch is taken to LDLBL when the statement just processed is permitted to have a label. A branch is taken to PALBL when a label is not permitted or is to be ignored. Both LDLBL and PALBL branch to the RDCRD subroutine (within phase 9) to read the next record just prior to their return to the overlay phase that called them.

ASSEMBLER COMMUNICATIONS AREA

The Communications Area (ASCOM) consists of all the indicators and switches referenced by more than one phase of the Assembler Program. All communication between phases is done through ASCOM. ASCOM resides in core storage following DISKZ and preceding the Overlay Area.

Refer to the program listings for details regarding the contents of ASCOM.

OVERLAY AREA

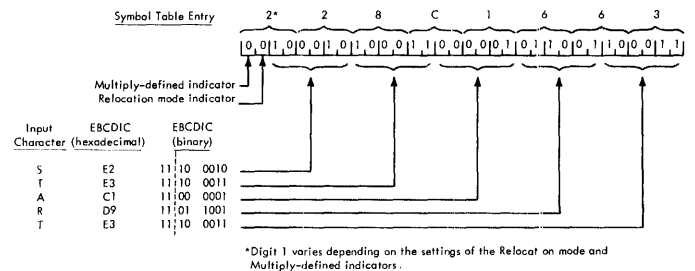
The Overlay Area is the area in core storage into which the statement processing (overlay) phases are loaded as required. Phase 0 is initially loaded into this area by the Monitor Control Record Analyzer (see Section 6. Supervisor). Phases 1, 1A, and 2 are sequentially loaded into this area at the start of passes 1 and 2. Phases 2A, 5, 6, 7, 7A, 8, 8A, @ARCV, and @APCV are loaded into this area during passes 1 and 2 as they are required to process specific mnemonics and constants, to handle output options, etc. Phase 12 is loaded into this area when the END statement is encountered in passes 1 and 2. Phase 4 is loaded into this area at the completion of pass 2. Phases 3, 9, 10, 10A, 11, and ERMSG are not loaded into the Overlay Area.

The Overlay Area resides in core storage following ASCOM and preceding phase 9.

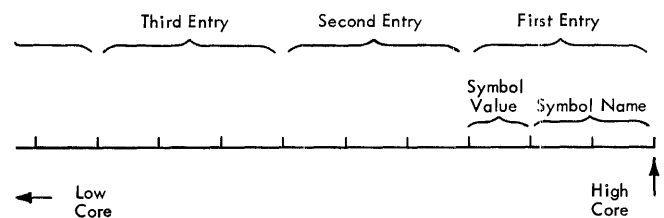
SYMBOL TABLE

As the source program is read and processed in pass 1, the Symbol Table is built. An entry is made

in the Symbol Table for each valid symbol defined in the source program. Each entry in the Symbol Table consists of three words. The first word contains the value of the symbol. Words 2 and 3 contain the symbol itself in name code. The following example shows the conversion of the symbol 'START' to name code for a Symbol Table entry:



The Symbol Table is built starting at the high-addressed end of core storage. Entries are added to the Symbol Table (see below) until its lower limit, the end of the principal print device subroutine, is reached. If symbols are added to the Symbol Table after the in-core Symbol Table has been filled (i. e., the lower limit has been reached), the overflow Symbol Table is saved, one sector at a time, in Working Storage on the disk. Note that Symbol Table overflow is possible only if the Assembler control record *OVERFLOW SECTORS reserving the required sectors on disk is present in the source program; otherwise, the assembly is terminated at the point of Symbol Table overflow.



As the source program is again read and processed in pass 2, the Symbol Table is searched each time a reference to a symbol is encountered. The in-core Symbol Table is searched first. If the symbol is not found in the in-core Symbol Table and

Symbol Table overflow has been written on disk, Phase 10A is fetched into core storage to perform the search of the overflow sectors.

A binary search technique is used in the Symbol Table search.

INTERMEDIATE I/O

During pass 1 of an assembly made in one-pass mode the source program statements are saved on disk for input to pass 2. Each source statement is saved starting at column 21 and ending with the rightmost non-blank column. As saved on disk each statement consists of a prefix word containing the number of words up to and including the prefix word for the following statement and the source statement packed two EBCDIC characters per word.

The intermediate I/O is written on disk in Working Storage, starting at the first sector of Working Storage if no OVERFLOW SECTORS were specified, or starting at the first sector following the last sector of Symbol Table overflow.

If a LIST Assembler control record is not present in the source program, comments statements are not saved in the intermediate I/O.

During pass 2 of an assembly in one-pass mode, the intermediate I/O is read into core storage, one sector at a time. The source statements are unpacked and placed into the two I/O buffers such that they are indistinguishable from statements read from the principal I/O device.

DOUBLE-BUFFERING

All card, paper tape, or Keyboard input to the Assembler Program is double-buffered, with one exception; if the assembly is being made in two-pass mode and the LIST DECK or LIST DECK E option has been specified, input during pass 2 is single-buffered to facilitate punching.

The Assembler Program uses two 80-word buffers for double-buffering. While a record is being read into one buffer, the record in the other buffer is being converted and processed.

Two locations in ASCOM are used to point to the two input buffers. One location (RDBFR) always con-

tains the address of the buffer in which a record is being processed. The other location (RDBFR+1) always contains the address of the buffer into which a record is being read. The RDCRD subroutine in phase 9, which interfaces with the principal I/O device subroutine, exchanges the buffer addresses in RDBFR and RDBFR+1 after each record is read.

PHASE DESCRIPTIONS

PHASE 0

Phase 0 serves as the Assembler Program's loader. First, the Communications Area (ASCOM) is initialized by phase 0, and the required I/O device subroutines are fetched into core storage. Phase 9, the resident portion of the Assembler Program, is also fetched into core storage. Phase 0 initializes the sector addresses and word counts of the various buffers utilized by the Assembler Program, and the boundary conditions for the System Symbol Table are established. Phase 0 then fetches phase 1 into the overlay area and transfers control to it.

The switches \$NDUP and \$NXEQ (in COMMA) are set non-zero to inhibit program execution and/or DUP functions in the event the assembly is terminated before completion.

A Master Overlay Control subroutine (P0130), the subroutine interfacing with DISKZ (DISK1), and the index register restoring subroutines (STXRS and LDXRS) are part of phase 0 and remain in core storage during the entire assembly; the rest of phase 0 is overlaid by phase 1. The Master Overlay Control subroutine performs the fetching of all the overlay phases and transfers control to the phase just read into core storage.

PHASE 1

Phase 1 reads, analyzes, and lists the Assembler control records. As each control record is analyzed, the various options specified by the control record are indicated in the Assembler Program's Communications Area (ASCOM). When the first non-control-type record is encountered, phase 1 transfers to the Master Overlay Control subroutine to fetch phase 1A and transfer control to it.

PHASE 1A

Phase 1A determines the address of the DSF buffer and initializes its IOAR header information in the Assembler Program's Communications Area (ASCOM). If the principal input device is either the 1134 Paper Tape Reader or the Keyboard, the current record is moved over 20 positions to the right and the read-in address is set for position 21 of the I/O area.

The number of overflow sectors assigned is checked for a maximum of 32 and the adjusted Working Storage boundary for the disk output of the source and object programs is initialized. Phase 1A then transfers control to the Master Overlay Control subroutine to fetch phase 2, which begins statement processing.

PHASE 2

Phase 2 handles the processing of all ENT, ISS, LIBR, ABS, EPR, SPR, ILS, and FILE statements. These statements are header mnemonics and must appear as the first non-control-type statements if they are included in the source program. For this reason phase 2 is the first overlay phase loaded into core storage to begin statement processing. As each particular header mnemonic is processed, the necessary indicators are set in the Assembler Program's Communications Area (ASCOM). The ordering and compatibility of the various header mnemonics is checked, and the program header record information is built and saved in ASCOM. When a mnemonic not handled by phase 2 is detected, control is transferred to the op code search (STRT9) in phase 9.

PHASE 2A

Phase 2A is called into core storage when a FILE statement is detected by phase 2. Phase 2A is loaded by a flipper routine within phase 2 and overlays part of phase 2. Phase 2A obtains the file information from the FILE statement and builds the 7-word DEFINE FILE Table. At the completion of phase 2A processing, control is returned to the flipper routine in phase 2. The flipper routine restores the overlaid portion of phase 2 and branches to the op code search (STRT9) in phase 9.

PHASE 3

Phase 3 is called into core storage as part 1 of the Assembler Program's exit to the Supervisor. Due to the size of phase 3, it overlays part of phase 9 instead of the Overlay Area.

The options of printing, punching, or saving the Symbol Table, as requested by the user on Assembler control records, are performed. At the completion of the Symbol Table options processing, phase 4 is fetched into core storage and control is transferred to it.

PHASE 4

Phase 4 performs the final processing for the Assembler Program and is called into core storage by phase 3. If overflow sectors were specified by Assembler control record, and if Symbol Table overflow occurred in assembling the program, the object program, which is residing on the disk, is moved back to the sector boundary at the start of Working Storage. Phase 4 then updates DCOM on all ready drives. The last record read by the Assembler Program, the record following the END statement, is moved to the Supervisor buffer.

In terminating the assembly, phase 4 prints four sign-off messages:

1. The number of errors flagged in the assembly
2. The number of symbols defined
3. The number of overflow sectors specified
4. The number of overflow sectors required

Phase 4 then exits to the Skeleton Supervisor at the \$EXIT entry point.

PHASE 5

Phase 5 is called into core storage to process HDNG, ORG, BSS, BES, EQU, LIST, EJCT, or SPAC statements. These mnemonics are all non-imperative type statements requiring similar processing and are all grouped, therefore, in the same overlay phase.

PHASE 6

Phase 6 processes all imperative instructions and the DC statement. Since these mnemonics are used most frequently, the processing for them was grouped into one overlay phase.

PHASE 7

The two mnemonics processed by phase 7 are XFLLC and DEC. The conversion of the data in the statement operands to binary is handled in phase 7A. Upon completion of the conversion, the data is formatted into the appropriate floating, fixed, or decimal format.

PHASE 7A

Phase 7A is fetched into core storage by a flipper routine within phase 7. Phase 7A converts the mantissa of a decimal integer, a fixed- or floating-point number to the binary equivalent.

Phase 7A contains a scanning process that converts the operand to its binary equivalent and a post-scanning process that converts from powers of 10 to powers of 2.

The converted decimal data is saved in a buffer that is part of the flipper routine. When the conversion is completed, phase 7A returns to the flipper routine that restores phase 7 and transfers control to it.

PHASE 8

Phase 8 processes the LIBF, CALL, EXIT, LINK, EBC, DSA, and DN statements. The processing of the program linking mnemonics -- LIBF, CALL, EXIT and LINK -- is combined with the processing of the data definition mnemonics -- EBC, DSA, and DN -- since otherwise they would constitute two small phases. This is satisfactory in terms of assembly time since EBC, DSA, and DN are not frequently used mnemonics.

PHASE 8A

Phase 8A processes the DMES statement. The DMES processing is performed by a scanning subroutine and a conversion subroutine. The scanning subroutine

scans and evaluates the operand field, one character at a time. The conversion subroutine contains a table of packed Console Printer codes and a table of packed 1403 Printer codes. The conversion is performed using an algorithm.

PHASE 9

The phase 9 communications area (PHSCO) consists of the entry addresses of the common subroutines within phase 9. Immediately following PHSCO is the op code search.

STRT9

The op code obtained from the input record is saved in ASCOM. All possible op codes are resident in a table, which has for each op code a two-word indicator followed by a corresponding one-word machine language mnemonic. The op code search is performed by means of a table lookup. When a match is found, the corresponding machine language mnemonic is picked up and saved in OPCNT in ASCOM. Bits 13-15 of OPCNT form a branch table displacement. Using this displacement, an indirect branch is taken thru the table to the overlay prepared to process this op code. If the overlay is in core storage, execution proceeds. If it is not, a return is made to the op code search to fetch the required overlay phase into core storage. Control is then passed to the overlay and statement processing continues.

BTHEX

BTHEX is a binary-hexadecimal conversion subroutine. The binary data is entered in the accumulator left-justified, and the hexadecimal output is stored by index register 1. The number of characters to be converted is in index register 2.

B4HEX

The B4HEX subroutine is entered when four hexadecimal output characters are desired. Index register 2 is set to four and a branch is made to BTHEX.

SCAN

The SCAN subroutine collects the elements of the operand field, character by character, performs any arithmetic functions necessary, and evaluates the operand.

GTHDG

GTHDG is the new page subroutine. Branches are made to the principal print subroutine to skip to channel 1, and print the heading as specified in the last HDNG statement encountered.

LDLBL

The LDLBL subroutine scans the label field of the statement. In pass 1, valid labels are added to the Symbol Table if they do not already appear there. The record is saved in the intermediate output buffer by INT1 and a branch is made to read the next record (RDCRD). When the next record is in core, a check is made for the last card. If the last card has been detected, a branch is made to the principal conversion routine (CVADR), and control is returned to the calling program.

In pass 2, the record is listed, if a listing has been requested or the record is in error. The next record is fetched and control is returned to the calling program.

PALBL

The PALBL subroutine is a secondary entry to the LDLBL subroutine. PALBL is entered when a label is not permitted on a statement being processed.

GETER

A branch is made to GETER when an error occurs during the assembly. GETER fetches the error message phase (ERMSG) into the first disk buffer (BUFI).

RDCRD

RDCRD is the interface subroutine for the principal input device subroutine. The input buffer is cleared

to EBCDIC blanks, the input buffer addresses in ASCOM are exchanged, and a branch is made to the principal input device subroutine to read a record. Index register 1 is set to point at the current input buffer and control is transferred back to the calling program.

If the record previously read was a monitor control record, phase 4 is fetched into core storage and control is transferred to it.

P9MVE

The P9MVE subroutine moves the input record from the input buffer to the print buffer. As it moves the record, each character is checked for validity. At the completion of the move, a branch is made to the principal print device subroutine to list the record and control is returned to the calling program.

INT1

INT1 is the subroutine used in pass 1 to pack the input record into two EBCDIC characters per word and save it in the intermediate output buffer. INT1 is overlaid by phase 11 for pass 2 processing. INT2 is the entry address for phase 11 during pass 2.

PHASE 10

Phase 10 consists of two subroutines: DTHDR and WRDFO. Phase 10 is fetched by phase 12 in pass 2 processing. Phase 10 overlays that section of phase 9 dealing with the inserting of labels into the Symbol Table.

DTHDR

DTHDR enters a data header into the object program output in disk system format (DSF) when required and completes the previous data header.

WRDFO

This subroutine writes one sector of disk system format (DSF) output when the DSF buffer is full. After the sector is written, those words past the 320th word of the buffer are moved back to the beginning of the buffer.

In the event the buffer is not full and WRDFO is entered from phase 12 END statement processing, the DSF buffer is written to Working Storage.

PHASE 10A

Phase 10A is fetched into core storage whenever necessary to handle Symbol Table overflow. When a symbol is to be added to an overflow sector in pass 1 or when the overflow sectors are to be searched for a symbol in pass 2, phase 10A is called. Phase 10A overlays the INT1 subroutine in phase 9 when called during pass 1; it overlays phase 11 (the INT2 subroutine) when called in pass 2 of an assembly in one-pass mode.

PHASE 11

Phase 11 is fetched into core storage in pass 1 during phase 12 END statement processing if the assembly is in one-pass mode. The function of phase 11 is to read the source statements from the disk during pass 2. The source statements saved in pass 1 are read back onto core storage in pass 2 in such a way that they are indistinguishable from statements read from the principal I/O device.

PHASE 12

In pass 1, phase 12 builds the program header record in the DSF buffer. Several counters are reinitialized in ASCOM and the buffer pointers are reset for disk system format (DSF) output. Phase 10 is fetched into core storage.

If the assembly is in two-pass mode, phase 1 is fetched and control is passed to it. If the assembly is in one-pass mode, the END statement is saved in the intermediate I/O buffer and the buffer is written to the disk. Phase 11 is fetched into core storage and the first sector of intermediate I/O is read into the first disk buffer (BUFI). A branch is then made to phase 11 to transfer the first statement from the intermediate I/O buffer to the source input buffer. Phase 1 is then fetched onto core storage and control is transferred to it.

In pass 2, phase 12 branches to DTHDR to build the end-of-program data header. If the source program is a type 3, 4, 5, 6, or 7 (not a mainline), an execution address of zero is saved in ASCOM and in

the source statement buffer. If the source program is a type 1 or 2 (a mainline), the execution address, i. e., the END statement operand, is saved in ASCOM and in the source statement buffer. The last sector of DSF output is written to the disk and the disk block count of the program is saved in ASCOM. Phase 12 then fetches phase 3 and transfers control to it.

ERMSG

ERMSG is called by the GETER subroutine within phase 9 when an error occurs during the assembly process. It is loaded into the first disk buffer (BUFI). A list of error messages is contained within ERMSG. This list is referenced by index register 2, which contains the appropriate error code at the time of entry to ERMSG. The error message is printed and control is then returned to GETER.

@ARCV

The Assembler Program's card conversion subroutine converts IBM Card Code to EBCDIC on input from the 2501 Card Reader or from the 1442 Card Read Punch, Models 6 and 7. The conversion is handled by a direct table access method, utilizing a 256 EBCDIC character table.

@APCV

The Assembler Program's punch conversion subroutine handles both the conversion and the punching of data on card output options. The punch conversion subroutine is fetched into the overlay area to perform the LIST DECK and LIST DECK E output options. It overlays the op code search (STR9) in phase 9 to perform the PUNCH SYMBOL TABLE output option.

CORE LAYOUT

Figure 14, panel 1 shows the layout of the contents of core storage after phase 0 of the Assembler Program has been fetched and given control by the Monitor Control Record Analyzer.

Figure 14, panel 2 shows the layout of the contents of core storage during pass 1.

Figure 14, panel 3 shows the layout of the contents of core storage during pass 2 of an assembly in two-pass mode.

Figure 14, panel 4 shows the layout of the contents of core storage during pass 2 of an assembly in one-pass mode.

①	②	③	④	⑤	⑥
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISKZ	- DISKZ	DISKZ	DISKZ	DISKZ	DISKZ
	ASCOM	ASCOM	ASCOM	ASCOM	ASCOM
Phase 0	Resident Phase 0	Resident Phase 0	Resident Phase 0	Resident Phase 0	Resident Phase 0
	Overlay Area	Overlay Area	Overlay Area	Overlay Area	Overlay Area
	Phase 9	Phase 9	Phase 9	Phase 9	Phase 9
			Phase 10	Phase 10	Phase 10
	INT1	Phase 10A	INT1	Phase 11	Phase 10A
			HDNG Buffer	HDNG Buffer	HDNG Buffer
	Card I/O Buffers	Card I/O Buffers	Card I/O Buffers	Card I/O Buffers	Card I/O Buffers
	Print Buffer	Print Buffer	Print Buffer	Print Buffer	Print Buffer
	Disk I/O Buffer	Disk I/O Buffer	Disk I/O Buffer	Disk I/O Buffer 1	Disk I/O Buffer 1
	Principal I/O Device Subroutine	Principal I/O Device Subroutine	Principal I/O Device Subroutine	Disk I/O Buffer 2	Disk I/O Buffer 2
	Principal Print Device Subroutine	Principal Print Device Subroutine	Principal Print Device Subroutine	Principal Print Device Subroutine	Principal Print Device Subroutine
	S. T. Overflow Buffer	S. T. Overflow Buffer	S. T. Overflow Buffer	S. T. Overflow Buffer	S. T. Overflow Buffer
	Symbol Table	Symbol Table	Symbol Table	Symbol Table	Symbol Table

Figure 14. Core Layout During Assembler Program Operation

FLOWCHARTS

General:	FOR01
Phase 1:	FOR02
Phase 2:	FOR03
Phase 3:	FOR04
Phase 4:	FOR05-FOR06
Phase 5:	FOR07-FOR08
Phase 6:	FOR09
Phase 7:	FOR10
Phase 8:	FOR11
Phase 9:	FOR12
Phase 10:	FOR13
Phase 11:	FOR14
Phase 12:	FOR15
Phase 13:	FOR16
Phase 14:	FOR17
Phase 15:	FOR18
Phase 16:	FOR19
Phase 17:	FOR20
Phase 18:	FOR21
Phase 19:	FOR22
Phase 20:	FOR23
Phase 21:	FOR24
Phase 22:	FOR25
Phase 23:	FOR26
Phase 24:	FOR27
Phase 25:	FOR28
Phase 26:	FOR29
Phase 27:	FOR30

The FORTRAN Compiler translates source programs written in the 1130 Basic FORTRAN IV Language into machine language object programs. The compiler also provides for the calling of the necessary arithmetic, function, conversion, and input/output sub-routines during the execution of the object program.

GENERAL COMPILER DESCRIPTION

The FORTRAN Compiler consists of 27 sequentially executed phases:

Phases 1 and 2 are initialization and control phases, processing the control records and building the initial statement string.

Phases 3 thru 10 are specification phases, processing the specification statements and other definitive information and building the basic Symbol Table.

Phases 11 thru 18 are compilation phases, analyzing and processing the source statements and replacing them with object coding.

Phases 19 thru 26 are the output phases.

Phase 27 is a recovery phase, terminating the compilation and executing a CALL EXIT.

Thus, the FORTRAN Compiler is a "phase" compiler; the compiler is passed-by the source program, which resides in core and is massaged into the object program.

PHASE OBJECTIVES

The following is a list of the compiler phases by number and name, and their major functions:

<u>Phase Number</u>	<u>Phase Name</u>	<u>Function</u>
1	Input	Process the control records; read the source statements and build the string.
2	Classifier	Determine the statement type and place the type code in the ID word.
3	Check Order/ Statement Number	Check for the presence and sequence of SUBROUTINE, FUNCTION, Type, DIMENSION, COMMON, and EQUIVALENCE statements; place statement numbers in the Symbol Table.

<u>Phase Number</u>	<u>Phase Name</u>	<u>Function</u>	<u>Phase Number</u>	<u>Phase Name</u>	<u>Function</u>
4	COMMON/SUBROUTINE or FUNCTION	Place COMMON variable names and dimension information in the Symbol Table; check for a SUBROUTINE or FUNCTION statement and, if found, place the name and dummy argument names in the Symbol Table.	11	Subscript Decomposition	Calculate the constants to be used in subscript calculation during execution of the object program.
5	DIMENSION/REAL, INTEGER, and EXTERNAL	Place DIMENSION variable names and dimension information in the Symbol Table; indicate the appropriate mode for REAL and INTEGER statement variables.	12	Ascan I	Check the syntax of all arithmetic, IF, CALL, and statement function statements.
6	Real Constant	Place the names of real constants in the Symbol Table.	13	Ascan II	Check the syntax of all READ, WRITE, FIND, and GO TO statements.
7	DEFINE FILE, CALL LINK, and CALL EXIT	Check the syntax of DEFINE FILE, CALL LINK, CALL EXIT statements; determine the defined file specifications.	14	DO, CONTINUE, etc.	Replace DO statements with a loop initialization statement and insert a DO test statement following the DO loop termination statement; process BACKSPACE, REWIND, END FILE, STOP, PAUSE, and END statements.
8	Variable and Statement Function	Place the names of variables, integer constants, and statement function parameters in the Symbol Table.	15	Subscript Optimize	Replace subscript expressions with an index register tag.
9	DATA Statement	Check the syntax of the DATA statement, check its variables for validity, and reformat the statement.	16	Scan	Change all READ, WRITE, GO TO, CALL, IF, arithmetic, and statement function statements into a modified form of Polish notation.
10	FORMAT	Convert FORMAT statements into a special form for use by the input/output subroutines during execution of the object program.	17	Expander I	Replace READ, WRITE, GO TO, and RETURN statements with object coding.
			18	Expander II	Replace CALL, IF, arithmetic, and statement function statements with object coding.

<u>Phase Number</u>	<u>Phase Name</u>	<u>Function</u>
19	Data Allocation	Allocate a storage area for variables in the object program.
20	Compilation Errors	List unreferenced statement numbers, undefined variables, and error codes for erroneous statements.
21	Statement Allocation	Determine the storage allocation for the object program coding.
22	List Statement Allocation	List the relative statement number addresses, if requested.
23	List Symbol Table	List the subprogram names in the Symbol Table and the System Library subroutine names in the string, if requested.
24	List Constants	Compute the addresses of the constants; list the addresses, if requested.
25	Output I	Build the program header and data header records and place them in Working Storage; place the real and integer constants into Working Storage.
26	Output II	Complete the conversion of the string to object coding and place the object program into Working Storage.
27	Recovery	Print the compilation termination message; exit by executing a CALL EXIT.

CORE LAYOUT

Figure 15, panel 1 shows the layout of the contents of core storage after the Monitor Control Record Analyzer has fetched phase 1 of the FORTRAN Compiler into core storage and has passed control to the control record analyzer portion. The principal print and principal input device subroutines have been fetched by phase 1. The card input and print buffers have been allocated by phase 1.

Figure 15, panel 2 shows the layout of the contents of core storage after the control record analyzer portion of phase 1 has passed control to the statement input portion. The control record analyzer portion of the phase is overlaid by the input statement string.

Figure 15, panel 3 shows the layout of the contents of core storage during the execution of phases 2 through 18. During these phases the boundary separating the statement string and Symbol Table fluctuates as the string and Symbol Table are massaged.

Figure 15, panels 4 and 5 show the layout of the contents of core storage during the execution of phases 19 through 24. Panel 4 reflects the contents of core storage before any printing has been performed in those phases. In panel 5 the lower-addressed portion of these phases has been overlaid by the print buffer when printing has been performed.

Figure 15, panels 6 and 7 show the layout of the contents of core storage during the execution of phase 25. Panel 6 reflects the contents of core storage before any object coding has been generated and written to disk. In panel 7 the lower-addressed portion of the phase has been overlaid by the disk output buffer when object coding has been generated and written to disk.

Figure 15, panel 8 shows the layout of the contents of core storage after control has been passed to phase 26. The disk buffer has been allocated by phase 26 and is not an overlay.

Figure 15, panel 9 shows the layout of the contents of core storage after control has been passed to phase 27. The principal print device subroutine has been fetched by the phase. The DCOM buffer has also been allocated by the phase.

①	②	③	④	⑤	⑥	⑦	⑧	⑨
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISKZ	DISKZ	DISKZ	DISKZ	DISKZ	DISKZ	DISKZ	DISKZ	DISKZ
	Statement String	Statement String	Statement String	Statement String	Statement String	Statement String	Statement String	DCOM Buffer
Phase 1, Control Record Analyzer Portion								Phase 27
Print Buffer	Print Buffer	Symbol Table	Symbol Table	Symbol Table	Symbol Table	Symbol Table	Symbol Table	
Primary Card Buffer	Primary Card Buffer							
FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	FORTRAN Communications Area	
Principal Input Device Subroutine	Principal Input Device Subroutine	Phases 2-18	Phases 19-24	Print Buffer	Phase 25	Disk Buffer	Disk Buffer	
Secondary Card Buffer	Card Buffer			Phases 19-24		Phase 25	Phase 25	
Phase 1, Statement Input Portion	Phase 1, Statement Input Portion		Principal Print Device Subroutine	Principal Print Device Subroutine		Principal Print Device Subroutine	Principal Print Device Subroutine	Principal Print Device Subroutine
ROL Subroutine	ROL Subroutine	ROL Subroutine	ROL Subroutine	ROL Subroutine	ROL Subroutine	ROL Subroutine	ROL Subroutine	

Figure 15. Core Layout During FORTRAN Compiler Operation

FORTRAN COMMUNICATIONS AREA

The FORTRAN Communications Area consists of 16 words of storage where information obtained

from the FORTRAN control records and compiler-generated addresses and indicators are kept. This information is available to any phase needing it. The contents of the FORTRAN Communications Area words are described in Table 3.

Table 3. The Contents of the FORTRAN Communications Area

Word	Symbolic Name	Description of Contents
1	SOFS	The address of the start of the string.
2	EOFS	The address of the end of the string.
3	SOFST	The address of the start of the Symbol Table.
4	SOFNS	The address of the start of the non-statement-number entries in the Symbol Table.
5	SOFXT	Phases 1-20. The address of the start of the Symbol Table entries for SGTs (subscript-generated temporary variables).
6	SOFGT	Phases 21-25. The work area word count. Phases 1-20. The address of the start of the Symbol Table entries for GTs (generated temporary storage locations). Phases 21-25. The constant area word count.
7	EOFST	The address of the end of the Symbol Table.
8	COMON	Phases 1-19. The address of the next available word for COMMON storage. Phase 20. The address of the highest-addressed word reserved for COMMON storage. Phase 21. Not used. Phases 22-25. Relative entry point.
9	CSIZE	All phases except Phase 20. The COMMON area word count. Phase 20. The address of the lowest-addressed word reserved for COMMON storage.
10	ERROR	Bit 15 set to 1 indicates overlap error. Bit 14 set to 1 indicates other error.
11-12	FNAME	The program name (obtained from the NAME control record, or a SUBROUTINE or FUNCTION statement) stored in name code.
13	SORF	Set positive to indicate FUNCTION. Set negative to indicate SUBROUTINE. 0 indicates mainline.
14	CCWD	Control card word. Bit 15 set to 1 indicates Transfer Trace. Bit 14 set to 1 indicates Arithmetic Trace. Bit 13 set to 1 indicates Extended Precision. Bit 12 set to 1 indicates List Symbol Table. Bit 11 set to 1 indicates List Subprogram Names Bit 10 set to 1 indicates List Source Program. Bit 9 set to 1 indicates One Word Integers.
15	IOCS	IOCS Control Card Word. Bit 15 set to 1 indicates 1442 Card Read Punch, Model 6 or 7. Bit 14 set to 1 indicates 1134/1055 Paper Tape Reader Punch. Bit 13 set to 1 indicates Console Printer. Bit 12 set to 1 indicates 1403 Printer. Bit 11 set to 1 indicates 2501 Card Reader. Bit 10 set to 1 indicates Keyboard. Bit 9 set to 1 indicates 1442 Card Punch, Model 5. Bit 8 set to 1 indicates Disk Storage. Bit 7 set to 1 indicates 1132 Printer. Bit 3 set to 1 indicates 1627 Plotter. Bit 1 set to 1 indicates Unformatted Disk I/O Area.
16	DFCNT	The number of files defined.

PHASE AREA

The Phase Area is the area into which the various phases of the compiler are read by the ROL subroutine. The size of the Phase Area is determined by the size of the largest phase of the compiler.

Each phase, when loaded into the Phase Area, overlays the preceding phase. There are two phases, however, that are exceptional in that they are loaded at some location other than the Phase Area origin. The ROL subroutine in phase 1 is loaded into high-addressed storage by phase 1 so that it occupies initially the position it will occupy throughout the compilation. The control record analysis portion of phase 1 is loaded into the String Area. This portion of the phase is in use only until the FORTRAN control records have been processed and is overlaid by the source statements. Phase 27, the Recovery Phase, which is executed after the object program has been produced, is also read into the String Area.

STRING AREA

During compilation the String Area contains both the statement string and the Symbol Table. The statement string is built by the Input Phase in an ascending chain beginning in the low-addressed words of the String Area. The Symbol Table is built during the compilation process in a descending chain beginning in the high-addressed words of the String Area.

The statement string expands and contracts as it is massaged during the compilation process. The Symbol Table expands as items are removed from the statement string and added to the Symbol Table. In addition, some phases move the entire statement string as far as possible toward the Symbol Table. The last statement of the string then resides next to the last Symbol Table entry. As the phase operates on the statement string, now referred to as the input string, it is rebuilt in the low-addressed end of the String Area. The rebuilt string is referred to as the output string. This procedure allows for expansion of the statement string.

If at any time during the compilation an entry cannot be made to the statement string or the Symbol Table due to the lack of sufficient storage, an overlap error condition exists. In the event of such an overlap condition, the remaining compilation is bypassed and an error message is printed (see Compilation Errors). Either the size of the source program or the number of symbols used must be decreased, or the program must be compiled on a machine of larger storage capacity.

SYMBOL TABLE

The Symbol Table contains entries for variables, constants, statement numbers, various compiler-generated labels, and compiler-generated temporary storage locations.

The first entry of the Symbol Table occupies the three highest-addressed words of the String Area, i. e., the 3 words just below the first word of the FORTRAN Communications Area. The second entry is positioned in the lower-numbered words adjacent to the first entry, etc.

During the initialization of the Symbol Table in phase 1, three words are reserved for the first Symbol Table entry. This entry is not made, however, until phase 3. From this point the size of the Symbol Table varies from phase to phase until it achieves its largest size in phase 18. Its size always includes the three words reserved for the next Symbol Table entry.

During phases 3 through 18, the Symbol Table contains variables, constants, and statement numbers. Information for these entries has been removed from the statement string and has been replaced by the address of the ID word of the corresponding Symbol Table entry. Also, the Symbol Table contains the various compiler-generated labels and temporary storage locations used in compilation.

During the output phases, 19 through 26, these entries in the Symbol Table are replaced by object program addresses that are inserted into the object coding by phase 26.

Format

All entries in the Symbol Table consist of three words -- an ID word and two Name-Data words. The entries for dimensioned variables are exceptional, however, in that they are six-word entries, the additional three words containing the dimension information.

The ID word occupies the lowest-addressed word of the three word entry. The Name-Data words occupy the two higher-addressed words.

A typical three-word entry is illustrated in the following example of the entry for the integer constant 290.

ID Word	Data Word 1	Data Word 2
	2 9 0	blank blank
1100000000000000	0110010111001110	0000000000000000
Lowest-Addressed Word		Highest-Addressed Word

Entry in hexadecimal form - E000 65CE 0000

The entry for the subprogram name COUNT would appear as:

ID Word	Name Word 1	Name Word 2
	C O U	N T
0000000010000000	1000011010110100	1100010101100011
Lowest-Addressed Word		Highest-Addressed Word

Entry in hexadecimal form - C080 86B4 C563

ID Word

The layout of the Symbol Table ID word is given in Table 4. The ID word is formed when the entry is placed in the Symbol Table.

Name-Data Words

The Name-Data words of Symbol Table entries have the following format:

<u>Word</u>	<u>Bit</u>	<u>Contents</u>
1	0	0, if the following 15 bits contain the first half of a constant; 1, if the following 15 bits contain anything other than the first half of a constant.
	1-15	First 15 bits of the 30-bit Symbol Table entry
2	0	Same as bit 0 of word 1
	1-15	Second 15 bits of the 30-bit Symbol Table entry

Dimensioned Entries

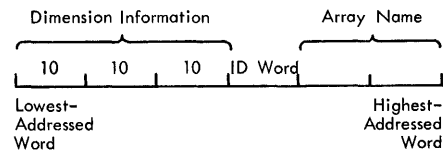
The Symbol Table entry for a dimensioned variable requires six words: two for the array name, one for the ID word, and three for the dimension information. The dimension information occupies the three lowest-addressed words, the ID word occupies the next higher-addressed word, and the Name-Data words occupy the two highest-addressed words.

Table 4. The Contents of the FORTRAN Symbol Table ID Word

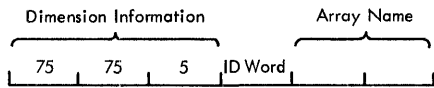
Bit Position	Status and Meaning
0	1 - Constant 0 - Variable
1	1 - Integer 0 - Real
2	1 - COMMON
3-4	01 - One dimension 10 - Two dimensions 11 - Three dimensions
5	1 - Statement function parameter/dummy argument
6	1 - Statement number
7	1 - Statement function name
8	1 - Subprogram name
9	1 - FORMAT statement number
10	1 - Referenced statement number or defined variable
11	1 - External
12	1 - Generated temporary storage location (GT)
13	1 - Subscript-generated temporary variable (SGT)
14	1 - Allocated variable
15	Not Used

Three words are always used for the dimension information regardless of the number of dimensions.

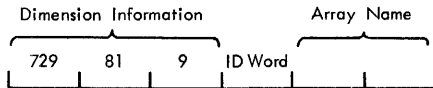
For one-dimensional arrays, all three dimension words contain the integer constant that specifies the dimension of the array. For example, the entry for array ARRAY (10) would appear as:



For two-dimensional arrays, the first (highest-addressed) dimension word contains the integer constant of the first dimension; the middle and last (lowest-addressed) dimension words both contain the product of the first and second dimension integer constants. Thus, the dimension information for array B(5, 15) appears as:



For three-dimensional arrays, the first dimension word contains the integer constant of the first array dimension; the middle dimension word contains the product of the first and second dimension integer constants; the third dimension word contains the product of the first, second, and third integer constants of the array dimensions. The dimension information for array C(9, 9, 9) appears as:



STATEMENT STRING

The source statements are read by the Input Phase, converted to EBCDIC, and stored in core storage. The first statement is stored starting at \$ZEND, and each succeeding statement is placed adjacent to the previous statement, thus forming the source statements into a string. The area within which the source statements are stored is referred to as the String Area.

ID Word

For identification purposes, as each statement is placed in the String Area, an ID word is added at the low-address end of each statement. The ID word has the following format:

<u>Bit</u>	<u>Contents</u>
0-4	Statement type code
5-13	Statement Norm
14	Varied; used for interphase communication
15	1, if statement is numbered; otherwise, 0

The Norm is the only portion of the ID word completed by the Input Phase. The Norm is a count of

the number of words used to store that statement, including the ID word and statement terminator.

The statement type codes, shown in Table 5, are added in the Classifier Phase (except for FORMAT statements).

Table 5. FORTRAN Statement ID Word Type Codes

Code	Statement Types
00000	Arithmetic
00001	BACKSPACE
00010	END
00011	END FILE
00100	SUBROUTINE
00101	REWIND
00110	CALL
00111	COMMON
01000	DIMENSION
01001	REAL
01010	INTEGER
01011	DO
01100	FORMAT
01101	FUNCTION
01110	GO TO
01111	IF
10000	RETURN
10001	WRITE
10010	READ
10011	PAUSE
10100	Error
10101	EQUIVALENCE
10110	CONTINUE
10111	STOP
11000	DO test
11001	EXTERNAL
11010	Statement Function
11011	Internal Output Format
11100	CALL LINK, CALL EXIT
11101	FIND
11110	DEFINE FILE
11111	DATA

Statement Body

Each statement, after being converted to EBCDIC, is packed two EBCDIC characters per word. This is the form in which the statements are initially added to the statement string.

Statement Terminator

Statements are separated by means of the statement terminator character, a semicolon. The statement terminator character indicates the end of the statement body. This character remains in the string entry throughout the compilation process for that particular statement type.

All statements in the statement string carry the statement terminator except for FORMAT and CONTINUE statements and compiler-generated error statements. Error statements inserted into the string by the compiler are inserted without the terminator character.

COMPILATION ERRORS

When an error is detected during the compilation process, the statement in error is replaced by an appropriate error statement in the statement string. Each error statement is added during the phase in which the corresponding error is detected and the procedure is the same in all phases.

1. The type code in the erroneous statement's ID word is changed to the error type.
2. The statement body is replaced by the appropriate error number. The statement number, if present, is retained in the Symbol Table and the Symbol Table address is retained in the error statement on the string.
3. The statement string is closed up, effectively deleting the erroneous statement from the statement string.

Error statements in the statement string are exceptional in that they do not carry the statement terminator character (semicolon).

Error indications are printed at the conclusion of compilation. If a compilation error has occurred, the message

OUTPUT HAS BEEN SUPPRESSED

is printed and no object program is placed in Working Storage.

Error messages appear in the following format:

CAA ERROR AT STATEMENT NUMBER
XXXXXX+YYY

where C indicates the FORTRAN Compiler, AA is the error number, XXXXX is the last encountered statement number, and YYY is the count of statements from the last statement number.

See the Programming and Operator's Guide publication for a list of the FORTRAN error numbers, their explanations, and the phases during which they are detected.

In addition to the errors, undefined variables are listed by name at the end of compilation. Undefined variables inhibit the output of the object program.

The initialization of each phase includes an overlap error check. If, at any time during the compilation, the statement string overlaps the Symbol Table, or vice-versa, or the remainder of the compilation is bypassed and the message

PROGRAM LENGTH EXCEEDS CAPACITY

is printed.

COMPILER I/O

The compiler uses the DISKZ subroutine for all disk I/O operations required during compilation.

The compiler uses the principal input device subroutine to read the control records and source statements to be compiled, and the principal input conversion subroutine to convert the source input to EBCDIC.

The principal print device subroutine is used to perform any printing required during the compilation.

FETCHING COMPILER PHASES

The ROL subroutine loaded into high-addressed storage as part of phase 1, obtains from each phase the word count and sector address of the next phase to be loaded. This subroutine then reads the next compiler phase into core storage and transfers control to it.

This subroutine also examines the Console Entry switches. If a request to dump is indicated in the switches, it calls the System Core Dump program via

the \$DUMP entry point in the Skeleton Supervisor to dump the statement string, the Symbol Table, and the FORTRAN Communications Area. At the completion of the dump the ROL subroutine regains control. It then fetches and transfers control to the next phase.

PHASE DESCRIPTIONS

PHASE 1

- Reads the control records; sets the corresponding indicators in the FORTRAN Communications Area.
- Reads the source statements; stores them in the String Area; precedes each statement with a partially completed ID word.
- Checks for a maximum of five continuation records per statement.
- Lists the source program, if requested.

Phase 1 is composed of two major segments; the first analyzes the control records and the second inputs the source statements. The control record analysis portion of phase 1 is loaded into the String Area, while the statement input portion is loaded at the normal Phase Area origin. The control record analysis subroutines, therefore, remain in storage until the processing of the control records is completed. They are then overlaid by the source statements as the string is built by the statement input portion.

Errors Detected

The errors detected by phase 1 are: 1 and 2.

PHASE 2

- Determines the statement type for each statement; inserts the type code into the statement ID word.
- Places the statement terminator character (semicolon) at the end of each statement.

- Converts subprogram names longer than five characters to five-character names.
- Converts FORTRAN-supplied subprogram names according to the specified precision.
- Generates the calls and parameters that initialize I/O subroutines during execution of the object program, if the IOCS control indicators are present.

According to the indicators set in the IOCS word (word 15) of the FORTRAN Communications Area by the previous phase, phase 2 generates the required calls to FORTRAN I/O. If the Unformatted Disk I/O Area indicator is on, a 'LIBF UFIO' followed by its parameter is inserted into the string. If the Disk indicator is on, a 'LIBF SDFIO' followed by its parameter is inserted into the statement string. If any other indicator in the IOCS word is on, phase 2 inserts the 'LIBF SFIO' followed by its parameters into the statement string. The table of device servicing subroutines (ISSs) is also built and inserted.

Phase 2, beginning with the first statement of the string, checks each statement in order to classify it into one of the 31 statement types. FORMAT statements, already having the type code, and compiler-generated error statements are not processed by this phase. Each statement name is compared to a table of valid FORTRAN statement names. Each recognized statement name is removed from the string and the corresponding ID type code is inserted into the statement ID word.

Arithmetic statements are detected by location of the equal sign (=) followed by an operator other than a comma. Because of this method of detection, arithmetic and statement function statements both carry the arithmetic statement ID type until phase 8, which distinguishes them.

Names within the statement body are converted into name code and stored. Names with only one or two characters are stored in one word.

Phase 2 converts all parentheses, commas, etc., into special operator codes. Each operator is stored in a separate word. Also, each arithmetic operator (+, -, /, *, **) is stored in a separate word, and a statement terminator character (semicolon) is placed after each statement, except for CONTINUE and FORMAT statements and compiler-generated error statements.

NOTE: The string words containing name or constant characters have a one in bit position 0. Bit

position 0 of string words containing arithmetic operator characters has a zero.

The standard FORTRAN-supplied subprogram names specified in the source program are changed, if necessary, to reflect the standard or extended precision option specified in the control records. Also, the six-character subprogram names of SLITET, OVERFL, and SSWTCH, which are allowed so as to be compatible with System/360 FORTRAN, are changed to SLITT, OVERF, and SSWTC, respectively.

The word FUNCTION appearing in a Type statement and the statement numbers of DO statements are isolated by the Classifier Phase. Isolation is accomplished by placing a one-word special operator (colon) just after the word or name to be isolated. This process aids later phases in detecting these words and numbers.

Errors Detected

The error detected by phase 2 is: 4.

PHASE 3

- Checks the subprogram and Specification statements for the proper order; removes any statement numbers from these statements.
- Checks to ensure that statements following IF, GO TO, CALL LINK, CALL EXIT, RETURN, and STOP statements have statement numbers.
- Removes CONTINUE statements that do not have statement numbers.
- Checks the statements for statement numbers; checks the Symbol Table for a previous entry of the same statement number.
- Places the statement number into the Symbol Table; places the address of the Symbol Table entry into the string.

Phase 3 makes two passes through the statement string. The first pass checks to ascertain the subprogram and Specification statements are in the following sequence:

SUBROUTINE or FUNCTION statement
Type statements (REAL, INTEGER)
EXTERNAL statements
DIMENSION statements
COMMON statements
EQUIVALENCE statements

A check is also made to ensure that all DATA and DEFINE FILE statements appear within the Specification statement group. Placement of these two statement types is optional; however, they must not be intermixed with EQUIVALENCE statements.

The SORF word (word 13) in the FORTRAN Communications Area is appropriately modified if a SUBROUTINE or FUNCTION statement is present.

The second pass of phase 3 scans the statement string for statements with statement numbers. Each unique statement number is placed into the Symbol Table and the address of the Symbol Table entry is placed into the string where the statement number previously resided.

All statements having statement numbers previously added to the Symbol Table (duplicates of other statement numbers) are in error.

Errors Detected

The errors detected by phase 3 are: 5, 6, and 9.

PHASE 4

- Places COMMON statement variables into the Symbol Table; includes dimension information in the Symbol Table entries, if present; removes the statement from the string.
- Checks for a SUBROUTINE or FUNCTION statement; places the names and dummy arguments of the SUBROUTINE or FUNCTION statement into the Symbol Table; deletes the statement from the statement string.
- Checks REAL and INTEGER statements for the word FUNCTION.

Phase 4 is a two-pass phase. The first pass processes COMMON statements; the second pass processes a SUBROUTINE or FUNCTION statement, including a FUNCTION designated in a REAL or INTEGER statement.

Pass 1 of phase 4 examines all COMMON statements, checking all variable names for validity. Valid, unique variable names found in COMMON statements are placed into the Symbol Table. Duplicate variable names are in error.

When dimension information is present in a COMMON statement, the Symbol Table entry for the dimensioned variable is expanded to six words, the dimension constants are changed to binary format, and this binary information is inserted into the Symbol Table entry. The Symbol Table ID word is updated to indicate the presence of the dimension information and the level of dimensioning.

When all the variables in a COMMON statement have been processed, it is removed from the string.

The second pass of phase 4 checks for a SUBROUTINE or FUNCTION statement among the Specification statements. If either is found, the SORF word (word 13) in the FORTRAN Communications Area is modified to indicate whichever is applicable. The subprogram name is checked for validity. If valid, the name is added to the Symbol Table and the address of the Symbol Table entry is placed into the FNAME words (words 11-12) in the FORTRAN Communications Area. The subprogram parameters are checked and, if valid, they are added to the Symbol Table and the statement is removed from the string.

The first REAL and INTEGER statements are examined for the presence of the word FUNCTION. If the FUNCTION specification is found, the REAL or INTEGER statement is processed in the same manner as a FUNCTION statement, except that the subprogram mode is specified explicitly by the statement type.

Errors Detected

The errors detected by phase 4 are: 7, 8, 10, 11, 12, 12, 13, 14, and 15.

PHASE 5

- Places DIMENSION statement variables into the Symbol Table; places dimension information into the Symbol Table entries; removes the statement from the string.
- Places variables and dimension information from REAL, INTEGER, and EXTERNAL statements into the Symbol Table.

- Indicates in the Symbol Table ID word the variable's mode (real or integer).
- Checks EXTERNAL statements for the names IFIX and FLOAT, which are not allowed.

The processing of phase 5 is done in two passes. The first pass analyzes DIMENSION statements. Each variable name found in a DIMENSION statement is first checked for validity. If the name is valid, the Symbol Table is searched for a duplicate. If no duplicate is found, the variable name, along with its dimension information, is added to the Symbol Table. If a duplicate is found that has not yet been dimensioned, the dimension information from the variable name is added to the existing Symbol Table entry. If a duplicate is found that has already been dimensioned, the variable name is in error.

In a subprogram compilation, a comparison is made to ensure that no variable name duplicates the subprogram name.

The second pass of phase 5 examines the REAL, INTEGER, and EXTERNAL statements found in the statement string. Each variable found in these types of statements is checked for validity. Valid variables are compared to the Symbol Table entries. Those variables duplicated in the Symbol Table as the result of prior COMMON or DIMENSION statement entries are in error. Those not equated to Symbol Table entries are added to the Symbol Table in the same manner as in the first pass of this phase.

Errors Detected

The errors detected by phase 5 are: 7, 8, 15, 17, 18, 19, 20, 21, and 22.

PHASE 6

- Scans all IF, CALL, and arithmetic statements for valid real constants.
- Converts real constants to standard or extended precision format, as specified.

Each valid real constant encountered in an arithmetic, IF, or CALL statement is converted to binary in the precision indicated by the FORTRAN Communications Area indicators. The Symbol Table is checked for a previous entry of the constant. If a previous

entry is found, no new entry is made. The constant operator, a special code indicating that the following word is the Symbol Table address of a constant, followed by the Symbol Table address of the constant already entered is inserted into the statement string in place of the constant.

If no previous entry in the Symbol Table is found, the converted constant is added to the Symbol Table. The constant operator along with the Symbol Table address replaces the constant in the statement string. The statement string is closed up following the alteration of the string.

Errors Detected

The following errors are detected by phase 6: 23 and 50.

PHASE 7

- Checks the syntax of DEFINE FILE, CALL EXIT, and CALL LINK statements.
- Determines the defined file specifications.

All variable names in DEFINE FILE, CALL LINK, and CALL EXIT statements are checked for validity and are added to the Symbol Table. All valid constants are converted to binary and are added to the Symbol Table.

Phase 7 checks to ensure that a DEFINE FILE statement does not appear in a subprogram.

This phase computes the file definition specifications, that is, a DEFINE FILE Table consisting of one entry for each unique file. Each entry consists of the file number, the number of records per file, the record length, the associated variable, a blank word for insertion of the file's sector address at the time the program is loaded for execution, the number of records per sector, and the number of disk blocks per file. A count is kept in the DFCNT word (word 16) in the FORTRAN Communications Area of the number of files defined.

Errors Detected

The errors detected by phase 7 are: 3, 70, 71, 72, and 73.

PHASE 8

- Places variables and integer constants into the Symbol Table.
- Places parameters from statement function statements into the Symbol Table.
- Replaces operators with pointers to the Forcing Table to be used in phase 16.
- Converts the left parenthesis of subscripts to a special dimension indicator.

Phase 8 checks the variable names found in the statement string for validity. Valid variables are added to the Symbol Table. A second check is made to ensure that all variable names conform to the implicit or explicit mode specifications (real and integer). Integer constants are also added to the Symbol Table, provided they are unique. However, names and integer constants that are found in subscript expressions are not added to the Symbol Table until a later phase.

When adding names and constants to the Symbol Table, phase 8 replaces them in the string by the address of their respective Symbol Table entries. The address replacing a constant or name is the address of the ID word of the Symbol Table entry for that constant or name.

Internal statement numbers are located in the Symbol Table and are replaced by the address of their corresponding Symbol Table entries; those statement numbers not found in the Symbol Table (i. e., not previously entered) are in error.

Phase 8 changes the ID word of the statement function statement, until now identical to that of an arithmetic statement, to the statement function type. Also, the statement function name and the parameters of statement functions are added to the Symbol Table. These entries in the Symbol Table are distinguished by their lack of a sign bit in the second word of the name.

During phase 8, the left parenthesis on subscripts is changed to a special left parenthesis operator that indicates the order of the dimension that follows.

This phase also converts all operators, except those in subscript expressions, from the 6-bit EBCDIC representation to a pointer value. This pointer value is derived from the Forcing Table. The conversion is done in preparation for the Scan Phase,

phase 16, when an arithmetic operational hierarchy will be determined through these pointer values.

Errors Detected

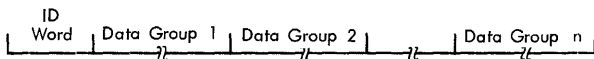
The following errors are detected in phase 8: 7, 24, 25, 26, and 43.

PHASE 9

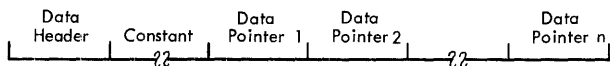
- Checks the DATA statement for correct syntax and valid variable references.
- Reformats the DATA statement into a string of data groups.

Each variable in the DATA statement is checked to ensure that it has been previously entered into the Symbol Table. A check is also made to ensure that a subscript expression for a DATA statement variable does not exceed the level of dimensioning indicated in the Symbol Table entry for the referenced variable.

Phases 9 converts the DATA statement into the following form:



Each data group has the following form:



Each data header has the following form:

<u>Bits</u>	<u>Contents</u>
0	0
1-12	Duplication factor
13-15	Length of the following constant

The constant may be one word in length (an integer), two or three words in length (a real number in standard or extended precision), or n words in length (a literal).

Each data pointer has the following form:

<u>Word</u>	<u>Bit</u>	<u>Contents</u>
1	0	1
	1	0, if no displacement word follows 1, if a displacement word follows
2	0	0, non-externally subscripted variable
	1	1, externally subscripted variable
3-4		Zeros
5-15		Symbol Table address of the variable
2	0-15	Displacement word (present only if variable is subscripted)

A displacement word follows data pointers to subscripted variables; its contents are the adjusted subscript offset.

The statement terminator is removed from the DATA statement in phase 9.

Errors Detected

Phase 9 detects the following errors: 75, 76, 77, 78, 79, 80, and 82.

PHASE 10

- Converts FORMAT statements into a chain of format specifications for interpretation by the FORTRAN I/O subroutines.
- Converts the Apostrophe (') type format to H type.

In decomposing the FORMAT statement, phase 10 converts each format type into a format specification (see Table 6). Where required, the Field Repeat, Group Repeat, and REDO counts are computed and inserted. At the completion of phase 10, the FORMAT statement is simply a chain of format specifications.

The conversion of a FORMAT statement is shown below:

```

999 FORMAT (3HAP=F6.3,4(E10.4/I8),T1,' ANSWER IS',3HXY=F9.4)

```

ID
Word 999 H,3 A,P =,b F,3,6 E,4,10 / I,8 RR,4 -3 T,1 H,10 b,A N,S W,E R,b I,S H,3 X,Y =,b F,4,9 RR,16

Table 6. Conversion of FORTRAN FORMAT Specifications

Format Type (input)	Format Specification (output)			
	4 Bits	5 Bits	7 Bits	16 Bits
E	0000	DD	WW	
F	0001	DD	WW	
I	0010		WW	
A	0011		WW	
X	0100		WW	
H	0101		WW	
T	0110		CC	
/	0111	Undefined		
Group Repeat	1000		NO	RR ₁ }
Field Repeat	1001		NO	
	1010	Not Used		
REDO	1011		RR	

DD (decimal width)	Maximum = 127 (used only in E and F type formats).	NO (number)	Positive count of the number of repetitions to be made of a field or group.
WW (total field width)	Maximum = 127 in E and F type formats, 145 in I, A, X, and H type formats.	RR ₁ (group repeat)	Negative count of the number of words back to the first specification of a group to be repeated.
CC (carriage control)	Positive count of the number of character positions to be skipped.	RR (REDO)	Positive count of the number of words back to the rightmost left parenthesis in the statement.

Errors Detected

The following errors are detected in phase 10: 27, 28, 29, and 30.

PHASE 11

- Calculates the constants needed for object program subscript computation.

- Sets up dummy arguments for the insertion of variables in the object coding.

Phase 11 bypasses all FORMAT, CONTINUE, and compiler-generated error statements. All other statements are scanned but only those statements that contain the special left parenthesis operator inserted by phase 8 are operated upon.

The subscripting information for each variable is checked for validity.

Phase 11 then calculates the subscript constant D_4 and, depending on the dimensioning level, the constants D_1 , D_2 , and D_3 . (See below for the method of derivation of these constants.)

These subscript constants are inserted into the subscript expression with the subscript indices. The right and left parenthesis enclosing the subscript expression are then changed to special operators to be used in a later phase.

Calculation of the Subscript Constants

Assuming the maximum subscript form

$$c^{*v+c'}$$

where

v represents an unsigned, nonsubscripted, integer variable and c and c' represent unsigned integer constants,

phase 11 computes the subscript constants (D-factors) as follows:

For a 1-dimension array --

$$A(C_1 * I + C_2)$$

$$D_1 = C_1 * S$$

$$D_4 = (C_2 - 1) * S$$

For a 2-dimension array --

$$A(C_1 * I + C_2, C_3 * J + C_4)$$

$$D_1 = C_1 * S$$

$$D_2 = L * C_3 * S$$

$$D_4 = [(C_2 - 1) + L * (C_4 - 1)] * S$$

For a 3-dimension array --

$$A(C_1 * I + C_2, C_3 * J + C_4, C_5 * K + C_6)$$

$$D_1 = C_1 * S$$

$$D_2 = L * C_3 * S$$

$$D_3 = L * M * C_5 * S$$

$$D_4 = [(C_2 - 1) + L * (C_4 - 1) + L * M * (C_6 - 1)] * S$$

In the above formulas,

L = first dimension factor

M = second dimension factor

S = size in words of the array entries

= 1 for one-word integers

= 2 for standard precision

= 3 for extended precision

C_1 and C_2 = constants in the first dimension value

C_3 and C_4 = constants in the second dimension value

C_5 and C_6 = constants in the third dimension value

I , J , and K are the subscript indices.

Errors Detected

The following errors are detected in phase 11: 31, 32, 33, 34, and 35.

PHASE 12

- Checks the syntax of arithmetic, IF, CALL, and statement function statements.
- Checks statement function calls, including nested calls, for valid names and the correct number of parameters.
- Checks for the definition of variables; checks for valid statement number references in IF statements.

The syntax of all CALL statements is checked. A call operator is inserted between the subprogram name and its dummy arguments for use in the Scan Phase, phase 16.

During the analysis of statement function statements a table is built containing the statement function name and the number of parameters associated with

that function. This table is used in analyzing statement function calls, including nested calls, to check for the proper number of parameters.

The syntax of the record number expression in Disk READ/WRITE statements is checked. The right parenthesis is changed to a colon operator which facilitates the scan of the Disk READ/WRITE statement in the following phase.

Errors Detected

The following errors are detected in phase 12: 36, 37, 38, 39, 40, 41, 42, and 43.

PHASE 13

- Checks FIND, READ, WRITE, and GO TO statements for correct syntax, valid FORMAT statement references, and valid variables.
- Detects implied DO loops in READ and WRITE statements; generates the indicators necessary for later processing of the DO loop.

When READ and WRITE statements are encountered in a mainline program, a check is made for the presence of IOCS indicators in the FORTRAN Communications Area. All READ and WRITE statements in a SUBROUTINE or FUNCTION subprogram do not require the presence of IOCS indicators.

READ, FIND, and WRITE statements are checked for valid variables and for proper syntax. READ and WRITE statements are checked for a valid FORMAT statement reference. Disk READ and WRITE statements are differentiated by means of the apostrophe (') separating the file number and record number parameters. The appropriate disk or non-disk I/O operator is generated for and inserted into each READ or WRITE statement.

READ and WRITE statements are also checked for implied DO loops. The necessary DO initialize and DO test operators are generated and inserted into the statement body.

Errors Detected

The following errors are detected in phase 13: 43, 44, 45, 46, 47, 48, 49, 50, and 68.

PHASE 14

- Checks for valid syntax in DO statements and in nested DO loops.
- Generates and inserts at the appropriate points the coding needed to perform the DO test.
- Checks the syntax of DO, CONTINUE, BACKSPACE, REWIND, END FILE, STOP, PAUSE, and END statements.
- Checks for a GO TO, IF, STOP, CALL LINK, CALL EXIT, or RETURN statement as the last executable statement of the source program.

BACKSPACE, END FILE, and REWIND statements are checked for valid unit addresses. Valid unit addresses are placed into the Symbol Table as integer constants. BACKSPACE, END FILE, and REWIND statements are then replaced on the statement string by a generated LIBF followed by the Symbol Table address of the unit address. This Symbol Table address becomes an argument to the LIBF.

Statements which follow STOP statements are checked to ensure that they are numbered statements. All integers found in PAUSE and STOP statements are checked to ensure that they are not greater than 9999. Valid integers are added to the Symbol Table as integer constants.

As the DO statements are analyzed for correct syntax, phase 14 constructs a DO Table in the following format:

<u>Word</u>	<u>Contents</u>
1	Index
2	DO test statement number or Generated Label
3	Test value
4	Increment
5	DO range statement number

A DO Table entry is made for each DO statement when it is detected. As the statements following the DO statement are scanned, the statement numbers are compared with the contents of word 5, the range limit, of the DO Table entries. When the range limit is found the DO test coding is inserted into the statement string.

The DO Table is built from low-to-high-addressed storage. It is scanned, however, from high-to-low-addressed storage. In this manner, nested DO loops that violate range limits are detected.

Errors Detected

The following errors are detected in phase 14: 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, and 62.

PHASE 15

- Scans READ, WRITE, IF, CALL, and arithmetic statements for subscript expressions.
- Optimizes subscript calculation by means of the Subscript Expression Table.
- Generates SGTs (Subscript-generated temporary storage locations) as necessary.

Each unique subscript expression is placed into a table called the Subscript Expression Table. Each entry in this table appears as follows:

<u>Word</u>	<u>Contents</u>
1	D ₄
2	I, first dimension subscript index
3	D ₁
4	J, second dimension subscript index
5	D ₂
6	K, third dimension subscript index
7	D ₃
8	/8010, if entry is not used /0010, if entry is used on this statement /0000, if entry was used on a previous statement

Also, each unique variable of a subscript expression is placed into a table called the Bound Variable Table.

As each subscript expression is entered into the Subscript Expression Table, it is removed from the string. The subscripted variable is then tagged with SGT indicator bits pointing to the Subscript Expression Table entry.

An SGT (Subscript-generated temporary storage location) is generated for each entry made to the Subscript Expression Table. The SGT is placed into an SGT Table. The SGT is also placed into the Symbol Table and the address of the Symbol Table entry is inserted into the statement in the string.

For each subscript expression encountered, a scan is made of the Bound Variable Table. If one or more of the variables in the subscript expression are not located in the Bound Variable Table, the subscript must be recalculated. Thus, a unique entry is made to the Subscript Expression Table for the expression and an associated SGT is generated.

If, however, all the variables of the subscript expression are located in the Bound Variable Table, the Subscript Expression Table is then scanned to determine if a duplicate subscript expression is already located in the table. If no equivalent is found, the subscript expression is added to the table as a unique entry and an associated SGT is generated.

If a duplicate expression is found in the Subscript Expression Table, the subscript expression is removed from the string and is replaced by a pointer to its duplicate in the Subscript Expression Table. Thus, identical subscript expressions share the same indices of a common entry in the Subscript Expression Table and the same SGT.

Whenever a variable is assigned a new value (i.e., appears to the left of the equal sign in an arithmetic expression, in the argument list of a subprogram, etc.), that variable, if found in the Bound Variable Table, is removed from the table. This removal from the Bound Variable Table causes all entries in the Subscript Expression Table containing that variable to be removed. The associated SGTs are also removed from the SGT Table but remain in the Symbol Table. The addresses in the string of the SGTs in the Symbol Table also remain.

If a statement is encountered containing a subscript expression and having a statement number that is referenced by some other statement, the entire Subscript Expression Table is cleared and all subscripts,

beginning with the subscript expression in the reference statement, must be recalculated.

The Subscript Expression Table is also cleared whenever a DO statement is encountered. Following subscripts must be recalculated. Implied DO loops, as in READ and WRITE statements, cause only those entries involving the index of the implied DO to be cleared. Only the subscripts involving that index must be recalculated.

Errors Detected

The following error is detected in phase 15: 63.

PHASE 16

- Converts all FIND, READ, WRITE, IF, GO TO, CALL, statement function, and arithmetic statements into a modified form of Polish notation.
- Establishes the order of arithmetic operational performance.
- Sets up the arguments for subroutine calls to be generated.

Phase 16 converts all READ, WRITE, GO TO, arithmetic, statement function, CALL, and IF statements to a modified form of Polish notation. This conversion is accomplished through the use of a Forcing Table, strings, and an Interpreter.

The Forcing Table is a table of 2-word entries. The first word contains the left and right forcing values for each operator. The first 8 bits constitute the left forcing value and the last 8 bits, the right forcing value. The second word of the 2-word entry contains the address of the string to be used by the Interpreter when the corresponding operator is forced. (See Table 7.)

The string address (word 2 of each Forcing Table entry) for each operator is a pointer used by the Interpreter. This pointer designates to the Interpreter a string of operations that must be performed by the Interpreter in order to convert the forced operator and its operands to the modified form of Polish notation.

NOTE: Strings may also contain pointers. These pointers designate substrings, which are detailed

Table 7. FORTRAN Forcing Table

Operator	Definition	Forcing Value		String Pointer
		Left	Right	
N	Name Operator	63	00	0
)	Normal Right Parenthesis	01	32	0
;	Statement Terminator	3C	3C	0
+,-	Add, Subtract	0A	0A	1
/,*	Divide, Multiply	05	05	1
↑	Exponentiation	05	04	1
←	Assign	3B	3C	1
(Normal Left Parenthesis	31	01	2
,	Comma	31	30	3
Ⓢ	Call Operator	01	01	4
(0), (1), (2), (3)	Special Parenthesis for Literals, Special Parenthesis for Dimensioned Arrays	31	01	6
U-	Unary Minus	0A	0A	7
RG	Range Operator	31	01	8
SKP	Special Right Parenthesis in Implied DOs	31	01	9
Ⓢ	Subscripting Right Parenthesis in Implied DOs	31	01	10
I/O	I/O Operator before Scan	30	01	11
IO	I/O Operator during and after Scan	30	01	12
DOA	Equal Sign in Implied DOs	31	01	13

extensions of the string and which are used by the Interpreter in the same manner as the strings.

A forcing condition exists if the forcing value of the right operator is equal to or greater than the forcing value of the left operator. This condition results in the left operator being forced; that is, the operation to the left has precedence. Whenever a non-forcing condition exists, the operands and operators involved remain in the statement. Operands and operators are removed from the string only when an operator is forced and the Interpreter-generated symbol FAC replaces them.

The Interpreter controls the conversion of the statement to the modified form of Polish notation. As each operator is forced, the Interpreter, using the string address from the Forcing Table, selects the

associated string and performs the string operations. These operations result in the output of the forced operator and its operands, resequenced in the order of operational performance. The forced operator and its operands are put out into the output buffer by the Interpreter and are replaced in the statement body by the symbol FAC.

The scan begins with the string pointer moving from left to right. When an operator is encountered, the scan looks two words to the left for a second operator. If none is found, the string pointer moves to the right, one word at a time, in search of another operator. If an operator is found to the left, the scan converts the left and right operators to their respective forcing values and checks for the forcing condition.

If a forcing condition does not exist, the scan again resumes, moving the pointer to the right. If, however, a forcing condition does exist, the Interpreter handles the operator and operands involved.

Upon return to the scanning process, the string pointer is positioned to the same operator that caused the previous force, the symbol FAC resides one word to the left in place of the forced operator and its operands, and a new operator resides two words to the left. These operators are then converted and the check is made for the forcing condition.

If at any time the symbol FAC is an operand of a forced operator, FAC is replaced by a GT (generated temporary storage location). The GT is then outputted as the operand in place of FAC. FAC again replaces the forced operator and operands in the statement body. New GTs are created as they are needed in order to maintain FAC in the statement body.

At the completion of the scan process the statement body has been reduced to the symbol FAC; the statement body now consists of less than four words. The output buffer contains the entire statement converted to the modified form of Polish notation.

If in looking for a left operator the scan must bypass the argument list of a call operator, the elements of this argument list are stored temporarily in a special buffer called the Push-down List. When the call operator is forced and placed into the output buffer, the Push-down List is then emptied into the output buffer in reverse order so that the arguments are restored in their original sequence following the call operator.

When the scan detects that the statement consists of less than four words (the symbol FAC only), the output buffer is placed into the statement string overlaying the symbol FAC, and the scan moves to the next statement.

The statement terminator (semicolon) serves as an operator that is scanned as any other operator.

Figure 16 illustrates the scanning process.

Errors Detected

The following error is detected in phase 16: 64.

PHASE 17

- Replaces FIND, READ, WRITE, GO TO, and RETURN statements with compiler-generated coding.
- Replaces those parts of arithmetic, IF, CALL, and statement function statements that involve subscripting of variables with compiler-generated coding.
- Checks subprograms for a RETURN statement; generates the return linkage coding.

Phase 17 replaces READ, WRITE, and FIND statements by a call to the appropriate I/O subroutine, along with the necessary arguments. Generated labels are added to READ and WRITE statements involving implied DO loops.

Also, statement function, arithmetic, IF, and CALL statements are examined for subscripted variables. Those parts of these statements that involve subscripts are replaced by compiler-generated coding.

In order to produce more efficient coding, phase 17, by means of an SGT Table, eliminates redundant loads of the same subscript offset. Also, the instructions used to load literal subscripts are placed immediately before the indexed operations.

Errors Detected

The following error is detected in phase 17: 69.

Arithmetic Statement
A = B + C * D - E

Step	Contents of the string	Contents of the Output Buffer	Comments
1	A = B+C*D-E ↑ P		A is not an operator, so the pointer, P, moves to the right. When scanning to the right for an operator, non-operators are simply skipped. When scanning for an operator to the left with which to force, non-operators are added to the push-down list.
2	A = B+C*D-E ↑ P		There is no operator to the left with which a forcing condition test can be made; therefore, the pointer moves to the right until an operator is encountered.
3	A = B+C*D-E; ↑ P		Using the forcing table, the pointer indicates the RV (right forcing value) and P - 2 (two positions to the left) indicates the LV (left forcing value). The RV of + is 0A; the LV of = is 3B. 0A is not equal to or greater than 3B. The left operator is not forced, and the pointer moves to the next operator.
4	A = B+C*D-E; ↑ P		The RV of * is 05; the LV of + is 0A. 05 is not equal to or greater than 0A. The left operator is not forced, and the pointer moves to the next operator.
5	A = B+C*D-E; ↑ P	*CD	The RV of - is 0A; the LV of * is 05. 0A is greater than 05. Hence, the left operator is forced. *CD is placed in the output buffer and the symbol FAC replaces the outputted operator and operands. The pointer is not moved; an attempt is made to force the new left operator.
6	A = B+FAC-E; ↑ P	*CD+B	The RV of - is 0A; the LV of + is 0A. 0A is equal to 0A. Hence, the left operator is forced. +B is added to the output buffer. The symbol FAC still replaces the contents of the output buffer. The pointer is not moved; an attempt is made to force the new left operator.
7	A = FAC-E; ↑ P	*CD+B	The RV of - is 0A; the LV of = is 3B. 0A is not equal to or greater than 3B. The left operator is not forced and the pointer moves to the next operator.
8	A = FAC-E; ↑ P	*CD+B-E	The RV of ; is 3C; the LV of - is 0A. 3C is greater than 0A. Hence, the left operator is forced. -E is added to the output buffer. The symbol FAC still replaces the contents of the output buffer. The pointer is not moved; an attempt is made to force the new left operator.
9	A = FAC; ↑ P	*CD+B-E=A	The RV of ; is 3C; the LV of = is 3B. 3C is greater than 3B. Hence, the left operator is forced. =A is added to the output buffer. The symbol FAC still replaces the contents of the output buffer.
10	FAC;	*CD+B-E=A	The original statement now consists of three words or less. This indicates that the statement has been scanned. The symbol FAC is now replaced by the contents of the output buffer.
11	*CD+B-E=A;		The scan is complete.

Figure 16. FORTRAN Scan Example

PHASE 18

- Replaces arithmetic, statement function, CALL, and IF statements not involving subscripted variables by compiler-generated coding.
- Completes the replacement of arithmetic, statement function, CALL, and IF statements that do involve subscripted variables by compiler-generated coding.
- Optimizes IF statement branch instructions.
- Handles mixed-mode arithmetic.

This phase generates the coding necessary to replace arithmetic, statement function, CALL, and IF statements. This phase wholly converts statements of these types that include no subscripted variables and merely completes the conversion, which was partially completed in phase 17, of statements of these types that do include subscripted variables.

Phase 18 generates the code to perform integer, real, and mixed-mode arithmetic. Where possible, integer arithmetic is done in-line. The remainder of the coding consists of calls to System Library subroutines, followed by argument lists.

As needed, calls to the FORTRAN-supplied FUNCTION subprograms IFIX and FLOAT are generated.

All calls to these subprograms are made LIBFs. However, calls to exponentiation subroutines generated by phase 18 are made CALLs.

GTs (generated temporary storage locations) detected in the string by this phase or generated by this phase for storing intermediate results of arithmetic calculations are made to agree in mode with the function of which they are a part.

Errors Detected

There are no errors detected in phase 18.

PHASE 19

- Fetches the principal print device subroutine for use by phases 19-24; also provides a print interface subroutine for these phases.
- Allocates storage for COMMON variables.
- Allocates all storage assignments aligned according to EQUIVALENCE statements.
- Assigns all allocations according to the specified precision of the program.
- Prints the allocations of the variables as they are assigned, if requested.

Phase 19 performs the allocation of the variables found in the Symbol Table; i. e., these variables are assigned storage addresses in the object coding. These addresses replace the Symbol Table entries for the variables.

The COMMON area resides in high-addressed storage during execution of the object program. Thus, all COMMON variables are assigned absolute addresses within this high-addressed area. The variable area resides in storage just below the object program during execution. All variables not in COMMON are assigned relative addresses within this area.

Phase 19 first allocates COMMON variables found in the Symbol Table. EQUIVALENCE statements that include COMMON variables are examined and the addresses are aligned during allocation to obtain an equivalence. A check is made to ensure that the equivalence does not cause a variable or array element to be allocated beyond the beginning of the COMMON area.

Variables that appear in EQUIVALENCE statements are allocated next, one combined equivalence nest at a time. The remaining variables in the Symbol Table are finally allocated, real variables first, followed by integer variables.

Phase 19 also computes the core requirements for constants after all defined variables have been allocated. The core requirements for variables and for COMMON are then stored in the FORTRAN Communications Area.

Errors Detected

The error detected by phase 19 is either: 65, 66, or 67.

PHASE 20

- Lists any errors that were detected during the compilation process.
- Rearranges the statement string if there were no errors detected.

Phase 20 deletes from the statement string EQUIVALENCE statements that do not have an error indicator and replaces EQUIVALENCE statements that have an error indicator by error statements.

The Symbol Table is scanned twice. The first scan detects unreferenced statement numbers and lists them on the principal print device. The second scan detects undefined variables and lists these also on the principal print device.

The statement string is rearranged (if there were no errors in the compilation) so that, if they are present, the DEFINE FILE Table, format specification strings, and arithmetic statement functions precede the first executable statement.

The statement string is scanned for error statements. Two counters are maintained during the scan. The first (STLAB) contains the statement number of the last numbered statement encountered. The second (STCNT) contains a count of the statements encountered since the last numbered statement. Compiler-generated statements and statement numbers are disregarded in these counts.

When an error statement is detected, these counters are inserted into the error message along with the error number for printing.

A check is made on all DATA statements. If a data constant is defined in COMMON, error 81 is indicated.

Errors Detected

The following error is detected in phase 20: 81.

PHASE 21

- Assigns the relative addresses to statement functions and numbered statements; inserts the allocations into the string.
- Creates the subroutine initialization call, if required.
- Calculates the core requirements of the program; stores the result in the FORTRAN Communications Area.
- Generates the statement function return linkage coding.

Phase 21 allocates relative addresses to all numbered statements and statement functions. The allocation is placed into the statement string entry, following the statement number or function label. A calculation of the object program storage requirements is made from the Location Counter at the end of allocation and stored in the SOFNS word (word 4) in the FORTRAN Communications Area.

If the program being compiled is a subprogram, this phase also creates the subroutine initialization call, CALL SUBIN, along with its dummy arguments; this subroutine directs the insertion of arguments during execution of the object program.

Errors Detected

There are no errors detected in this phase.

PHASE 22

- Inserts the statement allocations into the Symbol Table.
- Lists the statement allocations on the principal print device, if requested.

This phase scans the statement string for statement function statements and numbered statements. The

allocation found in each of these statements is entered in the Symbol Table. The allocation and the label are then deleted from the string entry.

Errors Detected

There are no errors detected in this phase.

PHASE 23

- Lists the Features Supported by the program as indicated in the FORTRAN Communications Area.
- Lists the System Library subroutines used by the program, if requested.
- Lists the subprogram names found in the Symbol Table, if requested.

Using the indicators in the CCWD word (word 14) in the FORTRAN Communications Area, phase 23 recreates the control records that were recognized in phase 1. These control records (with the exception of *IOCS) are listed on the principal print device under the title 'FEATURES SUPPORTED.'

According to the indicators in the CCWD word, phase 23 also alters (for purposes of printing only) the names of subprograms in the compiler-generated calls to reflect extended precision, if specified. (The actual compiler-generated coding is not altered until phase 26.)

If requested, a list is made of all the subprogram names that appear in the Symbol Table (all CALLs).

Phase 23 also scans the statement string, bypassing all one-word statements and tagging the names in the System Library table that are called by the program (all LIBFs). While scanning the System Library table, phase 23 checks the indicators in the IOCS word (word 15) in the FORTRAN Communications Area. A tag is added to the names of those subroutines that service the devices indicated in the IOCS word (i. e., the devices listed in the *IOCS control record). The names of the associated conversion subroutines (if required) are also tagged. The Subroutine table is then scanned and, if requested, the tagged System Library subroutine names are listed.

Errors Detected

There are no errors detected in this phase.

PHASE 24

- Lists the Core Requirements.
- Lists the constants and their addresses, if requested.

Under the heading 'CORE REQUIREMENTS,' phase 24 prints the amounts of storage used by the program, COMMON area, and variables. The program name is printed from the FNAME words (words 11-12) in the FORTRAN Communications Area.

If a list request is specified in the CCWD word (word 14), the real and integer constants are converted to output coding and listed with their relative addresses according to the specified precision. Real constants are listed first, followed by integer constants.

A check is made to see that the core requirements do not exceed 32767 words. If they do, the ERROR word (word 10) in the FORTRAN Communications Area is set and output to Working Storage is suppressed.

Errors Detected

There are no errors detected in this phase.

PHASE 25

- Builds the program header and data header records; places these records onto the disk in Working Storage.
- Places real and integer constants into Working Storage in absolute mode.

Phase 25 initially builds the program header and data header records. These records and the Buffer Communications Area carry information to phase 26. The program header and data header records are placed in Working Storage.

The statement string is searched for DEFINE FILE statements. These statements are analyzed and then

placed into Working Storage. The file specifications are outputted in absolute mode, except for the associated variable, which is in relocatable mode (the only relocatable constant). The statement string is also searched for DATA statements. All data constants are placed in Working Storage in absolute mode.

The Symbol Table is scanned twice. The first scan extracts real constants, computes their allocations, and inserts the allocations into the Symbol Table. The second scan performs the same operations for integer constants. All constants are placed into Working Storage in absolute mode.

Errors Detected

There are no errors detected in this phase.

PHASE 26

- Converts the compiled statement string to object coding.
- Places the object program into Working Storage.

According to the indicators in the CCWD word (word 14) in the FORTRAN Communications Area, phase 26 alters the subroutine names referenced by the compiled program to reflect, if necessary, extended precision as specified by the user. Phase 23 made the same conversion for listing purposes; this phase makes the conversion during the generation of the object program.

Phase 26 converts the statement string into the object program, which is written in Working Storage.

At the completion of the output, the termination subroutine (OUTER) inserts the necessary data into the FORTRAN Communications Area so that the Recovery Phase can complete the compilation.

Errors Detected

There are no errors detected in this phase.

PHASE 27

- Sets up the switches and parameters needed by the Monitor Control Record Analyzer to assume control.

This phase is the means by which the compiler returns control to the monitor system. The phase is entered under the following conditions:

1. Normal end of compilation, with or without program errors.
2. Disk work area exceeded.
3. Control record trap during input phase.

In each case the Recovery Phase sets indicators in the FORTRAN Communications Area in order to inform the monitor system program called next as to the results of the compilation. Compilation errors, the exceeding of Working Storage, and the trapping of

a monitor control record all cause the compilation output to be suppressed, the non-execute switch (\$NSEQ) to be set, and the non-DUP switch (\$NDUP) to be set.

If the compilation is successful, the program length, the number of disk blocks used to store the program, and the execution address are all transmitted to DCOM on the master cartridge and to DCOM of the Working Storage cartridge if it is other than the master cartridge.

Errors Detected

There are no errors detected in this phase.

FLOWCHARTS

Write Cartridge ID (ID):	UTL01
Fetch Phase IDs From SLET (FSLEN):	UTL02
Fetch System Subroutine (FSYSU):	UTL02
Write Working Storage Addresses (ADRWS):	UTL03
Initialize Disk Cartridge (DISC):	UTL04
Read *ID Record and Convert (RDREC):	UTL05
Print Cartridge ID (IDENT):	UTL06
Call System Print Subroutine (CALPR):	UTL07
Copy Disk Cartridges (COPY):	UTL08
Delete Core Image Buffer (DLCIB):	UTL09
Dump SLET (DSLET):	UTL10
Maintenance Program (MODIF):	UTL11- UTL13
SCAT2, Call Processing:	SCA01
SCAT2, Interrupt Processing:	SCA02- SCA03
SCAT3, Call Processing:	SCA04
SCAT3, Interrupt Processing:	SCA05- SCA07

FORTTRAN Non-disk I/O (SFIO):	FIO01- FIO03
CARDZ:	FIO04- FIO05
PRNTZ:	FIO06
PAPTZ:	FIO07
READZ:	FIO08
WRTYZ:	FIO09
PRNZ:	FIO10
PNCHZ:	FIO11
TYPEZ:	FIO12
HOLEZ:	FIO13

The System Library consists of (1) a complete library of input/output (except disk I/O), data conversion, arithmetic, and function subprograms, (2) selective dump subroutines, and (3) special programs for disk maintenance.

The following is a list of the contents of the System Library.

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
<u>Utility</u>					
Selective Dump on Console Printer	DMTD0, DMTX0	4	0	CALL	U5B00010
Selective Dump on 1132 Printer	DMPD1, DMPX1	4	0	CALL	U5C00010
Dump 80 Subroutine	DMP80	4	0	CALL	U5A00010
<u>Common FORTRAN</u>					
Test Console Entry Switches	DATSW	4	8	CALL	T3F00010
Divide Check Test	DVCHK	4	8	CALL	T3G00010
Functional Error Test	FCTST	4	8	CALL	T3H00010
Overflow Test	OVERF	4	8	CALL	T3J00010
Sense Light Control and Test	SLITE, SLITT	4	8	CALL	T3L00010
FORTTRAN Trace Stop	TSTOP	4	8	CALL	T3M00010
FORTTRAN Trace Start	TSTRT	4	8	CALL	T3N00010
Selective Dump	PDUMP	4	0	CALL	T3K00010
<u>FORTTRAN Sign Transfer</u>					
Extended Precision Transfer of Sign	ESIGN	4	8	CALL	S2F00010
Standard Precision Transfer of Sign	FSIGN	4	8	CALL	R2F00010
Integer Transfer of Sign	ISIGN	4	8	CALL	T3I00010
<u>Extended Precision Arithmetic/Function</u>					
Extended Precision Hyperbolic Tangent	ETANH, ETNH	4	8	CALL	S2I00010
Extended Precision A**B Function	EAXB, EAXBX	4	8	CALL	S2C00010

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
<u>Extended Precision Arithmetic/Function (Cont'd)</u>					
Extended Precision Natural Logarithm	ELN, EALOG	4	8	CALL	S2E00010
Extended Precision Exponential	EEXP, EXPN	4	8	CALL	S2D00010
Extended Precision Square Root	ESQR, ESQRT	4	8	CALL	S2H00010
Extended Precision Sine-Cosine	ESIN, ESINE, ECOS, ECOSN	4	8	CALL	S2G00010
Extended Precision Arctangent	EATN, EATAN	4	8	CALL	S2B00010
Extended Precision Absolute Value Function	EABS, EAVL	4	8	CALL	S2A00010
<u>Standard Precision Arithmetic/Function</u>					
Standard Precision Hyperbolic Tangent	FTANH, FTNH	4	8	CALL	R2I00010
Standard Precision A**B Function	FAXB, FAXBX	4	8	CALL	R2C00010
Standard Precision Natural Logarithm	FLN, FALOG	4	8	CALL	R2E00010
Standard Precision Exponential	FEXP, FXPN	4	8	CALL	R2D00010
Standard Precision Square Root	FSQR, FSQRT	4	8	CALL	R2H00010
Standard Precision Sine-Cosine	FSIN, FSINE, FCOS, FCOSN	4	8	CALL	R2G00010
Standard Precision Arctangent	FATN, FATAN	4	8	CALL	R2B00010
Standard Precision Absolute Value Function	FABS, FAVL	4	8	CALL	R2A00010
<u>Common Arithmetic/Function</u>					
Fixed Point (Fractional) Square Root	XSQR	4	8	CALL	T1C00010
Integer Absolute Function	IABS	4	8	CALL	T1B00010
Floating Binary/EBCDIC Decimal Conversions	FBTD (binary to decimal), FDTB (decimal to binary)	4	0	CALL	T1A00010
<u>System</u>					
LOCAL/SOCAL Flipper	FLIPR	3	0	-	U5D00010
DCOM Update	SYSUP	4	0	-	U5E00010
<u>FORTTRAN Trace</u>					
Extended Floating Variable Trace	SEAR, SEARX	3	0	LIBF	S2J00010
Fixed Variable Trace	SIAR, SIARX	3	0	LIBF	T6B00010
Standard Floating IF Trace	SFIF	3	0	LIBF	R2K00010
Extended Floating IF Trace	SEIF	3	0	LIBF	S2K00010
Fixed IF Trace	SIIF	3	0	LIBF	T6C00010
Standard Floating Variable Trace	SFAR, SFARX	3	0	LIBF	R2J00010
GOTO Trace	SGOTO	3	0	LIBF	T6A00010
<u>FORTTRAN I/O</u>					
Non-Disk Formatted FORTTRAN I/O	SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX	3	3	LIBF	T4C00010
FORTTRAN Find	SDFND	3	1	LIBF	T4B00010

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
Disk FORTRAN I/O	SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDF, SDI, SDIX, SDFX, SDAI	3	1	LIBF	T4A00010
Unformatted FORTRAN I/O	UFIO	3	1	LIBF	T4D00010
<u>Common FORTRAN</u>					
FORTTRAN Pause	PAUSE	3	2	LIBF	T2A00010
FORTTRAN Stop	STOP	3	2	LIBF	T2B00010
FORTTRAN Subscript Displacement Calculation	SUBSC	3	0	LIBF	T2D00010
FORTTRAN Subroutine Initialization	SUBIN	3	0	LIBF	T2C00010
FORTTRAN Trace Test and Set	TTEST, TSET	3	0	LIBF	T2E00010
<u>FORTTRAN I/O and Conversion</u>					
FORTTRAN Card 1442 (Read/Punch)	CARDZ	5	3	LIBF	T5A00010
FORTTRAN Card 1442-5 (Punch)	PNCHZ	5	3	LIBF	T5G00010
FORTTRAN Card 2501 (Read)	READZ	5	3	LIBF	T5J00010
FORTTRAN Paper Tape	PAPTZ	5	3	LIBF	T5F00010
FORTTRAN 1132 Printer	PRNTZ	5	3	LIBF	T5H00010
FORTTRAN 1403 Printer	PRNZ	5	3	LIBF	T5I00010
FORTTRAN Keyboard/Typewriter	TYPEZ	5	3	LIBF	T5K00010
FORTTRAN Typewriter	WRTYZ	5	3	LIBF	T5L00010
FORTTRAN Hollerith to EBCDIC Conversion	HOLEZ	3	3	LIBF	T5D00010
FORTTRAN Get Address Subroutine	GETAD	3	3	LIBF	T5C00010
FORTTRAN EBCDIC Table	EBCTB	3	3	-	T5B00010
FORTTRAN Hollerith Table	HOLTB	3	3	-	T5E00010
<u>Extended Precision Arithmetic/Function</u>					
Extended Precision Get Parameter	EGETP	3	2	LIBF	S1E00010
Extended Precision A**I Function	EAXI, EAXIX	3	2	LIBF	S1B00010
Extended Precision Divide	EDVR, EDVRX	3	2	LIBF	S1D00010
Extended Precision Float Divide	EDIV, EDIVX	3	2	LIBF	S1C00010
Extended Precision Float Multiply	EMPY, EMPYX	3	2	LIBF	S1G00010
Extended Precision Subtract Reverse	ESBR, ESBRX	3	2	LIBF	S1H00010
Extended Add-Subtract	EADD, ESUB, EADDX, ESUBX	3	2	LIBF	S1A00010
Extended Load-Store	ELD, ELDX, ESTO, ESTOX	3	0	LIBF	S1F00010
<u>Standard Precision Arithmetic/Function</u>					
Standard Precision Get Parameter	FGETP	3	2	LIBF	R1E00010
Standard Precision A**I Function	FAXI, FAXIX	3	2	LIBF	R1B00010
Standard Precision Divide Reverse	FDVR, FDVRX	3	2	LIBF	R1D00010
Standard Precision Float Divide	FDIV, FDIVX	3	2	LIBF	R1C00010
Standard Precision Float Multiply	FMPY, FMPYX	3	2	LIBF	R1G00010
Standard Precision Subtract Reverse	FSBR, FSBRX	3	2	LIBF	R1H00010

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
<u>Standard Precision Arithmetic/Function (Cont'd)</u>					
Standard Add/Subtract	FADD, FSUB, FADDX, FSUBX	3	2	LIBF	R1A00010
Standard Load/Store	FLD, FLDX, FSTO, FSTOX	3	0	LIBF	R1F00010
Standard Precision Fraction Multiply	XMDS	3	2	LIBF	S3I00010
<u>Common Arithmetic/Function</u>					
Fixed Point (Fraction) Double Divide	XDD	3	2	LIBF	S3G00010
Fixed Point (Fraction) Double Multiply	XMD	3	2	LIBF	S3H00010
Sign Reversal Function	SNR	3	2	LIBF	S3F00010
Integer to Floating Point Function	FLOAT	3	0	LIBF	S3C00010
Floating Point to Integer Function	IFIX	3	0	LIBF	S3D00010
I**J Integer Function	FIXI, FIXIX	3	2	LIBF	S3B00010
Normalize	NORM	3	0	LIBF	S3E00010
Floating Accumulator Range Check	FARC	3	2	LIBF	S3A00010
<u>Interrupt Service</u>					
Card Input/Output (No Error Parameter)	CARD0	5	0	LIBF	U2A00010
Card Input/Output (Error Parameter)	CARD1	5	0	LIBF	U2B00010
Disk Input/Output (No Preoperative Parameter Checking)	DISKZ*	-	-	Special	-
Disk Input/Output (No Simultaneity)	DISK1*	-	-	LIBF	-
High-Speed Disk Input/Output (Simultaneity)	DISKN*	-	-	LIBF	-
Paper Tape Input/Output	PAPT1	5	0	LIBF	U2D00010
Single Frame Paper Tape Input/Output	PAPTX	5	0	LIBF	U2F00010
Simultaneous Paper Tape Input/Output	PAPTn	5	0	LIBF	U2E00010
Plotter Output	PLOT1	5	0	LIBF	U2G00010
1132 Printer Output	PRNT1	5	0	LIBF	U2J00010
1403 Printer Output	PRNT3	5	0	LIBF	U2K00010
Keyboard/Console Printer Input/Output	TYPE0	5	0	LIBF	U2N00010
Console Printer Output	WRTY0	5	0	LIBF	U2O00010
1231 Optical Mark Page Reader	OMPR1	5	0	LIBF	U2C00010
2501 Card Input (No Error Parameter)	READ0	5	0	LIBF	U2L00010
2501 Card Input (Error Parameter)	READ1	5	0	LIBF	U2M00010
1442 Card Output (No Error Parameter)	PNCH0	5	0	LIBF	U2H00010
1442 Card Output (Error Parameter)	PNCH1	5	0	LIBF	U2I00010

*Note: Whereas DISKZ, DISK1, and DISKN are not strictly ISS subroutines (they are stored in the System Area by the System Loader), they are included in this list because they possess many characteristics of ISS subroutines.

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
<u>Conversion</u>					
16 bits to 6 Decimal Characters (Card Code)	BINDC	3	0	LIBF	U4B00010
32 bits to 11 Decimal Characters	BIDEC	3	0	LIBF	U4A00010
16 Bits to 4 Hexadecimal Characters (Card Code)	BINHX	3	0	LIBF	U4C00010
6 Decimal Characters (Card Code) to 16 bits	DCBIN	3	0	LIBF	U4G00010
11 Decimal Characters to 32 bits	DECBI	3	0	LIBF	U4H00010
EBCDIC to Console Printer Output Code	EBPRT	3	0	LIBF	U3A00010
Card Code to EBCDIC, EBCDIC to Card Code	HOLEB	3	0	LIBF	U3B00010
Card Code to Console Printer Output Code	HOLPR	3	0	LIBF	U3C00010
4 Hexadecimal Characters (Card Code) to 16 bits	HXBIN	3	0	LIBF	U3D00010
PTTC/8 to EBCDIC, EBCDIC to PTTC/8	PAPEB	3	0	LIBF	U3E00010
PTTC/8 to Card Code, Card Code to PTTC/8	PAPHL	3	0	LIBF	U3F00010
PTTC/8 to Console Printer Output Code	PAPPR	3	0	LIBF	U3G00010
Card Code to EBCDIC, EBCDIC to Card Code	SPEED	3	0	LIBF	U3H00010
Fast multi-purpose conversion	ZIPCO	3	0	LIBF	U3I00010
<u>Conversion Tables</u>					
EBCDIC and PTTC/8 Table	EBPA	3	0	-	U4K00010
Card Code Table	HOLL	3	0	-	U4P00010
Console Printer Output Code Table	PRTY	3	0	-	U4Q00010
EBCDIC to IBM Card Code	EBHOL	3	0	-	U4J00010
1403 Code to Console Printer Code	PT3CP	3	0	-	U4R00010
Console Printer Code to 1403 Code	CPPT3	3	0	-	U4F00010
1403 Code to EBCDIC	PT3EB	3	0	-	U4S00010
EBCDIC to 1403 Code	EBPT3	3	0	-	U4L00010
IBM Card Code to Console Printer Code	HOLCP	3	0	-	U4O00010
Console Printer Code to IBM Card Code	CPHOL	3	0	-	U4E00010
Console Printer Code to EBCDIC	CPEBC	3	0	-	U4D00010
EBCDIC to Console Printer Code	EBCCP	3	0	-	U4I00010
PTTC/8 Code to IBM Card Code	PTHOL	3	0	-	U4T00010
IBM Card Code to EBCDIC	HLEBC	3	0	-	U4M00010
IBM Card Code to 1403 Printer Code	HLPT3	3	0	-	U4N00010
<u>SCA Subroutines</u>					
	SCAT1	-	-	LIBF	W1F00010
	PRNT2	-	-	LIBF	W1E00010
	HOLCA	-	-	-	W1C00010
	STRTB	-	-	-	W1G00010

<u>Function</u>	<u>Name(s)</u>	<u>Type</u>	<u>Subtype</u>	<u>Reference</u>	<u>Deck ID</u>
<u>SCA Subroutines (Cont'd)</u>					
	HXCV	-	-	-	W1D00010
	EBC48	-	-	-	W1A00010
	HOL48	-	-	-	W1B00010
<u>Mainline</u>					
Initialize Disk Cartridge	DISC	2	-	-	U6C00010
Print Cartridge ID	IDENT	2	-	-	U6F00010
Write Cartridge ID	ID	2	-	-	U6G00010
Copy Disk Cartridges	COPY	2	-	-	U6B00010
Write WS Addresses	ADRWS	2	-	-	U6A00010
Delete CIB	DLCIB	2	-	-	U6D00010
Maintenance Program	MODIF	2	-	-	U6H00010
Dump SLET	DSLET	2	-	-	U6E00010
Paper Tape Utility	PTUTL	2	-	-	U6I00010
<u>System/Miscellaneous</u>					
Call System Print Subroutine	CALPR	4	0	CALL	U7A00010
Fetch Phase IDs from SLET	FSLEN	4	0	CALL	U7B00010
Fetch System Subroutine	FSYSU	4	0	CALL	U7B00010
Read *ID Record and Convert	RDREC	4	0	CALL	U7C00010
<u>Interrupt Level Subroutines</u>					
Interrupt Level Zero Subroutine	ILS00	7	-	-	U1A00010
Interrupt Level One Subroutine	ILS01	7	-	-	U1B00010
Interrupt Level Two Subroutine	ILS02*	7	1	-	U1C00010
Interrupt Level Three Subroutine	ILS03	7	-	-	U1D00010
Interrupt Level Four Subroutine	ILS04*	7	1	-	U1E00010
<u>1627 Plotter Subroutines</u>					
	SCALE	-	-	-	V1N00010
	SCALF	-	-	-	V1O00010
	EGRID	-	-	-	V1C00010
	FGRID	-	-	-	V1H00010
	EPLOT	-	-	-	V1D00010
	FPLOT	-	-	-	V1I00010
	POINT	-	-	-	V1M00010
	ECHAR	-	-	-	V1A00010
	FCHAR	-	-	-	V1F00010
	ECHRX, ECHRI, VCHRI	-	-	-	V1B00010
	FCHRX, FCHRI, WCHRI	-	-	-	V1G00010
	ERULE	-	-	-	V1E00010
	FRULE	-	-	-	V1J00010
	PLOTI	-	-	-	V1K00010
	XYPLT	-	-	-	V1P00010
	PLOTX	-	-	-	V1L00010

*These are special versions that consist only of IBT information.

If (1) cylinder zero is defective, (2) more than three cylinders are defective, or (3) it is impossible to write a sector address, DISC types out an error message indicating that the cartridge may not be used.

If the cartridge is good, DISC writes the defective cylinder addresses, if any, in the first three words of sector @IDAD. Wherever a defective cylinder does not exist, /0658 is written in the first three words of sector @IDAD. DISC also writes the cartridge ID in word four of sector @IDAD, writes zeros in words 7-30, and stores an error message program beginning in word 31. If a cold start is attempted using this non-system cartridge, control is passed from the Cold Start Loader to the error message program instead of the Cold Start Program. An error message is printed on the Console Printer and no cold start is effected.

DISC initializes the following words of DCOM (sector @DCOM):

<u>Location</u>	<u>Value Inserted</u>
#ANDU	/0200
#BNDU	/0200
#FPAD	/0020
#CIDN	Cartridge ID
#CIBA	/0008
#ULET	/0002

DISC initializes LET (sector 2) as follows:

<u>Word</u>	<u>Contents</u>
1	/0000
2	/0020
3	/0000
4	/0138
5	/0000
6	/7112
7	/4528
8	/0620

} The name
} 1DUMMY (in
} name code)

DISC terminates with a CALL EXIT.

PRINT CARTRIDGE ID (IDENT)

This program prints out the ID and the physical drive number of each disk cartridge mounted on the system.

IDENT first fetches the principal print device subroutine IOAR header from SLET using the

subroutine FSLEN. This IOAR header is used to call in the principal print device subroutine when it is needed by FSYSU.

Next, IDENT reads DCOM to obtain #PCID, the table of disk cartridge IDs and their related physical drive numbers.

IDENT then prints the cartridge ID and physical drive number from the table until all available cartridge IDs have been printed.

IDENT terminates with a CALL EXIT.

CHANGE CARTRIDGE ID (ID)

This program changes the ID on up to four disk cartridges.

The IOAR headers for the principal input device, principal print device, and principal conversion subroutines are obtained from SLET on the system cartridge. These subroutines are used for input/output.

Using the RDREC subroutine, the *ID record is fetched. RDREC also builds two tables in core storage from the FROM-TO fields of the *ID record, one in packed EBCDIC for printer output, the other in binary for matching the cartridge IDs. DCOM is fetched from the master cartridge to obtain the cartridge ID table (#CIDN).

Each drive on the system is selected. If the selected drive is present, the cartridge ID is fetched. The ID is matched with the IDs in #CIDN. If no matching ID is found, the ID is printed with an error message and the job is terminated. The cartridge ID of the selected drive is matched to the IDs in the FROM-TO table. When a match occurs, the cartridge ID is changed to the 'TO' ID; the ID for the cartridge in #CIDN is changed and the 'TO' ID is written onto the selected drive. IDs that do not match entries in the FROM-TO table are bypassed.

When all IDs have been processed, #CIDN is written back to the master cartridge and the FROM-TO table is printed. After printing the FROM-TO table, ID terminates with a CALL EXIT.

DISK COPY (COPY)

This program copies the contents of one or more cartridges (except words 0-3 of sector @IDAD) onto from one to four other cartridges.

COPY first fetches the system device subroutine IOAR headers from SLET using the RDREC subroutine.

The system device subroutines are called in by the RDREC subroutine as they are needed by the program. The RDREC subroutine also reads the *ID record and converts the numbers to binary and stores them in the FROM-TO table.

COPY then checks the FROM and TO field IDs to ensure that each specified cartridge is available. An error message is printed for the unavailable FROM or TO cartridges.

All available FROM-TO cartridge combinations are then processed. Sectors 0 thru 7 of cylinder 0 of each source cartridge are read and written, except for the defective cylinder table, to each specified destination cartridge. Sectors 0 thru 7 of the next 199 logical cylinders of each source cartridge are copied, 4 sectors at a time to each specified destination cartridge.

One cartridge at a time is processed and at the end of each, a check for a Keyboard interrupt is made. If any occurred during the previous copy, the interrupt is now processed.

After all cylinders from the specified cartridge have been copied, a completion message is printed using the principal print device subroutine.

COPY terminates with a CALL EXIT.

DELETE CIB (DLCIB)

This program deletes the Core Image Buffer (CIB) from a non-system cartridge to provide additional disk storage area for the User Area and Working Storage. An *ID record is used to specify the cartridge on which the CIB is to be deleted.

DLCIB uses the subroutine RDREC to obtain the system device subroutine IOAR headers from SLET on the master cartridge and to fetch the *ID record containing the affected cartridge ID. The RDREC subroutine also converts the specified cartridge ID to binary.

If the specified cartridge is not present, DLCIB prints an error message and terminates with a CALL EXIT.

The CIB of the specified cartridge is deleted. The User Area and Working Storage are moved one cylinder closer to cylinder zero. Accordingly, the file-protection address for the specified cartridge is altered in the \$FPAD quintuple in COMMA.

DCOM of the master cartridge is then read. The sector addresses of the CIB, User Area, and Working Storage are altered. DCOM is written back to the master cartridge and to the altered cartridge.

DLCIB prints the new User Area and Working Storage addresses for the specified cartridge using the principal print device subroutine.

DLCIB terminates with a CALL EXIT.

DUMP SLET TABLE (DSLET)

DSLET dumps the System Location Equivalence Table (SLET) to the principal print device. Four 4-word SLET entries are printed per line.

DSLET reads the SLET table into a 640-word buffer in core storage, prints the SLET table using the principal print device subroutine, and terminates with a CALL EXIT.

SECTION 13. SYSTEM DEVICE SUBROUTINES

The system device subroutines are a group of special subroutines used exclusively by the monitor system programs. These are the only device subroutines used by the monitor system programs, aside from DISKZ. They are listed below:

DISKZ

1403 Subroutine

1132 Subroutine

Console Printer Subroutine

2501/1442 Subroutine

1442/1442 Subroutine

1134/1055 Subroutine

Keyboard/Console Printer Subroutine

2501/1442 Conversion Subroutine

1134/1055 Conversion Subroutine (dummy)

Keyboard/Console Printer Conversion Subroutine (dummy)

DISK CARTRIDGE INITIALIZATION PROGRAM (DCIP)

When DCIP is entered, a message is printed instructing the user to select the particular DCIP function desired. Depending on his choice, one of the functions described below is performed.

All messages, entries through the Console Entry switches, and operator instructions are printed on the Console Printer. All user options are entered through the Console Entry switches.

DISK INITIALIZATION

A message is printed instructing the user to specify the number of the physical drive on which is mounted the cartridge to be initialized. At the same time, the user is given the option of doing an "address only" initialization, that is, an initialization that writes correct addresses on a cartridge without disturbing any of the data on that cartridge. The user is then asked to specify the cartridge ID.

An entire cylinder of the cartridge is written with one of three test patterns. The patterns used are /AAAA, /5555, and /0000. The cylinder is then read back into core storage, one sector at a time, using double-buffering.

While one sector is being read, every word of another is being examined to see that it compares with the data that was written. If no errors occur in any sector of the cylinder, the same procedure is repeated for the next pattern, and so on until all three patterns have been tested.

However, if any disk operation causes the error bit of the disk device status word (DSW) to be set, or if the data read does not compare with that written, then the entire write/read/compare procedure is repeated fifty times on the same cylinder with the same test pattern. A second error, while in the retry mode, causes DCIP to indicate the cylinder as being defective.

If (1) cylinder zero is defective, (2) more than three cylinders are defective, or (3) it is impossible to write a sector address, DCIP types out an error message indicating that the cartridge may not be used.

After every cylinder on the cartridge has been tested in the above manner, the program writes three defective cylinder addresses and the cartridge ID into the first four words of sector @IDAD. Where defective cylinder addresses do not exist, /0658 is written. Words 7-30 of sector @IDAD are set to zeros. DCIP also writes an error message program, beginning at word 31. If a cold start is attempted using this non-system cartridge, the error message program prints an appropriate message and no cold start is effected.

DCOM (sector @DCOM) is initialized as follows:

<u>Location</u>	<u>Value Inserted</u>
#ANDU	/0200
#BNDU	/0200
#FPAD	/0020
#CIDN	Cartridge ID
#CIBA	/0008
#ULET	/0002

LET (sector 2) is initialized as follows:

<u>Word</u>	<u>Contents</u>
1	/0000
2	/0020
3	/0000
4	/0138
5	/0000
6 }	/7112 } The name 1DUMY (in name
7 }	
8	/0620

A message indicating that the initialization is complete and the addresses of any defective cylinders are printed on the Console Printer.

At this time, the user is given the option of doing additional testing of the disk; i. e., the write/read/compare sequence may be repeated up to 31 times.

DISK DUMP

The principal print device is determined by first initiating a carriage space operation on the 1403 Printer. The device status word (DSW) for the 1403 is then sensed to see if the 1403 is busy. If it is not,

the same procedure is followed with the 1132 Printer. On the basis of the results of the above test, a word that points to the appropriate conversion table and a branch instruction that branches to the proper printer call are set up.

The user enters through the Console Entry switches the sector address (with the drive code) of the first sector to be dumped and the number of consecutive sectors to be dumped.

The logical sector address is determined in the following manner. The physical sector address is decremented by eight for each defective cylinder that has a lower sector address less than the cylinder to be dumped from. If the sector being dumped is on a defective cylinder, the sector is assigned the logical sector address of DEAD. Defective cylinder data for the cartridge is obtained from sector @IDAD.

Each of the 320 data words of the sector is converted from binary to four hexadecimal characters of the appropriate printer code. The data is then printed, sixteen words per line.

DISK COPY

DCIP requests the user to enter the numbers of the source and destination drives in the Console Entry switches. The defective cylinder table from the source cartridge is fetched and checked to verify that the values in it are under 1624 and in ascending order.

The source cartridge is copied sector by sector onto the destination cartridge. The cartridge ID and defective cylinder table in sector 0, cylinder 0 are not copied onto the destination cartridge. If a system cartridge is being copied, the cartridge ID found in DCOM is also not copied.

If a cylinder on the source cartridge is defective, the following cylinder is copied to the destination cartridge. If a cylinder on the destination cartridge is defective, the cylinder to be copied from the source cartridge is copied onto the following cylinder.

UCART

The user receives the 1130 Disk Monitor System on a disk cartridge. The contents of this cartridge are as follows: cylinder 0 contains a copy of the Resident Image, including DISKZ, a copy of the CARD0 subroutine, a special cold start program, and a disk-to-card dump program; cylinders 1 through 202 contain the system decks stored in card images, four cards per sector.

The execution of a cold start with this cartridge causes sector 0, cylinder 0 to be fetched. Sector 0, cylinder 0 contains DISKZ and the special cold start program. DISKZ is loaded into the locations it normally occupies in the Resident Monitor; the special cold start program immediately follows it. Control is transferred to the special cold start program.

The special cold start program fetches the Resident Image (sector 2, cylinder 0) into the locations it normally occupies in low core storage, fetches the CARD0 subroutine (sector 3, cylinder 0) into core storage at /0250, and fetches the disk-to-card dump program into core storage at /0390. Control is transferred to the disk-to-card dump program, which punches the system decks and terminates.

The disk-to-card dump program uses a one-cylinder buffer originated at its high-addressed end.

INTRODUCTION

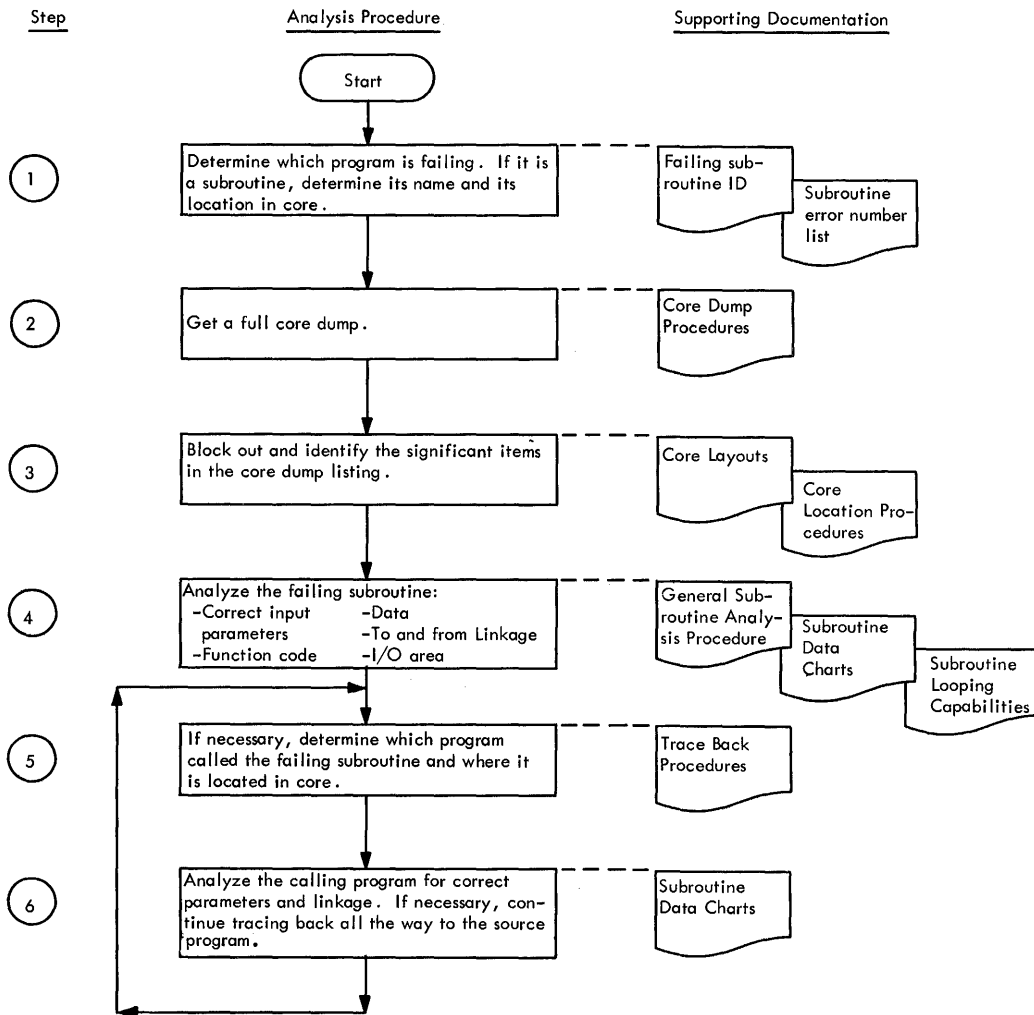
The purpose of the Program Analysis Procedures is to provide the user with a step-by-step method for analyzing the execution of any monitor system or user program. The procedure is problem-oriented; it begins with some program malfunction, assists the user in defining the failing component or function, and provides the facility for detailed analysis of that component or function.

PROGRAM ANALYSIS PROCEDURES SUMMARY

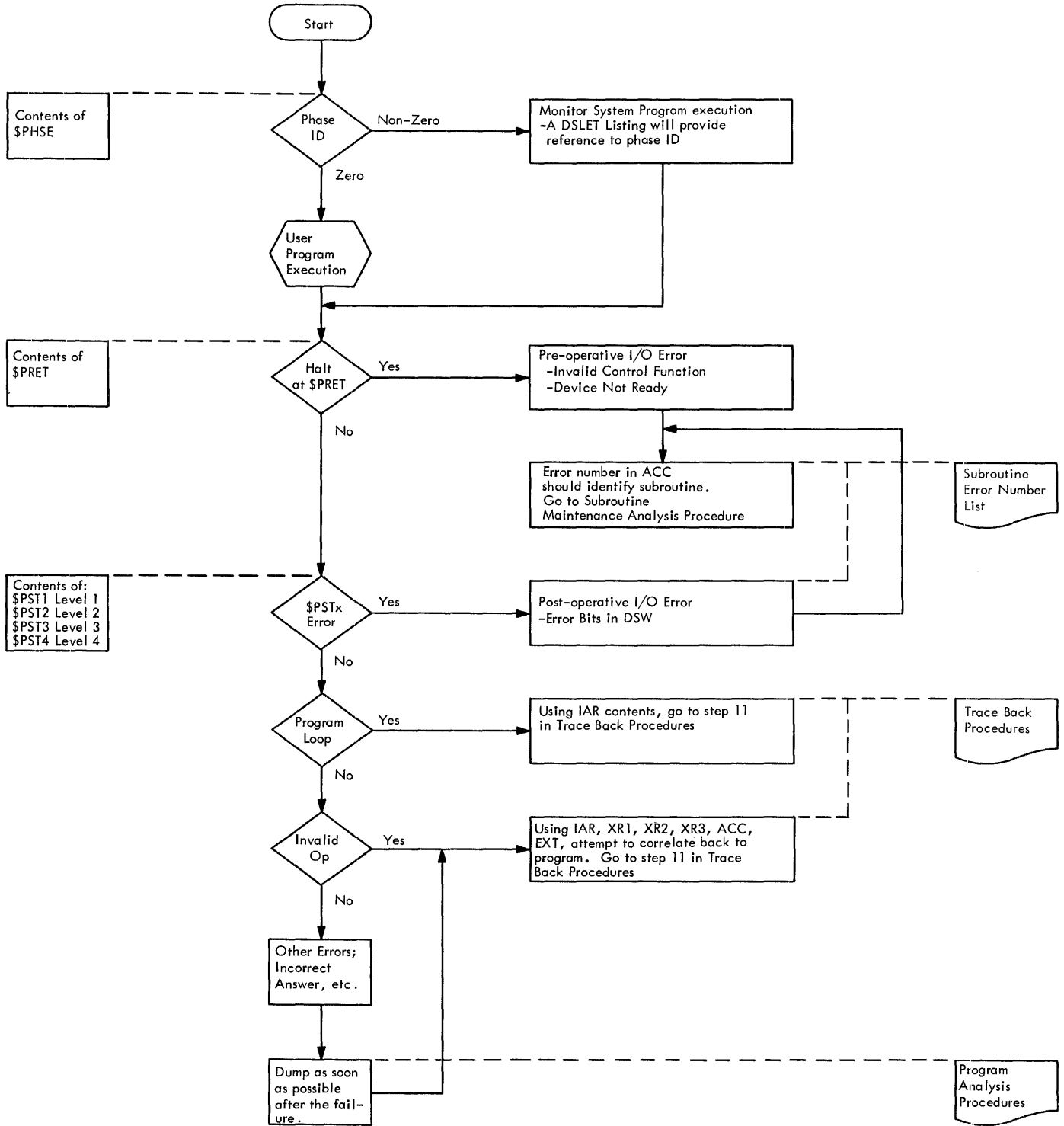
Flowdiagram 1 shows the procedure used for program analysis. At each step in the procedure, the parts of this document that apply to that step are indicated.

IDENTIFICATION OF THE FAILING COMPONENT OR FUNCTION

Flowdiagram 2 shows the procedure used to identify the component or function failing. Where applicable the parts of this document that are pertinent to that identification are indicated.



Flowdiagram 1. General Procedure for Program Analysis



Flowdiagram 2. Procedure for Identification of the Failing Component or Function

SUBROUTINE ERROR NUMBER/ERROR STOP
LISTS

Table 8 lists the errors detected by the System Library ISSs and system device subroutines by error code, describes the conditions under which the error is detected, and provides a list of corrective actions for those errors.

Table 9 lists the error stop addresses and their meanings.

CORE DUMP PROCEDURE

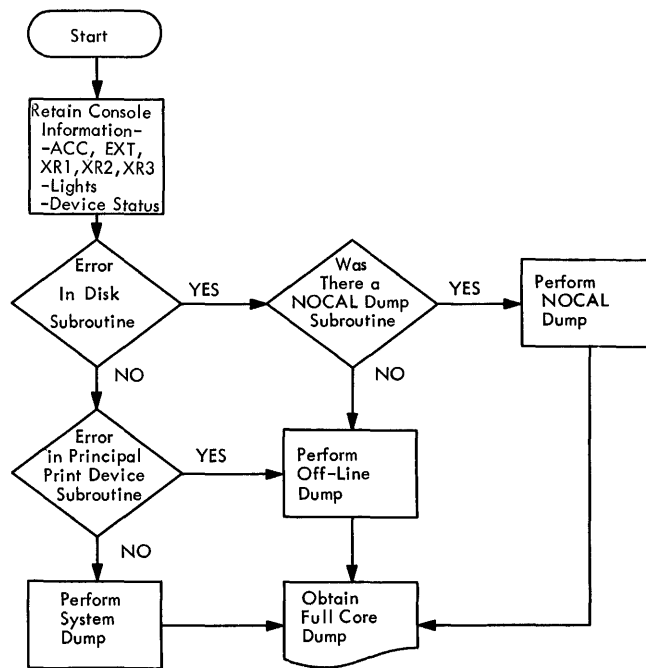
To obtain a dump of the contents of core storage, perform the following (see Flowdiagram 3):

1. If the error symptoms indicate that an error has occurred in the disk or disk I/O subroutine (i. e. DISKZ), the System Core Dump program should not be used, because this same disk I/O subroutine is used to load the System Core Dump program, thus destroying the information needed.
2. Was there a NOCAL Dump included in the core load? If there was, it may be used to obtain the core dump. To obtain the core dump, set the IAR to the entry point of the NOCAL Dump and start.
3. If the error symptoms indicate an error in the principal print device, then the System Core Dump program should not be used, as it would destroy any information needed.
4. To retain the maximum information, the off-line dump procedure should be used.
 - a. By displaying core storage, copy down locations /0000 to /0050.
 - b. Use the stand-alone printer dump to dump the rest of core storage.
5. To obtain the core dump using the System Core Dump program, set the IAR to /0000 and start.

STOP RESET START

CORE BLOCK DIAGRAMS

Figure 17, panel 1 shows the layout of the contents of core storage during the execution of a user's FORTRAN core load in which LOCAL subprograms,



Flowdiagram 3. Core Dump Procedure

files, arrays, and COMMON variables were defined and SOCALs were employed.

Figure 17, panel 2 shows the layout of the contents of core storage during the execution of a user's FORTRAN core load in which files, arrays, and COMMON variables were defined. No LOCALs were defined; no SOCALs were employed.

Figure 17, panel 3 shows the layout of the contents of core storage during the execution of a user's FORTRAN core load in which no LOCALs, files, arrays, or COMMON variables were defined and no SOCALs were employed.

Figure 17, panel 4 shows the layout of the contents of core storage during the execution of a user's Assembler Language core load.

CORE LOCATION PROCEDURES

The following core load elements are located by means of the procedures given with the elements.

FAC (Floating Accumulator)

- 3 words used as FORTRAN Floating Accumulator
- Located at XR3 + /007D.

Table 8. Error Number List

HEXADECIMAL ERROR NUMBER IN ACCUMULATOR	SYMBOLIC ERROR STOP ADDRESS	DETECTING SUBROUTINE DURING SYSTEM PROGRAM EXECUTION	DETECTING SUBROUTINE DURING FORTRAN CORE LOAD EXECUTION	DETECTING SUBROUTINE DURING ASSEMBLER LANGUAGE CORE LOAD EXECUTION	ERROR EXPLANATION	CORRECTIVE ACTION
1000	\$PRET	System 1442/1442 Subroutine	CARDZ PNCHZ	-	1442 - Device Not Ready	1442 - Ready the Device
		-	-	CARD0 CARD1	1442-6,-7 - Device Not Ready - Read initiated with Last Card Indicator on	
		-	-	PNCH0 PNCH1	1442-5 - Device Not Ready	
	\$PST4	System 1442/1442 Subroutine	CARDZ PNCHZ	CARD0 CARD1 PNCH0 PNCH1	1442 - Device Not Ready	1442-5 - Run out the punch - Ready the Device
						1442-6,-7 - Ready the Device
1001	\$PRET	-	-	CARD0 CARD1 PNCH0 PNCH1	1442 - Invalid Device Specified - Device not on system - Invalid Function Specified - Word Count over +80 - Word Count zero or negative	1442 - Use Trace Back Procedures to analyze calling program
2000	\$PRET	System Keyboard Subroutine, System Keyboard/ Console Printer Subroutine	TYPEZ WRTYZ	TYPE0 WRTY0	Console Printer/Keyboard - Device Not Ready	Console Printer/Keyboard - Ready the Device
	\$PST4	System Keyboard Subroutine, System Keyboard/ Console Printer Subroutine	TYPEZ WRTYZ	TYPE0 WRTY0	Console Printer/Keyboard - Device Not Ready	Console Printer/Keyboard - Ready the Device
2001	\$PRET	-	-	TYPE0 WRTY0	Console Printer/Keyboard - Device not on System - Invalid Function Specified - Word Count zero or negative	Console Printer/Keyboard - Use Trace Back Procedures to analyze calling program
3000	\$PRET	System 1134/1055 Subroutine	PAPTZ	PAPT1 PAPT X PAPT N	1134/1055 - Device Not Ready	1134/1055 - Ready the Device
	\$PST4	System 1134/1055 Subroutine	PAPTZ	PAPT1 PAPT X PAPT N	1134/1055 - Device Not Ready	1134/1055 - Ready the Device
3001	\$PRET	-	-	PAPT1 PAPT X PAPT N	1134/1055 - Invalid Function Specified - Invalid Check Digit - Word Count zero or negative	1134/1055 - Use Trace Back Procedures to analyze calling program
4000	\$PRET	System 2501/1442 Subroutine	READZ	READ0 READ1	2501 - Device Not Ready	2501 - Ready the Device
	\$PST4	-	READZ	READ0 READ1	2501 - Device Not Ready - Read Error - Feed Check	2501 - Ready the Device - Run out the reader and retry with last card read and cards run out

Table 8. Error Number List (Continued)

4001	\$PRET	-	-	READ0 READ1	2501 - Invalid Function Specified - Word Count over +80 - Word Count zero or negative	2501 - Use Trace Back Procedures to analyze calling program
5000	\$PRET	DISKZ	DISKZ	DISKZ DISK1 DISKN	Disk - Device Not Ready	Disk - Ready the Device
5001	\$PRET	-	-	DISK1 DISKN	Disk - Invalid Device Specified - Device not in System - Invalid Function Specified - Area to be written File-protected - Word Count zero or negative - Starting Sector Address over +1599	Disk - Use Trace Back Procedures to analyze calling program
	\$PST2	DISKZ	DISKZ	DISKZ DISK1 DISKN	Disk - Power Unsafe - Write Select	Disk - Turn power down, wait for CARTRIDGE UNLOCKED light to come on, turn power up, then retry - Call CE on persistent error
5002	\$PST2	DISKZ	DISKZ	DISKZ DISK1 DISKN	Disk - 16 retrys made without success	Disk - Initiate 16 more retrys - Use another drive - Use another cartridge - Reinitialize cartridge
5003	\$PRET	DISK1 DISKN	-	-	Disk - Invalid Device Specified - Device not in System - Invalid Function Specified - Area to be written File-protected - Word Count zero or negative - Starting Sector Address over +1599	Disk - Use Trace Back Procedures to analyze calling program
5004	\$PST2	DISKZ	DISKZ	DISKZ	Disk - Disk Error	Disk - Turn power down, wait for CARTRIDGE UNLOCKED light to come on, turn power up, then retry - Call CE on persistent error
6000	\$PRET	System 1132 Subroutine	PRNTZ	PRNT1 PRNT2 PRNT3	1132 - Device Not Ready - End of Forms	1132 - Ready the Device
6001	\$PRET	-	-	PRNT1 PRNT2 PRNT3	1132 - Invalid Function Specified - Word Count over +60 - Word Count zero or negative	1132 - Use Trace Back Procedures to analyze calling program
7000	\$PRET	-	-	PLOT1	1627 - Device Not Ready	1627 - Ready the Device
	\$PST3	-	-	PLOT1	1627 - Device Not Ready	1627 - Ready the Device
7001	\$PRET	-	-	PLOT1	1627 - Invalid Device Specified - Device not on System - Invalid Function Specified - Word Count zero or negative	1627 - Use Trace Back Procedures to analyze calling program
8001	\$PRET	-	-	SCAT1 SCAT2 SCAT3	SCA - Invalid Function Specified - Invalid Word Count - Invalid Subfunction Specified	SCA - Use Trace Back Procedures to analyze calling program

Table 8. Error Number List (Concluded)

8002	\$PRET	-	-	SCAT1	SCA - Receive operation not completed - Transmit operation not completed	SCA - Use Trace Back Procedures to analyze calling program
8003	\$PRET	-	-	SCAT1	SCA - Synchronization not established before attempting to perform some Transmit or Receive Operation - Attempting to Receive before receiving INQ sequence	SCA - Use Trace Back Procedures to analyze calling program
9000	\$PRET	System 1403 Subroutine	PRNZ	PRNT3	1403 - Device Not Ready - End of Forms	1403 - Ready the Device
	\$PST4	System 1403 Subroutine	PRNZ	PRNT3	1403 - Device Not Ready - Print Error	1403 - Ready the Device
9001	\$PRET	-	-	PRNT3	1403 - Invalid Function Specified - Word Count over +60 - Word Count zero or negative	1403 - Use Trace Back Procedures to analyze calling program
A000	\$PRET	-	-	OMPR1	1231 - Device Not Ready	1231 - Ready the Device
	\$PST4	-	-	OMPR1	1231 - Device Not Ready - Timing Mark Error - Read Error	1231 - Ready the Device - Retry with the sheet that has been selected into the stacker
A001	\$PRET	-	-	OMPR1	1231 - Invalid Function Specified	1231 - Use Trace Back Procedures to analyze calling program

Table 9. Error Stop List

Absolute Address	Symbolic Address	Program	Explanation
/0014	-	Cold Start Loader	-Invalid disk drive number in Console Entry Switches -Indicated disk drive not ready
/0044	-	Cold Start Loader	-Disk read error -Waiting for interrupt from seek operation
/0046	-	Cold Start Loader	-Waiting for interrupt from reading sector @IDAD
/0029	\$PRET+1	All ISSs	-Preoperative Error
/0082	\$PST1+1	Level 1 ISSs	-Post-operative Error on level 1
/0086	\$PST2+1	Level 2 ISSs	-Post-operative Error on level 2
/008A	\$PST3+1	Level 3 ISSs	-Post-operative Error on level 3
/008E	\$PST4+1	Level 4 ISSs	-Post-operative Error on level 4

ARITHMETIC AND FUNCTION SUBPROGRAM ERROR INDICATORS

- 3 words preceding FAC.
- First word (XR3 + /007A) is used for real arithmetic overflow and underflow indicators.
- Second word (XR3 + /007B) is used for divide check indicator.
- Third word (XR3 + /007C) is used for function subroutine indicators.
- The loader initializes all three words to zero.

LIBF TV (Library Function Transfer Vector)

- One 3-word entry for each LIBF listed in the core map.
- Located just preceding ARITH/FUNC ERROR INDICATORS.
- Higher core end is located at XR3 + /0079.
- First LIBF Entry (the beginning of LIBF TV) is located at (XR3 + /0077) - (3 times the number of LIBFs listed in core map).

①	②	③	④
COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor	COMMA, Skeleton Supervisor
DISKZ	DISKZ	DISKZ	DISK1 or DISKN
DEFINE FILE Table	DEFINE FILE Table	Constants, Integers	Mainline Program
Arrays	Arrays	Format Parameters	
Constants, Integers	Constants, Integers	Mainline Program	
Format Parameters	Format Parameters		
Mainline Program	Mainline Program	In-Core Subroutines	
In-Core Subroutines	In-Core Subroutines		
Flipper Table		Interrupt Level Subroutines	
FLIPR	Interrupt Level Subroutines		
LOCAL Area		Interrupt Level Subroutines	
SOCAL Area	Interrupt Level Subroutines		
Interrupt Level Subroutines		Interrupt Level Subroutines	
LIBF TV	LIBF TV		LIBF TV
CALL TV	CALL TV	CALL TV	CALL TV
COMMON	COMMON	COMMON	

Figure 17. Core Layout During User Core Load Execution

LIBF TV SOCAL LINKAGE

- The 6 or 9 words used to link to the SOCAL/LOCAL Flipper.
- Located just preceding the First LIBF Entry in the LIBF TV.

-- 6 words long if SOCAL option 1; 9 words long if SOCAL option 2.

CALL TV (Call Transfer Vector)

- One single-word entry for each call listed in the core map.
- Located immediately following FAC.
- First CALL TV Entry is at $XR3 + /0080$ (add 1 if address comes out odd).
- The Last CALL TV Entry is at (First CALL TV Entry-1) - (Number of CALLs listed in the core map).

DISK I/O SUBROUTINE

- All Disk I/O subroutines are loaded beginning at CORE LOCATION /00F2.
- The Disk I/O subroutines vary in length (see table)
- The type of disk subroutine in core is contained in \$DZ1N (see table)

Contents of \$DZ1N	Disk I/O Subroutine Currently in Core	Location in Core		First Word of User's Program
		First Word	Last Word	
FFFF	DISKZ	/00F2	/01DF	/01FE
0000	DISK1	/00F2	/0293	/02B2
0001	DISKN	/00F2	/03A1	/03C0

DFT (DEFINE FILE Table)

- 7 words for each file defined by the user.
- Located at 1 plus the end of the disk I/O subroutine.

ARRAYS (In User's Program Area)

- Located immediately following DEFINE FILE Table, if any.

CONSTANTS AND INTEGERS (In User's Program Area)

- Located immediately following ARRAYS, if any.

COMMON (The area at "End of Core" defined by COMMON statement)

- Length of COMMON is contained in \$COMN.
- Start of COMMON is highest core address, (XFFF), minus the Length of COMMON.

IN-CORE SUBROUTINES (subroutines that are in core all the time)

- Located immediately following user's mainline program.

-- Those subroutines listed in core map that are not SOCALLS or LOCALS are In-Core subroutines.
 -- The load address of these subroutines is listed with subroutine name.

LDD		LIST
BSI	L	ENTRY POINT
.		
.		
.		
LIST	DC	PARAMETER
	DC	PARAMETER

LOCAL/SOCAL FLIPPER (FLIPR)

-- Load address given in core map under the heading SYSTEM SUBROUTINES.

LOCAL AREA (The "Load-on-Call" overlay area)

-- Size depends upon largest LOCAL subroutine used.
 -- Beginning core address is FLIPR + /0066.
 -- Ending core address is address of SOCAL Area minus 1.

SOCAL AREA (System overlay area)

-- Located immediately following LOCAL Area.
 -- Beginning core address is found at FLIPR + /004D.
 -- The first word in SOCALL Area contains the word count of SOCALL Area.
 -- Ending core address is the beginning address + the word count of the SOCALL Area.

GENERALIZED SUBROUTINE MAINTENANCE/ANALYSIS PROCEDURE

Flowdiagram 4 provides the procedure to be used for detailed analysis of an I/O subroutine. The procedure is applicable to FORTRAN device, general ISS, and system device subroutines.

TRACE BACK PROCEDURES

Flowdiagram 5 provides the procedure to be used to trace back from a failing subroutine to the preceding portion of the core load, which called the subroutine. This procedure can be used to trace all the way back to the mainline program.

SUBROUTINE LOOPING CAPABILITIES

SYSTEM DEVICE SUBROUTINES

The linkages to system device subroutines are of the form:

To place the subroutine into a loop:

1. Obtain link word from the system device subroutine.
2. The contents of this link word point to the location following the long BSI instruction.
3. Insert into the location following the long BSI instruction an MDX instruction back to the LDD instruction.

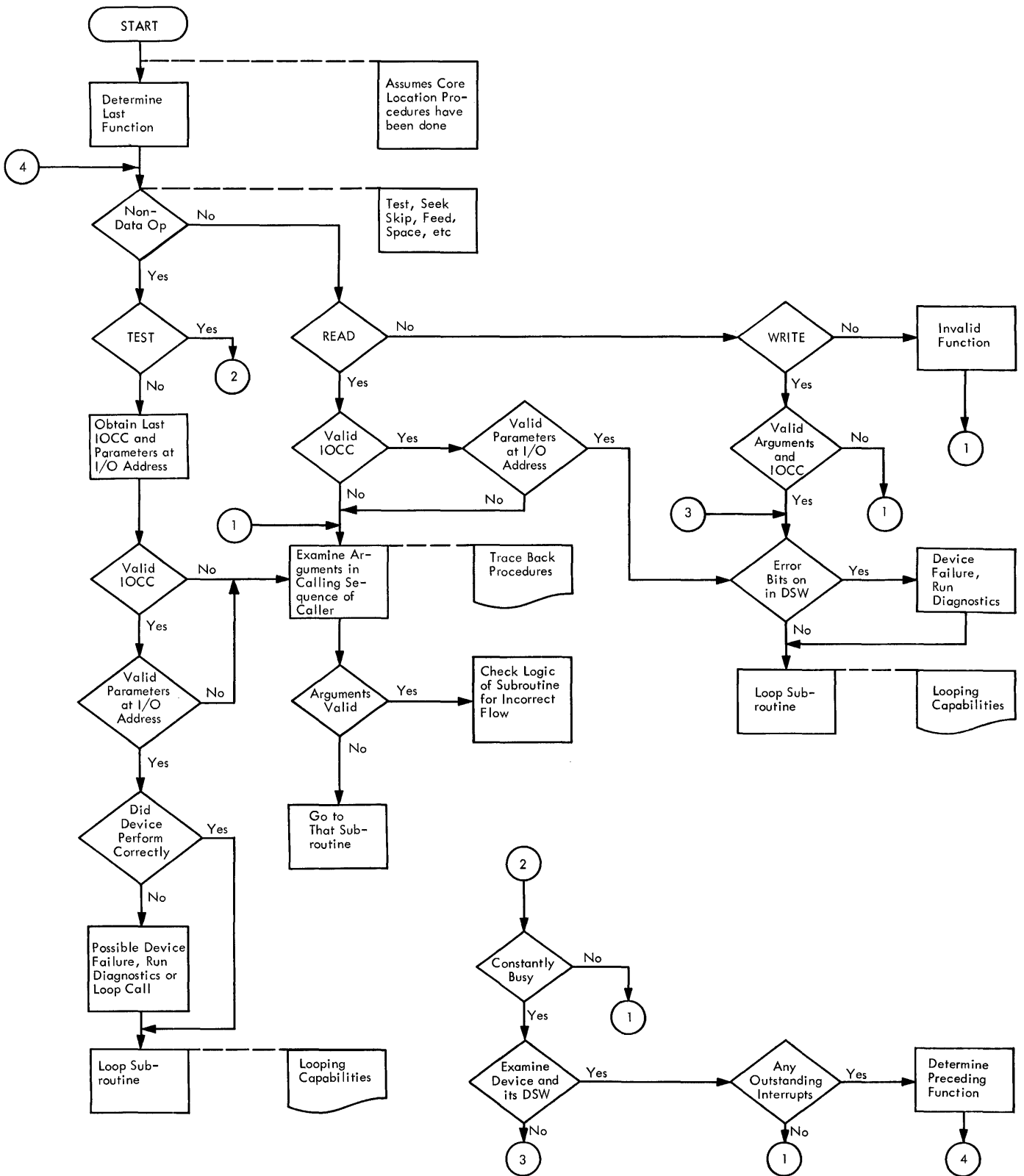
LIBRARY SUBROUTINES (except 'Z' subroutines and PLOTX)

The linkage to the System Library device subroutines (ISSs) are of the following form:

<u>LIBF</u>		<u>CALL</u>			
BSI	3	TVDISP	BSI	I	CALLTV
DC		CONTROL	DC		CONTROL
DC		ARG 1	DC		ARG 1
.			.		
.			.		
DC		ARGN	DC		ARGN

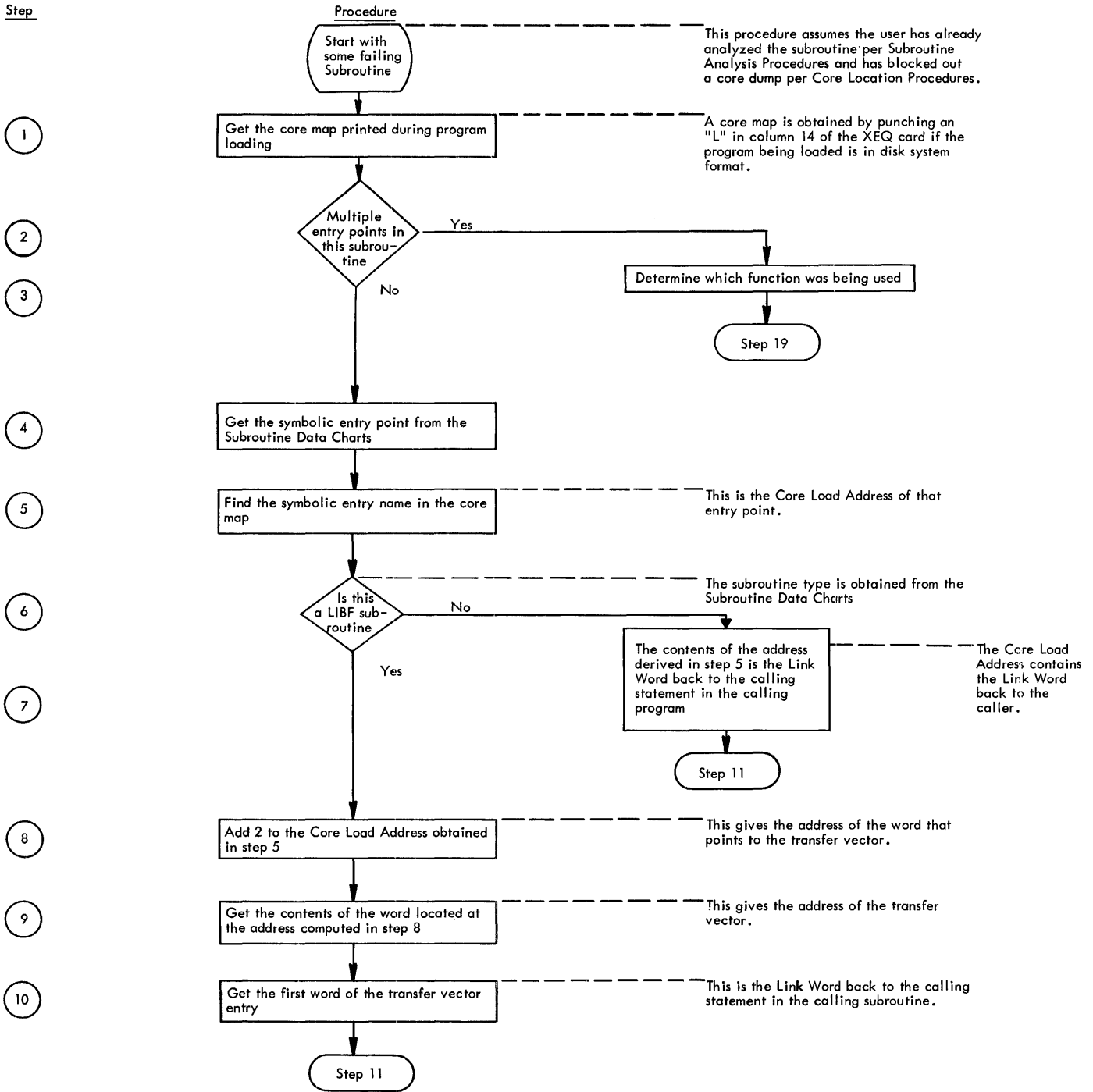
To place the subroutine into a loop:

1. Insert in the location following the last argument an MDX instruction back to the BSI instruction.
2. Some of the arguments may have to be changed to point to the BSI instruction because they are error exits or busy addresses.
3. Refer to subroutine data charts for unique operating characteristics.



Flowdiagram 4. Generalized Subroutine Maintenance/Analysis Procedure

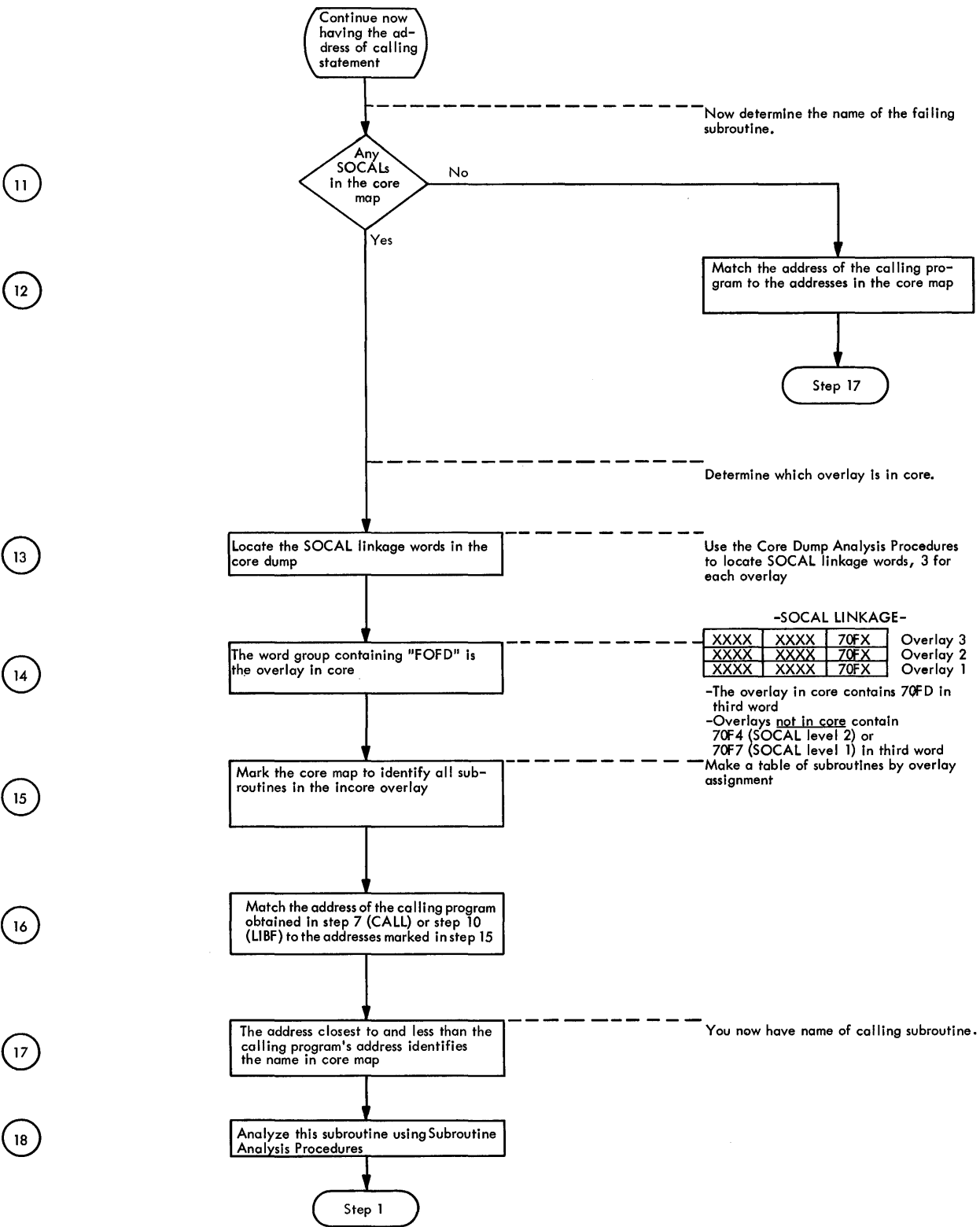
Step



Flowdiagram 5. Trace Back Procedures

Step

Procedure



Flowdiagram 5. Trace Back Procedures (Continued)

Step

19

Get the symbolic location of function code from the Subroutine Data Charts

The different entry points are related to the various "functions" performed by the subroutine.

20

Find the symbolic location in the subroutine listing

Microfiche reference to subroutine listing is given in Appendix D.

21

Get the relative address of the location of function code

The address at the symbolic location determined in step 20 is the relative address.

22

Get the load point address for this subroutine from core map

23

Calculate core location of function code

The relative address determined in step 21 plus the load point address determined in step 22 gives the location of the function code.

24

Get the function code from core dump and decode with Subroutine Data Chart

25

Get the symbolic name for each function entry point from the Subroutine Data Chart

Step 5

Flowdiagram 5. Trace Back Procedures (Concluded)

SUBROUTINE DATA CHARTS

SYSTEM DEVICE SUBROUTINE FOR KEYBOARD/CONSOLE PRINTER

Phase ID: @ KBCP

Used by: Monitor system programs

Subroutines required: ILS04

Linkage: LDD LIST
 BSI L KB000+1

 LIST DC FUNCTION CODE
 DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Read, Convert, Print	/7002	Address of I/O Area	Word Count

Postoperative conditions and entry points:

Function at KB080	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read, Convert, Print	/7002	KB000+1	KB020+1	KB020+1	

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		

Significant variables:

Symbolic location	Contents/Use
KB080	The function is placed here. It becomes an MDX *+2 when executed.
KB160	Original I/O area address.
KB170	Original word count.
KB270	Character buffer area, containing a 12-bit character read from the Keyboard, the rotate/tilt code character printed, or a control character.
KB280	Data area pointer, pointing to the next word in the data area into which the EBCDIC character will be placed.
KB290	Remaining word count.
KB370 and KB370+1	Read/Print Control IOCC.

SYSTEM DEVICE SUBROUTINE FOR 1442/1442

Phase ID: @ 1442

Used by: Monitor system programs

Subroutines required: ILS04

Linkage: LDD LIST
BSI L CD000+1

LIST DC FUNCTION CODE
DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Read	/7000	Address of the I/O Area	No word count is used but an 80 position area must be specified.
Punch	/7001	Address of the I/O Area	No word count is used but an 80 position area must be specified.
Read	/7002	Address of the I/O Area	No word count is used but an 80 position area must be specified.
Feed	/7003	Not used	Not used

Postoperative conditions and entry points:

Function	at CD090	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/7000	CD100+1	CD100+1	CD016+1 CD010+1	CD016+1 CD010+1	0 4
Punch	/7001	CD000+1	CD000+1	CD016+1 CD010+1	CD016+1 CD010+1	0 4
Read	/7002	CD000+1	CD000+1	CD016+1 CD010+1	CD016+1 CD010+1	0 4
Feed	/7003	CD000+1	CD000+1	CD010+1	CD010+1	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						CD120
	Used	X	X				
Interrupt level 0	Saved at/restored from symbolic location	CD190	CD190+1				CD018
	Used	X					
Interrupt level 4	Saved at/restored from symbolic location						
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
CD120	First column indicator.
CD250	Current column address.
CD260	Second half of the last IOCC performed, read or punch.
CD230	Second half of the last IOCC performed, start read or punch.
CD188	Skip indicator; non-zero = take one feed cycle.
\$LAST	Last card indicator; non-zero = last card.
\$CTSW	Control card switch; non-zero = control card read.
\$IBSY	Busy indicator for 1442; non-zero = busy.

SYSTEM DEVICE SUBROUTINE FOR 2501/1442

Phase ID: @ 2501

Used by: Monitor system programs

Subroutines required: ILS04

Linkage: LDD LIST
BSI L RP000+1

LIST DC FUNCTION CODE
DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Read	/7000	Address of I/O Area	Word count
Punch	/7001	Address of I/O Area	Not used
Read	/7002	Address of I/O Area	Word count
Feed	/7003	Not Used	Not used

Postoperative conditions and entry points:

Function	at RP360	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/7000	RP000+1	RP000+1	RP020+1	RP020+1	4
Punch	/7001	RP000+1	RP000+1	RP040+1 RP020+1	RP040+1 RP020+1	0 4
Read	/7002	RP000+1	RP000+1	RP020+1	RP020+1	4
Feed	/7003	RP000+1	RP000+1	RP020+1	RP020+1	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						RP440
	Used	X	X				
Interrupt level 0	Saved at/restored from symbolic location	RP480	RP480+1				RP060
	Used	X					
Interrupt level 4	Saved at/restored from symbolic location						
	Used	X	X		X		

Significant variables:

Symbolic location	Contents/Use
RP500	Current column address.
RP520	I/O address for restart information.
RP600	Word count for 2501 Reader.
RP200	Device last used: /1702 = 1442 /4F01 = 2501
\$LAST	Last card indicator; non-zero = last card.
\$CTSW	Control card switch; non-zero = control card read.
\$IBSY	Busy indicator; non-zero = busy.

SYSTEM DEVICE SUBROUTINE FOR CONSOLE PRINTER

Phase ID: @ CPTR

Used by: Monitor system programs

Subroutines required: ILS04

Linkage: LDD LIST
BSI L CP000+1

LIST DC FUNCTION CODE
DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Restore	/7000		
Write	/7001	Address of I/O Area	Word Count
Skip	/7002		

Postoperative conditions and entry points:

Function at	CP120	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
All		CP000+1	CP000+1	CP020+1	CP020+1	

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location				CP170+2		
	Used	X	X		X	X	

Significant variables:

Symbolic location	Contents/Use
CP120	<p>The function is placed here. This word is executed to decode the function. /7000 is a MDX * /7001 is a MDX * + 1 /7002 is a MDX * + 2</p>
CP200	<p>Carriage return counter, used for counting carriage returns for restore.</p>
CP350	<p>IOCC for printing on console.</p>
CP350+1	<p>CP350 contains address of CP450.</p>
CP370	<p>Actual word count of message not including trailing blanks.</p>
CP380	<p>Data area pointer, pointing to the word containing the 2 EBCDIC characters, one of which is being printed.</p>
CP450	<p>Print character buffer word. The IOCC points to this word, which contains the character or control character just printed.</p>

SYSTEM DEVICE SUBROUTINE FOR 1132

Phase ID: @1132

Used by: Monitor system programs

Subroutines required: ILS01

Linkage: LDD LIST
BSI L PN000

LIST DC FUNCTION CODE
DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Print	/7001	I/O Area address	0 ≤ word count ≤ 80
Skip to Channel 1	/7000		I/O Area is referenced
Space Immediate	/7002	Not used	Not used

Postoperative conditions and entry points:

Function	at PN380	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Print	/7001	PN000+1	PN000+1	PN010+1	PN010+1	1
Skip	/7000	PN000+1	PN000+1	PN010+1	PN010+1	1
Space	/7002	PN000+1	PN000+1	PN010+1	PN010+1	1

Register Status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location			PN400+1	PN400+3	PN400+5	
	Used	X	X	X	X	X	X
Interrupt level 1	Saved at/restored from symbolic location	PN200	PN200+1	PN440+1	PN440+3	PN440+5	PN450
	Used	X	X	X	X	X	X

Significant variables:

Symbolic location	Contents/Use
PN040	Last emitter character read as a result of a read emitter response interrupt.
PN050	Second word of Sense-DSW-with-reset IOCC.
PN060	Last DSW sensed in interrupt.
PN070	Second word of Sense-DSW-with-no-reset IOCC.
PN080	First word of Read emitter IOCC, contains the address of location PN040.
PN090	Second word of Read emitter IOCC.
PN100	First word of Start Printer IOCC, also the idle scan counter.
PN110	Second word of Start Printer IOCC.
PN120	Print scan counter.
PN130	Second half of Stop Printer IOCC.
PN150	Second half of Start Carriage IOCC.
PN170	Second half of Stop Carriage IOCC.
PN180	First half of Stop Carriage IOCC and mask to check bits 3, 5, and 6 of Printer DSW.
PN370+1	Address of the I/O area.
PN460+1	Word count.
PN470+1	Address of message.
\$PBSY	<p>/0001 indicates I/O buffer is busy and 49 print scan cycles have not been completed.</p> <p>/0000 indicates routine may still be busy completing the 16 idle scans; however, I/O buffer is ready to accept new input.</p>
\$CH12	<p>Zero = Channel 12 has not been sensed.</p> <p>Non-Zero = Channel 12 has been sensed and skip to channel 1 has not been performed.</p>

SYSTEM DEVICE SUBROUTINE FOR 1403

Phase ID: @1403

Used by: Monitor system programs

Subroutines required: ILS04

Linkage: LDD LIST
 BSI L PR000+1

 LIST DC FUNCTION CODE
 DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Print 1 Line	/7001	Address of I/O Area	$0 \leq \text{word count} \leq 60$
Skip	/7000	Address of I/O Area	I/O Area is referenced
Space Immediate	/7002	Not used	Not used

Postoperative conditions and entry points:

Function	at PR150	I/O Area word count	Location reference	Symbolic entry point	Return address at	Interrupt entry point
Print	/7001	Non-Zero		PR000+1	PR000+1	
Skip	/7000	Not used				
Space	/7002	Not Used				

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location			PR230+1	PR240+1	PR250+1	
	Used	X	X	X	X	X	X

Significant variables:

Symbolic location	Contents/Use
PR140	<p>A NOP after the following areas have been adjusted to the correct address as a result of relocation:</p> <ul style="list-style-type: none"> PR300 PR500+1 PR180+2 PR280+2 PR170+1 PR220+2 PR080 PR400+1 PR430+1
PR080 PR080+1	<p>First word of Print IOCC, contains address of print buffer. Second half of Print IOCC.</p>
PR110	<p>Second half of Sense-without-reset IOCC.</p>
PR300 PR300+1	<p>First word of Skip-to-channel-12 IOCC. Second word of Skip-to-channel-12 IOCC.</p>
PR090 PR090+1	<p>First word of Space immediate IOCC. Second word of Space immediate IOCC.</p>
PR290+1	<p>Address of the user's I/O area.</p>
PR060	<p>Storage location for the last DSW sensed during interrupt; also, first word of Sense-DSW-with-reset IOCC.</p>
PR070	<p>Second word of Sense-DSW-with-reset IOCC.</p>
PR110	<p>Second word of Sense-DSW-without-reset IOCC.</p>
PR330	<p>60-word buffer from which the line is printed.</p>
\$PGCT	<p>Binary page count, where $0 \leq \text{page count} \leq 32767$</p>
\$CH12	<p>Channel 12 switch indicating channel 12 detected in DSW during interrupt; not reset until a skip to channel 1 is requested by the user.</p>
\$PBSY	<p>Printer busy switch, modified during execution of routine.</p> <p>(\$PBSY) = Zero: routine and printer not busy. = Positive: transmission to printer is in progress; transmission complete has not been received; subroutine I/O buffer is busy. = Negative: transmission complete has been received; subroutine I/O buffer can now be set up with new message.</p>

SYSTEM DEVICE SUBROUTINE FOR 1134/1055

Phase ID: @ 1134

Used by: Monitor system programs

Subroutine required: none

Linkage: LDD LIST
BSI L PT000+1

LIST DC FUNCTION CODE
DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address
Read with conversion	/7000	Address of I/O Area	Word Count
Punch	/7001	Address of I/O Area	Word Count
Read without conversion	/7002	Address of I/O Area	Word Count

Postoperative conditions and entry points:

Function	at PT060	Symbolic entry point	Return address at	Interrupt entry point	Interrupt level
Read with conversion	/7000	PT000+1	PT000+1	PT010+1	
Punch	/7001	PT000+1	PT000+1	PT010+1	
Read without conversion	/7002	PT000+1	PT000+1	PT010+1	

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X			X	

Significant variables:

Symbolic location	Contents/Use
PT000+1	Return address to caller from main line.
PT010+1	Return address from interrupt.
PT060	Function code, executed as follows: /7000 = MDX * /7001 = MDX *+1 /7002 = MDC *+2
PT340	Data area pointer.
PT360	Remaining word count.
PT370	Switch used for reading or punching: /0001 = punch, /0002 = read.
PT380	Switch used to indicate if conversion of information read is needed: zero = no conversion, non-zero = conversion.
PT460 and PT460+1	IOCC for read and punch.
PT480	Data buffer. The character to be read or punched is contained here.
PT500	Counter for counting first 3 characters.

SYSTEM DEVICE SUBROUTINE FOR DISK -- DISKZ

Phase ID: @DZID

Used by: Monitor system programs
 Assembler Language programs
 FORTRAN programs

Subroutines required: ILS02

Linkage: LDD LIST
 BSI L DZ000
 .
 LIST DC FUNCTION CODE
 DC I/O AREA ADDRESS

Preoperative input parameters:

Function	ACC	EXT	I/O Area Address	I/O Area Address + 1
Read	/7000 or /0000	Address of the I/O Area (must be even)	0 ≤ word count ≤ length of defined data area	Drive code and sector address
Write	/7001	Address of the I/O Area (must be even)	0 ≤ word count ≤ length of area to be written on disk	Drive code and sector address
Find	/0000	Address of the I/O Area (must be even)	/0000	Drive code and sector address

Postoperative conditions and entry points:

Function	at DZ945	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/0000	DZ000	DZ100+5	DZ150	DZ150	2
Write	/0100	DZ000	DZ100+5	DZ150	DZ150	2
Find	/0000	DZ000	DZ100+5	DZ150	DZ150	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location			DZ100+1	DZ100+3		
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
DZ350+1	Address of the word in COMMA containing the current position of the heads on the referenced disk.
DZ230+1	Address of the first word of the I/O Area. $C[DZ230+1]$ = originally requested word count $C[DZ230+1] + 1$ = originally requested sector address
DZ904 and DZ905	First and second words of the last IOCC performed (excluding sense DSW).
DZ908 and DZ909	First and second words of forced-Read after-Seek IOCC.
DZ901	Sector address of previously executed forced Read.
DZ906 and DZ907	IOCC developed for user-requested function.
DZ975	Second word of Read-Back-Check IOCC.
DZ912	Word count remaining to be read or written from original.
DZ913	Next sector to be read or written.
DZ910	Second word of Seek IOCC.

CARDZ

Flowcharts: FIO04-05

Used by: SFIO

Subroutine required: HOLEZ, ILS01, ILS04

Linkage: LIBF CARDZ (BSI 3 TV DISP)

where ACC = FUNCTION CODE
 XR1 = I/O AREA ADDRESS
 XR2 = WORD COUNT

Preoperative input parameters:

Function	ACC	XR1	XR2
Read	/0000	I/O Area Address	Word Count
Write	/0002	I/O Area Address	Word Count

Postoperative conditions and entry points:

Function	at CZ912	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/0000	CARDZ	LIBF TV link word	CZ100 (column)	CZ100 (column)	1
Write	/0002	CARDZ	LIBF TV link word	CZ110 (op complete)	CZ110 (op complete)	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
CZ904 CZ904+1	<p>Start read or punch IOCC, set by program depending on function used to initiate operation.</p> <p>If read, $CZ904 + 1 = CZ906 + 1$</p> <p>If write, $CZ904 + 1 = CZ908 + 1$</p>
CZ902	<p>Read or Punch IOCC, set by program depending on function to read or punch columns.</p> <p>If read, $CZ902 + 1 = CZ906$</p> <p>If write, $CZ902 + 1 = CZ908$</p>
CZ923	<p>Address pointer to I/O area, incremented on each column interrupt.</p>
CZ925	<p>Original I/O area address -1.</p>
CZ920	<p>DSW is saved here on an operation-complete interrupt.</p>
CZ010	<p>Switch used for waiting for interrupt:</p> <p>Set positive when waiting for any interrupt.</p> <p>Set zero when column interrupt occurs.</p> <p>Set negative when op-complete interrupt occurs.</p>
\$RWCZ	<p>Previous operation switch:</p> <p>/0000 = previous operation was a read.</p> <p>/0002 = previous operation was a write.</p> <p>If a write function is to be performed and the previous operation was a write, this switch causes CARDZ to read a card and test for // in columns 1-2.</p>
CZ913	<p>Switch used to test for // card before writing on it; zero means only reading or previous operation before write was a read.</p>
CZ918-3 thru CZ918	<p>Buffer area for saving first 3 columns; rest of card is read into fourth word when reading before write.</p>

PNCHZ

Flowchart: FIO11

Used by: SFIO

Subroutines required: HOLEZ, ILS01, ILS04

Linkage: LIBF PNCHZ (BSI 3 TV DISP)

where ACC = FUNCTION CODE
 XR1 = I/O AREA ADDRESS
 XR2 = WORD COUNT

Preoperative input parameters:

Function	ACC	XR1	XR2
Write	/0002	I/O Area Address	Word (character) count of 80

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Write	PNCHZ	LIBF TV link word	PZ060 (column)	PZ060	0
			PZ080 (op complete)	PZ080	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
PZ340	Address pointer to I/O area; First word of Punch IOCC, Incremented on each column interrupt.
PZ340+1	Second word of Punch IOCC.
PZ360 and PZ360+1	Feed IOCC for initiating punch operation.
PZ400 PZ400+1	Error display indicator. Second word of last card feed IOCC.
PZ120+1	Original I/O area address.
PZ040	Switch used for waiting for operation -- complete interrupt: zero = op complete interrupt has occurred non-zero = waiting for op-complete

READZ

Flowchart: FIO08

Used by: SFIO

Subroutines required: HOLEZ, ILS04

Linkages: LIBF READZ (BSI 3 TV DISP)
 where XR1 = I/O AREA ADDRESS

Preoperative input parameters:

Function	XR1
Read	I/O Area Address

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	READZ	LIBF TV link word	RZ060	RZ060	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
RZ360	I/O Area Address; also, first word of Read IOCC.
RZ360+1	Second word of Read IOCC.
RZ380	Switch used for interrupt processing: non-zero = waiting for interrupt zero = set by occurrence of interrupt

TYPEZ

Flowchart: FIO12

Used by: SFIO

Subroutines required: HOLEZ, GETAD, ILS04

Linkage: LIBF TYPEZ (BSI 3 TV DISP)

where ACC = FUNCTION CODE
 XR1 = I/O AREA ADDRESS
 XR2 = WORD COUNT

Preoperative input parameters:

Function	ACC	XR1	XR2
Read	/0000	I/O Area Address	Word count, set to 80 by TYPEZ
Write	/0002	I/O Area Address	Character count

Postoperative conditions and entry points:

Function	at KZ910	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/0000	TYPEZ	TYPEZ+2	KZ100	KZ100	4
Write	/0002	TYPEZ	TYPEZ+2	KZ100	KZ100	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		

Significant variables:

Symbolic location	Contents/Use
KZ911	Original character count plus one.
KZ210+1	Original I/O area address.
KZ910	Read-Write function indicator word.
KZ906	IOCC used to print characters from KZ914.
KZ914	Buffer word used to hold character to be printed.
KZ913	Saved DSW from Sense-with-reset in interrupt routine.
KZ900	IOCC used to read Keyboard character into I/O area.
KZ902	IOCC used to release Keyboard.
KZ912	Number of remaining characters to be typed. (Each character read is typed.)

WRTYZ

Flowchart: FIO09

Used by: SFIO

Subroutines required: GETAB, EBCTR, ILS04

Linkage: LIBF WRTYZ (BSI 3 TV DISP)
 where XR1 = I/O AREA ADDRESS
 XR2 = WORD COUNT

Preoperative input parameters:

Function	XR1	XR2
Write	I/O Area Address	Character Count

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Write	WRTYZ	WRTYZ+2	TZ100	TZ100	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		

Significant variables:

Symbolic location	Contents/Use
TZ907	Number of characters remaining to be printed.
TZ908	Output buffer for printing character.
TZ902	IOCC used to print character out of TZ908.

PRNZ

Flowchart: FIO10

Used by: SFIO

Subroutines required: ILS04

Linkage: LIBF PRNZ (BSI 3 TV DISP)

where XR1 = I/O AREA ADDRESS

XR2 = WORD COUNT

Preoperative input parameters:

Function	XR1	XR2
Print	I/O Area Address	Word count, including 1 for carriage control character

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Print	PRNZ	PRNZ+2	INTZ	INTZ	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
PRNT	First word of Print IOCC, Address of output area.
SIO+1	Address for store after character conversion.
COUNT	Counter for character conversion.
TABL	EBCDIC-to-1403-Printer-code conversion table.
TRCSW	Transfer complete switch: zero = transfer complete.
CHAR	EBCDIC character being converted.

PRNTZ

Flowchart: FIO06

Used by: SFIO

Subroutines required: ILS02

Linkage: LIBF PRNTZ (BSI 3 TV DISP)

where XR1 = I/O AREA ADDRESS

XR2 = WORD COUNT

Preoperative input parameters:

Function	XR1	XR2
Print	Output buffer address (first character is carriage control)	Word count, including carriage control character.

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Print	PRNTZ	PRNTZ+2	AZ100	AZ100	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used	X	X	X	X		X
Interrupt level 2	Saved at/restored from symbolic location						
	Used						

Significant variables:

Symbolic location	Contents/Use
AZ150+1	Address of first data word in output buffer.
AZ919	Word count.
AZ900	Interrupt exit switch: + , if line is complete - , if idles complete 0 , if waiting
AZ922	Space counter (positive number of spaces).
AZ914	DSW storage.
AZ924	Scan counter (print).
AZ918	Emitter-character storage.

PAPTZ

Flowchart: FIO07

Used by: SFIO

Subroutine required: ILS04

Linkage: LIBF PAPTZ (BSI 3 TV DISP)

where ACC = FUNCTION CODE
 XR1 = I/O AREA ADDRESS
 XR2 = WORD COUNT

Preoperative input parameters:

Function	ACC	XR1	XR2
Read	/0000	Address of I/O Area	Word count, set to 120 by PAPTZ
Write	/0002	Address of I/O Area	Word count

Postoperative conditions and entry points:

Function	at BZ924	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Read	/0000	PAPTZ	PAPTZ+2	BZ100	BZ100	4
Write	/0002	PAPTZ	PAPTZ+2	BZ100	BZ100	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location						
	Used						

Significant variables:

Symbolic location	Contents/Use
BZ924	Read/Write indicator.
BZ929	Number of words remaining to be read or punched.
BZ300+1	Address of I/O Area.
BZ010	Routine busy indicator: zero, no interrupt waiting to be processed; non-zero, an interrupt waiting to be processed.
BZ902	IOCC used to Start paper tape reader.
BZ904	IOCC used to Read paper tape.
BZ925	Read area for BZ904, Read paper tape IOCC. Write area for BZ906, Punch paper tape IOCC.
BZ926	DSW from sense-with-reset in interrupt subroutine.
BZ906	IOCC used to Punch paper tape.

CARD0

Used by: Assembler Language programs

Subroutines required: ILS00, ILS04

Linkage: LIBF CARD0 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if busy	Return to this word if not busy	Not used
Read	/1000	I/O Area Address	NSI	Word count
Punch	/2000	I/O Area Address	NSI	Word count
Feed	/3000	Not used	NSI	Not used
Stack	/4000	Not used	NSI	Not used

Postoperative conditions and entry points:

Function	at CA20	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		CARD0	CA34+1			
Read	/7000	CARD0	CA34+1	INT1 INT2	INT1 INT2	0 4
Punch	/7001	CARD0	CA34+1	INT1 INT2	INT1 INT2	0 4
Feed	/7002	CARD0	CA34+1	INT2	INT2	4
Stack	/7003	CARD0	CA34+1			

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	TEMP		CA30+1	CA31+1		CA32
	Used	X		X	X		

Significant variables:

Symbolic location	Contents/Use
COUNT	Number of words left to be transferred.
COLM	Address being transferred to or from.
RSTRT	Word count for restart.
RSTRT+1	Starting address for restart.
BUSY	Busy indicator; non-zero = busy.
CHAR	Second half of the Sense DSW IOCC that was last executed.
ERROR	Skip indicator; non-zero = feed a card.

CARD1

Used by: Assembler Language programs

Subroutines required: ILS00, ILS04

Linkage: LIBF CARD1 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if busy	Return to this word if not busy	Not used
Read	/1000	I/O Area Address	Address of user error routine	Word count
Punch	/2000	I/O Area Address	Address of user error routine	Word count
Feed	/3000	I/O Area Address	Address of user error routine	Not used
Stack	/4000	Not used	Not used	Not used

Postoperative conditions and enter points:

Function	at CR24+1	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		CARD1	EXIT+1			
Read	/0001	CARD1	EXIT+1	INT1 INT2	INT1 INT2	0 4
Punch	/0002	CARD1	EXIT+1	INT1 INT2	INT1 INT2	0 4
Feed	/0003	CARD1	EXIT+1	INT2	INT2	4
Stack	/0004	CARD1	EXIT+1			

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	TEMP	CR42+1	CR44+1			CR46
	Used	X					X

Significant variables:

Symbolic location	Contents/Use
RESTR	Count information for restart.
RESTR+1	I/O area address for restart.
RESTR+2	Address of error routine; also, busy indicator.
ERROR	Skip indicator; non-zero = feed a card.
INDIC	Feed check at read station indicator; non-zero = feed check.
INIT	Last initiate command given.
COLM	Address being transferred to or from.
COUNT	Number of words to transfer.
CHAR	Second half of the Sense DSW IOCC used.

READ0

Used by: Assembler Language programs

Subroutines required: ILS04

Linkage: LIBF READ0 (BSI 3 TV DISP)

DC ARG1

DC ARG2

Preoperative input parameters:

Function	ARG1	ARG2	I/O Area
Test	/0000	Return to this word if busy	Not used
Read	/1000	Address of word count	Word count
Feed	/1000	Address of word count	Word count (must be zero)

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	READ0	RE180+1			
Read	READ0	RE180+1	RE048	RE048	4
Feed	READ0	RE180+1	RE048	RE048	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	RE324		RE144+1	RE156+1		RE168
	Used	X		X			

Significant variables:

Symbolic location	Contents/Use
RE228	Busy indicator; non-zero indicates busy.
RE264	I/O area address.

READ1

Used by: Assembler Language programs

Subroutines required: ILS04

Linkage: LIBF READ1 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if busy	Return to this word if not busy	Not used
Read	/1000	I/O Area Address	Address of user's error routine	Word count
Feed	/1000	I/O Area Address	Address of user's error routine	Word count (must be zero)

Preoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	READ1	RE180+1			
Read	READ1	RE180+1	RE048	RE048	4
Feed	READ1	RE180+1	RE048	RE048	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	RE324		RE144+1	RE156+1		RE168
	Used	X		X			

Significant variables:

Symbolic location	Contents/Use
RE228	Busy indicator; non-zero indicates busy.
RE264	I/O Area address.
RE360+2	Address of user's error routine for read error.
RE370+2	Address of user's error routine during last card.

PNCH0

Used by: Assembler Language programs

Subroutines required: ILS00, ILS04

Linkage: LIBF PNCH0 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if busy	Return to this word if not busy	Not used
Punch	/2000	Address of I/O Area	Return to this word following call	Word count
Feed	/3000	Not used	NSI	Not used

Postoperative conditions and entry points:

Function	at CA20	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PNCH0	CA34+1			
Punch	/7001	PNCH0	CA34+1	INT1 INT2	INT1 INT2	0 4
Feed	/7002	PNCH0	CA34+1	INT2	INT2	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	TEMP		CA30+1	CA31+1		CA32
	Used	X		X			

Significant variables:

Symbolic location	Contents/Use
CHAR	Second half of DSW last sensed.
COLM	Address being punched from.
BUSY	Non-zero indicates busy.
COUNT	Number of columns to be punched.
ERROR	Non-zero indicates feed a card (SKIP).
RSTRT	Word count for restart.
RSTRT	Data address for restart.

PNCH1

Used by: Assembler Language programs
 Subroutines required: ILS00, ILS04
 Linkage: LIBF PNCH1 (BSI 3 TV DISP)
 DC ARG1
 DC ARG2
 DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if busy.	Return to this word if not busy	Not used
Punch	/2000	Address of I/O Area	Address of user's error routine	Word count
Feed	/3000	Not used	Address of user's error routine	Not used

Postoperative conditions and entry points:

Function	at CA20	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PNCH1	CA34+1			
Punch	/7001	PNCH1	CA34+1	INT1 INT2	INT1 INT2	0 4
Feed	/7002	PNCH1	CA34+1	INT2	INT2	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic locations	TEMP		CA30+1	CA31+1		CA32
	Used	X		X			

Significant variables:

Symbolic location	Contents/Use
CHAR	Second half of Sense DSW IOCC.
COLM	Address being punched from.
BUSY	Busy indicator; non-zero = busy.
COUNT	Number of columns to punch.
ERROR	Non-zero indicates feed a card (SKIP).
INDIC	Read station feed check if non-zero.
RSTRT	Word count for restart.
RSTRT+1	Data address for restart.
RSTRT+2	Address of user's error routine.

TYPE0

Used by: Assembler Language programs
 Subroutine required: ILS04
 Linkage: LIBF TYPE0 (BSI 3 TV DISP)
 DC ARG1
 DC ARG2

Preoperative input parameters:

Function	ARG1	ARG2	I/O Area Address
Test	/0000	Return to this word if operation is not complete	
Read-Print	/1000	I/O Area Address	Word Count
Print	/2000	I/O Area Address	Word Count

Postoperative conditions and entry points:

Function	at TY24	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		TYPE0	EXIT+1			
Read Print	/7000	TYPE0	EXIT+1	INT1	INT1	4
Print	/7001	TYPE0	EXIT+1	INT1	INT1	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic locations	SAVAQ	SAVAQ+1	SAV1+1	SAV2+1		SAVST
	Used	X		X	X		X

Significant variables:

Symbolic location	Contents/Use
TY24	Functional branch instruction: /7000 = MDX * if Read/Print. /7001 = MDX *+1 if Print.
READ	Pointer to data input area.
READ+1	Last half of Read IOCC.
RSTRT+1	Data area address.
RSTRT+2	Word count.
COUNT	Contents depend on the function: /7000 - number of words remaining to be read. /7001 - count of remaining characters to be printed, initially set to twice the word count.
PRINT	IOCC used to print character from TEMPI.
INTT	IOCC used to release keyboard.
DSWRD	Device status word from sensing device in interrupt routine.
RIGHT	Switch indicating which character in TEMPI will be used next: /0000 = Use right character /0001 = Go get next word from data area and use left character.
TEMPI	Contents depend on the function: /7000 - rotate/tilt character converted from hollerith input character from keyboard. Character is printed on console from this area. /7001 - Temporary storage for printing a character (high order 8 bits was last character printed).
TY90+1	Address of Hollerith table.
TY92+1	Address of Rotate/Tilt character table.

PRNT1

Used by: Assembler Language programs

Subroutines required: ILS01

Linkage: LIBF PRNT1 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3
Test	/0000	Returns to this word if routine is busy	Returns to this word if routine is not busy
Print	/20X0 X = space control 0 , space after print 1 , suppress space	I/O Area Address	Error routine address
Control Carriage	/3XY0 X = immediate control Y = after print control	Return to this word.	
Print Numeric	/40X0 X = space control 0 , space after print 1 , suppress space	I/O Area Address	Error routine address

Postoperative conditions and entry points:

Function	at PAR1	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PRNT1	EXIT+1			
Print	/02X0	PRNT1	EXIT+1	INT1	INT1	1
Control Carriage	/3XY0	PRNT1	EXIT+1	INT1	INT1	1
Print Numeric	/40X0	PRNT1	EXIT+1	INT1	INT1	1

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	AQ	AQ+1	FC58+1	FC58+3		FC58+5
	Used	X	X	X	X		X
Interrupt level 1	Saved at/restored from symbolic location				OUT+1		
	Used	X			X		

Significant variables:

Symbolic location	Contents/Use
ILLGL+1	Address of call +1.
NEGWD	2's complement of the word count.
CLEAR	Last entry to the clear print buffer routine.
DSW	DSW from the last interrupt.
SPSK	Space count if (-). Skip if (+). Compare for skip response interrupt.
PASS	Interrupt switch.
FC16+1	Address of call +2.
STRE3+2	Address of call +3.
SCAN+1	End of the I/O area.
CTR48	Scan counter to determine when line is complete.
CTR16	Counter for 16 idles.

PRNT3

Used by: Assembler Language programs

Subroutines required: ILS04

Linkage: LIBF PRNT3 (BSI 3 TV DISP)
 DC ARG1
 DC ARG2
 DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3
Test	/0000	Return to this word if busy	Return to this word if not busy
Print	/20Z0 Z = space control 1, space suppressed 0, space after print	Address of I/O Area	
Control	/3XY0 X = immediate control Y = control after print		

Postoperative conditions and entry points:

Function		Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PRNT3	W3160+1			
Print	Third digit at W3540	PRNT3	W3160+1	W3020	W3020	4
Control	Third digit at W3540	PRNT3	W3160+1	W3020	W3020	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	W3520	W3520+1	W3100+1	W3120+1		W3140
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
W3540	First word of the Sense-DSW-without-reset IOCC; also, Carriage control character.
W3540+1	Routine busy switch, non-zero indicates the routine is processing a message.
W3920	First word of 60-word output buffer.
W3900	DSW in interrupt, except carriage interrupt; also carriage control in interrupt.
W3260+1	Address of user output area.

PAPT1

Used by: Assembler Language programs

Subroutines used: ILS04

Linkage: LIBF PAPT1 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return to this word if the operation is not complete	Return to this word if previous operation is complete	
Read	/1X00 X=0, Check X=1, No check	I/O Area Address	Address of user error routine	Word count (1/2 the number of characters)
Punch	/2X00 X=0, Check X=1, No check	I/O Area Address	Address of user error routine	Word count (1/2 the number of characters)

Postoperative conditions and entry points:

Function	at DEVIC	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PAPT1	RET+1			
Read	/0002	PAPT1	RET+1	INT1	INT1	4
Punch	/0001	PAPT1	RET+1	INT1	INT1	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVA		XRL+1			XRL+2
	Used	X		X			X

Significant variables:

Symbolic location	Contents/Use
DEVLC	Function indicator: /0002 = Read. /0001 = Punch.
CHECK	Switch for controlling checking function: /FF00 = Do not check. /0000 = Check for a Delete or Stop character.
WDCNT	Count of remaining words.
IOAR	Data area pointer.
USERR+2	Address of user error routine.
BUF	Temporary storage for word containing 2 characters to be punched.
SENSE	Sense DSW for paper tape.
READS	IOCC for starting paper tape reader.
IOCC	Read or punch control word.
CHART	Character switch: (Punch) Even = Both characters in the word have been punched, go get the next character. Odd = Punch the right character. (Read) Even = Second character of word was just read. Odd = First character of word was just read.

PAPTN

Used by: Assembler Language programs

Subroutine required: ILS04

Linkage: LIBF PAPTN (BSI 3 TV DISP)

DC ARG1
DC ARG2
DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000 Test reader /0001 Test punch	Return to this word if the previous operation is not complete	Return to this word if previous operation is complete	
Read	/1X00 X=0, Check X=1, No check	I/O Area Address	Address of user error routine	Word count (1/2 the number of characters)
Punch	/2X00 X=0, Check X=1, No check	I/O Area Address	Address of user error routine	Word count (1/2 the number of characters)

Postoperative conditions and entry points:

Function		Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		PAPTN	RET+1			
Read	RDTBL	PAPTN	RET+1	INTN	INTN	4
Punch	PNTBL	PAPTN	RET+1	INTN	INTN	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVA		XR1+1	XR2+1		XR2+2
	Used	X		X	X		X

Significant variables:

Symbolic location	Contents/Use
CHECK	/0000 = Check for a Delete or Stop character. /FF00 = Do not check.
WDCNT RWDCT	Number of words remaining to be punched. Number of words remaining to be read.
IOAR RIOAR	Address pointer to user data area (punch). Address pointer to user data area (read).
USER1 RUSE1	Address of user error routine (punch). Address of user error routine (read).
BUF RBUF	Temporary storage for word to be punched. Temporary storage for word to be read into.
Index Register 2	Address of RDTBL, if reading. Address of PNTBL, if punching.
IOCC READS	IOCC used for punching a character. IOCC used to start tape reader.
CHAR (RCHAR)	Switch used to indicate which half of the word is to be used: Even = Both characters in word used. Odd = First character of word was used.
SENSR	DSW received from Sense-with-reset IOCC.
RIOCC	IOCC used to read paper tape.
IOCC2	IOCC used to punch a Delete character.

PLOT1

Used by: Assembler Language programs

Subroutines used: ILS03

Linkage: LIBF PLOT1 (BSI 3 TV DISP)
 DC ARG1
 DC ARG2
 DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	I/O Area Address
Test	/0000	Return here if routine busy	Return here if routine not busy	Not used
Plot	/1000	I/O Area Address	Address of user's error routine	Word count

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	PLOT1	RET+1			
Plot	PLOT1	RET+1	INT1	INT1	3

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVAQ	SAVAQ+1	XR1+1			XR1+2
	Used	X		X			

Significant variables:

Symbolic location	Contents/Use
DEVIC	Non-zero indicates invalid device.
BUSY	Non-zero indicates busy.
DIGIT	Counter to determine which section of packed word is being used.
SENSE	Word count, number of words for this plot.
IOAR	Output area address in user program.
BUF	Word being decoded for plotting.
FIRST	Non-zero indicates first command.
DUPCT	Non-zero indicates repeat some command or plot.
CTRL	Last plot command executed.
WORK	Command that has been separated out of BUF.

OMPR1

Used by: Assembler Language programs

Subroutines required: ILS04

Linkage: LIBF OMPR1 (BSI 3 TV DISP)

DC ARG1

DC ARG2

DC ARG3

Preoperative input parameters:

Function	ARG1	ARG2	ARG3	EXT
Test	/0000	Return to this word if program is busy	Return to this word if program is not busy	
Timing Mark Test	/0001	Return to this word if bit 8 <u>on</u> in DSW	Return to this word if bit 8 is <u>off</u> in DSW	
Read	/2000	I/O Area Address	Address of user error routine	/X--- X = 1, Stacker Select X = 0, No Stacker Select
Read	/3000	NSI (return to this word after feed)	NSI	/X--- X = 1, Stacker Select X = 0, No Stacker Select
Disconnect	/4000	NSI (return to this word after disconnect)	NSI	
Stacker Select	/5000	NSI (return to this word after Stacker Select)	NSI	

Postoperative conditions and entry points:

Function	at	FUN	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test			OMPR1				
Timing Mark Test			OMPR1				
Read		/FFFE	OMPR1		INT1	INT1	4
Feed		/0000	OMPR1		INT1	INT1	4
Disconnect			OMPR1				
Stacker Select			OMPR1				

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVAQ	SAVAQ+1	SAVX1	MPR32+1		MPR34
	Used	X	X	X			X

Significant variables:

Symbolic location	Contents/Use
Accumulator	At entry to a user error routine: ACC = 0001, Master Mark detected. ACC = 0002, Read Error and/or Timing Mark Error. ACC = 0003, Hopper Empty. ACC = 0004, Document Selected.
ERREX+2	Entry to user's error routine.
FUN	Function: /FFFE = Read (2000) /0000 = Feed (3000)
READ	IOCC for Read function.
READ+1	Address of user I/O area.
FEED	/0000, Feed has already been performed. /0001, Feed has not been initiated for this Read function.
CNTRL and CNTRL+1	IOCC for Feed function.
STKSL and STKSL+1	IOCC for Stacker select.
DSCNT and DSCNT+1	IOCC for Disconnect function.
BUSY	Program busy indicator: zero = Not Busy. non-zero = Busy.
FCI	Zero, No first character interrupt. Non Zero, Have received first character interrupt.
MMARK	Master mark indicator switch: zero = No master mark. non-zero = Master mark read.
Index Register 1	Address of the calling sequence.

WRTY0

Used by: Assembler Language programs

Subroutines required: ILS04

Linkage: LIBF WRTY0 (BSI 3 TV DISP)

DC ARG1

DC ARG2

Preoperative input parameters:

Function	ARG1	ARG2	I/O Area Address
Test	/0000	Return to this word if operation is not complete	
Print	/2000	I/O Area Address	Word Count

Postoperative conditions and entry points:

Function	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	WRTY0	WR36+1			
Print	WRTY0	WR36+1	INT1	INT1	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVA		WR30+1	WR32+1		WR34
	Used	X		X	X		X

Significant variables:

Symbolic location	Contents/Use
COUNT	Dynamic word counter; number of words remaining to be printed.
IOAR	Pointer to data area, address of last word used.
TEMP1	Character to be printed, high order 8 bits was last character printed.
PRINT	First word of IOCC to print character out of TEMP1.
PRINT+1	Second word of IOCC to print character out of TEMP1.
RIGHT	Switch indicating which character in TEMP1 will be used next: /0000 = Right character. /0001 = Go get next word from data area and use left character.
TEMP	Right half of Sense-with-reset IOCC.

DISK1

Used by: Monitor system programs, Assembler Language programs

Subroutines required: ILS02

Linkage:* for monitor system programs

```

                LDD      LIST
                BSI     L    D0000
                .
LIST   DC      FUNCTION CODE
        DC      I/O AREA ADDRESS
    
```

for Assembler Language programs

```

LIBF      DISK1 (BSI 3 TV DISP)
DC        ARG1
DC        ARG2
DC        ARG3
DC        ARG4
    
```

*This subroutine may be entered by user-written Assembler Language programs via the LIBF TV or by monitor system programs via a direct branch. If a direct branch is used, DISK1 uses the parameters contained in the ACC and EXT to construct the parameters of a LIBF and simulates a LIBF entry by calling on itself.

Preoperative input parameters (for LIBF entry) † :

Function	ARG1	ARG2	ARG3	ARG4	I/O Area Address
Test	/0000	Not used	Return to this word if busy	Return to this word if not busy	Not used
Read	/1000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write W/O RBC	/2000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write With RBC	/3000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write Immediate	/4000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = not used Word 2 = drive code and sector address
Seek	/500x x = seek option displacement	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = not used Word 2 = drive code and sector address

†Applies to simulated LIBF.

Postoperative conditions and entry points:

Function	at D0945	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	/0000	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Read	/0001	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Write W/O RBC	/0002	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Write W/RBC	/0003	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Write Immediate	/0004	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Seek W/O Seek Option	/0005	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2
Seek With Seek Option	/0005 (D0946=x000)	D0000 (branch) D0007 (LIBF)	D0030	D0004	D0004	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	D0918	D0919	D0020+1	D0020+3		D0010
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
D0006	If DISK1 was entered by a monitor system program at D0000, this word is used by DISK1 to simulate a LIBF link word. The contents of this word should reference the simulated LIBF at symbolic location D0060.
D0008+1	Address of the link word in the LIBF TV if called by a user-written LIBF; address of D0006 (simulated LIBF TV) if routine was entered from a monitor system program via a direct branch.
D0235+1 (D0310+1)	Address of the word containing the beginning of the file-protected area of the disk on the specified drive.
D0280+1	Address of a table in COMMA containing the list of defective cylinders of the disk on the referenced drive.
D0305+1 (D0330+1)	Address of the device code to be used corresponding to the referenced drive.
D0350+1 (D0390+1)	Address of the word in COMMA containing the current position of the heads on the referenced disk.
D0901	Sector address of previously executed Forced Read.
D0904 and D0905	First and second words of the last IOCC performed (excluding Sense Device).
D0906 and D0907	First and second words of the IOCC for the user-requested operation: D0906 = address of the next I/O area; D0907 = area code/function to be performed and current or next sector address.
D0909	Second word of Read-after-Seek IOCC.
D0910	Second word of Seek IOCC.
D0911	Second word of Sense IOCC.
D0912	Word count remaining to be read or written from originally requested word count.
D0913	Next sequential sector to be read or written.
D0938	Current sector address.
D0947	If seek option was invoked, this word contains the displacement from the originally requested sector address.
D0908	Second word of Read-back-check IOCC.
DBUSY	Non-zero indicates routine is busy; this word must be cleared to zero before entry to this routine is permitted.
D0902 and D0903	Last 2 words of sector just read or written.

DISKN

Used by: Monitor system programs, Assembler Language programs

Subroutines required: ILS02

Linkage:* for monitor system programs

```

                LDD      LIST
                BSI     L   D0000
                .
LIST   DC      FUNCTION CODE
        DC      I/O AREA ADDRESS
    
```

for Assembler Language programs

```

LIBF      DISKN (BSI 3 TV DISP)
DC        ARG1
DC        ARG2
DC        ARG3
DC        ARG4
    
```

*This subroutine may be entered by user-written Assembler Language programs via the LIBF TV or by monitor system programs via a direct branch. If a direct branch is used, DISKN uses the parameters contained in the ACC and EXT to construct the parameters of a LIBF and simulates a LIBF entry by calling on itself.

Preoperative input parameters (for LIBF entry)[†]:

Function	ARG1	ARG2	ARG3	ARG4	I/O Area Address
Test	/0000	Not used	Return to this word if busy	Return to this word if not busy	Not used
Read	/1000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write W/O RBC	/2000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write With RBC	/3000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = word count Word 2 = drive code and sector address
Write Immediate	/1000	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = not used Word 2 = drive code and sector address
Seek	/500x x = seek option displacement	Address of the I/O Area	Address of user error routine	Return to this word after initiating operation	Word 1 = not used Word 2 = drive code and sector address

[†]Applies to simulated LIBF.

Postoperative conditions and entry points:

Function	at	D0995	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		/0000	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Read		/0001	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Write W/O RBC		/0002	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Write With RBC		/0003	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Write Immediate		/0004	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Seek W/O Seek Option		/0005	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2
Seek With Seek Option		/0005	D0000 (branch) D0025 (LIBF)	D0000 (branch) D0900+5 (LIBF)	D0004	D0004	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	D0918	D0918+1	D0900+1	D0900+3		D0080
	Used	X	X	X	X		X

Significant variables:

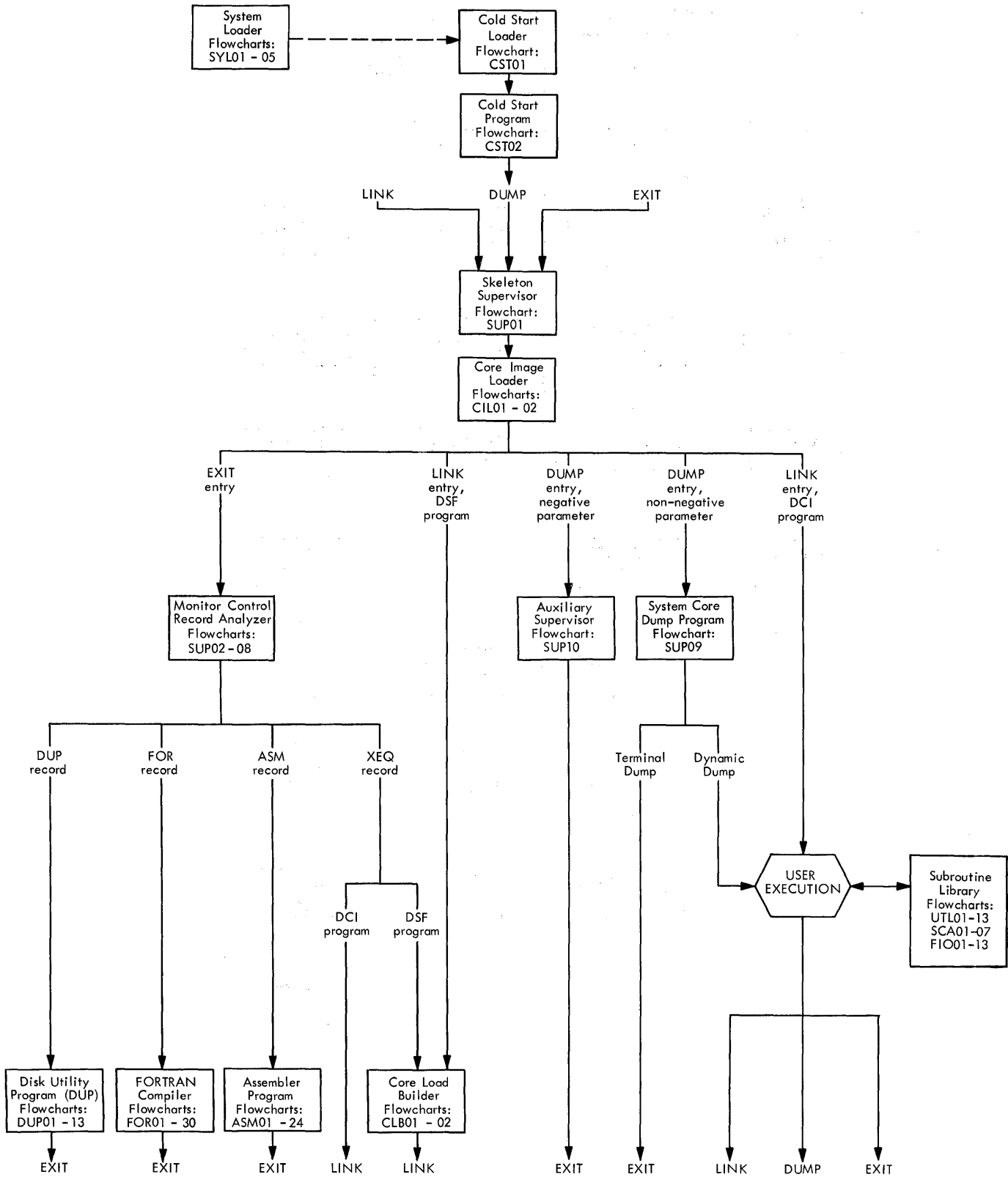
Symbolic location	Contents/Use
D0020	If DISKN was entered by a monitor system program at D0000, this word is used by DISKN to simulate a LIBF link word. The contents of this word should reference the simulated LIBF at D0010-1.
D0025+2	Contents of the LIBF link word if called by a user-written LIBF.
D0010 and D0010+1	Simulated LIBF parameters for direct branch input: D0010 = /1100 for read input = /2200 for write input D0010+1 = address I/O area
DISKN is capable of executing 5 drives simultaneously. Reference to the proper disk drive work areas is accomplished by use of a table. XR1 is used to point to the relative starting position for each specific drive in the table. The starting address in the table is computed as follows:	

Significant variables: (Continued)

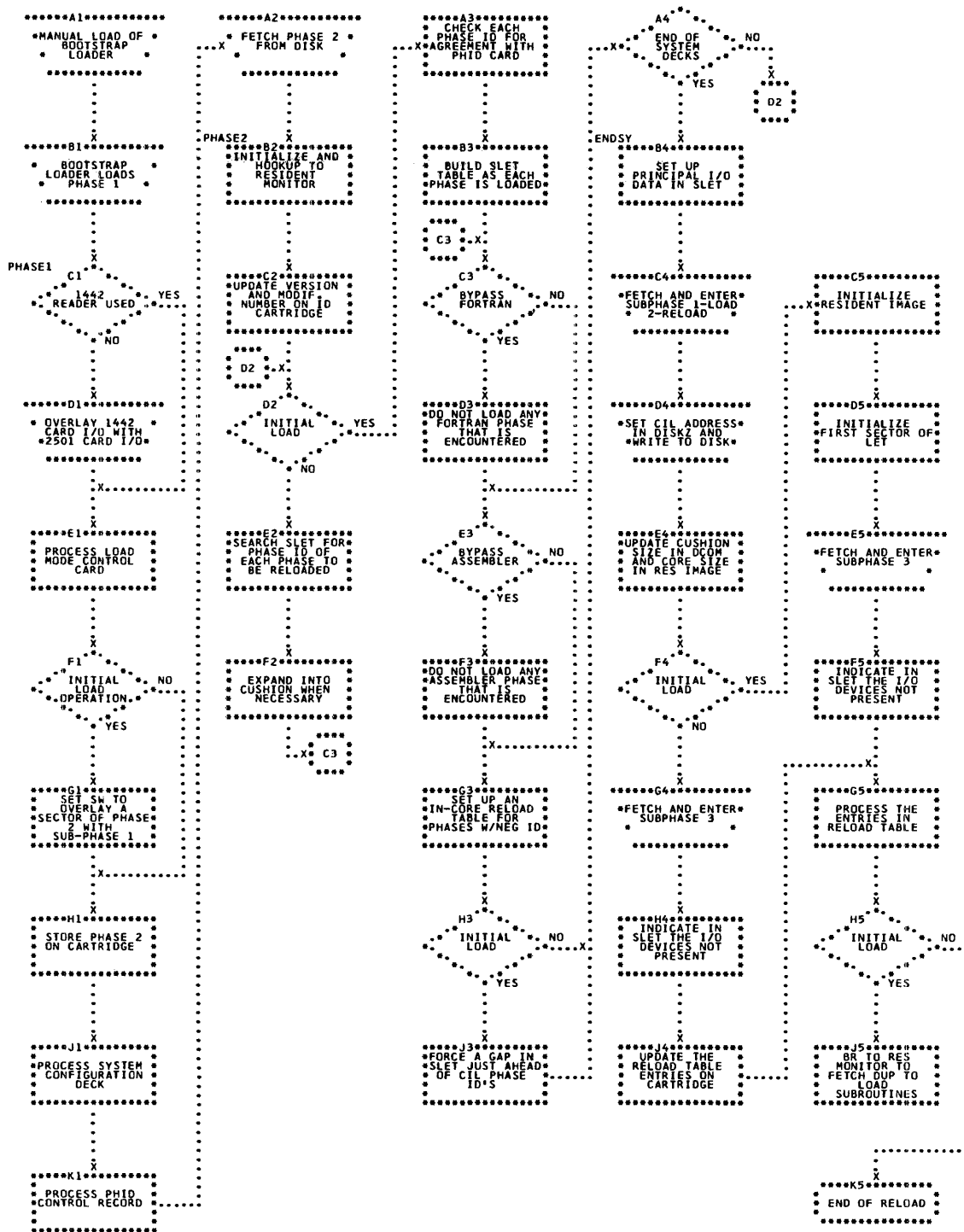
Symbolic location	Contents/Use
	<p>The contents of location D0800 + (drive code x 2)</p> <p>Example: Assume drive 1 is referenced. The address would be: $D0800 + (1 \times 2) = D0800+2$</p> <p>XR1 contains this computed address. References to specific significant variables used in the routine is X1 + some displacement value. The table (DT) contains many significant variables concerning the referenced drive.</p>
VWXYZ and VWXYZ+1	First and second words of Seek IOCC if one is required; also used as temporary storage.
STOX1+1	Drive determination: the drive currently being referenced may be determined by the contents of XR1 corresponding to the relative starting address in the drive table, or by examining the contents of STOX1+1. This location contains the drive code.
Index Register 2	Drive code and I/O area address at various times during execution of this routine.
D0800 (XR1 + 1)	Current position (sector address) of the heads on the referenced drive.
SVIOI and SVIOI+1 (XR1 + 10) (XR1 + 11)	Last two words of the I/O area just read into
RSTRT (XR1 + 20)	Address of the user's error routine.
SFLAG (XR1 + 21)	Current seeking status of the drive: /FFFF = drive is seeking, has just seeked, or seek is required. /0000 = seek not required, or not in process.
D0995 (XR1 + 30)	Originally requested user function.
SCNTR (XR1 + 31)	Error counter for referenced drive for this operation: $50_{10} - C(DT + 31) = \text{total errors occurred for this drive during requested function}$ (DT + 30)
IOCC and IOCC+1 (XR1 + 40) (XR1 + 41)	IOCC for the current user-requested operation, except for Seek and Sense DSW.
ORIGS (XR1 + 50)	Originally requested sector address.
SADR (XR1 + 51)	Current sector address for current IOCC.

Significant variables: (Continued)

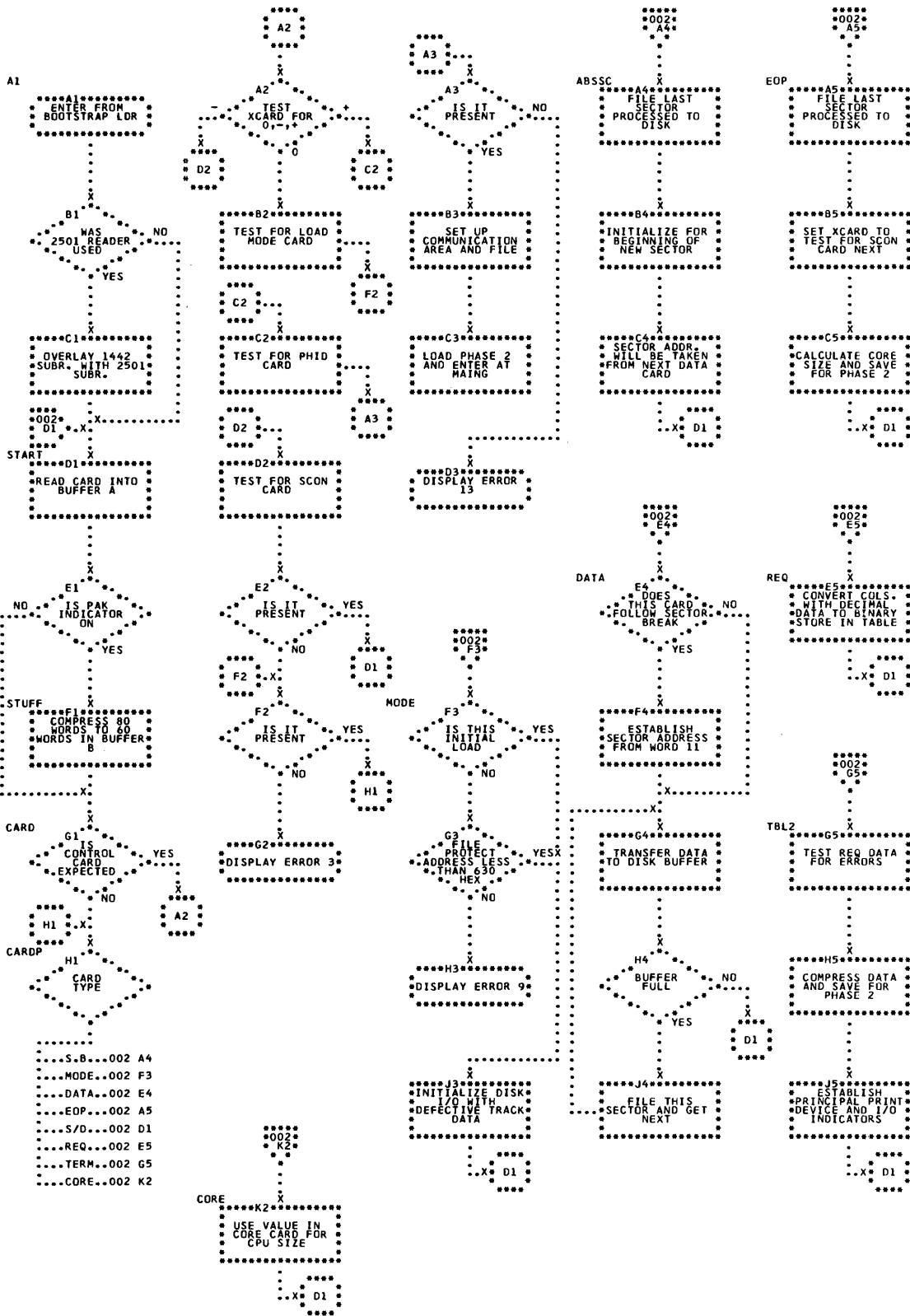
Symbolic location	Contents/Use
WCNT (XR1 + 60)	Remaining word count to be read or written.
COUNT (XR1 + 61)	Word count to be used in next I/O operation.
IOADR XR2 + 70)	Current I/O area address.
RBCNT (XR1 + 71)	Read-back-check counter: $100_{10} = C(DT + 71) = \text{total errors occurred attempting to perform RBC.}$
\$DBSY	Current status of each of the 5 possible drives; i. e. , if drives 0 and 2 are both busy, bits 0 and 2 of \$DBSY are set to 1.



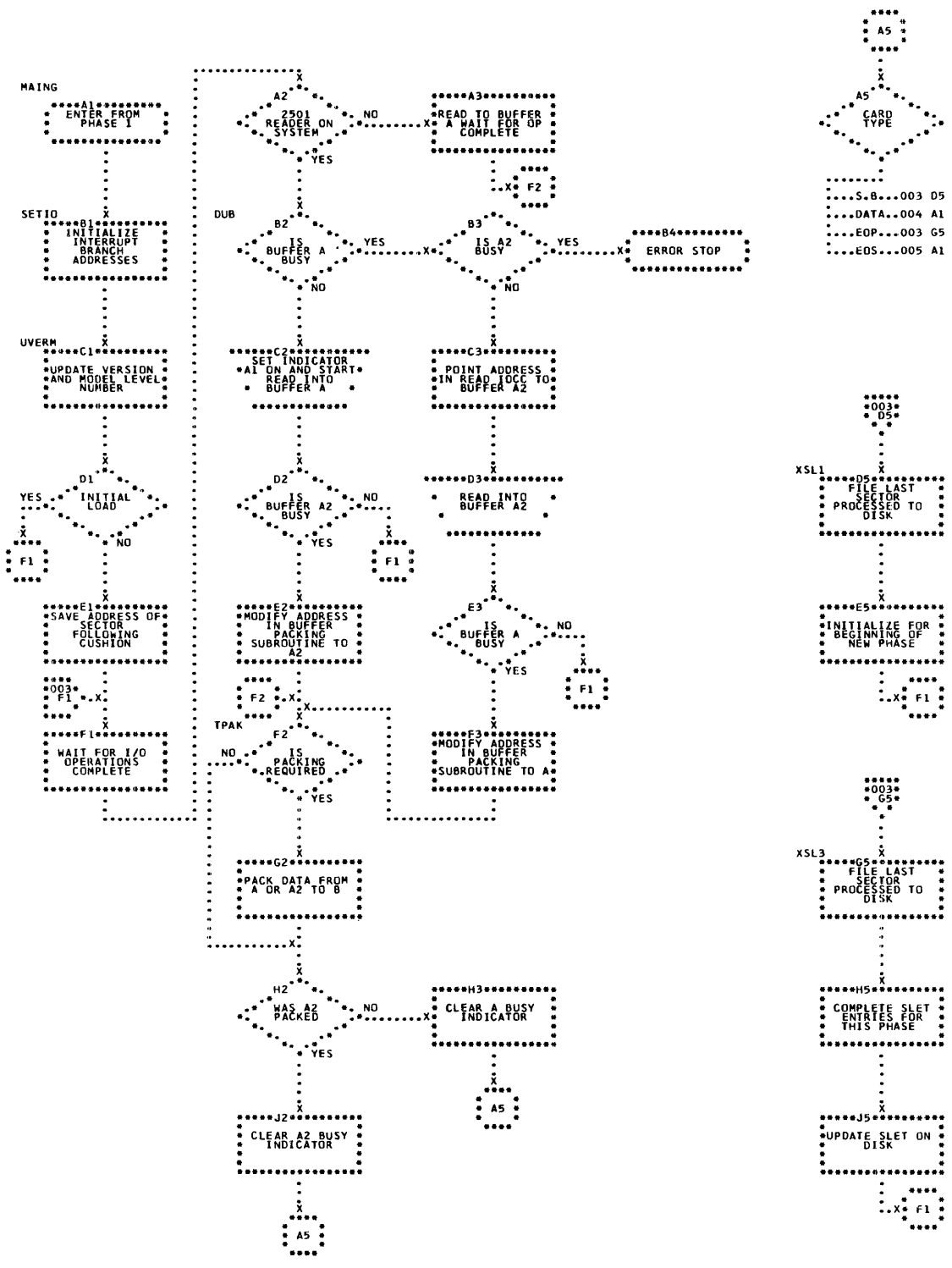
Flowchart DMS01. Disk Monitor System, System Overview



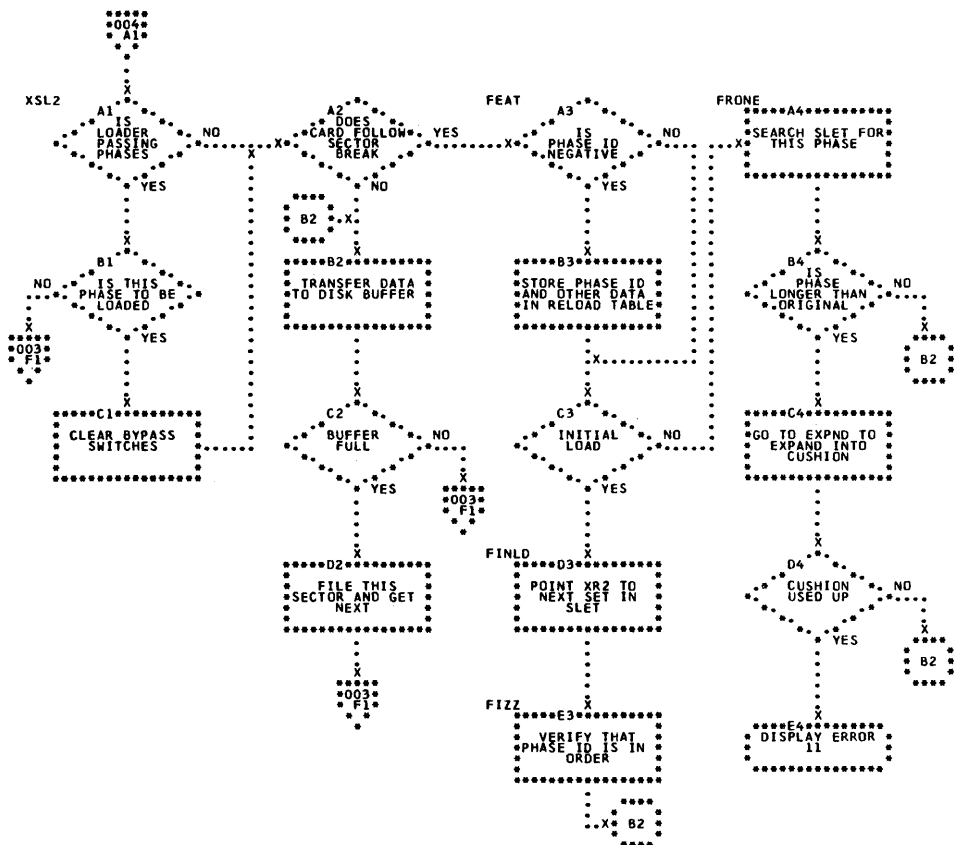
Flowchart SYL01. System Loader, General Flow



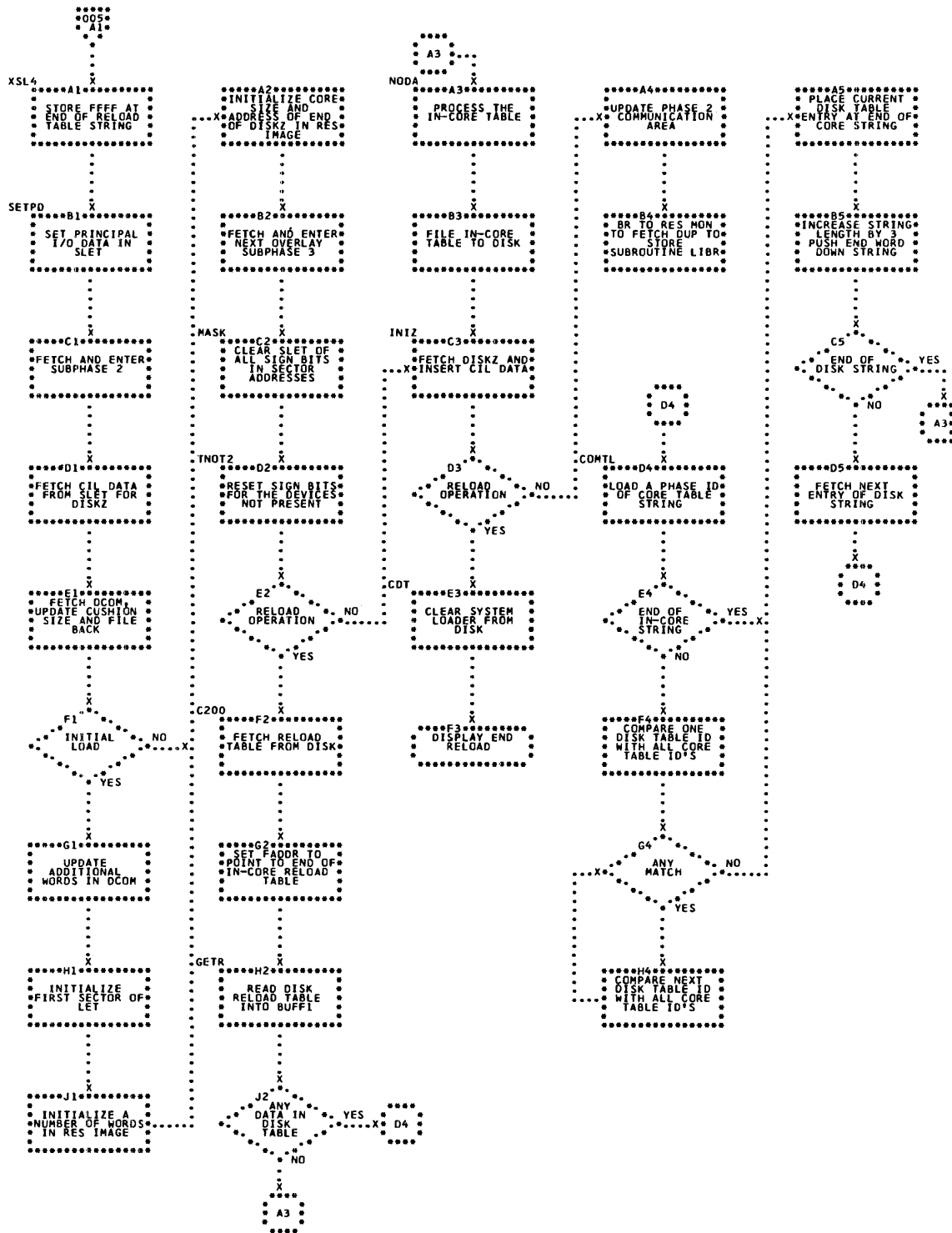
Flowchart SYL02. System Loader, Phase 1



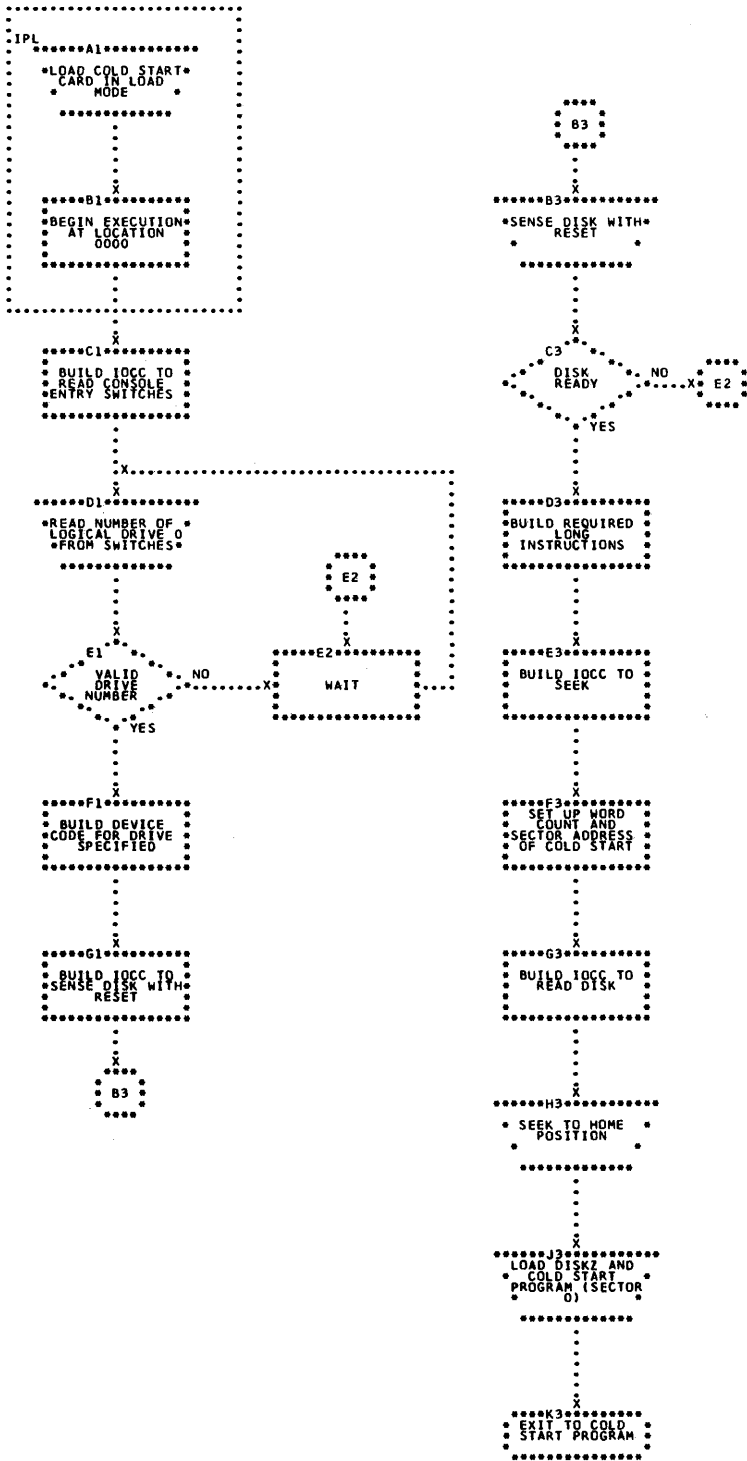
Flowchart SYL03. System Loader, Phase 2



Flowchart SYL04. System Loader, Phase 2



Flowchart SYL05. System Loader, Phase 2



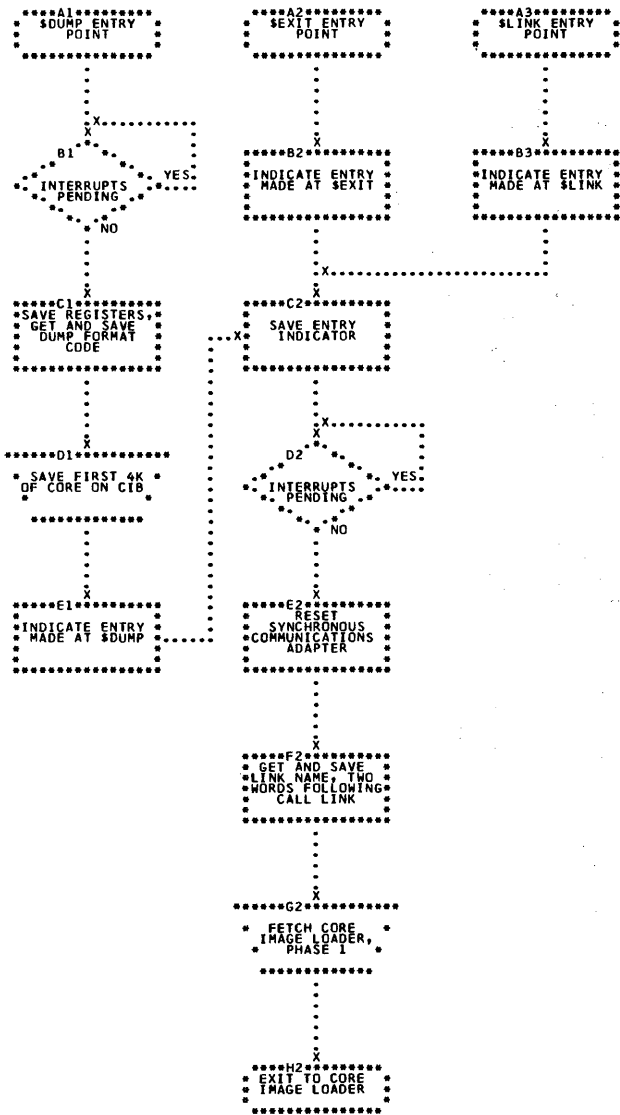
Flowchart CST01. Cold Start Loader

```

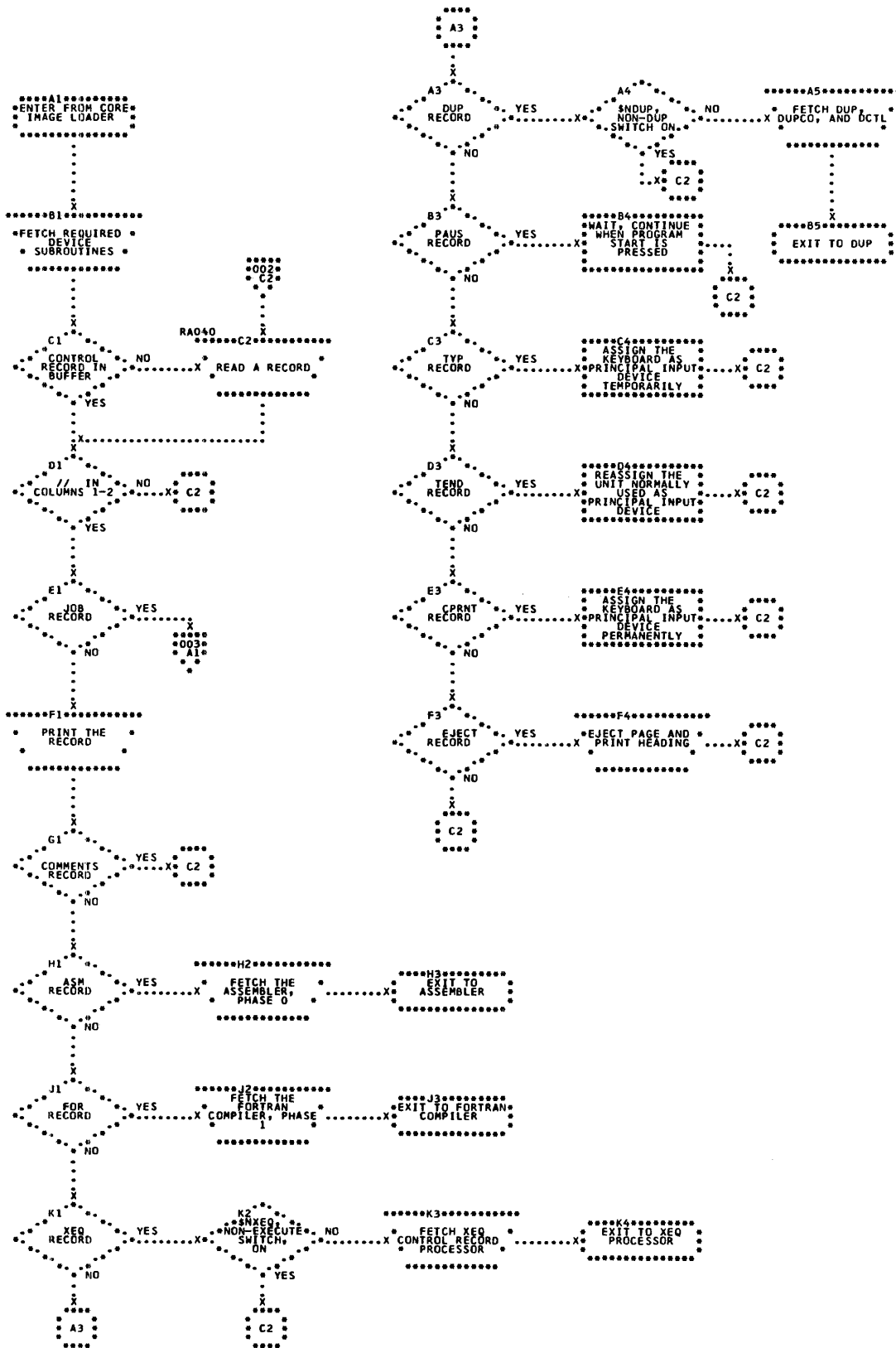
*****A1*****
*ENTER FROM COLD*
*START LOADER*
*****
.
.
.
X
*****B1*****
*GET DEVICE CODE*
*FOR LOGICAL 0*
*SET UP BY COLD*
*START LOADER*
*****
.
.
.
X
*****C1*****
*INITIALIZE*
*LOCATION 0 WITH*
*BRANCH TO $DUMP*
*****
.
.
.
X
*****D1*****
*SET UP WORD*
*COUNT AND*
*SECTOR ADDRESS*
*OF RESIDENT*
*IMAGE*
*****
.
.
.
X
*****E1*****
*INITIALIZE*
*$CYLN ENTRIES*
*FOR LOGICAL 0*
*TO POSITIVE*
*****
.
.
.
X
*****F1*****
*SET $DBSY AND*
*$CYLN TO ZERO*
*****
.
.
.
X
*****G1*****
*FETCH RESIDENT*
*IMAGE*
*****
.
.
.
X
*****H1*****
*INITIALIZE*
*$ACDE ENTRY FOR*
*LOGICAL 0*
*****
.
.
.
X
*****J1*****
*EXIT TO*
*AUXILIARY*
*SUPERVISOR*
*****

```

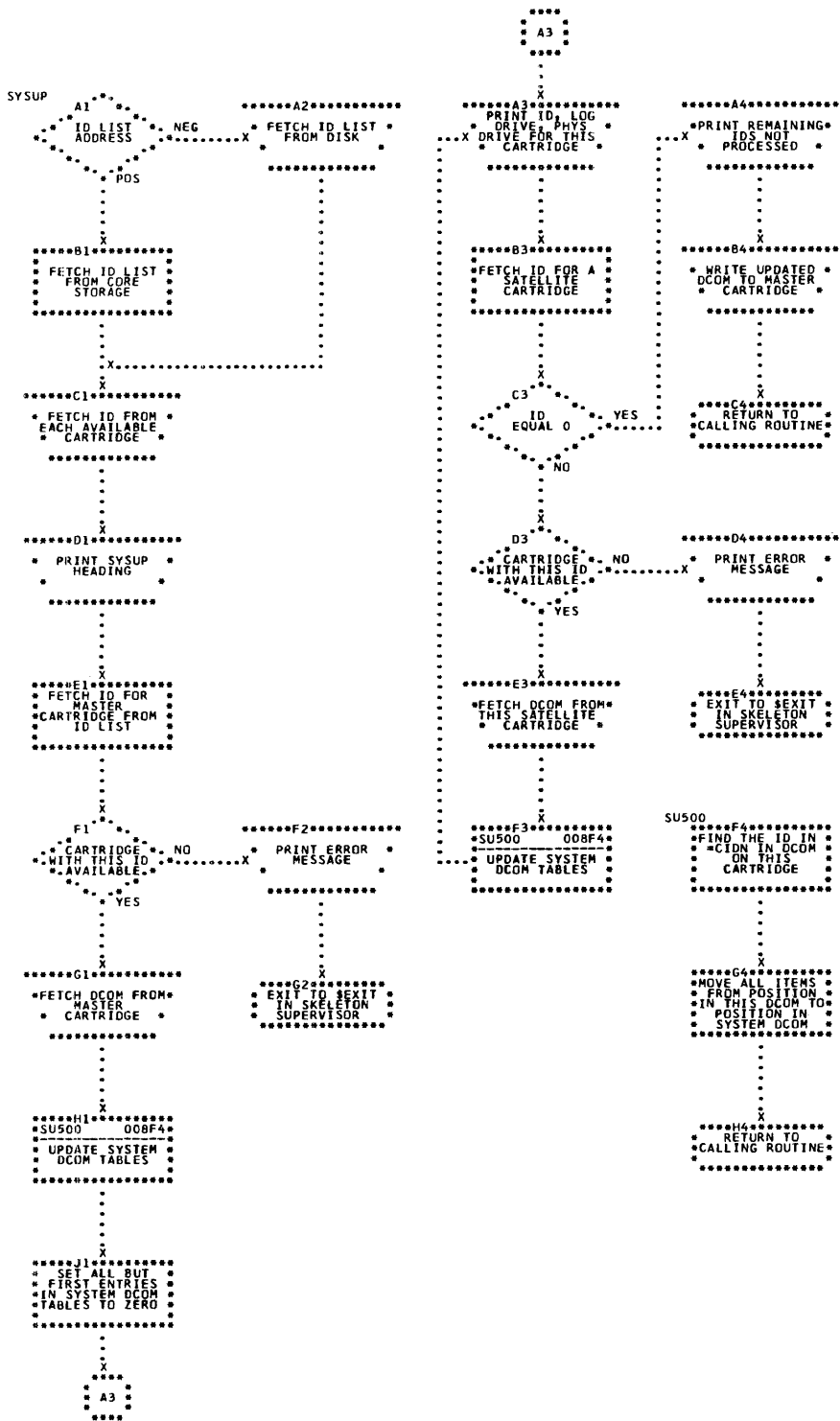
Flowchart CST02. Cold Start Program



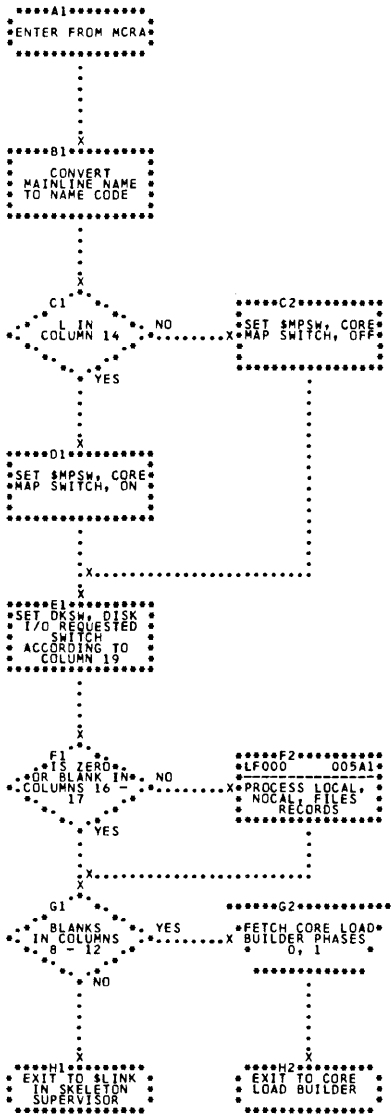
Flowchart SUP01. Supervisor, Skeleton Supervisor



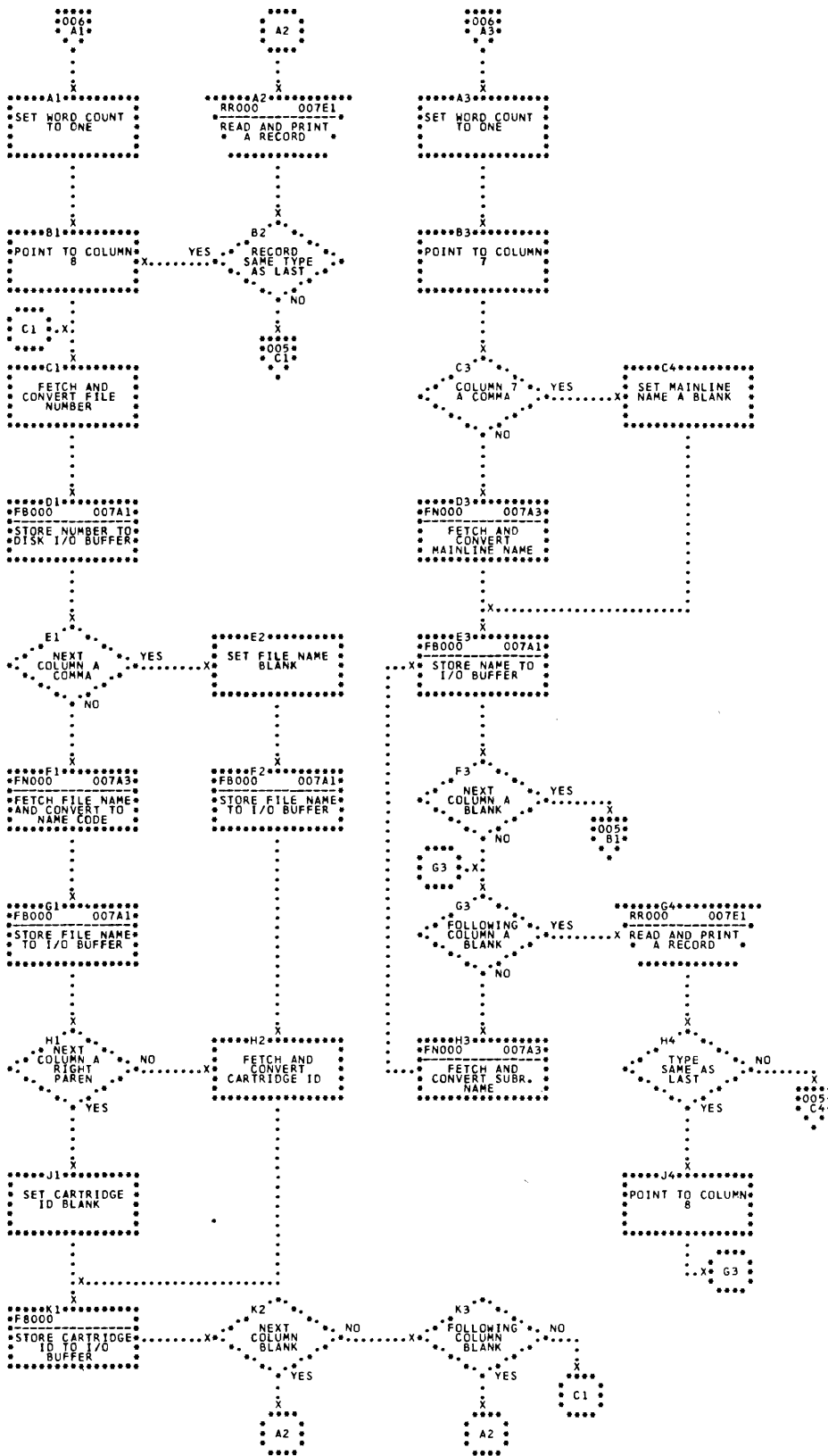
Flowchart SUP02. Supervisor, Monitor Control Record Analyzer



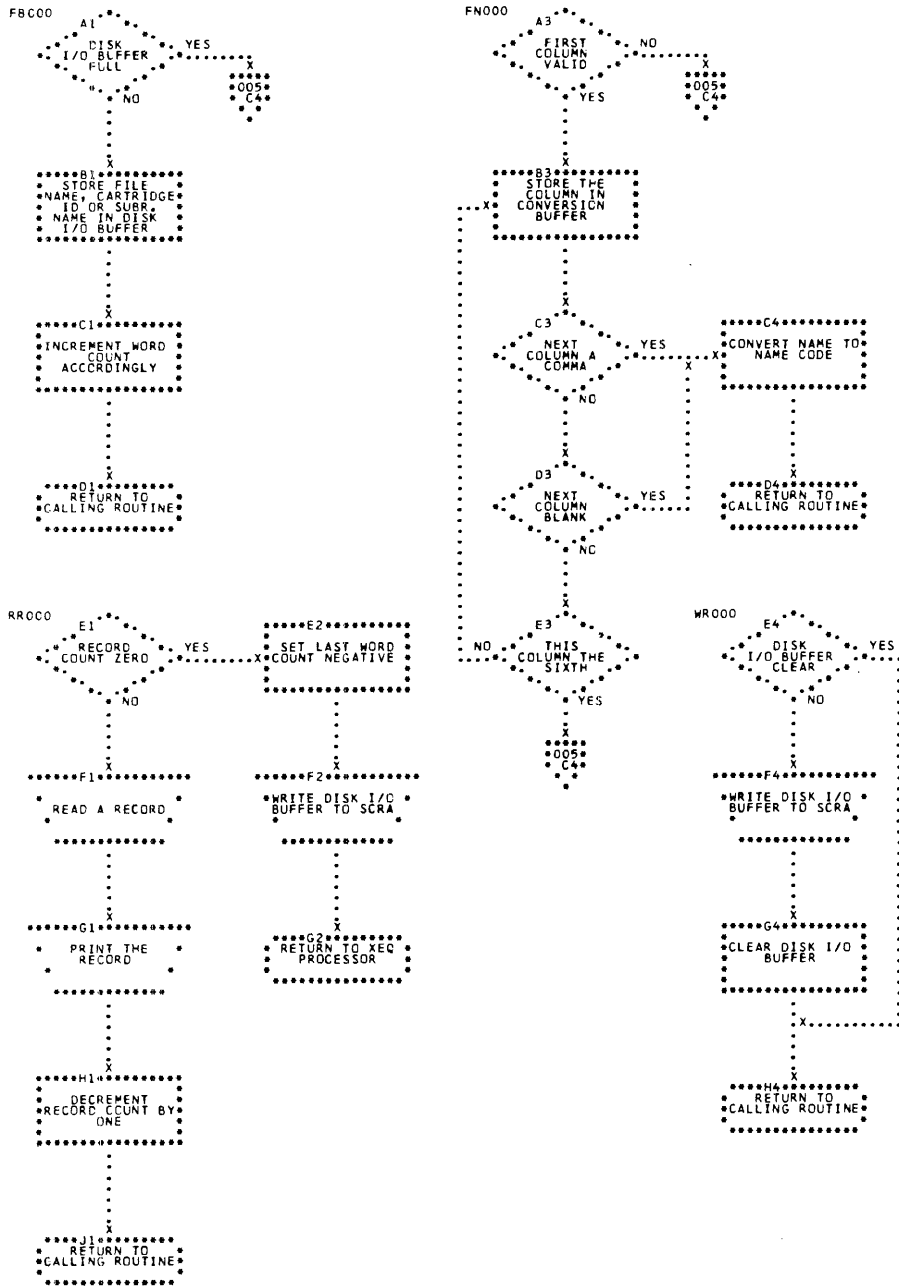
Flowchart SUP04. Supervisor, System Update Program



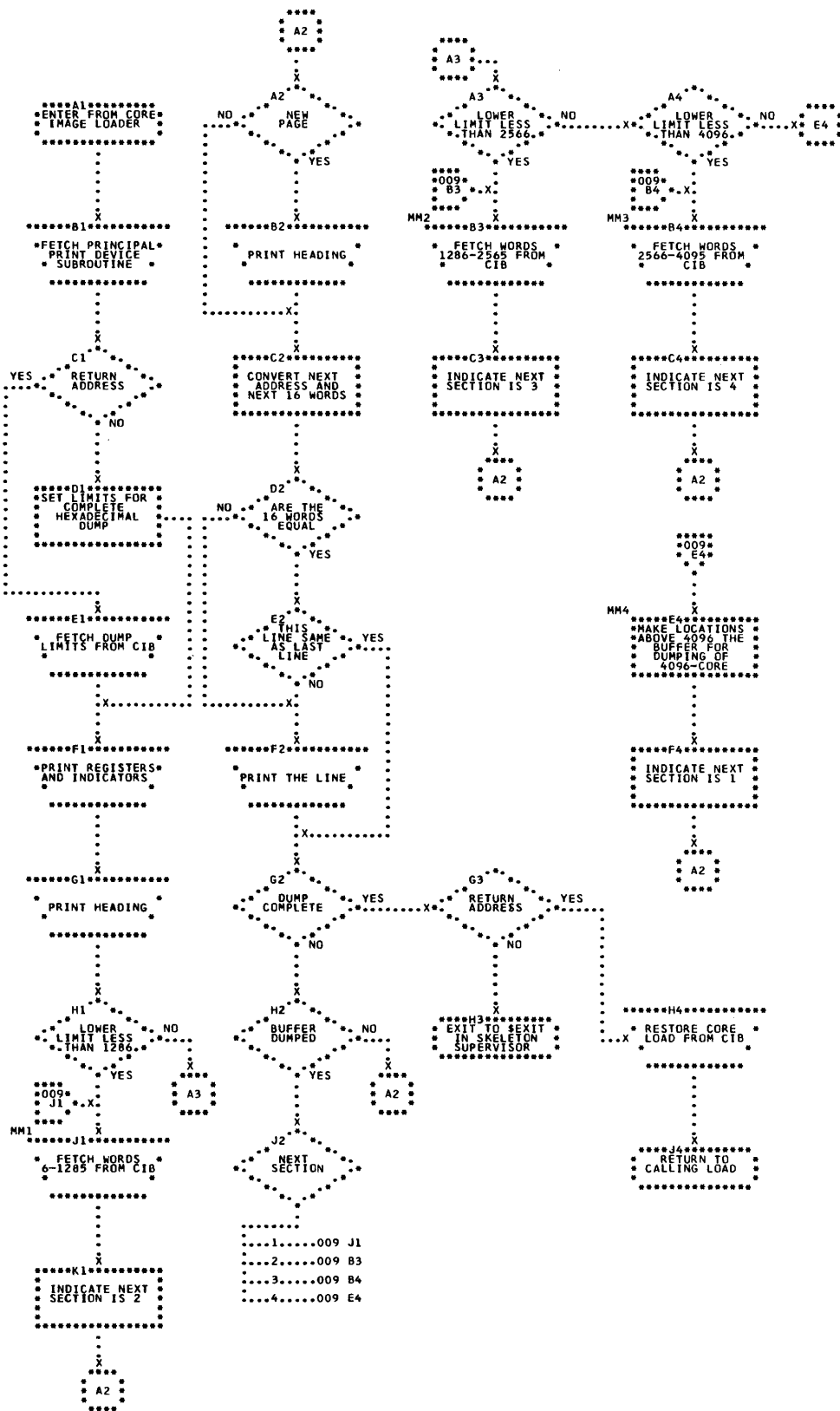
Flowchart SUP05. Supervisor, XEQ Control Record Analyzer



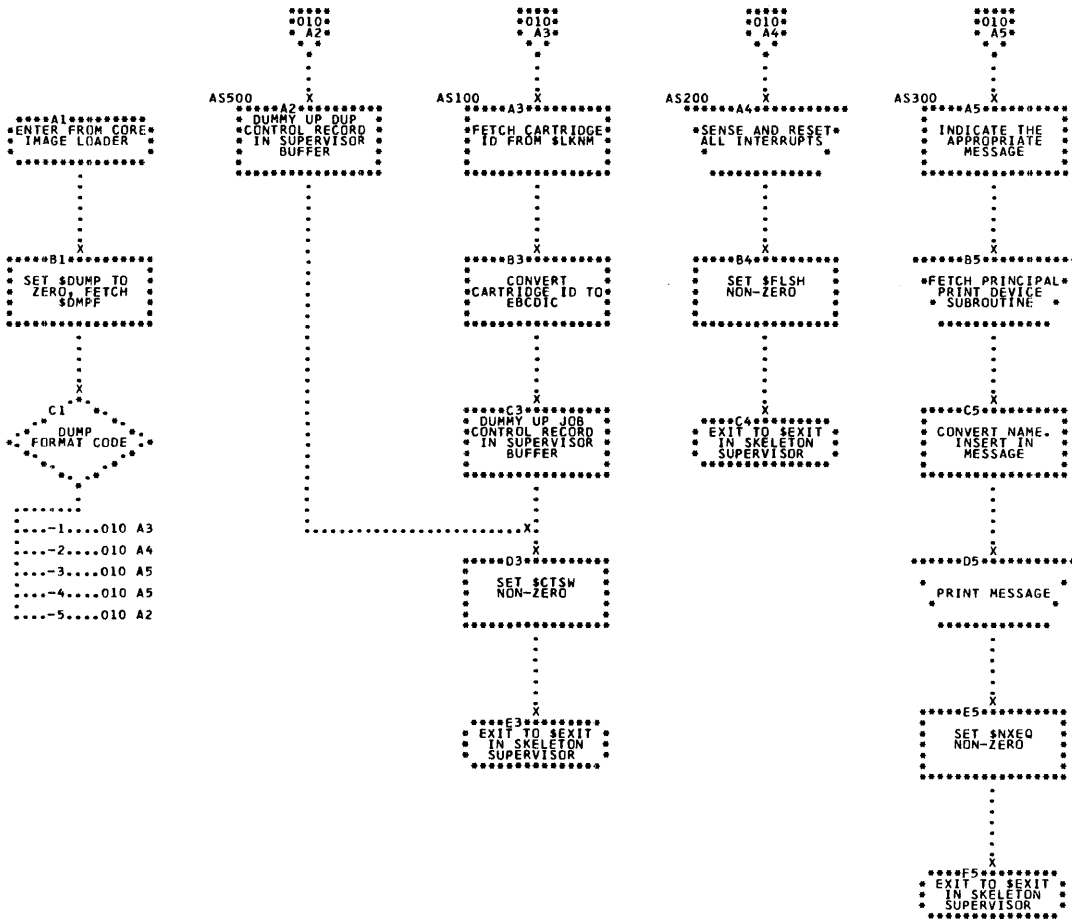
Flowchart SUP07. Supervisor, Supervisor Control Record Analyzer



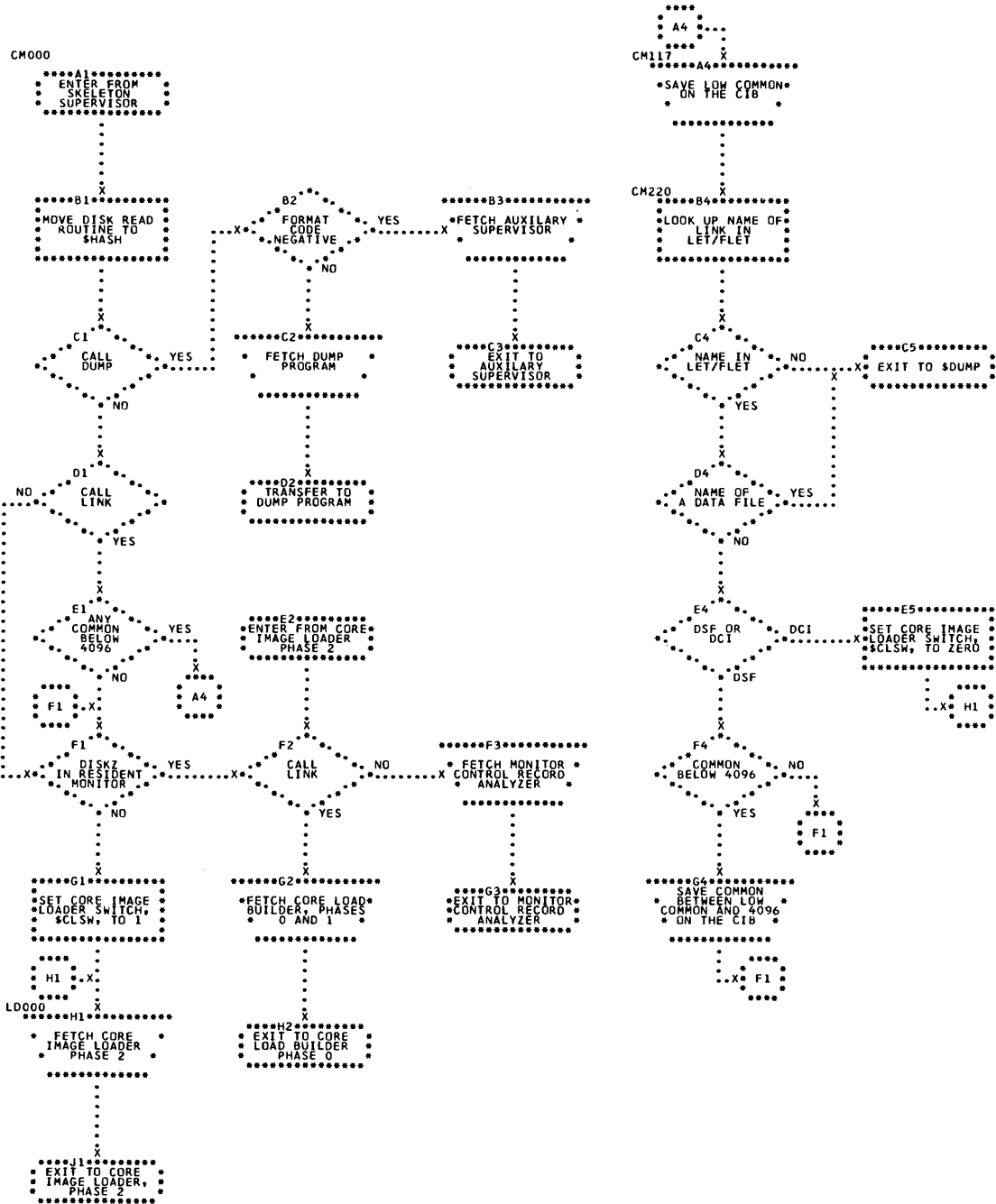
Flowchart SUP08. Supervisor, Supervisor Control Record Analyzer



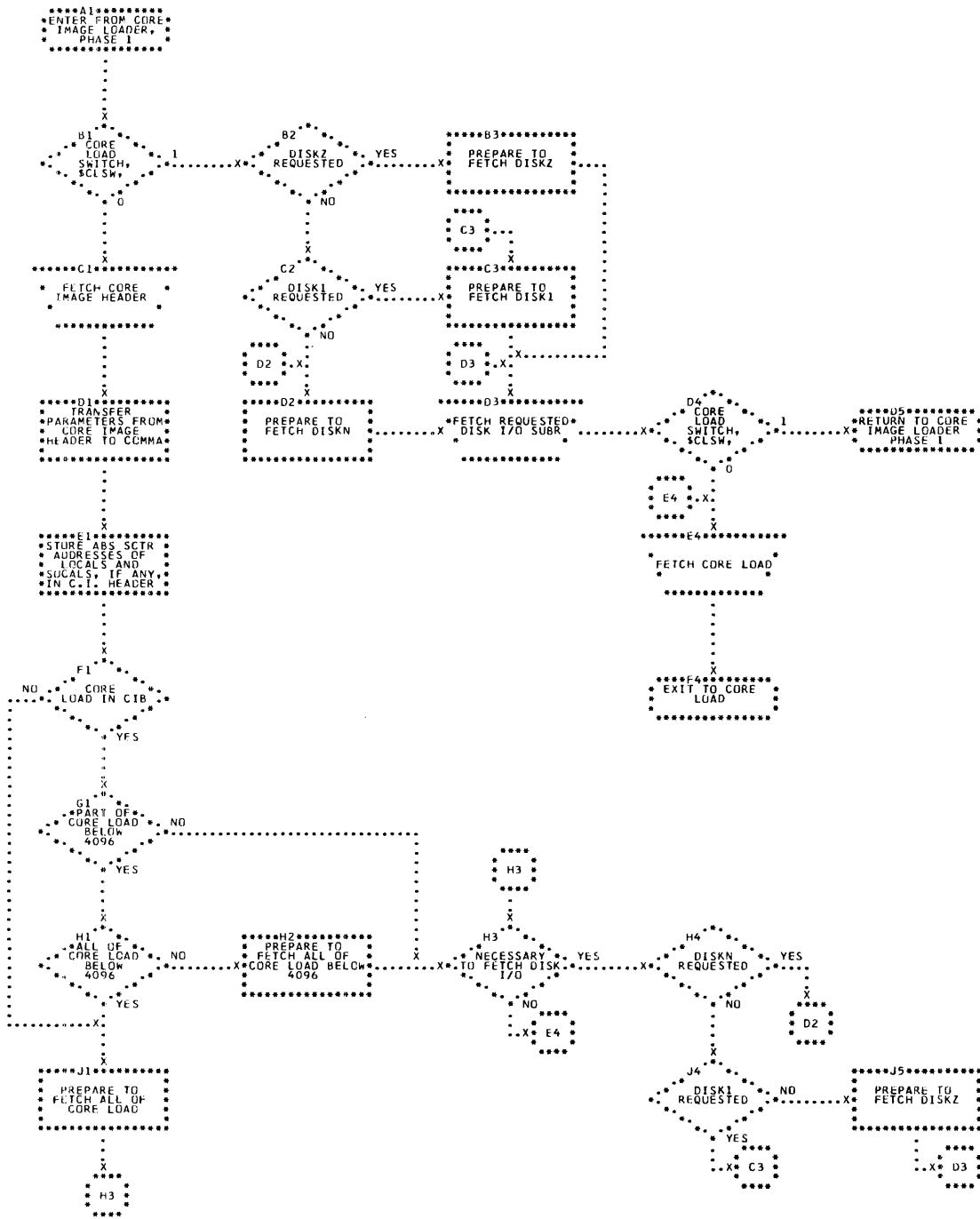
Flowchart SUP09. Supervisor, System Core Dump Program



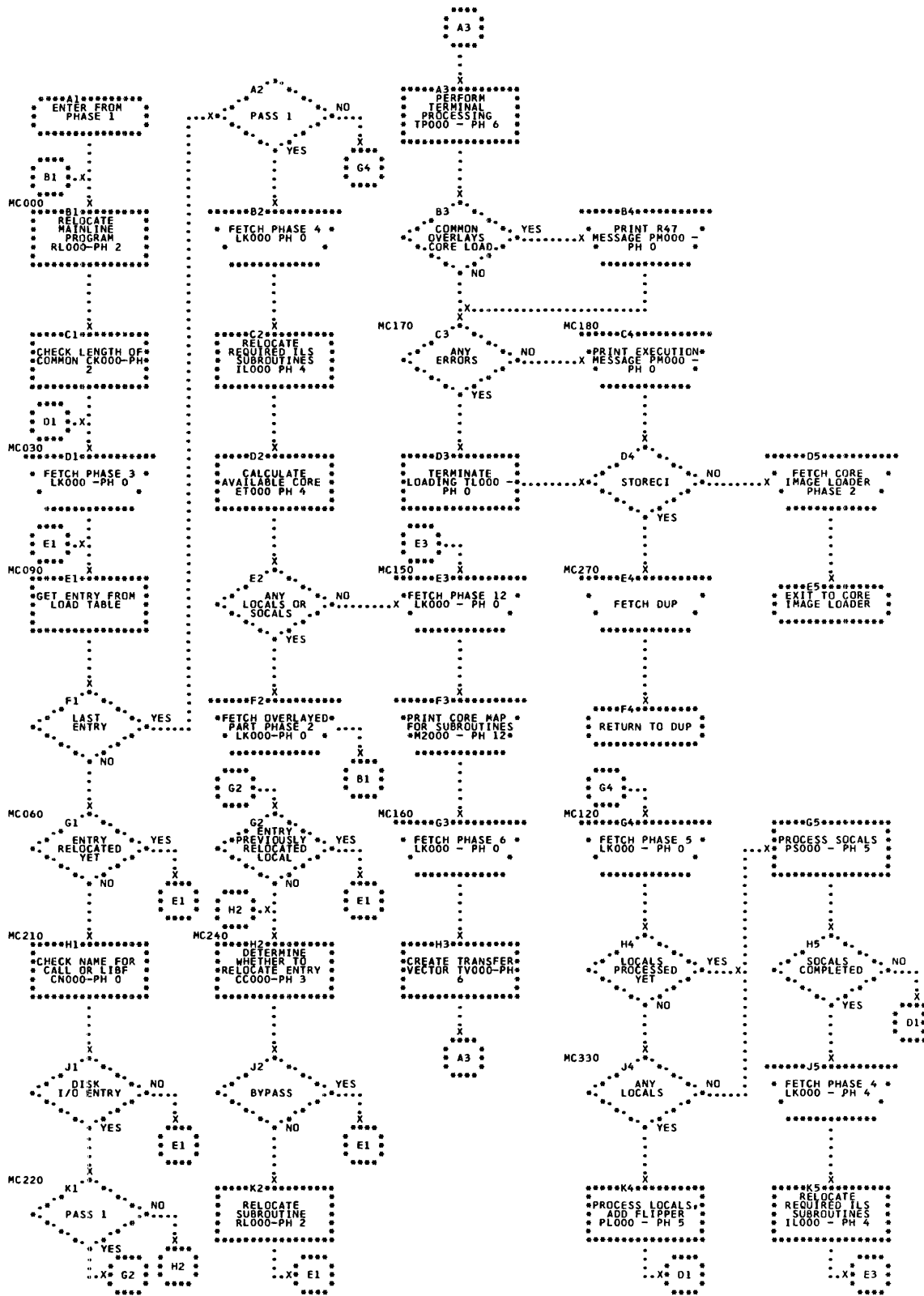
Flowchart SUP10. Supervisor, Auxiliary Supervisor



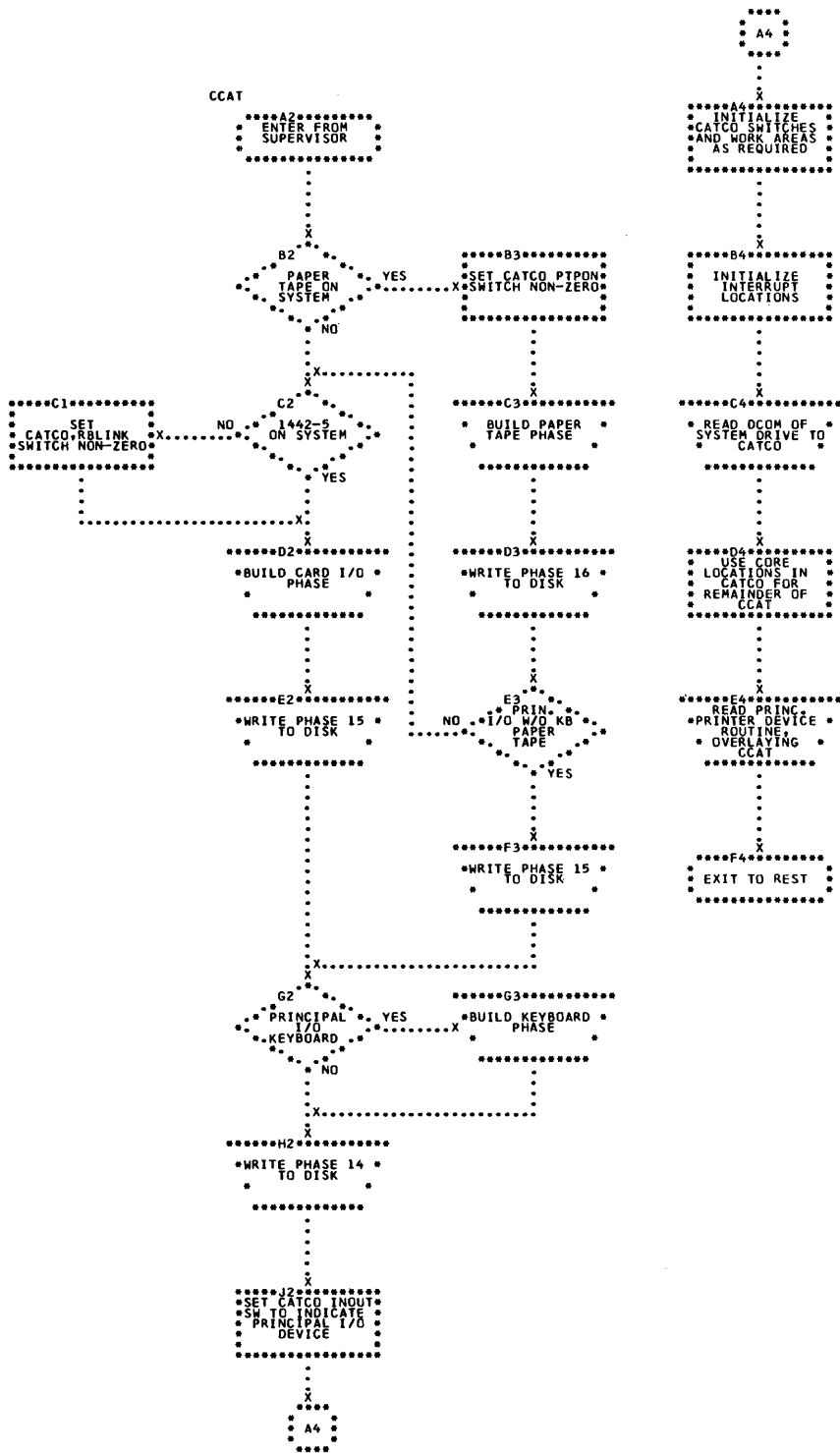
Flowchart CIL01. Core Image Loader, Phase 1



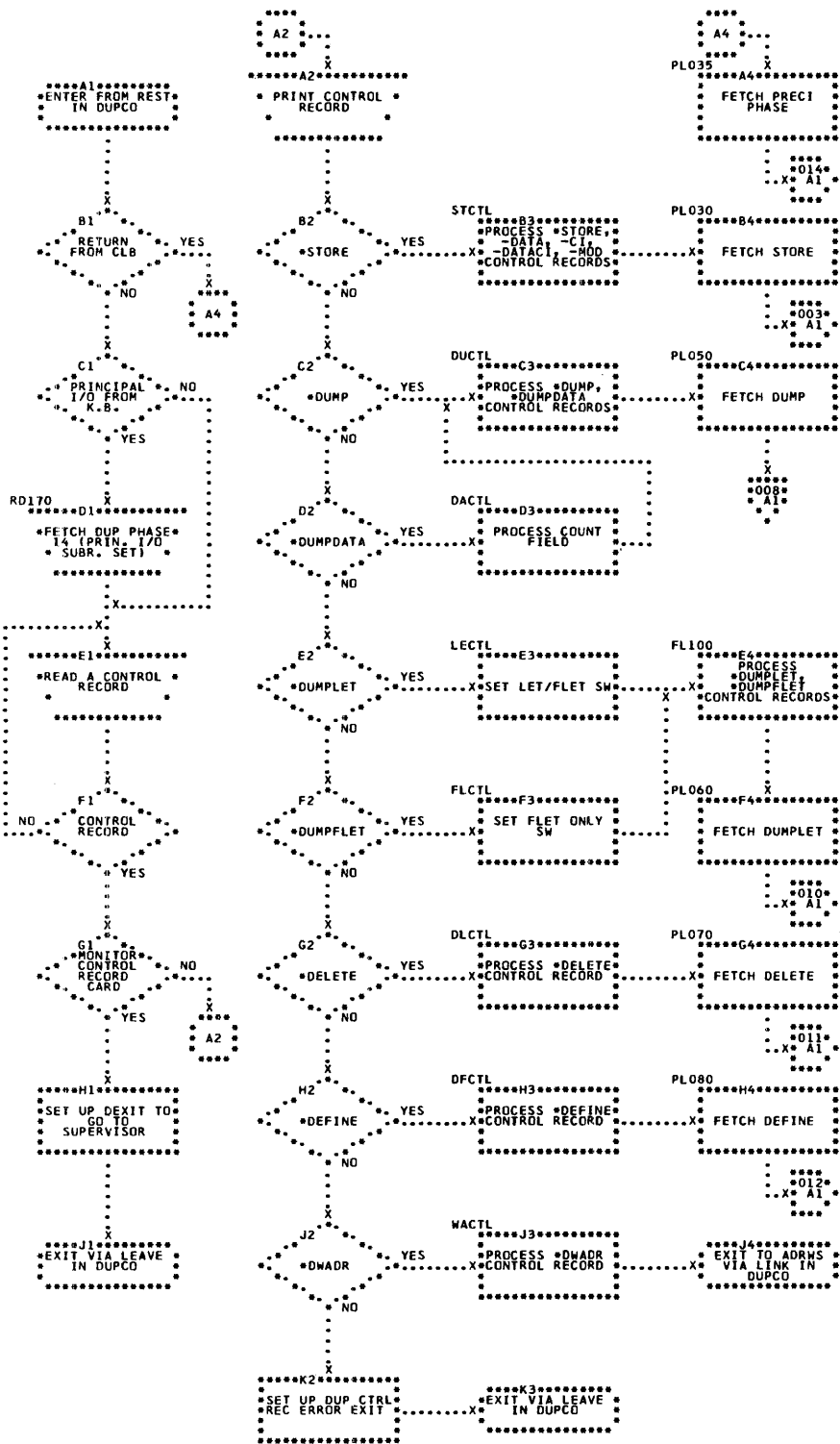
Flowchart CIL02. Core Image Loader, Phase 2



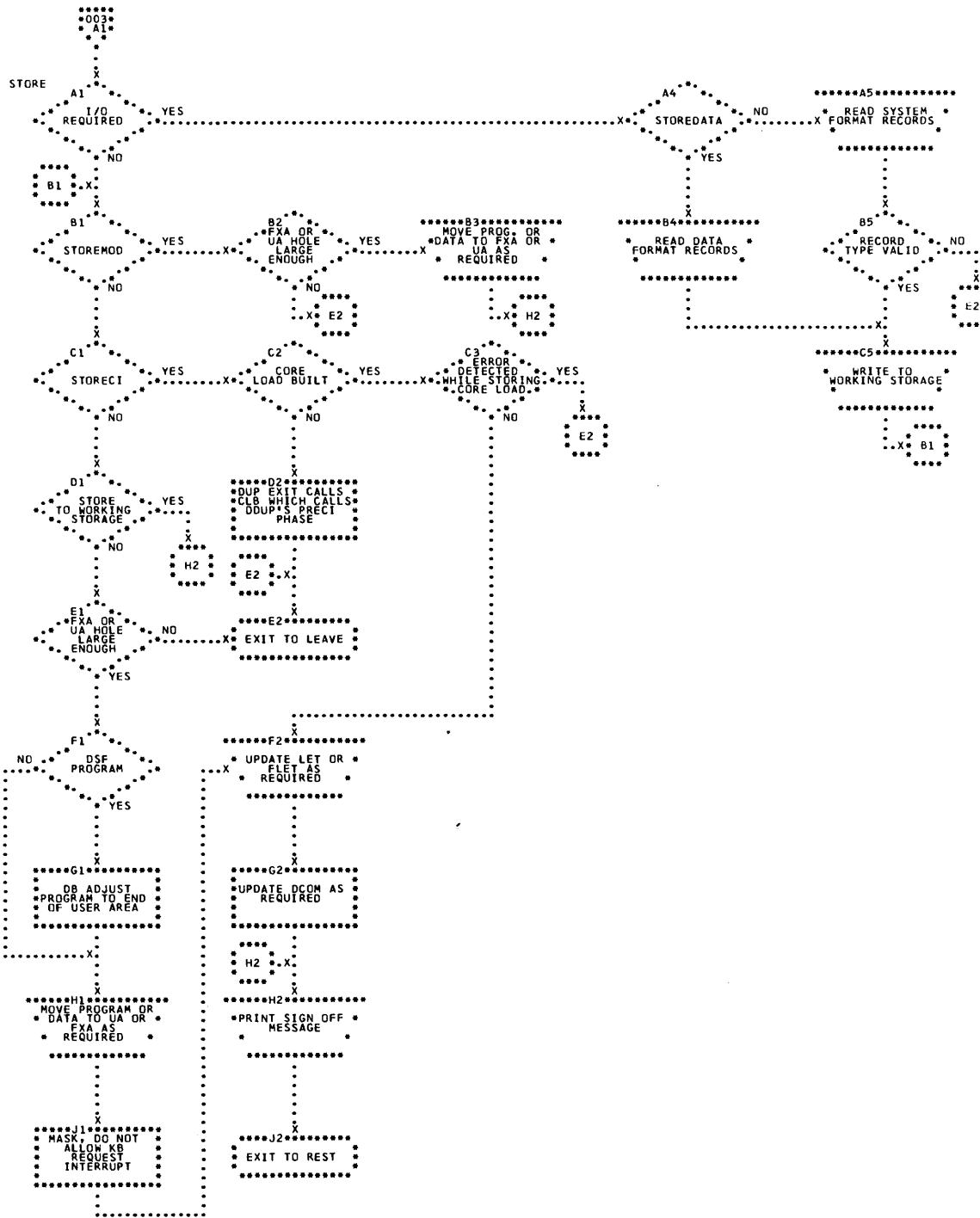
Flowchart CLB02. Core Load Builder, Master Control



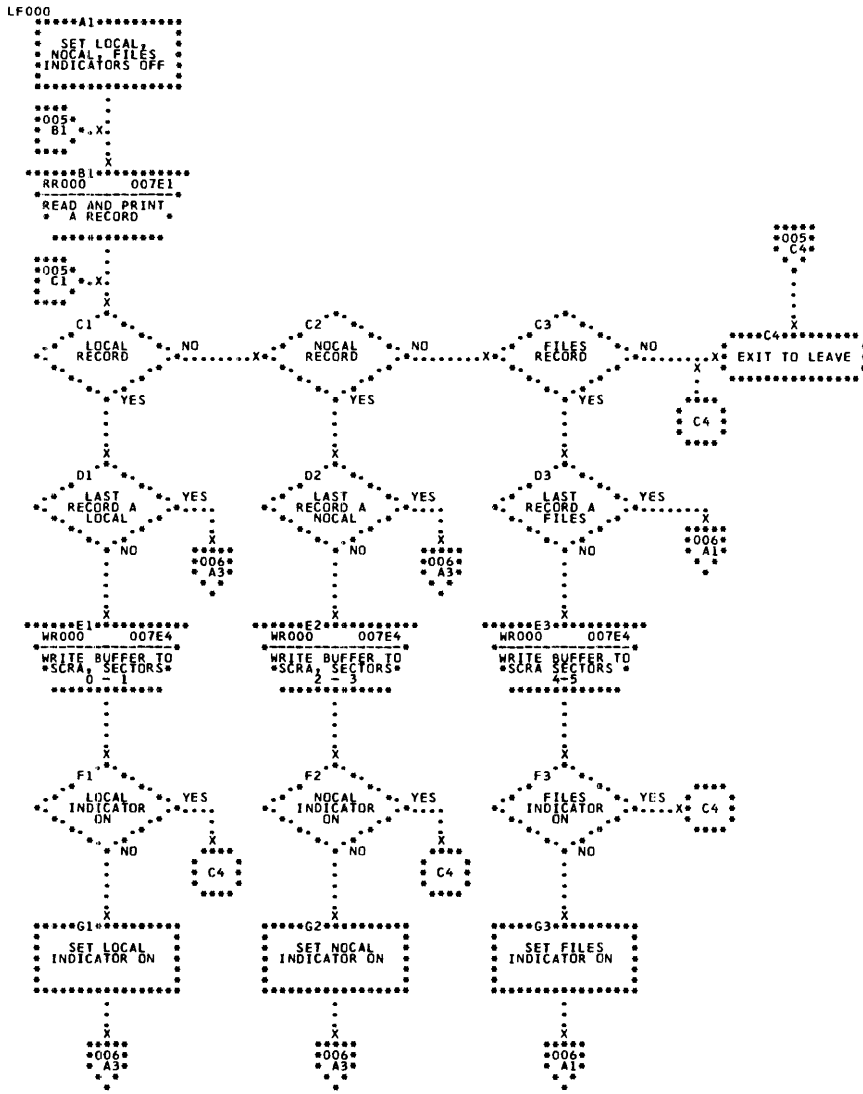
Flowchart DUP01. Disk Utility Program, CCAT



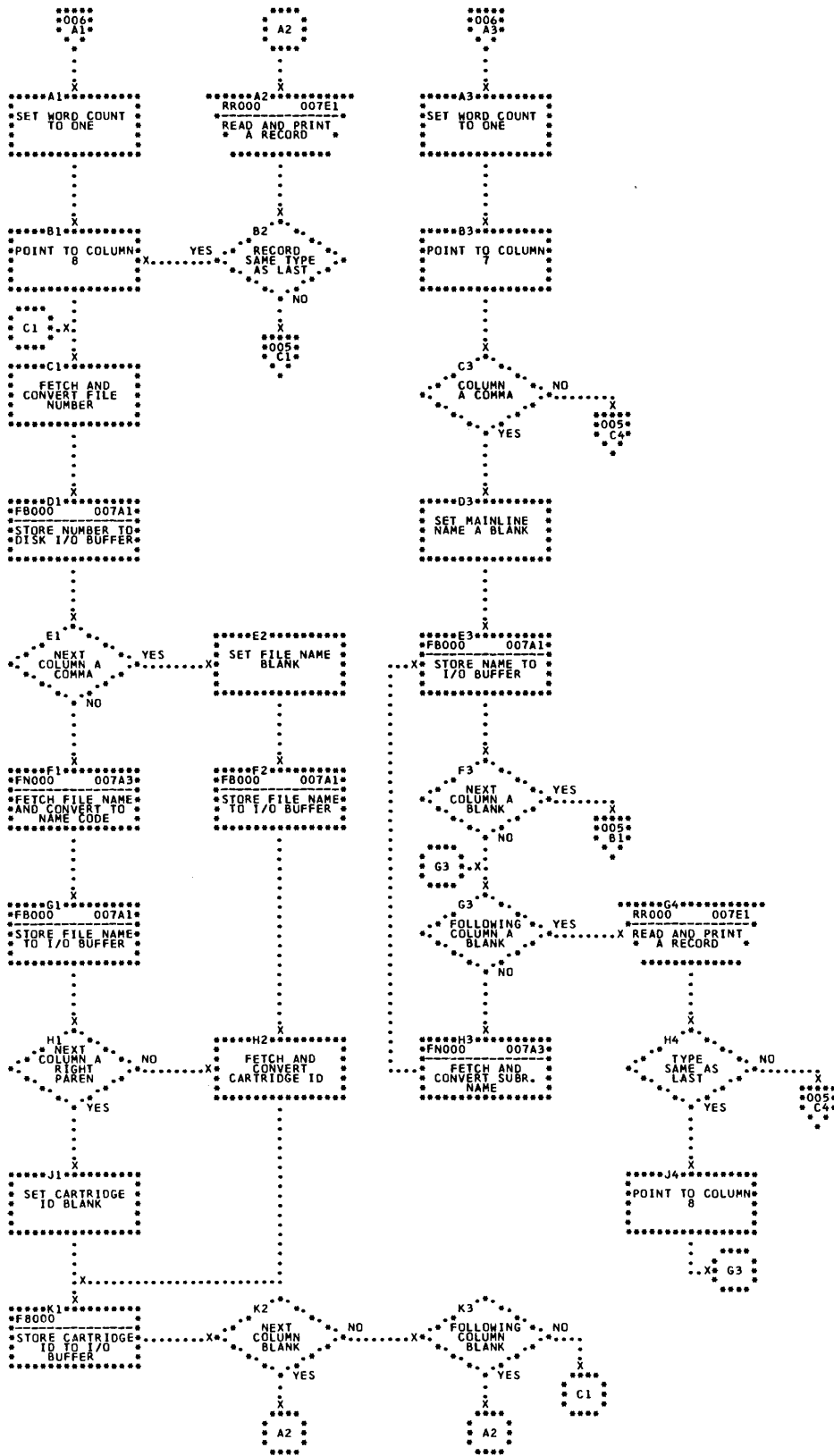
Flowchart DUP02. Disk Utility Program, DCTL



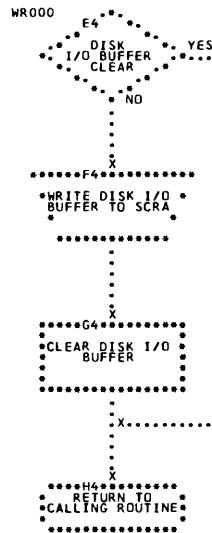
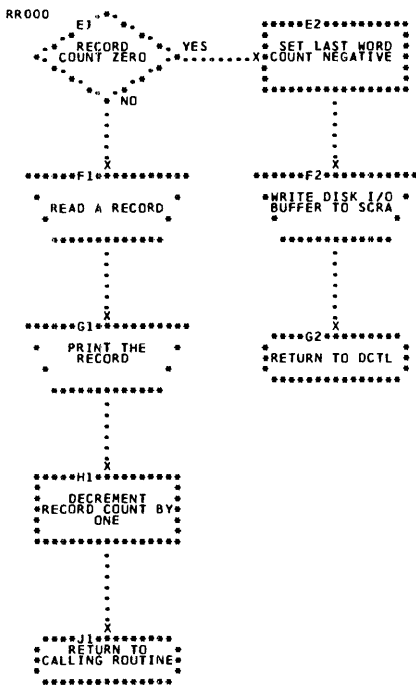
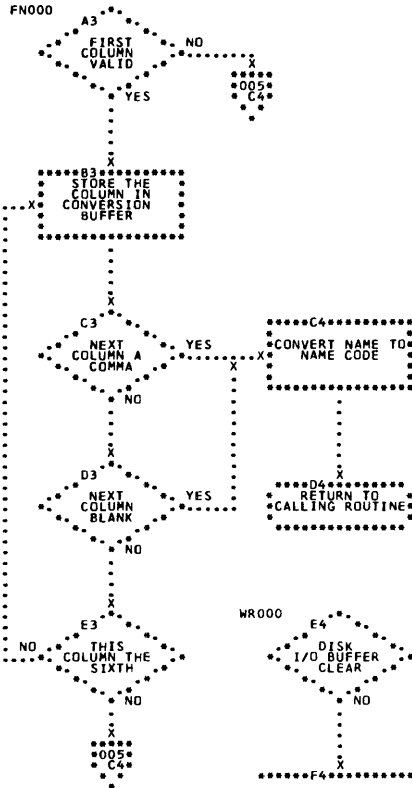
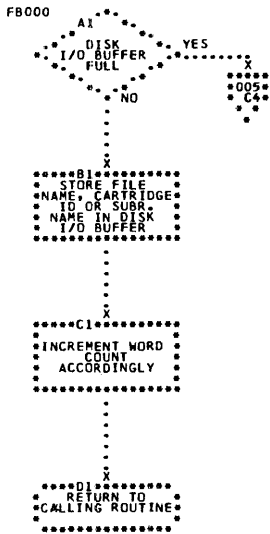
Flowchart DUP03. Disk Utility Program, STORE



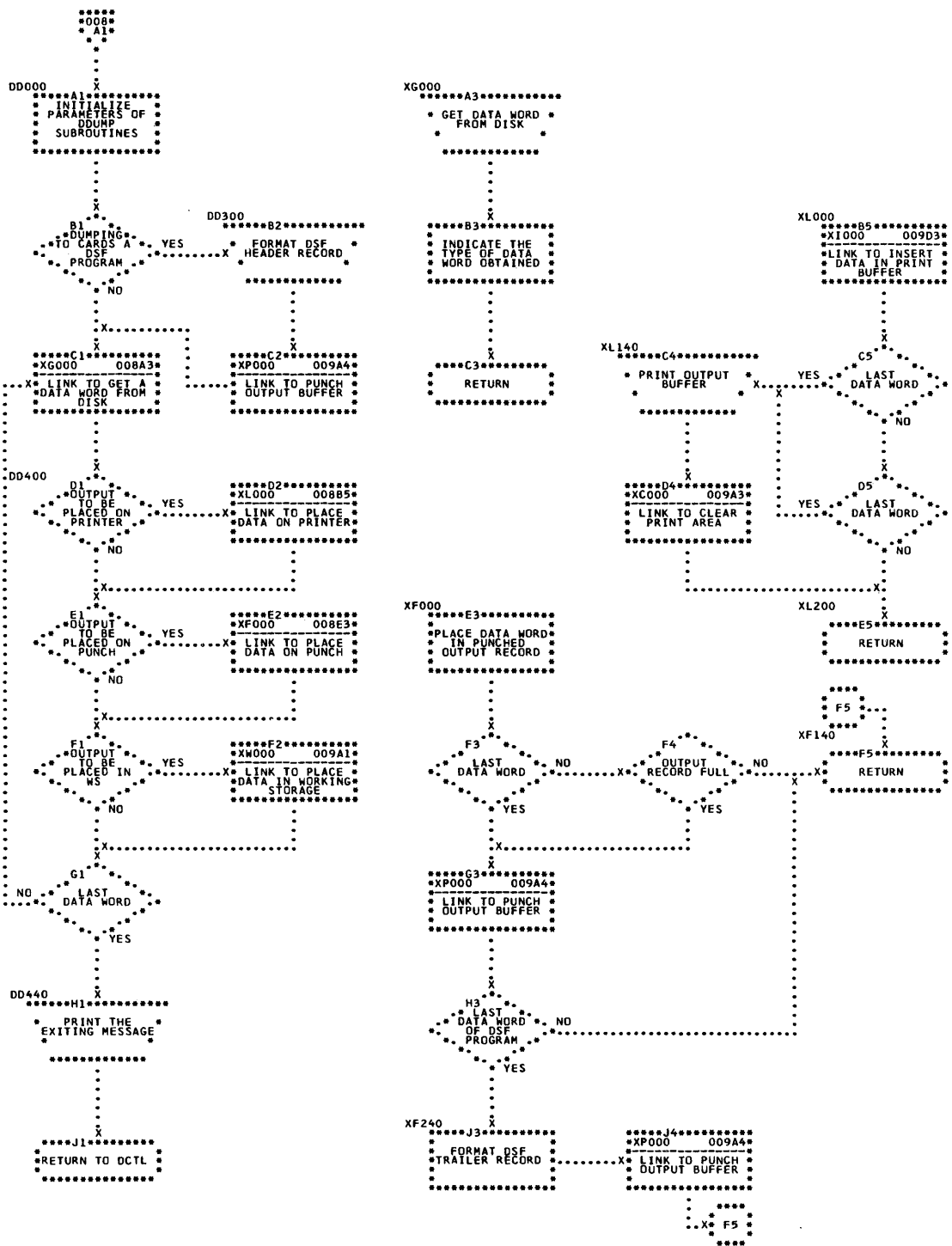
Flowchart DUP04. Disk Utility Program, FILEQ



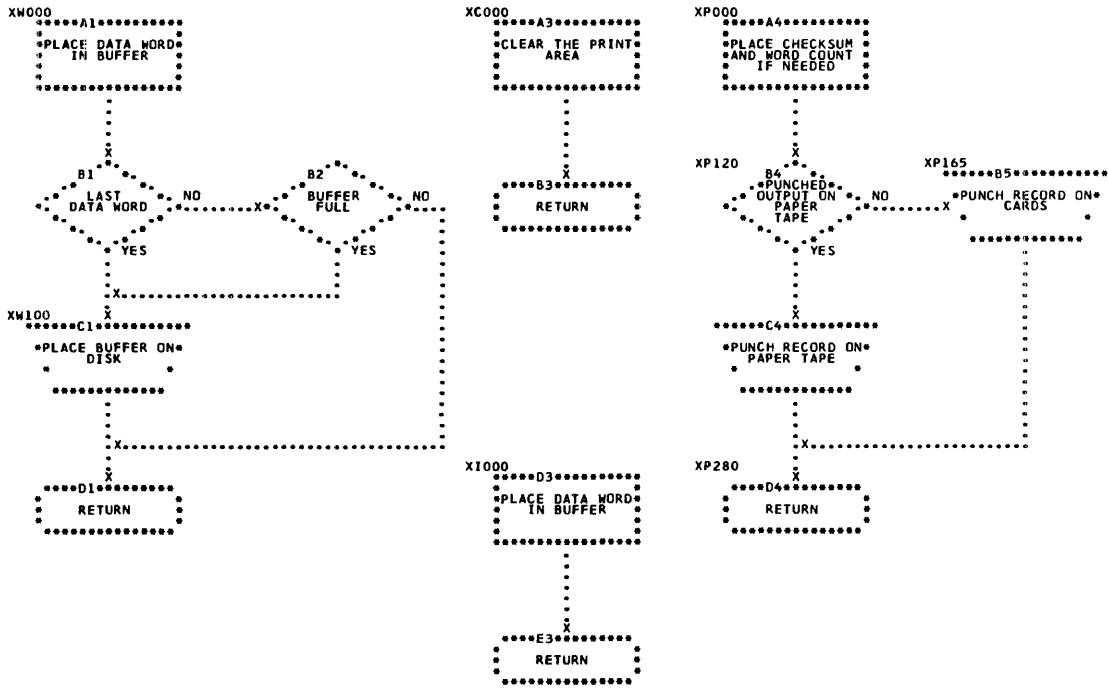
Flowchart DUP05. Disk Utility Program, FILEQ



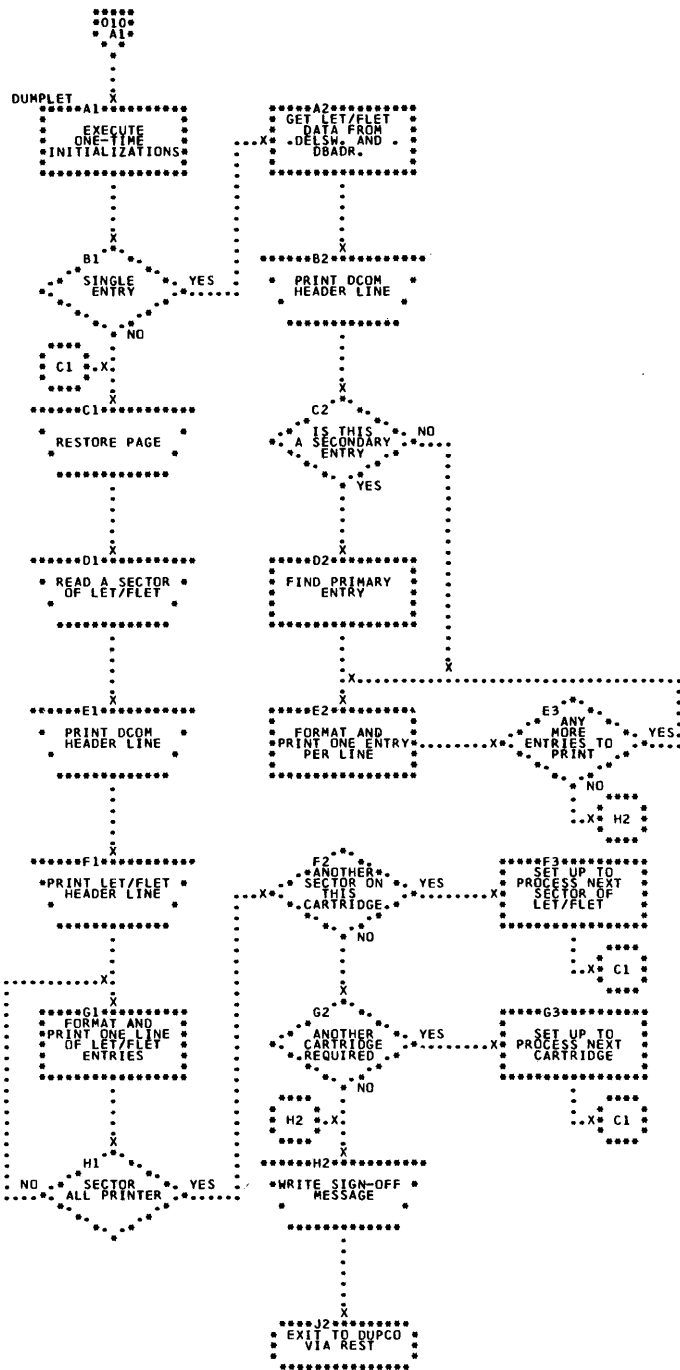
Flowchart DUPO6. Disk Utility Program, FILEQ



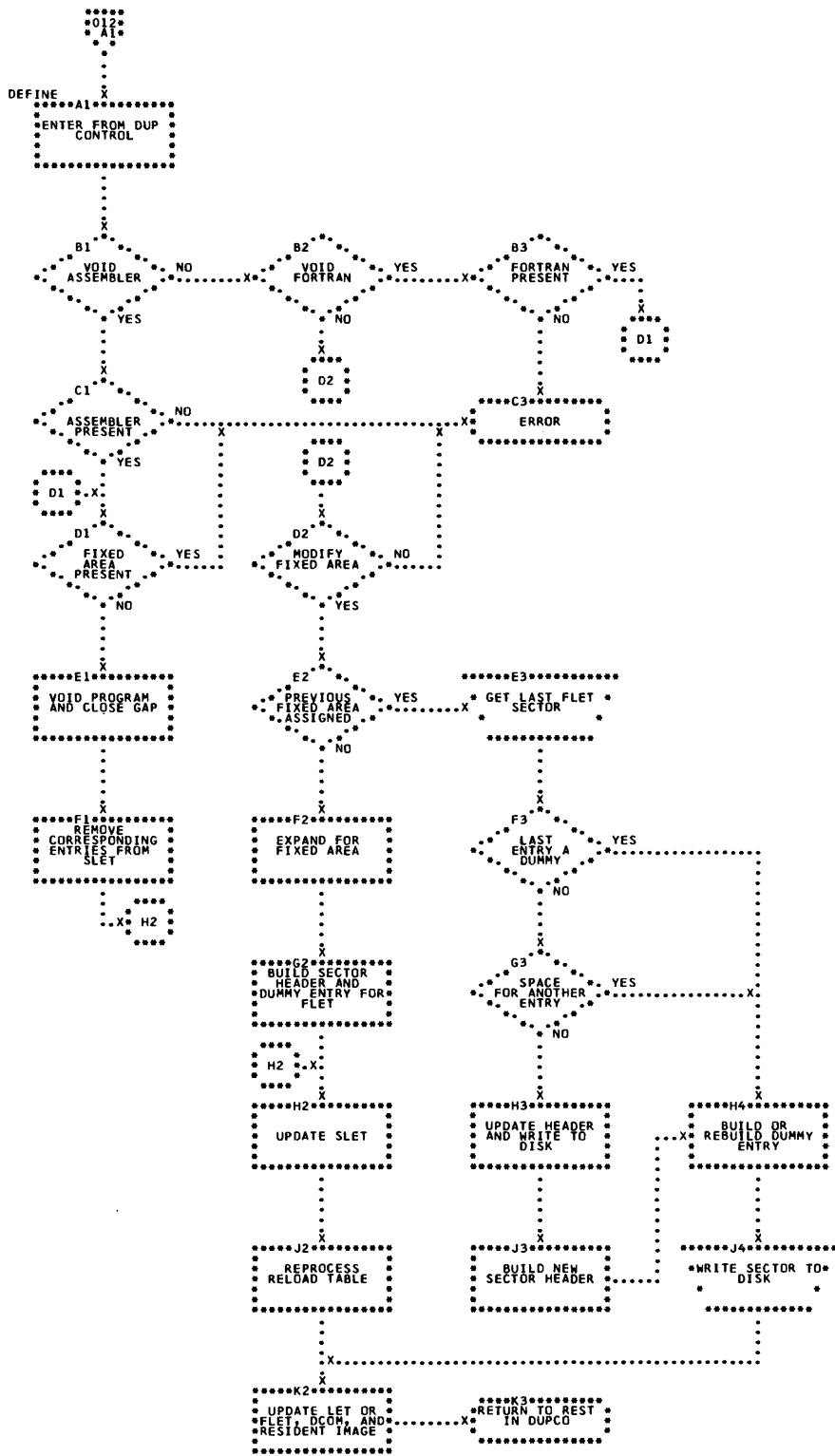
Flowchart DUP07. Disk Utility Program, DDUMP



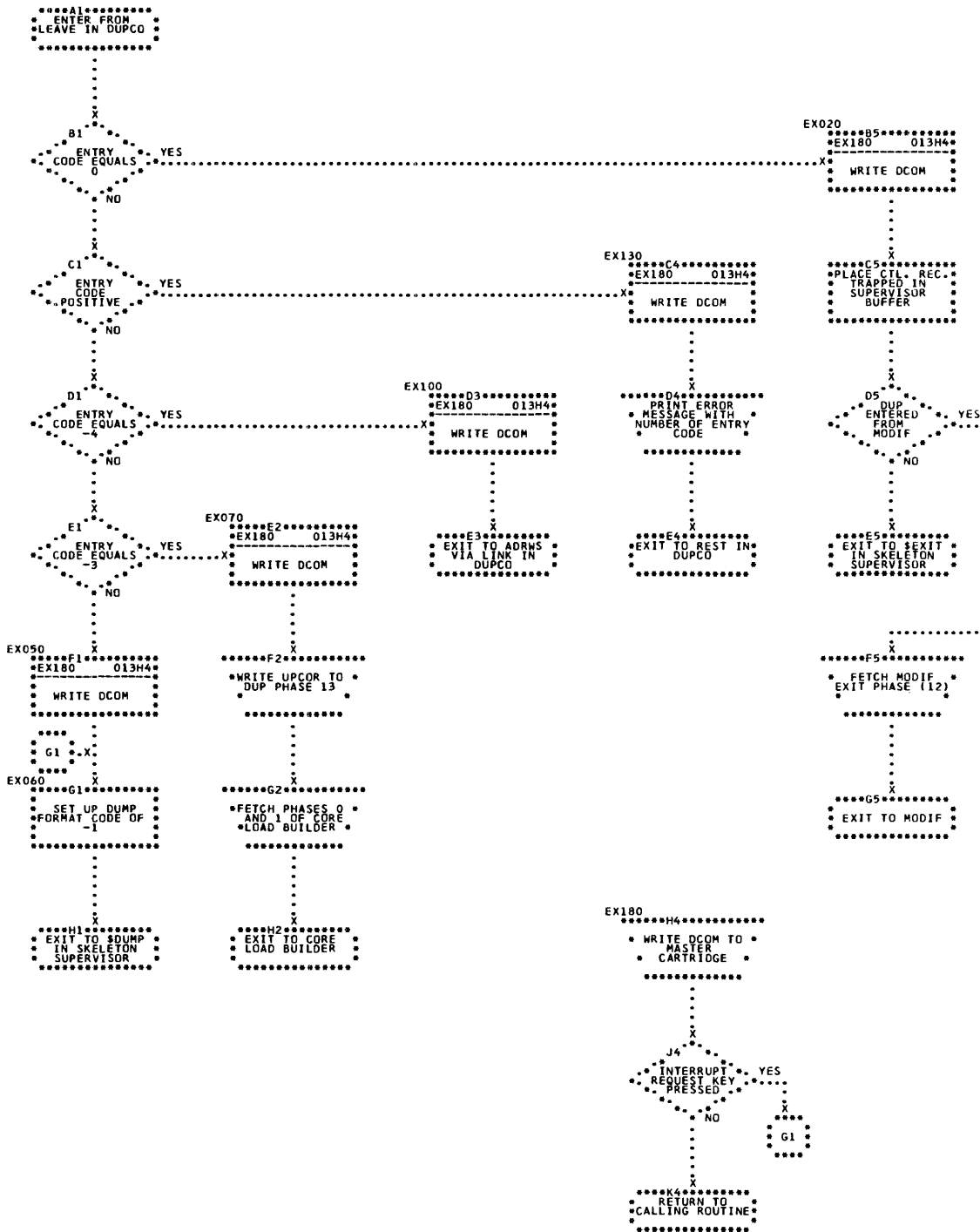
Flowchart DUPO8. Disk Utility Program, DDUMP



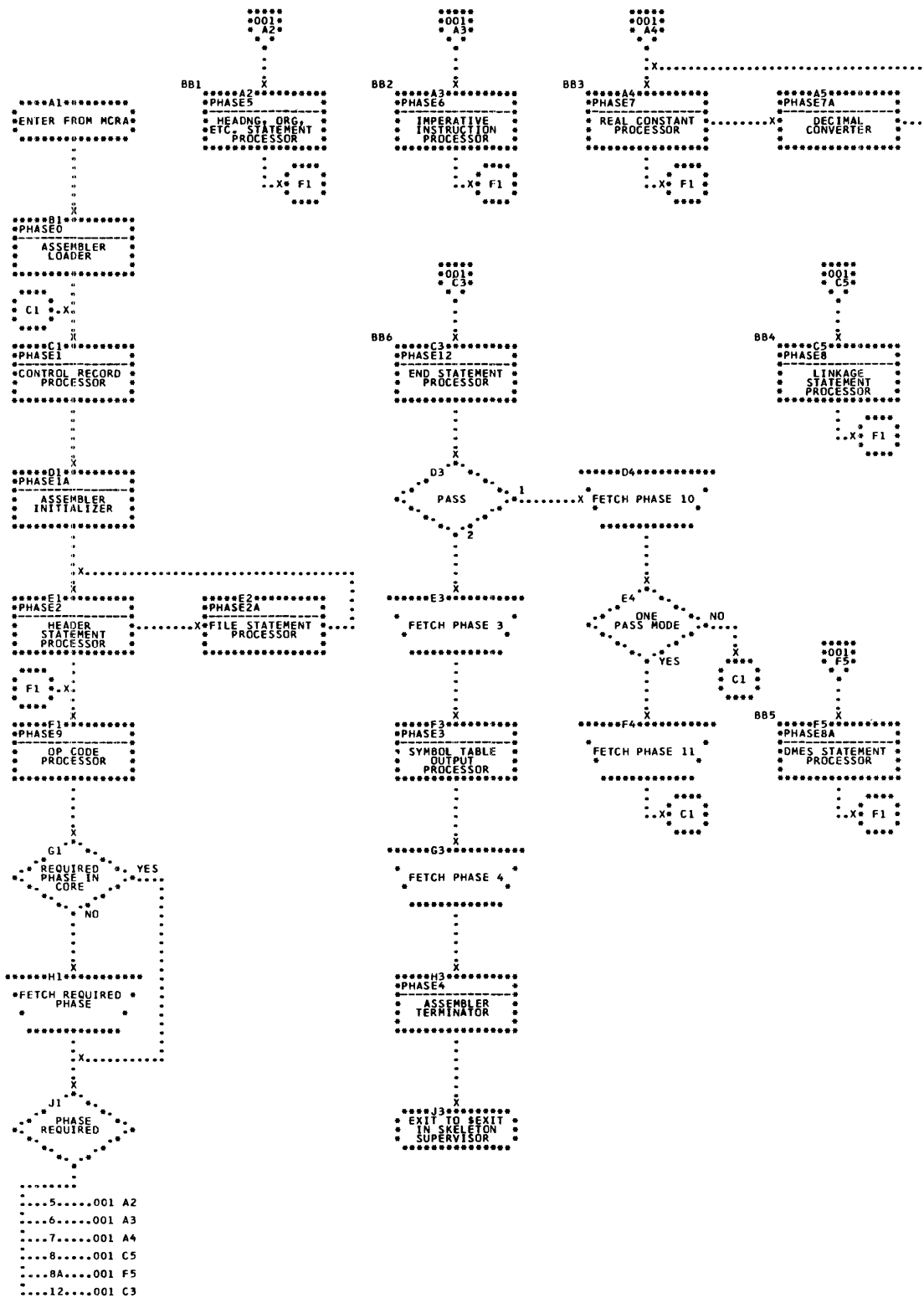
Flowchart DUPO9. Disk Utility Program, DUMPLET/DUMPFLET



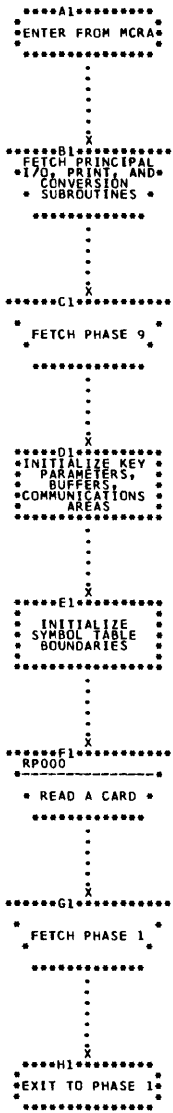
Flowchart DUP11. Disk Utility Program, DEFINE



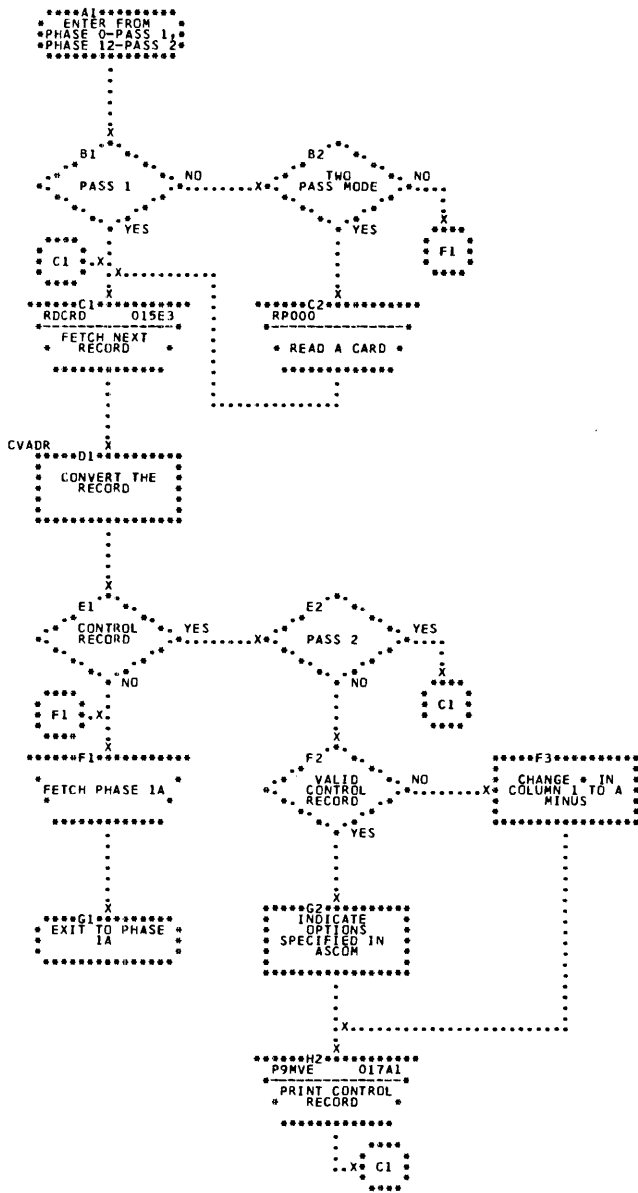
Flowchart DUP12. Disk Utility Program, DEXIT



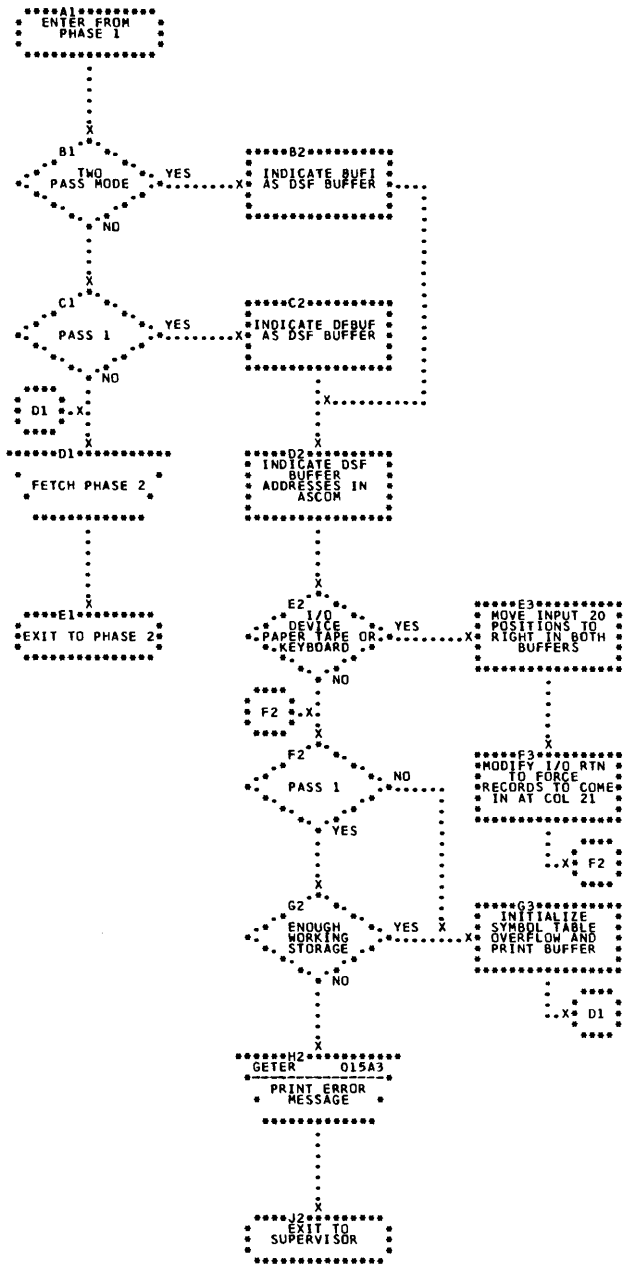
Flowchart ASM01. Assembler Program, General Flow



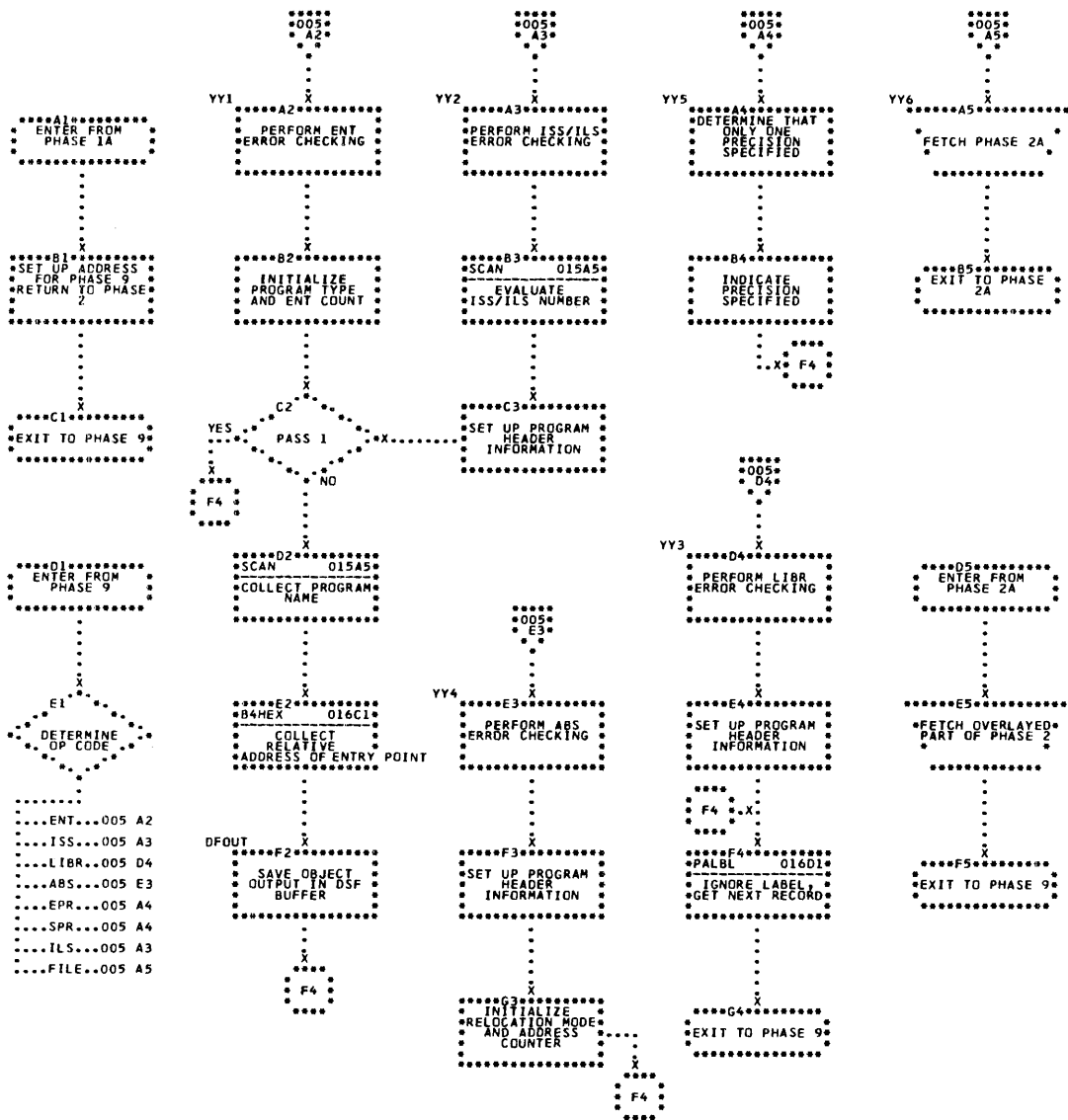
Flowchart ASM02. Assembler Program, Phase 0



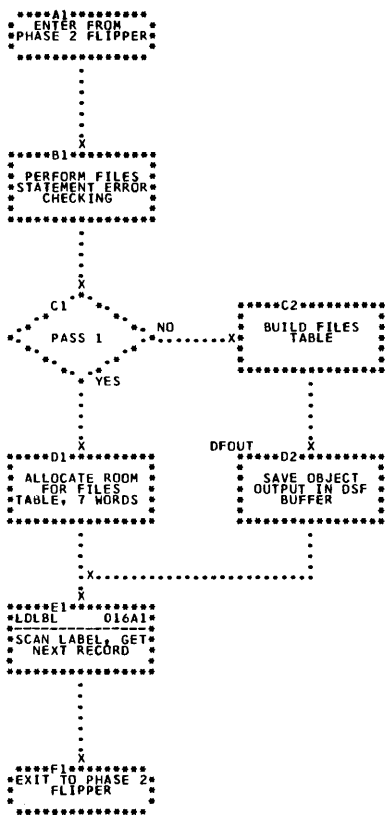
FlowChart ASM03, Assembler Program, Phase 1



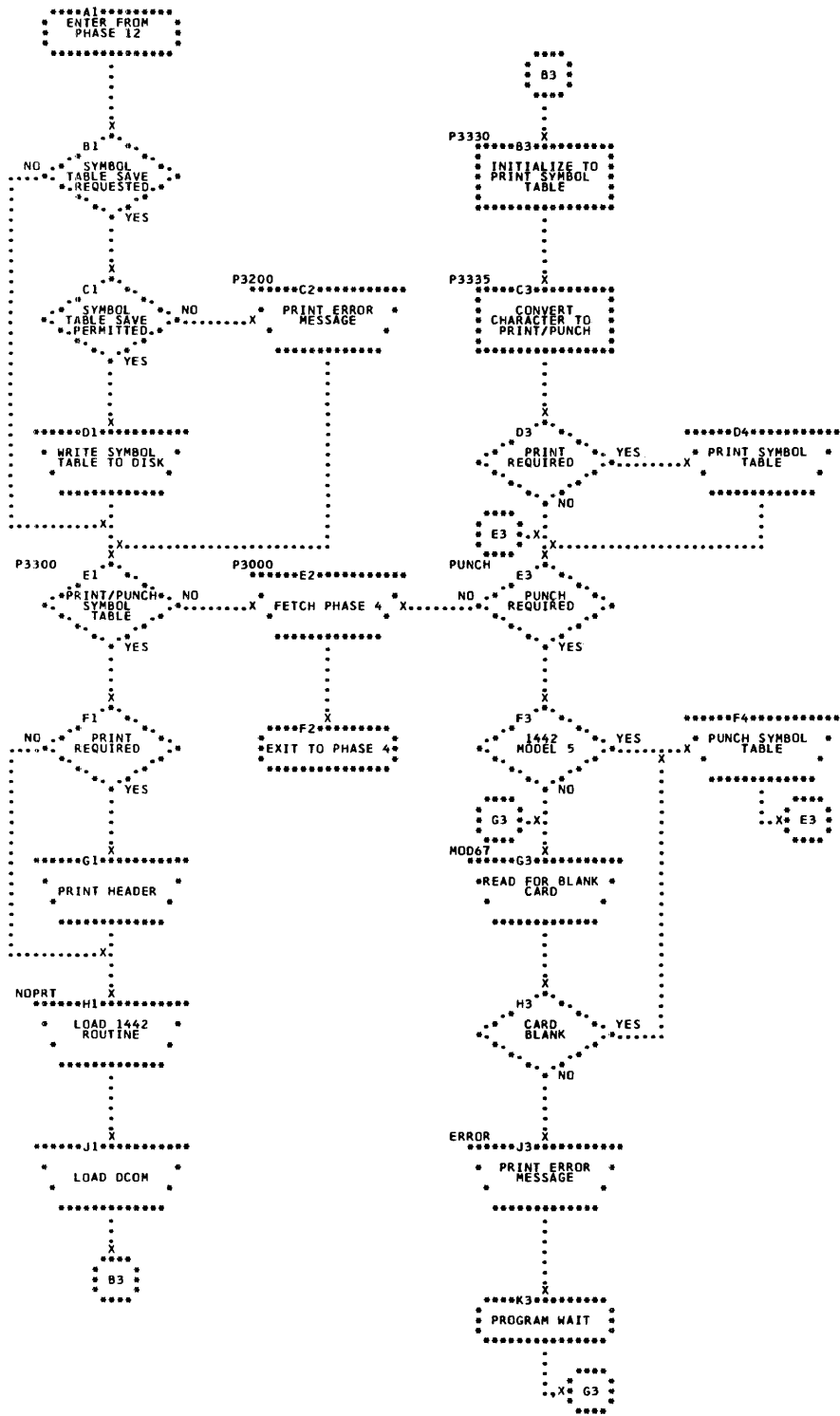
Flowchart ASM04. Assembler Program, Phase 1A



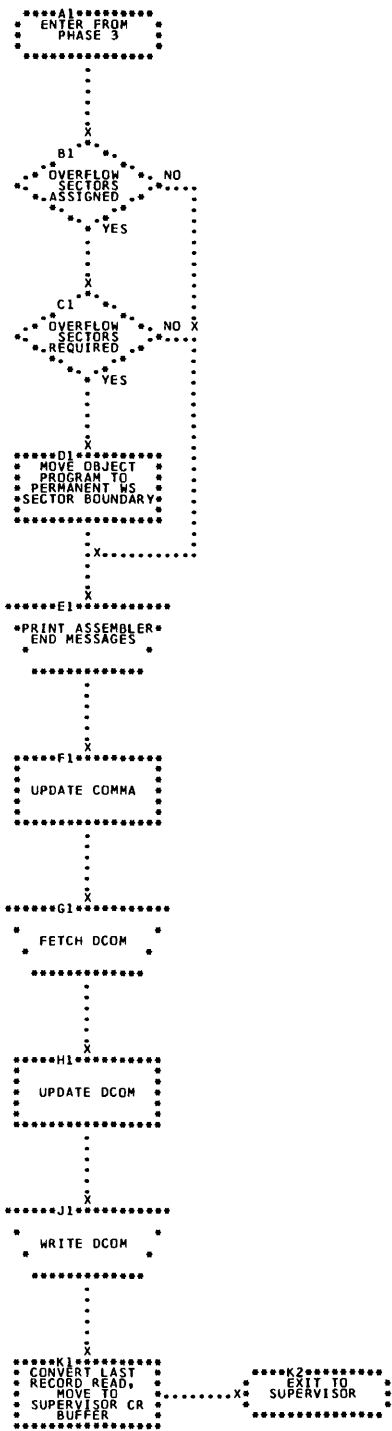
Flowchart ASM05. Assembler Program, Phase 2



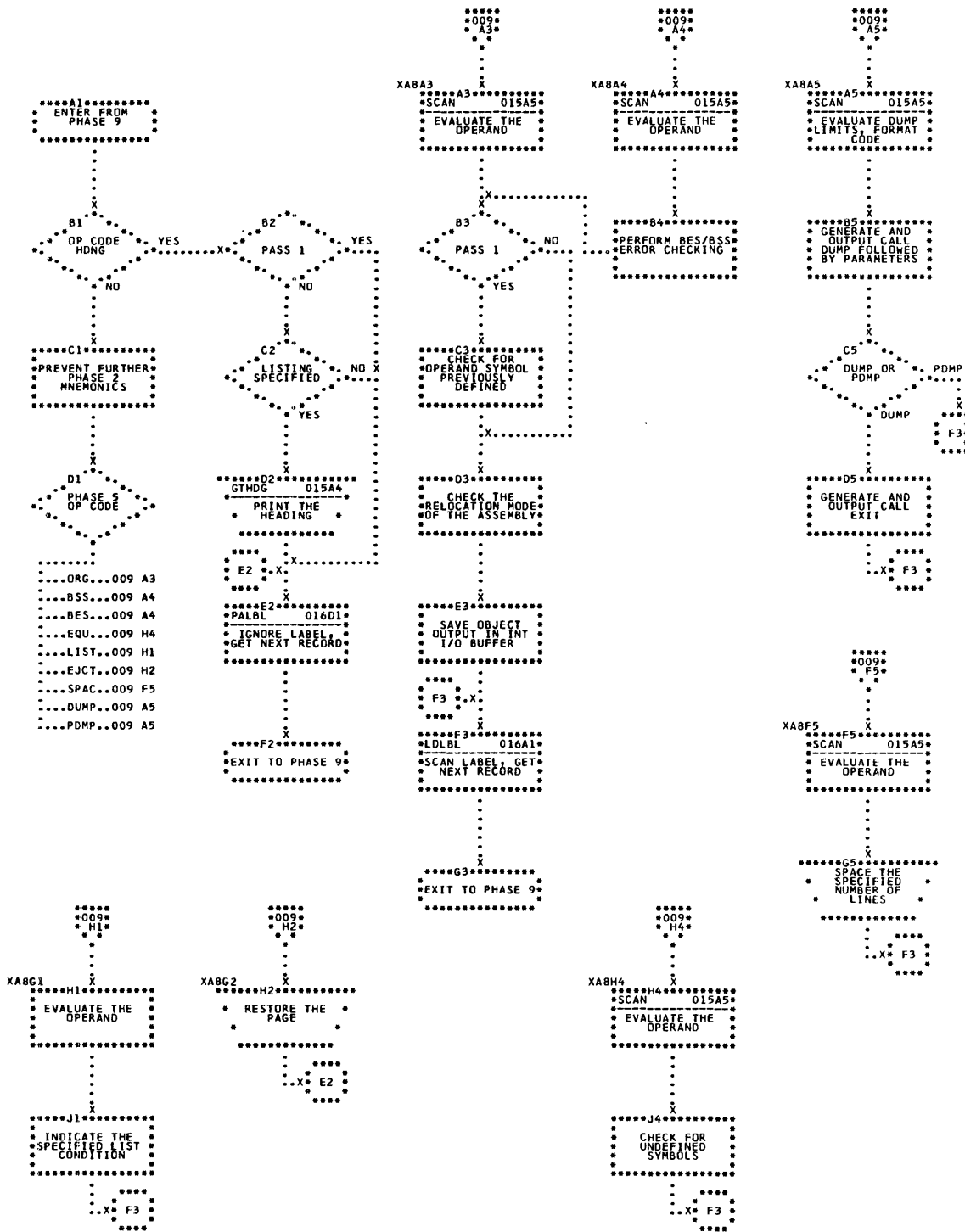
Flowchart ASM06. Assembler Program, Phase 2A



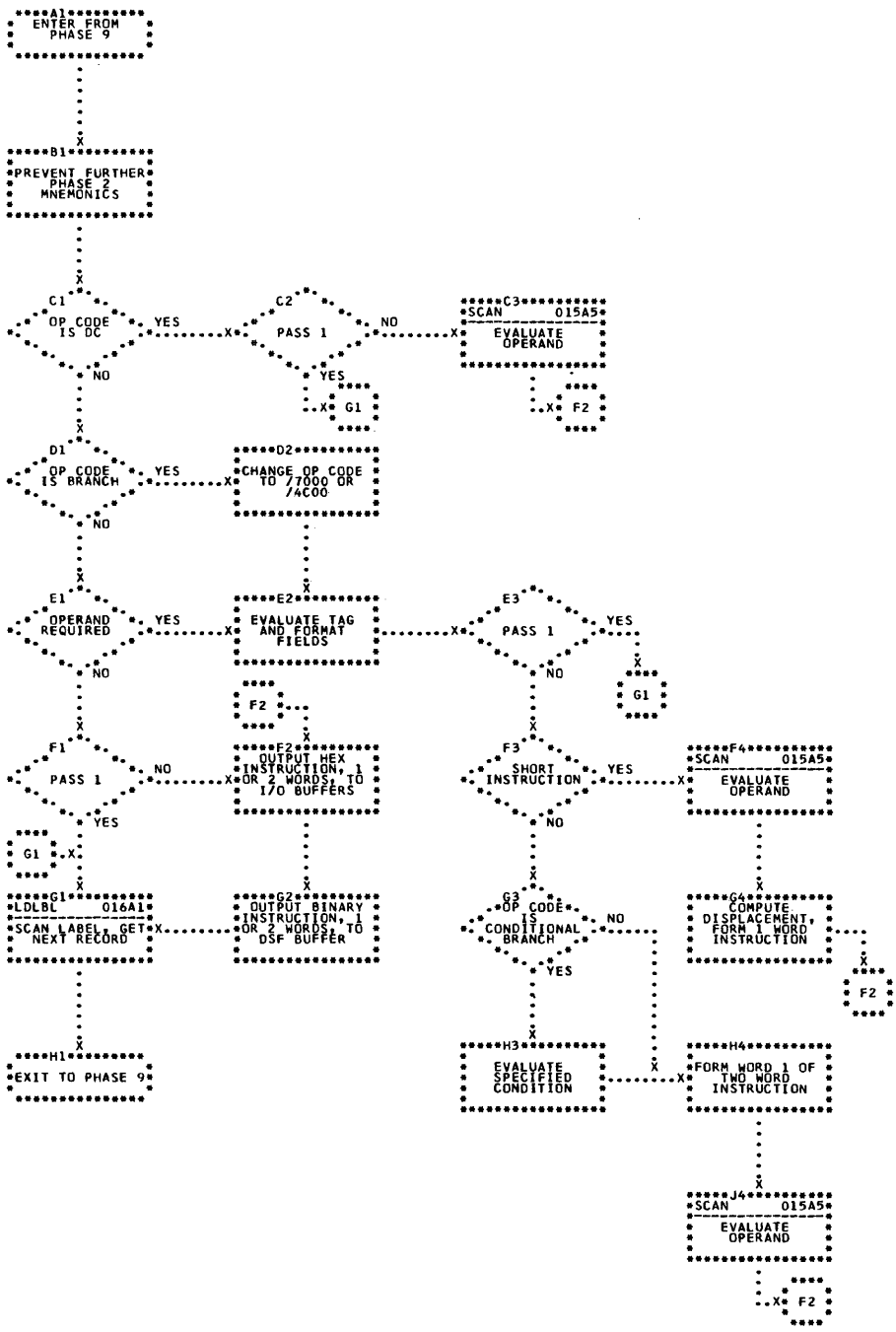
Flowchart ASM07. Assembler Program, Phase 3



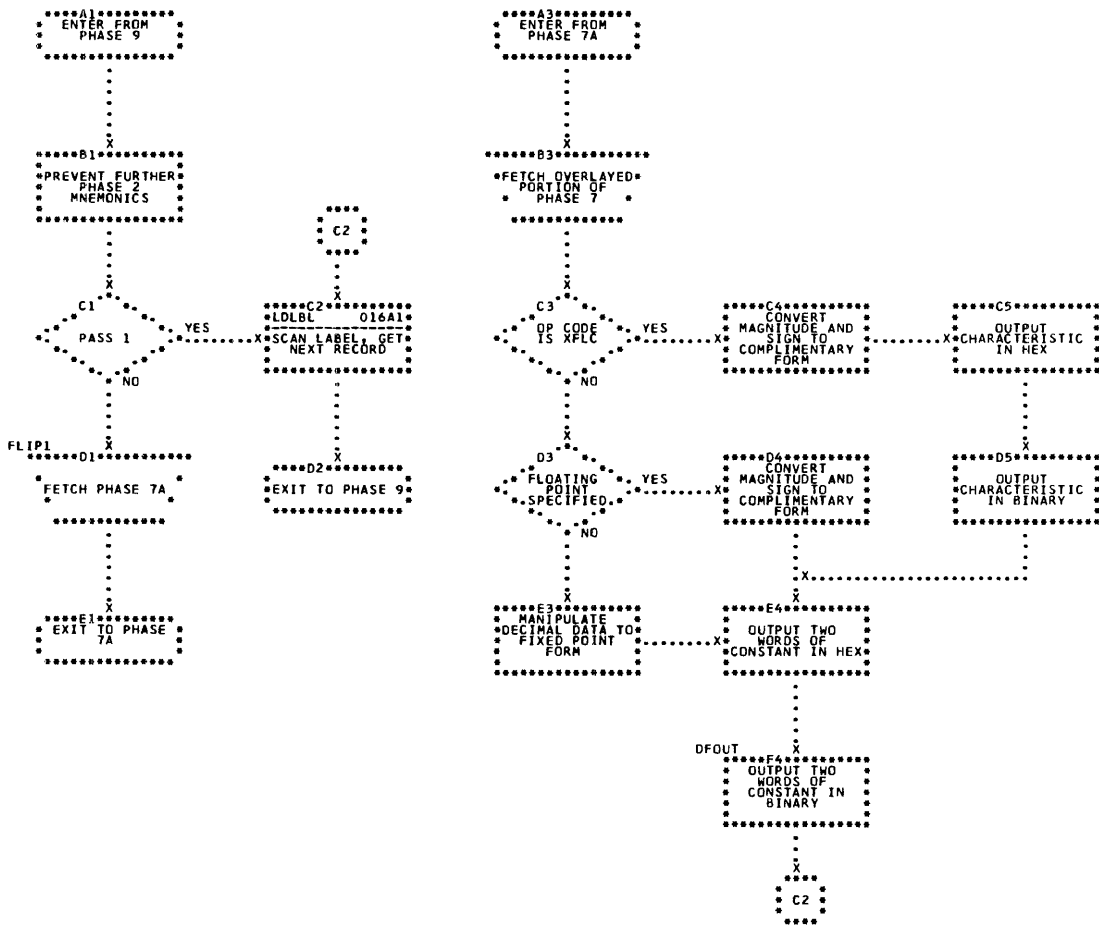
Flowchart ASM08. Assembler Program, Phase 4



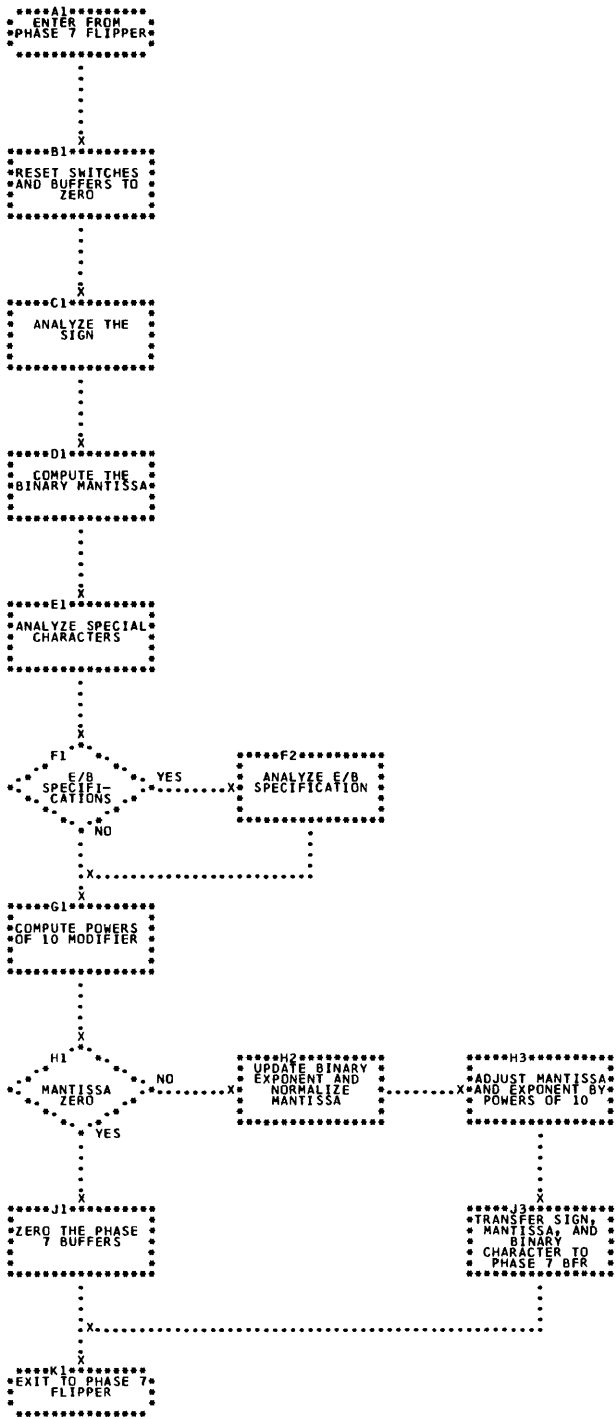
Flowchart ASM09. Assembler Program, Phase 5



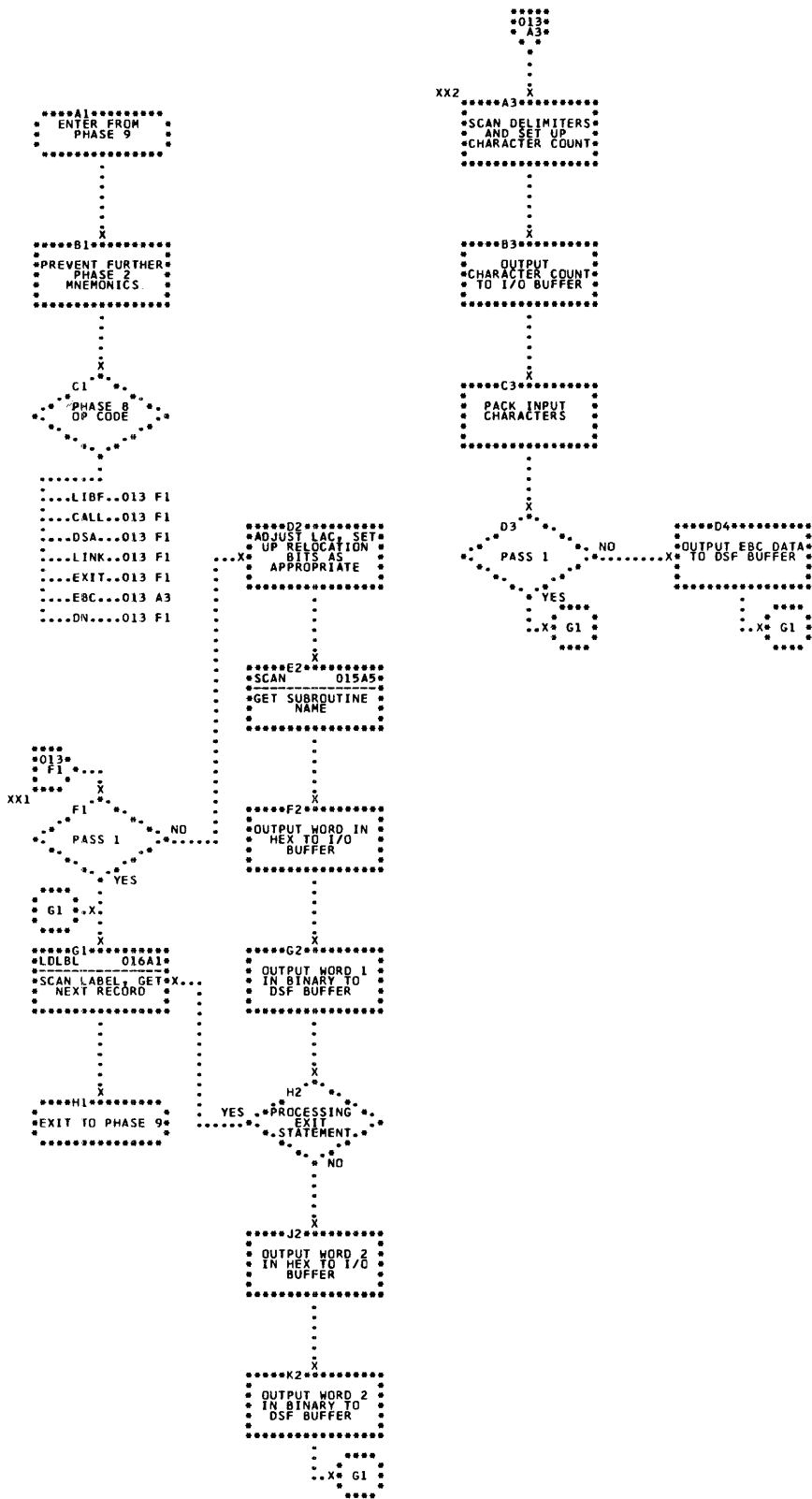
Flowchart ASM10. Assembler Program, Phase 6



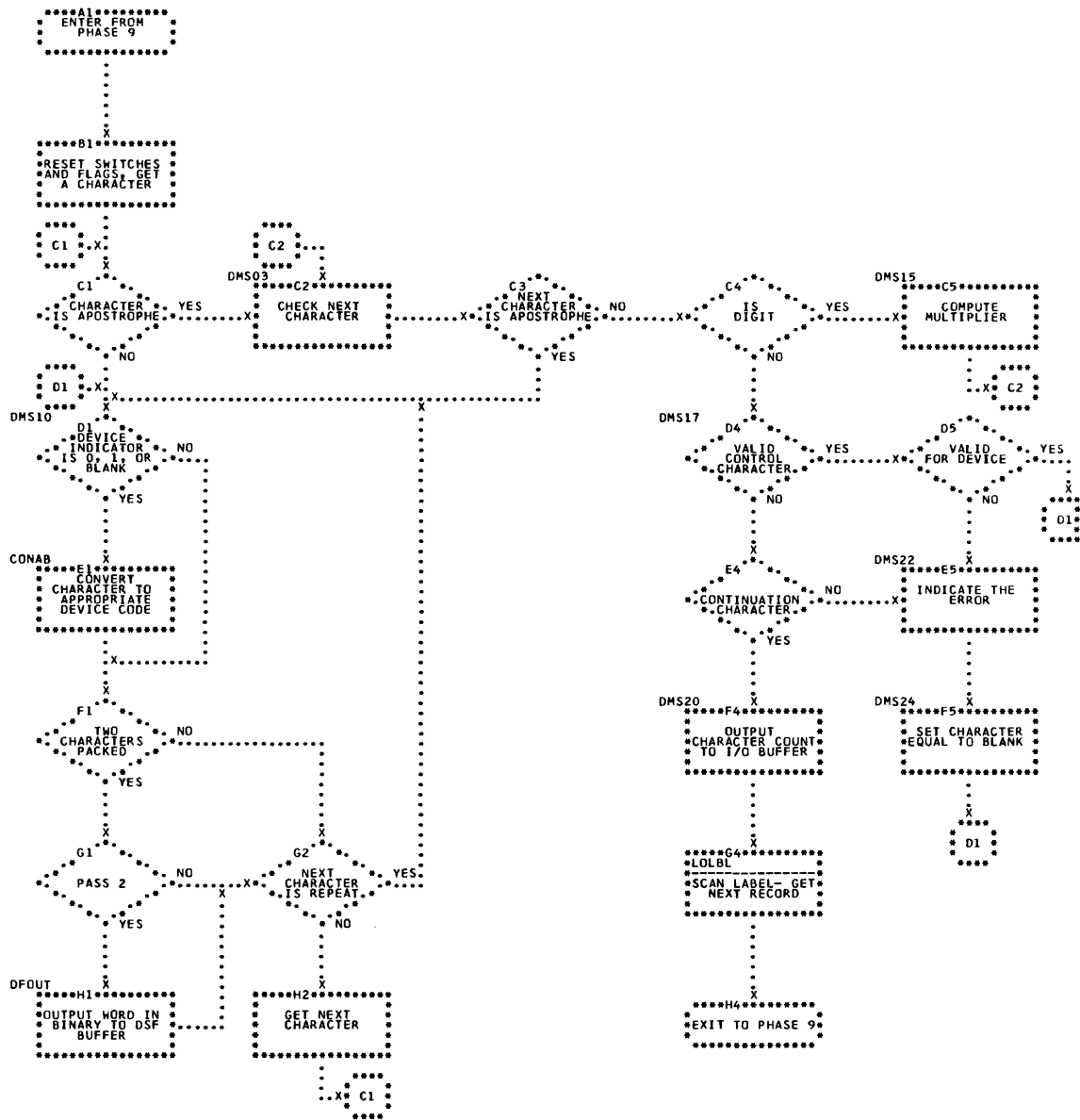
Flowchart ASM11. Assembler Program, Phase 7



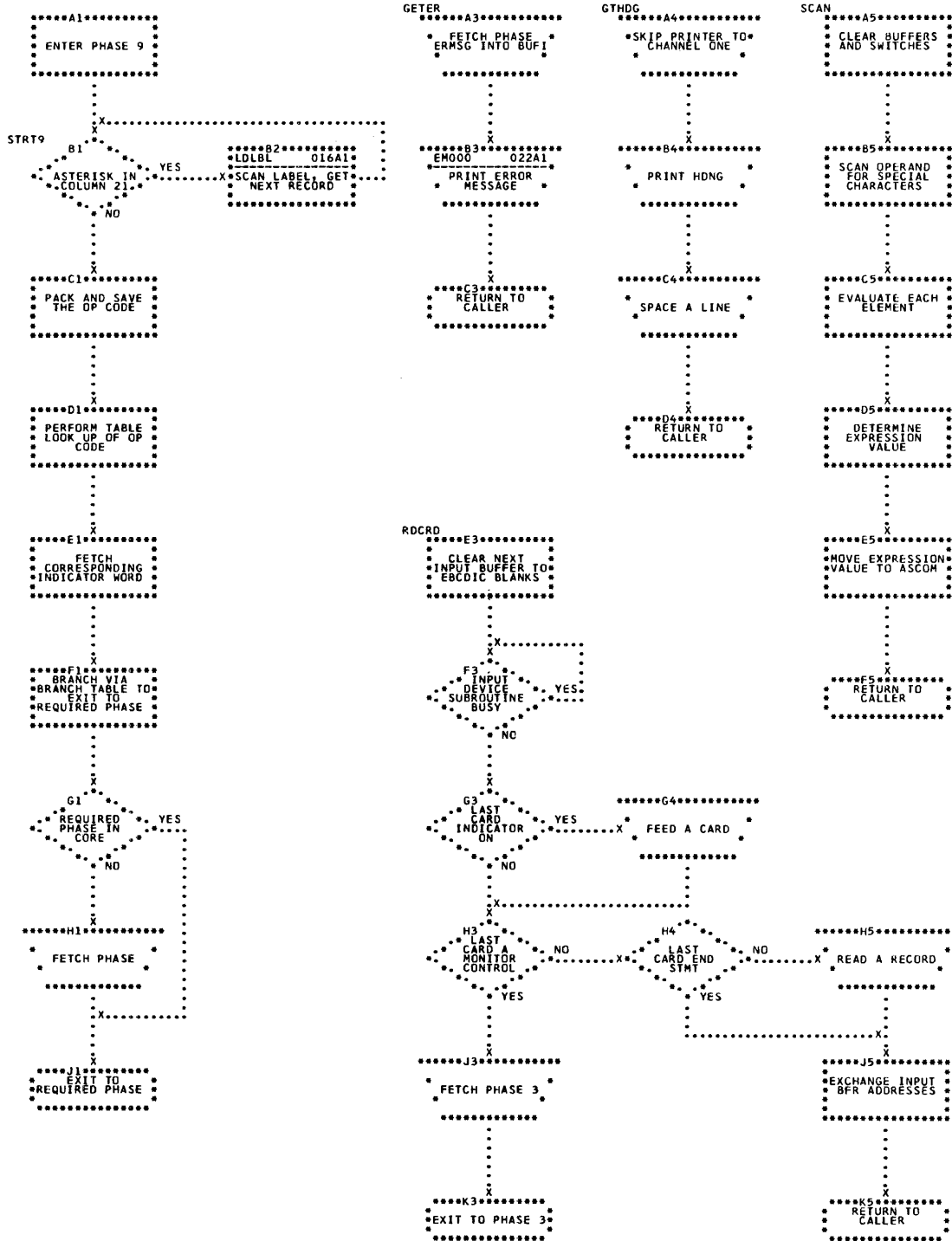
Flowchart ASM12. Assembler Program, Phase 7A



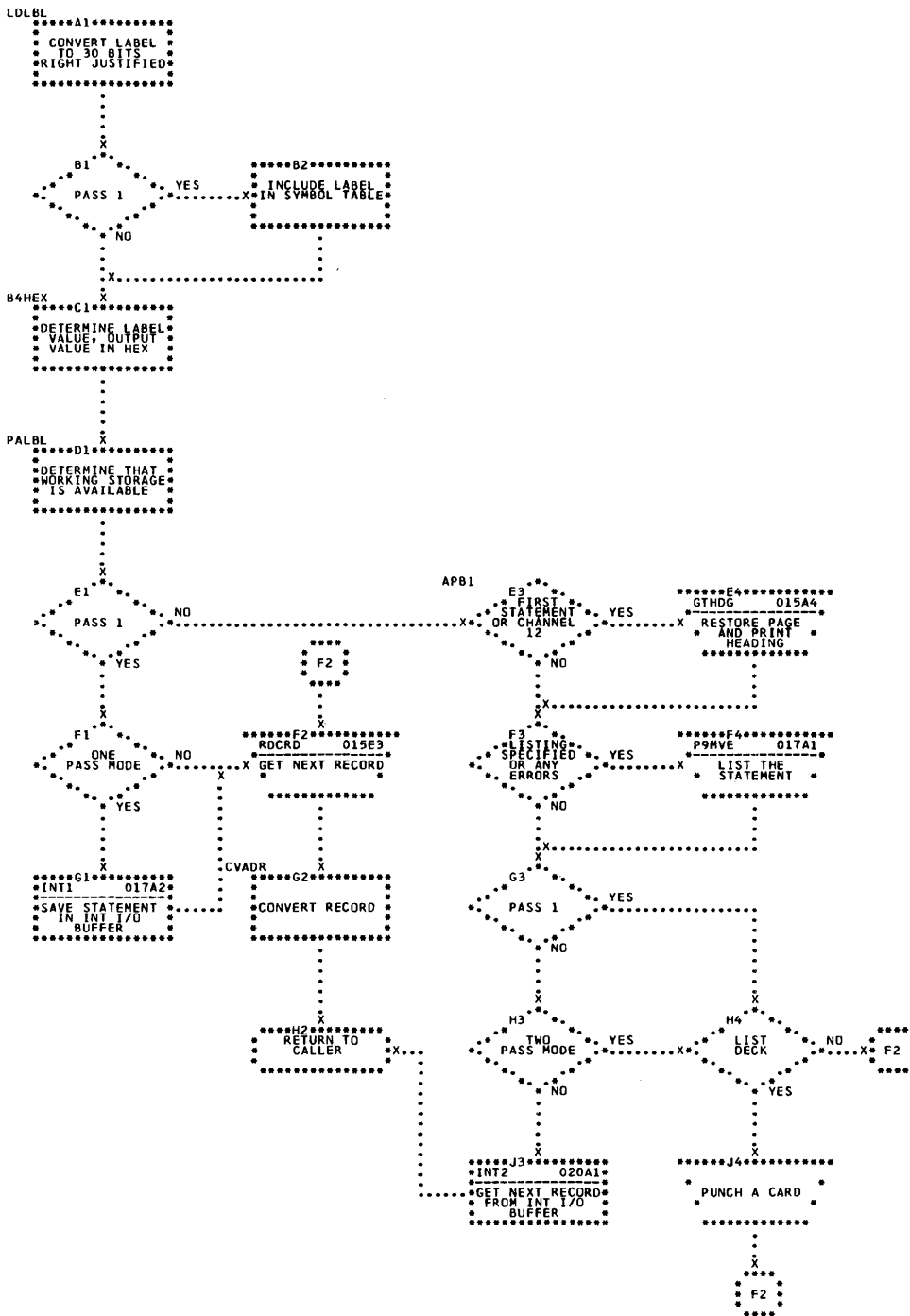
Flowchart ASM13. Assembler Program, Phase 8



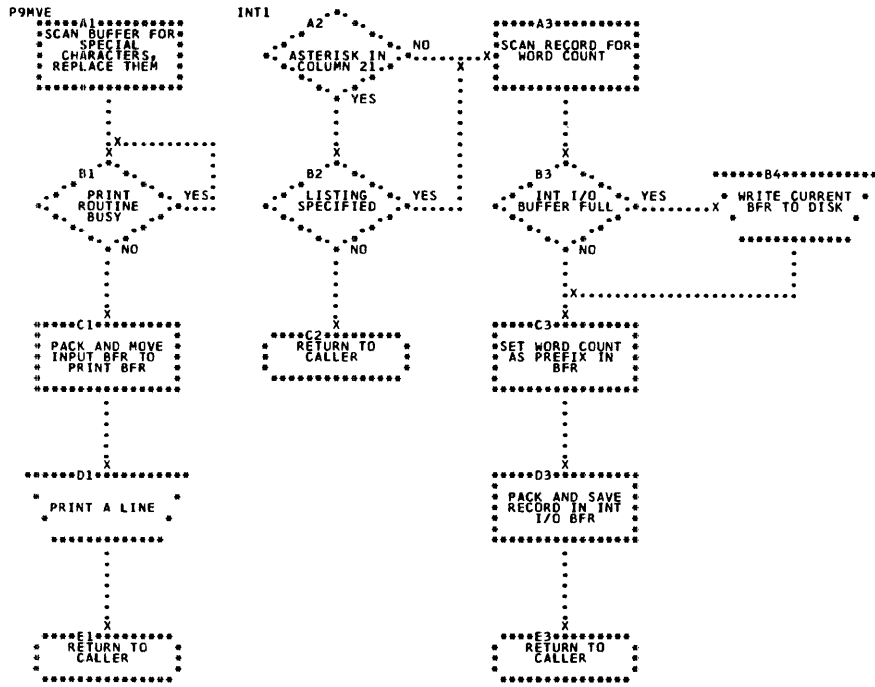
Flowchart ASM14. Assembler Program, Phase 8A



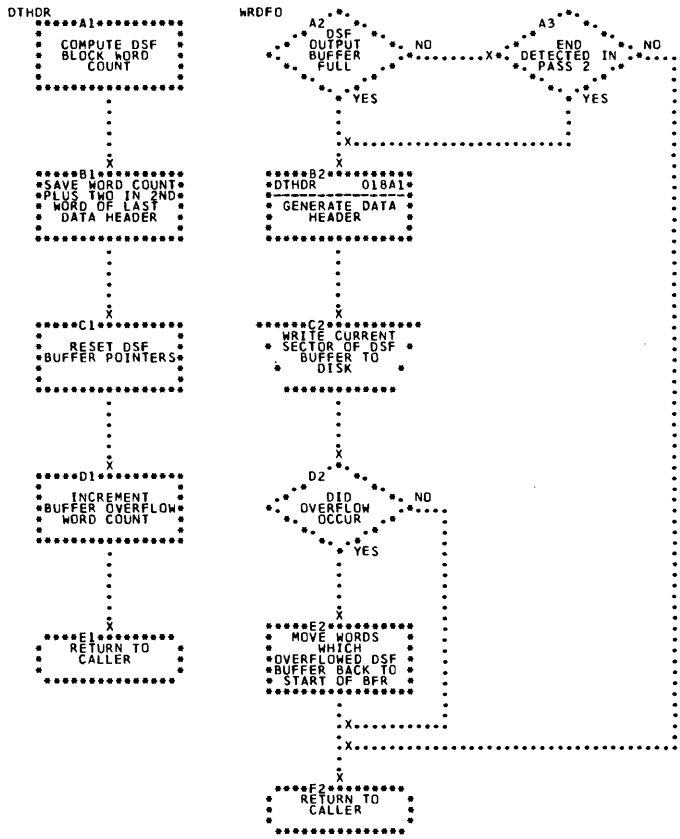
Flowchart ASM15, Assembler Program, Phase 9



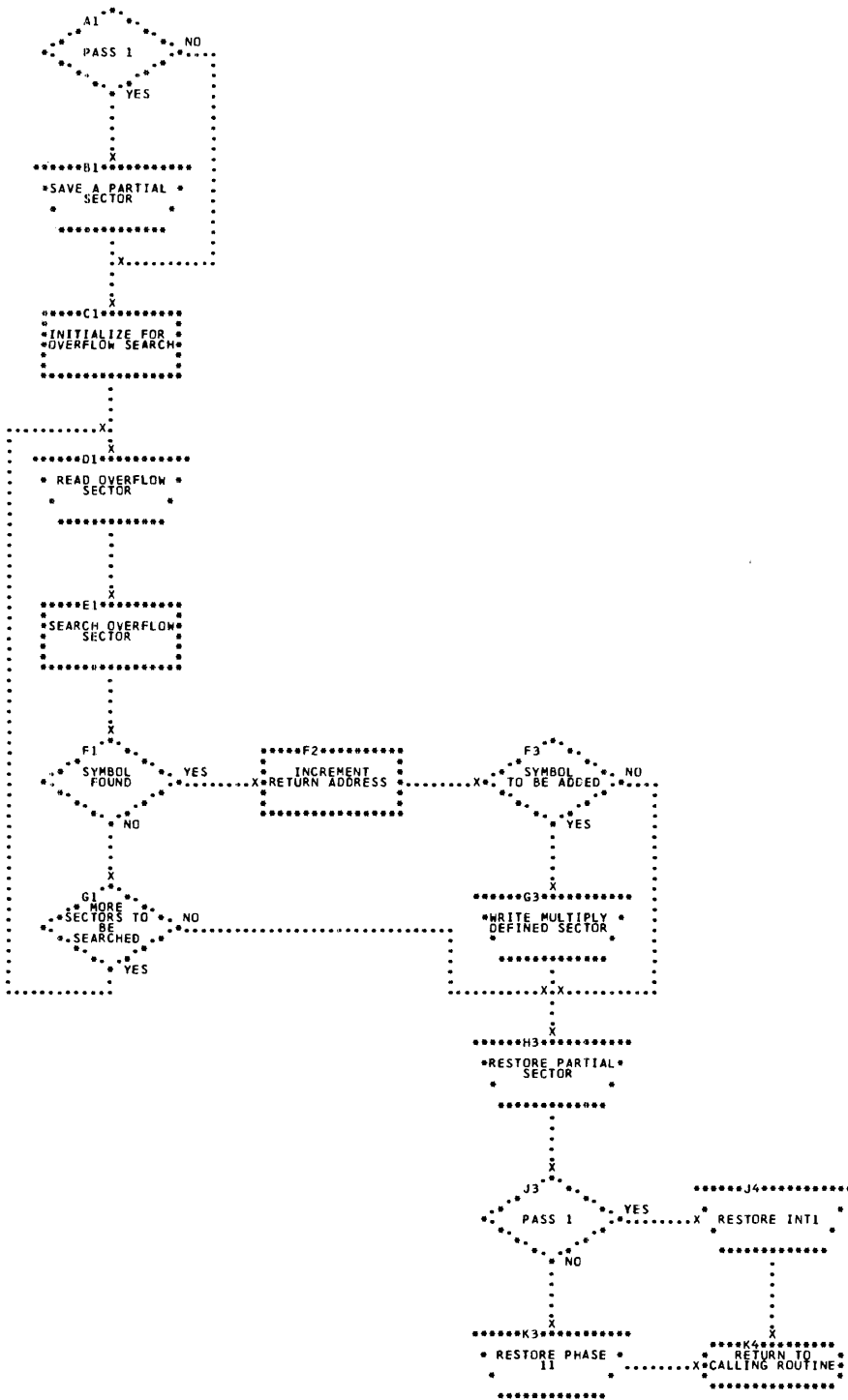
Flowchart ASM16. Assembler Program, Phase 9



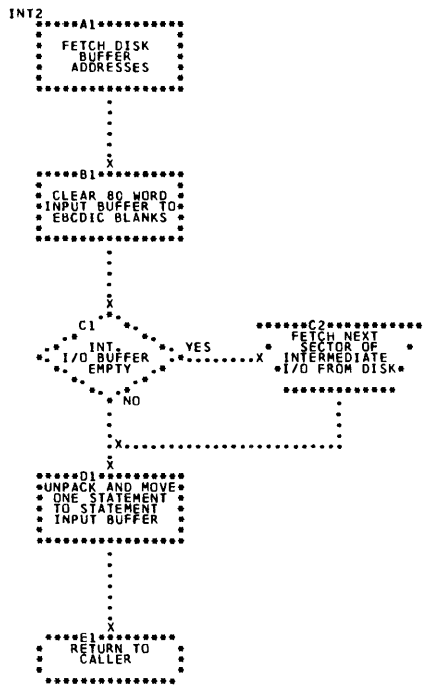
Flowchart ASM17. Assembler Program, Phase 9



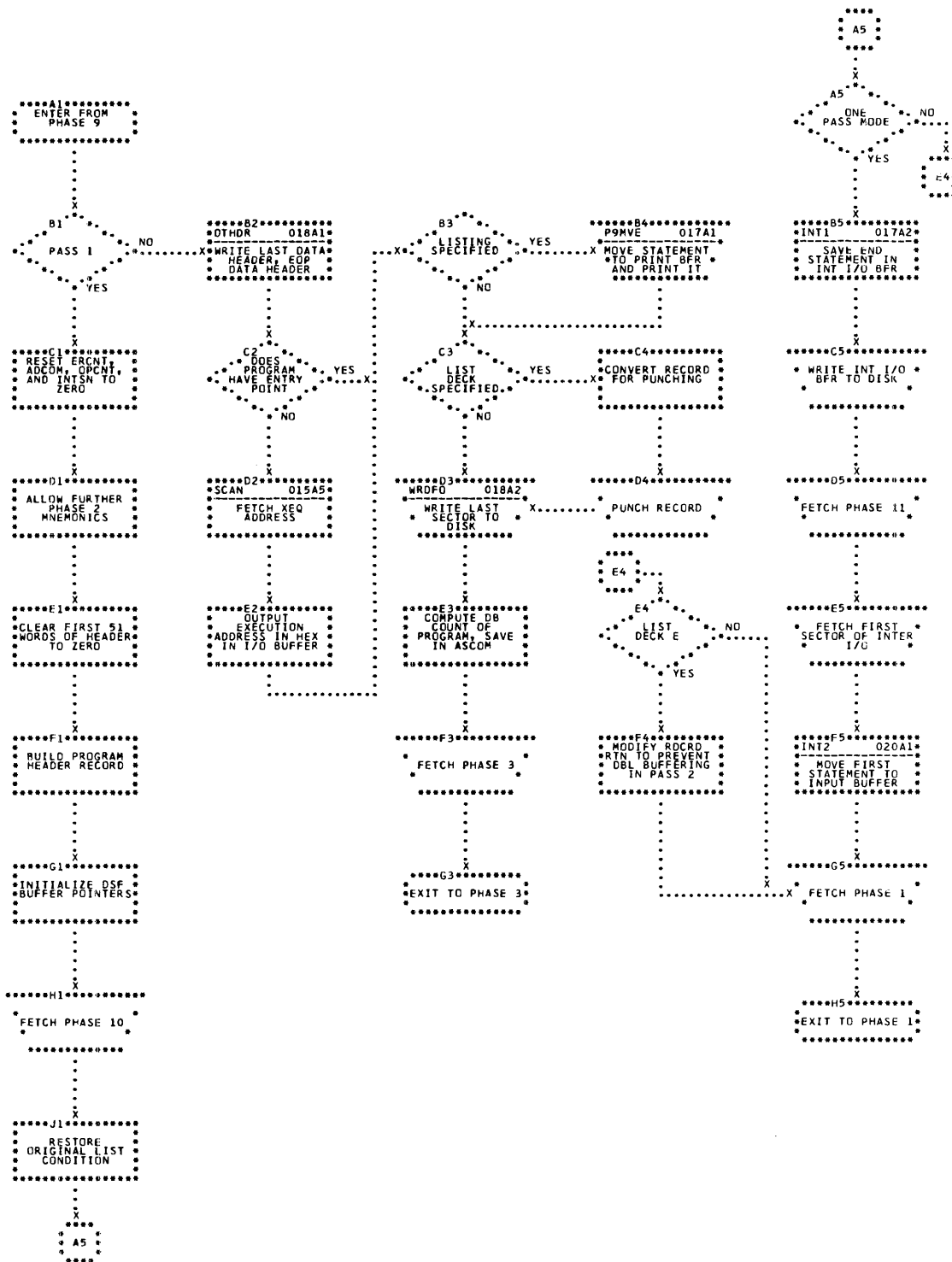
Flowchart ASM18. Assembler Program, Phase 10



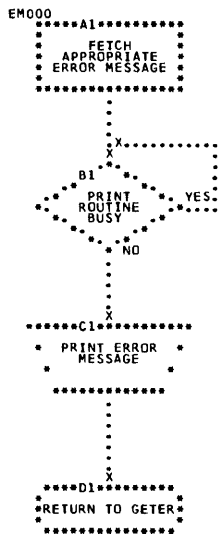
Flowchart ASM19. Assembler Program, Phase 10A



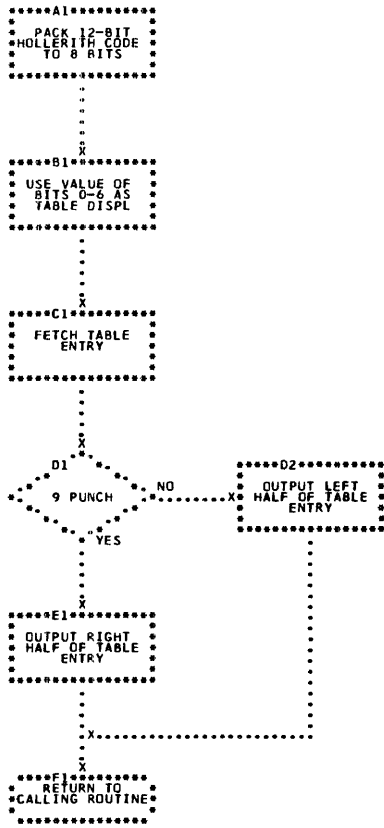
Flowchart ASM20. Assembler Program, Phase 11



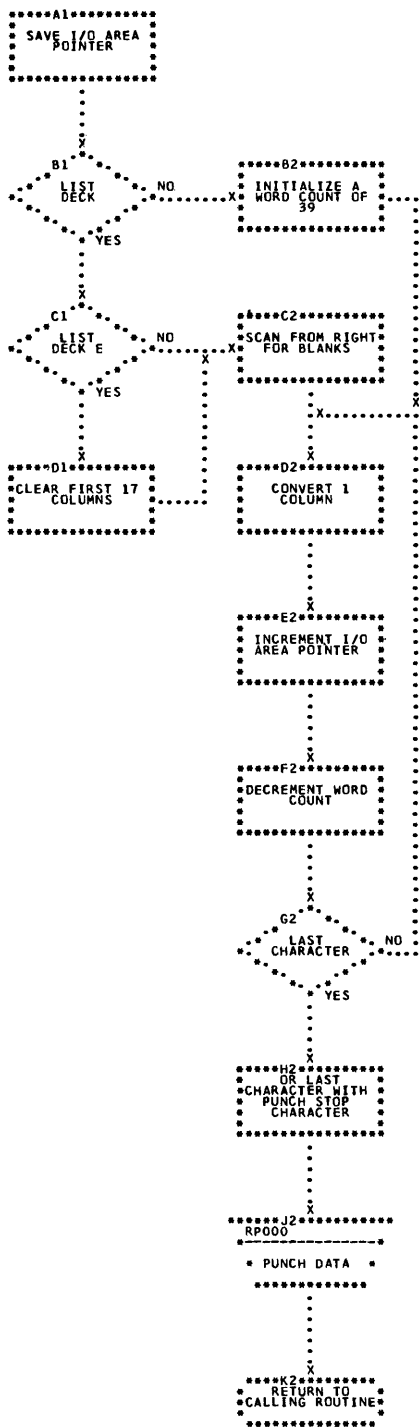
Flowchart ASM21, Assembler Program, Phase 12



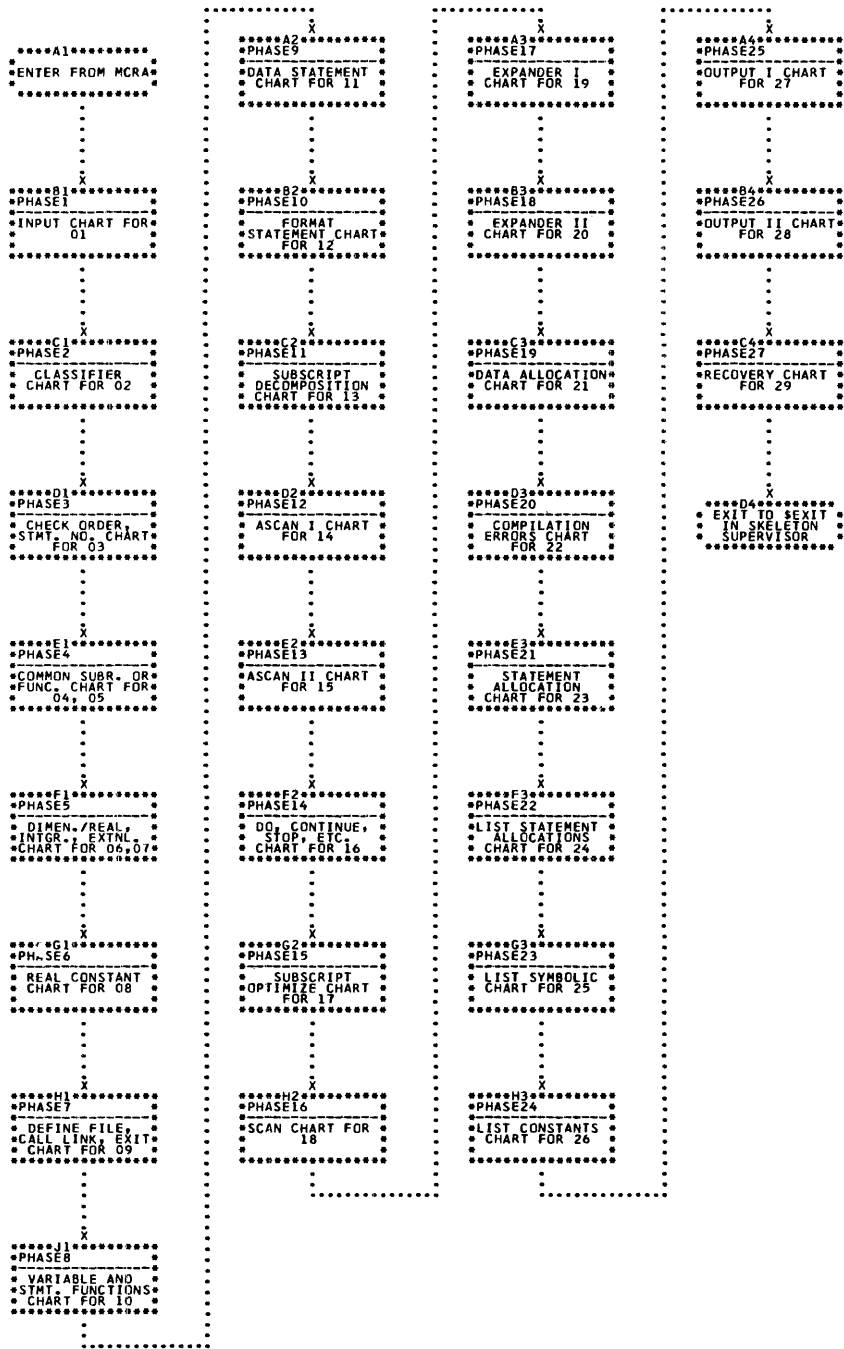
Flowchart ASM22. Assembler Program, Error Message Phase



Flowchart ASM23. Assembler Program, Read Conversion Phase



Flowchart ASM24. Assembler Program, Punch Conversion Phase



Flowchart FOR01. FORTRAN Compiler, General Flow

```

FZ000 .....A1.....
* LOAD SYSTEM
* INPUT PRINT
* SUBRS
* INITIALIZE
* IL504 TABLE
.....

```

```

FZZ50 .....B1.....
* GET MACHINE
* SIZE AND
* INITIALIZE
* PHASE
.....

```

```

F1000 .....C1.....
* F1500 002A2
* READ INPUT
* RECORD
.....

```

```

F1001 .....D1.....
* F1500 002A2
* READ INPUT
* RECORD
.....

```

```

.....E1.....
* CONVERT STMT
* TO EBCDIC CODE
.....

```

```

.....F1.....
* CONTROL
* RECORD
.....

```

```

F1500 .....A2.....
* END
* SWITCH SET
.....

```

```

F1501 .....B2.....
* END
* STMT READ
.....

```

```

F1510 .....C2.....
* READ INPUT
* RECORD
.....

```

```

F1512 .....D2.....
* RETURN
.....

```

```

.....F2.....
* ANALYZE AND
* STORE CONTROL
* RECORD DATA
.....

```

```

F1004 .....G1.....
* PRINT SOURCE
* STATEMENT IF
* REQUIRED
.....

```

```

.....H1.....
* CONTROL
* RECORD
.....

```

```

.....A4.....

```

```

F1005 .....A4.....
* COMMENT
* STATEMENT
.....

```

```

F1006 .....B4.....
* CONTINUATION
* STMT
.....

```

```

F1009 .....C4.....
* COLLECT
* STATEMENT
* NUMBER
.....

```

```

F1017 .....D4.....
* PACK STATEMENT
* AND PLACE ON
* STRING
.....

```

```

.....F4.....

```

```

ROL .....F4.....
* SET UP TO LOAD
* NEXT PHASE
.....

```

```

.....G4.....
* GET PHASE ID
* FROM BIT
* SWITCHES
.....

```

```

.....H4.....
* EQUAL
* TO ID THIS
* PHASE
.....

```

```

RLQ10 .....J4.....
* LOAD NEXT PHASE
.....

```

```

.....K4.....
* EXIT TO NEXT
* PHASE
.....

```

```

.....C1.....

```

```

F1016 .....B5.....
* TOO
* MANY
* CONTINUATION
* STMTS
.....

```

```

.....C4.....

```

```

F1054 .....D5.....
* REPLACE WITH
* STATEMENT
* ERROR
.....

```

```

.....D1.....

```

```

.....F4.....

```

```

.....F4.....

```

```

.....G4.....

```

```

.....H5.....
* DUMP STRING
* AREA, SYMBOL
* TABLE AREA, AND
* FCOR
.....

```

```

.....J4.....

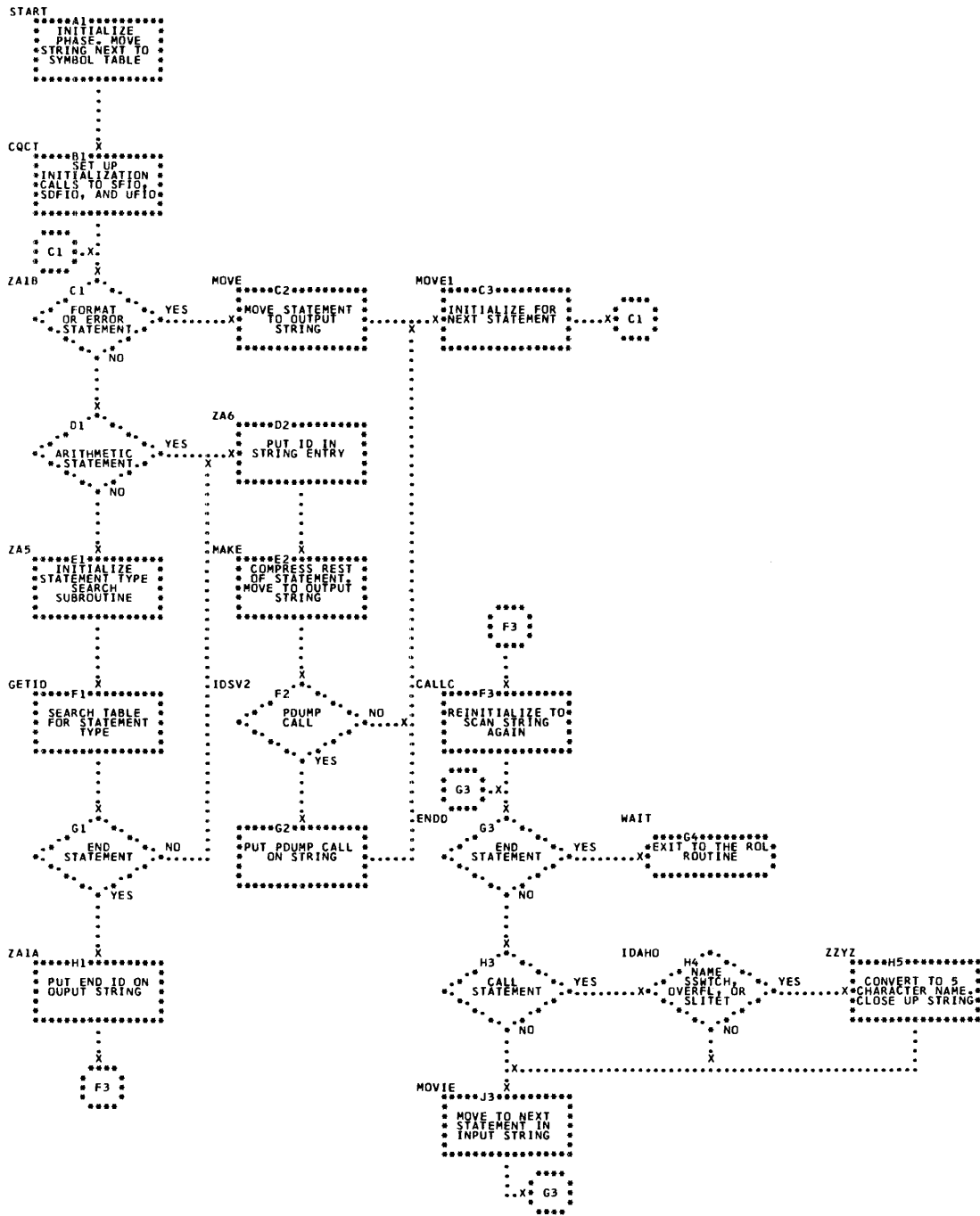
```

```

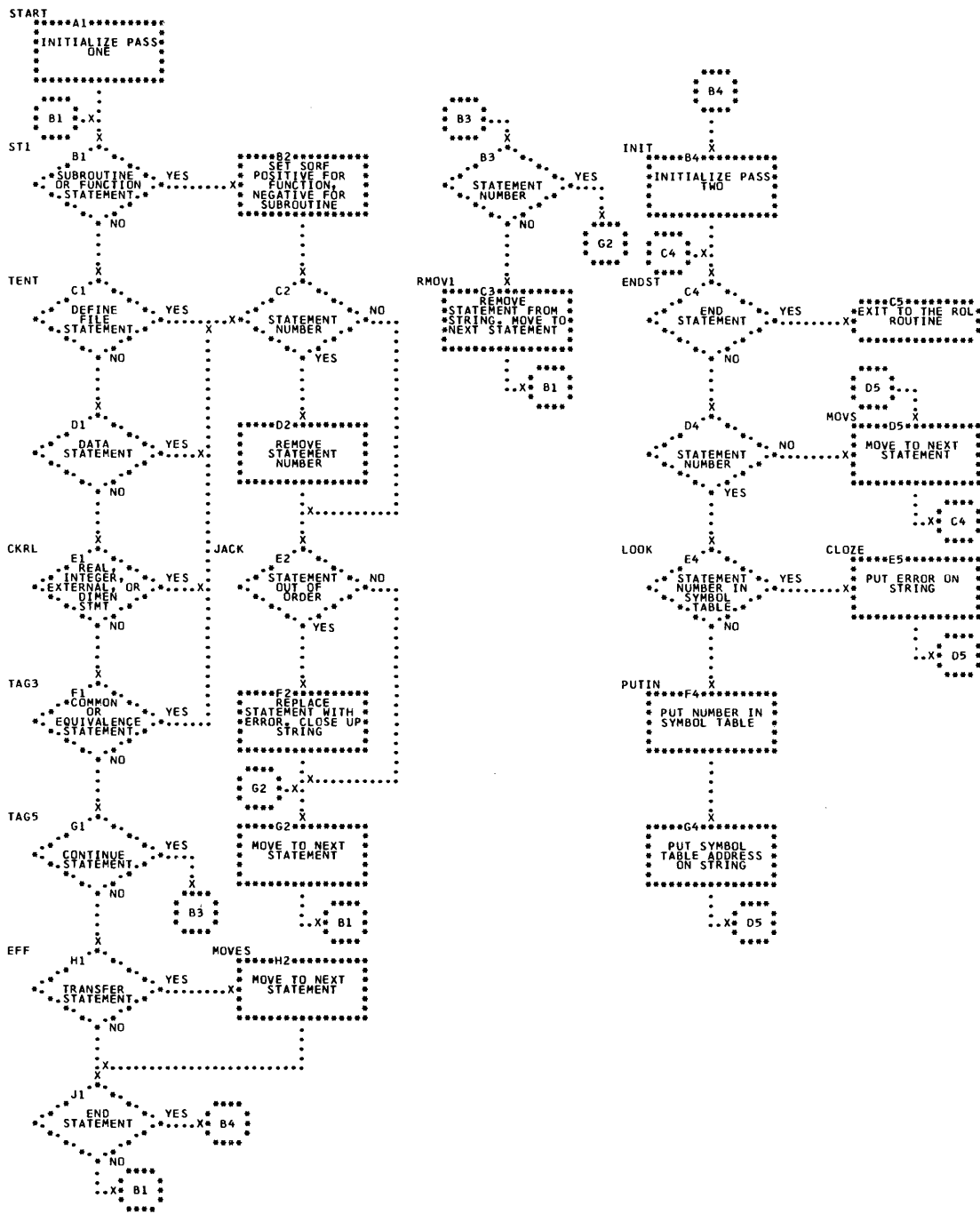
.....K4.....

```

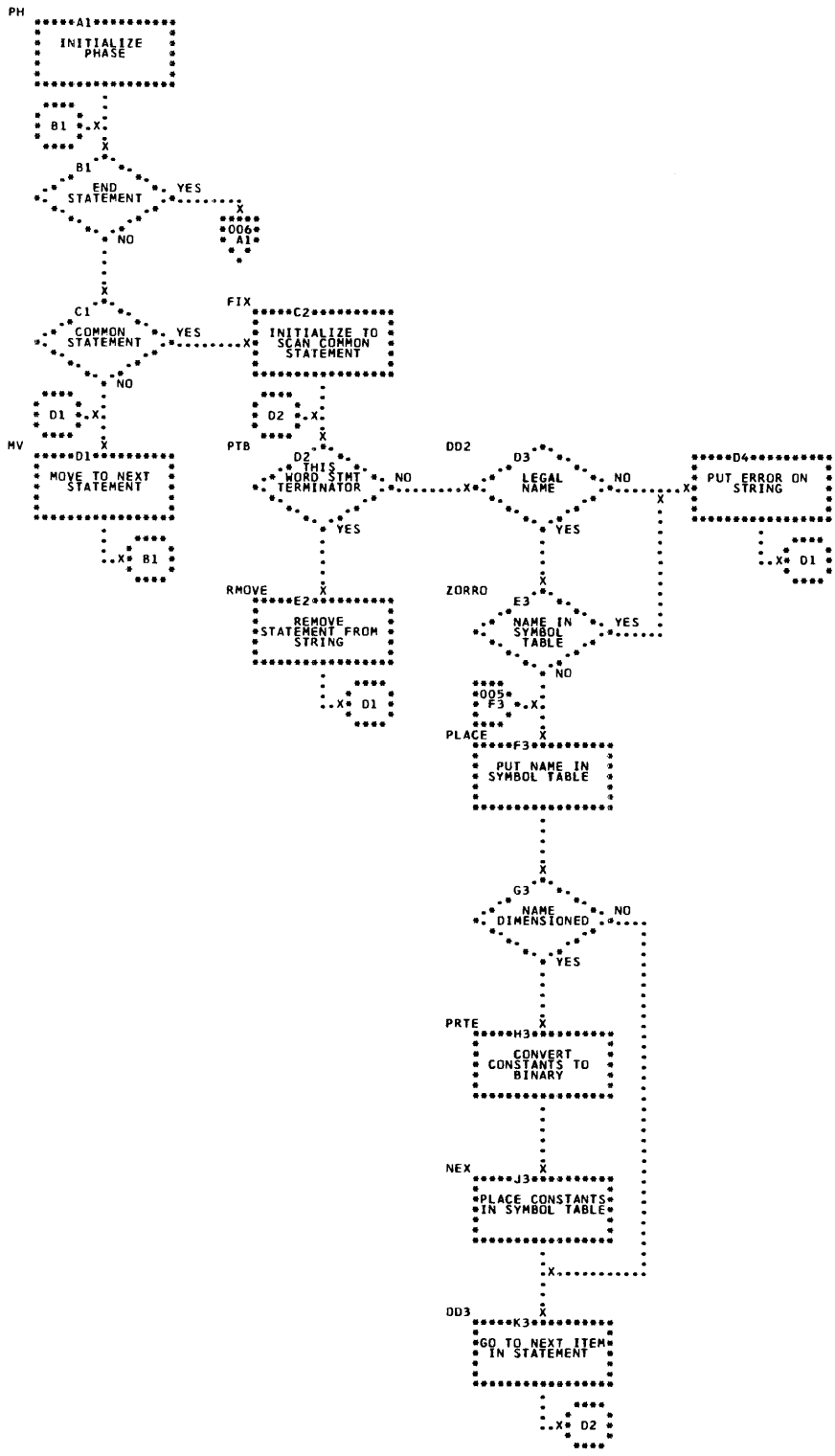
Flowchart FOR02. FORTRAN Compiler, Phase 1



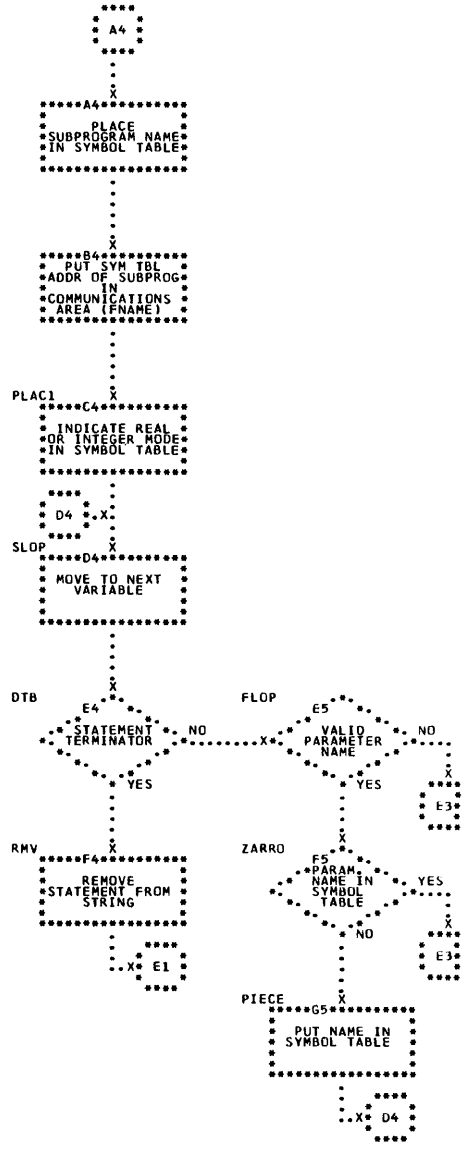
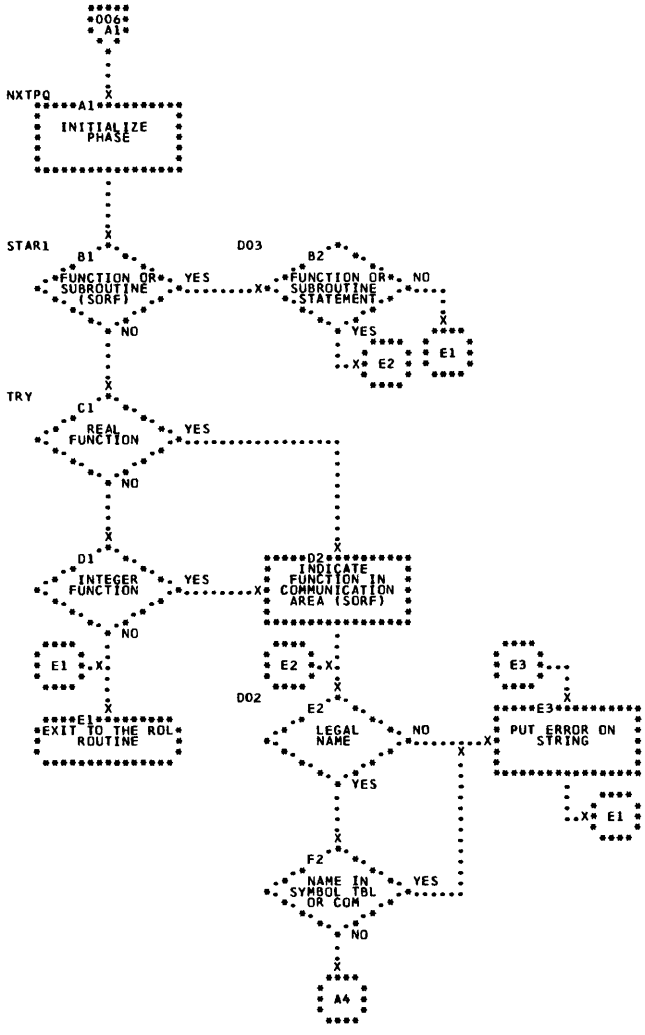
Flowchart FOR03. FORTRAN Compiler, Phase 2



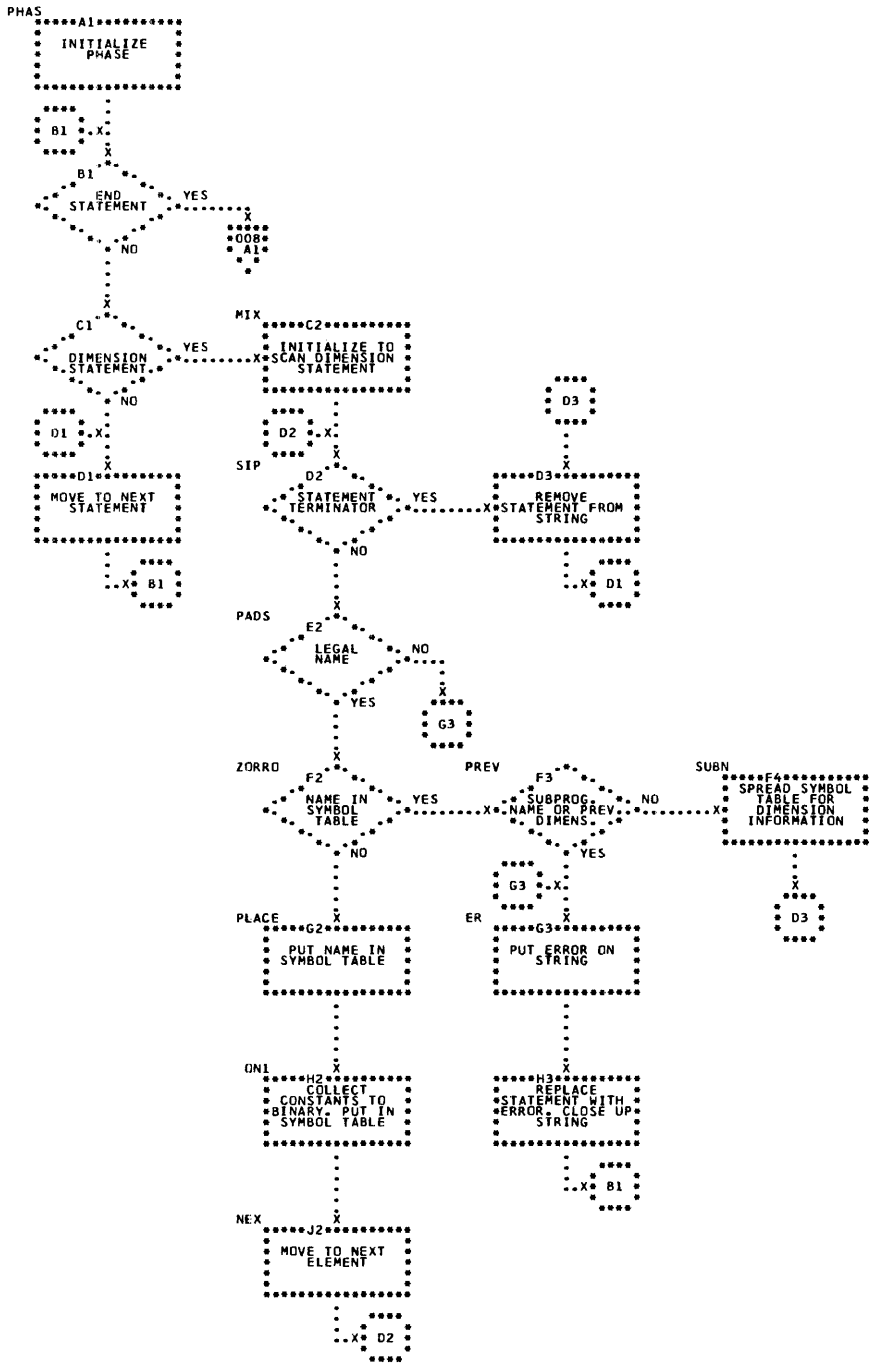
Flowchart FOR04. FORTRAN Compiler, Phase 3



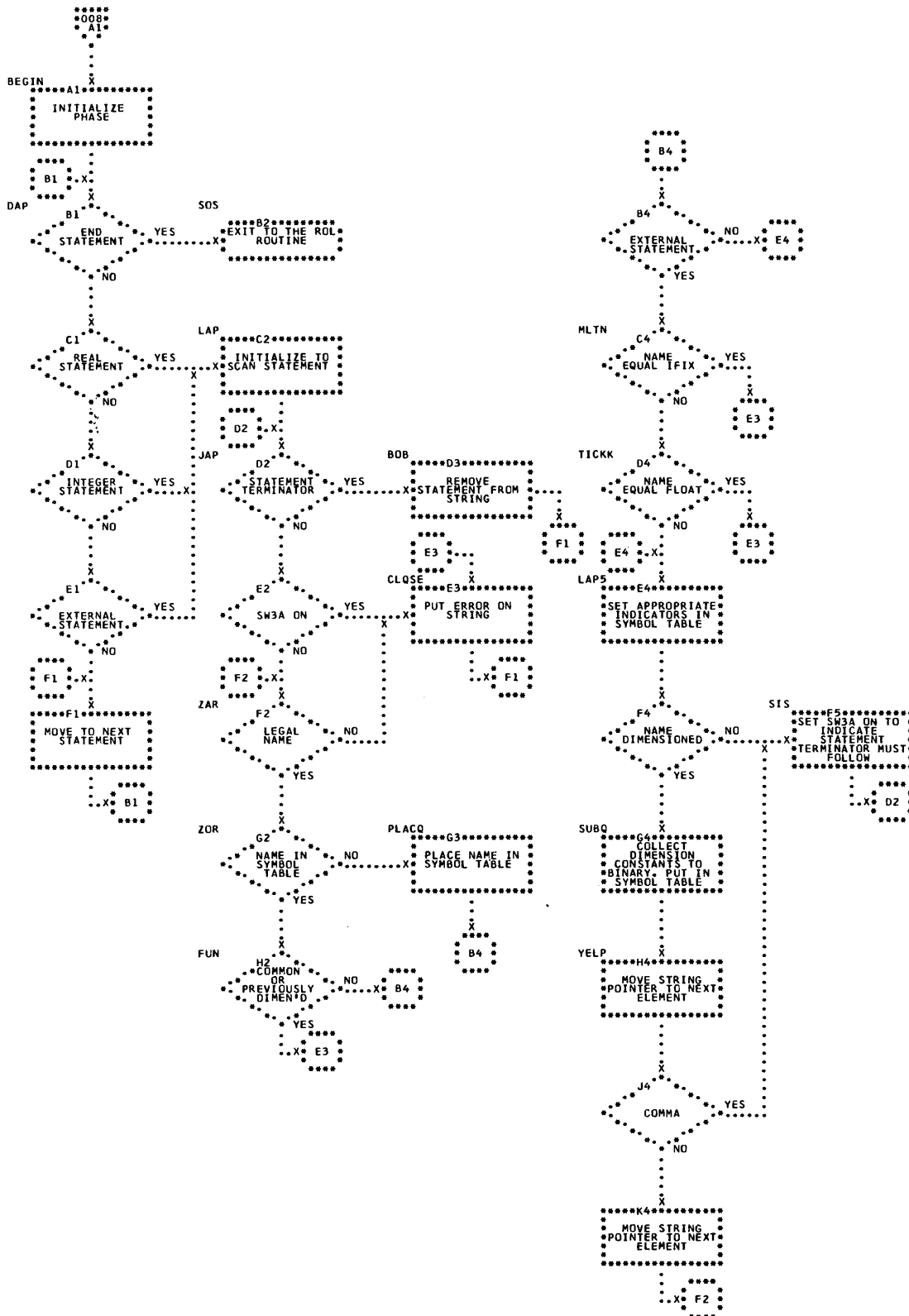
Flowchart FOR05. FORTRAN Compiler, Phase 4



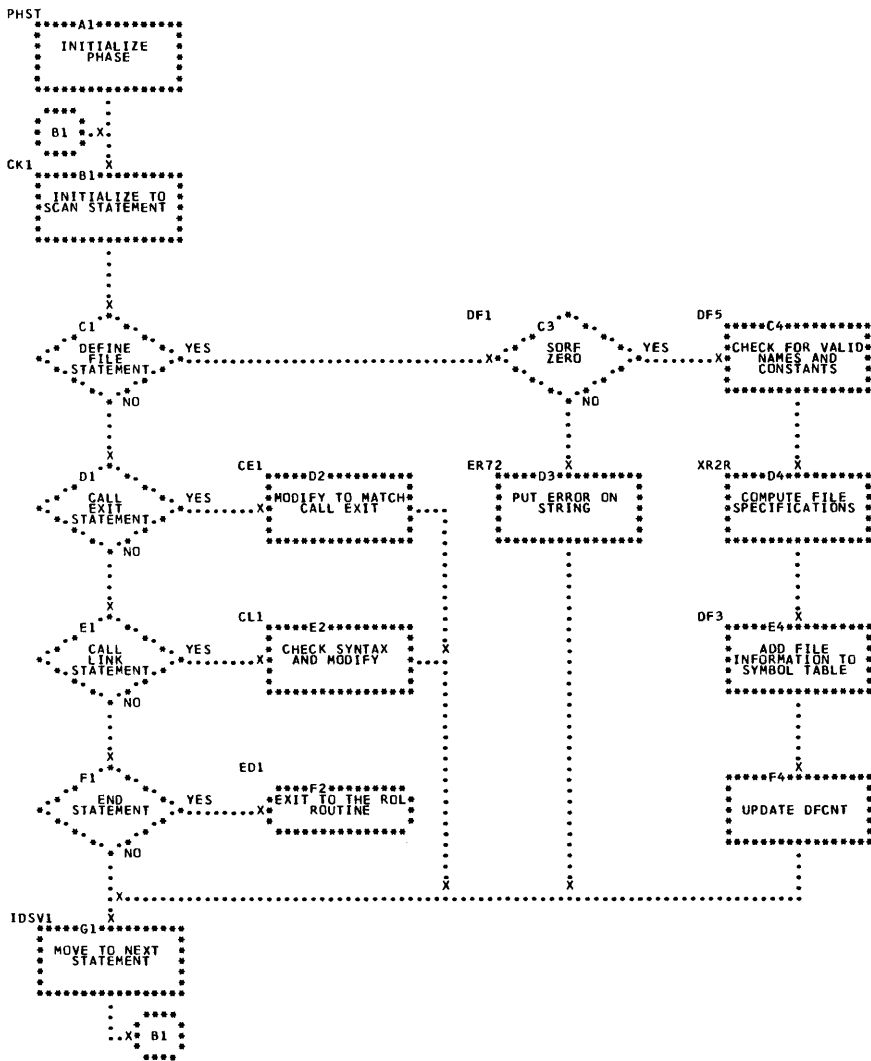
Flowchart FOR06. FORTRAN Compiler, Phase 4



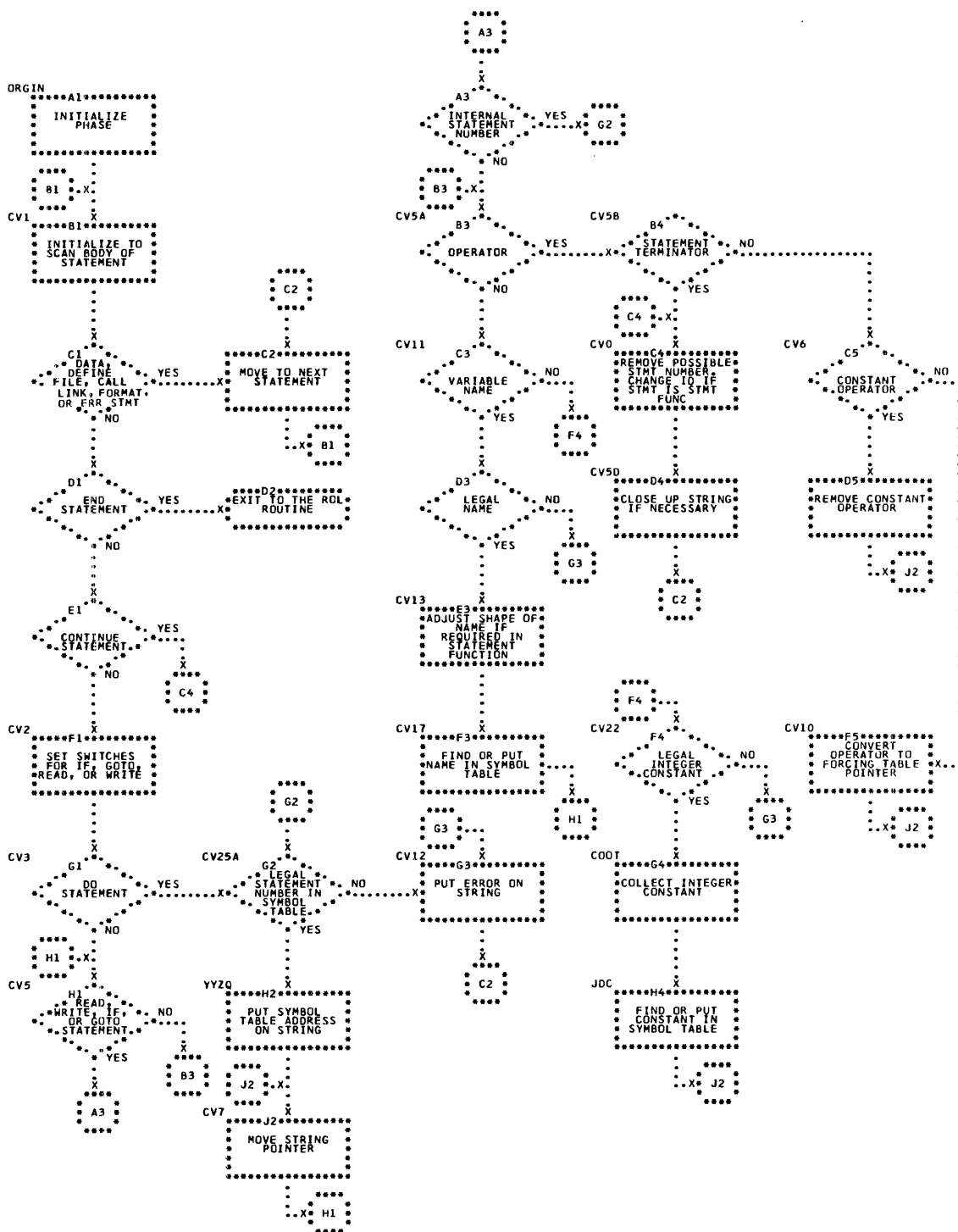
Flowchart FOR07. FORTRAN Compiler, Phase 5



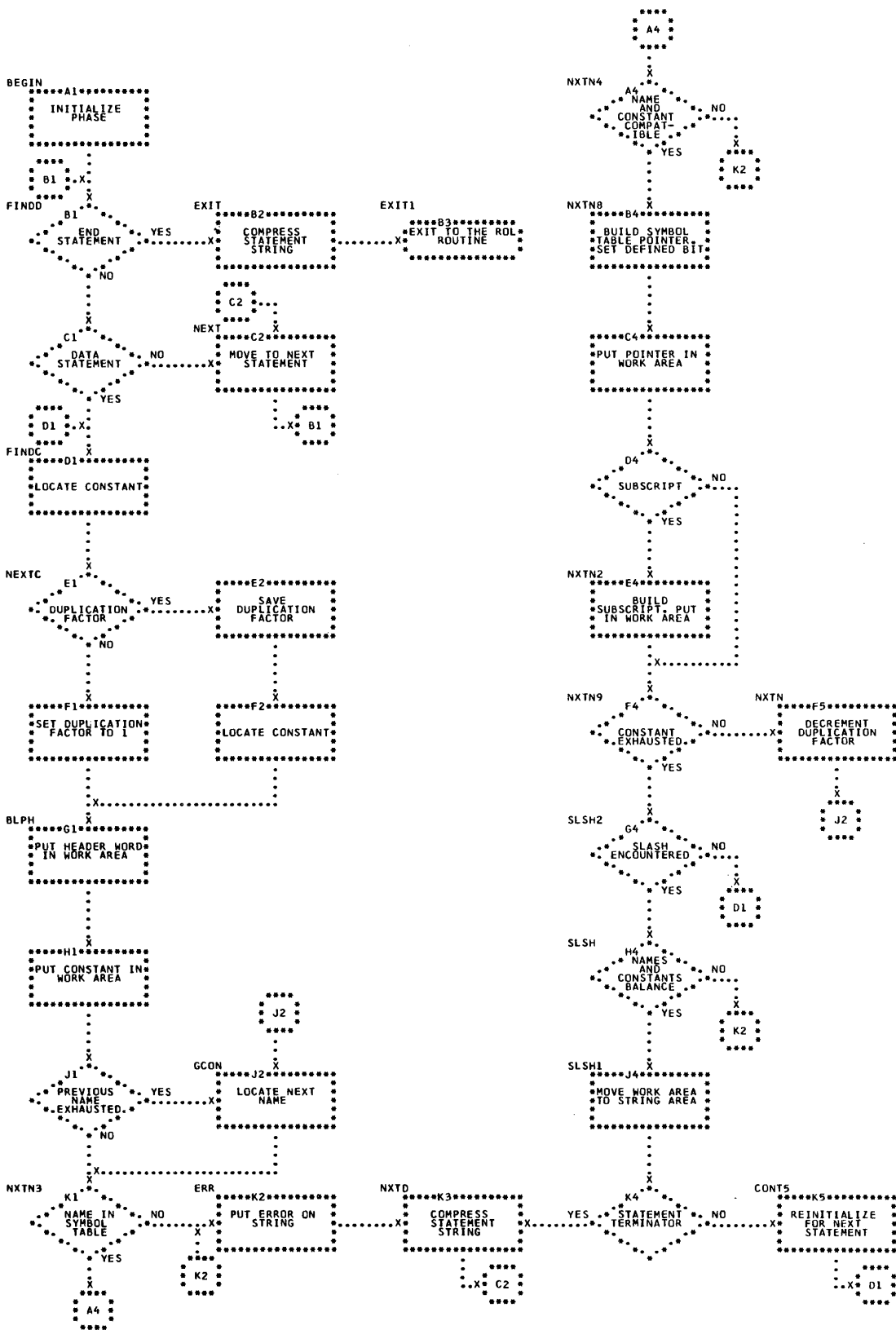
Flowchart FOR08, FORTRAN Compiler, Phase 5



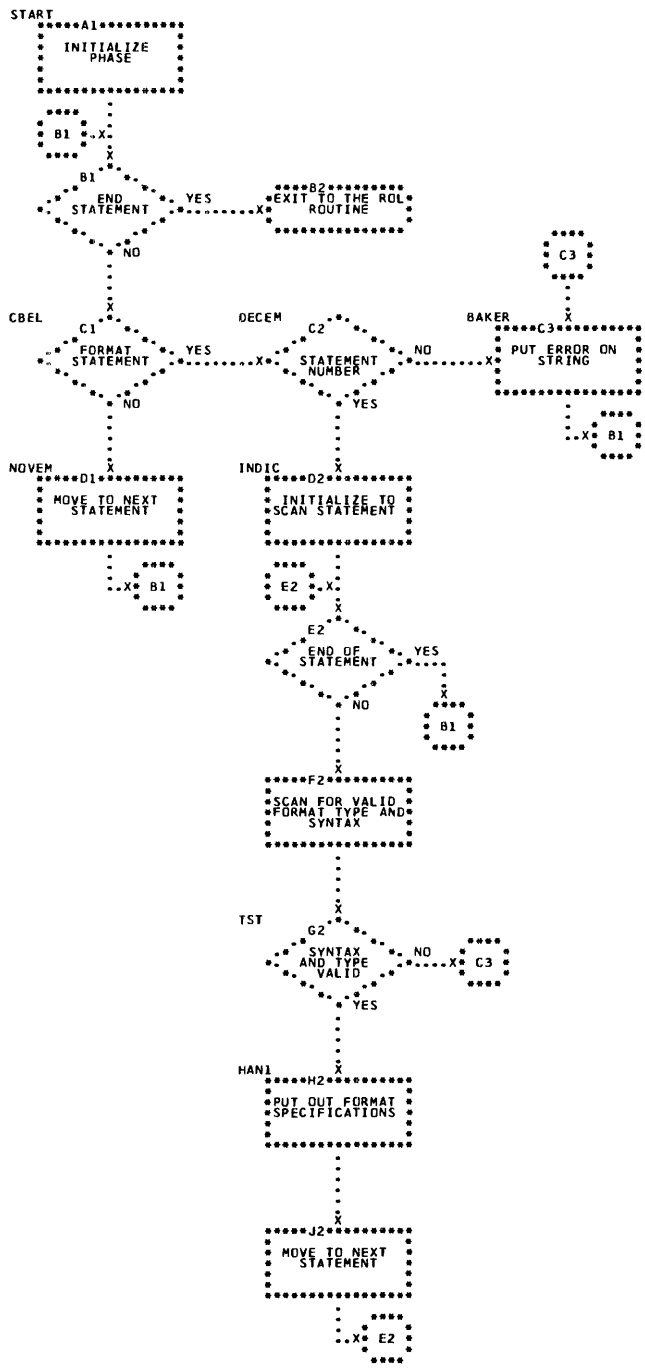
Flowchart FOR10. FORTRAN Compiler, Phase 7



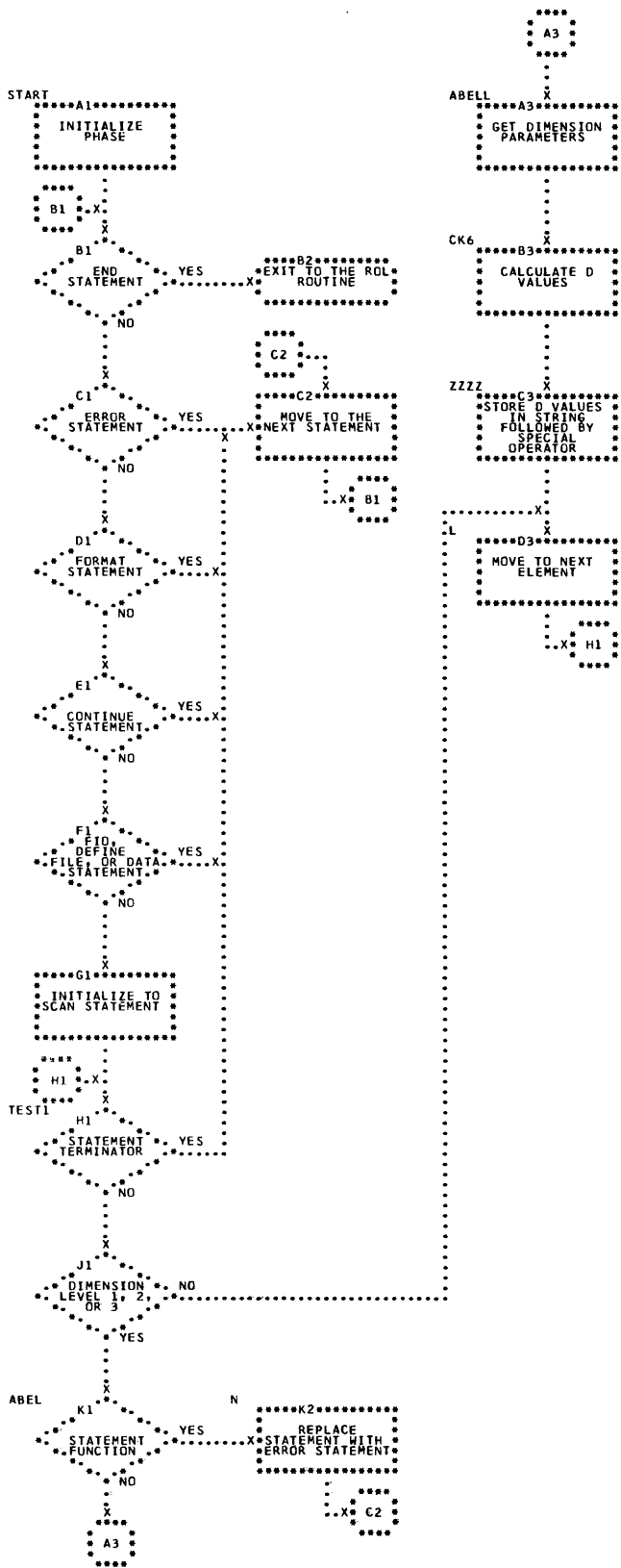
Flowchart FOR11. FORTRAN Compiler, Phase 8



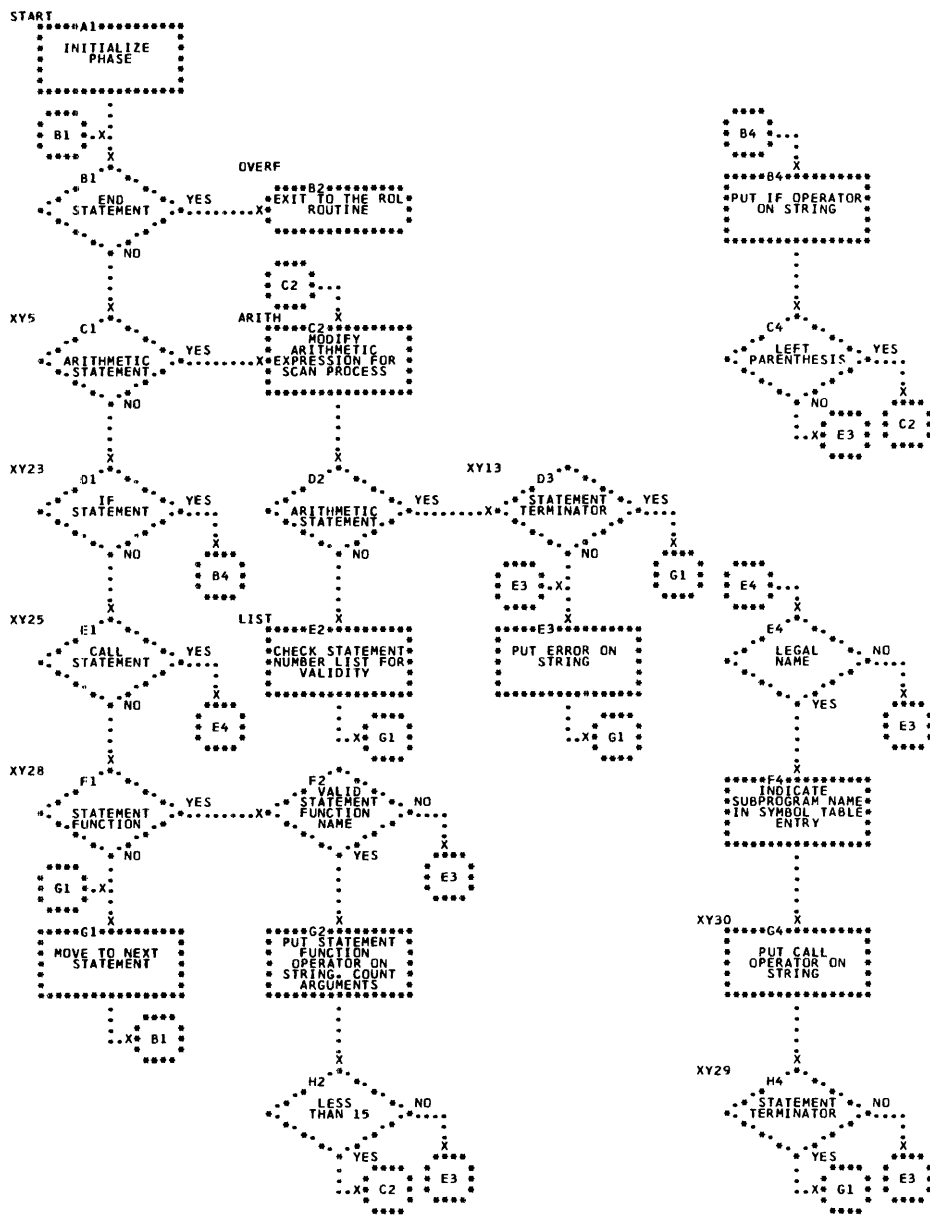
Flowchart FOR12. FORTRAN Compiler, Phase 9



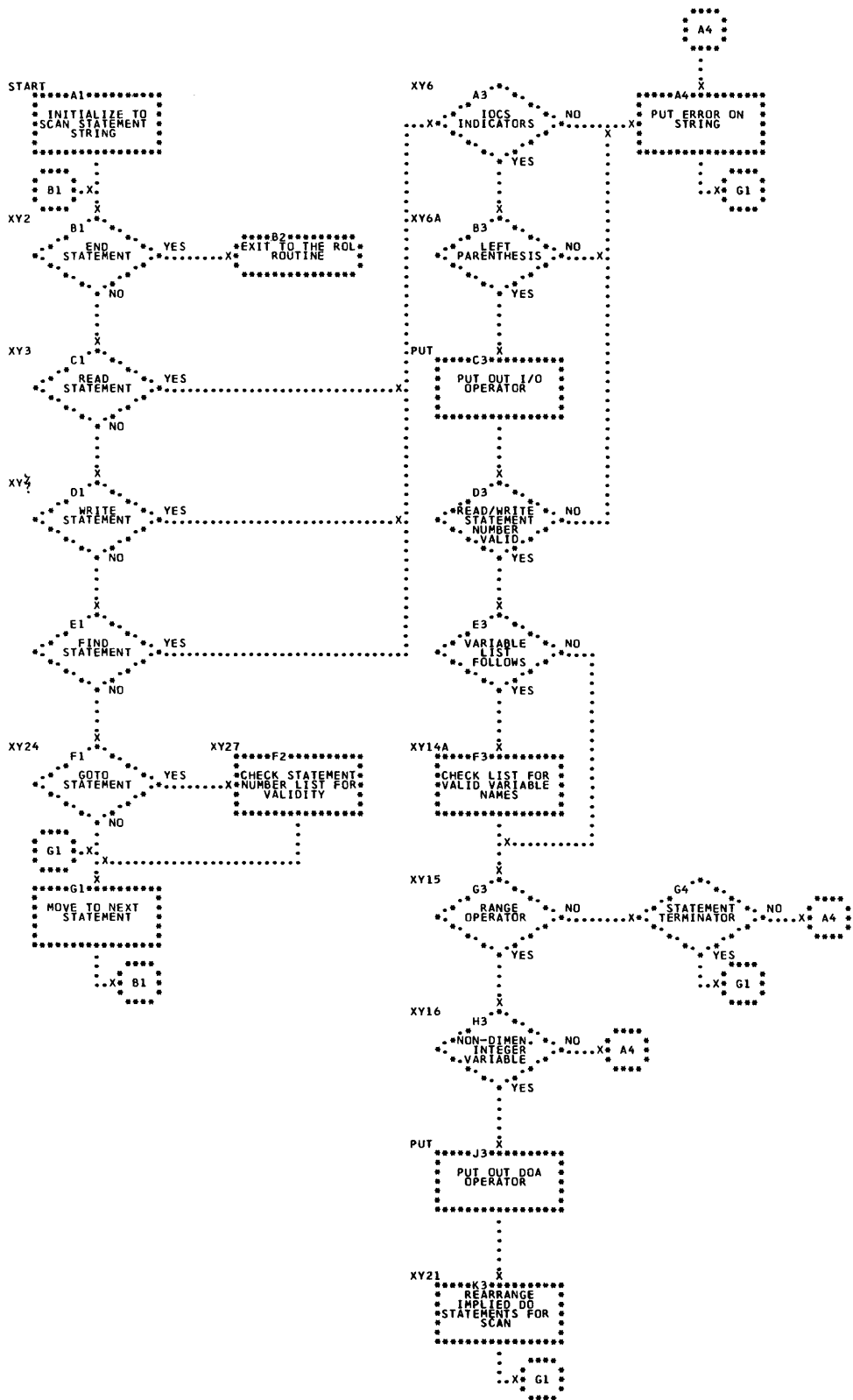
Flowchart FOR13. FORTRAN Compiler, Phase 10



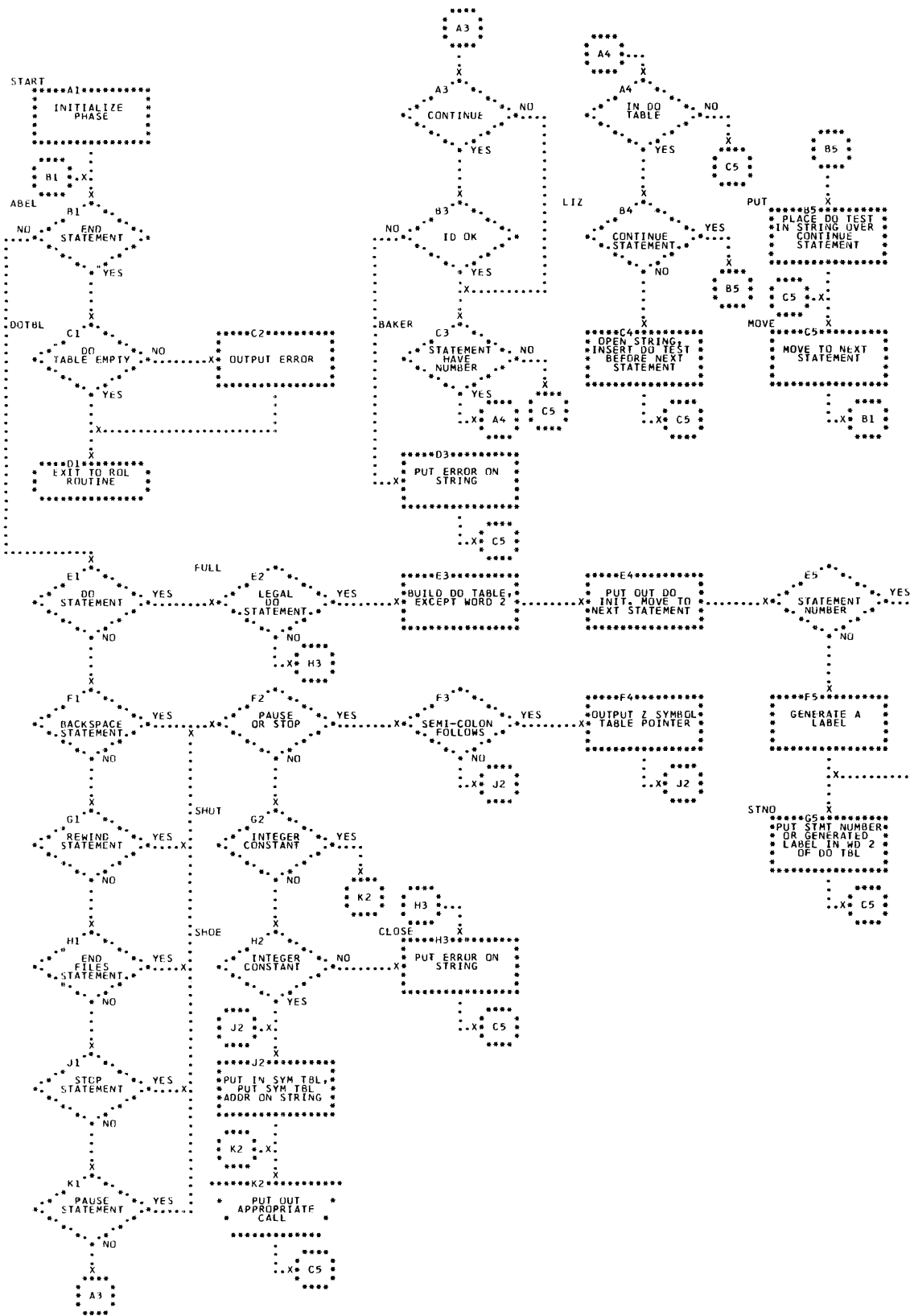
Flowchart FOR14. FORTRAN Compiler, Phase 11



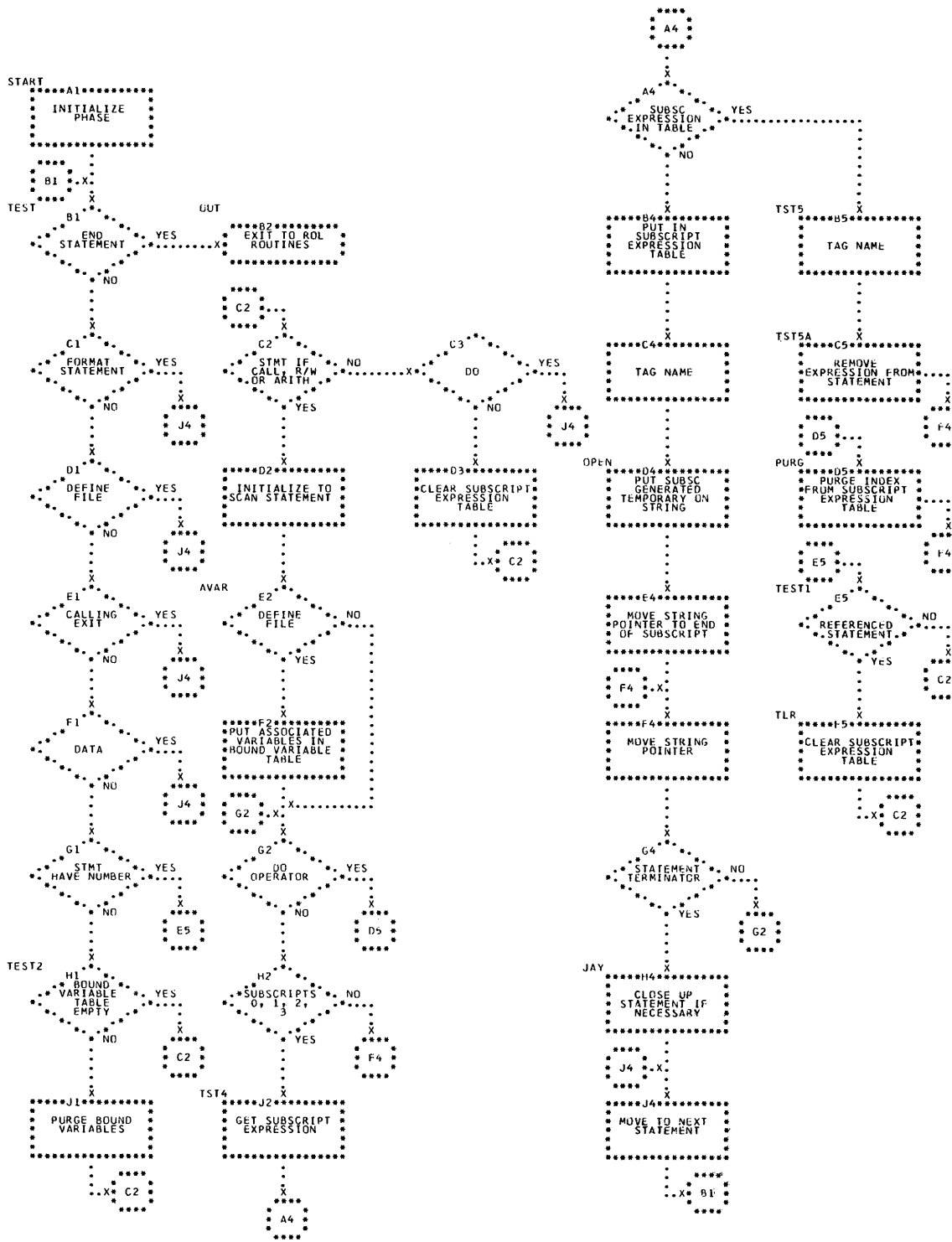
Flowchart FOR15. FORTRAN Compiler, Phase 12



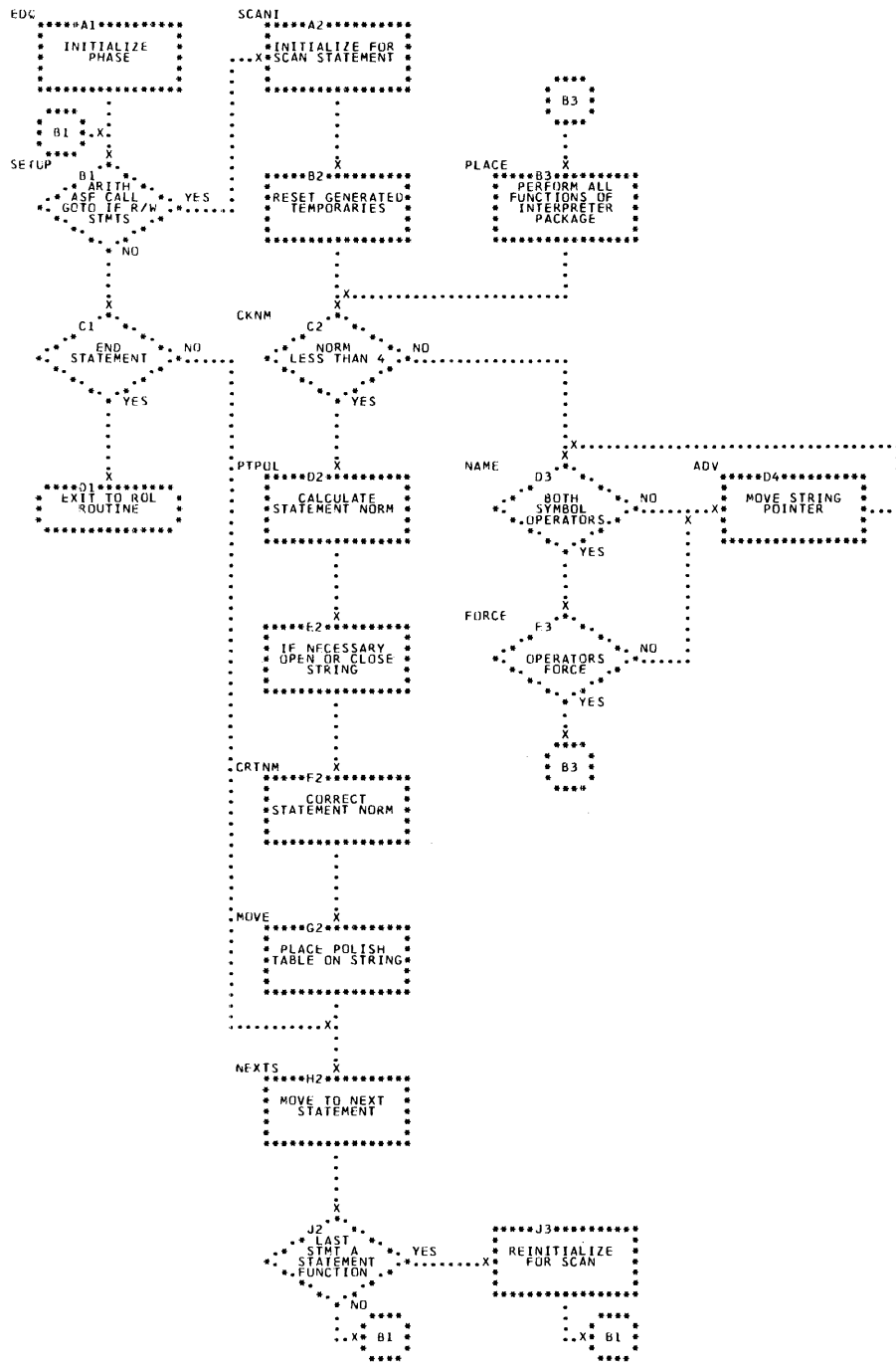
Flowchart FOR16. FORTRAN Compiler, Phase 13



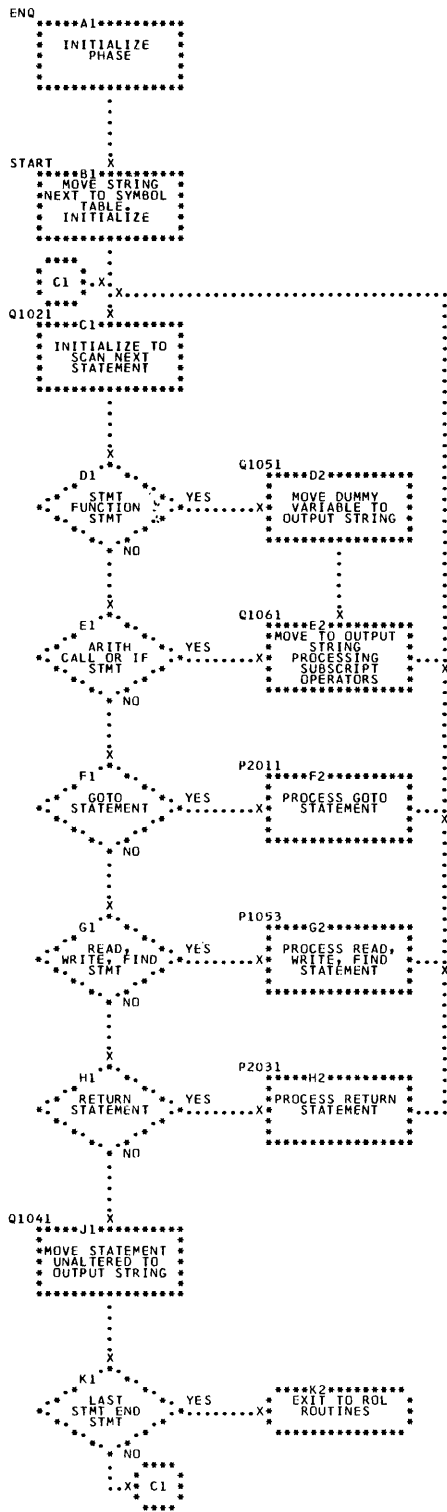
Flowchart FOR17. FORTRAN Compiler, Phase 14



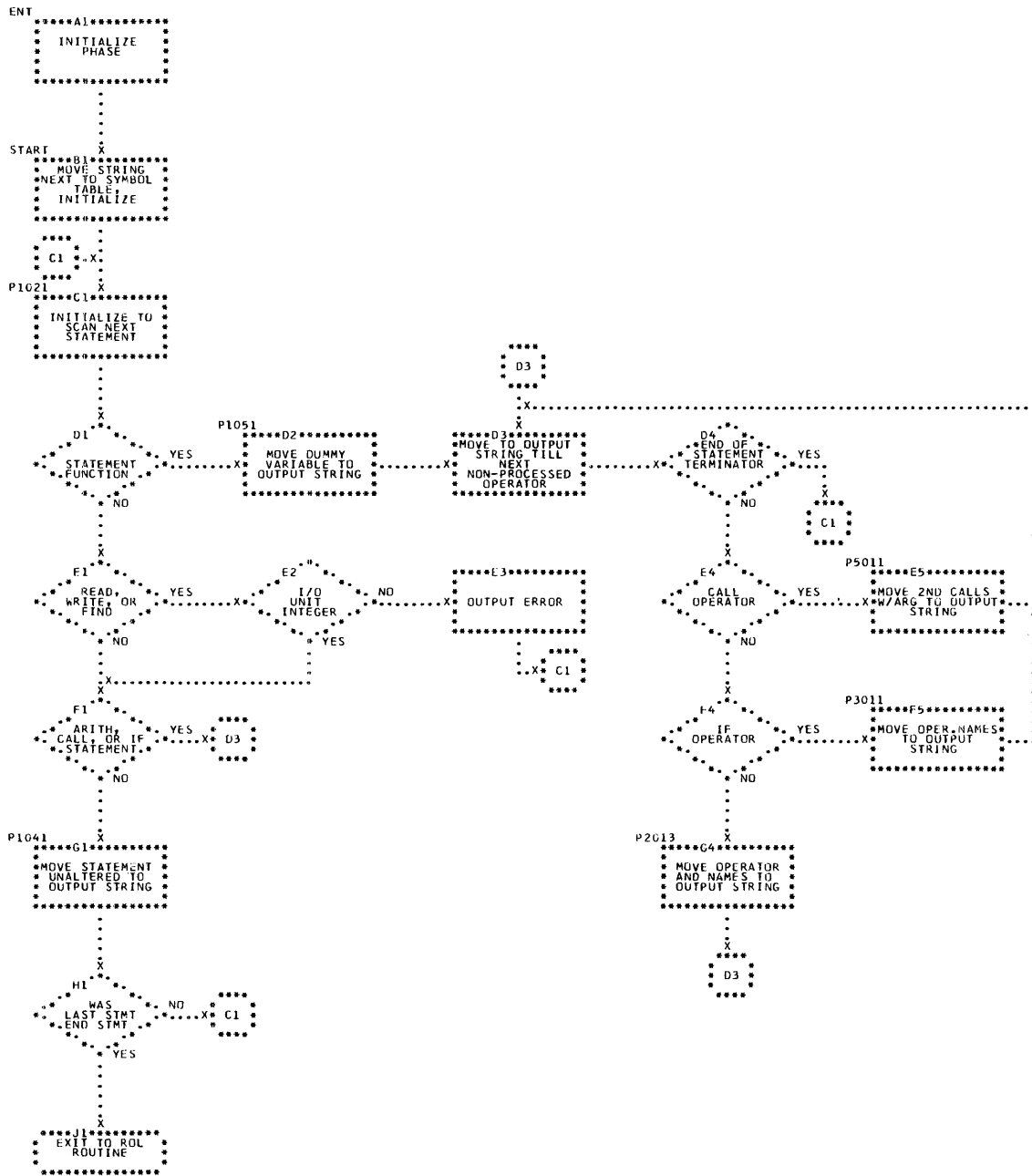
Flowchart FOR18. FORTRAN Compiler, Phase 15



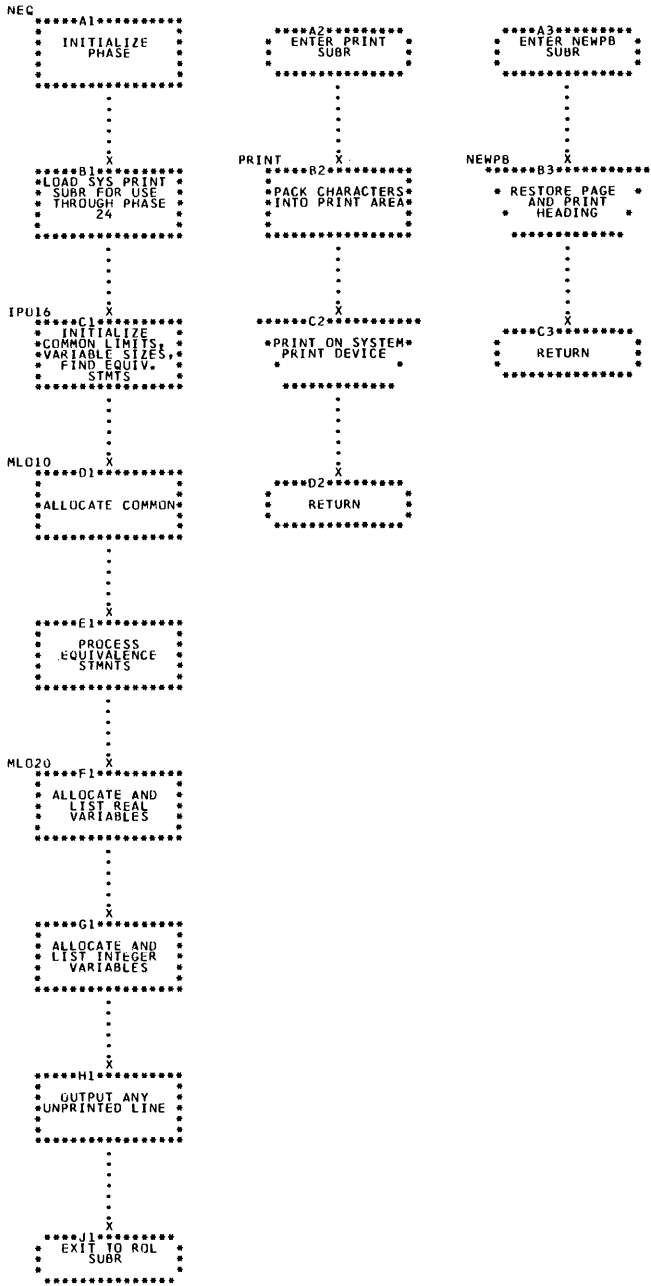
Flowchart FOR19. FORTRAN Compiler, Phase 16



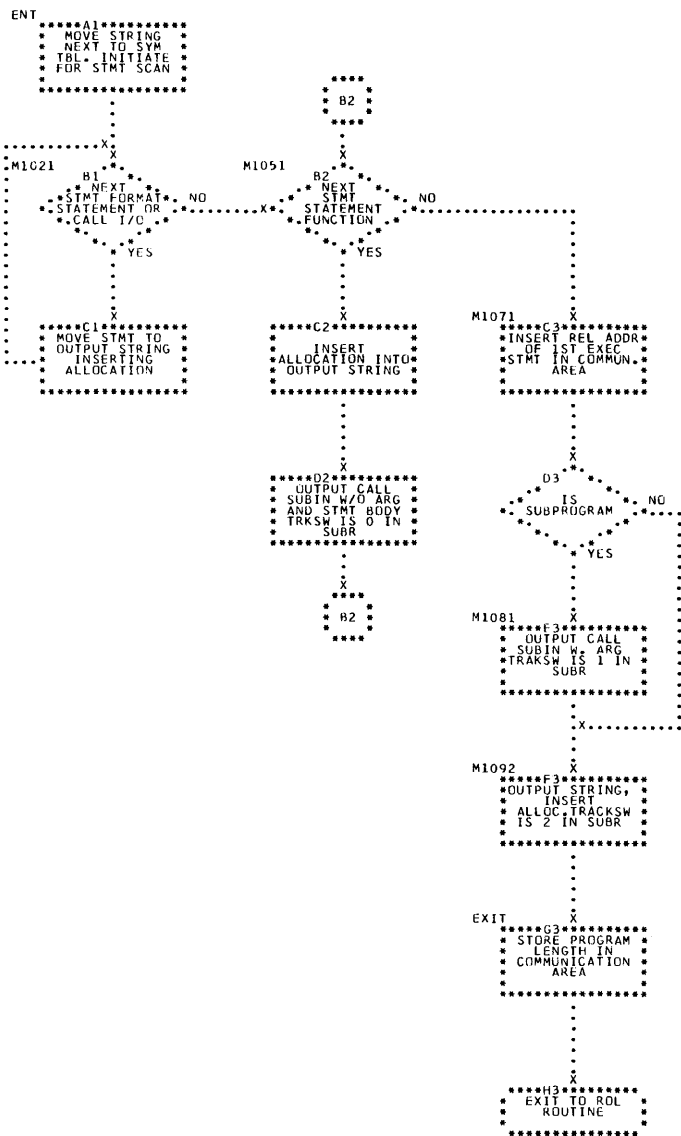
Flowchart FOR20. FORTRAN Compiler, Phase 17



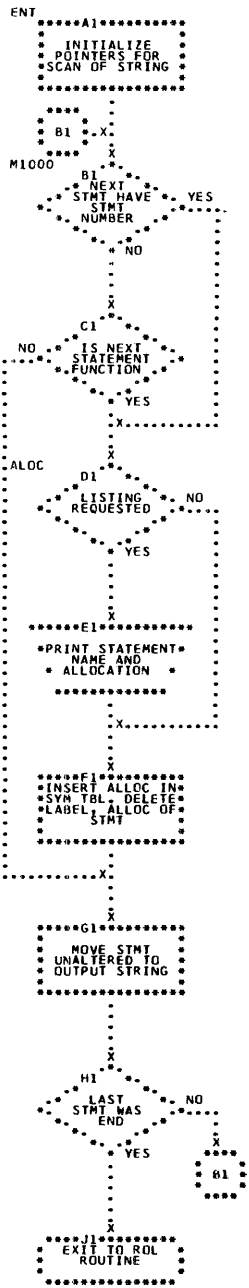
Flowchart FOR21. FORTRAN Compiler, Phase 18



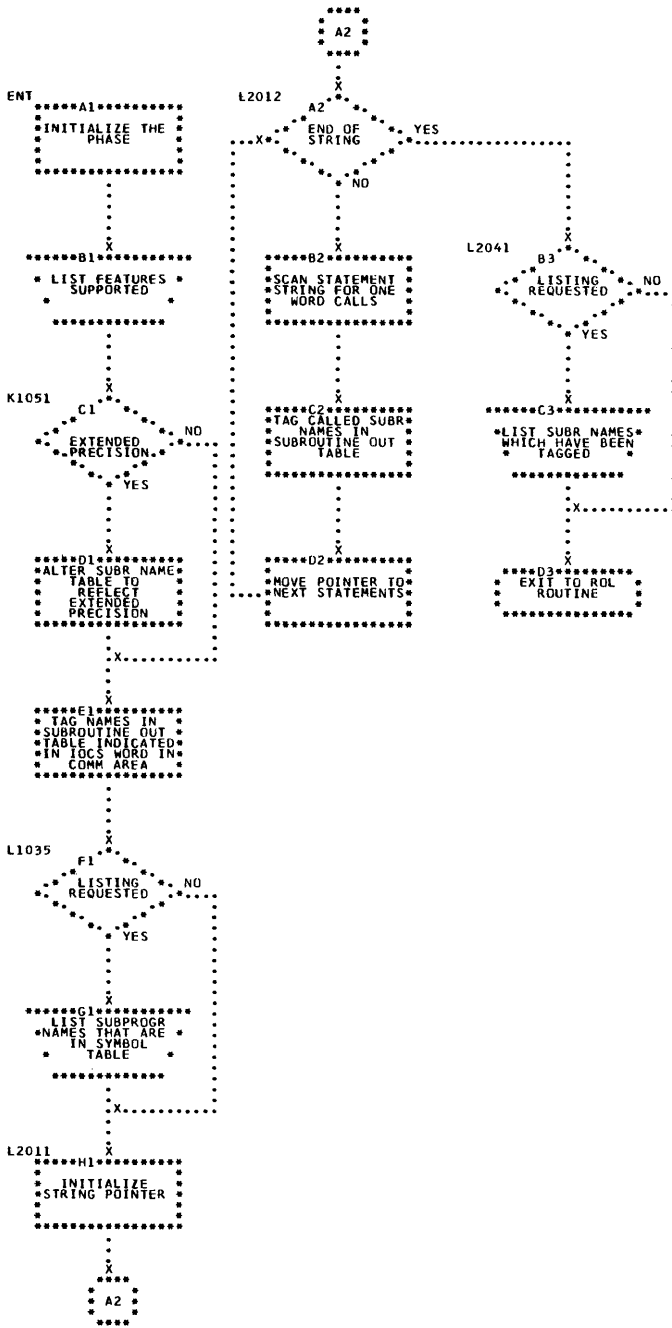
Flowchart FOR22. FORTRAN Compiler, Phase 19



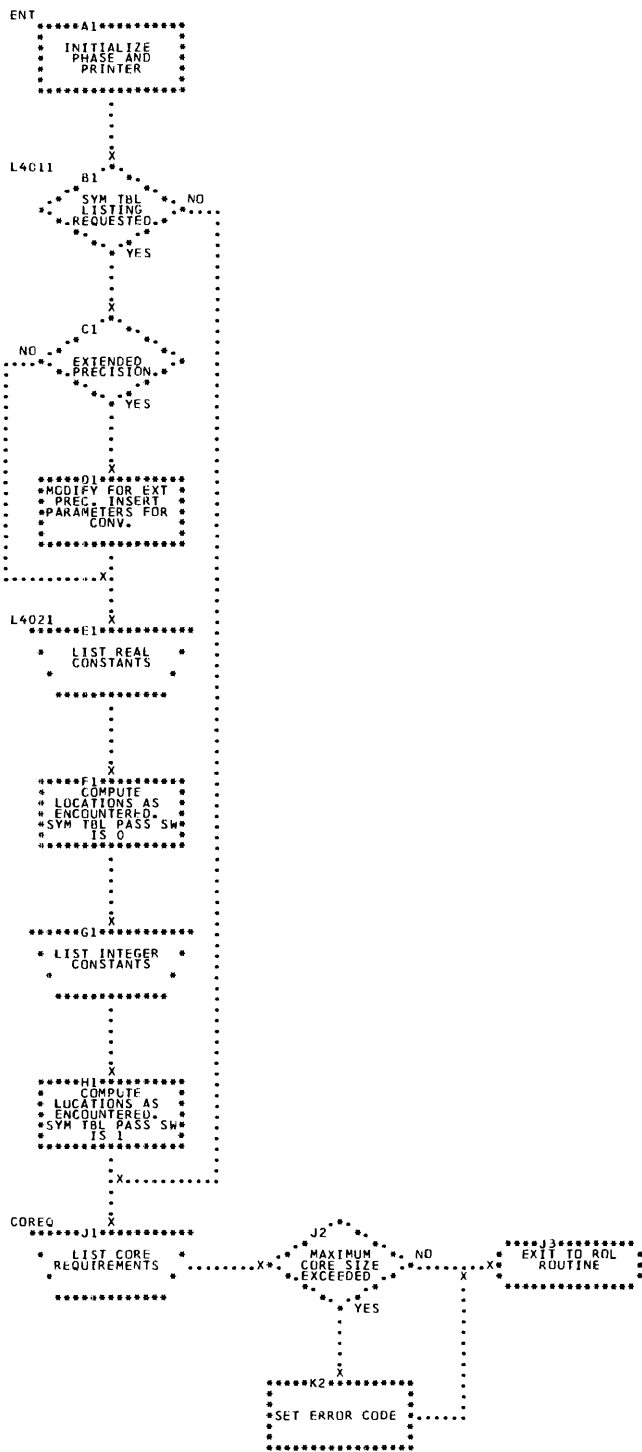
Flowchart FOR24. FORTRAN Compiler, Phase 21



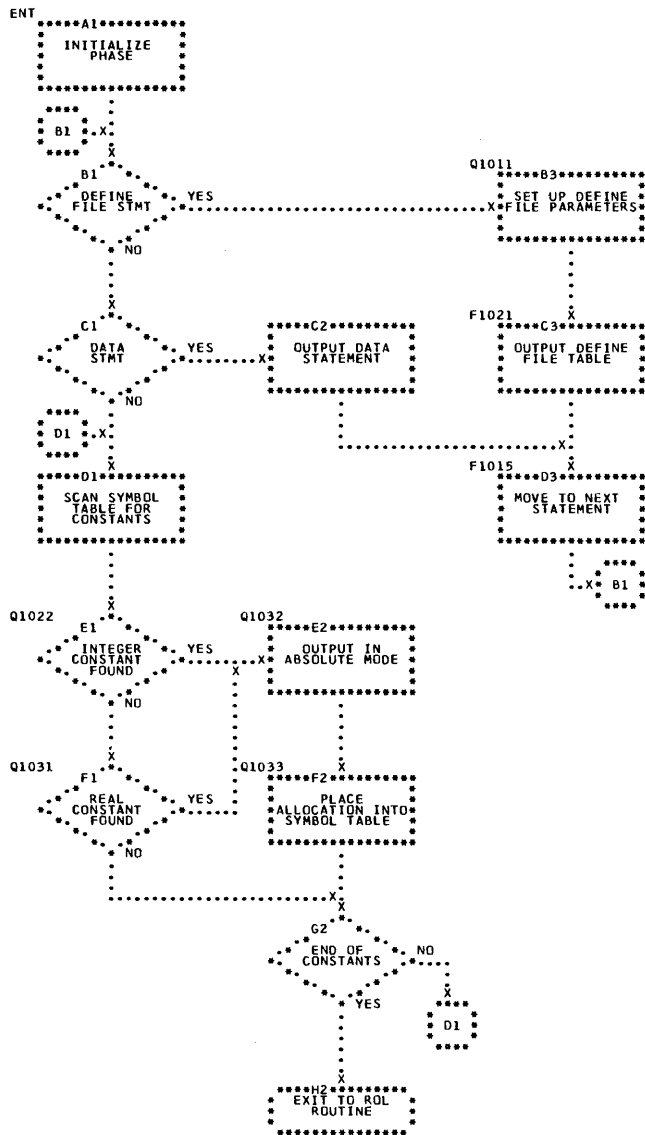
Flowchart FOR25. FORTRAN Compiler, Phase 22



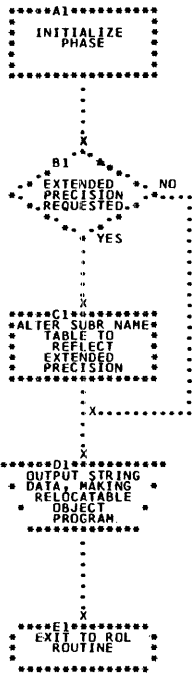
Flowchart FOR26. FORTRAN Compiler, Phase 23



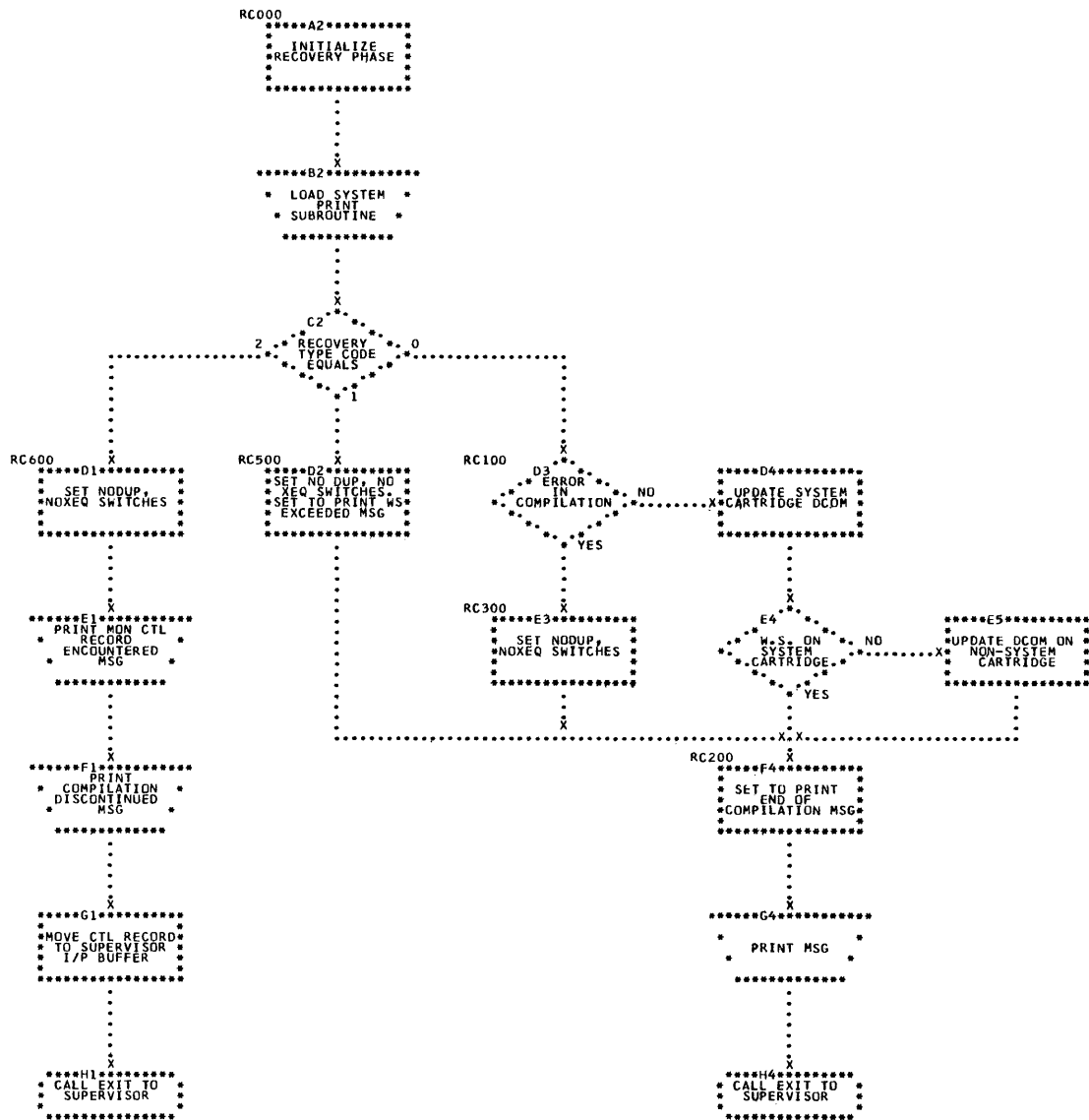
Flowchart FOR27. FORTRAN Compiler, Phase 24



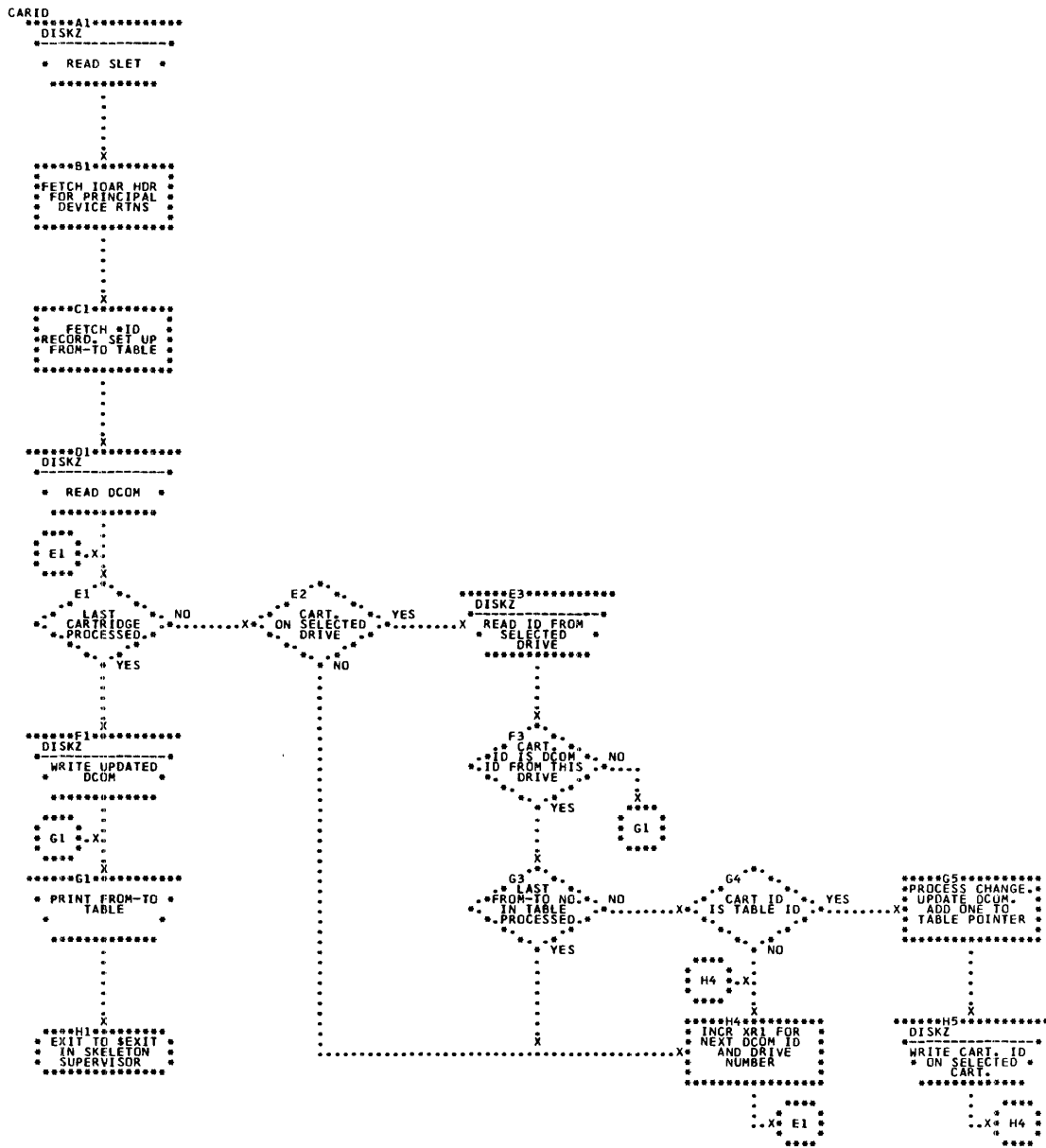
Flowchart FOR28. FORTRAN Compiler, Phase 25



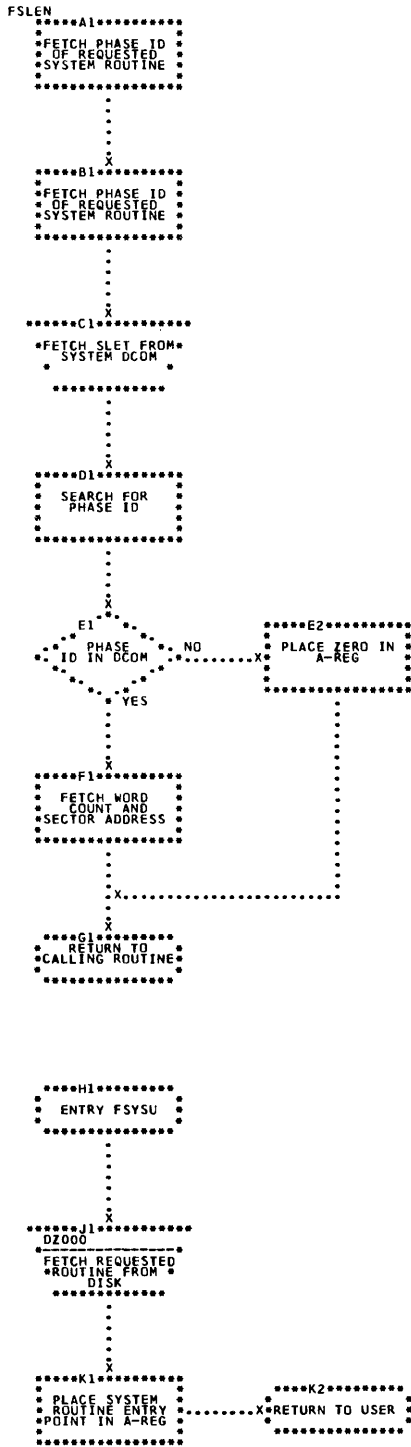
Flowchart FOR29. FORTRAN Compiler, Phase 26



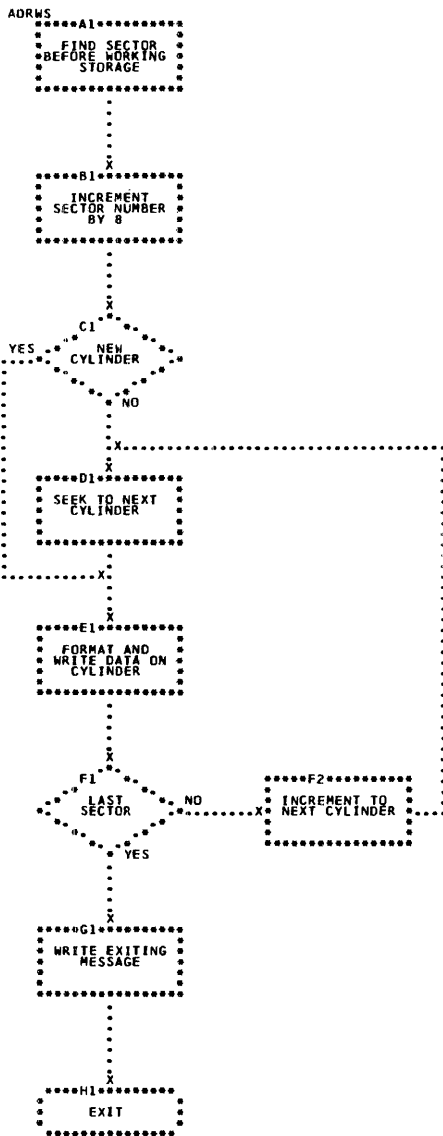
Flowchart FOR30. FORTRAN Compiler, Phase 27



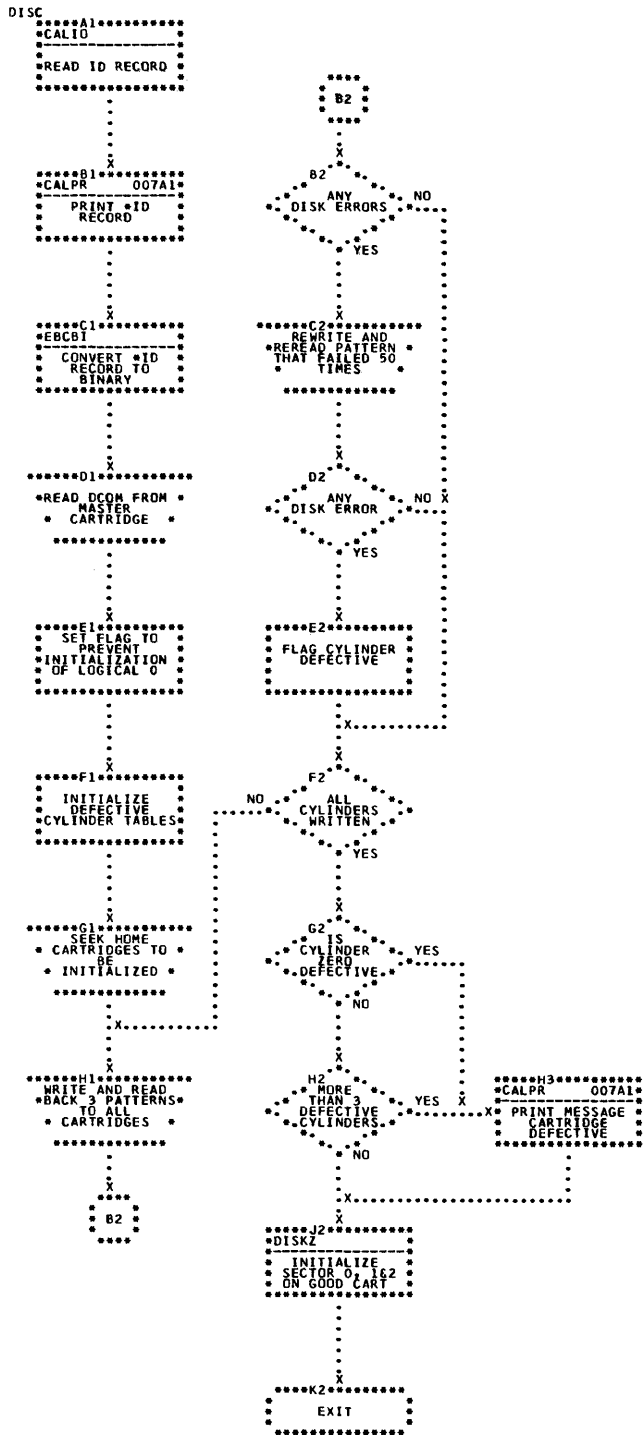
Flowchart UTL01. System Library, ID



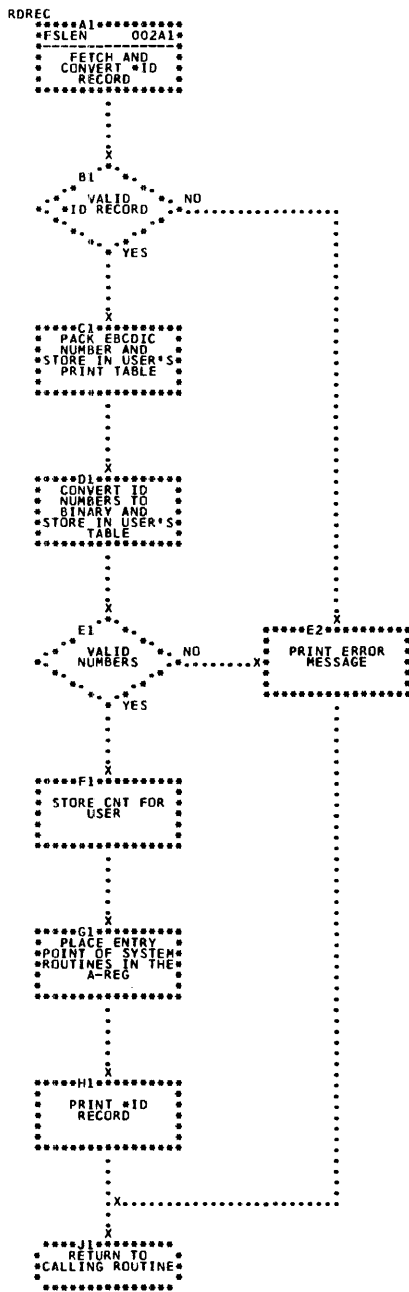
Flowchart UTLO2. System Library, FSLFN/FSYSU



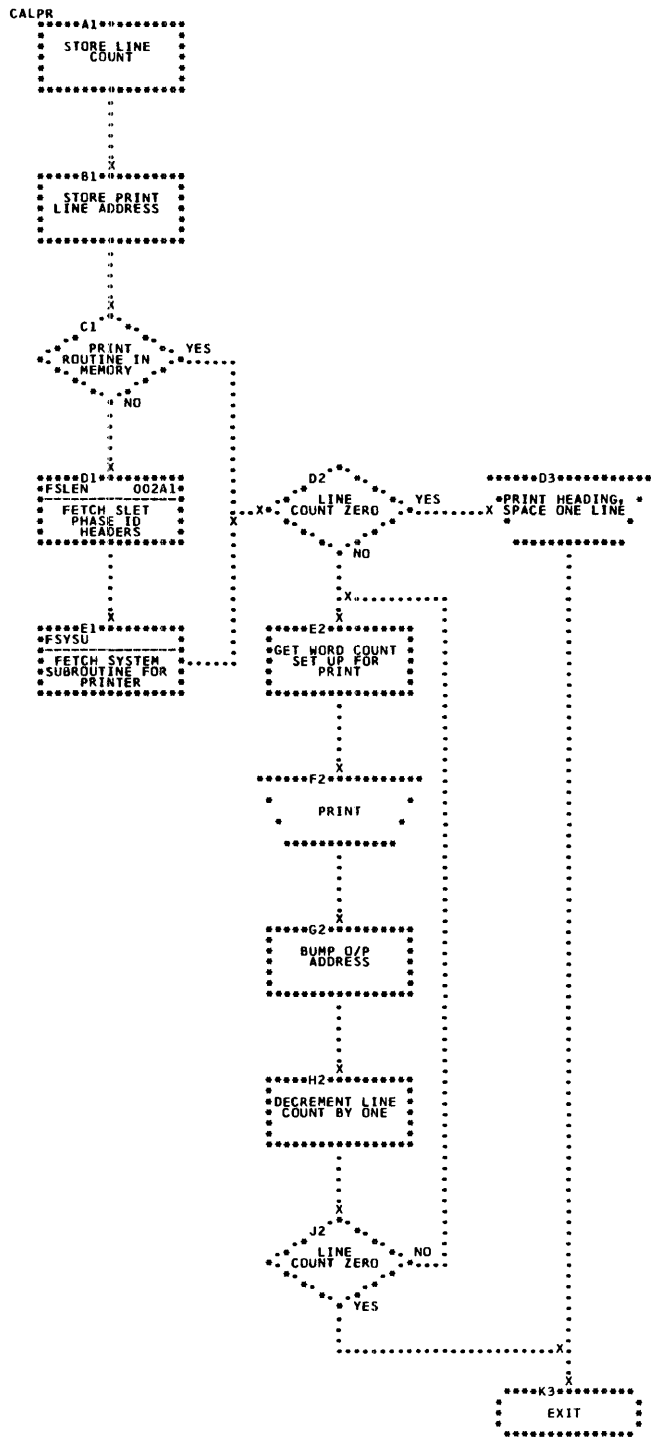
Flowchart UTL03. System Library, ADRWS



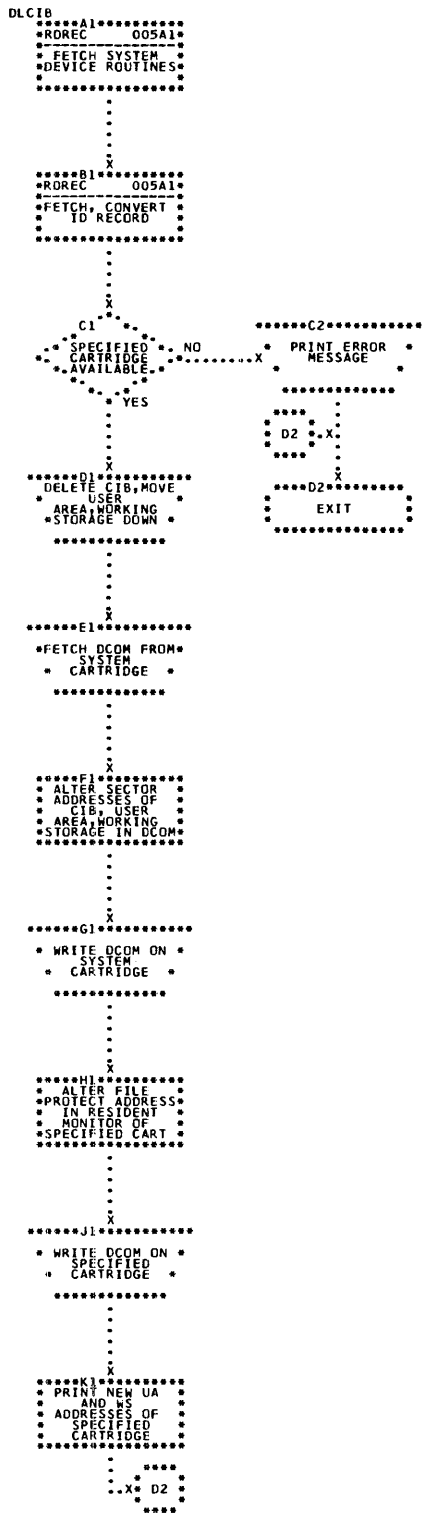
Flowchart UTL04. System Library, DISC



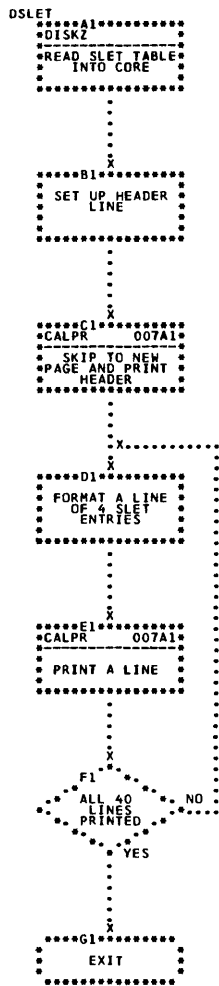
Flowchart UTL05. System Library, RDREC



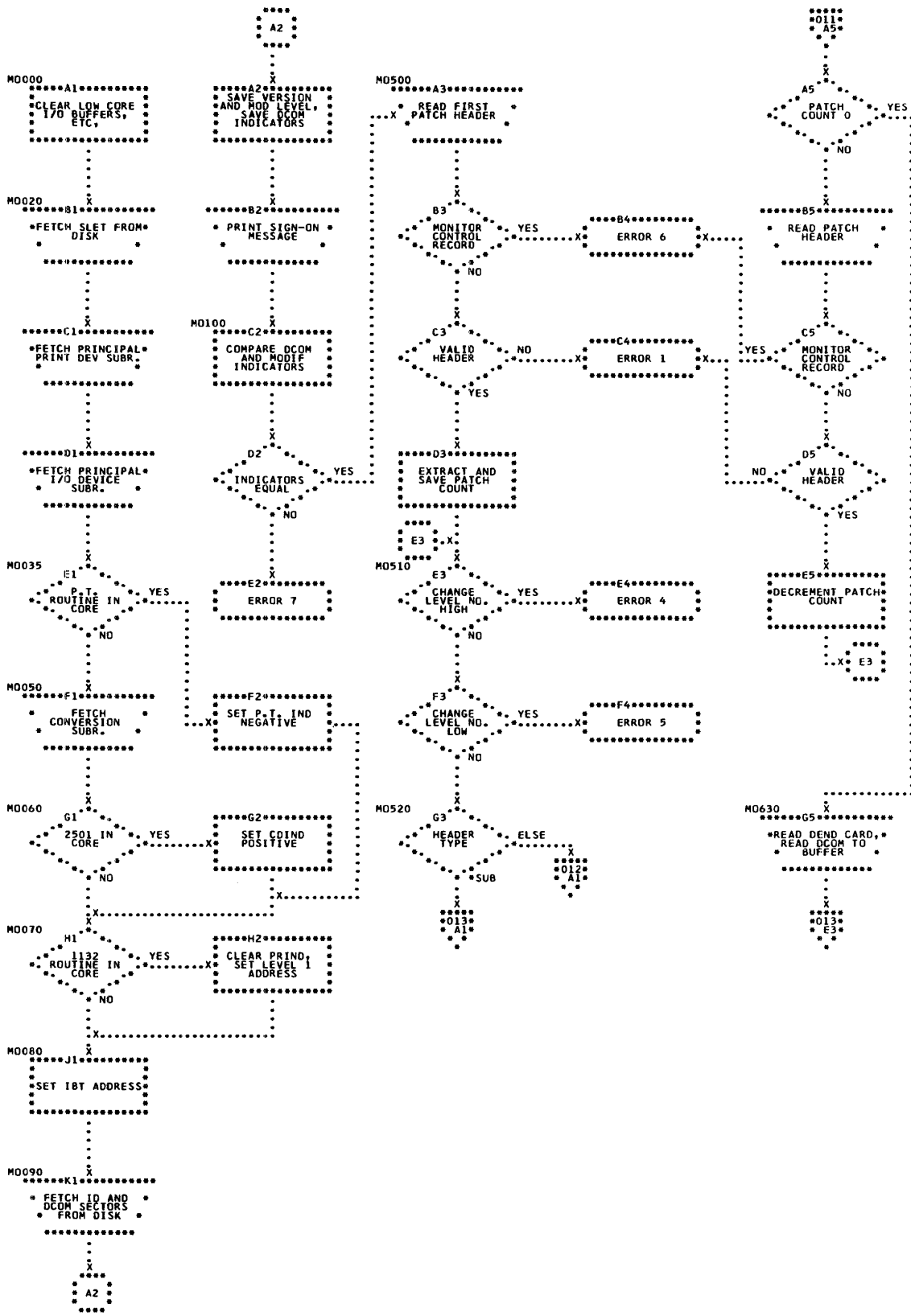
Flowchart UTL07. System Library, CALPR



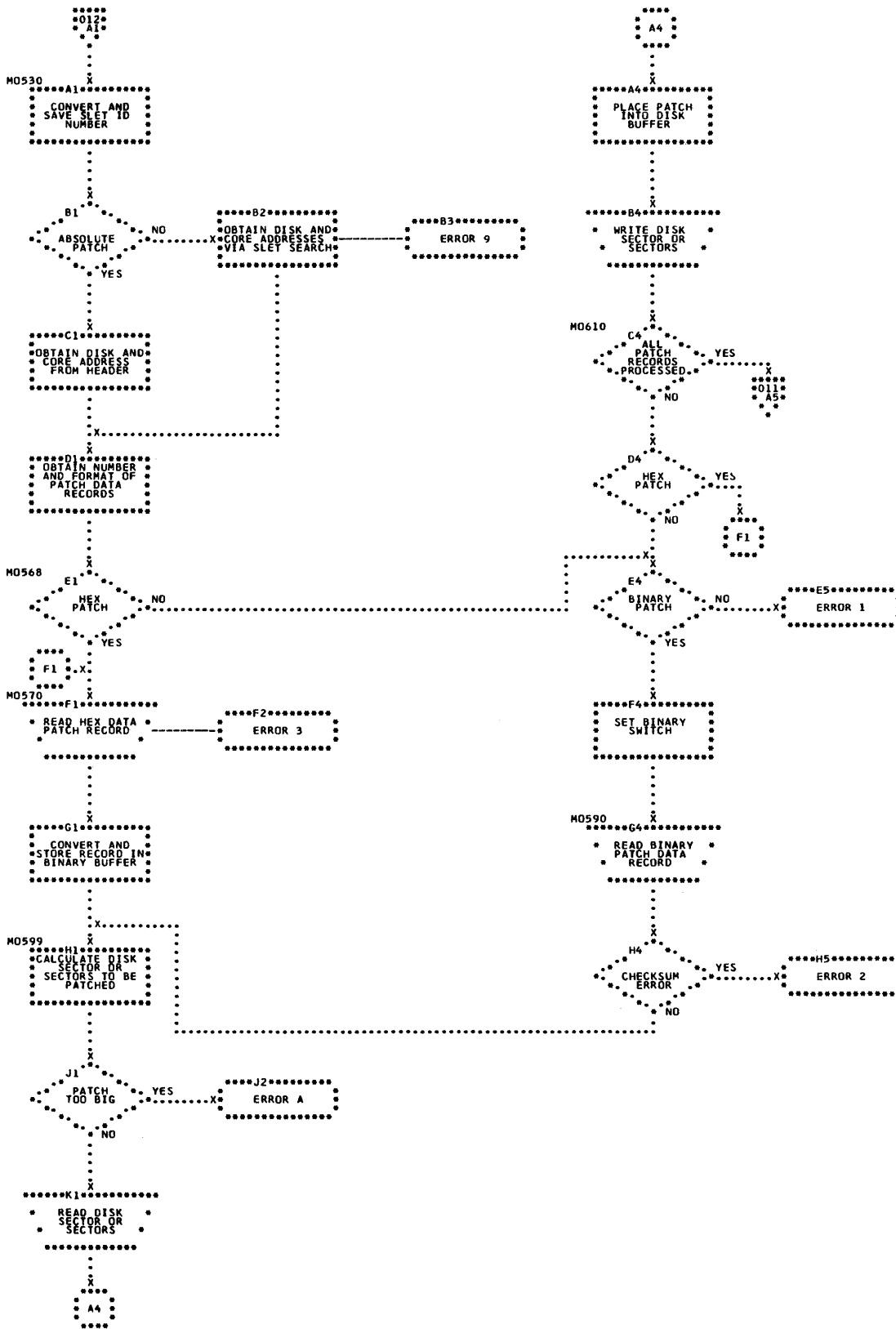
Flowchart UTL09. System Library, DLCIB



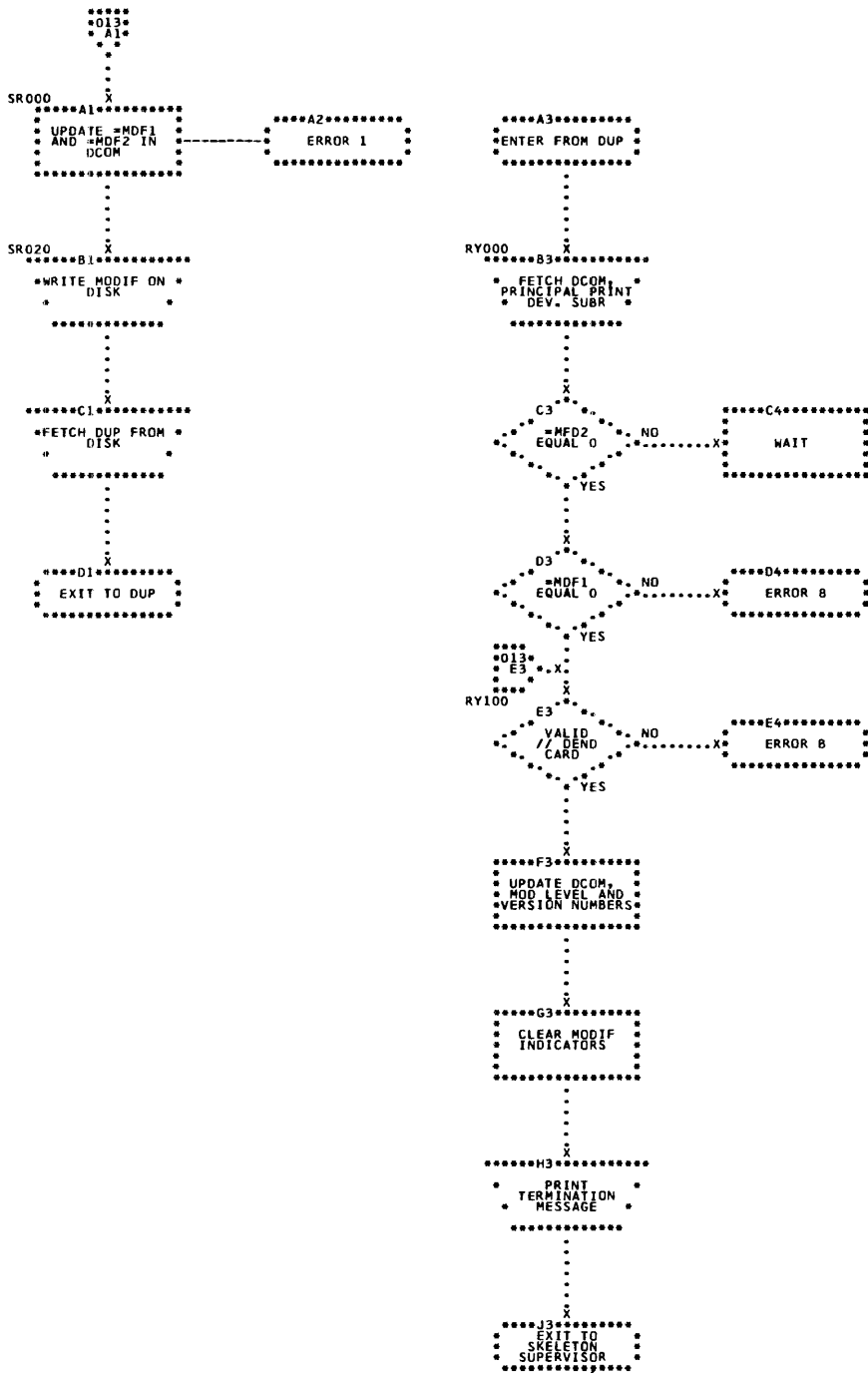
Flowchart UTL10. System Library, DSLET



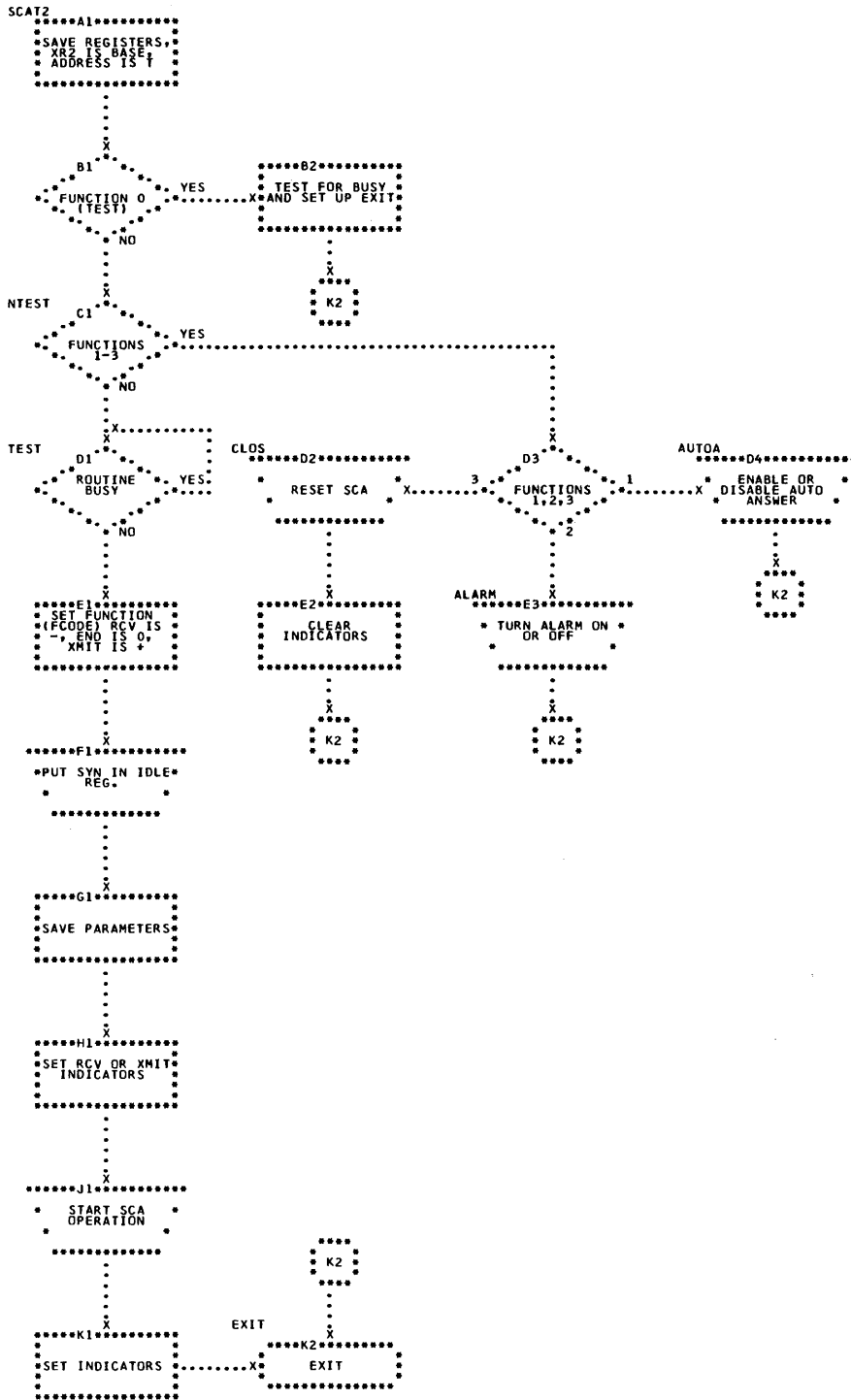
Flowchart: UTL11. System Library, MODIF



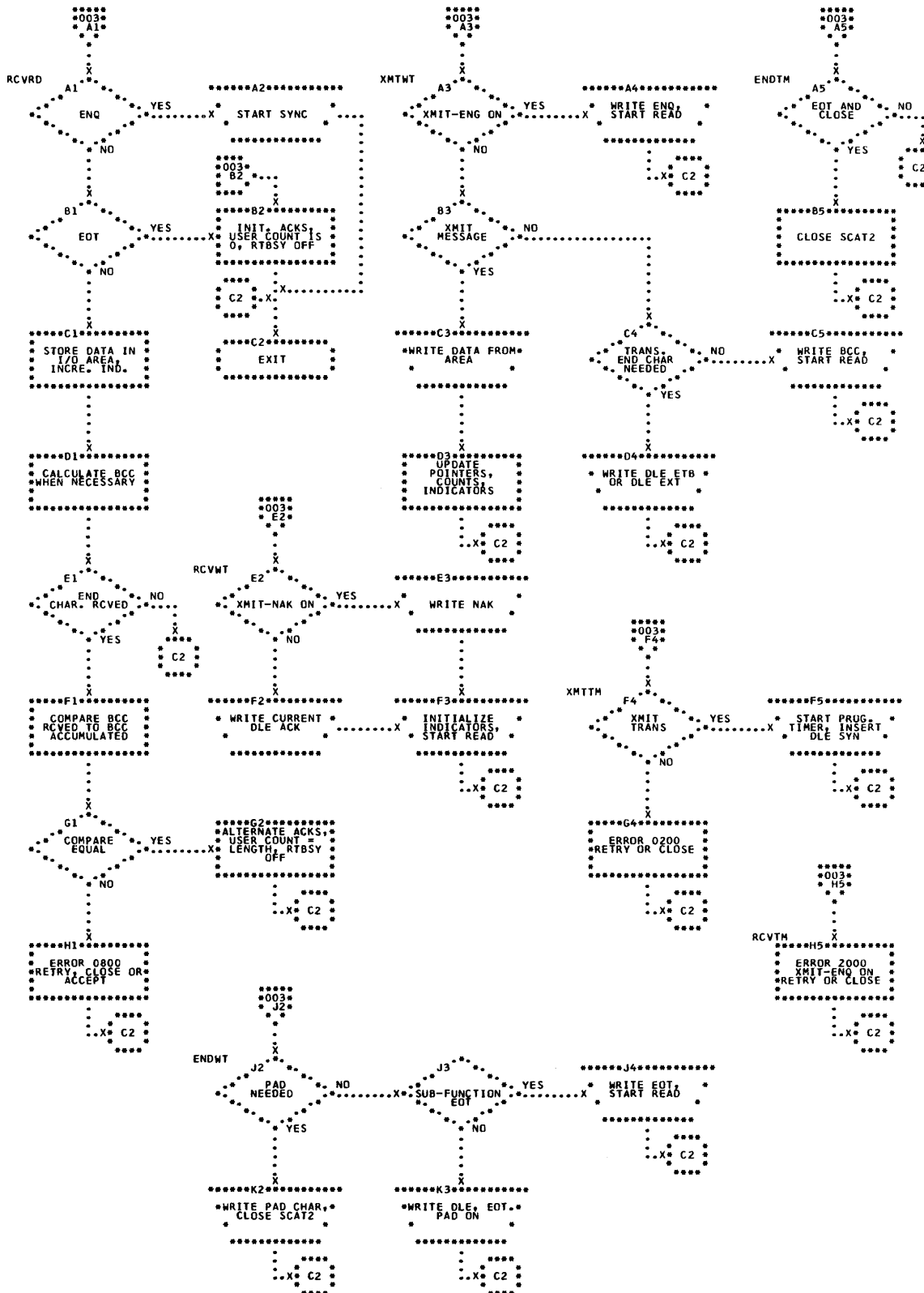
Flowchart UTL12. System Library, MODIF



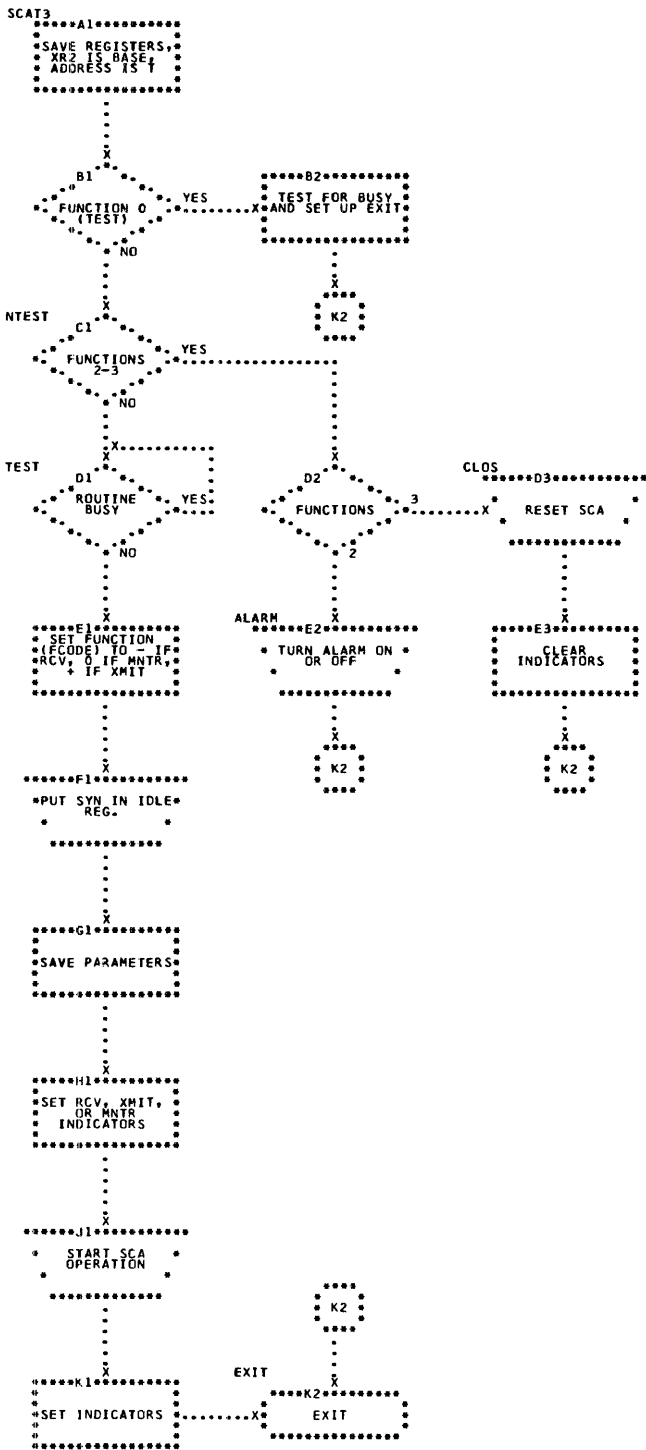
Flowchart UTL13. System Library, MODIF



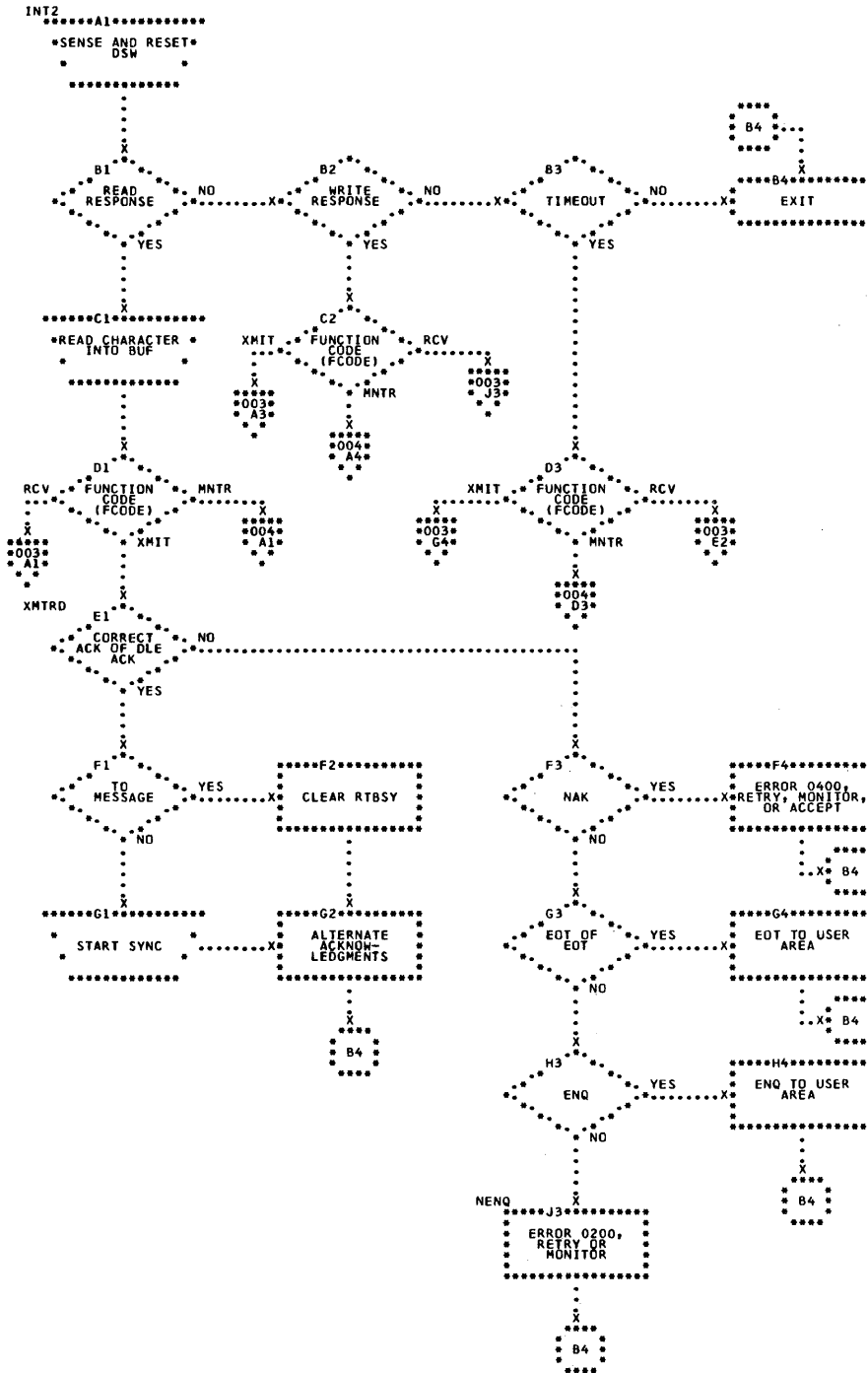
Flowchart SCA01. System Library, SCAT2 Call Processing



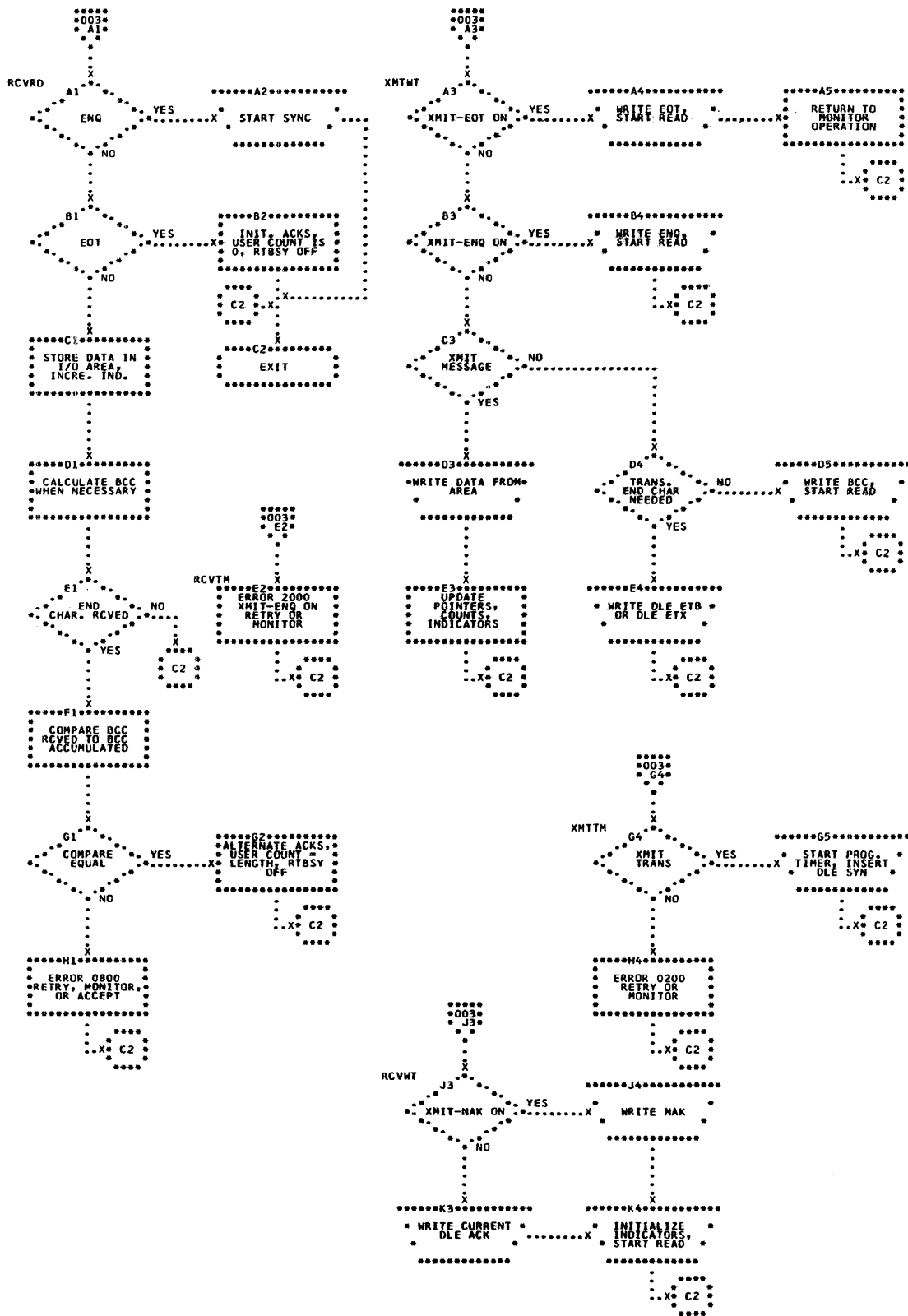
Flowchart SCA03, System Library, SCAT2 Interrupt Processing



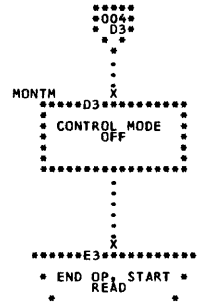
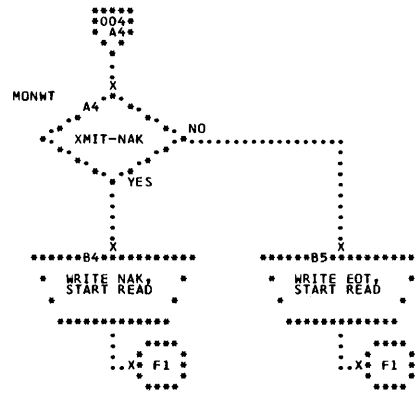
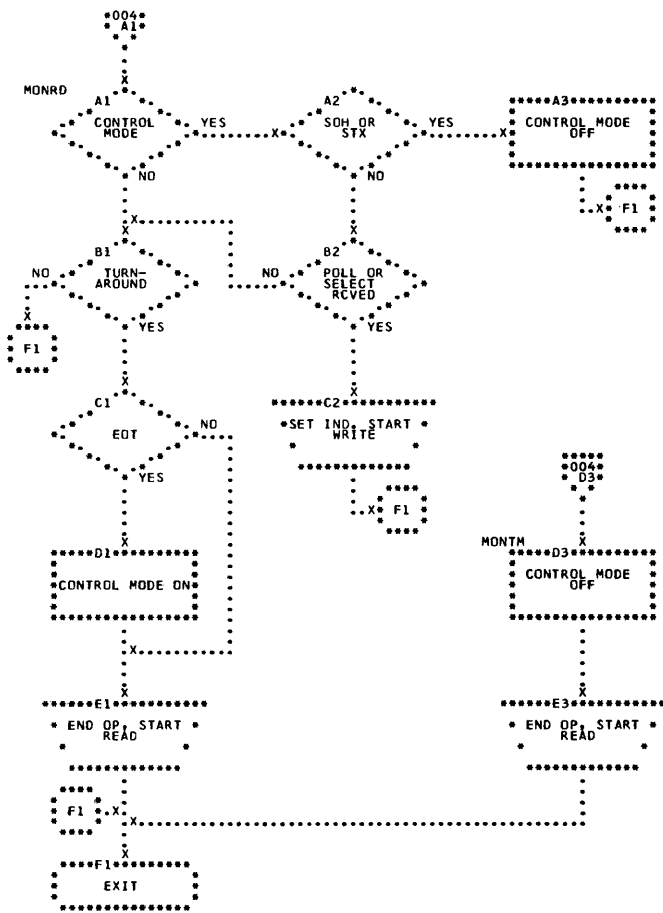
Flowchart SCA04. System Library, SCAT3 Call Processing



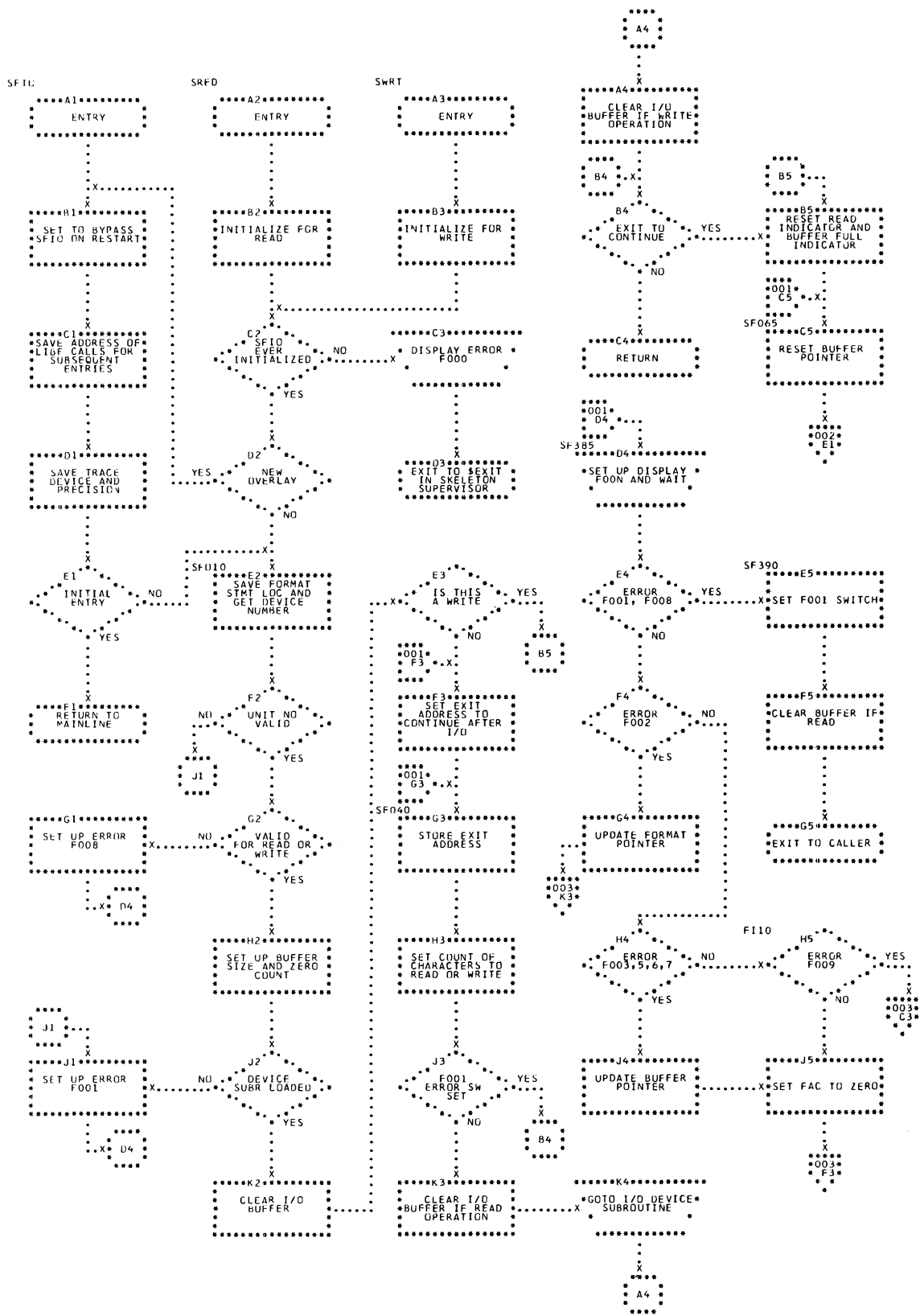
Flowchart SCA05. System Library, SCAT3 Interrupt Processing



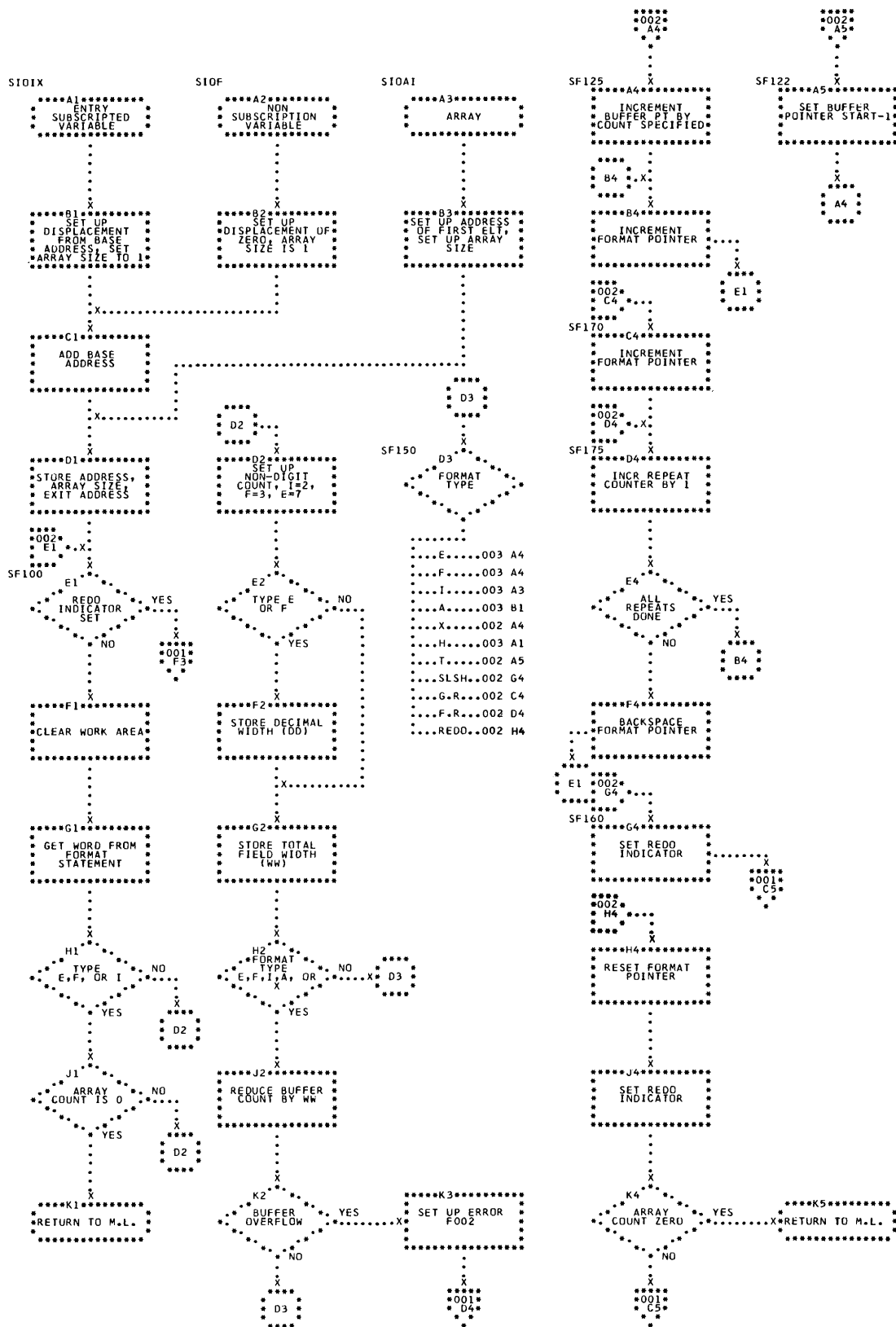
Flowchart SCA06. System Library, SCAT3 Interrupt Processing



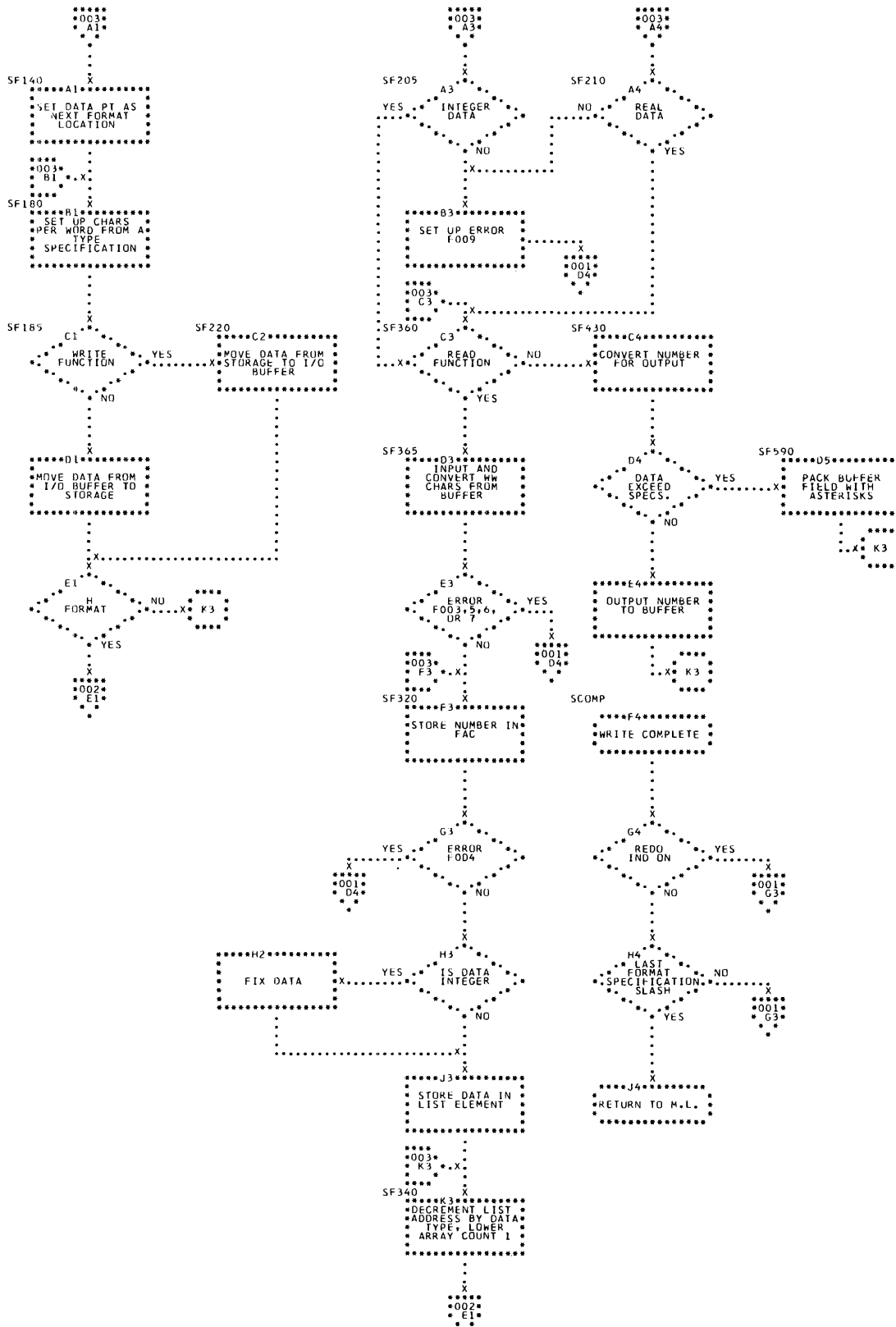
Flowchart SCA07. System Library, SCAT3 Interrupt Processing



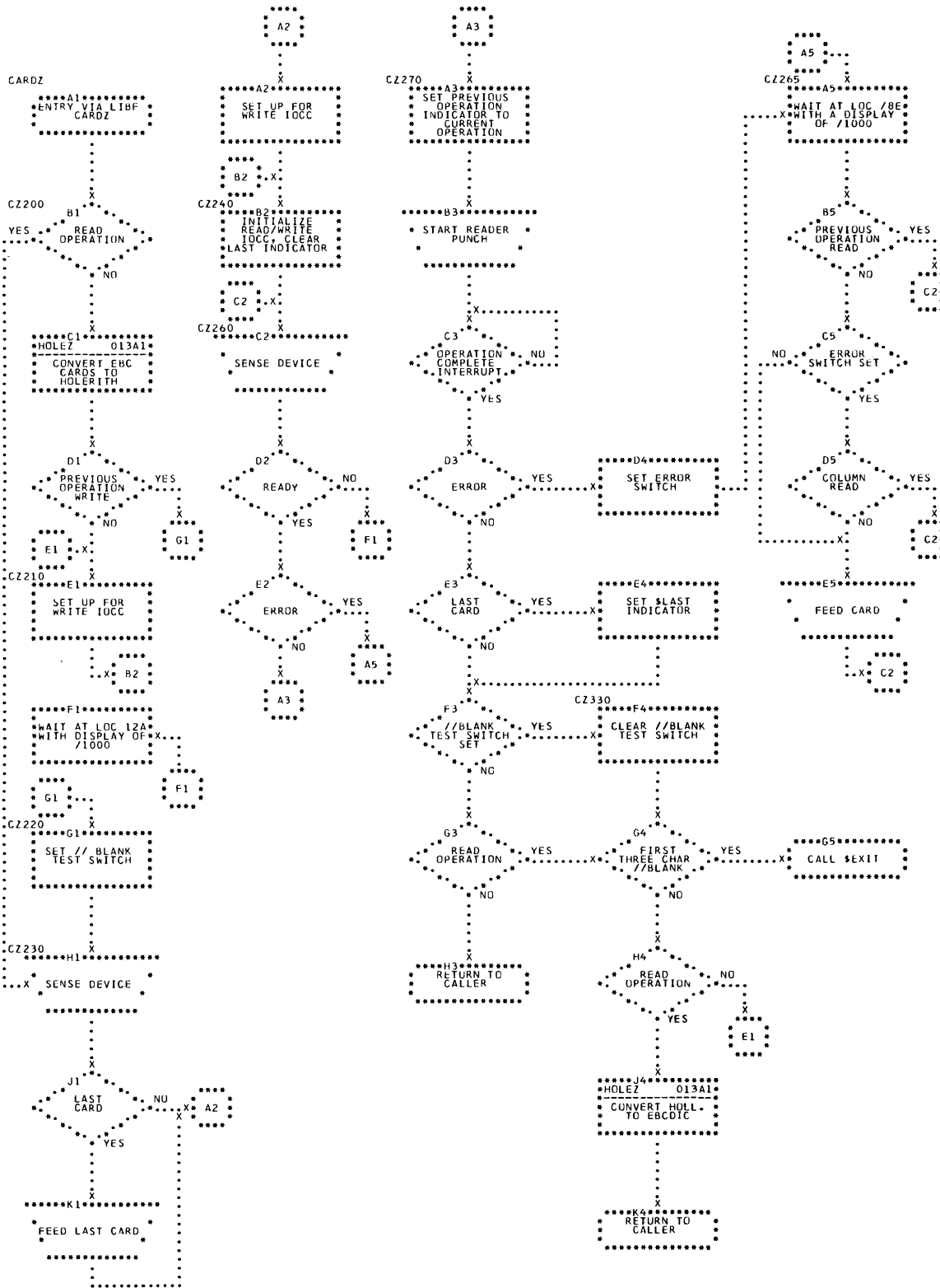
Flowchart FIO01. System Library, FORTRAN Non-disk I/O



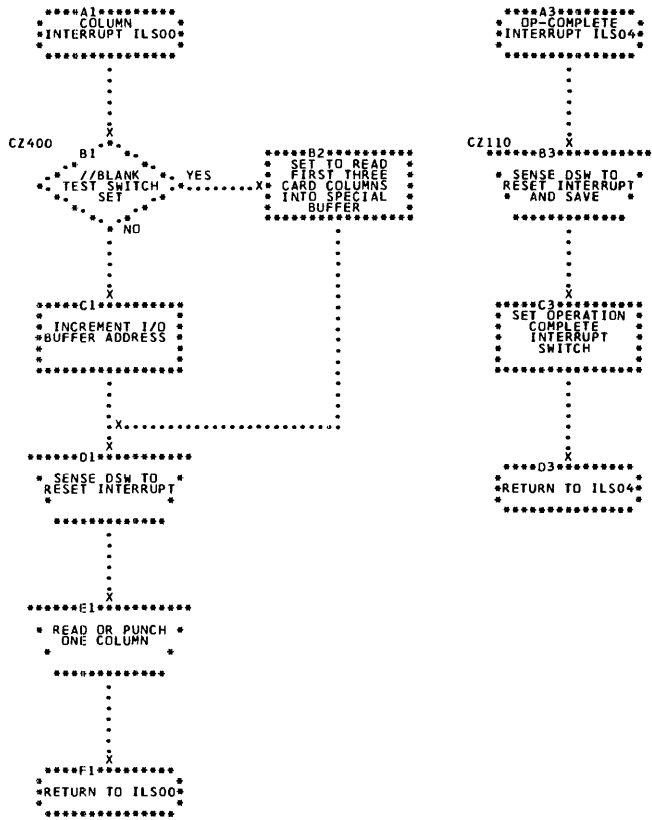
Flowchart FIO02. System Library, FORTRAN Non-disk I/O



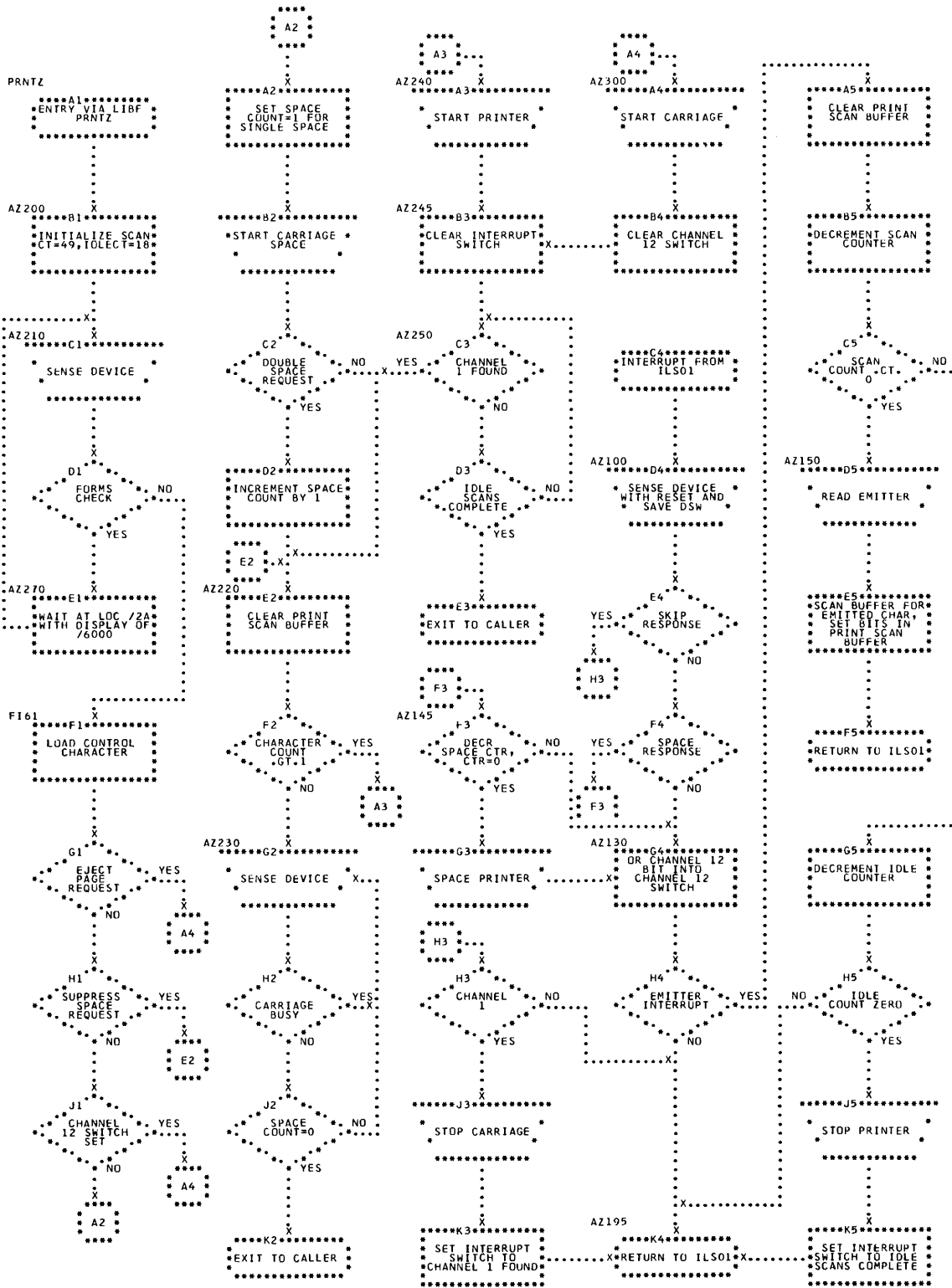
Flowchart FIO03. System Library, FORTRAN Non-disk I/O



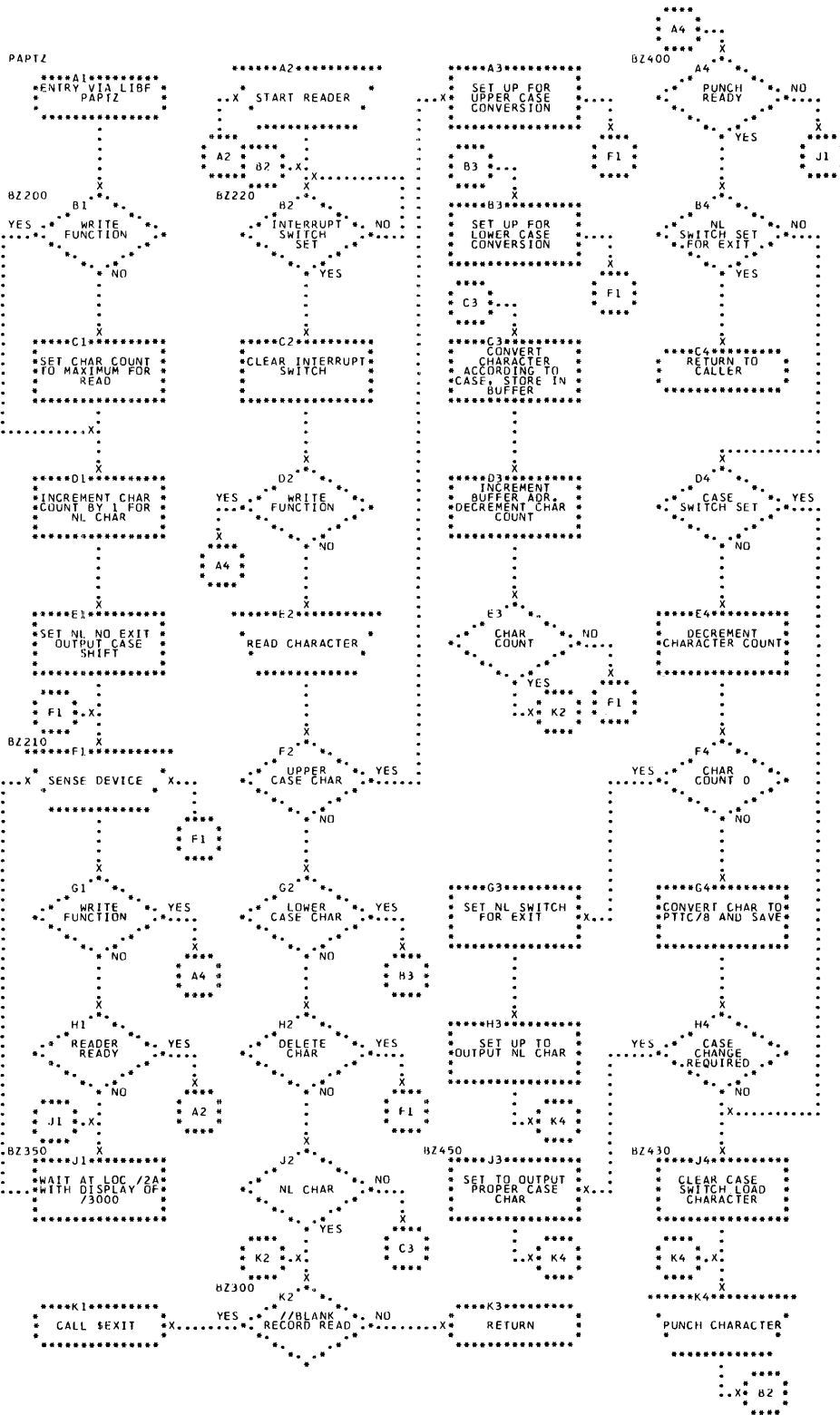
Flowchart FIO04. System Library, CARDZ



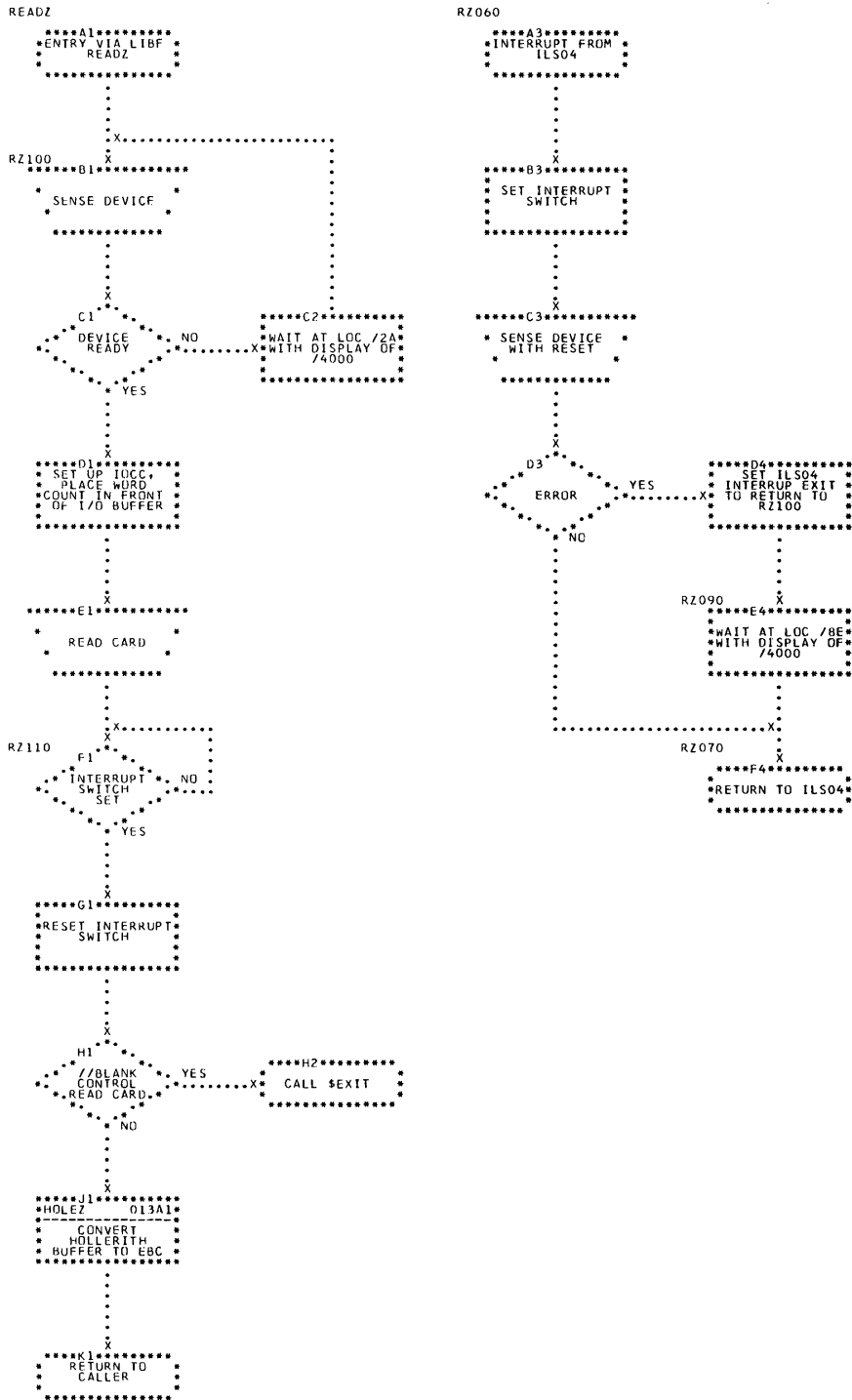
Flowchart FIO05. System Library, CARDZ



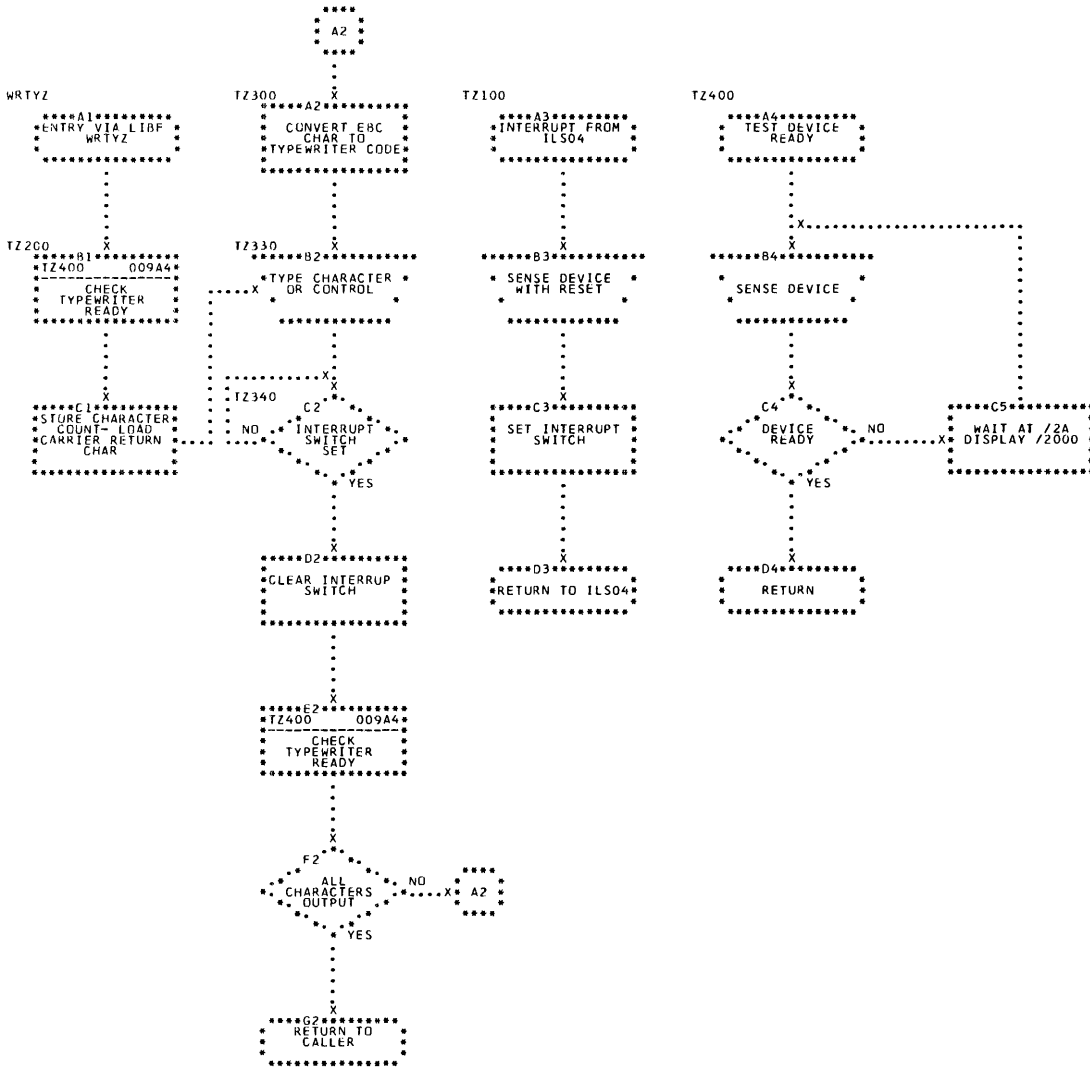
Flowchart FIO06. System Library, PRNTZ



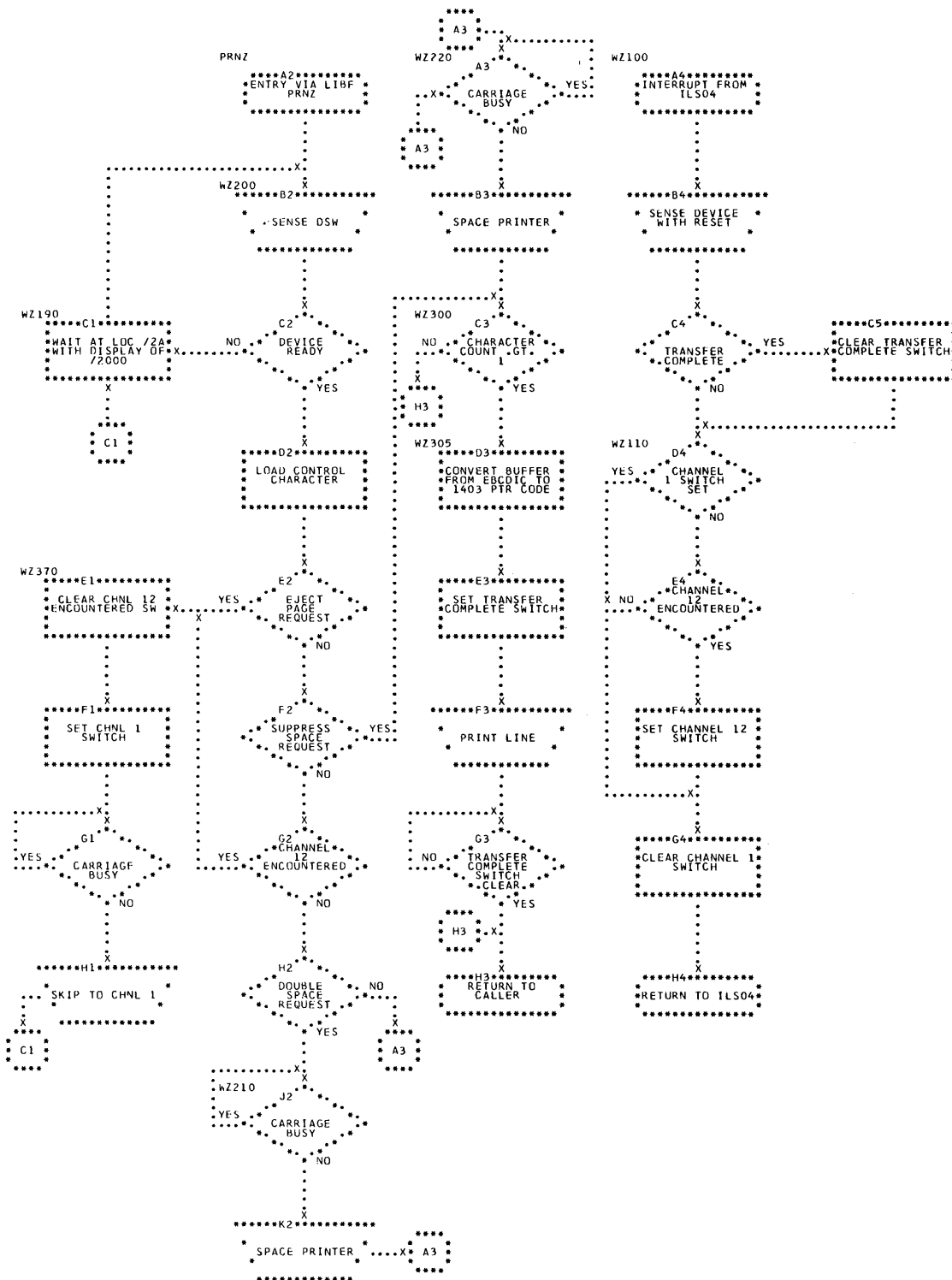
Flowchart FIO07. System Library, PAPTZ



Flowchart FIO08. System Library, READZ



Flowchart FIC09. System Library, WRXYZ



Flowchart FIO10. System Library, PRNZ

PZC60

*****A1*****
INTERRUPT FROM
ILS00

*****B1*****
SENSE DSW WITH
RESET

PZ200 *****C1*****
PUNCH A COLUMN

*****D1*****
INCREMENT TO
NEXT BUFFER
LOCATION

*****E1*****
RETURN TO ILS00

*****F1*****
FEED LAST CARD

PNCHZ

*****A2*****
ENTRY VIA LIBF
PNCHZ

PZ100 *****B2*****
HOLEZ 013A1
CONVERT BUFFER
EBCDIC TO HOLL

PZ120 *****C2*****
LOAD STARTING
BUFFER ADDRESS

*****D2*****
SET ROUTINE
BUSY IND CLEAR
LAST CARD
SWITCH

*****E2*****
SENSE DSW

*****F2*****
LAST CARD

*****G2*****
SENSE DSW

*****H2*****
DEVICE
READY

*****J2*****
START PUNCH
OPER.

*****K2*****
ROUTINE
BUSY IND
SET

PZ080

*****A4*****
INTERRUPT FROM
ILS04

*****B4*****
SENSE DSW WITH
RESET

*****C4*****
ERROR

PZ090 *****C5*****
SETUP ILS01
RETURN TO PZ120

*****D5*****
WAIT AT 8F
DISPLAY /1000

*****E4*****
LAST CARD

*****E5*****
SET LAST CARD
SWITCH

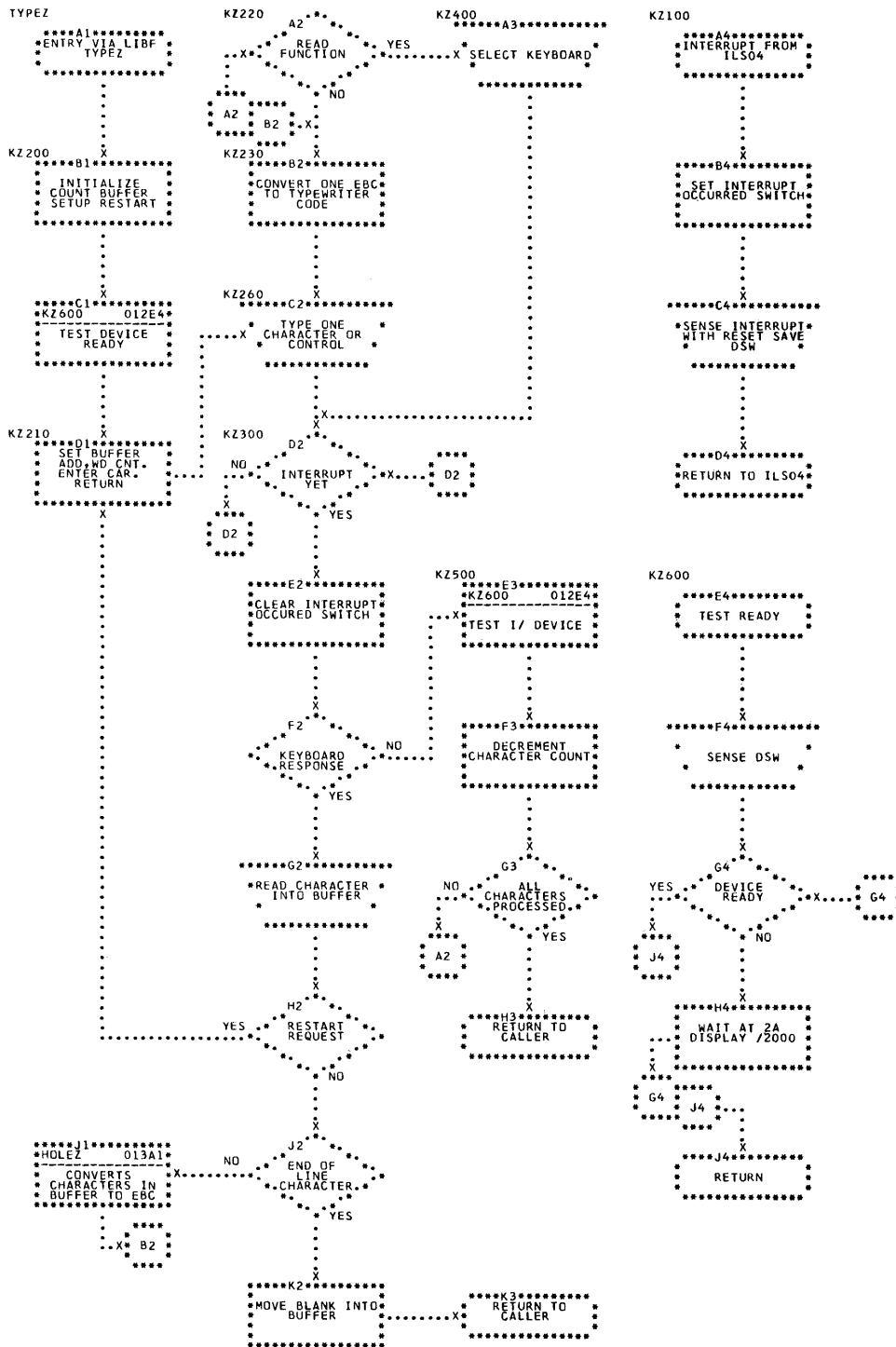
*****F4*****
FEED CHECK

*****C5*****

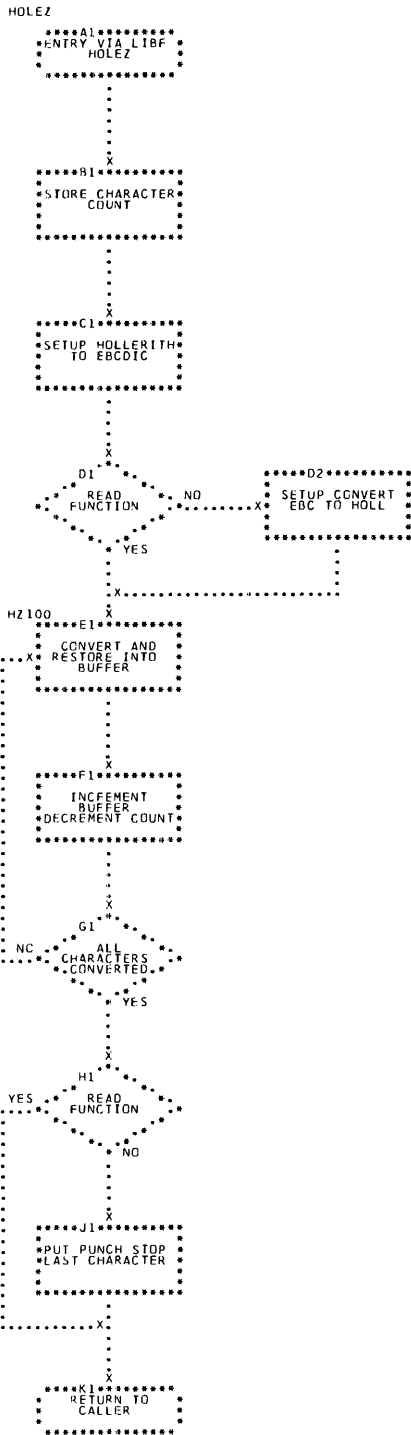
*****G4*****
CLEAR BUSY
INDICATOR

*****H4*****
RETURN TO ILS04

Flowchart FIO11. System Library, PNCHZ



Flowchart FIO12. System Library, TYPEZ



Flowchart FIO13. System Library, HOLEZ

APPENDIX A. EXAMPLES OF FORTRAN OBJECT CODING

This appendix shows, by example, the Assembler Language equivalent for the object coding generated

by the FORTRAN Compiler. A typical cross-section of FORTRAN statements is shown.

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace*</u>
<u>Arithmetic Statements - real, integer, and mixed modes</u>		
I=J	LD L J STO L I	. LIBF SIAR DC I
A=B	LIBF FLD DC B LIBF FSTO DC A	. . LIBF SFAR .
A=I	LD L I LIBF FLOAT LIBF FSTO DC A	. . LIBF SFAR .
I=A	LIBF FLD DC A LIBF IFIX STO L I	. . . LIBF SIAR DC I
I=K-M	LD L K S L M STO L I	. . LIBF SIAR DC I
A=I-B	LD L I LIBF FLOAT LIBF FSUB DC B LIBF FSTO DC A LIBF SFAR .

* The period in this column indicates that the generated coding is the same as in the Object Coding column. Trace refers to Arithmetic or Transfer Trace, whichever is applicable.

Source CodingObject CodingObject Coding
With Trace

A=B-I	LD	L	I	.		
	LIBF		FLOAT	.		
	LIBF		FSBR	.		
	DC		B	.		
	LIBF		FSTO	LIBF	SFAR	
	DC		A	.		
A=B+I-J or (A=I+B-J)	LD	L	I	.		
	LIBF		FLOAT	.		
	LIBF		FADD	.		
	DC		B	.		
	LIBF		FSTO	.		
	DC		GT1	.		
	LD	L	J	.		
	LIBF		FLOAT	.		
	LIBF		FSBR	.		
	DC		GT1	.		
	LIBF		FSTO	LIBF	SFAR	
	DC		A	.		
	I=J*K	LD	L	J	.	
		M	L	K	.	
SLT			16	.		
STQ		L	I	LIBF	SIAR	
				DC	I	
A=B*C	LIBF		FLD	.		
	DC		B	.		
	LIBF		FMPY	.		
	DC		C	.		
	LIBF		FSTO	LIBF	SFAR	
	DC		A	.		
A=B*I	LD	L	I	.		
	LIBF		FLOAT	.		
	LIBF		FMPY	.		
	DC		B	.		
	LIBF		FSTO	LIBF	SFAR	
	DC		A	.		

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
I=J / K	LD L J	.
	SRT 16	.
	D L K	.
	STO L I	LIBF SIAR DC I
A=B / C	LIBF FLD	.
	DC B	.
	LIBF FDIV	.
	DC C	.
	LIBF FSTO	LIBF SFAR
	DC A	.
I=J / (K+M)	LD L K	.
	A L M	.
	STO 3 +126	.
	LD L J	.
	SRT 16	.
	D 3 +126	.
	STO L I	LIBF SIAR DC I
I=A / J	LD L J	.
	LIBF FLOAT	.
	LIBF FDVR	.
	DC A	.
	LIBF IFIX	.
	STO L I	LIBF SIAR DC I
I=J**K	LD L J	.
	LIBF FIXI	.
	DC K	.
	STO L I	LIBF SIAR DC I

<u>Source Coding</u>	<u>Object Coding</u>		<u>Object Coding With Trace</u>	
A=B**I	LIBF	FLD	.	
	DC	B	.	
	LIBF	FAXI	.	
	DC	I	.	
	LIBF	FSTO	LIBF	SFAR
	DC	A	.	
A=B**C	LIBF	FLD	.	
	DC	B	.	
	CALL	FAXB	.	
	DC	C	.	
	LIBF	FSTO	LIBF	SFAR
	DC	A	.	
A=[**B	LD	L I	.	
	LIBF	FLOAT	.	
	CALL	FAXB	.	
	DC	B	.	
	LIBF	FSTO	LIBF	SFAR
	DC	A	.	
A=B**(I+J)	LD	L I	.	
	A	L J	.	
	STO	L GTI	.	
	LIBF	FLD	.	
	DC	B	.	
	LIBF	FAXI	.	
	DC	GTI	.	
	LIBF	FSTO	LIBF	SFAR
	DC	A	.	
A=B**(C+D)	LIBF	FLD	.	
	DC	C	.	
	LIBF	FADD	.	
	DC	D	.	
	LIBF	FSTO	.	
	DC	GTI	.	
	LIBF	FLD	.	
	DC	B	.	

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
	CALL FAXB	.
	DC GT1	.
	LIBF FSTO	LIBF SFAR
	DC A	.
A=B+C-(2**D*E-F)/G	LIBF FLD	.
	DC B	.
	LIBF FADD	.
	DC C	.
	LIBF FSTO	.
	DC GT1	.
	LD L ADR1	.
	LIBF FLOAT	.
	CALL FAXB	.
	DC D	.
	LIBF FMPY	.
	DC E	.
	LIBF FSUB	.
	DC F	.
	LIBF FDIV	.
	DC G	.
	LIBF FSBR	.
	DC GT1	.
	LIBF FSTO	LIBF SFAR
	DC A	.
	:	:
ADR1	DC 2	.

Arithmetic Statements - Subscripted Expressions

A(I)=B(I,J)+C(I,J)	LIBF SUBSC	.
	DC SGT1	.
	DC value D ₄ for array A	.
	DC I	.
	DC value D ₁ for array A (see Note 1)	.
	LIBF SUBSC	.
	DC SGT2	.
	DC value D ₄ for arrays B and C	.
	DC I	.

Source CodingObject CodingObject Coding
With Trace

	DC		value D_2 for arrays B and C	.		
	DC		J	.		
	DC		value D_1 for arrays B and C (see Note 1)	.		
	LIBF		FLDX	.		
	DC		B	.		
	LIBF		FADDX	.		
	DC		C	.		
	LDX	I1	SGT1	.		
	LIBF		FSTOX	.	LIBF	SFARX
	DC		A	.		
M=L(I, J, K)	LIBF		SUBSC	.		
	DC		SGT1	.		
	DC		value D_4 for array L	.		
	DC		I	.		
	DC		value D_3 for array L	.		
	DC		J	.		
	DC		value D_2 for array L	.		
	DC		K	.		
	DC		value D_1 for array L (see Note 1)	.		
	LD	L1	L	.		
	STO	L	M	.	LIBF	SIAR
				.	DC	M
M(I)=M(I+1)+M(J)	LIBF		SUBSC	.		
	DC		SGT1	.		
	DC		value D_4 for array M	.		
	DC		I	.		
	DC		value D_1 for array M (see Note 1)	.		
	LIBF		SUBSC	.		
	DC		SGT2	.		
	DC		value D_4 for array M	.		
	DC		I	.		
	DC		value D_1 for array M (see Note 1)	.		
	LIBF		SUBSC	.		
	DC		SGT3	.		
	DC		value D_4 for array M	.		
	DC		J	.		
	DC		value D_1 for array M (see Note 1)	.		

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
	LDX I1 SGT2	.
	LD L1 M	.
	LDX I1 SGT3	.
	A L1 M	.
	LDX I1 SGT1	.
	STO L1 M	LIBF SIARX DC M
M(1)=M(2)+M(3)	LDX L1 value D ₄ for literal subscript 2	.
	LD L1 M	.
	LDX L1 value D ₄ for literal subscript 3	.
	A L1 M	.
	LDX L1 value D ₄ for literal subscript 1	.
	STO L1 M	LIBF SIARX DC M
M(1)=N(1)+M(1)	LDX L1 value D ₄ for literal subscript 1	.
	LD L1 N	.
	A L1 M	.
	STO L1 M	LIBF SIARX DC M
<u>Statement Function Statements</u>		
A=JOE(B+C,D)+E	LIBF FLD	.
	DC B	.
	LIBF FADD	.
	DC C	.
	LIBF FSTO	.
	DC GT1	.
	CALL JOE	.
	DC GT1	.
	DC D	.
	LIBF FLOAT	.
	LIBF FADD	.
	DC E	.
	LIBF FSTO	LIBF SFAR
	DC A	.

<u>Source Coding</u>	<u>Object Coding</u>		<u>Object Coding With Trace</u>	
A=C(B, .5)+E	CALL	C	.	
	DC	B	.	
	DC	ADR1	.	
	LIBF	FADD	.	
	DC	E	.	
	LIBF	FSTO	LIBF	SFAR
	DC	A	.	
	:		:	
ADR1	DC	.5	.	
 <u>Call Statements</u>				
CALL XY	CALL	XY	.	
CALL YZ (A(2), A(I), B, C*D)	LIBF	SUBSC	.	
	DC	SGT1	.	
	DC	value D ₄ for array A	.	
	DC	I	.	
	DC	value D ₁ for array A (see Note 1)	.	
	LIBF	FLD	.	
	DC	C	.	
	LIBF	FMPY	.	
	DC	D	.	
	LIBF	FSTO	.	
	DC	GT1	.	
	LDX	L1 value D ₄ for literal subscript 2	.	
	MDX	L1 A	.	
	NOP		.	
	STX	L1 ADR1	.	
	LDX	I1 SGT1	.	
	MDX	L1 A	.	
	NOP		.	
	STX	L1 ADR2	.	
	CALL	YZ	.	
ADR1	DC	0	.	
ADR2	DC	0	.	
	DC	B	.	
	DC	GT1	.	

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
CALL YZ (A(I), B, C*D)	LIBF SUBSC	.
	DC SGT1	.
	DC value D ₄ for array A	.
	DC I	.
	DC value D ₁ for array A (see Note 1)	.
	LIBF FLD	.
	DC C	.
	LIBF FMPY	.
	DC D	.
	LIBF FSTO	.
	DC GT1	.
	LDX I1 SGT1	.
	MDX L1 A	.
	NOB	.
	STX L1 ADR1	.
	CALL YZ	.
ADR1	DC 0	.
	DC B	.
	DC GT1	.

DO and CONTINUE Statements

DO 10 I=J, K	LD L J	.
:	STO L I	.
:	ADR1 (next sequential instruction)	.
10 CONTINUE	:	:
	:	:
	:	:
	MDX L I, 1	.
	MDX *	.
	LD L I	.
	S L K	.
	BSC L ADR1, +	.
DO 10 I=J, K, M	LD L J	.
:	STO L I	.
:	ADR1 (next sequential instruction)	.
10 CONTINUE	:	:
	:	:
	:	:

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
	LD L I	.
	A L M	.
	STO L I	.
	S L K	.
	BSC L ADR1,+	.
 <u>GO TO Statement</u>		
GO TO 111	BSC L ADR1	.
	:	:
ADR1	(coding generated from statement 111)	.
 <u>Computed GO TO Statement</u>		
GO TO (111,112,113),I	LDX I1 I	.
LOC1	BSC I1 LOC1+1	LIBF SGOTO
	DC ADR1	.
	DC ADR2	.
	DC ADR3	.
	:	:
ADR1	(coding generated from statement 111)	.
	:	:
ADR2	(coding generated from statement 112)	.
	:	:
ADR3	(coding generated from statement 113)	.
 <u>IF Statements</u>		
IF (I) 111,112,113	LD L I	.
	BSC L ADR1,+Z	LIBF SIIF
	BSC L ADR2,+-	BSC L ADR1,+Z
	BSC L ADR3,-Z	BSC L ADR2,+-
	:	BSC L ADR3,-Z
	:	:
ADR1	(coding generated from statement 111)	.
	:	:
ADR2	(coding generated from statement 112)	.
	:	:
ADR3	(coding generated from statement 113)	.

<u>Source Coding</u>	<u>Object Coding</u>	<u>Object Coding With Trace</u>
IF (A) 111,100,113	LIBF FLD	.
100 CONTINUE	DC A	.
	LD 3 +126	LIBF SFIF
	BSC L ADR1,+Z	.
	BSC L ADR2,-Z	.
	:	:
ADR1	(coding generated from statement 111)	.
	:	:
ADR2	(coding generated from statement 113)	.
IF (A+I) 100,111,100	LD L I	.
100 CONTINUE	LIBF FLOAT	.
	LIBF FADD	.
	DC A	.
	LD 3 +126	LIBF SFIF
	BSC L ADR1,+-	.
	:	:
ADR1	(coding generated from statement 111)	.
IF (I) 111,111,112	LD L I	.
	BSC L ADR1,+	LIBF SIIF
	BSC L ADR2,-Z	BSC L ADR1,+
	:	BSC L ADR2,-Z
	:	:
ADR1	(coding generated from statement 111)	.
	:	:
ADR2	(coding generated from statement 112)	.
<u>PAUSE Statement</u>		
PAUSE 11	LIBF PAUSE	
	DC ADR1	
	:	
ADR1	DC 11	
<u>STOP Statement</u>		
STOP 21	LIBF STOP	
	DC ADR1	
	:	
ADR1	DC 21	

Source CodingObject CodingRETURN Statements

for (REAL) FUNCTION subprogram	LIBF	FLD	
where the subprogram name is COMP	DC	COMP	
	BSC	I	address of subprogram linkword in transfer vector
for (INTEGER) FUNCTION subprogram	LD	L	ICOMP
where the subprogram name is ICOMP	BSC	I	address of subprogram linkword in transfer vector
for SUBROUTINE subprogram	BSC	I	address of subprogram linkword in transfer vector

END Statement

The END statement produces no object coding.

I/O Initialization Calls

The following coding for the initialization of the FORTRAN I/O subroutines at execution time is generated and inserted into the program by the compiler.

LIBF	UFIO (see Note 2)
DC	/000X
LIBF	SDFIO (see Note 3)
DC	/000X
LIBF	SFIO (see Note 4)
DC	/00YX
DC	/0016
LIBF	TYPEZ or LIBF WRTYZ (see Note 5)
DC	0
LIBF	CARDZ (see Note 6)
DC	0
LIBF	PRNTZ
DC	0
LIBF	PAPTZ
DC	0
LIBF	PRNZ
DC	0
LIBF	TYPEZ (see Note 5)
DC	0
LIBF	WCHRI
DC	0
LIBF	READZ
DC	0
LIBF	PNCHZ (see Note 6)
DC	0

Source CodingObject CodingNon-Disk I/O Statements

READ (N, 101)A, I

```
LIBF      SRED
DC        N
DC        101
LIBF      SIOF
DC        A
LIBF      SIOI
DC        I
LIBF      SCOMP
```

READ (N, 101) X

where X is dimensioned

```
LIBF      SRED
DC        N
DC        101
LIBF      SIOAF
DC        X
DC        number of elements in array X
LIBF      SCOMP
```

READ (N, 101)(X(I), I=1, 5)

```
LIBF      SRED
DC        N
DC        101
LD        L  ADR1
STO       L  I
ADR3      LIBF  SUBSC
DC        SGT1
DC        value D4 for array X
DC        I
DC        value D1 for array X (see Note 1)
LIBF      SIOFX
DC        X
MDX       L  I, 1
LD        L  I
S         L  ADR2
BSC       L  ADR3, +
LIBF      SCOMP
:
ADR1      DC        1
ADR2      DC        5
```

Source CodingObject Coding

WRITE (N,101)A,I	LIBF	SWRT
	DC	N
	DC	101
	LIBF	SIOF
	DC	A
	LIBF	SIOI
	DC	I
	LIBF	SCOMP

Unformatted I/O Statements

READ (N) A,I	LIBF	URED
	DC	N
	LIBF	UIOF
	DC	A
	LIBF	UIOI
	DC	I
	LIBF	UCOMP

READ (N)	LIBF	URED
	DC	N
	LIBF	UCOMP
WRITE (N) A,I	LIBF	UWRT
	DC	N
	LIBF	UIOF
	DC	A
	LIBF	UIOI
	DC	I
	LIBF	UCOMP

Disk I/O Statements

READ (I*K) A,I	LIBF	SDRED
	DC	I
	DC	K
	LIBF	SDF
	DC	A
	LIBF	SDI
	DC	I
	LIBF	SDCOM

Source CodingObject Coding

WRITE (2*IBASE+IDISP (K)) A, B, X	LIBF	SUBSC
where A, B, and X are dimensioned	DC	SGT1
	DC	value D_4 for array IDISP
	DC	K
	DC	value D_1 for array IDISP (see Note 1)
	LD	L IBASE
	A	L1 IDISP
	STO	GT1
	LIBF	SDWRT
	DC	2
	DC	GT1
	LIBF	SDAF
	DC	A
	DC	number of elements in array A
	LIBF	SDAF
	DC	B
	DC	number of elements in array B
	LIBF	SDAF
	DC	X
	DC	number of elements in array X
FIND (J, K)	LIBF	SDCOM
	LIBF	SDFND
	DC	J
	DC	K

Manipulative I/O Statements

BACKSPACE I	LIBF	BCKSP
	DC	I
REWIND 10	LIBF	REWND
	DC	ADR1
	:	
ADR1	DC	10
END FILE 10	LIBF	EOF
	DC	ADR1
	:	
ADR1	DC	10

NOTES:

1. Tagged to indicate the end of the subscript argument list.
2. The LIBF UFIO and its parameter are inserted only if unformatted I/O statements are encountered in the program. X is the integer size and precision indicator.
3. The LIBF SDFIO and its parameter are inserted only if the Disk indicator (bit 8) in the IOCS word is on. X is the integer size and precision indicator.
4. The LIBF SFIO, its parameters, and the LIBF table are inserted only if indicators other than the Disk indicator in the IOCS word are on. Y is the trace device indicator. X is the integer size and precision indicator. The DC /0016 provides the number of words from the LIBF SFIO to the first word following the LIBF table, inclusive.

A LIBF to a FORTRAN I/O device subroutine is present in the LIBF table for each device indicated by a bit in the IOCS word. If a device is not indicated by a bit in the IOCS word, the LIBF is replaced by a DC 0.

5. If "KEYBOARD" is specified, the LIBF TYPEZ is inserted; if "TYPEWRITER" is specified, the LIBF WRTYZ is inserted.
6. If "CARD" is specified, the LIBF PNCHZ is not inserted; if "1442 PUNCH" is specified, the LIBF CARDZ is not inserted.

This appendix contains 1) assembly listings of the Resident Monitor, including DISKZ, and the Cold Start Program, 2) listings describing the contents of DCOM and the Resident Image, 3) the equivalences used throughout the monitor system programs, and

4) a cross-reference listing of all the symbols used in the above items.

Note that in the listings the character @ is printed as ' ', and the character # is printed as = .

DCOM		ST.NO.	LABEL	OPCD	FT	OPERANDS	I/O/SEQNO
ADDR	RFL	OBJECT					
0001	*		* RLTV ADDR*			* SYMBOL* DESCRIPTION	SYS00010
0002	*		*			*	SYS00020
0003	*	0-3	*			* RESERVED FOR EVEN BOUNDARIES	SYS00030
0004	*	4-5	*			* =NAME * NAME OF PROGRAM/CORE LOAD	SYS00040
0005	*	6	*			* =DBCT * BLOCK COUNT OF PROG/CORE LOAD	SYS00050
0006	*	7	*			* =FCNT * FILES SWITCH--ZERO MEANS NO	SYS00060
0007	*		*			* FILES HAVE BEEN EQUATED	SYS00070
0008	*	8	*			* =SYSC * SYS/NON-SYS CARTRIDGE INDR	SYS00080
0009	*	9	*			* =JBSW * JOB SWITCH-- NON-ZERO MEANS	SYS00090
0010	*		*			* TEMPORARY MODE	SYS00100
0011	*	10	*			* =CBSW * CLB-RETURN-TO-DUP SWITCH--	SYS00110
0012	*		*			* ZERO=CLB RETURN TO SUPV	SYS00120
0013	*	11	*			* =LCNT * NO. OF LOCALS	SYS00130
0014	*	12	*			* =MPSW * CORE MAP SWITCH--ZERO MEANS	SYS00140
0015	*		*			* DO NOT PRINT A CORE MAP	SYS00150
0016	*	13	*			* =MDF1 * NO. DUP CTRL RECDS (MODIF)	SYS00160
0017	*	14	*			* =MDF2 * ADDR OF MODIF BUFFER	SYS00170
0018	*	15	*			* =NCNT * NO. OF NUCALS	SYS00180
0019	*	16	*			* =ENTY * RLTV ENTRY ADDR OF PROGRAM	SYS00190
0020	*	17	*			* =RP67 * 1442-5 SW (0=1442-5 ON SYSTEM	SYS00200
0021	*	18	*			* =TODR * 'TO' WORKING STG DRIVE CODE	SYS00210
0022	*	19	*			* =FRDR * 'FROM' WORKING STG DRIVE CODE	SYS00220
0023	*	20	*			* =FHOL * ADDR OF LARGEST HOLE IN FXA	SYS00230
0024	*	21	*			* =FSZE * BLK CNT OF LARGEST HOLE IN FXA	SYS00240
0025	*	22	*			* =UHOL * ADDR OF LARGEST HOLE IN UA	SYS00250
0026	*	23	*			* =USZE * BLK CNT OF LARGEST HOLE IN UA	SYS00260
0027	*	24	*			* =DCSW * DUP CALL SW--NON-ZERO=DUP CALL	SYS00270
0028	*	25	*			* =PIOD * PRINCIPAL I/O DEVICE INDICATOR	SYS00280
0029	*	26	*			* =PPTR * PRINC. PRINT DEVICE INDICATOR	SYS00290
0030	*	27	*			* =CIAD * RLTV ADDR IN 'STRT OF CIL ADDR	SYS00300
0031	*	28-34	*			* RESERVED FOR FUTURE USE	SYS00310
0032	*	35	*			* =ANDU * 1+BLOCK ADDR OF END OF USER	SYS00320
0033	*		*			* AREA (ADJUSTED) LOGICAL DR 0	SYS00330
0034	*	36	*			* 1+BLOCK ADDR OF END OF USER	SYS00340
0035	*		*			* AREA (ADJUSTED) LOGICAL DR 1	SYS00350
0036	*	37	*			* 1+BLOCK ADDR OF END OF USER	SYS00360
0037	*		*			* AREA (ADJUSTED) LOGICAL DR 2	SYS00370
0038	*	38	*			* 1+BLOCK ADDR OF END OF USER	SYS00380
0039	*		*			* AREA (ADJUSTED) LOGICAL DR 3	SYS00390
0040	*	39	*			* 1+BLOCK ADDR OF END OF USER	SYS00400
0041	*		*			* AREA (ADJUSTED) LOGICAL DR 4	SYS00410
0042	*	40	*			* =HNDU * 1+BLOCK ADDR OF END OF USER	SYS00420
0043	*		*			* AREA (BASE) LOGICAL DRIVE 0	SYS00430
0044	*	41	*			* 1+BLOCK ADDR OF END OF USER	SYS00440
0045	*		*			* AREA (BASE) LOGICAL DRIVE 1	SYS00450
0046	*	42	*			* 1+BLOCK ADDR OF END OF USER	SYS00460
0047	*		*			* AREA (BASE) LOGICAL DRIVE 2	SYS00470
0048	*	43	*			* 1+BLOCK ADDR OF END OF USER	SYS00480
0049	*		*			* AREA (BASE) LOGICAL DRIVE 3	SYS00490
0050	*	44	*			* 1+BLOCK ADDR OF END OF USER	SYS00500
0051	*		*			* AREA (BASE) LOGICAL DRIVE 4	SYS00510
0052	*	45	*			* =FPAD * FILE PROTECT ADDR, LOGICAL	SYS00520
0053	*		*			* DRIVE 0 (BASE)	SYS00530
0054	*	46	*			* FILE PROTECT ADDR, LOGICAL	SYS00540
0055	*		*			* DRIVE 1 (BASE)	SYS00550
0056	*	47	*			* FILE PROTECT ADDR, LOGICAL	SYS00560
0057	*		*			* DRIVE 2 (BASE)	SYS00570
0058	*	48	*			* FILE PROTECT ADDR, LOGICAL	SYS00580
0059	*		*			* DRIVE 3 (BASE)	SYS00590
0060	*	49	*			* FILE PROTECT ADDR, LOGICAL	SYS00600
0061	*		*			* DRIVE 4 (BASE)	SYS00610
0062	*	50	*			* =PCID * CARTRIDGE ID, PHYSICAL DRIVE 0	SYS00620
0063	*	51	*			* CARTRIDGE ID, PHYSICAL DRIVE 1	SYS00630
0064	*	52	*			* CARTRIDGE ID, PHYSICAL DRIVE 2	SYS00640
0065	*	53	*			* CARTRIDGE ID, PHYSICAL DRIVE 3	SYS00650
0066	*	54	*			* CARTRIDGE ID, PHYSICAL DRIVE 4	SYS00660
0067	*	55	*			* =CIDN * CARTRIDGE ID, LOGICAL DRIVE 0	SYS00670
0068	*	56	*			* CARTRIDGE ID, LOGICAL DRIVE 1	SYS00680
0069	*	57	*			* CARTRIDGE ID, LOGICAL DRIVE 2	SYS00690
0070	*	58	*			* CARTRIDGE ID, LOGICAL DRIVE 3	SYS00700
0071	*	59	*			* CARTRIDGE ID, LOGICAL DRIVE 4	SYS00710
0072	*	60	*			* =CIBA * SCTR ADDR OF CIB, LOGICAL DR 0	SYS00720
0073	*	61	*			* SCTR ADDR OF CIB, LOGICAL DR 1	SYS00730
0074	*	62	*			* SCTR ADDR OF CIB, LOGICAL DR 2	SYS00740
0075	*	63	*			* SCTR ADDR OF CIB, LOGICAL DR 3	SYS00750
0076	*	64	*			* SCTR ADDR OF CIB, LOGICAL DR 4	SYS00760
0077	*	65	*			* =SCRA * SCRA, LOGICAL DRIVE 0	SYS00770
0078	*	66	*			* SCRA, LOGICAL DRIVE 1	SYS00780
0079	*	67	*			* SCRA, LOGICAL DRIVE 2	SYS00790
0080	*	68	*			* SCRA, LOGICAL DRIVE 3	SYS00800
0081	*	69	*			* SCRA, LOGICAL DRIVE 4	SYS00810
0082	*	70	*			* =FMAT * FORMAT OF PROG IN WS, DRIVE 0	SYS00820
0083	*	71	*			* FORMAT OF PROG IN WS, DRIVE 1	SYS00830
0084	*	72	*			* FORMAT OF PROG IN WS, DRIVE 2	SYS00840

DCOM ADDR REL OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
	0085	* 73	*		* FORMAT OF PROG IN WS, DRIVE 3	SYS00850
	0086	* 74	*		* FORMAT OF PROG IN WS, DRIVE 4	SYS00860
	0087	* 75	*	=FLET	* FLET SCTR ADDR, LOGICAL DR 0	SYS00870
	0088	* 76	*		* FLET SCTR ADDR, LOGICAL DR 1	SYS00880
	0089	* 77	*		* FLET SCTR ADDR, LOGICAL DR 2	SYS00890
	0090	* 78	*		* FLET SCTR ADDR, LOGICAL DR 3	SYS00900
	0091	* 79	*		* FLET SCTR ADDR, LOGICAL DR 4	SYS00910
	0092	* 80	*	=ULET	* LET SCTR ADDR, LOGICAL DR 0	SYS00920
	0093	* 81	*		* LET SCTR ADDR, LOGICAL DR 1	SYS00930
	0094	* 82	*		* LET SCTR ADDR, LOGICAL DR 2	SYS00940
	0095	* 83	*		* LET SCTR ADDR, LOGICAL DR 3	SYS00950
	0096	* 84	*		* LET SCTR ADDR, LOGICAL DR 4	SYS00960
	0097	* 85	*	=WSCT	* BLK CNT OF PROG IN WS, DRIVE 0	SYS00970
	0098	* 86	*		* BLK CNT OF PROG IN WS, DRIVE 1	SYS00980
	0099	* 87	*		* BLK CNT OF PROG IN WS, DRIVE 2	SYS00990
	0100	* 88	*		* BLK CNT OF PROG IN WS, DRIVE 3	SYS01000
	0101	* 89	*		* BLK CNT OF PROG IN WS, DRIVE 4	SYS01010
	0102	* 90	*	=CSHN	* SCTR CNT CUSHION, LOGICAL DR 0	SYS01020
	0103	* 91	*		* SCTR CNT CUSHION, LOGICAL DR 1	SYS01030
	0104	* 92	*		* SCTR CNT CUSHION, LOGICAL DR 2	SYS01040
	0105	* 93	*		* SCTR CNT CUSHION, LOGICAL DR 3	SYS01050
	0106	* 94	*		* SCTR CNT CUSHION, LOGICAL DR 4	SYS01060
	0107	* 95-319	*		* RESERVED FOR FUTURE USE	SYS01070

RESIDENT IMAGE

ADDR REL OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
	0109	*				SYS01090
	0110	* RLV ADDR	*	SYMBOL	* DESCRIPTION	SYS01100
	0111	*	*	*		SYS01110
	0112	* 0-209	*		* INITIAL VALUES FOR WORDS 4-213	SYS01120
	0113	*	*		* OF THE RESIDENT MONITOR	SYS01130
	0114	* 210-319	*		* RESERVED FOR FUTURE USE	SYS01140

RESIDENT MONITOR

ADDR REL OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
	0116	*****				SYS01160
	0117	*				SYS01170
	0118	*STATUS-VERSION 2, MODIFICATION 0				SYS01180
	0119	*				SYS01190
	0120	*FUNCTION/OPERATION-				SYS01200
	0121	* THIS SECTION ALWAYS REMAINS IN CORE. IT				SYS01210
	0122	* IS COMPRISED OF THE COMMUNICATIONS				SYS01220
	0123	* AREA (COMMA), THE SKELETON SUPERVISOR, AND				SYS01230
	0124	* A DISK I/O SUBROUTINE, NOMINALLY DISKZ. (THE				SYS01240
	0125	* FIRST TWO OF THESE SECTIONS ARE INTERMIXED.)				SYS01250
	0126	* COMMA CONTAINS THE SYSTEM PARAMETERS REQUIR-				SYS01260
	0127	* ED TO FETCH A CORE LOAD IN CORE IMAGE FOR-				SYS01270
	0128	* MAT. THE SKELETON SUPERVISOR PROVIDES IN-				SYS01280
	0129	* STRUCTIONS FOR INITIATING A CALL EXIT, A				SYS01290
	0130	* CALL LINK, A DUMP-TO-PRINTER OR A CALL TO THE				SYS01300
	0131	* AUXILIARY SUPERVISOR. IN ADDITION, THE SKELE-				SYS01310
	0132	* TON SUPERVISOR CONTAINS SEVERAL TRAPS FOR CER-				SYS01320
	0133	* TAIN I/O FUNCTIONS/CONDITIONS. THE DISK I/O				SYS01330
	0134	* SECTION CONSISTS OF A SUBROUTINE FOR READING				SYS01340
	0135	* FROM OR WRITING ON A DISK CARTRIDGE ON A				SYS01350
	0136	* GIVEN LOGICAL DISK DRIVE.				SYS01360
	0137	*				SYS01370
	0138	*ENTRY POINTS-				SYS01380
	0139	* * \$PRET-A TRAP FOR PREOPERATIVE I/O ERRORS.				SYS01390
	0140	* THE CALLING SEQUENCE IS				SYS01400
	0141	* BSI L \$PRET				SYS01410
	0142	* * \$PSTX-A POSTOPERATIVE ERROR TRAP FOR I/O				SYS01420
	0143	* DEVICES ON LEVEL X (X=1,2,3,OR 4).				SYS01430
	0144	* THE CALLING SEQUENCE IS				SYS01440
	0145	* BSI L \$PSTX				SYS01450
	0146	* * \$STOP-THE PROGRAM STOP KEY TRAP.				SYS01460
	0147	* * \$EXIT-THE ENTRY POINT FOR THE EXIT/CALL				SYS01470
	0148	* EXIT STATEMENT. THE CALLING SEQUENCE IS				SYS01480
	0149	* LDX 0 \$EXIT				SYS01490
	0150	* * \$LINK-THE ENTRY POINT FOR THE LINK/CALL				SYS01500
	0151	* LINK STATEMENT. THE CALLING SEQUENCE IS				SYS01510
	0152	* BSI L \$LINK				SYS01520
	0153	* * \$DUMP-THE ENTRY POINT FOR THE DUMP/PDMP				SYS01530
	0154	* STATEMENT. THE CALLING SEQUENCE IS				SYS01540
	0155	* BSI L \$DUMP				SYS01550

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
0156	*			DC			FORMAT	* SYS01560
0157	*			DC			LIMIT1	* SYS01570
0158	*			DC			LIMIT2	* SYS01580
0159	*			WHERE LIMIT1 AND LIMIT2 ARE THE LIMITS				* SYS01590
0160	*			BETWEEN WHICH THE DUMP IS TO OCCUR, AND				* SYS01600
0161	*			FORMAT IS A CODE INDICATING THE FORMAT				* SYS01610
0162	*			OF THE DUMP. IF FORMAT IS NEGATIVE,				* SYS01620
0163	*			THE AUXILIARY SUPERVISOR IS FETCHED				* SYS01630
0164	*			AND CONTROL PASSED TO IT.				* SYS01640
0165	*			* DZ000-ENTERED WHEN THE CALLER WISHES TO				* SYS01650
0166	*			PERFORM A DISK I/O OPERATION. THE				* SYS01660
0167	*			CALLING SEQUENCE VARIES WITH THE				* SYS01670
0168	*			VERSION OF THE DISK I/O SUBROUTINE.				* SYS01680
0169	*			* \$I200/\$I400-ENTERED WHEN THE OPERATION-				* SYS01690
0170	*			COMPLETE INTERRUPT OCCURS ON				* SYS01700
0171	*			LEVEL 2/4.				* SYS01710
0172	*							* SYS01720
0173	*			*INPUT-N/A				* SYS01730
0174	*							* SYS01740
0175	*			*OUTPUT-WORDS 6-4090 SAVED ON THE CIB ON A CALL				* SYS01750
0176	*			DUMP				* SYS01760
0177	*							* SYS01770
0178	*			*EXTERNAL REFERENCES-N/A				* SYS01780
0179	*							* SYS01790
0180	*			*EXITS-				* SYS01800
0181	*			* * NORMAL				* SYS01810
0182	*			*THE EXITS FROM THE SUBROUTINES AT \$PRET				* SYS01820
0183	*			*\$PST1, \$PST2, \$PST3, \$PST4, AND \$STOP				* SYS01830
0184	*			ARE BRANCH INSTRUCTIONS FOLLOWING A				* SYS01840
0185	*			WAIT INSTRUCTION. \$STOP TURNS OFF IN-				* SYS01850
0186	*			TERRUPT LEVEL 5 AFTER THE START KEY IS				* SYS01860
0187	*			DEPRESSED.				* SYS01870
0188	*			*THE EXITS FROM \$EXIT,\$LINK,AND \$DUMP ARE				* SYS01880
0189	*			TO THE CORE IMAGE LOADER, PHASE 1,				* SYS01890
0190	*			AFTER THAT PHASE HAS BEEN FETCHED.				* SYS01900
0191	*			*THE EXIT FROM DZ000 IS BACK TO THE				* SYS01910
0192	*			CALLER AFTER THE REQUESTED DISK OPERA-				* SYS01920
0193	*			TION HAS BEEN INITIATED.				* SYS01930
0194	*			*THE EXITS FROM \$I200/\$I400 ARE BACK TO				* SYS01940
0195	*			THE ADDRESSES FROM WHICH THE DISK OP-				* SYS01950
0196	*			ERATION COMPLETE INTERRUPT OCCURED				* SYS01960
0197	*			AFTER THE INTERRUPT HAS BEEN SERVICED				* SYS01970
0198	*			BY THE APPROPRIATE ISS.				* SYS01980
0199	*			* ERROR-N/A				* SYS01990
0200	*							* SYS02000
0201	*			*TABLES/WORK AREAS-				* SYS02010
0202	*			* * \$ACDE				* SYS02020
0203	*			* * \$CH12				* SYS02030
0204	*			* * \$CILA				* SYS02040
0205	*			* * \$CLSW				* SYS02050
0206	*			* * \$COMN				* SYS02060
0207	*			* * \$CORE				* SYS02070
0208	*			* * \$CTSW				* SYS02080
0209	*			* * \$CXR1				* SYS02090
0210	*			* * \$CYLN				* SYS02100
0211	*			* * \$DABL				* SYS02110
0212	*			* * \$DADR				* SYS02120
0213	*			* * \$DBSY				* SYS02130
0214	*			* * \$DCYL				* SYS02140
0215	*			* * \$DMPF				* SYS02150
0216	*			* * \$DREQ				* SYS02160
0217	*			* * \$FPAD				* SYS02170
0218	*			* * \$HASH				* SYS02180
0219	*			* * \$IBT2				* SYS02190
0220	*			* * \$IBT4				* SYS02200
0221	*			* * \$IBSY				* SYS02210
0222	*			* * \$IOCT				* SYS02220
0223	*			* * \$KCSW				* SYS02230
0224	*			* * \$LAST				* SYS02240
0225	*			* * \$NDUP				* SYS02250
0226	*			* * \$NXEQ				* SYS02260
0227	*			* * \$PBSY				* SYS02270
0228	*			* * \$PGCT				* SYS02280
0229	*			* * \$PHSE				* SYS02290
0230	*			* * \$RMSW				* SYS02300
0231	*			* * \$SNLT				* SYS02310
0232	*			* * \$UFIO				* SYS02320
0233	*			* * \$ULET				* SYS02330
0234	*			* * \$WRD1				* SYS02340
0235	*			* * \$WSDR				* SYS02350
0236	*							* SYS02360
0237	*			*ATTRIBUTES-REUSABLE				* SYS02370
0238	*							* SYS02380
0239	*			*NOTES-				* SYS02390
0240	*			* THERE ARE WAIT INSTRUCTIONS AT \$PRET+1,				* SYS02400
0241	*			*\$STOP+1, AND \$PSTX+1. DEPRESSING THE START				* SYS02410
0242	*			* KEY WILL RETURN CONTROL TO THE CALLER IN ALL				* SYS02420
0243	*			* CASES.				* SYS02430
0244	*			*****				* SYS02440

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0246	*			PROVIDE PARAMETERS FOR SYSTEM LOADER	SYS02460
			0247	*				SYS02470
			0248				ABS	SYS02480
0280			0249				ORG 4	SYS02490
0004 0	OFFA		0250				DC 4095-* WD CNT FOR WRITING CORE ON CIB	SYS02500
0005 0	0000		0251	\$CIBA	DC		*-* SCTR ADDR OF THE CIB	SYS02510
0006 0	0000		0252	\$CH12	DC		*-* ADDR OF CHANNEL 12 INDICATOR	SYS02520
0007 0	0000		0253	\$CGMN	DC		*-* LENGTH OF COMMON (IN WORDS)	SYS02530
			0254	*				SYS02540
			0255	*			ULTIMATE RESIDENCE OF THE INTERRUPT TV	SYS02550
			0256	*				SYS02560
0008 0	0000		0257	\$LEVO	DC		*-* LEVEL 0 BRANCH ADDRESS	SYS02570
0009 0	0000		0258	\$LEV1	DC		*-* LEVEL 1 BRANCH ADDRESS	SYS02580
000A 0	00B3		0259	\$LEV2	DC	\$I200	LEVEL 2 BRANCH ADDR	SYS02590
000B 0	0000		0260	\$LEV3	DC		*-* LEVEL 3 BRANCH ADDRESS	SYS02600
000C 0	00C4		0261	\$LEV4	DC	\$I400	LEVEL 4 BRANCH ADDR	SYS02610
000D 0	0091		0262	\$LEV5	DC	\$STOP	LEVEL 5 BRANCH ADDR	SYS02620
			0263	*				SYS02630
			0264	*				SYS02640
000E 0	0000		0265	\$CORE	DC		*-* SIZE OF CORE, E.G., 4096=4K	SYS02650
000F 0	0000		0266	\$CTSW	DC		*-* CONTROL RECORD TRAP SWITCH	SYS02660
0010 0	0000		0267	\$DADR	DC		*-* SCTR ADDR OF PROG TO BE LOADED	SYS02670
0011 0	5540		0268	\$DABL	DC	/5540	IOCC FOR SCA RESET (ODD ADDR)	SYS02680
0012 0	0000		0269	\$DREQ	DC		*-* IND. FOR REQUESTED VERSION DKI/O	SYS02690
0013 0	0000		0270	\$IBSY	DC		*-* NON-ZERO IF CD/PAP TP DEV. BUSY	SYS02700
0014	000C		0271	\$HASH	BSS	E 12	WORK AREA	SYS02710
			0272	*				SYS02720
			0273	*				SYS02730
0020	0008		0274	\$SCAN	BSS	8 1132	SCAN AREA	32 SYS02740
			0275	*				SYS02750
			0276	*				SYS02760
			0277	*				SYS02770
			0278	*			TRAP FOR PREOPERATIVE I/O ERRORS	SYS02780
			0279	*				SYS02790
0028 0	0000		0280	\$PRET	DC		*-* ENTRY POINT	SYS02800
0029 0	3000		0281		WAIT		WAIT TIL START KEY PUSHED	SYS02810
002A 00	4C800028		0282		BSC I	\$PRET	RETURN TO CALLER	SYS02820
			0283	*				SYS02830
			0284	*				SYS02840
002C 0	0000		0285	\$IREQ	DC		*-* ADDR OF INT REQUEST SUBROUTINE	SYS02850
002D 0	0000		0286	\$ULET	DC		*-* ADDR OF LET, LOGICAL DR 0	SYS02860
002E 0	0000		0287		DC		*-* ADDR OF LET, LOGICAL DR 1	SYS02870
002F 0	0000		0288		DC		*-* ADDR OF LET, LOGICAL DR 2	SYS02880
0030 0	0000		0289		DC		*-* ADDR OF LET, LOGICAL DR 3	SYS02890
0031 0	0000		0290		DC		*-* ADDR OF LET, LOGICAL DR 4	SYS02900
0032 0	0000		0291	\$IGCT	DC		*-* ZERO IF NO I/O IN PROGRESS	50 SYS02910
0033 0	0000		0292	\$LAST	DC		*-* NON-ZERO WHEN LAST CARD SENSED	SYS02920
0034 0	0000		0293	\$NDUP	DC		*-* DO NOT DUP IF NON-ZERO	SYS02930
0035 0	0000		0294	\$NXEQ	DC		*-* DO NOT EXECUTE IF NON-ZERO	SYS02940
0036 0	0000		0295	\$PBSY	DC		*-* NON-ZERO WHEN PRINTER BUSY	SYS02950
0037 0	0000		0296	\$PGCT	DC		*-* PAGE NO. FOR HEADINGS	SYS02960
			0297	*				SYS02970
			0298	*			CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR	SYS02980
			0299	*				SYS02990
0038 0	7014		0300	\$EXIT	MDX	\$S000	BR TO FETCH CIL, PHASE 1	56 SYS03000
			0301	*				SYS03010
			0302	***			CALL LINK ENTRY POINT	SYS03020
			0303	*				SYS03030
0039 0	0000		0304	\$LINK	DC		*-* ENTRY POINT	57 SYS03040
003A 0	1810		0305		SRA	16		SYS03050
003B 0	7012		0306		MDX	\$S100	BR TO FETCH CIL, PHASE 1	SYS03060
003C	0000		0307		BSS E	0		SYS03070
003C 0	0001		0308	\$S900	DC	1	DISK PARAMETERS FOR SAVING CORE	SYS03080
003D 0	0004		0309		DC	\$CIBA-1	*IN CONNECTION WITH DUMP	SYS03090
003E 0	FFFF		0310	\$S910	DC	-1	CALL EXIT INDICATOR	SYS03100
			0311	*				SYS03110
			0312	***			SAVE 1ST 4K OF CORE ON THE CIB	SYS03120
			0313	*				SYS03130
003F 0	0000		0314	\$DUMP	DC		*-* ENTRY POINT	63 SYS03140
0040 00	74C00032		0315		MDX L	\$IOCT,0	SKIP IF NO INTERRUPT	SYS03150
0042 0	70F0		0316		MDX	*-3	*PENDING, BR OTHERWISE	SYS03160
0043 0	D8D6		0317		STD	\$ACEX	SAVE ACCUMULATOR, EXTENSION	SYS03170
0044 0	69D2		0318		STX	1 \$CXRI	SAVE XRI	SYS03180
0045 00	C480003F		0319		LD	I \$DUMP		SYS03190
0047 0	D0D1		0320		STO	\$DMPF	SAVE DUMP FORMAT CODE	SYS03200
0048 0	C8F3		0321		LDD	\$S900		SYS03210
0049 00	440000F2		0322		BSI L	DZ000	SAVE WDS 6-4095 ON CIB	SYS03220
004B 0	C0F0		0323		LD	\$S900		SYS03230
004C 0	7001		0324		MDX	\$S100	BR TO FETCH CIL, PHASE 1	SYS03240
			0326	*				SYS03260
			0327	***			FETCH CORE IMAGE LOADER, PHASE 1	SYS03270
			0328	*				SYS03280
004D 0	C0F0		0329	\$S000	LD	\$S910		SYS03290
004E 0	DOC7		0330	\$S100	STD	\$RMSW	SAVE EXIT-LINK-DUMP SWITCH	SYS03300

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO	
004F	00	74000032	0331	MDX	L		\$IOCT,0 SKIP IF NO INTERRUPT LEFT	SYS03310	
0051	0	70FD	0332	MDX		*-3	BR IF INTERRUPT(S) PENDING	SYS03320	
0052	0	08BD	0333	XIO		\$DABL-1	TURN OFF THE SCA	SYS03330	
0053	00	65800039	0334	LDX	I1	\$LINK	LINK ADDR TO XR1	SYS03340	
0055	0	C101	0335	LD	I	1	FETCH 2ND WD OF LINK NAME	SYS03350	
0056	0	18D0	0336	RTE		16		SYS03360	
0057	0	C100	0337	LD	I	0	FETCH 1ST WD OF LINK NAME	SYS03370	
0058	0	D8BB	0338	STD		\$LKNM	SAVE TEMPORARILY	SYS03380	
			0339	* \$S150+1		CONTAINS ADDR	LAST WD OF DISK I/O MINUS 3	SYS03390	
0059	00	65000000	0340	\$S150	LDX	I1	*-*	ADDR END OF DKI/O-1 TO XR1	SYS03400
005B	0	C0FE	0341	LD		\$CILA		SYS03410	
005C	0	1890	0342	SRT		16		SYS03420	
005D	00	440000F2	0343	BSI	L	DZ000	FETCH CI LOADER, PHASE 1	SYS03430	
005F	00	740000EE	0344	MDX	L	\$DBSY,0	SKIP IF READ OP DONE	SYS03440	
0061	0	70FD	0345	MDX		*-3	BR UNTIL READ FINISHED	SYS03450	
0062	0	4102	0346	BSI	I	2	BR TO CI LOADER, PHASE 1	SYS03460	
			0347	*				SYS03470	
0063		0005	0348	BSS		5	RESERVED FOR THE 2250	SYS03480	
0068		0009	0349	BSS		9	PATCH AREA	SYS03490	
			0350	*				SYS03500	
0071	0	0000	0351	\$FLSH	DC		*-*	FLUSH-TO-NEXT-JOB SWITCH 1=FLUSH	SYS03510
0072		0000	0352	BSS	E	0		SYS03520	
0072	0	0000	0353	\$CWCT	DC		*-*	WORD COUNT AND SECTOR ADDRESS	SYS03530
0073	0	0000	0354	DC			*-*	*FOR SAVING/RESTORING COMMON	SYS03540
0074	0	0000	0355	\$CCAD	DC		*-*	ADDR FOR SAVING/RESTORING COMMON	SYS03550
0075	0	0000	0356	\$LSAD	DC		*-*	SCTR ADDR OF 1ST LOCAL/SOCL	SYS03560
0076	0	0000	0357	\$DZ1N	DC		*-*	DISKZ/1/N INDICATOR {-1,0,+1}	SYS03570
0077	0	0000	0358	\$DCDE	DC		*-*	LOGICAL DRIVE CODE FOR PROGRAM	SYS03580
0078	0	0000	0359	\$PHSE	DC		*-*	NO. OF PHASE NOW IN CORE	SYS03590
0079	0	0000	0360	\$UFIO	DC		*-*	UNFORMATTED I/O RECORD NO.	SYS03600
007A	0	0000	0361	\$WSDR	DC		*-*	WORKING STORAGE DRIVE CODE	SYS03610
007B	0	0000	0362	\$WRD1	DC		*-*	LOADING ADDR OF THE CORE LOAD	SYS03620
007C	0	0000	0363	\$KCSW	DC		*-*	1 IF KB,CP BOTH UTILIZED	SYS03630
007D	0	0000	0364	\$UFDR	DC		*-*	UNFORMATTED I/O DRIVE CODE	SYS03640
007E	0	0000	0365	\$CPTR	DC		*-*	CHANNEL 12 INDICATOR FOR CP	SYS03650
007F	0	0000	0366	\$1132	DC		*-*	CHANNEL 12 INDICATOR FOR 1132	SYS03660
0080	0	0000	0367	\$1403	DC		*-*	CHANNEL 12 INDICATOR FOR 1403	SYS03670
			0369	* TRAP FOR			POSTOPERATIVE I/O ERRORS ON LEVEL 1	SYS03690	
			0370	*				SYS03700	
0081	0	0000	0371	\$PST1	DC		*-*	ENTRY POINT	SYS03710
0082	0	3000	0372	WAIT				SYS03720	
0083	00	4C800081	0373	BSC	I	\$PST1	RETURN TO DEVICE SUBROUTINE	SYS03730	
			0374	*				SYS03740	
			0375	* TRAP FOR			POSTOPERATIVE I/O ERRORS ON LEVEL 2	SYS03750	
			0376	*				SYS03760	
0085	0	0000	0377	\$PST2	DC		*-*	ENTRY POINT	SYS03770
0086	0	3000	0378	WAIT				SYS03780	
0087	00	4C800085	0379	BSC	I	\$PST2	RETURN TO DEVICE SUBROUTINE	SYS03790	
			0380	*				SYS03800	
			0381	* TRAP FOR			POSTOPERATIVE I/O ERRORS ON LEVEL 3	SYS03810	
			0382	*				SYS03820	
0089	0	0000	0383	\$PST3	DC		*-*	ENTRY POINT	SYS03830
008A	0	3000	0384	WAIT				SYS03840	
008B	00	4C800089	0385	BSC	I	\$PST3	RETURN TO DEVICE SUBROUTINE	SYS03850	
			0386	*				SYS03860	
			0387	* TRAP FOR			POSTOPERATIVE I/O ERRORS ON LEVEL 4	SYS03870	
			0388	*				SYS03880	
008D	0	0000	0389	\$PST4	DC		*-*	ENTRY POINT	SYS03890
008E	0	3000	0390	WAIT				SYS03900	
008F	00	4C80008D	0391	BSC	I	\$PST4	RETURN TO DEVICE SUBROUTINE	SYS03910	
			0392	*				SYS03920	
			0393	*				SYS03930	
			0394	* PROGRAM STOP			KEY TRAP	SYS03940	
			0395	*				SYS03950	
0091	0	0000	0396	\$STOP	DC		*-*	ENTRY POINT	SYS03960
0092	0	3000	0397	WAIT			WAIT TIL START KEY PUSHED	SYS03970	
0093	00	4CC00091	0398	BOSC	I	\$STOP	RETURN TO CALLER	SYS03980	
			0400	*				SYS04000	
			0401	* PARAMETERS USED BY THE DISK I/O SUBROUTINES. THE				SYS04010	
			0402	* LOGICAL DRIVE CODE IS FOUND IN BITS 1-3 FOR ALL				SYS04020	
			0403	* BUT THE AREA CODE. BIT 0 WILL ALWAYS BE ZERO.				SYS04030	
			0404	*				SYS04040	
			0405	*				SYS04050	
			0406	*** DISK1 AND DISKN WILL NOT WRITE BELOW THE				SYS04060	
			0407	*** FOLLOWING SCTR ADDRESSES (EXCEPT WRITE IMMED).				SYS04070	
			0408	*				SYS04080	
0095	0	0000	0409	\$FPAD	DC		*-*	FILE PROTECT ADDR, LOGICAL DR 0	SYS04090
0096	0	0000	0410	DC			*-*	FILE PROTECT ADDR, LOGICAL DR 1	SYS04100
0097	0	0000	0411	DC			*-*	FILE PROTECT ADDR, LOGICAL DR 2	SYS04110
0098	0	0000	0412	DC			*-*	FILE PROTECT ADDR, LOGICAL DR 3	SYS04120
0099	0	0000	0413	DC			*-*	FILE PROTECT ADDR, LOGICAL DR 4	SYS04130
			0414	*				SYS04140	
			0415	*** THE ARM POSITION IS UPDATED WHENEVER A SEEK				SYS04150	
			0416	*** OCCURS.				SYS04160	
			0417	*				SYS04170	

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
009A	0	0000	0418	\$CYLN	DC	0	ARM POSITION FOR LOGICAL DRIVE 0	SYS04180
009B	0	0000	0419		DC	0	ARM POSITION FOR LOGICAL DRIVE 1	SYS04190
009C	0	0000	0420		DC	0	ARM POSITION FOR LOGICAL DRIVE 2	SYS04200
009D	0	0000	0421		DC	0	ARM POSITION FOR LOGICAL DRIVE 3	SYS04210
009E	0	0000	0422		DC	0	ARM POSITION FOR LOGICAL DRIVE 4	SYS04220
			0423	*				SYS04230
			0424	*** BELOW ARE THE			DISK AREA CODES. A ZERO	SYS04240
			0425	*** INDICATES THE			CORRESPONDING DRIVE IS NOT	SYS04250
			0426	*** ON THE SYSTEM				SYS04260
			0427	*				SYS04270
009F	0	0000	0428	\$ACDE	DC	**	AREA CODE FOR LOGICAL DRIVE 0	SYS04280
00A0	0	0000	0429		DC	**	AREA CODE FOR LOGICAL DRIVE 1	SYS04290
00A1	0	0000	0430		DC	**	AREA CODE FOR LOGICAL DRIVE 2	SYS04300
00A2	0	0000	0431		DC	**	AREA CODE FOR LOGICAL DRIVE 3	SYS04310
00A3	0	0000	0432		DC	**	AREA CODE FOR LOGICAL DRIVE 4	SYS04320
			0433	*				SYS04330
			0434	*** THE ADR OF THE			CYLINDER IN WHICH A DEFECT OC-	SYS04340
			0435	*** CURS, IF ANY, IS			STORED IN THE 1ST, 2ND, OR 3RD	SYS04350
			0436	*** WORD BELOW, DEPEND-			ING ON WHETHER IT IS THE 1ST,	SYS04360
			0437	*** 2ND, OR 3RD DEFECT			ON THE CARTRIDGE.	SYS04370
			0438	*				SYS04380
00A4	0	0000	0439	\$DCYL	DC	**	DEFECTIVE CYLINDER ADDRESSES	SYS04390
00A5	0	0000	0440		DC	**	*FOR LOGICAL DRIVE 0	SYS04400
00A6	0	0000	0441		DC	**		SYS04410
00A7	0	0000	0442		DC	**	DEFECTIVE CYLINDER ADDRESSES	SYS04420
00A8	0	0000	0443		DC	**	*FOR LOGICAL DRIVE 1	SYS04430
00A9	0	0000	0444		DC	**		SYS04440
00AA	0	0000	0445		DC	**	DEFECTIVE CYLINDER ADDRESSES	SYS04450
00AB	0	0000	0446		DC	**	*FOR LOGICAL DRIVE 2	SYS04460
00AC	0	0000	0447		DC	**		SYS04470
00AD	0	0000	0448		DC	**	DEFECTIVE CYLINDER ADDRESSES	SYS04480
00AE	0	0000	0449		DC	**	*FOR LOGICAL DRIVE 3	SYS04490
00AF	0	0000	0450		DC	**		SYS04500
00B0	0	0000	0451		DC	**	DEFECTIVE CYLINDER ADDRESSES	SYS04510
00B1	0	0000	0452		DC	**	*FOR LOGICAL DRIVE 4	SYS04520
00B2	0	0000	0453		DC	**		SYS04530
			0454	*				SYS04550
			0455	*				SYS04560
			0456	* ILS02--THIS SUBROUTINE			SAVES XR1, XR2, STATUS,	SYS04570
			0457	* AND THE ACCUMULATOR			AND ITS EXTENSION.	SYS04580
			0458	* THE ADDRESS OF THE			INTERRUPT SERVICE ROUT-	SYS04590
			0459	* TIME IS STORED IN \$I205			BY PHASE 2 OF	SYS04600
			0460	* THE CORE IMAGE LOADER.			WORD 10 ALWAYS	SYS04610
			0461	* CONTAINS THE ADDRESS OF			\$I200.	SYS04620
			0462	*				SYS04630
			0463	*				SYS04640
			0464	*				SYS04650
00B3	0	0000	0465	\$I200	DC	**	ENTRY PT (LEVEL 2 INTRUPT)	SYS04660
00B4	0	6906	0466		STX	1	\$I210+1 SAVE XR1	SYS04670
00B5	0	6A07	0467		STX	2	\$I210+3 SAVE XR2	SYS04680
00B6	0	2807	0468		STS		\$I210+4 STORE STATUS	SYS04690
00B7	0	D80A	0469		STD		\$I290 SAVE ACCUMULATOR,EXTENSION	SYS04700
			0470	* \$I205+1 CONTAINS ADDR			INTERRUPT ENTRY PT TO DKI/O	SYS04710
00B8	00	44000000	0471	\$I205	BSI	L	** BR TO SERVICE THE INTERRUPT	SYS04720
00BA	00	65000000	0472	\$I210	LX	L1	** RESTORE XR1	SYS04730
00BC	00	66000000	0473		LX	L2	** RESTORE XR2	SYS04740
00BE	0	2000	0474		LDS	0	RESTORE STATUS	SYS04750
00BF	0	C802	0475		LDD		\$I290 RESTORE ACCUMULATOR,EXT	SYS04760
00C0	00	4CC000B3	0476		BOSC	I	\$I200 RETURN FROM INTERRUPT	SYS04770
00C2	0	0000	0477	\$I290	BSS	E	0	SYS04780
00C2	0	0000	0478		DC	**	CONTENTS OF ACCUMULATOR AND	SYS04790
00C3	0	0000	0479		DC	**	*EXTENTION	SYS04800
			0481	*				SYS04820
			0482	* ILS04--THIS SUBROUTINE			SAVES XR1, XR2, STATUS,	SYS04830
			0483	* AND THE ACCUMULATOR			AND ITS EXTENSION.	SYS04840
			0484	* IF THE INTERRUPT IS FOR			A KEYBOARD REQ-	SYS04850
			0485	* UEST, AND IF A MONITOR			PROGRAM IS IN CON-	SYS04860
			0486	* TROL, CONTROL IS PASSED			TO DUMP, OTHER-	SYS04870
			0487	* WISE, CONTROL IS PASSED			TO THE KEYBOARD/	SYS04880
			0488	* CONSOLE PRINTER SUBROUTINE.			WORD 12 AL-	SYS04890
			0489	* WAYS CONTAINS THE ADDRESS			OF \$I400.	SYS04900
			0490	*				SYS04910
			0491	* THE TABLE BELCW			CONTAINS THE ADDRESSES OF THE	SYS04920
			0492	* INTERRUPT SERVICE			ROUTINES FOR ALL THE DEVICES	SYS04930
			0493	* ON LEVEL 4.				SYS04940
			0494	*				SYS04950
			0495	*				SYS04960
			0496	*				SYS04970
00C4	0	0000	0497	\$I400	DC	**	ENTRY POINT	SYS04980
00C5	0	D818	0498		STD		\$I490 SAVE ACCUMULATOR, EXTENSION	SYS04990
00C6	0	280E	0499		STS		\$I410 SAVE STATUS	SYS05000
00C7	0	690F	0500		STX	1	\$I410+2 SAVE XR1	SYS05010
00C8	0	6A10	0501		STX	2	\$I410+4 SAVE XR2	SYS05020
00C9	0	0816	0502		XIO		\$I492 SENSE DSW	SYS05030
00CA	0	1002	0503		SLA	2	IS THIS INTERRUPT REQUEST	SYS05040
00CB	00	4C1000D0	0504		BSC	L	\$I403,- BR IF NOT INTERRUPT REQUEST	SYS05050
00CD	00	4480002C	0505		BSI	I	\$IREQ BR IF INTERRUPT REQUEST	SYS05050

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
00CF	0	FFFE	0506		DC	-2	ERROR CODE	SYS05060
00D0	0	61C9	0507	\$I403	LDX	1 9	NO. DEVICES ON LEVEL TO XR1	SYS05070
00D1	0	0810	0508		XIO	\$I494	SENSE ILSW	SYS05080
00D2	0	1140	0509		SLCA	1	FIND CAUSE OF INTERRUPT	SYS05090
			0510	*	\$I405+1		CONTAINS ADDR OF LEVEL 4 IBT MINUS 1	SYS05100
00D3	00	45800000	0511	\$I405	BSI	I1 *--	BR TO SERVICE THE INTERRUPT	SYS05110
00D5	0	2000	0512	\$I410	LDS	0	RESTORE STATUS	SYS05120
00D6	00	65000000	0513		LDX	L1 *--	RESTORE XR1	SYS05130
00D8	00	66000000	0514		LDX	L2 *--	RESTORE XR2	SYS05140
00DA	0	C803	0515		LDD	\$I490	RESTORE ACCUMULATOR, EXT.	SYS05150
00DB	00	4CC000C4	0516		BOSC	I \$I400	RETURN	SYS05160
			0517	*				SYS05170
			0518	*			CONSTANTS AND WORK AREAS	SYS05180
			0519	*			EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES	SYS05190
			0520	*				SYS05200
00DD	0	0000	0521	\$DDSW	DC	*--	DSW FOR THE DISK	SYS05210
00DE	0	0002	0522	\$I490	BSS	E 2	CONTENTS OF ACCUMULATOR, EXT.	SYS05220
00E0	0	0000	0523	\$I492	DC	*--		SYS05230
00E0	0		0524	\$SYSC	EQU	*-1	VERSION AND MOD NO.	SYS05240
00E1	0	0F00	0525		DC	/0F00	IOCC FOR SENSE IOCC FOR KB/CP	SYS05250
00E2	0	0001	0526	\$I494	BSS	1	PATCH AREA	SYS05260
00E3	0	0300	0527		DC	/0300	IOCC FOR SENSING ILSW04	SYS05270
00E4	0	000A	0529		BSS	10	PATCH AREA	SYS05290
00EE	0	0000	0530	\$DBSY	DC	*--	NON-ZERO WHEN DISK I/O BUSY	SYS05300

DISKZ

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0532	*****				SYS05320
			0533	*				SYS05330
			0534	*PROGRAM NAME-				SYS05340
			0535	* *FULL NAME-FORTRAN/SYSTEM DISK I/O SUBROUTINE				SYS05350
			0536	* *CALLING SEQUENCE-				SYS05360
			0537	* LDD PARAM				SYS05370
			0538	* BSI L DISKZ				SYS05380
			0539	* WHERE PARAM IS THE LABEL OF A DOUBLE-WORD				SYS05390
			0540	* CELL CONTAINING THE FUNCTION CODE AND THE				SYS05400
			0541	* ADDR OF THE I/O BUFFER, I.E., ADDR OF WD CNT.				SYS05410
			0542	* SEE 'CAPABILITIES' FOR DISCUSSION OF PARAM-				SYS05420
			0543	* ETTERS.				SYS05430
			0544	*				SYS05440
			0545	*PURPOSE-				SYS05450
			0546	* TO PROVIDE A SUBROUTINE TO PERFORM DISK OPERA-				SYS05460
			0547	* TIONS. THIS SUBROUTINE IS INTENDED FOR USE BY				SYS05470
			0548	* MONITOR PROGRAMS AND USER PROGRAMS WRITTEN IN				SYS05480
			0549	* FORTRAN. THUS, IT IS INTENDED FOR USE IN AN				SYS05490
			0550	* ERROR-FREE ENVIRONMENT.				SYS05500
			0551	*				SYS05510
			0552	*METHOD-				SYS05520
			0553	* DISKZ REQUIRES A BUFFER, THE LENGTH OF WHICH IS				SYS05530
			0554	* 2 GREATER THAN THE NO. WORDS TO BE READ/WRIT-				SYS05540
			0555	* TEN.				SYS05550
			0556	*				SYS05560
			0557	*CAPABILITIES AND LIMITATIONS-				SYS05570
			0558	* THE WD CNT, AS WELL AS DZ000, MUST BE ON AN EVEN				SYS05580
			0559	* BOUNDARY, MUST BE IN THE RANGE 0-32767. THE				SYS05590
			0560	* DRIVE CODE MUST BE IN BITS 1-3 OF THE SECTOR				SYS05600
			0561	* ADDR, WHICH FOLLOWS THE WD CNT. THE FUNCTION				SYS05610
			0562	* INDICATOR MUST BE XX00 FOR A READ OR XX01 FOR				SYS05620
			0563	* A WRITE, WHERE 'XX' MEANS ANY 2 HEXADECIMAL				SYS05630
			0564	* CHARACTERS. A WD CNT OF ZERO INDICATES A SEEK.				SYS05640
			0565	* (READ OR WRITE MAY BE INDICATED.) AUTOMATIC				SYS05650
			0566	* SEEKING IS PROVIDED AS A PART OF READ/WRITE.				SYS05660
			0567	* A WRITE IS ALWAYS WITH A READ-BACK-CHECK.				SYS05670
			0568	* DISKZ MAKES NO PREOPERATIVE PARAMETER CHECKS.				SYS05680
			0569	*				SYS05690
			0570	*SPECIAL FEATURES-				SYS05700
			0571	* DISKZ PROVIDES ONLY THOSE FUNCTIONS MENTIONED				SYS05710
			0572	* ABOVE. DISK1 AND DISKN OFFER THIS BASIC SET OF				SYS05720
			0573	* FUNCTIONS PLUS OTHERS.				SYS05730
			0574	*				SYS05740
			0575	*****				SYS05750
			0577	* PROVIDE PARAMETERS FOR SYSTEM LOADER				SYS05770
			0578	*				SYS05780
00F0	00C0		0579		BSS	E 0		SYS05790
00F0	0	00EF	0580		DC	\$ZEND-*	DISKZ WORD COUNT	SYS05800
00F1	0	FF6A	0581		DC	-1DZID	PHASE ID	SYS05810
00F2	0	00E8	0582		DC	\$ZEND-6-*+1	ADDR OF SLET EXTRACT	SYS05820
00F3	0	00C1	0583		DC	1	NO. ENTRIES IN SLET EXTRACT	SYS05830
00F4			0584		ORG	*-2		SYS05840

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
00F2	0	0000	0586	DZ000	DC	**	ENTRY POINT	SYS05860
00F3	00	740000EE	0587	MDX	L	\$DBSY,0	LOOP UNTIL OPERATION IN	SYS05870
00F5	0	70FD	0588	MDX	**	-3	*PROGRESS IS COMPLETE	SYS05880
00F6	0	7002	0589	MDX		DZ020	BR AROUND INT ENTRY POINT	SYS05890
			0590	*				SYS05900
			0591	*			* INTERRUPT ENTRY POINT	SYS05910
			0592	*				SYS05920
00F7	0	0000	0593	DZ010	DC	**	INTERRUPT ADDRESS	SYS05930
00F8	0	7015	0594	MDX		DZ180	BR TO SERVICE INTERRUPT	SYS05940
00F9	0	690F	0595	DZ020	STX	1 DZ100+1	SAVE XR1	SYS05950
00FA	0	6A10	0596	STX		2 DZ100+3	SAVE XR2	SYS05960
00FB	0	1008	0597	SLA		8	SHIFT INDICATOR 8 BITS	SYS05970
00FC	0	D03C	0598	STO		DZ945	SAVE FUNCTION INDICATOR	SYS05980
00FD	0	1800	0599	RTE		16		SYS05990
00FE	0	D05B	0600	STO		DZ235+1	SAVE ADDR OF THE I/O AREA	SYS06000
00FF	0	6211	0601	DZ030	LX	2 *TCNT	TURN BUSY INDICATOR ON AND	SYS06010
0100	0	6AED	0602	STX		2 \$DBSY	*SET RETRY COUNT	SYS06020
0101	0	C0F0	0603	LD		DZ000		SYS06030
0102	0	D0F4	0604	STO		DZ010		SYS06040
0103	0	7053	0605	MDX		DZ230	BR TO CONTINUE	SYS06050
0104	00	4C000000	0606	DZ060	BSC	L **	BR TO SERVICE THE INTERRUPT	SYS06060
			0607	*				SYS06070
			0608	*			* START ALL DISK OPERATIONS	SYS06080
			0609	*				SYS06090
0106	0	6908	0610	DZ070	STX	1 DZ180+1	SAVE ADDR OF THE I/O AREA	SYS06100
0107	0	081E	0611	XIO		DZ904	START AN OPERATION	SYS06110
			0612	*				SYS06120
			0613	*			* RETURN TO USER	SYS06130
			0614	*				SYS06140
0108	00	65000000	0615	DZ100	LX	L1 **	RESTORE XR1	SYS06150
010A	00	66000000	0616	LX		L2 **	RESTORE XR2	SYS06160
010C	00	4C8000F7	0617	BSC	I	DZ010	RETURN	SYS06170
			0618	*				SYS06180
			0619	*			* SERVICE ALL INTERRUPTS	SYS06190
			0620	*				SYS06200
010E	00	65000000	0621	DZ180	LX	L1 **	ADDR OF I/O AREA TO XR1	SYS06210
0110	00	660000F2	0622	LX		L2 DZ000	ADDR OF DZ000 TO XR2	SYS06220
0112	0	0819	0623	XIO		DZ910	SENSE THE DSW	SYS06230
0113	0	D0C9	0624	STO		\$DDSW	SAVE THE DSW	SYS06240
0114	0	4850	0625	BOSC		-	SKIP IF ERROR BIT SET	SYS06250
0115	0	70EE	0626	MDX		DZ060	BRANCH IF ERROR BIT NOT SET	SYS06260
0116	0	C80D	0627	DZ185	LDD	DZ902	RESTORE WORD COUNT	SYS06270
0117	0	D900	0628	STD		1 0	*AND SECTOR ADDRESS	SYS06280
0118	00	74FF00EE	0629	MDX	L	\$DBSY,-1	SKIP IF 16 RETRIES DONE	SYS06290
011A	0	703E	0630	MDX		DZ235	BRANCH IF LESS THAN 16	SYS06300
			0631	*				SYS06310
			0632	*			* TRAP OUT TO POSTOPERATIVE TRAP	SYS06320
			0633	*				SYS06330
0118	0	C812	0634	LDD		DZ912	1+SCTR ADDR TO EXTENSION	SYS06340
011C	0	C014	0635	LD		DZ915		SYS06350
011D	0	4293	0636	DZ190	BSI	2 \$PST2-X2	BR TO POSTOPERATIVE ER TRAP	SYS06360
011E	0	1810	0637	SRA		16	CLEAR	SYS06370
011F	00	D4800198	0638	STO	I	DZ350+1	*ARM POSITION	SYS06380
0121	0	70DD	0639	MDX		DZ030	RETRY OPERATION	SYS06390
			0640	*				SYS06400
			0641	*			* CONSTANTS AND WORK AREAS	SYS06410
			0642	*				SYS06420
0122	0000		0643	BSS	E	0		SYS06430
			0644	*			* EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES	SYS06440
0122	0	0001	0645	DZ900	DC	1	CONSTANT,READ-AFTER-SEEK WD CNT	SYS06450
0123	0	0000	0646	DZ901	DC	0	CURRENT ARM POSITION	SYS06460
0124	0	0000	0647	DZ902	DC	**	LAST TWO WORDS OF SECTOR	SYS06470
0125	0	0000	0648	DC	**	**	*PREVIOUSLY READ	SYS06480
0126	0	0000	0649	DZ904	DC	**	IOCC FOR OPERATION CURRENTLY	SYS06490
0127	0	0000	0650	DZ905	DC	**	*BEING PERFORMED	SYS06500
0128	0	0000	0651	DZ906	DC	**	SAVE AREA FOR IOCC FOR	SYS06510
0129	0	0000	0652	DZ907	DC	**	*USER-REQUESTED OPERATION	SYS06520
012A	0	0122	0653	DZ908	DC	DZ900	IOCC FOR READ	SYS06530
012B	0	0000	0654	DZ909	DC	**	*AFTER SEEK	SYS06540
012C	0	0000	0655	DZ910	DC	**	2ND WORD OF SEEK IOCC	SYS06550
012D	0	0000	0656	DZ911	DC	**	SENSE IOCC	SYS06560
012E	0	0000	0657	DZ912	DC	**	INTERMEDIATE WORD COUNT	SYS06570
012F	0	0000	0658	DZ913	DC	**	ADDR OF NEXT SEQUENTIAL SECTOR	SYS06580
0130	0	5002	0659	DZ914	DC	/5002	WRITE SELECT/POWER UNSAFE INDR	SYS06590
0131	0	5004	0660	DZ915	DC	/5004	READ/WRITE/SEEK ERROR INDICATOR	SYS06600
0132	0	FEC0	0661	DZ916	DC	-320	TO BE USED TO SIMULTANEOUSLY	SYS06610
0133	0	0001	0662	DC		1	*DECR WD CNT, INCR SCTR ADDR	SYS06620
0134	0	0080	0663	DZ920	DC	/0080	READ CHECK BIT FOR IOCC	SYS06630
0135	0	0600	0664	DZ925	DC	/0600	2ND WD OF READ IOCC W/O AREA CD	SYS06640
0136	0	0008	0665	DZ930	DC	8	NO. SECTORS PER CYLINDER	SYS06650
0137	0	5000	0666	DZ935	DC	/5000	NOT READY DISPLAY CODE	SYS06660
0138	0	0FF8	0667	DZ940	DC	/0FF8	'AND' OUT DR CODE, SCTR ADDR	SYS06670
0139	0	0000	0668	DZ945	DC	**	FUNC INDICATOR (0=READ,1=WRITE)	SYS06680
013A	0	0701	0669	DZ950	DC	/0701	SENSE IOCC W/O AREA CODE	SYS06690
013B	0	0007	0670	DZ955	DC	/0007	'AND' OUT ALL BUT SCTR NO.	SYS06700
013C	0	000A	0671	DZ960	DC	\$DCYL-\$CYLN	BASE DEFECTIVE CYL ADDR	SYS06710

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
013D	0	009F	0672	DZ965	DC		\$ACDE BASE AREA CODE ADDR	SYS06720
013E	0	FFF8	0673	DZ970	DC		\$CYLN-\$ACDE BASE ARM POSITION ADDR	SYS06730
013F	0	0000	0674	DZ975	DC		*-- 2ND WORD OF READ CHECK IOCC	SYS06740
0140	0	0400	0675	DZ980	DC		/0400 2ND WD OF SEEK IOCC W/O AREA CD	SYS06750
0141	0	0141	0676	DZ985	DC		321 NO. WORDS PER SECTOR (W/ ADDR)	SYS06760
0142	0	0000	0677	DZ990	DC		*-- CURRENT SECTOR NO.	SYS06770
0143	0	FFFF	0678	DZ995	DC		-1 MASK FOR COMPLEMENTING	SYS06780
			0679	*				SYS06790
			0680	*			* RESERVED FOR SAVING CORE ON A DUMP ENTRY TO SKEL	SYS06800
			0681	*				SYS06810
0144	0002		0682		BSS	2	THIS AREA MUST BE AT \$CIBA+3I9	SYS06820
00F2	0		0683	X2	EQU	DZ000		SYS06830
			0684	*				SYS06840
			0685	*				SYS06850
			0686	*				SYS06860
0146	0	1810	0687	DZ210	SRA	16		SYS06870
0147	0	D0A6	0688	STO	\$DBSY		CLEAR BUSY INDICATOR	SYS06880
0148	00	74FF0032	0689	MDX	L	\$IOCT,-1	DECREMENT IOCS COUNTER	SYS06890
014A	0	1000	0690	NOP				SYS06900
014B	0	70BC	0691	MDX		DZ100	TO EXIT	SYS06910
			0692	*				SYS06920
			0693	*			* PREPARE TO TRAP OUT ON *POWER UNSAFE* CONDITION	SYS06930
			0694	*				SYS06940
014C	0	COE3	0695	DZ215	LD	DZ914		SYS06950
014D	0	70CF	0696	MDX		DZ190	BR TO TPAP OUT	SYS06960
			0697	*				SYS06970
			0698	*			* PREPARE TO TRAP OUT ON *NOT READY* CONDITION	SYS06980
			0699	*				SYS06990
014E	0	COE8	0700	DZ220	LD	DZ935	FETCH ERROR CODE	SYS07000
014F	00	44000028	0701	BSI	L	\$PRET	BR TO PREOPERATIVE ERR TRAP	SYS07010
0151	0	703A	0702	MDX		DZ340	RETRY THE OPERATION	SYS07020
			0703	*				SYS07030
			0704	*				SYS07040
			0705	*				SYS07050
0152	0	COD9	0706	DZ225	LD	DZ910	FETCH 1ST WD SEEK IOCC	SYS07060
0153	0	18D0	0707	RTE		16		SYS07070
0154	0	C101	0708	LD		1 1	FETCH DESIRED CYLINDER ADDR	SYS07080
0155	0	1803	0709	SRA		3		SYS07090
0156	0	704D	0710	MDX		DZ380	BR TO PERFORM THE SEEK	SYS07100
			0711	*				SYS07110
			0712	*				SYS07120
			0713	*				SYS07130
0157	00	74010032	0714	DZ230	MDX	L \$IOCT,1	INCREMENT IOCS COUNTER	SYS07140
0159	00	65000000	0715	DZ235	L	L1 *--	ADDR I/O AREA TO XR1	SYS07150
015B	0	C900	0716	LDD		1 0		SYS07160
015C	0	D8C7	0717	STD		DZ902	SAVE WORD COUNT, SCTR ADDR	SYS07170
015D	0	D8D0	0718	STD		DZ912		SYS07180
015E	0	1810	0719	DZ240	SRA	16		SYS07190
015F	0	1084	0720	SLT		4	DRIVE CODE IN BITS 12-15	SYS07200
0160	0	D00E	0721	STO		DZ280+1		SYS07210
0161	0	80DB	0722	A		DZ965	COMPUTE AND STORE THE	SYS07220
0162	0	D01B	0723	STO		DZ330+1	*ADDR OF THE AREA CODE	SYS07230
0163	0	80DA	0724	A		DZ970	COMPUTE AND STORE THE	SYS07240
0164	0	D033	0725	STO		DZ350+1	*ADDR OF THE ARM POSITION	SYS07250
0165	0	80D6	0726	A		DZ960	ADD IN BASE DT ADDR	SYS07260
0166	0	8008	0727	A		DZ280+1	ADD IN THE DRIVE	SYS07270
0167	0	8007	0728	A		DZ280+1	*CODE TWICE MORE	SYS07280
0168	0	D00E	0729	STO		DZ280+1		SYS07290
0169	0	62FD	0730	L		2 -3	INITIALIZE COUNTER FOR LOOP	SYS07300
016A	0	698D	0731	STX		1 DZ906		SYS07310
016B	0	C101	0732	LD		1 1	FETCH DESIRED SECTOR ADDR	SYS07320
016C	0	E0CB	0733	AND		DZ940	*AND* OUT SECTOR NO.	SYS07330
016D	0	D101	0734	DZ250	STO	1 1	*AND DRIVE CODE	SYS07340
016E	00	94000000	0735	DZ280	S	L *--	SUB DEFECTIVE CYLINDER ADDR	SYS07350
0170	0	4828	0736	BSC		Z+	SKIP IF BAD CYLINDER	SYS07360
0171	0	7006	0737	MDX		DZ300	BR TO CONTINUE PROCESSING	SYS07370
0172	0	C101	0738	LD		1 1		SYS07380
0173	0	80C2	0739	A		DZ930	INCREMENT SCTR ADDR BY 8	SYS07390
0174	00	7401016F	0740	MDX	L	DZ280+1,1	POINT TO NEXT DEFECTIVE CYL	SYS07400
0176	0	7201	0741	MDX		2 1	SKIP AFTER 3RD PASS	SYS07410
0177	0	70F5	0742	MDX		DZ250	COMPARE W/ NEXT DEF CYL ADR	SYS07420
			0743	*				SYS07430
			0744	*			* CONSTRUCT THE 2ND WORD OF ALL IOCC*S	SYS07440
			0745	*				SYS07450
0178	00	660000F2	0746	DZ300	L	L2 DZ000	ADDR OF DZ000 TO XR2	SYS07460
017A	0	C23D	0747	LD		2 DZ913-X2	FETCH SECTOR ADDRESS	SYS07470
017B	0	E249	0748	AND		2 DZ955-X2	*AND* OUT ALL BUT SECTOR NO	SYS07480
017C	0	D250	0749	STO		2 DZ990-X2	SAVE SECTOR NO.	SYS07490
017D	00	C4000000	0750	DZ330	LD	L *--	FETCH AREA CODE	SYS07500
017F	0	EA4E	0751	OR		2 DZ980-X2	*OR* IN SEEK FUNCTION CODE	SYS07510
0180	0	D23A	0752	STO		2 DZ910-X2	SEEK IOCC MINUS DIRECTION	SYS07520
0181	0	EA43	0753	OR		2 DZ925-X2	*OR* IN READ FUNCTION CODE	SYS07530
0182	0	D239	0754	STO		2 DZ909-X2	IOCC FOR READ-AFTER-SEEK	SYS07540
0183	0	EA50	0755	OR		2 DZ990-X2	*OR* IN SECTOR NO.	SYS07550
0184	0	9247	0756	S		2 DZ945-X2	COMPLETE READ/WRITE CODE	SYS07560
0185	0	D237	0757	STO		2 DZ907-X2	2ND WD OF READ/WRITE IOCC	SYS07570

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO	
0186	0	EA42	0758		CR	2	DZ920-X2 *OR' IN READ CHECK BIT	SYS07580	
0187	0	8247	0759		A	2	DZ945-X2	SYS07590	
0188	0	D24D	0760		STO	2	DZ975-X2 2ND WD OF READ CHECK IOCC	SYS07600	
0189	0	EA48	0761		OR	2	DZ950-X2 *OR' IN SENSE IOCC BITS	SYS07610	
018A	0	D23B	0762		STO	2	DZ911-X2 COMPLETED SENSE IOCC	SYS07620	
018B	0	CA3C	0763		LDD	2	DZ912-X2 1+SCTR ADDR TO EXTENSION	SYS07630	
018C	0	0A3A	0764	DZ340	XIO	2	DZ910-X2 SENSE FOR DISK READY	SYS07640	
018D	0	02EB	0765		STO	2	\$ODSW-X2 SAVE THE DSW	SYS07650	
018E	0	4828	0766		BSC	Z+	SKIP UNLESS POWER UNSAFE OR	SYS07660	
018F	0	70BC	0767		MDX	DZ215	*WRITE SELECT, BR OTHERWISE	SYS07670	
0190	0	1002	0768		SLA	2	BR TO PREOPERATIVE ERR TRAP	SYS07680	
0191	0	4828	0769		BSC	Z+	*IF DISK NOT READY, SKIP	SYS07690	
0192	0	70BB	0770		MDX	DZ220	*OTHERWISE	SYS07700	
0193	0	100B	0771		SLA	11		SYS07710	
0194	0	4828	0772		BSC	Z+		SYS07720	
0195	0	70BC	0773		MDX	DZ225		SYS07730	
0196	0	C101	0774		LD	1	1	FETCH DESIRED CYLINDER ADDR	SYS07740
0197	00	94000000	0775	DZ350	S	L	**	SUBTRACT ARM POSITION	SYS07750
0199	0	4818	0776		BSC	+-	SKIP IF SEEK NECESSARY	SYS07760	
019A	0	7014	0777		MDX	DZ400	BRANCH TO PERFORM OPERATION	SYS07770	
			0778		*			SYS07780	
			0779		* SEEK			SYS07790	
			0780		*			SYS07800	
019B	0	1893	0781		SRT	19	PUT NO. CYLINDERS IN EXT	SYS07810	
019C	0	180F	0782		SRA	15	+ OR - SIGN TO BIT 15	SYS07820	
019D	0	1002	0783		SLA	2	SHIFT SIGN TO BIT 13	SYS07830	
019E	0	EA3A	0784		OR	2	DZ910-X2 OR IN REMAINDER OF IOCC	SYS07840	
019F	0	18D0	0785		RTE	16		SYS07850	
01A0	0	4810	0786		BSC	-	SKIP IF SEEK TOWARD HOME	SYS07860	
01A1	0	7002	0787		MDX	DZ380	BRANCH IF SEEK TOWARD CENTR	SYS07870	
01A2	0	F251	0788		EOR	2	DZ995-X2 COMPLEMENT NO. CYLS TO BE	SYS07880	
01A3	0	8230	0789		A	2	DZ900-X2 *SOUGHT TO GET POSITIVE NO.	SYS07890	
01A4	0	DA34	0790	DZ380	STD	2	DZ904-X2	SYS07900	
01A5	0	4213	0791		BSI	2	DZ070-1-X2 START SEEK	SYS07910	
			0792		*			SYS07920	
			0793		* SEEK COMPLETE INTERRUPT PROCESSING			SYS07930	
			0794		*			SYS07940	
01A6	0	CA38	0795		LDD	2	DZ908-X2 SET UP IOCC FOR	SYS07950	
01A7	0	DA34	0796		STD	2	DZ904-X2 *READ AFTER SEEK	SYS07960	
01A8	0	4213	0797		BSI	2	DZ070-1-X2 START READ-AFTER-SEEK	SYS07970	
			0798		*			SYS07980	
			0799		* READ-AFTER-SEEK COMPLETE INTERRUPT PROCESSING			SYS07990	
			0800		*			SYS08000	
01A9	0	C231	0801		LD	2	DZ901-X2 FETCH ADR OF SCTR JUST READ	SYS08010	
01AA	00	D4800198	0802		STO	I	DZ350+1 UPDATE ARM POSITION	SYS08020	
01AC	C	9101	0803		S	1	1 SUB DESIRED SCTR ADDR	SYS08030	
01AD	00	4C200116	0804		BSC	L	DZ185,Z BR IF SEEK UNSUCCESSFUL	SYS08040	
			0805		*			SYS08050	
			0806		*			SYS08060	
			0807		* READ/WRITE			SYS08070	
			0808		*			SYS08080	
01AF	0	CA3C	0809	DZ400	LDD	2	DZ912-X2 FETCH INTERMEDIATE WD CNT	SYS08090	
01B0	0	4808	0810		BSC	+	SKIP, WD CNT NOT EXHAUSTED	SYS08100	
01B1	0	7094	0811	DZ410	MDX	DZ210	BRANCH IF READ/WRITE DONE	SYS08110	
01B2	0	8A40	0812		AD	2	DZ916-X2 DECREMENT WORD COUNT AND	SYS08120	
01B3	0	DA3C	0813		STD	2	DZ912-X2 *INCREMENT SECTOR ADDRESS	SYS08130	
01B4	0	4830	0814		BSC	Z-	SKIP IF THIS IS LAST SECTOR	SYS08140	
01B5	0	1810	0815		SRA	16	CLEAR ACCUMULATOR	SYS08150	
01B6	0	824F	0816		A	2	DZ985-X2 ADD BACK 321 TO WD CNT	SYS08160	
01B7	0	0100	0817		STO	1	0 STORE RESULT IN I/O AREA	SYS08170	
01B8	0	CA36	0818		LDD	2	DZ906-X2 RESTORE IOCC FOR ORIGINALLY	SYS08180	
01B9	0	DA34	0819		STD	2	DZ904-X2 *REQUESTED OPERATION	SYS08190	
01BA	0	C101	0820		LD	1	1 ADD SECTOR NO. TO SECTOR	SYS08200	
01BB	0	EA50	0821		OR	2	DZ990-X2 *ADDRESS	SYS08210	
01BC	0	D101	0822		STO	1	1	SYS08220	
01BD	0	4213	0823		BSI	2	DZ070-1-X2 START READ/WRITE OPERATION	SYS08230	
			0824		*			SYS08240	
			0825		* READ/WRITE COMPLETE INTERRUPT PROCESSING			SYS08250	
			0826		*			SYS08260	
01BE	0	C240	0827		LD	2	DZ975-X2 SET UP FOR READ CHECK	SYS08270	
01BF	0	D235	0828		STO	2	DZ905-X2	SYS08280	
01C0	0	C247	0829		LD	2	DZ945-X2 FETCH FUNCTION INDICATOR	SYS08290	
01C1	0	4820	0830		BSC	Z	SKIP IF READ REQUESTED	SYS08300	
01C2	0	4213	0831		BSI	2	DZ070-1-X2 START READ CHECK OPERATION	SYS08310	
01C3	0	CA32	0832		LDD	2	DZ902-X2 RESTORE LAST 2 WDS OF SEC-	SYS08320	
01C4	0	D900	0833		STD	1	0 *TOR PREVIOUSLY READ	SYS08330	
01C5	0	C23C	0834		LD	2	DZ912-X2 FETCH INTERMEDIATE WD CNT	SYS08340	
01C6	0	4808	0835		BSC	+	SKIP IF MORE READING/WRTING	SYS08350	
01C7	0	70E9	0836		MDX	DZ410	BRANCH IF FINISHED	SYS08360	
01C8	00	75000140	0837		MDX	L1	320 POINT XRI TO NEW I/O AREA	SYS08370	
01CA	C	C900	0838		LDD	1	0 SAVE LAST 2 WDS OF SECTOR	SYS08380	
01CB	0	DA32	0839		STD	2	DZ902-X2 *JUST READ/WRTTEN	SYS08390	
01CC	0	CA3C	0840		LDD	2	DZ912-X2 WD CNT, SCTR ADDR NEXT OP	SYS08400	
01CD	0	D900	0841		STD	1	0 STORE BOTH IN NEW I/O AREA	SYS08410	
01CE	0	70BF	0842		MDX	DZ240	BACK TO SET UP NEXT OPERATN	SYS08420	

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0843	*				SYS08430
			0844	*				SYS08440
01CF		000B	0845		BSS	11	PATCH AREA	SYS08450
			0846	*				SYS08460
			0847	*				SYS08470
01DA	0	00A0	0848		DC		'CIL1 ID NO. OF CORE IMAGE LDR,PH	SYS08480
C1DR	0	00C0	0849	\$CIDN	DC	**	CORE ADDR/CID NO.	SYS08490
010C	0	00C0	0850		DC	**	WORD COUNT	SYS08500
01DD	0	00C0	0851		DC	**	SCTR ADDR	SYS08510
01DE		0002	0852		BSS	2	WD CNT, SCTR ADDR CORE LDS	SYS08520
01E0	C		0853	\$ZEND	EQU	*	1 + END OF DISKZ	SYS08530

EQUIVALENCES

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0855	*				SYS08550
			0856	*	EQUIVALENCES FOR PHASE ID NUMBERS			SYS08560
			0857	*				SYS08570
006E	0		0858	'SUP1	EQU	110	PHASE ID FOR MCRA	SYS08580
0070	C		0859	'SUP3	EQU	112	PHASE ID FOR DUMP PROGRAM	SYS08590
0071	C		0860	'SUP4	EQU	113	PHASE ID FOR ERR ANAL PROG	SYS08600
0078	0		0861	'CLB0	EQU	120	PHASE ID FOR CLB, PHASE 0/1	SYS08610
008C	0		0862	'1403	EQU	140	PHASE ID FOR SYS 1403 SUBR	SYS08620
008D	C		0863	'1132	EQU	141	PHASE ID FOR SYS 1132 SUBR	SYS08630
008E	0		0864	'CPTR	EQU	142	PHASE ID FOR SYS CP SUBR	SYS08640
008F	0		0865	'2501	EQU	143	PHASE ID FOR SYS 2501 SUBR	SYS08650
0090	C		0866	'1442	EQU	144	PHASE ID FOR SYS 1442 SUBR	SYS08660
0091	C		0867	'1134	EQU	145	PHASE ID FOR SYS 1134 SUBR	SYS08670
0092	0		0868	'KBCP	EQU	146	PHASE ID FOR SYS KB/CP SUBR	SYS08680
0093	C		0869	'CDCV	EQU	147	PHASE ID FOR SYS CD CONV	SYS08690
0094	C		0870	'PTCV	EQU	148	PHASE ID FOR SYS 1134 CONV	SYS08700
0095	0		0871	'KBCV	EQU	149	PHASE ID FOR SYS KB CONV	SYS08710
0096	0		0872	'DZID	EQU	150	PHASE ID FOR DISKZ	SYS08720
0097	C		0873	'D1ID	EQU	151	PHASE ID FOR DISKI	SYS08730
0098	0		0874	'DNID	EQU	152	PHASE ID FOR DISKN	SYS08740
00A0	0		0875	'CIL1	EQU	160	PHASE ID FOR CI LOADER,PH 1	SYS08750
00A1	0		0876	'CIL2	EQU	161	PHASE ID FOR CI LOADER,PH 2	SYS08760
			0877	*				SYS08770
			0878	*	EQUIVALENCES FOR RESIDENT MONITOR			SYS08780
			0879	*				SYS08790
0014	0		0880	\$LKNM	EQU	\$HASH	SAVE AREA FOR NAME OF LINK	SYS08800
0016	0		0881	\$RMSW	EQU	\$HASH+2	EXIT-LINK-DUMP SW(-1,0,+1)	SYS08810
0017	0		0882	\$CXRI	EQU	\$HASH+3	SAVE AREA FOR XR1	SYS08820
0018	0		0883	\$CLSW	EQU	\$HASH+4	SW FOR CORE IMAGE LDR,PH 2	SYS08830
0019	C		0884	\$DMPF	EQU	\$HASH+5	DUMP FORMAT CODE	SYS08840
001A	0		0885	\$ACEX	EQU	\$HASH+6	ACC AND EXT WHEN ENTER DUMP	SYS08850
005A	0		0886	\$CILA	EQU	\$S150+1	ADDR OF END OF DK I/O - 3	SYS08860
00B9	0		0887	\$IBT2	EQU	\$I205+1	ADDR OF SERVICE PART OF DK10	SYS08870
00D4	0		0888	\$IBT4	EQU	\$I405+1	ADDR OF THE IBT	SYS08880
00EF	0		0889	\$SNLT	EQU	\$DBSY+1	SENSE LIGHT INDICATOR	SYS08890
00F0	0		0890	\$PAUS	EQU	DZ000-2	PAUSE, INTERRUPT INDICATOR	SYS08900
00F1	0		0891	\$RWCZ	EQU	DZ000-1	READ/WRITE SWITCH (CARDZ)	SYS08910
			0892	*				SYS08920
			0893	*	EQUIVALENCES FOR ABSOLUTE SECTOR ADDRESSES			SYS08930
			0894	*				SYS08940
0000	0		0895	'IDAD	EQU	0	ADDR OF SCTR WITH ID,DEF CYL ADDR	SYS08950
0001	0		0896	'DCGM	EQU	1	ADDR OF SCTR CONTAINING DCOM	SYS08960
0002	0		0897	'RIAD	EQU	2	ADDR OF SCTR CONTAINING RES IMG	SYS08970
0003	0		0898	'SLET	EQU	3	ADDR OF SCTR CONTAINING SLET	SYS08980
0006	0		0899	'RTBL	EQU	6	ADDR OF SCTR CONTAINING RELO TBL	SYS08990
0007	0		0900	'HDNG	EQU	7	ADDR OF SCTR CONTAINING PAGE HDR	SYS09000
0000	0		0901	'SSTR	EQU	0	ADDR OF SCTR W/ COLD START PROG	SYS09010
			0902	*				SYS09020
			0903	*	EQUIVALENCES FOR THE CORE IMAGE HEADER			SYS09030
			0904	*				SYS09040
0000	C		0905	'XEGA	EQU	0	RLTV ADDR OF CORE LOAD EXEC ADDR	SYS09050
0001	C		0906	'CMON	EQU	1	RLTV ADDR OF WD CNT OF COMMON	SYS09060
0002	C		0907	'DREQ	EQU	2	RLTV ADDR OF DISK I/O INDICATOR	SYS09070
0003	0		0908	'FILE	EQU	3	RLTV ADDR OF NO. FILES DEFINED	SYS09080
0004	0		0909	'HWCT	EQU	4	RLTV ADDR OF WD CNT OF CI HEADER	SYS09090
0005	C		0910	'LSCT	EQU	5	SCTR CNT OF FILES IN WK STORAGE	SYS09100
0006	C		0911	'LOAD	EQU	6	RLTV ADDR OF LOAD ADDR CORE LOAD	SYS09110
0007	0		0912	'XCTL	EQU	7	RLTV ADDR DISK1/DISKN EXIT CTRL	SYS09120
0008	0		0913	'TVWC	EQU	8	RLTV ADDR OF WD CNT OF TV	SYS09130
0009	C		0914	'WCNT	EQU	9	RLTV ADDR OF WD CNT OF CORE LOAD	SYS09140
000A	0		0915	'XR3X	EQU	10	RLTV ADDR OF EXEC SETTING OF XR3	SYS09150
000B	0		0916	'ITVX	EQU	11	RLTV ADDR OF 1ST WD OF ITV	SYS09160
0011	0		0917	'ILS4	EQU	17	RLTV ADDR OF 1ST WD OF IBT4	SYS09170
001A	0		0918	'OVSW	EQU	26	RLTV ADDR OF LOCAL/SOCAL SWITCH	SYS09180
001C	0		0919	'CCRE	EQU	28	CORE SIZE OF BUILDING SYSTEM	SYS09190

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
001D	C		0920	*HEND	EQU	29	RLTV ADDR OF LAST WD OF CI HDR	SYS09200
			0921	*				SYS09210
			0922	*EQUIVALENCES			FOR LET/FLET	SYS09220
			0923	*				SYS09230
0005	C		0924	*LFHD	EQU	5	WORD COUNT OF LET/FLET HEADER	SYS09240
0003	C		0925	*LFEN	EQU	3	NO OF WDS PER LET/FLET ENTRY	SYS09250
0000	C		0926	*SCTN	EQU	0	RLTY ADDR OF LET/FLET SCTR NO.	SYS09260
0001	C		0927	*UAFX	EQU	1	RLTV ADDR OF SCTR ADDR OF UA/FXA	SYS09270
0003	C		0928	*WDSA	EQU	3	RLTV ADDR OF WDS AVAIL IN SCTR	SYS09280
0004	C		0929	*NEXT	EQU	4	RLTV ADDR OF ADDR NEXT SCTR	SYS09290
0000	C		0930	*LFNM	EQU	0	RLTV ADDR OF LET/FLET ENTRY NAME	SYS09300
0002	C		0931	*BLCT	EQU	2	RLTV ADDR OF LET/FLET ENTRY DBCT	SYS09310
			0932	*				SYS09320
			0933	*MISCELLANEOUS			EQUIVALENCES	SYS09330
			0934	*				SYS09340
0005	C		0935	*MXDR	EQU	5	MAX NO. DRIVES SUPPORTED	SYS09350
0380	C		0936	*COMZ	EQU	896	LOW COMMON LIMIT FOR DISKZ	SYS09360
04C0	C		0937	*COM1	EQU	1216	LOW COMMON LIMIT FOR DISK1	SYS09370
0600	C		0938	*CCM2	EQU	1536	LOW COMMON LIMIT OF DISKN	SYS09380
0011	C		0939	*TCNT	EQU	17	NO. TRIES BEFORE DISK ERROR	SYS09390
00F9	C		0940	*DKEP	EQU	D2000+7	LIBF ENTRY TO DISK1/N	SYS09400
00F7	C		0941	*DKIP	EQU	D2000+5	DISK I/O INTERRUPT ENTRY PT	SYS09410
007F	C		0942	Y	EQU	127		SYS09420

COLD START PROGRAM

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0944	*****				SYS09440
			0945	*				SYS09450
			0946	*STATUS - VERSION 2, MODIFICATION LEVEL 0.				SYS09460
			0947	*				SYS09470
			0948	*FUNCTION/OPERATION -				SYS09480
			0949	* THIS PROGRAM IS READ INTO CORE FROM SECTOR 0				SYS09490
			0950	* OF THE SYSTEM CARTRIDGE AND TRANSFERRED TO BY				SYS09500
			0951	* THE COLD START CARD. DEFECTIVE CYLINDER				SYS09510
			0952	* ADDRESSES, CARTRIDGE ID AND DISKZ ARE ALSO ON				SYS09520
			0953	* SECTOR 0 AND ARE READ IN AT THE SAME TIME.				SYS09530
			0954	* ALL THAT REMAINS FOR THE COLD START PROGRAM IS				SYS09540
			0955	* TO READ IN THE RESIDENT IMAGE, SAVE THE				SYS09550
			0956	* CARTRIDGE ID AND TRANSFER TO THE AUXILIARY				SYS09560
			0957	* SUPERVISOR THROUGH \$DUMP IN THE RESIDENT				SYS09570
			0958	* MONITOR.				SYS09580
			0959	*				SYS09590
			0960	*ENTRY - CRO10-2				SYS09600
			0961	* ENTER PROGRAM BY TRANSFER FROM COLD START CARD				SYS09610
			0962	*				SYS09620
			0963	*INPUT -				SYS09630
			0964	* THE CARTRIDGE ID OF LOGICAL DRIVE ZERO (THE				SYS09640
			0965	* SYSTEM CARTRIDGE) IS READ IN FROM SECTOR 0				SYS09650
			0966	* WITH THE COLD START PROGRAM.				SYS09660
			0967	*				SYS09670
			0968	*OUTPUT -				SYS09680
			0969	* * THE RESIDENT IMAGE IS READ INTO CORE FROM				SYS09690
			0970	* THE DISK.				SYS09700
			0971	* * IN COMMA-				SYS09710
			0972	* \$ACDE				SYS09720
			0973	* \$CIBA-1				SYS09730
			0974	* \$CIDN				SYS09740
			0975	* \$CYLN				SYS09750
			0976	* \$DBSY				SYS09760
			0977	* \$IOCT				SYS09770
			0978	*				SYS09780
			0979	*EXTERNAL REFERENCES -				SYS09790
			0980	* DZ000 SUBROUTINE TO PERFORM DISK I/O.				SYS09800
			0981	*				SYS09810
			0982	*EXITS -				SYS09820
			0983	* THE ONLY EXIT IS TO THE AUXILIARY SUPERVISOR				SYS09830
			0984	* AS FOLLOWS-				SYS09840
			0985	* BSI \$DUMP				SYS09850
			0986	* DC -1				SYS09860
			0987	*				SYS09870
			0988	*TABLES/WORK AREAS - N/A				SYS09880
			0989	*				SYS09890
			0990	*ATTRIBUTES -				SYS09900
			0991	* THIS PROGRAM IS NOT NATURALLY RELOCATABLE.				SYS09910
			0992	*				SYS09920
			0993	*NOTES -				SYS09930
			0994	* DISK ERRORS RESULT IN A WAIT AT \$PST2.				SYS09940
			0995	*****				SYS09950

ADDR	REL	OBJECT	ST.NO.	LABEL	OPCD	FT	OPERANDS	ID/SEQNO
			0997	*				SYS09970
			0998	* READ THE RESIDENT IMAGE INTO CORE				SYS09980
			0999	*				SYS09990
01E0	0	617F	1000	LDX	1	Y		SYS10000
01E1	0	C82E	1001	LDD		CR920	SET UP WORD COUNT AND SCTR	SYS10010
01E2	00	DC000004	1002	CR010	STD	L \$CIBA-1	*ADDR OF RESIDENT IMAGE	SYS10020
01E4	0	D125	1003	STO	1	\$CCYL-Y	*INITIALIZE DEF CYL NO. 1	SYS10030
01E5	0	C184	1004	LD	1	3-Y	FETCH LOG DRIVE 0 AREA CODE	SYS10040
01E6	0	D120	1005	STO	1	\$ACDE-Y	*AND STORE IT IN COMMA	SYS10050
01E7	0	D029	1006	STO		CR920+1	SAVE THE AREA CODE	SYS10060
01E8	0	C156	1007	LD	1	DZ000-2-27-Y	FETCH AND SAVE THE	SYS10070
01E9	0	D0F1	1008	STO		\$CIDN	*CARTRIDGE ID	SYS10080
01EA	0	C0F8	1009	LD		CR010+1	FETCH CORE ADDR OF RESIDENT	SYS10090
01EB	0	1890	1010	SRT		16	*IMAGE AND PUT IN EXTENSION	SYS10100
01EC	0	D16F	1011	STO	1	\$DBSY-Y	CLEAR DISK BUSY INDICATOR	SYS10110
01ED	0	D11B	1012	STO	1	\$CYLN-Y	INITIALIZE ARM POSITION	SYS10120
01EE	0	4173	1013	BSI	1	DZ000-Y	FETCH RESIDENT IMAGE	SYS10130
01EF	0	3000	1014	WAIT			WAIT OUT THE INTERRUPT	SYS10140
			1015	*				SYS10150
			1016	* INITIALIZE ITEMS IN COMMA				SYS10160
			1017	*				SYS10170
01F0	0	1810	1018	SRA		16		SYS10180
01F1	0	D1B3	1019	STO	1	\$IOCT-Y	CLEAR IOCS COUNTER	SYS10190
01F2	0	C81B	1020	LDD		CR910		SYS10200
01F3	0	D985	1021	STD	1	\$CIBA-1-Y	*FOR SAVING CORE ON THE CIB	SYS10210
01F4	0	C01C	1022	LD		CR920+1	FETCH AREA CODE	SYS10220
01F5	0	D120	1023	STO	1	\$ACDE-Y	RESET AREA CODE	SYS10230
01F6	0	C016	1024	LD		CR905	INITIALIZE WD ZERO TO BR TO	SYS10240
01F7	0	D181	1025	STO	1	0-Y	*DUMP ENTRY POINT PLUS 1	SYS10250
			1026	*				SYS10260
			1027	* TRANSFER TO THE AUXILIARY SUPERVISOR				SYS10270
			1028	* TO COMPLETE INITIALIZATION				SYS10280
			1029	*				SYS10290
01F8	0	41C0	1030	BSI	1	\$DUMP-Y	BR TO AUXILLIARY SUPERVISOR	SYS10300
01F9	0	FFFF	1031	DC		-1	*FOR JOB PROCESSING	SYS10310
			1032	*				SYS10320
01FA		0013	1033	BSS		19	PATCH AREA	SYS10330
			1034	*				SYS10340
			1035	* CONSTANTS AND WORK AREAS				SYS10350
			1036	*				SYS10360
020D	0	703F	1037	CR905	MDX	X \$DUMP+1-1	TO BE STORED IN LOCN ZERO	SYS10370
020E	0	0001	1038	CR910	DC	1	WD CNT,SCTR ADDR OF CAUSE	SYS10380
020F	0	0007	1039	DC		*HDNG	*HARMLESS WRITE TO DISK	SYS10390
0210	0	00E8	1040	CR920	DC	\$DBSY-\$CH12	WD CNT AND SCTR	SYS10400
0211	0	0002	1041	DC		*RIAD	*ADDR OF RESIDENT IMAGE	SYS10410
0212	0212		1042	END		*		SYS10420

CROSS-REFERENCE

SYMBOL	VALUE	REL	DEFN	REFERENCES
CR010	01E2	0	1002	1009
CR905	020D	0	1037	1024
CR910	020E	0	1038	1020
CR920	0210	0	1040	1001 1006 1022
DZ000	00F2	0	0586	0322 0343 0603 0622 0683 0746 0890 0891 0940 0941 1007 1013
DZ010	00F7	0	0593	0604 0617
DZ020	00F9	0	0595	0589
DZ030	00FF	0	0601	0639
DZ060	0104	0	0606	0626
DZ070	0106	0	0610	0791 0797 0823 0831
DZ100	0108	0	0615	0595 0596 0691
DZ180	010E	0	0621	0594 0610
DZ185	0116	0	0627	0804
DZ190	0110	0	0636	0696
DZ210	0146	0	0687	0811
DZ215	014C	0	0695	0767
DZ220	014E	0	0700	0770
DZ225	0152	0	0706	0773
DZ230	0157	0	0714	0605
DZ235	0159	0	0715	0600 0630
DZ240	015E	0	0719	0842
DZ250	016D	0	0734	0742
DZ280	016E	0	0735	0721 0727 0728 0729 0740
DZ300	0178	0	0746	0737
DZ330	017D	0	0750	0723
DZ340	018C	0	0764	0702
DZ350	0197	0	0775	0638 0725 0802
DZ380	01A4	0	0790	0710 0787
DZ400	01AF	0	0809	0777
DZ410	01B1	0	0811	0836

SYMBOL	VALUE	REL	DEFN	REFERENCES
DZ900	0122	0	0645	0653 0789
DZ901	C123	0	0646	0801
DZ902	C124	0	0647	0627 0717 0832 0839
DZ904	0126	0	0649	0611 0790 0796 0819
DZ905	0127	0	0650	0828
DZ906	C128	0	0651	0731 0818
DZ907	0129	0	0652	0757
DZ908	C12A	0	0653	0795
DZ909	C12B	0	0654	0754
DZ910	012C	0	0655	0623 0706 0752 0764 0784
DZ911	012D	0	0656	0762
DZ912	C12E	0	0657	0634 0718 0763 0809 0813 0834 0840
DZ913	012F	0	0658	0747
DZ914	C130	0	0659	0695
DZ915	C131	0	0660	0635
DZ916	0132	0	0661	0812
DZ920	0134	0	0663	0758
DZ925	C135	0	0664	0753
DZ930	0136	0	0665	0739
DZ935	0137	0	0666	0700
DZ940	C138	0	0667	0733
DZ945	0139	0	0668	0598 0756 0759 0829
DZ950	013A	0	0669	0761
DZ955	013B	0	0670	0748
DZ960	013C	0	0671	0726
DZ965	013D	0	0672	0722
DZ970	013E	0	0673	0724
DZ975	013F	0	0674	0760 0827
DZ980	0140	0	0675	0751
DZ985	0141	0	0676	0816
DZ990	C142	0	0677	0749 0755 0821
DZ995	0143	0	0678	0788
\$ACDE	009F	0	0428	0672 0673 1005 1023
\$ACEX	C01A	0	0885	0317
\$CCAD	0074	0	0355	
\$CH12	0006	0	0252	1040
\$CIBA	C005	0	0251	0309 1002 1021
\$CIDN	01DB	0	0849	1008
\$CILA	005A	0	0886	0341
\$CLSW	0018	0	0883	
\$COMN	C007	0	0253	
\$CORE	000E	0	0265	
\$CPTR	C07E	0	0365	
\$CTSW	C00F	0	0266	
\$CWCT	C072	0	0353	
\$CXRI	C017	0	0882	0318
\$CYLN	C09A	0	0418	0671 0673 1012
\$DABL	0011	0	0268	0333
\$DADR	C010	0	0267	
\$DBSY	C0EE	0	0530	0344 0587 0602 0629 0688 0889 1011 1040
\$DCDE	0077	0	0358	
\$DCYL	C0A4	0	0439	0671 1003
\$DDSW	C0DD	0	0521	0624 0765
\$DMPF	0019	0	0884	0320
\$DREQ	C012	0	0269	
\$DUMP	C03F	0	0314	0319 1030 1037
\$DZIN	C076	0	0357	
\$EXIT	C038	0	0300	
\$FLSH	0071	0	0351	
\$FPAD	C095	0	0409	
\$HASH	C014	0	0271	0880 0881 0882 0883 0884 0885
\$IBSY	C013	0	0270	
\$IBT2	C0B9	0	0887	
\$IBT4	C0D4	0	0888	
\$IOCT	C032	0	0291	0315 0331 0689 0714 1019
\$IREQ	C02C	0	0285	0505
\$I200	C0B3	0	0465	0259 0476
\$I205	C0B8	0	0471	0887
\$I210	C0BA	0	0472	0466 0467 0468
\$I290	C0C2	0	0477	0469 0475
\$I400	C0C4	0	0497	0261 0516
\$I403	C0D0	0	0507	0504
\$I405	C0D3	0	0511	0888
\$I410	C0D5	0	0512	0499 0500 0501
\$I490	C0DE	0	0522	0498 0515
\$I492	C0E0	0	0523	0502
\$I494	C0E2	0	0526	0508
\$KCSW	C07C	0	0363	
\$LAST	C033	0	0292	
\$LEVO	C008	0	0257	
\$LEV1	C009	0	0258	
\$LEV2	C00A	0	0259	
\$LEV3	C00B	0	0260	
\$LEV4	C00C	0	0261	
\$LEV5	C00D	0	0262	
\$LINK	C039	0	0304	0334

SYMBOL	VALUE	REL	DEFN	REFERENCES
\$LKNM	0014	0	0880	0338
\$LSAD	0075	0	0356	
\$NDUP	0034	0	0293	
\$NXEQ	0035	0	0294	
\$PAUS	00F0	0	0890	
\$PBSY	0036	0	0295	
\$PGCT	0037	0	0296	
\$PHSE	0078	0	0359	
\$PRET	0028	0	0280	0282 0701
\$PST1	0081	0	0371	0373
\$PST2	0085	0	0377	0379 0636
\$PST3	0089	0	0383	0385
\$PST4	008D	0	0389	0391
\$RMSW	0016	0	0881	0330
\$RWCZ	00F1	0	0891	
\$SCAN	0020	0	0274	
\$SNLT	00EF	0	0889	
\$STOP	0091	0	0396	0262 0398
\$SYSC	00E0	0	0524	
\$S000	004D	0	0329	0300
\$S100	004F	0	0330	0306 0324
\$S150	0059	0	0340	0886
\$S900	003C	0	0308	0321 0323
\$S910	003E	0	0310	0329
\$UFDR	007D	0	0364	
\$UFIO	0079	0	0360	
\$ULET	002D	0	0286	
\$WRD1	0078	0	0362	
\$WSDR	007A	0	0361	
\$ZEND	01E0	0	0853	0580 0582
\$1132	007F	0	0366	
\$1403	0080	0	0367	
X2	00F2	0	0683	0636 0747 0748 0749 0751 0752 0753 0754 0755 0756 0757 0758 0759 0760 0761 0762 0763 0764 0765 0784 0788 0789 0790 0791 0795 0796 0797 0801 0809 0812 0813 0816 0818 0819 0821 0823 0827 0828 0829 0831 0832 0834 0839 0840
Y	007F	0	0942	1000 1003 1004 1005 1007 1011 1012 1013 1019 1021 1023 1025 1030
*BLCT	0002	0	0931	
*CDCV	0093	0	0869	
*CIL1	00A0	0	0875	0848
*CIL2	00A1	0	0876	
*CLB0	0078	0	0861	
*CMON	0001	0	0906	
*COMZ	0380	0	0936	
*COM1	04C0	0	0937	
*COM2	0600	0	0938	
*CORE	001C	0	0919	
*CPTR	008E	0	0864	
*DCOM	0001	0	0896	
*DKEP	00F9	0	0940	
*DKIP	00F7	0	0941	
*DNID	0098	0	0874	
*DREQ	0002	0	0907	
*DZID	0096	0	0872	0581
*D1ID	0097	0	0873	
*FILE	0003	0	0908	
*HDNG	0007	0	0900	1039
*HEND	001D	0	0920	
*HWCT	0004	0	0909	
*IDAD	0000	0	0895	
*ILS4	0011	0	0917	
*ITVX	0008	0	0916	
*KBCP	0092	0	0868	
*KBCV	0095	0	0871	
*LDAD	0006	0	0911	
*LFEN	0003	0	0925	
*LFHD	0005	0	0924	
*LFNM	0000	0	0930	
*LSCT	0005	0	0910	
*MXDR	0005	0	0935	
*NEXT	0004	0	0929	
*OVSW	001A	0	0918	
*PTCV	0094	0	0870	
*RIAD	0002	0	0897	1041
*RTBL	0006	0	0899	
*SCTN	0000	0	0926	
*SLET	0003	0	0898	
*STRT	0000	0	0901	
*SUP1	006E	0	0858	
*SUP3	0070	0	0859	
*SUP4	0071	0	0860	
*TCNT	0011	0	0939	0601
*TVWC	0008	0	0913	
*UAFX	0001	0	0927	
*NCNT	0009	0	0914	
*WDSA	0003	0	0928	
*XCTL	0007	0	0912	
*XEQA	0000	0	0905	
*XR3X	000A	0	0915	
*1132	008D	0	0863	
*1134	0091	0	0867	
*1403	008C	0	0862	
*1442	0090	0	0866	
*2501	008F	0	0865	

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

APPENDIX C. ABBREVIATIONS

Below is a list of the abbreviations used in the listings of the 1130 Disk Monitor System, Version 2. Included in this list are the abbreviations used in current 1130 and 1800 systems.

<u>Abbreviation</u>	<u>Meaning</u>
ABS	Absolute
ACC	Accumulator, Accumulate
ACCT	Account
ACT	Actual
ADDL	Additional
ADDR	Address
ADJ	Adjust
ADV	Advance
AI	Analog Input
ALG	Algebraic
ALLOC	Allocate
ALLOCN	Allocation
ALPHA	Alphabetic
ALT	Alternate, Alteration
AO	Analog Output
APPDGE	Appendage
APPROX	Approximate
ARITH	Arithmetic
ASDNG	Ascending
ASM	Assembler
ASMBL	Assemble
ASGN	Assign
AUX	Auxiliary
AVAIL	Availability
AVG	Average
BEGNG	Beginning
BFR	Buffer
BKSP	Backspace
BLK	Block
BLKCNT	Block Count
BLNK	Blank
BR	Branch
BM	Buffer Mark
CAD	Core Address
CALC	Calculate, Calculator
CAR	Channel Address Register
CARR	Carriage
CART	Cartridge
CAT	Catalog
CATLGD	Cataloged

<u>Abbreviation</u>	<u>Meaning</u>
CC	Card Column
CD	Card
CDE	Code
CHAN	Channel
CHAR	Character
CHK	Check
CHG	Change
CHKPT	Checkpoint
CIB	Core Image Buffer
CIL	Core Image Loader
CLB	Core Load Builder
CLD	Core Load
CLR	Clear
CLS	Close
CMN	COMMON
CMP	Compare
CMPL	Complement
COMMA	Communication Area
COMP	Compute
CNSL	Console
CNT	Count
COL	Column
COMM	Communication
CON	Constant
COND	Condition
CONT	Continue
CORR	Correction
CP	Console Printer, Control Parameter
CPLD	Coupled
CPTN	Computation
CTR	Counter
CTRL	Control
CURR	Current
CVRT	Convert
CYL	Cylinder
DAO	Digital-Analog Output
DB	Disk Block
DC	Data Channel
DCMT	Document
DEC	Decision
DECML	Decimal
DECR	Decrement
DEF	Defective
DEFN	Define
DEL	Delete
DESCG	Descending

<u>Abbreviation</u>	<u>Meaning</u>	<u>Abbreviation</u>	<u>Meaning</u>
DETM	Determine	FREQ	Frequency
DEVC	Device	FUNC	Function
DGT	Digit	FWD	Forward
DI	Digital Input	FXA	Fixed Area
DICT	Dictionary	FXD	Fixed
DIM	Dimension		
DIRCTY	Directory	GEN	Generator
DISP	Displacement	GENL	General
DISPCHG	Dispatching	GM	Group Mark
DK	Disk	GT	Greater Than
DLMTER	Delimiter	GTE	Greater Than or Equal To
DPC	Direct Program Control		
DR	Drive	HDLER	Handler
DSW	Device Status Word	HDR	Header
DT	Defective Track	HEX	Hexadecimal
DUP	Disk Utility Program	HI	High
DUPCTN	Duplication	HLT	Halt
		HSK	Housekeeping
EBC	EBCDIC	HYPHER	Hypertape
ELIM	Eliminate		
ELT	Element	IAR	Instruction Address Register
ENT	Entry	IC	Instruction Counter
EOF	End Of File	ID	Identification
EOJ	End Of Job	IDX	Index
EOR	End Of Reel	ILS	Interrupt Level Subroutine
EP	Extended Precision	ILSW	Interrupt Level Status Word
EQ	Equal, Equate	INCR	Increment
EQU	Equate	IND	Indicate
ERP	Error Parameter	INDN	Indication
ERR	Error	INDR	Indicator
ES	Electronic Switch	INFO	Information
ETV	Executive Transfer Vector	INITLZ	Initialize
EVAL	Evaluate	INQ	Inquire
EXCH	Exchange	INT	Initial
EXEC	Execute	INTFCE	Interface
EXP	Exponent	INTLD	Interlude
EXPR	Expression	INTM	Interim
EXTYP	Exit Type	INTMD	Intermediate
EXTR	Extract	INTNL	Internal
		INTRPT	Interrupt
FAC	Floating Accumulator	I/O	Input/Output
FOR	FORTTRAN	IOAP	I/O Area Parameter
FIO	FORTTRAN I/O	IOCC	I/O Control Command
FLD	Field	IOCR	I/O Control Routine
FLDL	Field Length	INST	Instruction
FIG	Figure	INTERP	Interpret
FLET	The Location Equivalence Table for the Fixed Area	INVAL	Invalid
FLT	Floating	ISS	Interrupt Service Subroutine
FMT	Format	ITER	Iterate, Iteration
FR	From	ITG	Integer
		I/P	Input

<u>Abbreviation</u>	<u>Meaning</u>	<u>Abbreviation</u>	<u>Meaning</u>
KB	Keyboard	OPN	Open
KP	Keypunch	OPND	Operand
LBL	Label	OPTN	Option
LCT	List Control Table	OPTR	Operator
LD	Load	ORG	Origin
LET	Location Equivalence Table for User Area	OVFLO	Overflow
LFT	Left	OVLP	Overlap
LH	Left-Hand, Leftmost	OVLY	Overlay
LINKB	Link/Busy Word	PAPT	Paper Tape
LIT	Literal	PARAM	Parameter
LN	Line	PARTL	Partial
LNG	Length	PERF	Perforate, Perforated, Perforation
LO	Low	PERPHL	Peripheral
LOC	Location	PFM	Perform
LOCAL	Load-on-Call Subroutine	PG	Page
LT	Less Than	PGLIN	Page and Line
LTE	Less Than or Equal To	PH	Phase
LTR	Letter	PHYS	Physical
LVL	Level	PK	Pack
MACH	Machine	PKD	Packed
MAGT	Magnetic Tape	PNCH	Punch
MAINT	Maintain, Maintenance	PNDG	Pending
MALF	Malfunction	POS	Position
MAX	Maximum	PR	Print
MEM	Memory	PREC	Precision
MIN	Minimum	PREV	Previous
MISC	Miscellaneous	PRGE	Purge
ML	Mainline	PRI	Priority
MN	Mnemonic	PRINC	Principal
MOD	Modification	PROC	Process
MON	Monitor	PROG	Program
MPXR	Multiplexor	PROT	Protect
MPY	Multiply	PRTN	Partition
MRGE	Merge	PRVNT	Prevent
MSG	Message	PT	Pointer, Point
MSTR	Master	PTR	Printer
NEC	Necessary	PTV	Positive
NEG	Negative	QUALFD	Qualified
NO.	Number	QUANT	Quantity
NORM	Normalize, Normalized	QUE	Queue
NUM	Numeric	RAND	Random
NXT	Next	R + S	Reset and Start
OBJ	Object	R/W	Read/Write
OP	Operation	RCD	Record
O/P	Output	RCV	Receive
		RD	Read
		RDY	Ready

<u>Abbreviation</u>	<u>Meaning</u>	<u>Abbreviation</u>	<u>Meaning</u>
REF	Reference	SUB	Subtract
REG	Register	SUBP	Subprogram
REL	Release	SUBR	Subroutine
RELOC	Relocate, Relocatable	SUBSC	Subscript
REQ	Request, Require	SUMM	Summarize
RET	Return	SUP	Suppress
RH	Right-Hand, Rightmost	SUP	Supervisor
RI	Read in	SYNC	Synchronize, Synchronizer
RLS	Reels	SYM	Symbol
RLTV	Relative	YSR _x	System Reserved Word (x is a digit)
RM	Record Mark	SYST	System
RO	Read Out	SW	Switch
RPT	Report		
RSLT	Result	TBL	Table
RST	Reset	TECHNQE	Technique
RSTRT	Restart	TEMP	Temporary
RT	Right	TERM	Terminal, Terminate
RTE	Route	TM	Tapemark
RTN	Routine	TMN	Transmission
RWD	Rewind	TMT	Transmit
		TOT	Total
SAD	Sector Address	TP	Tape
SAT	Satellite	TR	Transfer
SAR	System Action Required	TRK	Track
SCHED	Schedule, Scheduler	TRLR	Trailer
SCN	Scan	TRUNC	Truncate, Truncation
SCTR	Sector	TST	Test
SECT	Section	TU	Tape Unit
SEL	Select	TV	Transfer Vector
SEN	Sense	TW	Typewriter
SEQ	Sequence		
SEQNO	Sequence Number	UA	User Area
SER	Serial	UAR	User Action Required
SEG	Segment	UFLO	Underflow
SIG	Signal	UNC	Unconditional
SIM	Simulator	UNLD	Unload
SK	Skeleton	UNPKD	Unpacked
SLET	System Location Equivalence Table	UTIL	Utility
SM	Storage Mask		
SNGL	Single	V	Version
SOCAL	System Load-On-Call Subroutine	VAL	Value
SP	Space	VAR	Variable
SRCH	Search	VIOL	Violation
SPEC	Specification, Specify	VOL	Volume
ST	Store		
STA	Station	WD	Word
STD	Standard	WM	Word Mark
STG	Storage, Storing	WR	Write
STMNT	Statement	WRK	Work
STP	Standard Precision	WS	Working Storage

<u>Abbreviation</u>	<u>Meaning</u>	<u>Abbreviation</u>	<u>Meaning</u>
W/	With	1st	First
W/O	Without	2nd	Second
		3rd	Third
XPL	Explain, Explanation	4th	Fourth
XR1	Index Register 1	5th	Fifth
XR2	Index Register 2	6th	Sixth
XR3	Index Register 3	7th	Seventh
XTR	Extra	8th	Eighth
		9th	Ninth
Z	Zero		
ZN	Zone		

APPENDIX D. MICROFICHE REFERENCE TABLE

<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>	<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>
System Loader (Paper Tape)		Phase 17	KAA.017.00
Bootstrap	BAA.001.00	Phase 18	KAA.018.00
Phase 1	BAA.002.00	Phase 19	KAA.019.00
Phase 2	BAA.003.00	Phase 20	KAA.020.00
System Loader (Card)		Phase 21	KAA.021.00
Bootstrap	BAA.004.00	Phase 22	KAA.022.00
Core Image Loader	BAA.005.00	Phase 23	KAA.023.00
Phase 1	BAA.006.00	Phase 24	KAA.024.00
Phase 2	BAA.007.00	Phase 25	KAA.025.00
		Phase 26	KAA.026.00
		Phase 27	KAA.027.00
Disk Utility Program		Assembler Program	
DUPCO	JAA.001.00	Phase 0	MAA.001.00
DCTL	JAA.002.00	Phase 1	MAA.007.00
STORE	JAA.003.00	Phase 1A	MAA.008.00
FILEQ	JAA.004.00	Phase 2	MAA.012.00
DDUMP	JAA.005.00	Phase 2A	MAA.013.00
DMPLT	JAA.006.00	Phase 3	MAA.010.00
DELET	JAA.007.00	Phase 4	MAA.011.00
DFINE	JAA.008.00	Phase 5	MAA.015.00
DEXIT	JAA.009.00	Phase 6	MAA.016.00
CDFAC	JAA.010.00	Phase 7	MAA.017.00
KBFAC	JAA.011.00	Phase 7A	MAA.018.00
PTFAC	JAA.012.00	Phase 8	MAA.019.00
PRECI	JAA.013.00	Phase 8A	MAA.020.00
		Phase 9	MAA.014.00
FORTRAN Compiler		Phase 10	MAA.003.00
Phase 1	KAA.001.00	Phase 10A	MAA.009.00
Phase 2	KAA.002.00	Phase 11	MAA.004.00
Phase 3	KAA.003.00	Phase 12	MAA.005.00
Phase 4	KAA.004.00	Error Message Phase	MAA.006.00
Phase 5	KAA.005.00	Punch Conversion	
Phase 6	KAA.006.00	Phase	MAA.021.00
Phase 7	KAA.007.00	Read Conversion Phase	MAA.002.00
Phase 8	KAA.008.00		
Phase 9	KAA.009.00	Supervisor	
Phase 10	KAA.010.00	Monitor Control Record	
Phase 11	KAA.011.00	Analyzer	NAA.001.00
Phase 12	KAA.012.00	System Core Dump	
Phase 13	KAA.013.00	Program	NAA.002.00
Phase 14	KAA.014.00	Auxiliary Supervisor	NAA.003.00
Phase 15	KAA.015.00		
Phase 16	KAA.016.00	Core Load Builder	OAA.001.00

<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>	<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>
System Device Subroutines	PAA. 001. 00	FIXI	SAA. 005. 00
Standard Precision Arithmetic and Function Subroutines		FLOAT	SAA. 005. 00
FADD	RAA. 001. 00	IFIX	SAA. 005. 00
FAXI	RAA. 001. 00	NORM	SAA. 005. 00
FDIV	RAA. 001. 00	SNR	SAA. 005. 00
FDVR	RAA. 001. 00	XDD	SAA. 006. 00
FGETP	RAA. 001. 00	XMD	SAA. 006. 00
FLD	RAA. 002. 00	XMDS	SAA. 006. 00
FMPY	RAA. 002. 00		
FSBR	RAA. 002. 00	FORTTRAN Common Subroutines (No Precision)	
FABS	RAA. 002. 00	FBTD	TAA. 001. 00
FATN	RAA. 002. 00	IABS	TAA. 001. 00
FAXB	RAA. 003. 00	XSQR	TAA. 001. 00
FEXP	RAA. 003. 00	PAUSE	TAA. 001. 00
FLN	RAA. 003. 00	STOP	TAA. 002. 00
FSIGN	RAA. 003. 00	SUBIN	TAA. 002. 00
FSIN	RAA. 003. 00	SUBSC	TAA. 002. 00
FSQR	RAA. 004. 00	TTEST	TAA. 002. 00
FTANH	RAA. 004. 00	DATSW	TAA. 002. 00
SFAR	RAA. 004. 00	DVCHK	TAA. 003. 00
SFIF	RAA. 004. 00	FCTST	TAA. 003. 00
Extended Precision Arithmetic and Function Subroutines		ISIGN	TAA. 003. 00
EADD	SAA. 001. 00	OVERF	TAA. 003. 00
EAXI	SAA. 001. 00	PDUMP	TAA. 003. 00
EDIV	SAA. 001. 00	SLITE	TAA. 004. 00
EDVR	SAA. 001. 00	TSTOP	TAA. 004. 00
EGETP	SAA. 001. 00	TSTRT	TAA. 004. 00
ELD	SAA. 002. 00		
EMPY	SAA. 002. 00	FORTTRAN Trace Subroutines	
ESBR	SAA. 002. 00	SGOTO	TAA. 009. 00
EABS	SAA. 002. 00	SIAR	TAA. 009. 00
EATN	SAA. 002. 00	SIIF	TAA. 009. 00
EAXB	SAA. 003. 00		
EEXP	SAA. 003. 00	FORTTRAN Conversion Subroutines and Tables	
ELN	SAA. 003. 00	EBCTB	TAA. 006. 00
ESIGN	SAA. 003. 00	GETAD	TAA. 006. 00
ESIN	SAA. 003. 00	HOLEZ	TAA. 007. 00
ESQR	SAA. 004. 00	HOLTB	TAA. 007. 00
ETANH	SAA. 004. 00		
SEAR	SAA. 004. 00	FORTTRAN I/O Subroutines	
SEIF	SAA. 004. 00	SDFIO	TAA. 004. 00
FARC	SAA. 004. 00	SDFND	TAA. 005. 00
		SFIO	TAA. 005. 00
		UFIO	TAA. 006. 00

<u>Disk Monitor</u> <u>System Component</u>	<u>Microfiche</u> <u>Identification</u>	<u>Disk Monitor</u> <u>System Component</u>	<u>Microfiche</u> <u>Identification</u>
FORTTRAN Device		ZIPCO	UAA.007.00
Subroutines		BIDEC	UAA.007.00
CARDZ	TAA.006.00	BINDC	UAA.007.00
PAPTZ	TAA.007.00	BINHX	UAA.007.00
PNCHZ	TAA.007.00	CPEBC	UAA.008.00
PRNTZ	TAA.008.00	CPHOL	UAA.008.00
PRNZ	TAA.008.00	CPPT3	UAA.008.00
READZ	TAA.008.00	DCBIN	UAA.008.00
TYPEZ	TAA.008.00	DECBI	UAA.008.00
WRTYZ	TAA.008.00	EBCCP	UAA.009.00
		EBHOL	UAA.009.00
Interrupt Level		EBPA	UAA.009.00
Subroutines (ILSs)		EBPT3	UAA.009.00
ILS00	UAA.001.00	HLEBC	UAA.009.00
ILS01	UAA.001.00	HLPT3	UAA.010.00
ILS02	UAA.001.00	HOLCP	UAA.010.00
ILS03	UAA.001.00	HOLL	UAA.010.00
ILS04	UAA.001.00	PRTY	UAA.010.00
		PT3CP	UAA.010.00
Interrupt Service		PT3EB	UAA.011.00
Subroutines (ISSs)		PTHOL	UAA.011.00
CARD0	UAA.002.00		
CARD1	UAA.002.00	Utility Dump Subroutines	
OMPR1	UAA.002.00	DMP80	UAA.011.00
PAPT1	UAA.002.00	DMTD0	UAA.011.00
PAPTN	UAA.002.00	DMPD1	UAA.012.00
PAPTX	UAA.003.00		
PLOT1	UAA.003.00	System Subroutines	
PNCH0	UAA.003.00	SYSUP	UAA.013.00
PNCH1	UAA.003.00	FLIPR	UAA.012.00
PRNT1	UAA.004.00		
PRNT3	UAA.004.00	Mainline Programs	
READ0	UAA.004.00	ADRWS	UAA.014.00
READ1	UAA.004.00	COPY	UAA.014.00
TYPE0	UAA.005.00	DISC	UAA.015.00
WRTY0	UAA.005.00	DLCIB	UAA.016.00
		DSLET	UAA.016.00
Conversion Subroutines		IDENT	UAA.017.00
and Tables		ID	UAA.017.00
EBPRT	UAA.005.00	MODIF	UAA.018.00
HOLEB	UAA.005.00	PTUTL	UAA.019.00
HOLPR	UAA.006.00	CALPR	UAA.019.00
HXBIN	UAA.006.00	FSLEN	UAA.019.00
PAPEB	UAA.006.00	RDREC	UAA.020.00
PAPHL	UAA.006.00		
PAPPR	UAA.006.00	Plotter Subroutines	
SPEED	UAA.007.00	ECHAR	VAA.001.00

<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>	<u>Disk Monitor System Component</u>	<u>Microfiche Identification</u>
ECHRX	VAA.001.00	PRNT2	WAA.002.00
EGRID	VAA.001.00	SCAT1	WAA.003.00
EPLOT	VAA.001.00	STRTB	WAA.004.00
ERULE	VAA.002.00		
FCHAR	VAA.002.00	Stand-alone Programs	
FCHRX	VAA.002.00	Cold Start Card	ZAA.001.00
FGRID	VAA.002.00	Console Printer	
FPLOT	VAA.003.00	Core Dump	
FRULE	VAA.003.00	(Paper Tape)	ZAA.003.00
PLOTI	VAA.003.00	Console Printer	
PLOTX	VAA.003.00	Core Dump	
POINT	VAA.003.00	(Card)	ZAA.002.00
SCALE	VAA.004.00	DCIP	ZAA.004.00
SCALF	VAA.004.00	1132/1403 Printer	
XYPLT	VAA.004.00	Core Dump	
		(Paper Tape)	
SCA Interrupt Service		Phase 1	ZAA.007.00
Subroutines (ISSs),		Phase 2	ZAA.008.00
Conversion Subroutines,		1132/1403 Printer	
and Conversion Tables		Core Dump (Card)	
EBC48	WAA.001.00	Phase 1	ZAA.005.00
HOL48	WAA.001.00	Phase 2	ZAA.006.00
HOLCA	WAA.001.00	PTREP	ZAA.009.00
HXCV	WAA.001.00	UCART	ZAA.010.00

- Assembler program 65
 - Communications area 66
 - Core layout 71
 - Double-buffering 67
 - Error message phase 71
 - Flowcharts 220
 - General 65
 - Intermediate I/O 67
 - Introduction 2
 - Overlay area 66
 - Phase 0 67
 - Phase 1 67
 - Phase 1A 68
 - Phase 2 68
 - Phase 2A 68
 - Phase 3 68
 - Phase 4 68
 - Phase 5 68
 - Phase 6 69
 - Phase 7 69
 - Phase 7A 69
 - Phase 8 69
 - Phase 8A 69
 - Phase 9 69
 - Phase 10 70
 - Phase 10A 71
 - Phase 11 71
 - Phase 12 71
 - Program operation 65
 - Punch conversion phase 71
 - Read conversion phase 71
 - Symbol table 66
- Auxiliary supervisor 23
- Cartridge-dependent parameters
 - COMMA 3
 - DCOM 3
- CIB (see core image buffer)
- Cold start loader 15
- Cold start program 15
- Cold start programs 15
 - Cold start loader 15
 - Cold start program 15
- Core layout 15
- Flowcharts 191
- Introduction 1
- COMMA (in-core communication area) 3
 - (see also resident monitor)
- Communications areas 3
 - Cartridge-dependent parameters 3
 - Disk-resident (DCOM) 3
 - Drive-dependent parameters 3
 - Introduction 1
 - In-core (COMMA) 3
- Core image buffer (CIB)
 - Core load builder 34
- Core image loader 27
 - Core layout 28
 - Debugging/analysis aids 30
 - Flowcharts 203
 - Introduction 1
 - Phase 1 27
 - Phase 2 27
 - Special techniques 27, 28
- Core load builder 33
 - Core image buffer (CIB) 34
 - Core layout 33
 - Debugging/analysis aids 42
 - DEFINE FILE table 39
 - Disk buffers 34
 - FILES information 35
 - Flowcharts 205
 - General comments 33
 - Interrupt branch table (IBT) 35
 - Interrupt level subroutines (ILSs) 37
 - Introduction 2
 - ISS table 35
 - Linkage to LOCALs 37
 - Linkage to SOCALs 38
 - Load table 35
 - LOCAL information 35
 - LOCALs 37
 - NOCAL information 35
 - Overlay scheme 33
 - Pass 1 36
 - Pass 2 36
 - Phase 0 40
 - Phase 1 40
 - Phase 2 40
 - Phase 3 41
 - Phase 4 41
 - Phase 5 41
 - Phase 6 42
 - Phase 7 42
 - Phase 8 42
 - Phase 9 42
 - Phase 10 42
 - Phase 11 42
 - Phase 12 42
 - SOCALs 37
 - Transfer vector (TV) 37
- DCIP 111
- DCOM 3
- DEFINE FILE table (DFT)
 - Core load builder 39
- DFT (see DEFINE FILE table)
- Disk cartridge initialization program (DCIP) 111

Disk I/O subroutine
 Resident monitor 17
 Subroutine data charts 125
 System library 102
 Disk utility program (DUP) 43
 CFACE phase 59
 Communications area (CATCO) 45
 Control records 44
 Core layout 43
 DCTL phase 51
 DDUMP phase 55
 DEFINE phase 57
 DELETE phase 57
 DEXIT phase 58
 Diagnostic aids 63
 DUMPLET/DUMPFLET phase 56
 DUPCO phase 49
 FILEQ phase 54
 Fixed location equivalence table (FLET) 44
 Flowcharts 207
 Introduction 2
 KFACE phase 60
 Location equivalence table (LET) 44
 PFACE phase 60
 PRECI phase 62
 STORE phase 52
 Disk-resident communications area (DCOM) 3
 Drive-dependent parameters
 COMMA 3
 DCOM 3
 DUP (see disk utility program)

 FILES control record processing
 Disk utility program 54
 Supervisor 22
 FILES information in SCRA
 Disk utility program 55
 Supervisor 22
 Fixed location equivalence table (FLET) 44
 FLET 44
 Flowcharts 185
 Assembler program 220
 Cold start programs 191
 Core image loader 203
 Core load builder 205
 Disk utility program 207
 FORTRAN compiler 244
 Supervisor 193
 System library 274
 System loader 186
 FORTRAN compiler 73
 Communications area 76
 Compilation errors 81
 Compiler I/O 81
 Core layout 75
 Flowcharts 244
 General 73
 Introduction 2

 Phase area 77
 Phase objectives 73
 Phase 1 82
 Phase 2 82
 Phase 3 83
 Phase 4 83
 Phase 5 84
 Phase 6 84
 Phase 7 85
 Phase 8 85
 Phase 9 86
 Phase 10 86
 Phase 11 87
 Phase 12 88
 Phase 13 89
 Phase 14 89
 Phase 15 90
 Phase 16 91
 Phase 17 92
 Phase 18 93
 Phase 19 94
 Phase 20 94
 Phase 21 95
 Phase 22 95
 Phase 23 95
 Phase 24 96
 Phase 25 96
 Phase 26 96
 Phase 27 96
 Statement string 80
 String area 77
 Symbol table 78

 IBT (see interrupt branch table)
 ILSs (see interrupt level subroutines)
 Interrupt branch table (IBT)
 Core load builder 35
 Interrupt level subroutines (ILSs)
 Core load builder 37
 Skeleton supervisor 17
 System library 105
 Interrupt service subroutines (ISSs)
 Subroutine data charts 152
 System library 102
 Introduction 1
 ISSs (see interrupt service subroutines)

 LET 44
 LOCAL control record processing
 Disk utility program 54
 Supervisor 22
 LOCAL information in SCRA
 Disk utility program 55
 Supervisor 22
 LOCAL linkage in TV 37
 LOCALs 37
 Location equivalence table (LET) 44

MCRA (see monitor control record analyzer)
 Monitor control record analyzer (MCRA) 19
 JOB control record processing 19
 Other control record processing 20
 System update program 20

 NOCAL control record processing
 Disk utility program 54
 Supervisor 22
 NOCAL information in SCRA
 Disk utility program 55
 Supervisor 22

 Procedures
 Core dump 115
 Core location 115
 Generalized subroutine maintenance/analysis 120
 Identification of the failing component or function 113
 Program analysis 113
 Subroutine looping 120
 Trace back 120
 Program analysis procedures 113
 Core block diagrams 115
 Core dump procedures 115
 Core location procedures 115
 Generalized subroutine maintenance/analysis procedure 120
 Identification of the failing component or function 113
 Introduction 113
 Subroutine data charts 125
 Subroutine error number list 115
 Subroutine error stop list 115
 Subroutine looping capabilities 120
 Summary 113
 Trace back procedure 120

 Reload table
 System loader 14
 Resident monitor 17
 COMMA 17
 Disk I/O subroutine 17
 Flowcharts 193
 Introduction 1
 Skeleton supervisor 17

 SCRA (see supervisor control record area)
 Skeleton supervisor 17
 CALL LINK, CALL EXIT, CALL DUMP processor 17
 Error traps 17
 ILSs 17
 SLET
 System loader 13
 SOCAL linkage in TV 38
 SOCALs 37
 Stand-alone utilities 111
 DCIP 111
 Introduction 2
 UCART 112

 Subroutine data charts 125
 CARDZ 140
 CARD0 152
 CARD1 154
 DISKN 180
 DISKZ 138
 DISK1 177
 OMPR1 174
 PAPTN 170
 PAPTZ 150
 PAPT1 168
 PLOT1 172
 PNCHZ 142
 PNCH0 158
 PNCH1 160
 PRNTZ 148
 PRNT1 164
 PRNT3 166
 PRNZ 147
 READZ 144
 READ0 156
 READ1 157
 System device subroutine for console printer 130
 System device subroutine for disk 138
 System device subroutine for keyboard/console printer 125
 System device subroutine for 1132 132
 System device subroutine for 1134/1055 136
 System device subroutine for 1403 134
 System device subroutine for 1442/1442 126
 System device subroutine for 2501/1442 128
 TYPEZ 145
 TYPE0 162
 WRTYZ 146
 WRTY0 176
 Subroutine error numbers 115
 Subroutine error stops 115
 Supervisor 19
 Auxiliary supervisor 23
 Core layout 24
 Introduction 1
 Monitor control record analyzer (MCRA) 19
 XEQ control record processor 21
 System core dump program 23
 Supervisor control record area (SCRA)
 Disk utility program 55
 Supervisor 22
 Supervisor control record processing
 Disk utility program 54
 Supervisor 22
 Symbol table
 Assembler program 66
 FORTRAN compiler 78
 System core dump program 23
 System device subroutines 109
 Introduction 2
 Subroutine data charts 125
 System library 99
 Arithmetic and function subroutines, common 100, 102

- Arithmetic and function subroutines, extended precision 99, 101
- Arithmetic and function subroutines, standard precision 100, 101
- Conversion subroutines 103
- Conversion tables 103
- Flowcharts 274
- FORTTRAN common 99, 101
- FORTTRAN conversion 101
- FORTTRAN I/O 100
- FORTTRAN sign transfer 99
- FORTTRAN trace 100
- Interrupt level subroutines (ILSs) 104, 105
- Interrupt service subroutines (ISSs) 102
- Introduction 2
- Mainline programs 104, 105
- Plotter subroutines 104
- SCA subroutines 103
- Utility subroutines 99
- System Loader 9
- Cartridge identification sector 13
- Core layout 12
- Flowcharts 186
- Introduction 1
- Phase 1 9
- Phase 2 10
- Reload table 14
- System location equivalence table (SLET) 13
- System location equivalence table (SLET)
 - System loader 13
- Transfer vector (TV)
 - Core load builder 37
- TV (see transfer vector)
- XEQ control record processor 21
 - Supervisor control record area (SCRA) 22
 - Supervisor control record processing 22
 - XEQ control record processing 21

READER'S COMMENT FORM

IBM 1130 Disk Monitor Programming System, Version 2
Program Logic Manual

Form Y26-3714-0

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

- | | Yes | No |
|--|---|--------------------------|
| • Does this publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the material: | | |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? _____ | | |
| • How do you use this publication? | | |
| As an introduction to the subject? <input type="checkbox"/> | As an instructor in a class? <input type="checkbox"/> | |
| For advanced knowledge of the subject? <input type="checkbox"/> | As a student in a class? <input type="checkbox"/> | |
| For information about operating procedures? <input type="checkbox"/> | As a reference manual? <input type="checkbox"/> | |
| Other _____ | | |
| • Please give specific page and line references with your comments when appropriate. | | |

COMMENTS

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

YOUR COMMENTS, PLEASE...

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold

fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114



Attention: Programming Publications, Dept. 232

fold

fold



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]