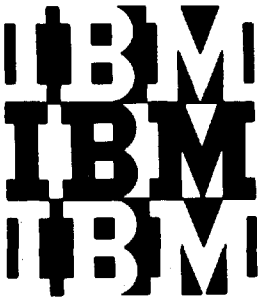
File Number 1130-36
Form Y-26-3714-2



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.

CONVENTIONS OBSERVED

The following conventions have been observed in this publication:

- Numbers written in the form /XXXX are hexadecimal numbers; numbers written without a preceding slash (/) are decimal numbers.
- 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.
- The use of absolute addresses has been avoided in this manual; symbolic addresses have been used instead. The absolute equivalents to these addresses may be found in the listing in Appendix B.

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)

Third Edition (Feb 1969)

This is a major revision of the previous edition of this manual (Y26-3714-1), which is now obsolete. The manual has been updated to agree with Modification 5 of the 1130 Disk Monitor System, Version 2. The new material includes information on changes and additions to the Monitor System, which give the programmer the ability to access the Graphic Subroutine Package to support the 1130/2250 system in a stand-alone environment. It also includes information on RPG and Disk Data File Conversion Program (DFCNV)
Text changes are indicated by a vertical line to the left of the affected text.
Figure changes are indicated by a bullet to the left of the figure caption.
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 Nordic Laboratory, Technical Communications, Box 962, Lidingö 9, Sweden.

Figures

Tables

Flowdiagrams

Flowcharts

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:

- Make available the contents of disk storage in punched or printed format -- DUMP, DUMPDATA.
- Print a map of the contents of the variable portions of disk storage -- DUMPLET, DUMPFLET.
- Store information on the disk in disk system format (DSF), disk data format (DDF), or disk core image format (DCI) -- STORE, STOREDATA, STOREDATACI, STORECI, STOREMOD.
- Remove information from the User/Fixed Area -- DELETE.
- 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.
- Initialize the Working Storage area on disk -- DWADR.
- 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).

RPG Compiler

This component is the program that translates specifications written in RPG language into a program in disk system format (DSF). 1130 RPG logic is described in the publication, IBM 1130 RPG Program Logic Manual, Form Y21-0010.

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:

- The Disk Cartridge Initialization Program (DCIP)
- 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: 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: 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: 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.
$CXR1	$CXR1 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: Setting — Meaning zero — Disk I/O not in progress non-zero — Disk I/O in progress $DBSY is simultaneously used as a retry counter by DISKZ and DISK1.
$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: 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: 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.
$GCOM	$GCOM contains the address of GCOM, the communication area for the Graphic Subroutine Package (GSP). GCOM contains common areas, pointers, and indicators that are used by most of the GSP subroutines. GCOM also contains the system display program for the GSP.
$GRIN	$GRIN is the Graphic Initialization program indicator.
$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: 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: 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 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.
$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: Settings — Meaning zero — Last card has not been read non-zero — Last card has been read

LABEL	DESCRIPTION
$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.
$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: 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: 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: Setting — Meaning zero — Exit from the PAUS subroutine non-zero — Branch back to the WAIT in the PAUS subroutine
$PBSY	$PBSY 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: 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: Setting — Meaning positive — Entry at $DUMP zero — Entry at $LINK negative — Entry at $EXIT
$RWCZ	$RWCZ is a switch indicating the type of operation last performed by the CARDZ subroutine. The switch settings are as follows: 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	$SNLT is the location used for sense light simulation by FORTRAN programs. The bits are used as follows: 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.
$XR3X	$XR3X contains the setting of index register 3 for the core load in core.
$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	$1132 is a switch indicating whether or not channel 12 has been detected on the 1132 Printer. The switch settings are as follows: Setting — Meaning zero — Channel 12 not detected or a skip to channel 1 executed non-zero — Channel 12 detected
$1403	$1403 is a switch indicating whether or not channel 12 has been detected on the 1403 Printer. The switch settings are as follows: 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	#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.
#BNDU through #BNDU+4	#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.
#CBSW	#CBSW is a switch indicating to the Core Load Builder the type of exit to be made. The switch settings are as follows: Setting — Meaning zero — Return to Core Image Loader non-zero — Return to DUP
#CIAD	#CIAD contains the relative location in sector @IDAD (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	#DCSW is a switch indicating to the ADRWS program the type of exit to be made. The switch settings are as follows: 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.
#ECNT	#ECNT contains the number of EQUAT records that have been processed by the Supervisor and stored on the SCRA.

•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: Setting — Meaning -2 — disk core image format (DCI) zero — disk system format (DSF) +1 — 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.
#GCNT	#GCNT contains the number of G2250 subroutines to be defined for the core load being built or the execution to be initiated.
#JBSW	#JBSW is a switch indicating the mode of operation established by the last JOB monitor control record. The switch settings are as follows: 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.
#LOSW	#LOSW is aswitch used by CLB to indicate the following. Setting — Meaning zero — LOCAL may not call a LOCAL one — LOCAL may call a LOCAL

LABEL	DESCRIPTION
#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: Setting — Meaning zero — Do not return to MODIF non-zero — Return to MODIF
#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: 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: 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: 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: 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.

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 (1DUMY) 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 (1DUMY) 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

Phase 1: SYL02

Phase 2: SYL03 - SYL05

PHASE 1

FUNCTIONS

Phase 1 determines which card I/O subroutine is required, i.e., 1442 or 2501. If the 2501 is the card reader, the System 2501 Subroutine overlays the System 1442 Subroutine. Both subroutines are assembled as part of Phase 1 and are naturally relocatable.

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 /063D) 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 /0635 , overlaying a part of the reload processing subroutine. Subphases 2 and 3 are loaded to sectors /063A and /063C.

Phase 1 stores the Resident Image on sector @RIAD. (DISKZ is stored on disk by phase 2.)

On reload operations, the SLET and Reload Table are tested for agreement with the checksum residing in the last word of the Reload Table sector. If no checksum is present, the cartridge is assumed to have been loaded by a modification level 0 System Loader. Consequently the first two sectors of SLET will be packed, and the third SLET sector cleared.

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(s), checks for errors, and saves the data obtained for use by phase 2. The version and modification level number is taken from the first PHID control record and saved.

Phase ID pairs must be in ascending order on the PHID record(s) with no intervening blank fields. If two PHID records are present, the second may contain blank fields after the last phase ID pair.

Phase 1 fetches phase 2 from disk into core storage and transfers control to it at BA000.

BUFFERS AND I/O AREAS

Card Input

AA904: 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.

AA902: a 60-word buffer that contains the 12 bit/word data from buffer AA904 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 DEL 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

BUFFR: a 640-word buffer used in all disk I/O operations.

BUFR1: a 320-word buffer, used only for the SLET/RELOAD table checksum calculation.

COMMUNICATION FROM PHASE 1 TO PHASE 2

The Console Printer Subroutine (WRYTZ) and the System I/O Subroutine (1442 or 2501) are in a low core position not overlayed by Phase 2. Phase 2 is then able to use these subroutines.

Likewise, information acquired by Phase 1 that must be passed on to Phase 2 and error messages required by both phases reside in a low core position so as not to be overlayed by Phase 2.

PHASE 2

FUNCTIONS, INITIAL LOAD AND RELOAD

Phase 2 checks the phase ID number sequence for ascending order and completeness throughout an initial load, and during the addition of one or more programs during a reload. At other times during a reload, the phases present should be in ascending order, but omissions are allowed.

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 in columns 12 through 15 of the Load Mode control record, phase 2 bypasses the programs described by phase ID pairs 2, 3, 8 and/or 9 from the PHID control record(s).

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 first 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 program described by phase ID pair 2, 3, 8 or 9 and that program is not present on the cartridge. For example, the Assembler Program (represented by phase ID pair 3) may have been deleted or never loaded.

At the conclusion of either an initial load or reload, the SLET and Reload Tables are checksummed and the result stored in the last word of the Reload Table sector.

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

FUNCTIONS, INITIAL LOAD ONLY

Phase 1 has cleared 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 one or more programs 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 a program is bypassed, no gap is left in SLET. The next program loaded follows immediately on the cartridge and the SLET entry for its first phase fills the first available location in SLET.

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.)

Each time a phase with a positive phase ID number is reloaded, the Reload Table is searched. If that phase ID is present, it is removed and the table repacked.

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 overlayed.

The phase ID range of an existing program may be expanded at the 'upper end' to permit addition of one or more phases to that program. In such a case, the phase whose ID is one less than the first phase to be added must be present so that a pointer can locate the position in SLET where data for the added phase will be inserted. SLET entries above the added phase will be shifted to provide 4 SLET words for

the new phase. System programs above the added phase will be shifted toward and into the cushion area as necessary.

System programs may be added to the system program area if each such program meets the following requirements:

1. It is described by a phase ID pair in the second PHID record.

2. It is described by a phase ID pair whose lower limit is greater than any phase ID in SLET.

3. No fixed area exists on the cartridge.

Before the program is stored, the SCRA, CIB, LET and UA will be shifted so that the 'high' end adjoins the system loader storage area at the end of the cartridge. When the type '81' record is processed, the SCRA, CIB, LET and UA will be shifted back from their temporary location to new sector addresses determined by the length of the program(s) added. A new cushion area will be established. User Area and chain addresses in LET are updated.

Once a program addition has started, the system loader reload operation is similar to an initial load. All phases must be present and in sequence from the first phase of the added program up to and including the last phase ID present in the second PHID record. The additional disk storage area required will be equivalent to the length of the added program(s), plus up to 15 sectors to rebuild the cushion area.

Since the System Loader resides on the last two cylinders during its operations, working storage should be at least equal to the length of the added program(s) plus 31 sectors before performing the addition.

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

CARD1: 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.

CARD2: an 80-word buffer used in conjunction with buffer CARD1 for double buffering capability. This buffer is used only by the 2501 card I/O subroutine.

PKBFR: a 60-word buffer into which 12 bits/word data is compressed from buffer CARD1 when a 1442 is used as the input device, or from buffers CARD1 and CARD2 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 DEL 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

BUFR1: 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 BUFR1. This buffer is also used in calculating the SLET/RELOAD Table checksum.

BUFR2: a 320-word buffer used in disk I/O operations.

BUFR3: 320 words + core above 4K 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 BUFR3 is saved by writing it on the first sector of the CIB. The CIB is not cleared afterward.

SUBPHASES

Subphase 1

Subphase 1 contains the subroutines that are used only during an initial load. Phase 1 determines from the Load Mode control record the type of load being performed, and either overlays subphase 0 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 eacn phase as it is encountered. An error message 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 into the overlay area.

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.

A SLET search is performed for the word count and sector address of the Core Image Loader to be stored at the end of the Disk I/O Subroutine, DISKZ.

Subphase 2 reads DCOM into the buffer BUFR3 and initializes or updates the following:

Location	Value Inserted
#SYSC	version and modification level from PHID control record
#RP67	a positive value if a 1442, Model 6 or 7 is present, zero otherwise
#PIOD	the value indicating the principal I/O device, as determined from REQ control records (+=2501/1442, 0=1442/1442, -=1134/1055)
#PPTR	the value indicating the principal print device, as determined from REQ control records (+=1403, 0=1132, -=Console Printer)
#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 sector address
#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.

Subphase 2 reads the Resident Image from disk into the buffer BUFR2 and initializes or updates the following:

Location	Value Inserted
$CH12	address of channel 12 indicator for the principal print device, as determined from REQ control records, i.e., $1403, $ 1132, or $CPTR
$CORE	core size (may be actual, or set by CORE control record)
$DREQ	a negative value (indicating DISKZ)
$IREQ	address of DUMP entry point ($DUMP)
$ULET	sector address of LET for logical drive 0
$CILA	address in which the word count and sector address of Phase 1 and the Core Image Loader is to be phaced ($ZEND-4)
$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

On an initial load the first sector of LET is established and initialized with a 1DUMY 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. On a reload in which one or more programs are called, the chain addresses in LET are updated to reflect the new position of LET on the disk due to the shift to make room for the new programs(s).

Subphase 2 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 RLTBL subroutine in phase 2.

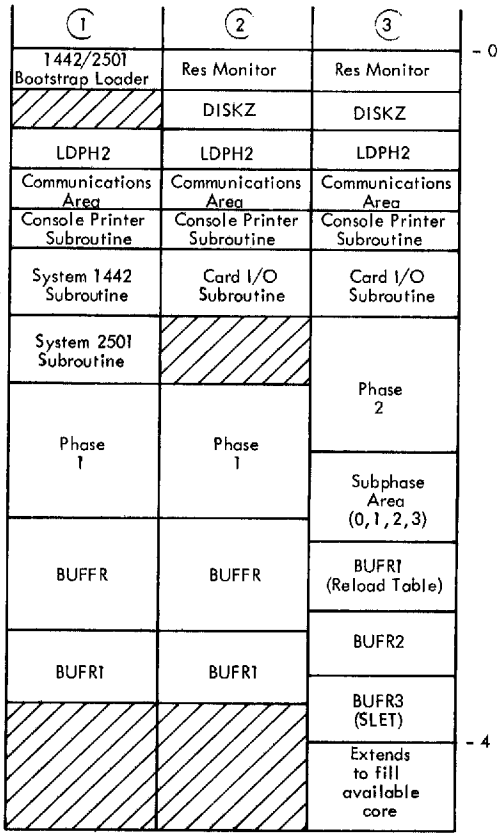
Subphase 3 places the word count and sector adress of phase 1 of the Core Image Loader, obtained from SLET, into DISKZ in core storage and into DISKZ on cylinder 0.

If a normal reload is being performed, subphase 3 prints "END RELOAD" to the console printer and branches to $EXIT. If the '81' record was followed by a "// XEQ MODIF" record, subphase 3 branches to the Auxiliary Supervisor with a minus 6 parameter causing a "// XEQ MODIF" record to be placed in the Supervisor buffer. The Supervisor then executes MODIF.

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 store the System Library.

CORE LAYOUT

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



●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.

The seventh word of the sector is the cartridge load status word. Minus 2 indicates that an initialization has been performed and an initial load only is permitted. Plus 2 indicates that the cartridge has been initial-loaded and a reload only is permitted.

SYSTEM LOCATION EQUIVALENCE TABLE (SLET)

SLET occupies three 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. (A program or phase not included during an initial load cannot later be included in a reload operation unless the program to be reloaded meets the requirements described under FUNCTIONS, RELOAD ONLY. Otherwise 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 subroutines are built by the System Loader and do not come directly from program decks.

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 address 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. The last word of the Reload Table is used for the SLET and Reload Table checksum. 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 IPL 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 and given control by 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 in such a fashion that all locations occupy their permanent positions in core storage.

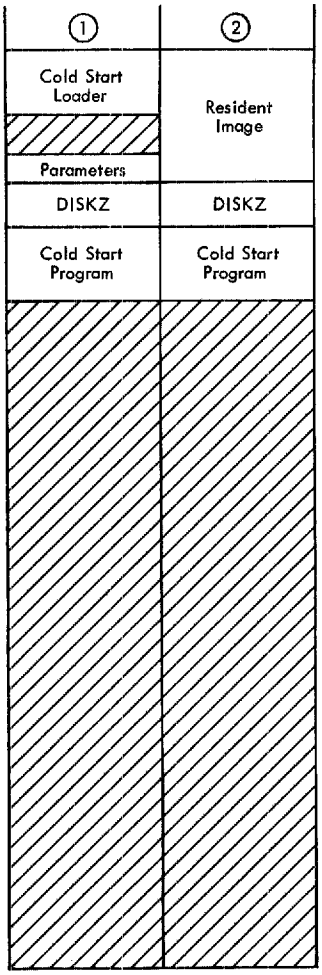


Figure 2. Core Layout During Cold Start

Page Blank In Original

Page Blank In Original

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 phase 1 of 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.

If a JOB record is recognized Phase 2 is fetched and control passed to it.

Upon detecting an ASM, FOR or RPG 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 $PRET. 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 System Loader with a reload function.) 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.

Upon detecting an XEG 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 PHASE 4 (see below). Otherwise, the Monitor Control Record Analyzer prints an error message and reads the next control

Page Blank In Original

Page Blank In Original

blank (the mainline is in Working Storage) the name in the SCRA is set zero.

EQUAT Control Record Processing

The pair(s) of subroutine names found in the EQUAT Control Record are converted to name code and stored to sector 7 of the SCRA.

SUPERVISOR CONTROL RECORD AREA (SCRA)

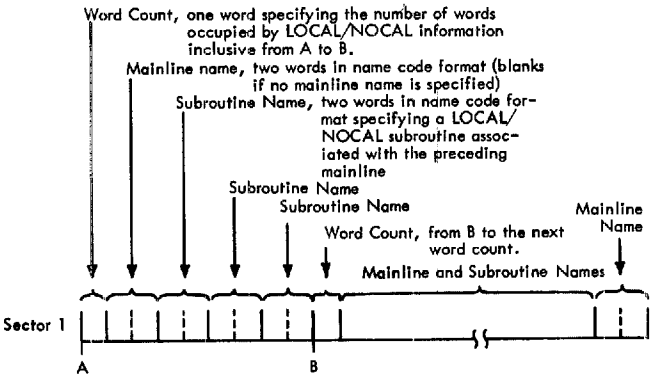
Sectors 0 and 1 of the SCRA are occupied by the LOCAL information for the core load or execution currently in progress (see diagram, below). The first word of sector 0 contains the word count of the information stored in the two LOCAL sectors.

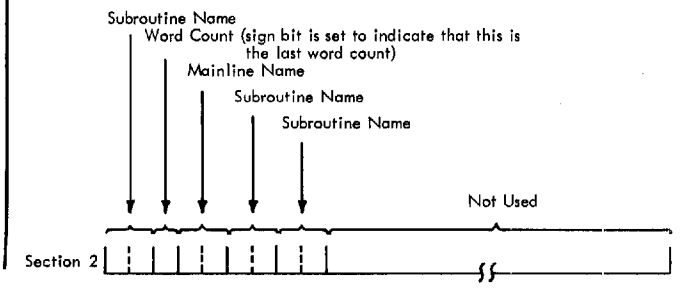
Sectors 2 and 3 of the SCRA are occupied by the NOCAL information for the core load or execution currently in progress (see diagram, below). The first word of sector 2 contains the word count of the information stored in the two NOCAL sectors.

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.

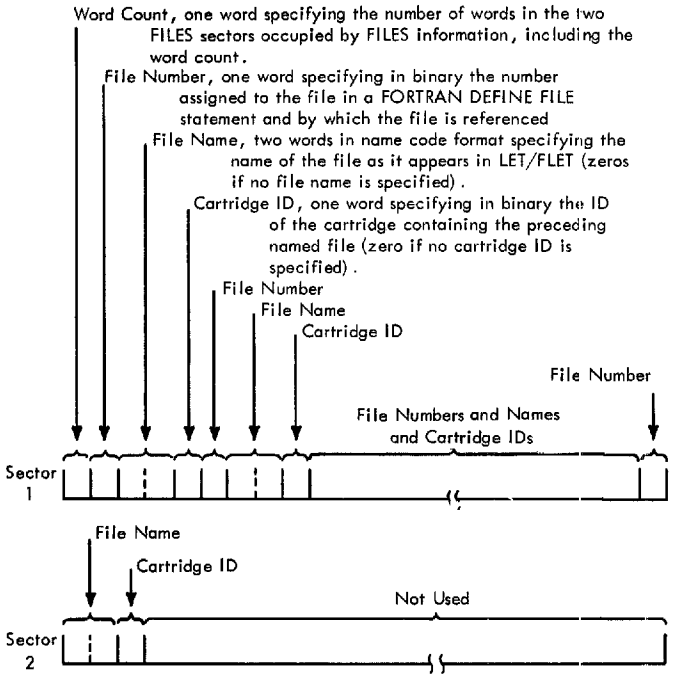
Sector 6 is occupied with G2250 information for the core load or execution currently in progress.

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

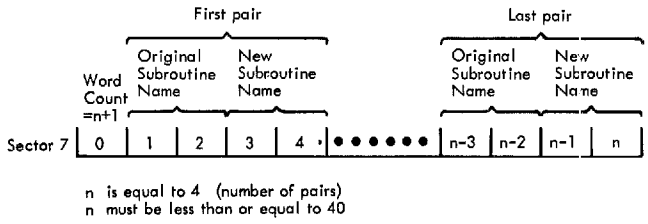




The format of information in the FILES sectors is as follows:

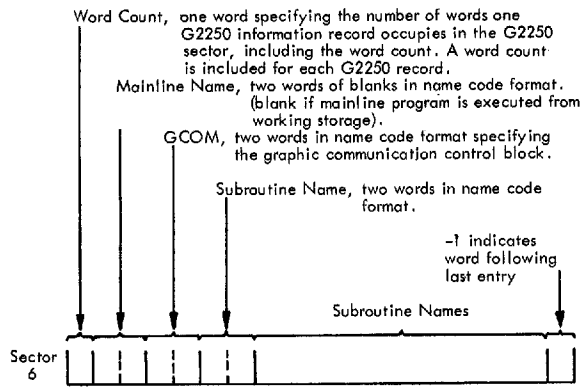


The information in the EQUAT sectors is as follows:



n is equal to 4 (number of pairs)
n must be less than or equal to 40

The format of the information in the G2250 sector is as follows.



SYSTEM CORE DUMP PROGRAM - PHASE 6

If an entry was made to the Skeleton Supervisor at $DUMP, the Skeleton Supervisor writes the contents of location $CIBA+1 through 4095 on the CIB, then fetches and transfers control to phase 1 of 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, these checks are made:

- If both limits are zero, the lower limit is left zero and the upper limit is set to core size.
- If a limit is larger than core size, the limit is subtracted from the core size and the difference is used as the limit.
- 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 offset 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 $DUMP contains no return address (i.e., is zero), the Dump program executes a CALL EXIT.

If $DUMP 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. Control is then returned to the restored core load at the location following the dump parameters.

AUXILIARY SUPERVISOR - PHASE 7

If an entry was made to the Skeleton Supervisor at $DUMP and $DMPF (the dump format indicator) is negative, phase 1 of the Core Image Loader fetches into core storage and transfers control to the Auxiliary Supervisor.

The Auxiliary Supervisor has these functions:

- It writes dummy monitor control records in the Supervisor buffer for processing by the Monitor Control Record Analyzer.
- It prints error messages for errors detected by the Core Image Loader.
- 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 executes a CALL EXIT.

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

Page Blank In Original

Page Blank In Original

Page Blank In Original

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 $DUMP ($RMSW is positive), phase 1 tests $DUMPF, 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 $EXIT ($RMSW is negative), phase 1 tests $DZIN 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 $LINK ($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:

Disk I/O Subroutine	Low COMMON Decimal	Low COMMON Hexadecimal
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-

Page Blank In Original

Page Blank In Original

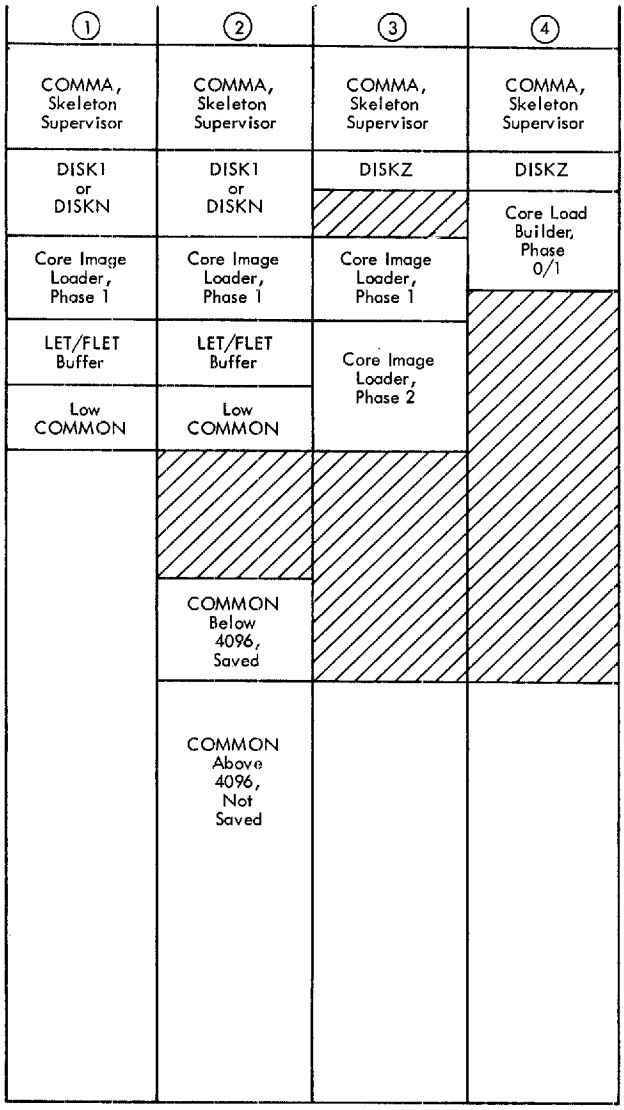


Figure 7. Core Layout on Supervisor at $LINK (Link in Disk System Format)

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. Panels 4, 5, and 6 represent three different cases. In panel 4 the program being linked to is a DCI program, whereas in panels 5 and 6 it is a DSF program. Although the Core Load Builder is called into core in the DSF cases, it is not shown here in order to simplify the diagram. 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.

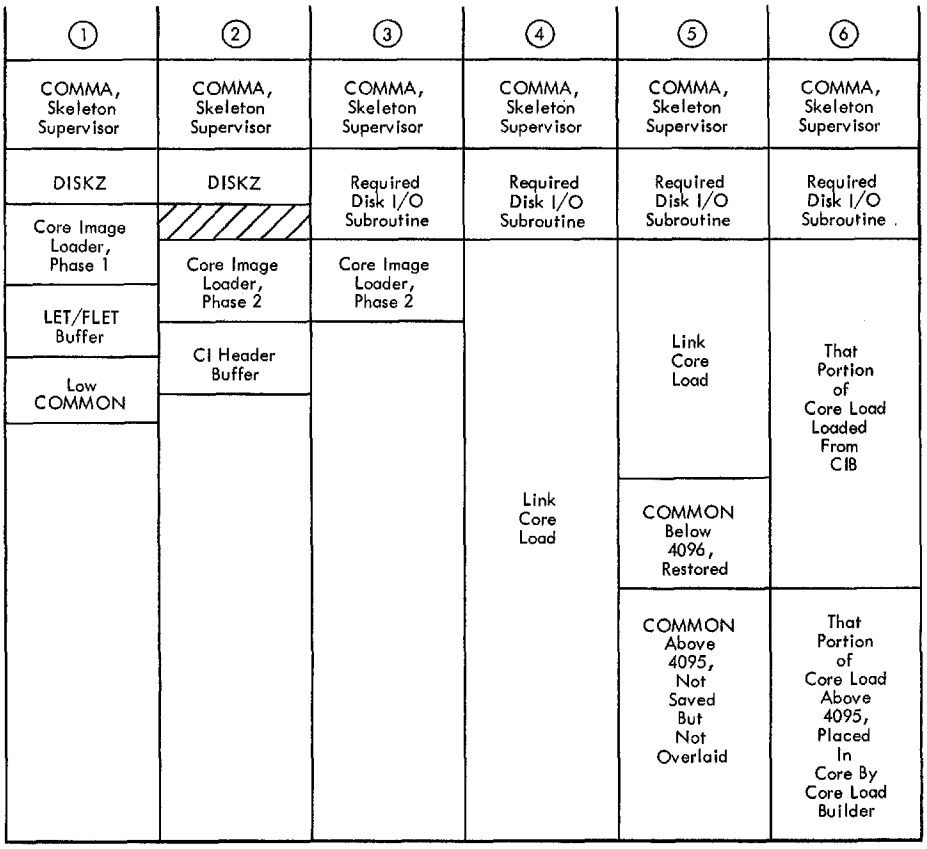


Figure 8. Core Layout on Supervisor Entry at $LINK (Link in Core Image Format)

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

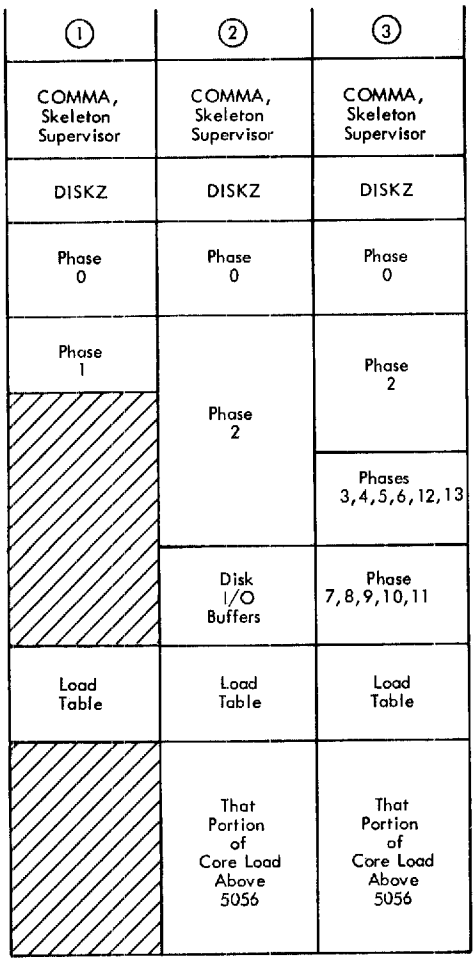


Figure 9. Core Layout During Core Load Builder Operation

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, FILES, and G2250 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 if the size of core is 4K, otherwise the part of the core load below 5056. The first word of the mainline is stored in the first word of the CIB following the core image header, 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 if core size is 4K, otherwise locations 5046-5049, 5042-5045, 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. As an explanation, suppose that 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. These terms are useful in the following context.

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, (3) /008 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.

Since locations 4097-5056 are reserved for the Load Table (core size greater than 4K) CLB has to read in that part of the core load prior to exit to CIL or DUP.

EQUAT TABLE

The Equat Table consists of at most ten pairs of names of subroutines or names of files defined in a DSA statement. The first name in each pair is the subroutine or the file to be substituted with the second name of each pair. This checking is done during relocating of a mainline or its subroutines. Thus no name is added to the Load Table before it has been checked against the Equat Table.

LOCAL, NOCAL, FILES, and G2250 INFORMATION

LOCAL, NOCAL, FILES, and G2250 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, FILES, and G2250 information in the SCRA, see Section 6: Supervisor or Section 9: Disk Utility Program.

EQUAT information is obtained from the SCRA. This information is supplied by the Supervisor Control Record Processing phase of the Supervisor (see Section 6: Supervisor). For the format of EQUAT information in the SCRA, see Section 6: Supervisor.

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

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 compared with the Equate Table and if no match is found they are put in the Load Table. But if a match is found the original subroutine name is replaced with the new subroutine name before putting it in the Load Table. Hence the Load Table will contain the name of subroutines that will be loaded rather than the subroutines that are called. 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) in the core image header 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 real-number pseudo-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 between it and the CALL TV in order to make the pseudo-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,

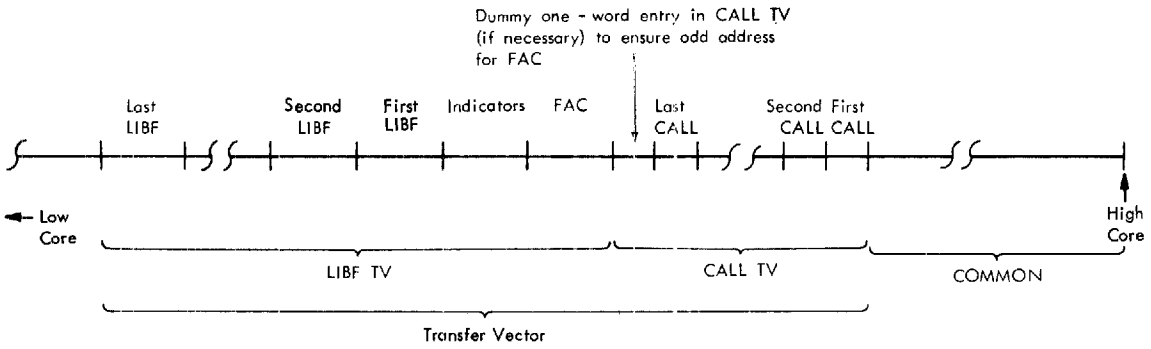


Figure 10. Layout of the Transfer Vector

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:

Word	Description
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:

Word	Description
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.

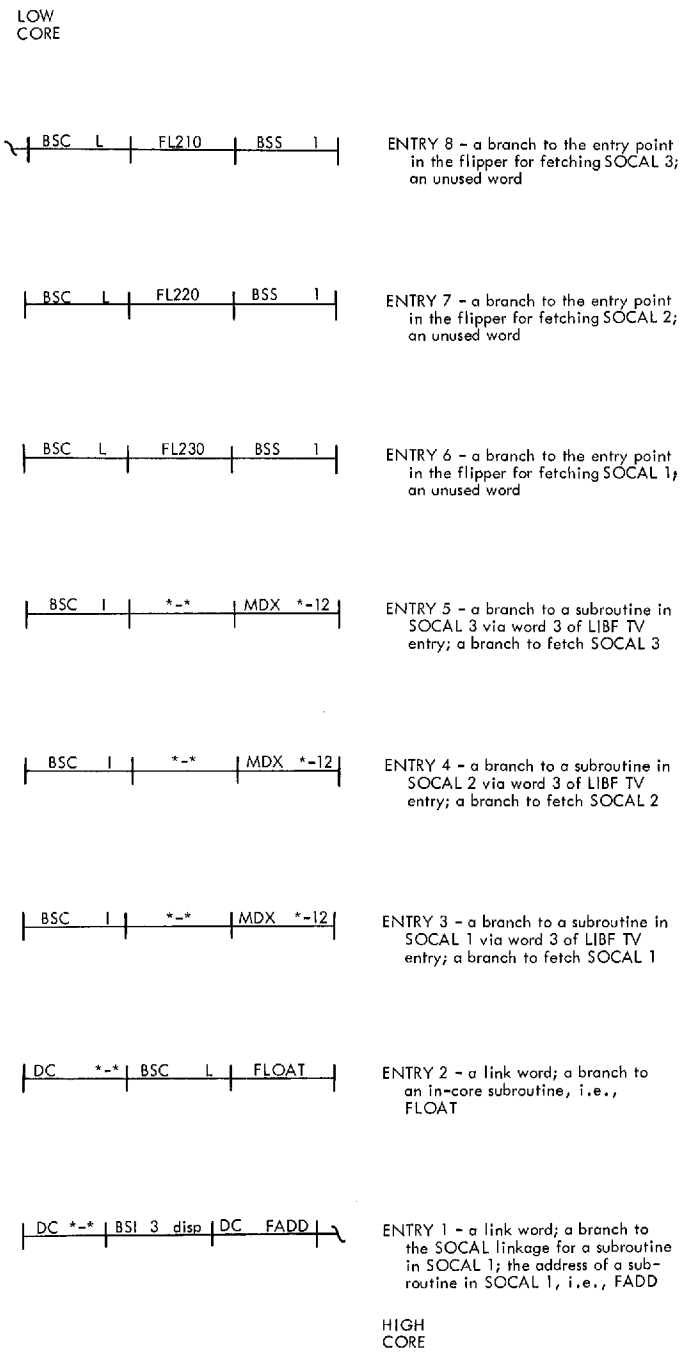


Figure 11. SOCAL Linkage in the LIBF Transfer Vector

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.

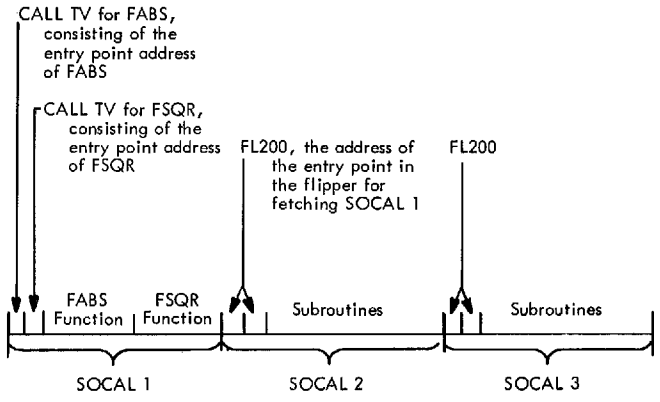


Figure 12. CALL Transfer Vector for SOCALs

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 absolute 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 numbers in the DEFINE FILE Table, the file is set up in Working Storage, i.e., the relative (to beginning of Working Storage) sector address is stored in the DEFINE FILE Table. The sign bit is set to indicate that this is a relative address. 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:

Word	Description	Symbolic Reference
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.

Subroutine	Function
LK000	Fetch a phase
LS000	Search LET/FLET
RH000	Fetch the mainline program header record
NW000	Fetch the next word in sequence fro from the data buffer, reading a new sector when nexessary
BT000	Add an entry to the Load Table
PM000	Fetch one of the message phase and transfer control to it
GP000	Read from or write to the disk
TL000	Terminate Loading
EX000	Print a message and exit
CN000	Test a subroutine name for disk I/O

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 stored there that are needed by the Core Load Builder and fetching the mainline leader record and saving the information therein.

In addition, phase 1 makes an entry in the Load Table for each LOCAL, NOCAL and G2250 specified in the LOCAL, NOCAL and G2250 information in the SCRA, if any.

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

Subroutine	Function
IN000	Initialize the Core Load Builder, process the mainline header process LOCAL, NOCAL and G2250 names from the SCRA.
LN000	Enter LOCAL, NOCAL and G2250 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 lo logic for relocating a program, i.e., incorporating rating 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.

Subroutine	Function
MC000	Master control for the Core Load Builder
RL000	Relocate a program; convert it from disk system format to core image format

The data and CIB buffers are combined into a single 640-word buffer for the purpose of fetching the *LOCAL, *NOCAL, *FILES and *G2250 information from the SCRA.

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.

Subroutine	Function
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
CQ000	Check name of subroutines or files defined in a DSA statement against the Equate Table.
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.

Subroutine	Function
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.

Subroutine	Function
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.

Subroutine	Function
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.

Subroutine	Function
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.

PHASE 13

Phase 13 is entered from phase 2 when a GSB, GBE, or GBCE order is encountered. The GSB processing adds the relocation factor to the GSB address to insure that it is below core location 8192. The GBE and GBCE processing enters the name of the external subroutine in the Load Table. The GBE or GBCE order is then replaced with a GB or GBC Indirect through the transfer vector entry.

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.

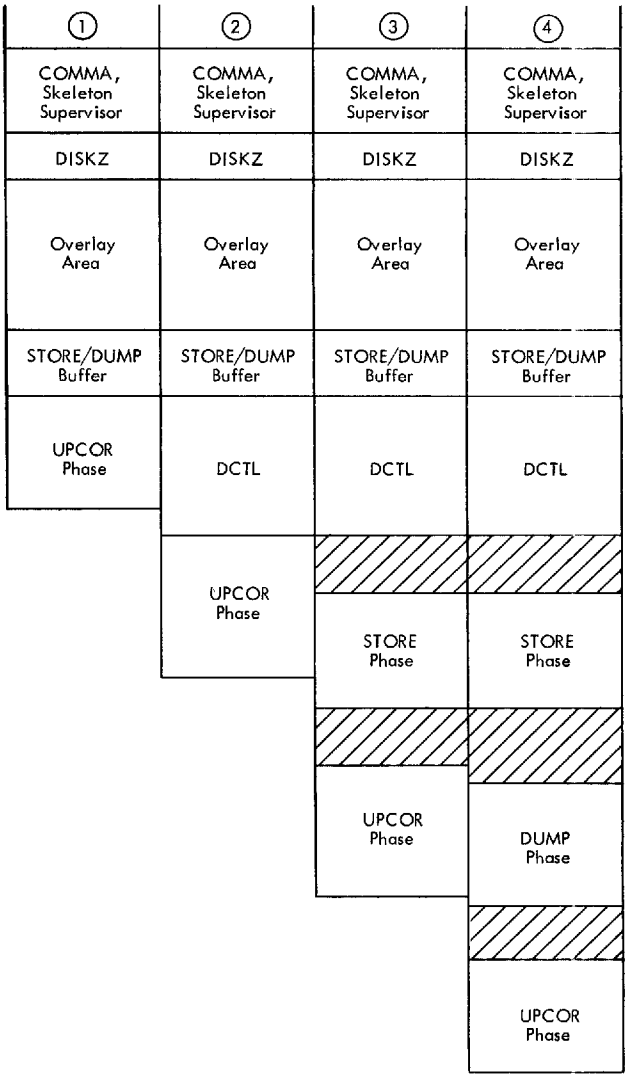


Figure 13. Core Layout During Disk Utility Program Operation

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 STOREDATACI	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 DEFINE VOID RPG	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 is inserted wherever a DCI or DDF entry is preceded by a DSF entry, and is reflected in LET as if a program called "1DUMY" were stored. That is, each instance of padding generates a 1DUMY entry in LET, and the block count for each of these entries is the number of disk blocks to the nearest sector boundary (may be zero). The last entry in LET is always a 1DUMY 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 1DUMY 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:

Location	Relative Address (decimal)
#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
#GCNT	30
#JBSW	9
#LCNT	11
#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:

Location	Phase Name	Phase Number
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

Location	Phase Name	Phase Number
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 $IREQ 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 $IREQ, 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 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.

EBCSW -- Set non-zero by DCTL when a STOREDATA or a DUMPDATA control record contains an E in column 11. Used by STORE when input data is to be stored in packed EBCDIC format, i.e. 80 card columns to 40 words, and by DUMP when such data is to be converted and dumped to cards or 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.

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.

RPGSW -- See ASMSW and FORSW.

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 STOREDATACI 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 1DUMY 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:

Control Record	Subroutine Called
*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 STOREDATACI 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 STOREDATACI 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 STOREDATACI.
- 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. Four types of STORECI control records are processed by this phase; *LOCAL, *NOCAL, *FILES, and G2250. FILEQ consists of the subroutines LC000, FR000, GR200, 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/NOCAL information for the core load to be built is stored in the SCRA.

The format of LOCAL/NOCAL information in the SCRA is shown on page 22.

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.

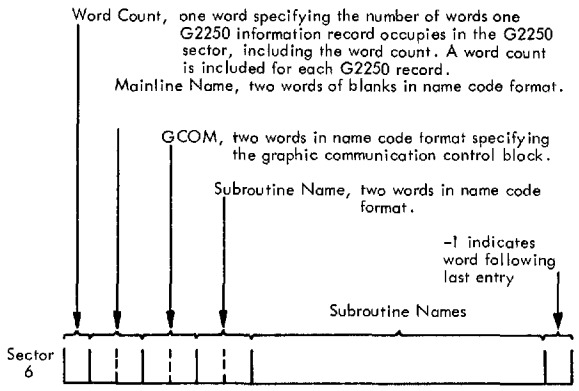
FILES information for the core load to be built is stored in the SCRA.

The format of FILES information in the SCRA is shown on page 22.

GR000

This subroutine processes *G2250 records. A mainline name is not specified on these records when they follow the STORECI control record. An error is indicated if a mainline name is specified. Thus, the mainline name is set to blanks. All G2250 subroutine names determined from G2250 control record information are converted to name code.

G2250 information for the core load to be built is stored in sector 6 of the SCRA. The format of G2250 information in the SCRA is shown below.



The following subroutine names will be entered in the G2250 sector of the SCRA if they were specified in the G2250 information record.

GCHAR or GUPER, the character stroke table.
GSP12, the verification direct entry subroutine.
GSP06, the scissoring subroutine.
GSP05, the ICA area expansion subroutine.
GSP11, the indexed entity scan subroutine.

Note: As each subroutine is specified, it will occupy the next available position.

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 when DCTL recognizes a DUMP control record. 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.

LF200

This subroutine provides the exit for FILEQ. When all *LOCAL, *NOCAL, *FILES, and G2250 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.

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.

XE000

This subroutine converts from packed EBCDIC to card code and places the data in a buffer for punching.

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 1DUMY entry of the same size. If there are adjacent 1DUMY entries, they are combined to form a single 1DUMY entry and FLET is compressed by 0, 3, or 6 words depending upon whether there are 0, 1, or 2 adjacent 1DUMY 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 1DUMY 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. The 1DUMY entry, which normally precedes this 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 1DUMY entry did not precede the DCI program or data file, one of the appropriate size is inserted to sectorize the DCI program or data file. The shrinkage or packing continues until the last 1DUMY 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, the RPG 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 a program has been eliminated, it cannot 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 DCOM 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 or the RPG Compiler is 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.

RPGSW -- non-zero when the RPG Compiler is to be deleted

Errors Detected

The following DUP errors are detected in DEFINE: D15, D70, D80, D81, D82, D83, D84, D85, D86 and D87.

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 $SCAN-$SCAN+7. 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 subroutine 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 // in columns 1 and 2 or * in column 1. If this information is not found in the first two columns of the card, the subroutine returns immediately to DCTL. In this way, invalid records can be bypassed at maximum card speed without using a double-buffering technique.

If // or * is found in column 1, 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 RTURN 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.

PRECI

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.

PRECI 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.

Code in column 35	Phase(s) dumped
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:

Subroutine in execution	Contents of $PHSE
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.

Digit 2 of $PHSE	Disk I/O Operation
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 PHDM 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
Phase 13:	ASM25-ASM26

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.

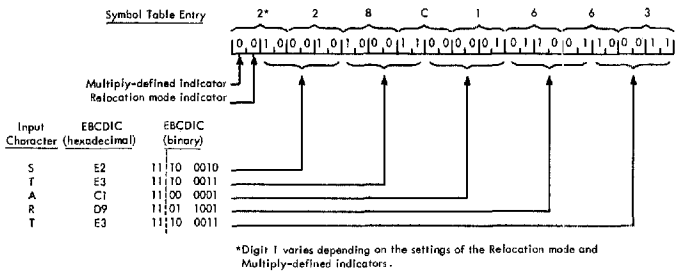
OVERLAY AREA

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, 13, @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.

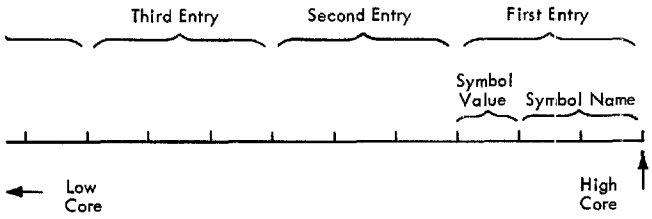
The Overlay Area resides in core storage following ASCOM and preceding phase 9.

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 contains 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.

DCOM of the master cartridge, and DCOM of the cartridge defined to contain System Working Storage is updated as follows:

- The first entry point name of a Type 3, 4, 5, or 6 subroutine is placed into the two words specified by # NAME, in internal 1130 NAME code.
- The execution address (or first entry point address) is placed in # ENTY.
- The length of the program in disk blocks is placed in one word of the # WSCT quintuple corresponding to the logical drive that contains System Working Storage.
- A zero is placed in one word of the # FMAT quintuple corresponding to the logical drive that contains System Working Storage to indicate that this Working Storage contains DSF.

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 XFLC 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.

If the search is terminated due to an invalid op code, it is determined whether the graphics phase is in core. If it is in core, control is passed to it, otherwise the instruction is processed as an invalid op code.

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 (BUF1). 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.

PHASE 13

Phase 13 searches the master graphics op code table to determine if the mnemonic in the current input record is a valid 2250 mnemonic. If the mnemonic is not in the op code table, control passes to Phase 9 to post the error code for an invalid mnemonic. When a 2250 mnemonic is found, a branch is made to G4040.

G4040 determines if the Assembler is in pass 1 or pass 2. During pass 1 the Location Assignment Counter (LAC) is incremented by 1 or 2, depending upon whether a 1- or 2-word order is being processed. During pass 2 a branch is made to G4070 which causes a branch to be made to the address contained in the fourth word of the op code table entry. This is the address of the section that will process the order.

The format, tag, and operand fields of the order are tested. Control passes to the SCAN routine in phase 9 to evaluate any orders which contain operands. When control returns, the operands are tested to determine whether they are valid. If an error is encountered, control passes to ERFLG to post the error. The operand is put in the op code buffer and a branch is made to G4090 to transfer the contents of the op code buffer to the Disk System Format (DSF) buffer. G4090 passes control to B4HEX in Phase 9 to convert the contents of the op code buffer from binary to 4 hexadecimal characters. When control returns it passes to DFOUT which moves the word to the DSF buffer. The LAC is incremented by one and the op code buffer is set to zero. Control passes to LDLBL to access the next record.

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 (BUF1). A list of error messages is contained within ERMSG.

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 subroutines 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:

Phase Number	Phase Name	Function
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.

Phase Number	Phase Name	Function
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.
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.
6	Real Constant	Place the names of real constants in the Symbol Table.
7	DEFINE FILE, CALL LINK, and CALL EXIT	Check the syntax of DEFINE FILE, CALL LINK, CALL EXIT statements; determine the defined file specifications.
8	Variable and Statement Function	Place the names of variables, integer constants, and statement function parameters in the Symbol Table.
9	DATA Statement	Check the syntax of the DATA statement, check its variables for validity, and reformat the statement.
10	FORMAT	Convert FORMAT statements into a special form for use by the input/output subroutines during execution of the object program.
11	Subscript Decomposition	Calculate the constants to be used in subscript calculation during execution of the object program.
12	Ascan I	Check the syntax of all arithmetic, IF, CALL, and statement function statements.
13	Ascan II	Check the syntax of all READ, WRITE, FIND, and GO TO statements.
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.
15	Subscript Optimize	Replace subscript expressions with an index register tag.
16	Scan	Change all READ, WRITE, GO TO, CALL, IF, arithmetic, and statement function statements into a modified form of Polish notation.
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.

Phase Number	Phase Name	Function
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 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.

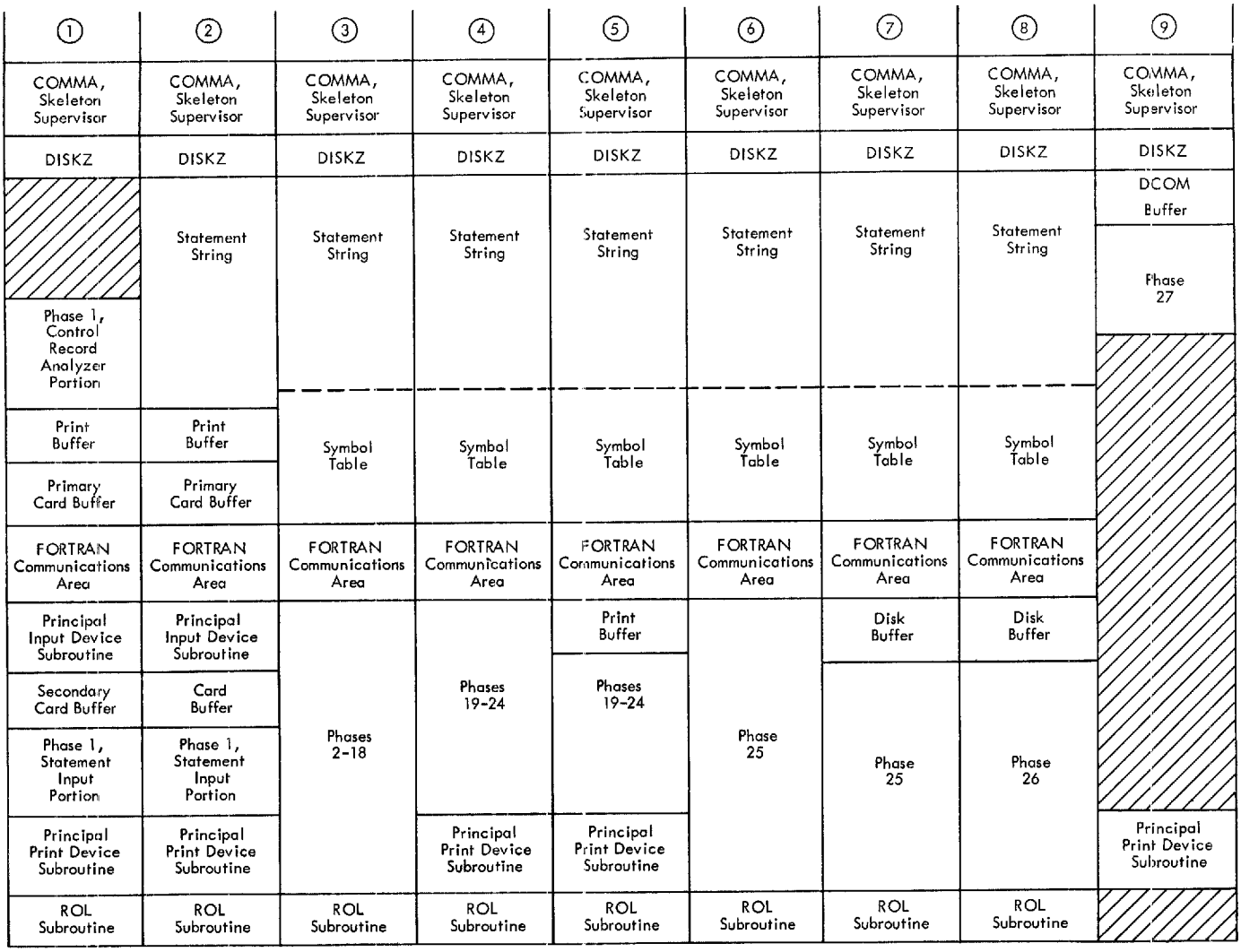


Figure 15. Core Layout During FORTRAN Compiler Operation

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	ORG	The origin address for an absolute program.
2	SOFS	The address of the start of the string.
3	EOFS	The address of the end of the string.
4	SOFST	The address of the start of the Symbol Table.
5	SOFNS	The address of the start of the non-statement-number entries in the Symbol Table.
6	SOFXT	Phases 1-20. The address of the start of the Symbol Table entries for SGTs (subscript-generated temporary variables). Phases 21-25. The work area word count.
7	SOFGT	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.
8	EOFST	The address of the end of the Symbol Table.
9	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.
10	CSIZE	All phases except Phase 20. The COMMON area word count. Phase 20. The address of the lowest-addressed word reserved for COMMON storage.
11	ERROR	Bit 15 set to 1 indicates overlap error. Bit 14 set to 1 indicates other error.
12-13	FNAME	The program name (obtained from the NAME control record, or a SUBROUTINE or FUNCTION statement) stored in name code.
14	SORF	Set positive to indicate FUNCTION. Set negative to indicate SUBROUTINE. 0 indicates mainline.
15	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. Bit 8 set to 1 indicates Origin.
16	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.
17	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.

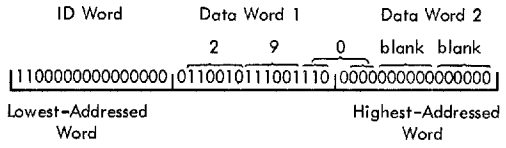
During the output phases, 19 through 26, these entries in the Symbol Table are replaced by object program addresses that are inserted into the object coding by phase 26.

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.

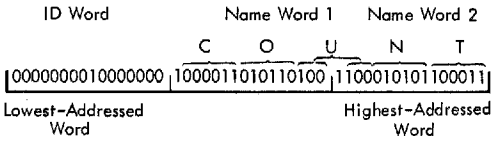
The ID word occupies the lowest-addressed word of the three word entry. The Name-Data words occupy the two higher-addressed words.

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



Entry in hexadecimal form –E000 65CE 0000

The entry for the subprogram name COUNT would appear as:



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.

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

Name-Data Words

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

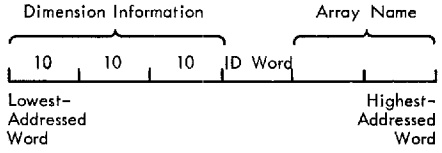
Word	Bit	Contents
1	0	0, if the following 15 bits contain the first half of a constant; 1, if the following 15 bits contain anything other than the first half of a constant.
	1-15	First 15 bits of the 30-bit Symbol Table entry
2	0	Same as bit 0 of word 1
	1-15	Second 15 bits of the 30-bit Symbol Table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

Dimensioned Entries

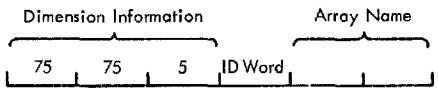
The Symbol Table entry for a dimensioned variable requires six words: two for the array name, one for the ID word, and three for the dimension information. The dimension information occupies the three lowest-addressed words, the ID word occupies the next higher-addressed word, and the Name-Data words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

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

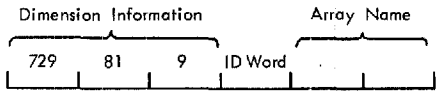
For one-dimensional arrays, all three dimension words contain the integer constant that specifies the dimension of the array. For example, the entry for array ARRAY (10) would appear as:



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:

Bit	Contents
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

COMPILATION DISCONTINUED

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

Error messages appear in the following format:

CAA ERROR AT STATEMENT NUMBER XXXXX+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.

There are also error messages in the format

CAA ZZZZZZZZZZ...

where ZZZZZ.... is an explanation of the error condition.

See the Programming and Operator's Guide publication for a list of the FORTRAN error numbers, their explanation, 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.

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.

If a small change is made to the ROL subroutine, it 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 16) 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 14) 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 14) 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 12-13) 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 17) in the FORTRAN Communications Area of the number of files defined.

Errors Detected

The errors detected by phase 7 are: 3, 70, 71, 72, 73 and 74.

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, 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, except if they are found in subscript expressions. 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

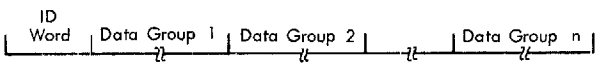
The following errors are detected in phase 8: 7, 24, 25, 26, and 43.

PHASE 9

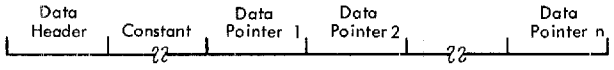
- Checks the DATA statement for correct syntax and valid variable references.
- Reformats the DATA statement into a string of data groups.

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:

Bits	Contents
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:

Word	Bit	Contents
1	0	1
	1	0, if no displacement word follows 1, if a displacement word follows
	2	0, non-externally subscripted variable 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:

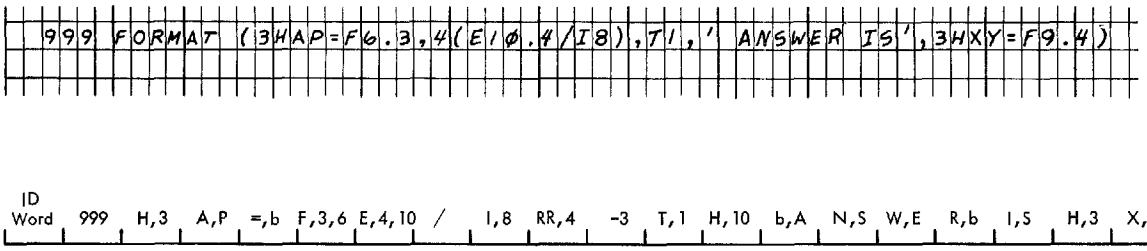


Table 6. Conversion of FORTRAN FORMAT Specifications

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 --

$A(C_1 * I + C_2, C_3 * J + C_4)$

$D_1 = C_1 * S$

$D_2 = L * C_3 * S$

$D_4 = \left[(C_2 - 1) + L * (C_4 - 1)\right] * S$

For a 3-dimension array --

$A(C_1 * I + C_2, C_3 * J + C_4, C_5 * K + C_6)$

$D_1 = C_1 * S$

$D_2 = L * C_3 * S$

$D_3 = L * M * C_5 * S$

$D_4 = \left[(C_2 - 1) + L * (C_4 - 1) + L * M * (C_6 - 1)\right] * S$

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:

Word	Contents
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:

Word	Contents
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 extensions of the string and which are used by the Interpreter in the same manner as the strings.

Table 7. FORTRAN Forcing Table

Operator	Definition	Forcing Value Left	Forcing Value Right	String Pointer
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
C	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
ß	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

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 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 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 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 15) 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 16) 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 15), 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 11) 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 or absolute mode. 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 15) 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.
- Prints error messages 86, 96 and 99

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 ($NXEQ) to be set, and the non-DUP switch ($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.

UPDATING THE MASTER CARTRIDGE

If the compilation is free of errors, DCOM on the master cartridge is updated as follows:

- The program length in disk blocks is placed in # DBCT and in one entry in the # WSCT quintuple corresponding to the cartridge that contains System Working Storage.
- The execution address (or entry point address) is placed in #ENTY.
- One entry in the # FMAT quintuple corresponding to the cartridge that contains System Working Storage is set to zero to indicate that the Working Storage contains DSF.

In addition, if System Working Storage is on other than the master cartridge, # WSCT and # FMAT of the cartridge containing System Working Storage are updated as described above.

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
FORTRAN 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.

Function	Name(s)	Type	Subtype	Reference	Deck ID
Utility					
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
Common FORTRAN					
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
FORTRAN Trace Stop	TSTOP	4	8	CALL	T3M00010
FORTRAN Trace Start	TSTRT	4	8	CALL	T3N00010
Selective Dump	PDUMP	4	0	CALL	T3K00010
FORTRAN Sign Transfer					
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
Extended Precision Arithmetic/Function					
Extended Precision Hyperbolic Tangent	ETANH, ETNH	4	8	CALL	S2I00010
Extended Precision A**B Function	EAXB, EAXBX	4	8	CALL	S2C00010

Function	Name(s)	Type	Subtype	Reference	Deck ID
Extended Precision Arithmetic/Function (Cont'd)					
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
Standard Precision Arithmetic/Function					
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
Common Arithmetic/Function					
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
System					
LOCAL/SOCAL Flipper	FLIPR	3	0	-	U5D00010
DCOM Update	SYSUP	4	0	CALL	U5E00010
FORTRAN Trace					
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
FORTRAN I/O					
Non-Disk Formatted FORTRAN I/O	SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX	3	3	LIBF	T4C00010
FORTRAN Find	SDFND	3	1	LIBF	T4B00010

Function	Name(s)	Type	Subtype	Reference	Deck ID
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
	URED, UWRT, UIOI, UIOF, UIOAI, UIOAF, UIOFX, UIOIX, UCOMP, BCKSP, EOF, REWND				
Common FORTRAN					
FORTRAN Pause	PAUSE	3	2	LIBF	T2A00010
FORTRAN Stop	STOP	3	2	LIBF	T2B00010
FORTRAN Subscript Displacement Calculation	SUBSC	3	0	LIBF	T2D00010
FORTRAN Subroutine Initialization	SUBIN	3	0	LIBF	T2C00010
FORTRAN Trace Test and Set	TTEST, TSET	3	0	LIBF	T2E00010
FORTRAN I/O and Conversion					
FORTRAN Card 1442 (Read/Punch)	CARDZ	5	3	LIBF	T5A00010
FORTRAN Card 1442-5 (Punch)	PNCHZ	5	3	LIBF	T5G00010
FORTRAN Card 2501 (Read)	READZ	5	3	LIBF	T5J00010
FORTRAN Paper Tape	PAPTZ	5	3	LIBF	T5F00010
FORTRAN 1132 Printer	PRNTZ	5	3	LIBF	T5H00010
FORTRAN 1403 Printer	PRNZ	5	3	LIBF	T5I00010
FORTRAN Keyboard/Typewriter	TYPEZ	5	3	LIBF	T5K00010
FORTRAN Typewriter	WRTYZ	5	3	LIBF	T5L00010
FORTRAN Hollerith to EBCDIC Conversion	HOLEZ	3	3	LIBF	T5D00010
FORTRAN Get Address Subroutine	GETAD	3	3	LIBF	T5C00010
FORTRAN EBCDIC Table	EBCTB	3	3	-	T5B00010
FORTRAN Hollerith Table	HOLTB	3	3	-	T5E00010
Extended Precision Arithmetic/Function					
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
Standard Precision Arithmetic/Function					
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

Function	Name(s)	Type	Subtype	Reference	Deck ID
Standard Precision Arithmetic/Function (Cont'd)					
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
Common Arithmetic/Function					
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
Interrupt Service					
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.

Function	Name(s)	Type	Subtype	Reference	Deck ID
Conversion					
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
Conversion Tables					
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
SCA Subroutines					
	SCAT1	5	0	LIBF	W1F00010
	PRNT2	5	0	LIBF	W1E00010
	HOLCA	3	0	-	W1C00010
	STRTB	3	0	-	W1G00010
	SCAT2	5	0	LIBF	W1H00010
	SCAT3	5	0	LIBF	W1I00010

Function	Name(s)	Type	Subtype	Reference	Deck ID
SCA Subroutines (Cont'd)					
	HXCV	3	0	-	W1D00010
	EBC48	3	0	-	W1A00010
	HOL48	3	0	-	W1B00010
Mainline					
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
System/Miscellaneous					
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
Interrupt Level Subroutines					
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
1627 Plotter Subroutines					
Scale (Extended Prec.)	SCALE	4	0	CALL	V1N00010
Scale (Std. Prec.)	SCALF	4	0	CALL	V1C00010
Grid (Extended Prec.)	EGRID	4	0	CALL	V1C00010
Grid (Std. Prec.)	FGRID	4	0	CALL	V1H00010
Plot (Extended Prec.)	EPLOT	4	0	CALL	V1D00010
Plot (Std. Prec.)	FPLOT	4	0	CALL	V1I00010
Point Characters	POINT	4	0	CALL	V1M00010
Character (Extended Prec.)	ECHAR	4	0	CALL	V1A00010
Character (Std. Prec.)	FCHAR	4	0	CALL	V1F00010
Annotation (Extended Prec.)	ECHRX, ECHRI, VCHRI	3	0	LIBF	V1B00010
Annotation (Std. Prec.)	FCHRX, FCHRI, WCHRI	3	0	LIBF	V1G00010
Scaler (Extended Prec.)	ERULE	3	0	LIBF	V1E00010
Scaler (Std. Prec.)	FRULE	3	0	LIBF	V1J00010
Interface	PLOTI	3	0	LIBF	V1K00010
Pen Mover	XYPLT	3	0	LIBF	V1P00010
1627 Plot	PLOTX	5	0	LIBF	V1L00010

*These are special versions that consist only of IBT information.

Function	Name(s)	Type	Subtype	Reference	Deck ID
Disk I/O					
Sequential Access	SEQOP, SEQIO, SEQCL	3	0	LIBF	ZSA
Direct Access	DAOPN, DAIO, DACLS	3	0	LIBF	ZDA
ISAM Load	ISLDO, ISLD, ISLDC	3	0	LIBF	ZIL
ISAM Add	ISADO, ISAD, ISADC	3	0	LIBF	ZIA
ISAM Sequential	ISEQO, ISETL, ISEQ, ISEQC	3	0	LIBF	ZIS
ISAM Random	ISRDO, ISRD, ISRDC	3	0	LIBF	ZIR
RPG Decimal Arithmetic					
Add, Subtract and Numeric Compare	RGADD, RGSUB, RGNCP	3	0	LIBF	YAI
Multiply	RGMLT	3	0	LIBF	YMV
Divide	RGDIV	3	0	LIBF	YDV
Move Remainder	RGMVR	3	0	LIBF	YMR
Binary Conversion	RGBTD, RGDTB	3	0	LIBF	YCN
RPG Sterling and Edit					
Sterling Input Conversion	RGSTI	3	0	LIBF	YSI
Sterling Output Conversion	RGSTO	3	0	LIBF	YSO
Edit	RGEDT	3	0	LIBF	YED
RPG Move					
From I/O Buffer to Core	RGMV1, RGMV5	3	0	LIBF	YM1
From Core to I/O Buffer	RGMV2	3	0	LIBF	YM2
MOVE Operation	RGMV3	3	0	LIBF	YM3
MOVEL Operation	RGMV4	3	0	LIBF	YM4
RPG Compare					
Alphameric	RGCMP	3	0	LIBF	YAC
RPG Indicators					
Test	RGSI1	3	0	LIBF	YI1
Set Resulting On	RGSI2	3	0	LIBF	YI2
Set On	RGSI3	3	0	LIBF	YI3
Set Off	RGSI4	3	0	LIBF	YI4
Test for 0 or Blank	RGSI5	3	0	LIBF	YI5
RPG Miscellaneous					
Test Zone	RGTSZ	3	0	CALL	YTZ
Convert to Binary	RGCVB	3	0	CALL	YCB
Object Time Error	RGERR	3	0	LIBF	YER
Blank After	RGBLK	3	0	LIBF	YBK
Alternating Sequence (User-Written)	ALTSE	-	-	LIBF	-

INTERRUPT LEVEL SUBROUTINES

INTERRUPT LEVEL TWO SUBROUTINE (ILS02)

This interrupt level subroutine is actually a part of the Skeleton Supervisor. However, the Core Load Builder requires that a dummy ILS for level two be stored in the System Library. The dummy supplied by IBM is stored in the System Library as subtype 1, type 7. The coding in the dummy ILS02 is immaterial, because the Core Load Builder merely bypasses it when it discovers the subtype 1.

If the user supplies his own ILS02, it must be stored in the System Library as subtype 0, type 7.

INTERRUPT LEVEL FOUR SUBROUTINE (ILS04)

This interrupt level subroutine is actually a part of the Skeleton Supervisor. However, the Core Load Builder requires that a dummy ILS for level four be stored in the System Library. The dummy supplied by IBM is stored in the System Library as subtype 1, type 7. The dummy ILS04 consists only of a nine-word table followed by a zero, as follows:

```
DC    /0434    RESERVED
DC    /0434    RESERVED
DC    /0434    RESERVED
DC    /043E    1,2,3,1
DC    /043D    1,4,0,3
DC    /0438    2,5,0,1
DC    /0735    1,4,4,2
DC    /0436    KEYBOARD/CONSOLE PRINTER
DC    /0437    1,1,3,4/1,0,5,5
DC    0        END-OF-TABLE INDICATOR
```

The leftmost eight bits of each word contain the relative entry point to the ISS for the associated device, and the rightmost eight bits contain @ISTV plus the ISS number. These eight words are used by the Core Load Builder to construct the IBT for interrupt level 4.

If the user supplies his own ILS04, it must be stored in the System Library as subtype 0, type 7.

MAINLINE PROGRAMS

DISK INITIALIZATION PROGRAM (DISC)

The disk initialization program has three basic functions:

- Establishes that the cartridges specified in the *ID record have no more than 3 defective cylinders and that cylinder 0 is not defective
- Changes the cartridge labels as specified in the *ID record
- Initializes portions of sectors 0, 1, and 2 to set up the cartridges specified as non-system cartridges

DISC first reads an *ID record to obtain FROM and TO cartridge IDs. It then reads current cartridge IDs from the master cartridge DCOM (#PCID) and compares them with the FROM IDs specified in the *ID record. If there are any IDs not found, an error message is printed on the principal print device.

DISC next seeks home on all drives to be initialized (up to four), and writes each of 3 patterns to an entire cylinder, one sector at a time. The patterns used, in sequence, are /AAAA, /5555, and /0000.

DISC then reads back each sector and compares it with the pattern written (including the sector address). If no error occurs, DISC writes the next pattern to the same cylinder.

If the error bit of the DSW is set at any time or if the data read does not compare with the pattern written, DISC repeats the write/read sequence for the entire cylinder 50 times, using the same pattern.

If a second error occurs, DISC puts the address of the first sector on the cylinder in which the error occurred in the defective cylinder table.

DISC performs this write/read sequence for each of the 203 cylinders.

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):

Location	Value Inserted
#ANDU	/0200
#BNDU	/0200
#FPAD	/0020
#CIDN	Cartridge ID
#CIBA	/0008
#ULET	/0002

DISC initializes LET (sector 2) as follows:

Word	Contents	
1	/0000	
2	/0020	
3	/0000	
4	/0138	
5	/0000	
6	/7112	The name 1DUMY (in name code)
7	/4528	
8	/0620	

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.

1130 Disk Data File Conversion Program

The following information is designed to assist in understanding the program flowcharts by presenting an overall view of the purpose of each major part of the program.

FLOWCHARTS:

CHART A
CHART B
CHART C
CHART D
CHART E
CHART F
CHART G

PROGRAM NAME: DFCNV

GENERAL PROGRAM DESCRIPTION: The program converts one 1130 FORTRAN and/or Commercial Subroutine Package (1130-SE-25X) disk data file to one 1130 RPG disk data file. FORTRAN files created using logical unit number 10 cannot be converted by DFCNV. Converted files cannot be processed as ISAM files. The program accepts all FORTRAN and Commercial Subroutine Package (CSP) disk data formats and a two-word integer format (see Appendix E). The input data may be a disk data file or the corresponding cards in card data format. All printing is performed on the principal printer. The subroutine DISK1 is used to perform all disk I/O operations.

PART 1:

ENTRY POINT: FC000 (CHART A)

The system device subroutines for the principal input and print devices and input data conversion are read into core and pertinent interrupt pointers are set.

- The File Description card (D in column 72) is read and printed, and its fields are diagnosed for errors.
- LET/FLET searches are performed for input (if disk file input specified) and output files and calculated file sizes are checked against actual file sizes.

INTERNAL SUBROUTINES:

FC015 - The card read and/or print function in this section of the program is not specifically subroutinized, but it is the general card input function for the entire program.

CONVT- Subroutine which converts a right-justified EBCDIC coded decimal field of variable length to a one word binary value and advances the field pointer beyond the field just converted. It accounts for leading blanks in a File Description (D) card field and causes immediate program termination when a D-card field error is detected.

SEARC- Subroutine which checks the file name referenced by the field pointer for validity, packs the file name, adds the disk data format indicator to the packed file name and performs the LET/FLET search for the file.

ERRORS DETECTED: The errors detected in part 1 are F01, F02, F03, F06 and F08 (see Appendix F).

PART 2:

ENTRY POINT: FC016 (CHARTS B and C)

- The Field Specification cards (S in column 72) are read and printed, each field specification is diagnosed for errors and the specification information is compressed and saved.
- If it is present, the Commercial Subroutine Package A3 format translation table (A in column 72) is read and printed and the 40 translation characters are saved.
- The end-of-file card (/ * in column 1 and 2) is read and printed.

Note: A general flowchart of the compress/save operation described above has been provided in Chart C although this operation is in fact specific to field type. See Appendix for a description of each field type compression.

INTERNAL SUBROUTINE:

CONVT: Subroutine is described in part 1.

ERRORS DETECTED: The errors detected in part 2 are F04, F06 and F07. It is noted that only one F04 message is printed for each field specification in error although more than one error may occur within a field specification.

PART 3:

ENTRY POINT: FC026 (CHART D)

- Final error checking is performed and program

termination is effected if any control card or specification errors have occurred.

- The input data is read, converted and inserted into the RPG data file.
- The operation complete message is printed and control is returned to the supervisor.

INTERNAL SUBROUTINE:

CDDSK - Subroutine which converts card data formatted data to disk data format when card input is specified and relocates input data so that it appears to be disk data.

Each of the field conversion subroutines is discussed under part 4.

ERRORS DETECTED: The errors detected in part 3 are F05, F06 and F09.

NOTE: The Disk File protection deliminates $FPAD - $FPAD+4 of COMMA are modified during the conversion portion of DFCNV. These modified COMMA words must be restored prior to further processing if unforseen problems (accidential immediate stop) cause abnormal termination of DFCNV These words are restored by DFCNV under normal program termination following successful conversion.

PART 4

FIELD CONVERSION SUBROUTINE (CHARTS E, F and G):

- These subroutines are designed to retrieve the input field and use the specifications contained in the compression area to convert the field.
- The internal subroutine INSRT (described below and on CHART G) is called by each conversion subroutine to place the converted field into the output record. Control is then returned to the mainline part of the program (part 3) to convert the next field.

I-conversion (ICNVT): The sign of the integer is saved and its absolute value is converted to 5 decimal digits (by repeated division by 10). Each digit is placed with positive zone (F) into a 31 word work area which has been initialized to /F0. The field is then placed in the RPG record.

J-conversion (JCNVT): The sign of the integer is saved and its absolute value is converted to 10 decimal digits (by repeated division by 10). The decimal conversion is performed by first determining the range of the two-word integer, converting the low order 7 decimal digits, then converting the high order 3 decimal digits after adjusting for the range factor. Each digit is placed with positive zone (F) into a 31 word work area which has been initialized to /F0. The field is then placed in the RPG record.

D-conversion (DCNVT): The sign of the CSP D1 field is saved (from the low order of the field) and the ones' complement of the low order word is taken. Each digit (see D1 data format, Appendix E) is placed with positive zone (F) into a 31 word work area which has been initialized to /F0. The field is then placed in the RPG record.

E-conversion (ECNVT): The CSP D4 field (see Appendix E) is converted to D1 format and placed in a 200 work work area. The subroutine DCNVT is then used to place the field in the RPG record.

B(C)-conversion (BCNVT): The subroutine INSRT is called to insert the field into the RPG record (not shown on charts).

X-conversion (XCNVT): The input data pointer is adjusted to bypass the number of words specified (not shown on charts).

F-conversion (FCNVT): Each word of the input field is decoded into 3 translation table displacements by means of the following formulae.

$I > 0$, $N1=(I/1600)+20$
$I > 0$, $N1=(I+32000)/1600$
$N2=(I-(N1-20)*1600)/40$
$N3=I-(N1-20)*1600-(N2*40)$

where I represents the input word and N1, N2 and N3 represent the translation table displacements relative to card column 40 of the first, second and third characters.

The 3 translation characters are then retrieved and placed in the RPG record. This procedure continues until the entire input field has been converted.

R-conversion (RCNVT): The real number is loaded into the FAC and normalized. If the mantissa is zero, the 31 word work area (which has been initialized to /F0) is used to fill the RPG field and control is returned to the mainline part of DFCNV. If the mantissa is non-zero, the sign of the field is saved and the absolute value of the mantissa is used in conjunction with FBTN2 and MPY to obtain the decimal digits. Each digit is placed with positive zone (F)

into a 31 word work area which has been initialized to /F0. The field is then placed in the RPG record.

The RPG field is set to zeroes or nines if the real number being converted is too small or too large respectively to fit into the RPG field.

INTERNAL SUBROUTINES:

INSRT - Subroutine which places the converted field into the RPG record. It uses the technique of placing one eight-bit character (or two four-bit packed digits) into the RPG field at a time, alternating between the left and right halves of a word. If a numeric field is packed, two decimal digits are retrieved at a time; if unpacked, single digit retrieval is used. Alphabetic retrieval corresponds to that of numeric unpacked except that characters are returned from units of 1, 2 or 3 words rather than a one character per word work area. Once a numeric field has been retrieved, the sign sign is added to the Field (F=positive; D=negative). See DFCNV field type examples section of IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide, Form C26-Form C26-3717.

FBTN2 - Subroutine which generates the first decimal digit and the decimal exponent of a real number field (adapted from RBTD subroutine, see IBM 1130 Subroutine Library, Form C26-5929).

MPY - Subroutine which generates the decimal digits of a real number field (adapted from FBTD subroutine, see IBM 1130 Subroutine Library, Form C26-5929).

EXTERNAL SUBROUTINES:

FLD- Subroutine which loads a standard precision floating point number into the FAC (see IBM 1130 Subroutine Library, Form C26-5929).

ELD- Subroutine which loads an extended precision floating point number into the FAC (see IBM 1130 Subroutine Library, From C26-5929).

NORM- Subroutine which normalizes the number in the FAC (see IBM 1130 Subroutine Library, Form C26-5929).

ERRORS DETECTED: The F10 error is detected in the R-conversion subroutine RCNVT.

SYNCHRONOUS COMMUNICATIONS ADAPTER SUBROUTINE (SCAT1)

Call Processing

The call processing routine

- Checks the call parameters for errors
- Sets up the processing of subsequent interrupts depending on the operation requested by the call
- Initiates the requested operation

Operation

When entered via a LIBF, SCAT1

1. saves XR1, XR2, accumulator and extension, and status
2. examines the control parameter to determine the operation requested
3. obtains and saves the I/O area address and error routine address parameters if the operation is an Open, Transmit, or Acknowledge and Receive and Error Statistics
4. determines, if the operation is a Transmit or Acknowledge and Receive, from digit 3 of the control parameter if ILRCs will be present in transmitted or received data records (digit 3 non-zero indicates ILRCs will be present, zero indicates ILRCs will not be present)

Test. If the operation is a Test, SCAT1

1. checks the routine busy indicator (RTBSY)
2. returns to the calling routine at LIBF+3 if RTBSY is off (zero)
3. returns to the calling routine at LIBF+2 if RTBSY is on (non-zero)

Auto Answer. If the operation is the enabling/disabling of the auto answer interrupt, SCAT1

1. (if digit 4 of the control parameter is zero) enables the auto answer interrupt, saves the I/O area address, and returns to the calling routine at LIBF+3
2. (if digit 4 of the control parameter is non-zero) disables the auto answer interrupt and returns to the calling routine at LIBF+2

Alarm. If the operation is the turning on/off of the audible alarm, SCAT1

1. turns the audible alarm on in the local system if digit 4 of the control parameter is zero
2. turns the audible alarm off in the local system if digit 4 of the control parameter is non-zero
3. returns to the calling routine at LIBF+2.

Close. If the operation is a Close, SCAT1

1. resets the SCA
2. clears the programmed indicators in SCAT1
3. decrements the $SCAT counter by 1 (word 17 in COMMA)
4. returns to the calling routine at LIBF+2

Open. If the operation is an Open, SCAT1

1. determines the requested mode of operation from digit 4 of the control parameter (digit 4 zero indicates Data In, non-zero indicates Data Out)
2. puts the addresses of the read response, write response, and timeout routines for the Open operation into the interrupt branch addresses
3. begins transmission of the IDLE character for 1.5 seconds
4. increments the $SCAT counter by 1
5. sets the retry counter to 7
6. sets RTBSY on (non-zero)
7. returns to the calling routine at LIBF+4

Transmit. If the operation is a Transmit, SCAT1

1. puts the addresses for the read response, write response, and timeout routines for the Transmit operation into the interrupt branch addresses
2. determines the type of transmission from digit 2 of the control parameter (digit 2 equal to 0 indicates transmission of a data record, 1 indicates transmission of the End of Transmission sequence, and 2 indicates transmission of the Telephone sequence)
3. performs a Start Write (puts the SCA in the transmit mode and causes a write response interrupt)
4. sets the retry counter to 7
5. sets RTBSY on (non-zero)
6. returns to the calling routine at LIBF+4

Acknowledge and Receive. If the operation is an Acknowledge and Receive, SCAT1

1. puts the addresses of the read response, write response, and timeout routines for the Receive operation into the interrupt branch addresses
2. determines the type of acknowledgement to be transmitted from digit 2 of the control parameter (digit 2 zero indicates a positive acknowledgement, non-zero indicates a negative acknowledgment)
3. performs a Start Write (puts the SCA in the transmit mode and causes a write response interrupt)
4. sets the retry counter to 7
5. sets RTBSY on (non-zero)
6. returns to the calling routine at LIBF+4

Errors

When SCAT1 detects an error in the calling processing, a code indicating the error found is placed in the accumulator, the address of the LIBF is placed into location 40 in COMMA, and a branch is taken to location 41, a WAIT. When PROGRAM START is pressed, a branch is taken to the address in location 40 (i.e., the LIBF is re-executed).

Interrupt Processing

The interrupt processing routine

- Handles read response, write response, and timeout interrupts for the Open, Transmit, and Receive operations
- Handles the auto answer request interrupt
- Maintains an error statistics log of certain error
- Is able to log timeouts and all characters transmitted and received

Operation

The paragraphs below describe the processing performed by SCAT1 to accomplish an entire operation (Open-Data In, Open-Data Out, Transmit, or Acknowledge and Receive). Actually, this processing is accomplished in a series of interrupts initiated by the user's call to SCAT1.

An interrupt occurs for each character received or transmitted. SCAT1, by means of programmed indicators, keeps track of the characters to be received or transmitted next. After transmission or reception of each character, SCAT1 returns to the interrupted program.

At the completion of an entire operation, a turnaround of the line occurs and IDLE characters are transmitted for 1.5 seconds. During this 1.5 seconds the user must call SCAT1, specifying the next operation to be performed.

When entered from ILS01 due to an interrupt by the SCA, SCAT1 senses the device status word to determine the type of interrupt.

Write Response Interrupt. If a write response interrupt has occurred, SCAT1

1. branches to the write response routine pointed to by the interrupt response addresses as set up in the call processing

Read Response Interrupt. If a read response interrupt has occurred, SCAT1

1. checks to determine if a simultaneous timeout interrupt has occurred
2. branches to the timeout routine pointed to by the interrupt response addresses as set up in the call processing if a simultaneous timeout interrupt has occurred
3. reads the character received by the SCA into core storage if no simultaneous timeout interrupt has occurred
4. branches to the read response routine pointed to by the interrupt response addresses as set up in the call processing

Timeout Interrupt. If a timeout interrupt has occurred, SCAT1

1. branches to the timeout routine pointed to by the interrupt response addresses as set by the call processing

Auto Answer Interrupt. If an auto answer interrupt has occurred, SCAT1

1. determines from bit 6 of the DSW if the auto answer interrupt is disabled/enabled (bit 6 zero indicates disabled, 1 enabled)
2. returns to ILS01 if the auto answer interrupt is disabled
3. stores a non-zero value at the I/O area address if the auto answer interrupt is enabled
4. disables the auto answer interrupt
5. returns to ILS01

Write Response (Open). If the mode of operation is Data In, the write response routine

1. transmits the End of Idle sequence
2. performs a Start Read (puts the SCA in the

receive mode and causes a read response interrupt)

If the mode of operation is Data out, the write response routine

1. transmits the End of Idle sequence
2. performs a Start Read
3. (when the read response routine has received the End of Idle sequence in response to the End of Idle sequence) transmits the Inquiry sequence
4. performs a Start Read

Read Response (Open). If the mode of operation is Data In, the read response routine

1. (when the write response routine has transmitted the End of Idle sequence) expects to receive the Inquiry sequence
2. (if the Inquiry sequence is received) sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds (the Open-Data In operation is complete)
3. (if the End of Idle sequence is received) turns the line around and begins transmission of the IDLE character for 1.5 seconds
4. (if the Inquiry sequence is not received) branches to the timeout routine

If the mode of operation is Data Out, the read response routine

1. (when the write response routine has transmitted the End of Idle sequence) expects to receive the End of Idle sequence
2. (if the End of Idle sequence is received) turns the line around and begins transmission of the IDLE character for 1.5 seconds
3. (if the End of Idle sequence is not received) branches to the timeout routine
4. (when the End of Idle sequence has been received and the write response routine has transmitted the Inquiry sequence) expects to receive the CL character followed by the ACK2 character
5. (if the CL character followed by the ACK2 character is received) sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds (the Open-Data Out operation is complete)
6. (if the CL character is not received, or if the CL character is not followed by the ACK2 character) branches to the timeout routine

Timeout (Open). If the mode of operation is data in, the timeout routine

1. branches to the user's error routine if a data set failure has occurred
2. (on the first through seventh entries to this routine) decrements the retry counter by 1, begins transmission of the IDLE character for 1.5 seconds, and returns to the interrupted routine while another attempt is made to receive the expected sequence
3. (on the eighth entry to this routine) branches to the user's error routine if synchronization has not been established (after 7 attempts the End of Idle sequence was not received in response to the End of Idle sequence)
4. (on the eighth entry to this routine) branches to the user's error routine if nothing is received in response to the End of Idle sequence
5. (on return from the user's error routine) resets the retry counter to 7 if the accumulator is non-zero and returns to ILS01
6. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero and returns to ILS01

If the mode of operation is Data Out, the timeout routine

1. branches to the user's error routine if a data set failure has occurred
2. (on the first through seventh entries to this routine) decrements the retry counter by 1, begins transmission of the IDLE character for 1.5 seconds, and returns to the interrupted routine while another attempt is made to receive the expected sequence
3. (on the eighth entry to this routine) branches to the user's error routine if synchronization has not been established (after 7 attempts the End of Idle sequence was not received in response to the End of Idle sequence)
4. (on the eighth entry to this routine) branches to the user's error routine if nothing is received in response to the End of Idle sequence
5. (on the eighth entry to this routine) branches to the user's error routine if the Inquiry sequence is received in response to the End of Idle sequence or in response to the Inquiry sequence
6. (on return from the user's error routine) resets the retry counter to 7 if the accumulator is non-zero and returns to ILS01
7. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero and returns to ILS01

Write Response (Transmit). If the type of transmission is Transmit Data, the write response routine

1. (following the initial write response interrupt) transmits the Start of Record sequence, using the appropriate start of record character--SOR1 for odd numbered records, SOR2 for even numbered records
2. next, transmits the data characters from the I/O area, building and transmitting ILRCs, if required
3. and finally, transmits the End of Record sequence at the depletion of the data word count
4. performs a Start Read (puts the SCA in the receive mode and causes a read response interrupt)

If the type of transmission is Transmit EOT, the write response routine

1. (following the initial write response interrupt) transmits the End of Transmission sequence
2. performs a Start Read

If the type of transmission is Transmit TEL, the write response routine

1. (following the initial write response interrupt) transmits the Telephone sequence
2. performs a Start Read

Read Response (Transmit). If the type of transmission is Transmit Data, the read response routine

1. (when the write response routine has transmitted the Start of Record sequence, using the appropriate start of record character, followed by the data characters, followed by the End of Record sequence) expects to receive an acknowledgement including the appropriate acknowledgement character--ACK1 for odd numbered records, ACK2 for even numbered records--or the Error Received sequence
2. (if the appropriate acknowledgement is received) turns the line around, and begins transmission of the IDLE character for 1.5 seconds (the Transmit Data operation is complete)
3. (if the Error Received sequence is received) decrements the retry counter by 1 and re-issues the initial Start Write to transmit the data record
4. (if, after 7 attempts to transmit the data record, the appropriate acknowledgement is not received) branches to the user's error routine
5. (on return from the user's error routine) resets the retry counter to 7 and re-issues the initial Start Write to transmit the data record if the accumulator is positive
6. sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds if the accumulator is negative (the Transmit Data operation is complete)
7. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero
8. (if nothing is received in response to the transmitted data record) a timeout occurs
9. (if the something other than the appropriate acknowledgement or the Error Received sequence is received) decrements the retry counter by 1 and issues the Start Write to transmit the Inquiry sequence; then turns the line around and expects to receive the appropriate acknowledgement

If the type of transmission is Transmit EOT, the read response routine

1. (when the write response routine has transmitted the End of Transmission sequence) expects to receive the End of Transmission sequence
2. (if the End of Transmission sequence is received) sets RTBSY off (zero), turns the line around, and begins the transmission of the IDLE character for 1.5 seconds (the Transmit EOT operation is complete)
3. (if the End of Transmission sequence is not received) decrements the retry counter by 1 and re-issues the initial Start Write to transmit the End of Transmission sequence
4. (if, after 7 attempts to transmit the End of Transmission sequence, the End of Transmission is not received in response) branches to the user's error routine
5. (on return from the user's error routine) resets the retry counter to 7 and re-issues the initial Start Write to transmit the End of Transmission sequence if the accumulator is positive
6. sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds if the accumulator is negative (the Transmit EOT operation is complete)
7. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero

If the type of transmission is Transmit TEL, the read response routine

1. (when the write response routine has transmitted the Telephone sequence) expects to receive the Telephone sequence in response

2. (if the Telephone sequence is received) sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds (the Transmit TEL operation is complete)
3. (if the Telephone sequence is not received) decrements the retry counter by 1 and re-issues the initial Start Write to transmit the Telephone sequence
4. (if, after 7 attempts to transmit the Telephone sequence, the Telephone sequence is not received in response) branches to the user's error routine
5. (on return from the user's error routine) resets the retry counter to 7 and re-issues the initial Start Write to transmit the Telephone sequence if the accumulator is positive.
6. sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds if the accumulator is negative (the Transmit TEL operation is complete)
7. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero

Timeout (Transmit). If a timeout interrupt occurs, the timeout routine

1. branches to the read response routine to handle the timeout as if an invalid response was received

Write Response (Receive). The write response routine

1. (if a data record was received in the last Acknowledge and Receive operation) transmits an acknowledgement, including the appropriate acknowledgement character--ACK1 for odd numbered records, ACK2 for even numbered records (if the acknowledge is to be positive); then performs a Start Read (puts the SCA in the receive mode and causes a read response interrupt)
2. (if a data record was received in the last Acknowledge and Receive operation) transmits the Error Received sequence (if the acknowledgement is to be negative); then performs a Start Read
3. (if the End of Transmission sequence has been received) transmits the End of Transmission sequence; then performs a Start Read
4. (if the Telephone sequence has been received) transmits the Telephone sequence; then performs a Start Read

Read Response (Receive). The read response routine

1. (after the appropriate acknowledgement has been transmitted) expects to receive (1) the Start of Record sequence, followed by the data record (including ILRCs, if required), followed by the End of Record sequence, (2) the End of Transmission sequence, (3) the Telephone sequence, or (4) the Inquiry sequence
2. (while receiving data characters) builds an LRC to check against the ILRC/LRC received
3. (if 1 above is received) compares the LRC built against the ILRC/LRC and indicates an error in the data record if they are not identical
4. (if 1 without errors, or 2 above is received) sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds (the Acknowledge and Receive operation is complete)
5. (if 3 above is received) branches to the user's error routine
6. (on return from the user's error routine) transmits the Telephone sequence; then performs a Start Read (puts the SCA in receive mode and causes a read response interrupt)
7. (if 4 above is received) turns the line around, re-transmits the appropriate acknowledgement; then performs a Start Read
8. (if characters other than one of the above--1, 2, 3, or 4--or 1 above, with errors, are received) decrements the retry counter by 1, turns the line around, and transmits the Error Received sequence; then performs a Start Read
9. (if, after 7 attempts, one of 1, 2, 3, or 4 above is not received) branches to the user's error routine
10. (on return from the user's error routine) resets the retry counter to 7 and re-transmits the Error Received sequence attempting again to receive one of 1, 2, 3, or 4 above, if the accumulator is positive
11. sets RTBSY off (zero), turns the line around, and begins transmission of the IDLE character for 1.5 seconds if the accumulator is negative (the Acknowledge and Receive operation is complete)
12. performs a Close operation (see "Close" under Call Processing, above) if the accumulator is zero
13. (if nothing is received in response to the last acknowledgement transmitted) a timeout occurs

Timeout (Receive). The timeout routine

1. (if nothing is received in response to the last acknowledgement transmitted) branches to the

read response routine to handle the timeout as if an invalid response was received

SYNCHRONOUS COMMUNICATIONS ADAPTER SUBROUTINE (SCAT2)

Call Processing

Chart: GA

The call processing portion of SCAT2

- Checks the call parameters for errors. SCAT2 exits to the pre-operative error trap (word 40_{10} in COMMA) on any errors detected.
- Performs an Auto Answer, Alarm, Test, or Close operation.
- Sets up, according to the call parameters, the required switches and storage locations. These switches and storage locations are used by the interrupt processing portion of SCAT2 to perform the requested operation during subsequent interrupts.
- Initiates a Receive, Transmit Block, Transmit Text, or Transmit End operation.

When entered via LIBF, SCAT2 saves the contents of index registers 1 and 2, the accumulator, the extension, and the status indicators. SCAT2 then performs the processing described below for the various operations.

Test

If RTBSY is non-zero, SCAT2 returns to the calling routine at LIBF+2. If RTBSY equals zero, SCAT2 returns to the calling routine at LIBF+3.

Auto Answer

According to digit 3 of the control parameter, SCAT2 either disables the auto answer interrupt and returns to the calling routine at LIBF+2 or enables the auto answer interrupt, saves the I/O area address parameter at ANS, and returns to the calling routine at LIBF+3.

Alarm

According to digit 3 of the control parameter, SCAT2 either turns the audible alarm on or turns it off. In either case, SCAT2 returns to the calling routine at LIBF+2.

Close

SCAT2 resets the SCA, clears the appropriate switches and storage locations, disconnects the SCA from the communications line, and returns to the calling routine at LIBF+2.

Receive, Transmit Block, Transmit Text, Transmit End

SCAT2, according to the call parameters, sets up the switches and storage locations required for the requested operation (see Switches and Storage Locations). In all cases the idle register is loaded with the SYN character. The acknowledgements are initialized for a Receive Initial or Transmit Initial operation. The SCA is placed in the receive mode for a Receive Initial operation; otherwise, it is placed in the synchronize mode. SCAT2 then sets RTBSY non-zero and returns to the calling routine at LIBF+4.

Error Statistics

The address to an error statistics log is given to the user.

Options

According to digit 2 of the control parameter, the options specified in digit 4 of the control parameter are either enabled or disabled.

Interrupt Processing

Charts: GB, GC

The interrupt processing portion of SCAT2

- Handles read response, write response, and time-out interrupts for Receive, Transmit Block, Transmit Text, and Transmit End operations according to the pertinent switches and storage locations.
- Handles the auto answer request interrupt.
- Checks for errors in the data and communication sequences received. SCAT2 exits to the user's error routine to process any errors detected.
- Maintains an error statistics log of certain errors.
- Logs timeouts and all characters transmitted and received.

During the series of interrupts initiated by the call processing portion of SCAT2, the interrupt processing portion performs the Receive, Transmit Block, Transmit Text, or Transmit End operation according to the switches and storage locations set up in the call processing. An interrupt occurs (1) when a character has been received in or transmitted by the SCA, (2) on a timeout of the receive (3 second) timer, (3) on a timeout of the transmit (1.5 second) timer used during transmission of Normal EBCDIC text, (4) on a timeout of the program (approximately .22 second) timer used during transmission of Full-Transparent text, or (5) on a timeout of the program timer during a Receive Initial operation if the program timer is specified to be used. The interrupt processing consists of updating the switches and storage locations(see Switches and Storage Locations), error checking, and reception or transmission of the next message or control character, as appropriate. After processing an interrupt, SCAT2 returns to the interrupted program.

When entered from ILS01 due to an interrupt by the SCA, SCAT2 senses and resets the device status word (DSW) to determine the type of interrupt. SCAT2 interrogates the DSW bits in the following order: auto answer request (bit 4), ready (bit 7), read response (bit 0), write response (bit 1), and timeout (bit 3). According to the DSW bits SCAT2 performs the processing described below for the various types of interrupt.

Auto Answer Request

If the auto answer request interrupt is disabled, SCAT2 returns to ILS01. If it is enabled, SCAT2 stores a non-zero value at the address found in ANS, disables the auto answer request interrupt, and normally returns to ILS01. If an exit to a user's routine is specified in an optional call to SCAT2, this exit with the accumulator equal to zero is taken before the return to ILS01.

Ready

The ready bit of the DSW must equal one in order for SCAT2 to process a read response, write response, or timeout interrupt. If the ready bit equals zero, an error is indicated and SCAT2 branches to the user's error routine.

Read Response

In the receiving station, read response interrupts occur as message characters are received; in the transmitting station, read response interrupts occur as acknowledgements to the messages transmitted are received.

SCAT2 reads the character received by the SCA into storage at BUF. Then, according to the switches and storage locations set up in the call processing or in previous interrupt processing, SCAT2 transfers a message character to the user's I/O area or determines which control character was received. SCAT2 tests, sets, and/or resets the switches and storage locations depending on the character received and returns to ILS01.

During the reception of a message SCAT2 resets the 3-second timer each time the synchronous idle sequence is received to prevent an erroneous receive timeout.

Write Response

In the receiving station, write response interrupts occur as acknowledgements to the messages received are transmitted; in the transmitting station, write response interrupts occur as message characters are transmitted.

SCAT2 determines from the switches and storage locations set up in the call processing or in previous interrupt processing the message or control character to be transmitted. SCAT2 tests, sets, and/or resets the switches and storage locations depending on the character to be transmitted. SCAT2 then loads the character to be transmitted into the SCA for transmission and returns to ILS01.

During the transmission of Full-Transparent text, SCAT2 loads the idle register in the SCA with the DLE character so that the proper synchronous idle sequence (DLE SYN) is transmitted whenever a character gap occurs.

Timeout

SCAT2 tests, sets, and/or resets switches and storage locations depending on the operation in progress and the point at which the timeout occurred. SCAT2 then, if appropriate, performs special processing according to the switches and storage locations and returns to ILS01.

The program timer causes a timeout interrupt every .22 seconds (approximately) during the transmission of Full-Transparent text. This interrupt does not occur simultaneously with a write response interrupt. As the result of this interrupt SCAT2 sets TOIND non-zero, causing the synchronous idle sequence (DLE SYN) to be transmitted on the next two write response interrupts.

The transmit timer causes a timeout interrupt every 1.5 seconds during the transmission of

Normal EBCDIC text. This interrupt occurs simultaneously with a write response interrupt. As the result of this interrupt SCAT2 sets TOIND non-zero, causing the synchronous idle sequence (SYN SYN) to be transmitted.

The receive timer determines the maximum time (3 seconds) that (1) a transmitting station will wait for a reply or (2) a receiving station will receive message characters between synchronous idle sequences. The receive timer is reset and restarted each time the synchronous idle sequence is received. As the result of a receive timer interrupt, SCAT2 sets XMENQ non-zero, causing the transmitting station to transmit ENQ next.

The program timer causes a timeout interrupt during a Receive Initial operation if no ENQ is received and the program timer is specified to be used.

Switches and Storage Locations

ACK and ACK+1

Use/Contents. Storage locations containing the alternating positive acknowledgements. ACK contains the current acknowledgement; ACK+1 contains the alternative acknowledgement.

Notes. During the call processing for a Receive/Transmit Initial or Transmit End operation, ACK and ACK+1 are loaded with ACK0 and ACK1, respectively. During interrupt processing the contents of ACK and ACK+1 are exchanged following the successful completion of each Receive Continue, Transmit Block, or Transmit Text operation.

ANS

Use/Contents. A storage location containing the address of the location where the auto answer indication is to be stored.

Notes. During the call processing for an Auto Answer Enable operation, the I/O area address parameter is saved at ANS. During interrupt processing the location whose address is contained in ANS is set non-zero whenever an auto answer request interrupt occurs.

BCCA

Use/Contents. A storage location containing the CRC-16 accumulated for the message just transmitted/received.

Notes. The CRC-16 is accumulated in BCCA by the CALC subroutine.

BCCR

Use/Contents. A storage location containing the CRC-16 received following the message just received.

Notes. The CRC-16 received in BCCR is compared to the CRC-16 accumulated by the receiving station in BCCA to determine whether or not the message was received correctly.

BCC1

Use. A switch indicating in a Transmit operation which half of the CRC-16 is being transmitted; in a Receive operation that the next character to be received is the CRC-16.

Settings. For a Transmit operation,

Zero = Transmit the second half of the CRC-16
Non-zero = Transmit the first half of the CRC-16

For a Receive operation,

Zero = negative of non-zero
Non-zero = CRC-16 is next character to be received

Notes. BCC1 is set non-zero in a Receive operation when the end character (ETB or ETX) has been received or in a Transmit operation when the number of characters to be transmitted (see COUNT) has been decremented to zero. BCC1 is set to zero after the entire CRC-16 has been received (Receive operation) or after the first half of the CRC-16 has been transmitted (Transmit operation).

BCC2

Use. A switch indicating which half of the CRC-16 is being received.

Settings.

Zero = Receive the first half of the CRC-16
Non-zero = Receive the second half of the CRC-16

Notes. BCC2 is set non-zero after the first half of the CRC-16 has been received. BCC2 is set to zero after the second half of the CRC-16 has been received.

BUF

Use/Contents. A storage location through which message characters pass when being transferred from the I/O area to the SCA or from the SCA to the I/O area.

Notes. The character in BUF is the character received by the SCA that caused the last read response interrupt or the character to be transmitted next by the SCA.

CLOSE

Use. A switch indicating that a Close operation is to be performed if no response is received to a transmitted EOT.

Settings.

Zero = Close if no response to EOT
Non-zero = Do NOT Close if no response to EOT

Notes. CLOSE is set during the call processing for a Transmit End operation according to digit 4 of the control parameter.

COUNT

Use/Contents. A storage location containing the number of characters transmitted from or the number of characters that were stored into the user's I/O area.

Notes. The number in COUNT is incremented during interrupt processing as characters are transmitted/received. At the end of a Transmit operation, COUNT equals the number of characters transmitted. At the end of a Receive operation, COUNT equals the number of characters stored in the I/O area.

DLSTX

Use. A switch indicating that the DLE STX sequence, which precedes a message or portion of a message in Full-Transparent text, has not yet been encountered in the message being transmitted.

Settings.

Zero = negative of non-zero
Non-zero = DLE STX sequence not yet encountered

Notes. DLSTX is set non-zero if the first character of a message in Full-Transparent text is not DLE. DLSTX is set to zero after the character following the first DLE (i.e., STX) has been transmitted.

DSW

Use/Contents. A storage location containing the device status word (DSW) saved following the last interrupt.

EOTRP

Use/Contents. A switch indicating whether a read response should be performed after transmitting an EOT or not.

Settings.

Zero = Read response to EOT
Non-zero = Do NOT Read response to EOT

Notes. EOTRP is set during call processing for a Transmit End operation according to digit 3 of the control parameter.

ERRU

Use/Contents. A storage location containing the error code to be placed in the accumulator before branching to the user's error routine.

Notes. ERRU contains the error code for the post-operation error detected by SCAT2.

FCODE

Use. A switch indicating the function to be performed.

Settings.

Negative = Receive
Zero = Transmit End
Positive = Transmit Block/Text

Notes. FCODE is set during call processing according to digit 1 of the control parameter.

FIRST

Use. A switch indicating whether or not the first character has been transmitted from or received in the user's I/O area.

Settings.

Zero = First character transmitted/received
Non-zero = First character not yet transmitted/received

Notes. FIRST is set non-zero during call processing. FIRST is set to zero during interrupt processing when the first character is transferred to or from the user's I/O area.

IOAR

Use/Contents. A storage location containing the address of the user's I/O area.

Notes. The I/O area address parameter is saved at IOAR during call processing.

LSDLE

Use. A switch indicating that the last character transmitted or received was DLE.

Settings.

Zero = negative of non-zero
Non-zero = Last character was DLE

Notes. LSDLE is set to zero during call processing; it is set to zero during interrupt processing whenever it is non-zero and the character being transmitted is not DLE. LSDLE is set non-zero during interrupt processing when DLE is transmitted or received.

NXTPD

Use/Contents. A switch indicating that a pad (FF_{16}) character is to be transmitted next.

Settings.

Zero = Do not transmit the pad character
Non-zero = Transmit the pad character

Notes. NXTPD is set non-zero after each turnaround character or sequence has been transmitted. NTXPD is set to zero after the pad character has been transmitted.

OPERR

Use/Contents. A storage location containing the address of a user's routine which is to be entered as specified in an optional call to SCAT2.

Notes. The address parameter is saved at OPERR during call processing.

OPTSW

Use/Contents. A switch indicating which user options are active.

Settings. The options are indicated by bit switches.

Bit 13 = 1 A no-error exit to a user's routine is provided (acc = 0000_{16}) when the last interrupt of an operation has been serviced.

Bit 14 = 1 An immediate exit to a user's routine is provided (acc = 0020_{16}) when a timeout occurs prior to receiving an ENQ on a Receive Initial operation using the third timer.

Bit 15 = 1 An immediate exit to a user's routine is provided (acc = 0020_{16}) when a timeout occurs prior to receiving an ENQ on a Receive Initial operation using the normal 3-second timer.

Notes: The bit switches are set and reset during call processing.

OVFLO

Use. A switch indicating that the message received exceeds the size of the user's I/O area.

Settings.

Zero = negative of non-zero
Non-zero = Overflow of the user's I/O area has occurred

Notes. OVFLO is set non-zero during interrupt processing when the character just received cannot be stored in the user's I/O area (i.e., COUNT is now equal to WDCNT+1).

PACK

Use/Contents. A switch indicating whether the message to be transmitted/received is specified as unpacked (i.e. one character per word) or packed (two characters per word).

Settings.

Zero = Message specified as unpacked
Non-zero = Message specified as packed

Notes. PACK is set during call processing according to digit 2 of the control parameter.

POINT

Use/Contents. A storage location containing the address of the location within the user's I/O area where the next character to be transmitted is located or where the next character received is to be stored.

Notes. The address in POINT is incremented during interrupt processing as characters are transmitted or received.

RETRY

Use/Contents. A storage location used as the retry counter. RETRY contains the number of times an operation is to be attempted.

Notes. RETRY is set to eight during call processing. RETRY is decremented by one each time an operation is reattempted. When RETRY equals zero, it is reset to 7 and an exit is made to the user's error routine.

RTBSY

Use. A switch indicating whether or not SCAT2 is busy, i.e., is performing a previously initiated operation that has not been completed.

Settings.

Zero = SCAT2 not busy
Non-zero = SCAT2 busy

Notes. RTBSY is set non-zero during the call processing for Receive, Transmit Block, Transmit Text, and Transmit End operations. RTBSY is set to zero during interrupt processing when the operation is completed.

SLVMS

Use. A switch indicating whether this station, in the event of contention, is to be the master station or the slave station.

Settings.

Zero = Master
Non-zero = Slave

Notes. SLVMS is set during the call processing for a Transmit Initial Block/Text operation according to digit 3 of the control parameter.

STXIN

Use. A switch indicating that STX is the next character to be transmitted or received.

Settings.

Zero = negative of non-zero
Non-zero = Next character to be transmitted/received is STX

Notes. STXIN is set non-zero when the first DLE character of a message in Full-Transparent text has been transmitted/received. STXIN is set to zero after the character following the first DLE (i.e., STX) has been transmitted/received.

SUBF

Use. A switch indicating the sub-function of the function to be performed.

Settings.

Negative = Receive/Transmit Initial or Transmit EOT
Zero = Receive/Transmit Continue or Transmit DLE EOT
Positive = Receive Repeat

Notes. SUBF is set during call processing according to digit 2 of the control parameter.

SYN2

Use. A switch indicating that the synchronous idle sequence is to be transmitted next.

Settings.

Zero = negative of non-zero
Non-zero = Insert the synchronous idle sequence following a transmit timeout

Notes. SYN2 is set to zero during the call processing for a Transmit Initial or Transmit Continue operation; it is set to zero during interrupt processing after the synchronous idle sequence has been transmitted. SYN2 is set non-zero during interrupt processing when a timeout occurs during transmission.

SYN5

Use/Contents. A storage location containing the number of SYN (padding) characters that are to precede every transmission.

Notes. During call processing SYN5 is set to five. During interrupt processing the number in SYN5 is decremented by one for each SYN character transmitted. When SYN5 equals zero, transmission of the message characters or control sequence follows.

TBTX

Use. A switch indicating whether Block or Text transmission is to be performed.

Settings.

Zero = Block
Non-zero = Text

Notes. TBTX is set during call processing according to digit 1 of the control parameter.

TEND

Use. A switch indicating that the next character to be transmitted is ETB (in a Transmit Block operation) or ETX (in a Transmit Text operation).

Settings.

Zero = negative of non-zero
Non-zero = ETB or ETX is next character to be transmitted

Notes. TEND is set non-zero when all characters of a message in Full-Transparent text has been transmitted. TEND is set to zero after the end character (ETB or ETX) has been transmitted.

TEXTM

Use. A switch indicating the type of text to be transmitted.

Settings.

Zero = Normal EBCDIC text
Non-zero = Full-Transparent text

Notes. TEXTM is set during the call processing for a Transmit Block/Text operation according to digit 4 of the control parameter.

TOIND

Use. A switch indicating that a receive timeout has occurred.

Settings.

Zero = negative of non-zero
Non-zero = Receive timeout has occurred

Notes. TOIND is set non-zero when a receive timeout occurs in the transmitting station while waiting to receive an acknowledgement. TOIND is set to zero when an acknowledgement is received.

TRANS

Use. A switch indicating that Full-Transparent text is being transmitted/received or that the CRC-16 is the next character to be received.

Settings.

Zero = negative of non-zero
Non-zero = Full-Transparent text being transmitted/received, or CRC-16 is next character to be received (if BCC1 is also non-zero)

Notes. TRANS is set to zero during call processing; it is set to zero during interrupt processing after the end character (ETB or ETX) has been transmitted or after an acknowledgement has been transmitted. TRANS is set non-zero when the sequence DLE STX is encountered in transmitting or receiving, or when the end character (ETB or ETX) is received (see BCC1).

USERR

Use/Contents. A storage location containing the address of the user's error routine.

Notes. The error routine address parameter is saved at USERR during call processing.

WDCNT

Use/Contents. A storage location containing the word count of the message for a Transmit operation or the length of the I/O area for a Receive operation, if the message is specified as unpacked; i.e. one character per word. If the message is specified as packed, i.e. two characters per word, the storage location contains the number of characters to be transmitted or the maximum number of characters to be received.

Notes. The contents of the first word of the user's I/O area are obtained and saved at WDCNT during call processing.

WD17I

Use. A switch indicating whether or not the SCAT counter, word 17_{10} in COMMA, has been incremented by one.

Settings.

Zero = Word 17_{10} has been decremented by 1 or has not yet been incremented by 1.
Non-zero = Word 17_{10} has been incremented by 1.

Notes. WD17I is set non-zero during call processing when the first operation other than a Close operation is initiated, at the same time that word 17_{10} is incremented by one. WD17I is set to zero when a Close operation is performed, at the same time that word 17_{10} is decremented by one.

WRACK

Use. A switch indicating that an incorrect acknowledgement was received prior to a receive timeout.

Settings.

Zero = negative of non-zero
Non-zero = Incorrect acknowledgement received prior to a receive timeout

XMENQ

Use. A switch indicating that the next character to be transmitted or the first character to be received must be ENQ.

Settings.

Zero = negative of non-zero
Non-zero = Next character transmitted or first character received must be ENQ

Notes. XMENQ is set non-zero during the call processing for a Receive/Transmit Initial operation; it is set non-zero during interrupt processing whenever a receive timeout occurs while attempting to receive an acknowledgement. XMENQ is set to zero during interrupt processing when ENQ has been transmitted or received.

XMESS

Use. A switch indicating that a message is to be transmitted.

Settings.

Zero = negative of non-zero
Non-zero = Entire message has not yet been transmitted

Notes. XMESS is set non-zero during the call processing for a Transmit Initial or Transmit Continue operation. XMESS is set to zero during interrupt processing when the entire message has been transmitted.

XNAK

Use. A switch indicating that NAK is the next character to be transmitted.

Settings.

Zero = negative of non-zero
Non-zero = Next character to be transmitted is NAK

Notes. XNAK is set non-zero during the call processing for a Receive Repeat operation; it is set non-zero during interrupt processing when an error is detected while receiving a message. XNAK is set to zero during interrupt processing when NAK has been transmitted.

SYNCHRONOUS COMMUNICATIONS ADAPTER SUBROUTINE (SCAT3)

Call Processing

Chart: GD

The call processing performed by the SCAT3 subroutine is identical to that performed by the SCAT2 subroutine, except that (1) SCAT3 does not perform the Auto Answer operation as does SCAT2, (2) no options can be specified as in SCAT2, (3) SCAT3 initiates the Monitor operation in addition to those initiated by SCAT2, and (4) SCAT3, in initiating a Transmit Initial operation, places the SCA in the receive mode initially instead of in the synchronize mode, as in SCAT2.

Monitor

SCAT3 saves the selection address and the polling address specified in the Monitor operation in locations SELA and POLLA, respectively. SCAT3 sets up the

switches and storage locations required for the Monitor operation (see Switches and Storage Locations). The idle register is loaded with the SYN character and the SCA is placed in the receive mode. SCAT3 then returns to the calling routine at LIBF+4.

Interrupt Processing

Charts: GE, GF, GG

The interrupt processing performed by the SCAT3 subroutine is identical to that performed by the SCAT2 subroutine, except that (1) the auto answer request interrupt, processed by SCAT2, is not processed by SCAT3, (2) interrupt processing for the Monitor operation, not found in SCAT2, is performed, and (3) in the case of an error detected during a Monitor operation, the operation is retried indefinitely, and no exit is made to the user's error routine, as in SCAT2.

Switches and Storage Locations

The switches and storage locations used by SCAT3 are identical to those used by SCAT2, except as noted below.

ADDR

Use. A switch indicating that the polling or selection address was received while monitoring.

Settings.

Positive = Polling address received
Negative = Selection address received
Zero = No address received

Notes. If ADDR is set non-zero and the next character received is not ENQ, NOTME is then set non-zero.

ANS

There is no storage location ANS in SCAT3.

CLOSE

There is no switch CLOSE in SCAT3.

CMODE

Use. A switch indicating that SCAT3 is in control mode, i.e., is able to recognize the polling or selection address as such.

Settings.

Zero = negative of non-zero
Non-zero = SCAT3 is in control mode

Notes. CMODE is set to zero during the call processing for a Receive/Transmit Initial or Monitor operation; it is set to zero during interrupt processing whenever an SOH or STX character is received. CMODE is set non-zero during interrupt processing when EOT has been received.

EOTRP

There is no switch EOTRP in SCAT3.

FCODE

Use. A switch indicating the function to be performed.

Settings.

Negative = Receive
Zero = Monitor
Positive = Transmit Block/Text/End

Notes. FCODE is set during call processing according to digit 1 of the control parameter. During interrupt processing FCODE is set to zero if SCAT3 returns to the Monitor operation.

ITBSK

Use. A switch indicating that the next two characters to be received while monitoring are to be ignored.

Settings.

Zero = negative of non-zero
Non-zero = Ignore the next two characters

Notes. ITBSK is set non-zero when an intermediate block check (ITB) character or the sequence DLE ITB is received while monitoring, in order that the two following block check characters not be mistaken for control characters.

LSSYN

Use. A switch indicating that the last character recognized while monitoring Normal EBCDIC text was a SYN character. When the sequence SYN SYN has been recognized while monitoring Normal EBCDIC text, the receive timer is reset.

Settings.

Zero = negative of non-zero
Non-zero = Last character recognized was a SYN character

Notes. LSSYN is set non-zero during interrupt processing whenever a SYN character has been recognized while monitoring Normal EBCDIC text. LSSYN is set to zero during interrupt processing whenever any character other than the SYN character has been recognized while monitoring Normal EBCDIC text or whenever a timeout has occurred.

MONIT

Use. A switch indicating that a Receive Initial or Transmit Initial operation is monitoring for the selection or polling address, respectively.

Settings.

Zero = negative of non-zero
Non-zero = Monitor for selection or polling address

Notes. MONIT is set non-zero during the call processing for a Receive Initial or Transmit Initial operation. MONIT is set to zero when the selection address is received during a Receive Initial operation, or when the polling address is received during a Transmit Initial operation.

NOTME

Use. A switch indicating that, while monitoring, an address was received that did not match the specified polling or selection address.

Settings.

Zero = negative of non-zero
Non-zero = Station's address was not received (but another station's was received)

Notes. NOTME is used to determine whether a response should be transmitted when an ENQ has been received.

OPERR

There is no storage location OPERR in SCAT3.

OPTSW

There is no switch OPTSW in SCAT3.

POLLA

Use/Contents. A storage location containing the polling address specified in the last Monitor operation initiated.

Notes. The polling address is saved at POLLA during call processing.

POLLI

Use/Contents. A storage location containing the address of the location following the location in the user's program that contains the polling address. The address found at POLLI is used by SCAT3 in storing a non-zero value in the location following the polling address whenever that polling address is received, except during a Transmit Initial operation.

Notes. The appropriate address is saved at POLLI during call processing.

SELA

Use/Contents. A storage location containing the selection address specified in the last Monitor operation initiated.

Notes. The selection address is saved at SELA during call processing.

SELI

Use/Contents. A storage location containing the address of the location following the location in the user's program that contains the selection address. The address found at SELI is used by SCAT3 in storing a non-zero value in the location following the selection address whenever that selection address is received, except during a Receive Initial operation.

Notes. The appropriate address is saved at SELI during call processing.

SLVMS

There is no switch SLVMS in SCAT3.

TBTX

Use. A switch indicating whether Block, Text, or EOT transmission is to be performed.

Settings.

Negative = EOT
Zero = Block
Positive = Text

Notes. TBTX is set during call processing according to digit 1 of the control parameter.

TRNSP

Use. A switch indicating the type of text being monitored by SCAT3.

Settings.

Zero = Monitoring Normal EBCDIC text
Non-zero = Monitoring Full-Transparent text

Notes. TRNSP is set non-zero during interrupt processing when DLE STX is received while monitoring. TRNSP is set to zero during interrupt processing when SCAT3 resynchronizes with the transmitting station.

XNAK

Use. A switch indicating that NAK is the next character to be transmitted.

Settings.

Zero = negative of non-zero
Non-zero = Next character to be transmitted is NAK

Notes. XNAK is set non-zero during the call processing for a Receive Repeat operation. XNAK is set non-zero during interrupt processing when an error is detected while receiving a message or when the selection address is received and SCAT3 is not prepared to receive data. XNAK is set to zero during interrupt processing when NAK has been transmitted.

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:

Location	Value Inserted
#ANDU	/0200
#BNDU	/0200
#FPAD	/0020
#CIDN	Cartridge ID
#CIBA	/0008
#ULET	/0002

LET (sector 2) is initialized as follows:

Word	Contents	
1	/0000	
2	/0020	
3	/0000	
4	/0138	
5	/0000	
6	/7112	The name 1DUMY (in name code)
7	/4528	
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 origined at its high-addressed end.

INTRODUCTION

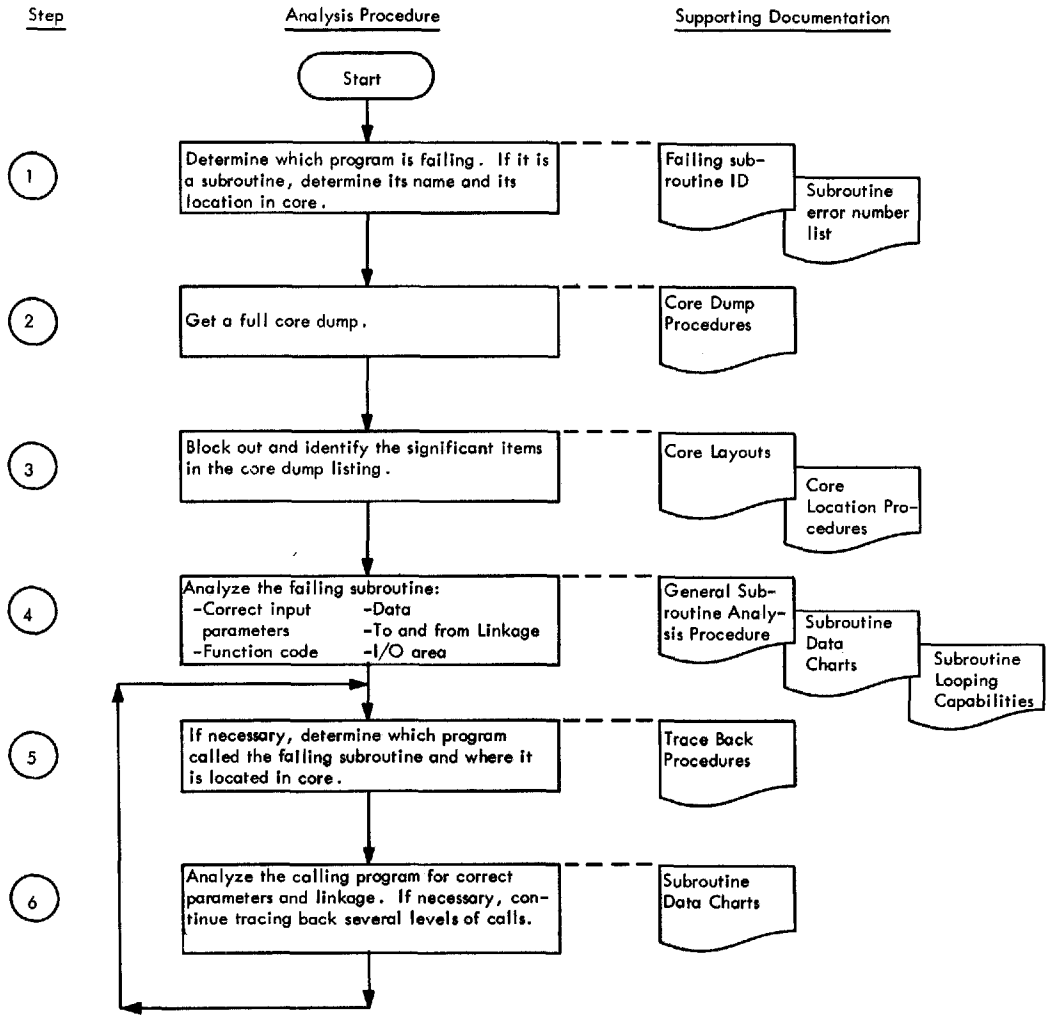
The purpose of the Program Analysis Procedures is to provide the user with a step-by-step method for analyzing the execution of any monitor system or user program. The procedure is problem-oriented; it begins with some program malfunction, assists the user in defining the failing component or function, and provides the facility for detailed analysis of that component or function.

PROGRAM ANALYSIS PROCEDURES SUMMARY

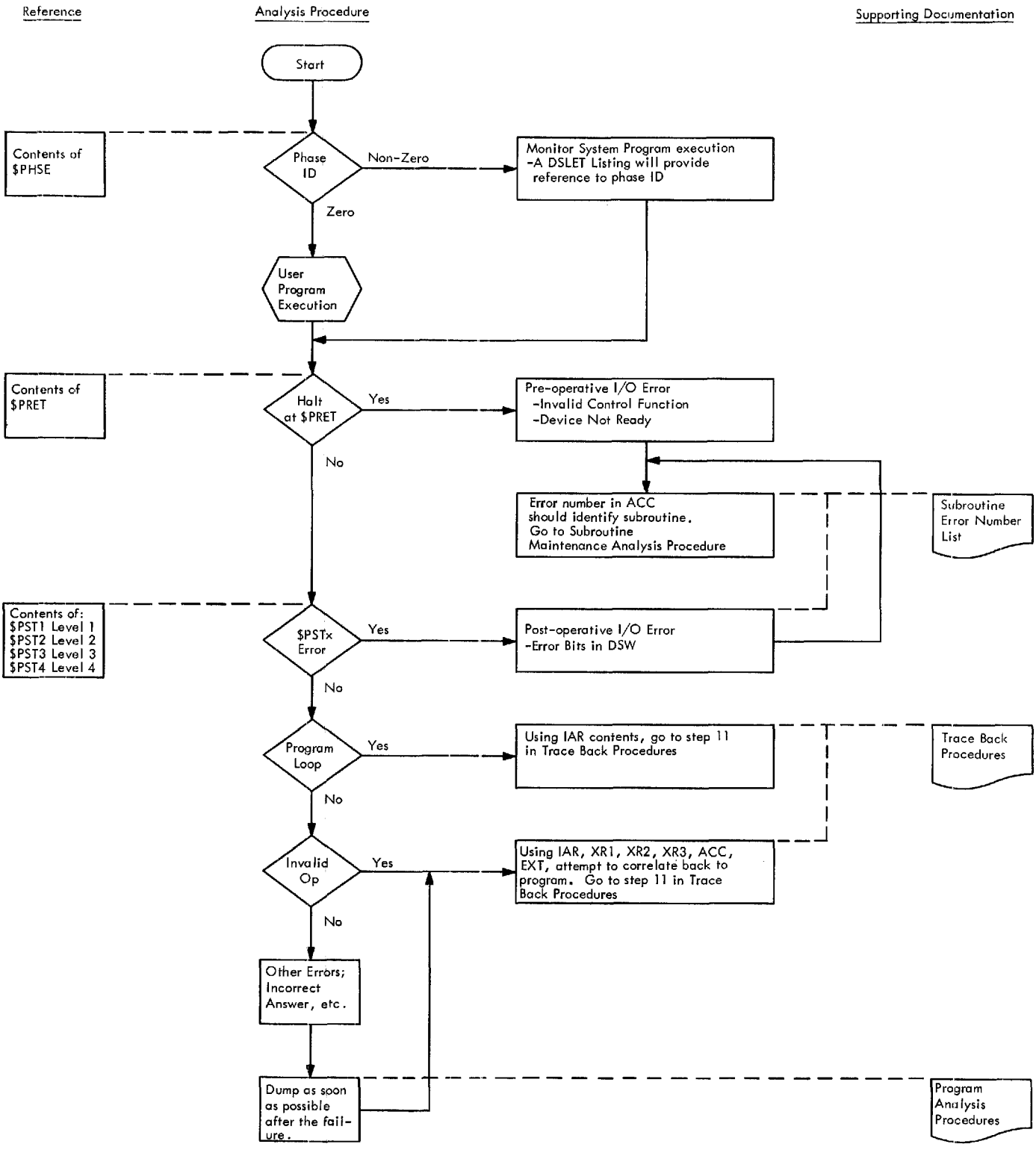
Flowdiagram 1 shows the procedure used for program analysis. At each step in the procedure, the parts of this document that apply to that step are indicated.

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

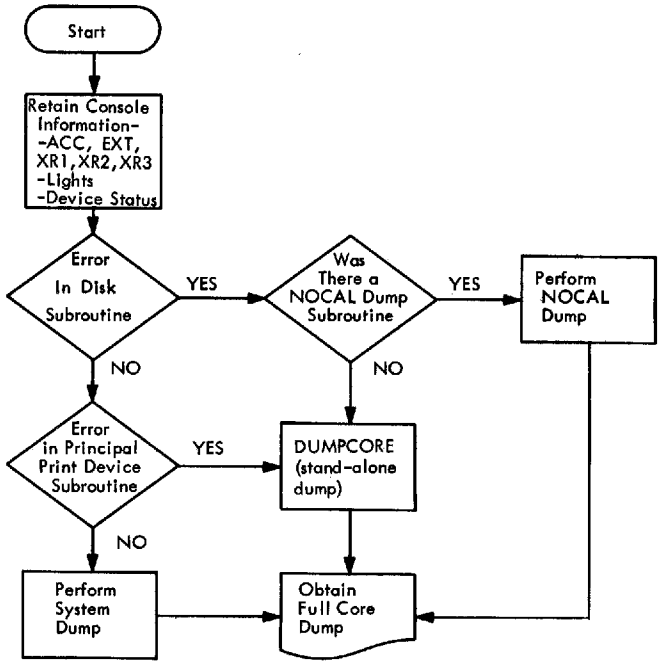
Table 8 lists the errors detected by the System Library ISSs and system device subroutines by error code, describes the conditions under which the error is detected, and provides a list of corrective actions for those errors.

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 stand-alone 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.



Flowdiagram 3. Core Dump Procedure

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, 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, SOCALs were employed. COMMON variables were defined.

Figure 17, panel 4 shows the layout of the contents of core storage during the execution of a user's Assembler Language core load.

Figure 17, panel 5 shows the layout of the contents of core storage during the execution of an RPG core load. For a detailed description of the RPG core load see IBM 1130 RPG Program Logic Manual, Form Y21-0010.

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	DETECTING SUBROUTINE DURING RPG 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	CARD0	1442-6,-7 - Device Not Ready - Read initiated with Last Card Indicator on	
				PNCH0 PNCH1	PNCH0	1442-5 - Device Not Ready	
	$PST4	System 1442/1442 Subroutine	CARDZ PNCHZ	CARD0 CARD1 PNCH0 PNCH1	CARD0 PNCH0	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	CARD0 PNCH0	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	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	WRTY0	Console Printer/Keyboard - Device Not Ready	Console Printer/Keyboard - Ready the Device
2001	$PRET	-	-	TYPE0 WRTY0	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	PAPTI PAPTX PAPTN	-	1134/1055 - Device Not Ready	1134/1055 - Ready the Device
	$PST4	System 1134/1055 Subroutine	PAPTZ	PAPTI PAPTX PAPTN	-	1134/1055 - Device Not Ready	1134/1055 - Ready the Device
3001	$PRET	-	-	PAPTI PAPTX PAPTN	-	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	READ0	2501 - Device Not Ready	2501 - Ready the Device
	$PST4	-	READZ	READ0 READ1	READ0	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	READ0	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	DISKZ	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	DISKZ	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	DISKZ	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	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	PRNT1	1132 - Device Not Ready - End of Forms	1132 - Ready the Device
6001	$PRET	-	-	PRNT1 PRNT2 PRNT3	PRNT1	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 (Continued)

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	PRNT3	1403 - Device Not Ready - End of Forms	1403 - Ready the Device
	$PST4	System 1403 Subroutine	PRNZ	PRNT3	PRNT3	1403 - Device Not Ready - Print Error	1403 - Ready the Device
9001	$PRET	-	-	PRNT3	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

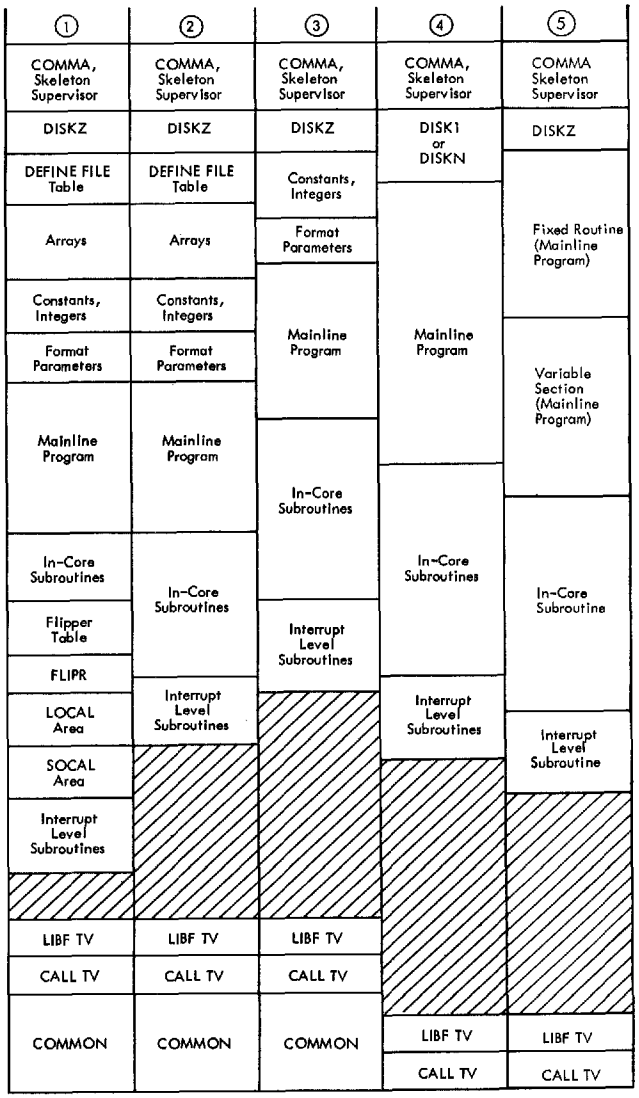
Absolute Address	Symbolic Address	Program	Explanation
/0012	-	Cold Start Loader	-Invalid disk drive number in Console Entry Switches -Indicated disk drive not ready
/0044	-	Cold Start Loader	-Disk read error -Waiting for interrupt from seek operation
/0046	-	Cold Start Loader	-Waiting for interrupt from 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).



● 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 SOCALS or LOCALS are In-Core subroutines.
-- The load address of these subroutines is listed with subroutine name.

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 SOCAL Area contains the word count of SOCAL Area.
-- Ending core address is the beginning address + the word count of the SOCAL 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:

```
                LDD         LIST
                BSI     L   ENTRY POINT
                .
                .
                .
LIST            DC          PARAMETER
                DC          PARAMETER
```

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.

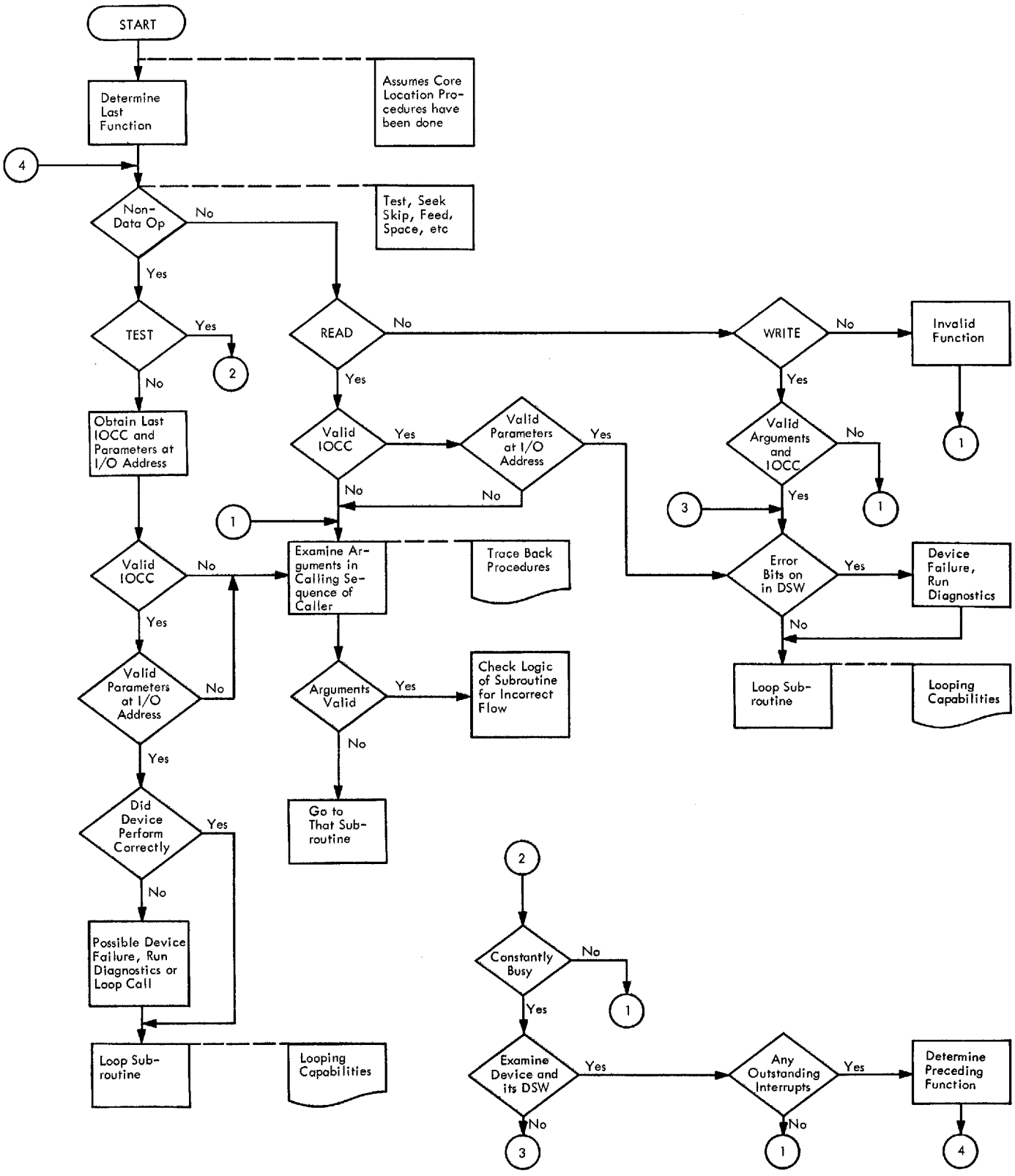
LIBRARY SUBROUTINES (except 'Z' subroutines and PLOTX)

The linkage to the System Library device subroutines (ISSs) are of the following form:

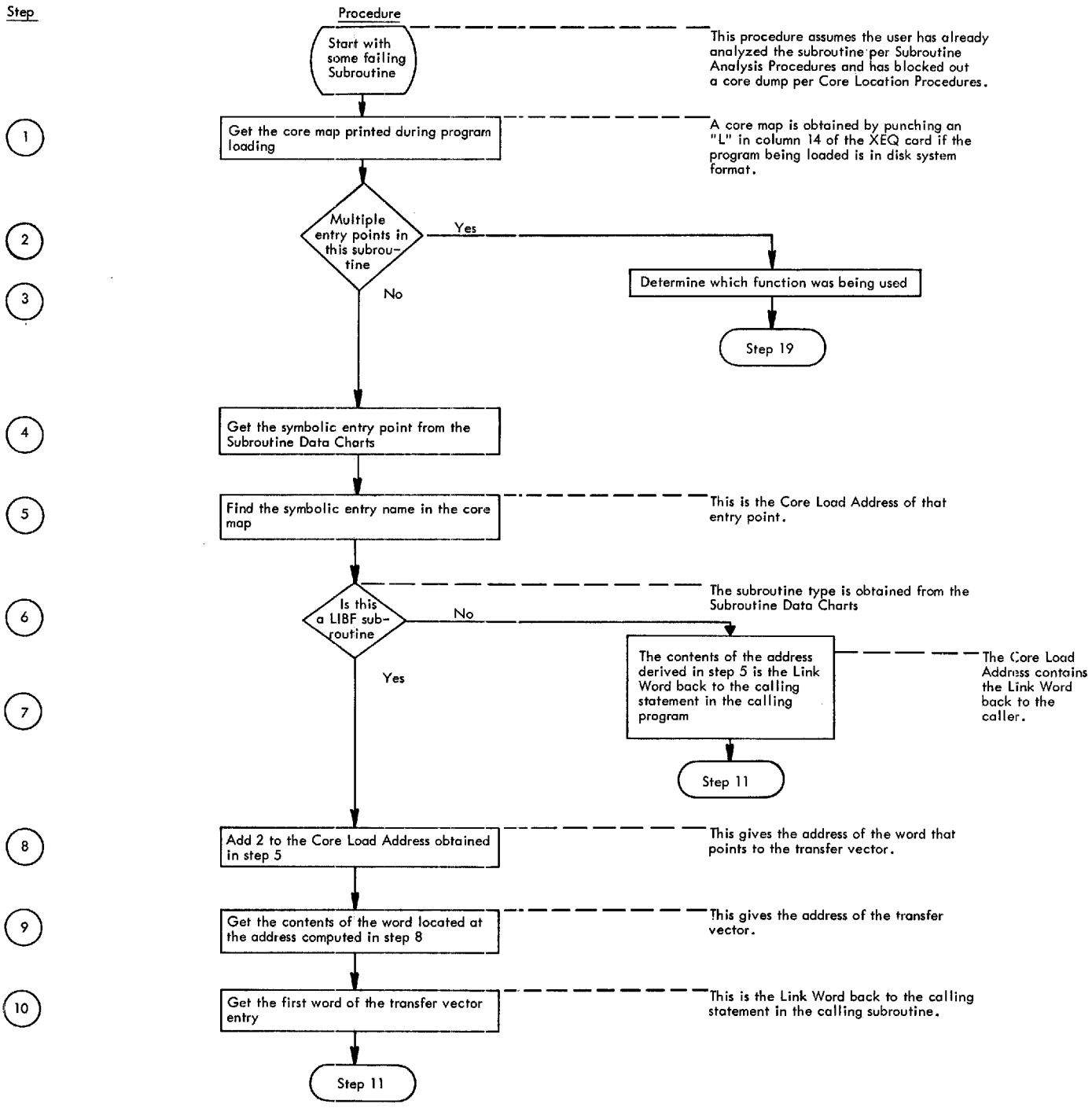
```
LIBF                            CALL
    BSI   3   TVDISP        BSI   I   CALLTV
    DC        CONTROL       DC        CONTROL
    DC        ARG 1         DC        ARG 1
      .                       .
      .                       .
      .                       .
    DC        ARGN          DC        ARGN
```

To place the subroutine into a loop:

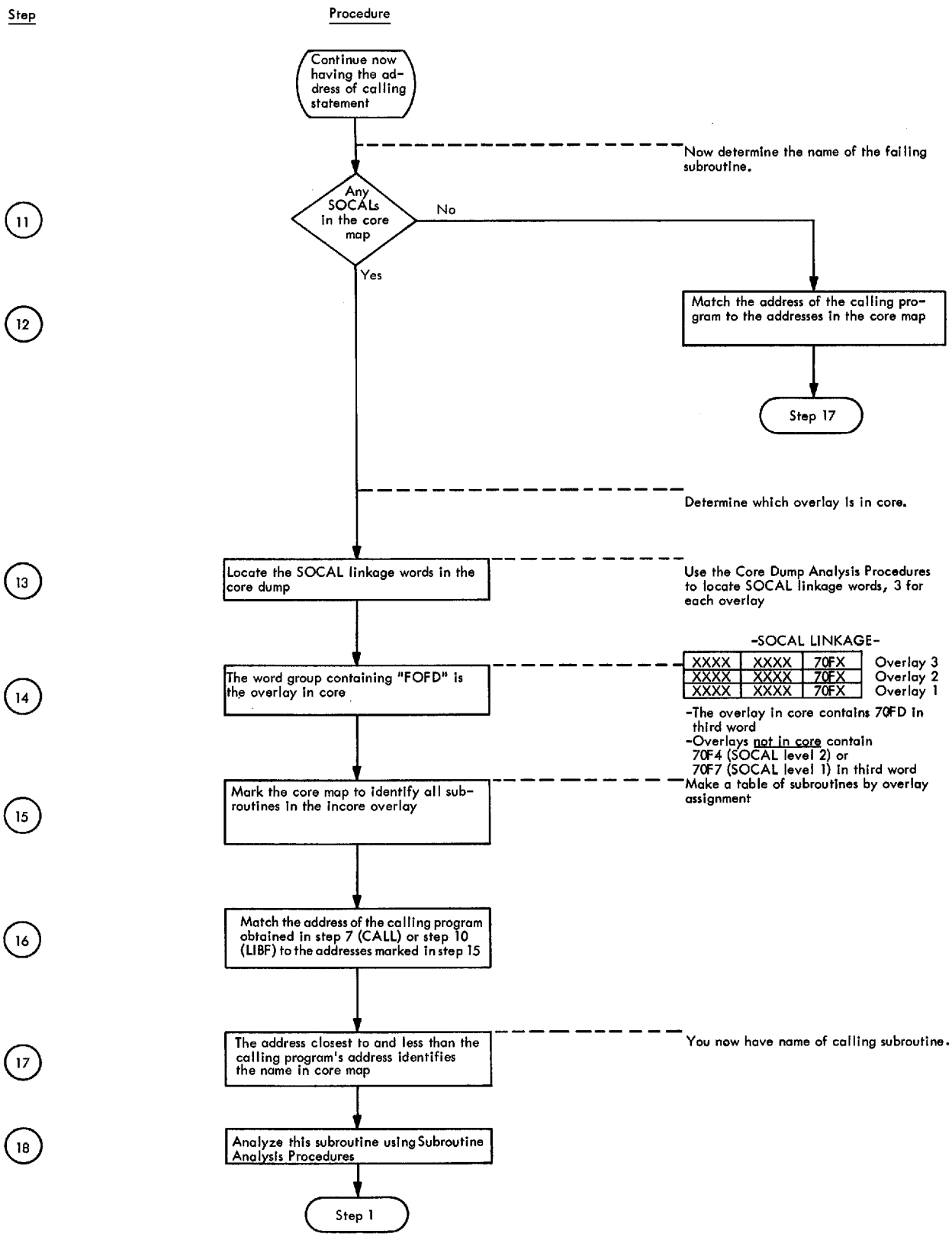
1. Insert in the location following the last argument an MDX instruction back to the BSI instruction.
2. Some of the arguments may have to be changed to point to the BSI instruction because they are error exits or busy addresses.
3. Refer to subroutine data charts for unique operating characteristics.



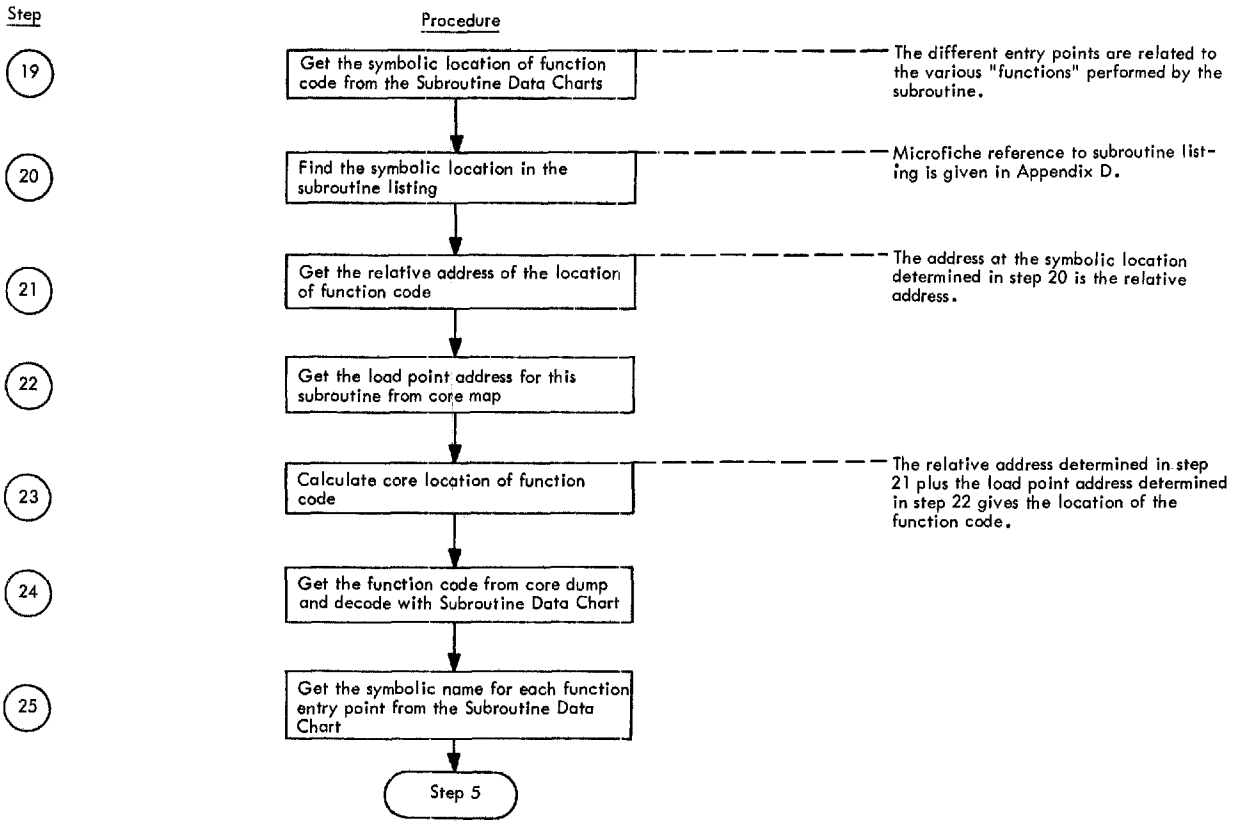
Flowdiagram 4. Generalized Subroutine Maintenance/Analysis Procedure



Flowdiagram 5. Trace Back Procedures



Flowdiagram 5. Trace Back Procedures (Continued)



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	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 word count.
KB170	Original I/O area address.
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	CD000+1	CD000+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
CD210	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	Address of page heading buffer (@HONG)	
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	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
CP200	Carriage return counter, used for counting carriage returns for restore.
CP350	IOCC for printing on console.
CP350+1	CP350 contains address of CP450.
CP370	Actual word count of message not including trailing blanks.
CP380	Data area pointer, pointing to the word containing the 2 EBCDIC characters, one of which is being printed.
CP450	Print character buffer word. The IOCC points to this word, which contains the character or control character just printed.

SYSTEM DEVICE SUBROUTINE FOR 1132

Phase ID: @1132
Used by: Monitor system programs
Subroutines required: ILS01
Linkage:

```
        LDD     L       LIST
        BSI     L       PN000+1
        .
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 address	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	/0001 indicates I/O buffer is busy and 49 print scan cycles have not been completed. /0000 indicates routine may still be busy completing the 16 idle scans; however, I/O buffer is ready to accept new input.
$CH12	Zero = Channel 12 has not been sensed. Non-Zero = Channel 12 has been sensed and skip to channel 1 has not been performed.

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 ≤ word count ≤ 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	Symbolic entry point	Return address at	Interrupt entry point
Print	/7001	Non-Zero	PR000+1	PR000	PR010
Skip	/7000	Not used	PR000+1	PR000+1	PR010
Space	/7002	Not Used	PR000+1	PR000+1	PR010

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	A NOP after the following areas have been adjusted to the correct address as a result of relocation: PR300 PR500+1 PR180+2 PR280+2 PR170+1 PR220+2 PR080 PR400+1 PR430+1
PR080	First word of Print IOCC, contains address of print buffer.
PR080+1	Second half of Print IOCC.
PR110	Second half of Sense-without-reset IOCC.
PR300	First word of Skip-to-channel-12 IOCC.
PR300+1	Second word of Skip-to-channel-12 IOCC.
PR090	First word of Space immediate IOCC.
PR090+1	Second word of Space immediate IOCC.
PR290+1	Address of the user's I/O area.
PR060	Storage location for the last DSW sensed during interrupt; also, first word of Sense-DSW-with-reset IOCC.
PR070	Second word of Sense-DSW-with-reset IOCC.
PR110	Second word of Sense-DSW-without-reset IOCC.
PR390	60-word buffer from which the line is printed.
$PGCT	Binary page count, where 0 ≤ page count ≤ 32767
$CH12	Channel 12 switch indicating channel 12 detected in DSW during interrupt; not reset until a skip to channel 1 is requested by the user.
$PBSY	Printer busy switch, modified during execution of routine. ($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.

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 without conversion	/7000	Address of I/O Area	Word Count
Punch	/7001	Address of I/O Area	Word Count
Read with 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	4
Punch	/7001	PT000+1	PT000+1	PT010+1	4
Read without conversion	/7002	PT000+1	PT000+1	PT010+1	4

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
RPG 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	DZ010	DZ010	2
Write	/0100	DZ000	DZ100+5	DZ010	DZ010	2
Find	/0000	DZ000	DZ100+5	DZ010	DZ010	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.
DZ235+1	Address of the first word of the I/O Area. C[C(DZ235+1)] = originally requested word count C[C(DZ235+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

Peroperative 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	Start read or punch IOCC, set by program depending on function used to initiate operation. If read, CZ904 + 1 = CZ906 + 1 If write, CZ904 + 1 = CZ908 + 1
CZ902	Read or Punch IOCC, set by program depending on function to read or punch columns. If read, CZ902 + 1 = CZ906 If write, CZ902 + 1 = CZ908
CZ923	Address pointer to I/O area, incremented on each column interrupt.
CZ925	Original I/O area address -1.
CZ920	DSW is saved here on an operation-complete interrupt.
CZ010	Switch used for waiting for interrupt: Set positive when waiting for any interrupt. Set zero when column interrupt occurs. Set negative when op-complete interrupt occurs.
$RWCZ	Previous operation switch: /0000 = previous operation was a read. /0002 = previous operation was a write. 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.
CZ918	Switch used to test for // card before writing on it; zero means only reading or previous operation before write was a read.
CZ918-3 thru CZ918	Buffer area for saving first 3 columns; rest of card is read into fourth word when reading before write.

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	Error display indicator.
PZ400+1	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	via LIBF TV link word	KZ100	KZ100	4
Write	/0002	TYPEZ	via LIBF TV link word	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	WZ100	WZ100	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
WZ904	First word of Print IOCC, Address of output area.
WZ906	Address for store after character conversion.
WZ908	Counter for character conversion.
WZ990	EBCDIC-to-1403-Printer-code conversion table.
WZ934	Transfer complete switch: zero = transfer complete.
WZ933	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	X	X	X	X		

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 and RPG generated 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 and RPG generated 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 and RPG generated 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+1	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.
INIT	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 and RPG generated 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	/20X0	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+4
	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+2	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 and RPG generated 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	Address of error subroutine, error parameter required
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	W3080+1			
Print	Third digit at W3920	PRNT3	W3080+1	W3010	W3010	4
Control	Third digit at W3920	PRNT3	W3080+1	W3010	W3010	4

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	W3905	W3905+1	W3060+1	W3050+1		W3070
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
W3920	First word of the Sense-DSW-without-reset IOCC; also, Carriage control character.
W3935	Routine busy switch, non-zero indicates the routine is processing a message.
W3990	First word of 60-word output buffer.
W3975	DSW saved for checking of indicator bits at appropriate time; error bits during print complete ch. 9/12 during carriage operation complete.
W3130+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		XR1+1			XR1+2
	Used	X		X			X

Significant variables:

Symbolic location	Contents/Use
DEVIC	Function indicator: /0002 = Read. /0001 = Punch.
CHECK	Switch for controlling checking function: /0000 = Do not check. /FF00 = 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.
FCRD	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	PAPTN	RET+1	INTN	INTN	4
Punch	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	/FF00 = Check for a Delete or Stop character. /0000 = Do not check.
WDCNT	Number of words remaining to be punched.
RWDCT	Number of words remaining to be read.
IOAR	Address pointer to user data area (punch).
RIOAR	Address pointer to user data area (read).
USER1	Address of user error routine (punch).
RUSE1	Address of user error routine (read).
BUF	Temporary storage for word to be punched.
RBUF	Temporary storage for word to be read into.
Index Register 2	Address of RDTBL, if reading. Address of PNTBL, if punching.
IOCC	IOCC used for punching a character.
READS	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	NOTE
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 on in DSW	Return to this word if bit 8 is off in DSW	
Read	/1X00	I/O Area Address	Address of user error routine	X = 1, Stacker Select X = 0, No Stacker Select
Feed	/3000	NSI (return to this word after feed)	NSI	
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 MPR62		Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test		OMPR1	MPR59+1			
Timing Mark Test		OMPR1	MPR59+1			
Read	/FFFE	OMPR1	MPR59+1	INT1	INT1	4
Feed	/0000	OMPR1	MPR59+1	INT1	INT1	4
Disconnect		OMPR1	MPR59+1			
Stacker Select		OMPR1	MPR59+1			

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	SAVAQ	SAVAQ+1	SAVX1	MPR56+1		MPR58
	Used	X	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.
MPR84+2	Entry to user's error routine.
MPR62	Function: /FFFE = Read (2000) /0000 = Feed (3000)
READ	IOCC for Read function.
READ+1	Address of user I/O area.
MPR68	NONZERO, Feed has already been performed. ZERO, 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.
MPR64	Zero, No first character interrupt. Non Zero, Have received first character interrupt.
MPR66	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 and RPG generated 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 date 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 initing operation	Word 1 = not used Word 2 = drive code and sector address

†Applies to simulated LIBF.

Postoperative conditions and entry points:

Function at D1928		Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	/0000	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Read	/0001	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Write W/O RBC	/0002	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Write W/RBC	/0003	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Write Immediate	/0004	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Seek W/O Seek Option	/0005	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2
Seek With Seek Option	/0005 (D1929=x000)	D1000 (branch) D1020+2 (LIBF)	D1070	D1020	D1020	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	D1912+1	D1912	D1060+2	D1060+4		D1060
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
D0006	If DISK1 was entered by a monitor system program at D1000, 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.
D1000	Monitor system program entry.
D1020+2	LIBF TV entry.
D1020+4	LIBF TV link word. Contains the address of the first parameter in the calling sequence.
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.
D1901	Current arm position obtained by reading the sector address after seeking.
D1902 and D1903	Last two words of sector previously read. The area is used to store an intermediate word count and sector address.
D1904 and D1905	IOCC for the operation currently being performed.
D1906 and D1907	IOCC for the user-requested function.
D1908 and D1909	IOCC to reader sector address into D1901 after a seek operation.
D1911	Second word of Sense IOCC.
D1912 and D1913	Used to save and restore the ACC and EXT.
D1925	Word count remaining to be read or written from originally requested word count.
D1926	Next sequential sector to be read or written.
D1929	Non-zero if the seek option was requested.
D1930	Non-zero if the displacement option was requested.
D1932	Second word of Seek IOCC.
D1938	Current sector address.
$DBSY	Non-zero indicated routine is busy; this word must be cleared to zero before entry to this routine is permitted.

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 paramaters (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	/4000	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 DN984+XR1	Symbolic entry point	Return address at	Interrupt entry point	Return address at	Interrupt level
Test	/0000	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Read	/0001	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Write W/O RBC	/0002	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Write With RBC	/0003	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Write Immediate	/0004	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Seek W/O Seek Option	/0005	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2
Seek With Seek Option	/0005	DN000 (branch) DN020+2 (LIBF)	DN120	DN020	DN020	2

Register status:

		ACC	EXT	XR1	XR2	XR3	Status
Mainline	Saved at/restored from symbolic location	DN902	DN902+1	DN110+1	DN110+3		DN100
	Used	X	X	X	X		X

Significant variables:

Symbolic location	Contents/Use
DN230+1	If DISKN was entered by a monitor system program at DN000, this word is used by DISKN to simulate a LIBF link word. The contents of this word should reference the simulated LIBF at DN902.
DN020+4	Contents of the LIBF link word if called by a user-written LIBF.
DN902 and DN902+1	Simulated LIBF parameters for direct branch input: D0010 = /1100 for read input = /2200 for write input D0010+1 = address I/O area

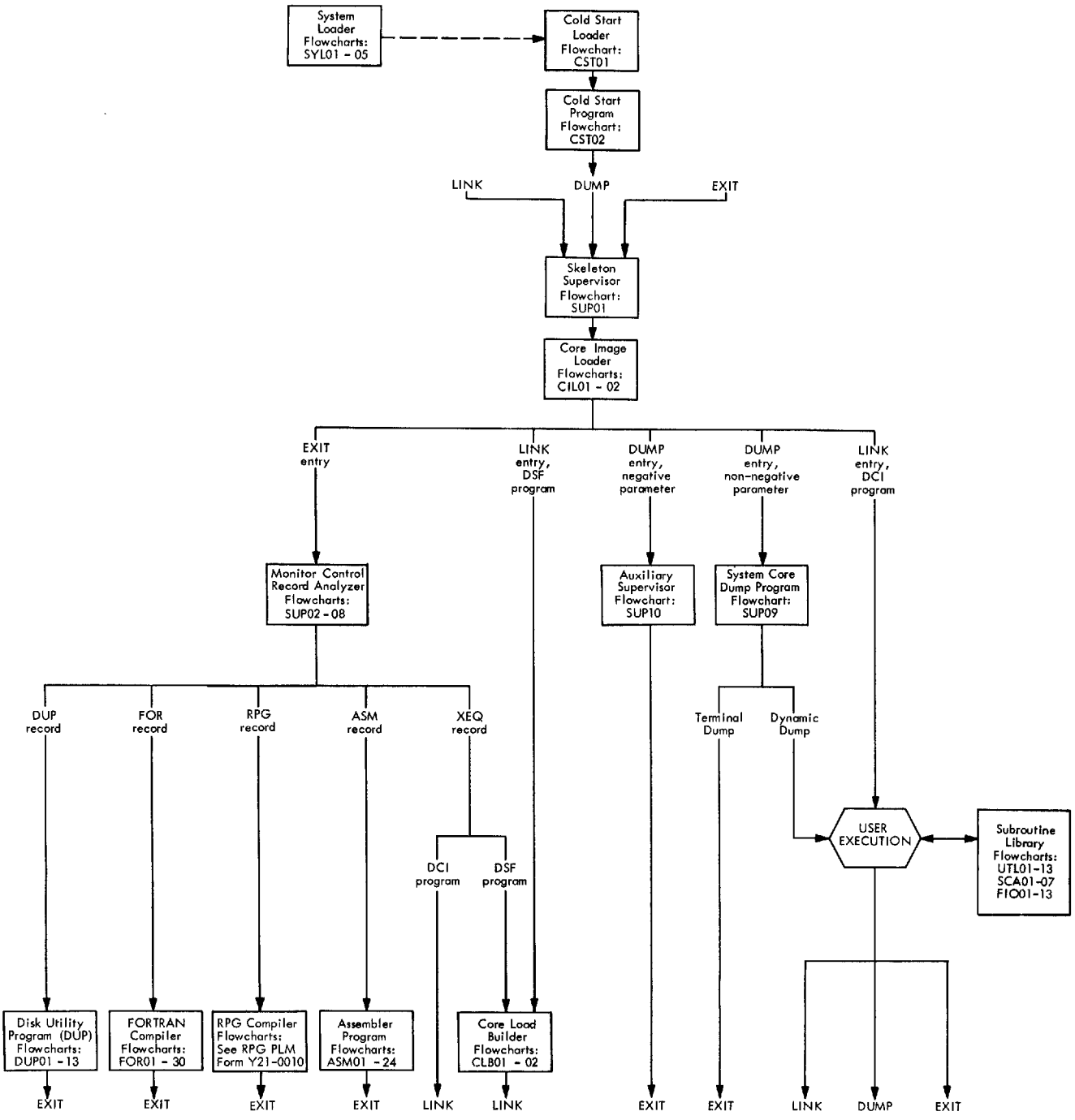
DISKN is capable of executing 5 drives simultaneously. Reference to the proper disk drive work areas is accomplished by use of a table. XR1 is used to point to the relative starting position for each specific drive in the table. The starting address in the table is computed as follows:

Significant variables (Continued):

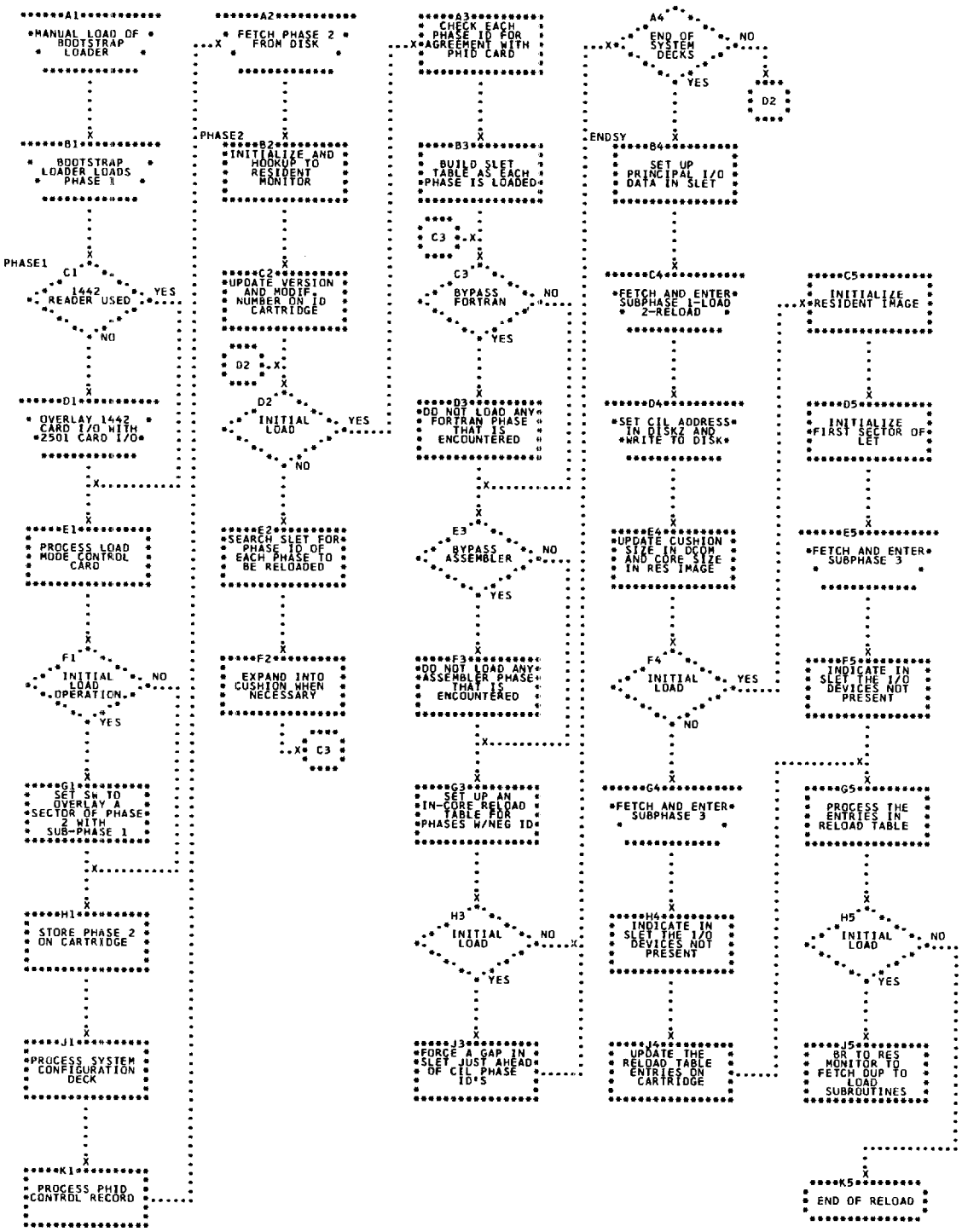
Symbolic location	Contents/Usw
	The contents of location DNXXX+XR1 where DNXXX is the first entry in the table and XR1 contains twice the drive code. Example: Assume drive 1 is referenced. The address would be: DNXXX+(1 x 2) = DNXXX + 2
DN952 and DN952+1	First and second words of Seek IOCC if one is required; also used as temporary storage.
DN050+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 twice the drive code.
Index Register 2	Drive code and I/O area address at various times during execution of this routine.
DN978+XR1	Current position (sector address) of the heads on the referenced drive.
DN980+XR1	Last two words of the I/O area just read into.
DN982+XR1	Address of the user's error routine.
DN983+XR1	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.
DN970+XR1	Originally requested user function.
DN985+XR1	Error counter for referenced drive for this operation: 50_{10} - C(DN985+XR1) = total errors occurred for this drive during requested function.
DN986+XR1 and DN986+1+XR1	IOCC for the current user-requested operation, except for Seek and Sense DSW.
DN990+XR1	Originally requested sector address.
DN991+XR1	Current sector address for current IOCC.

Significant variables: (Continued)

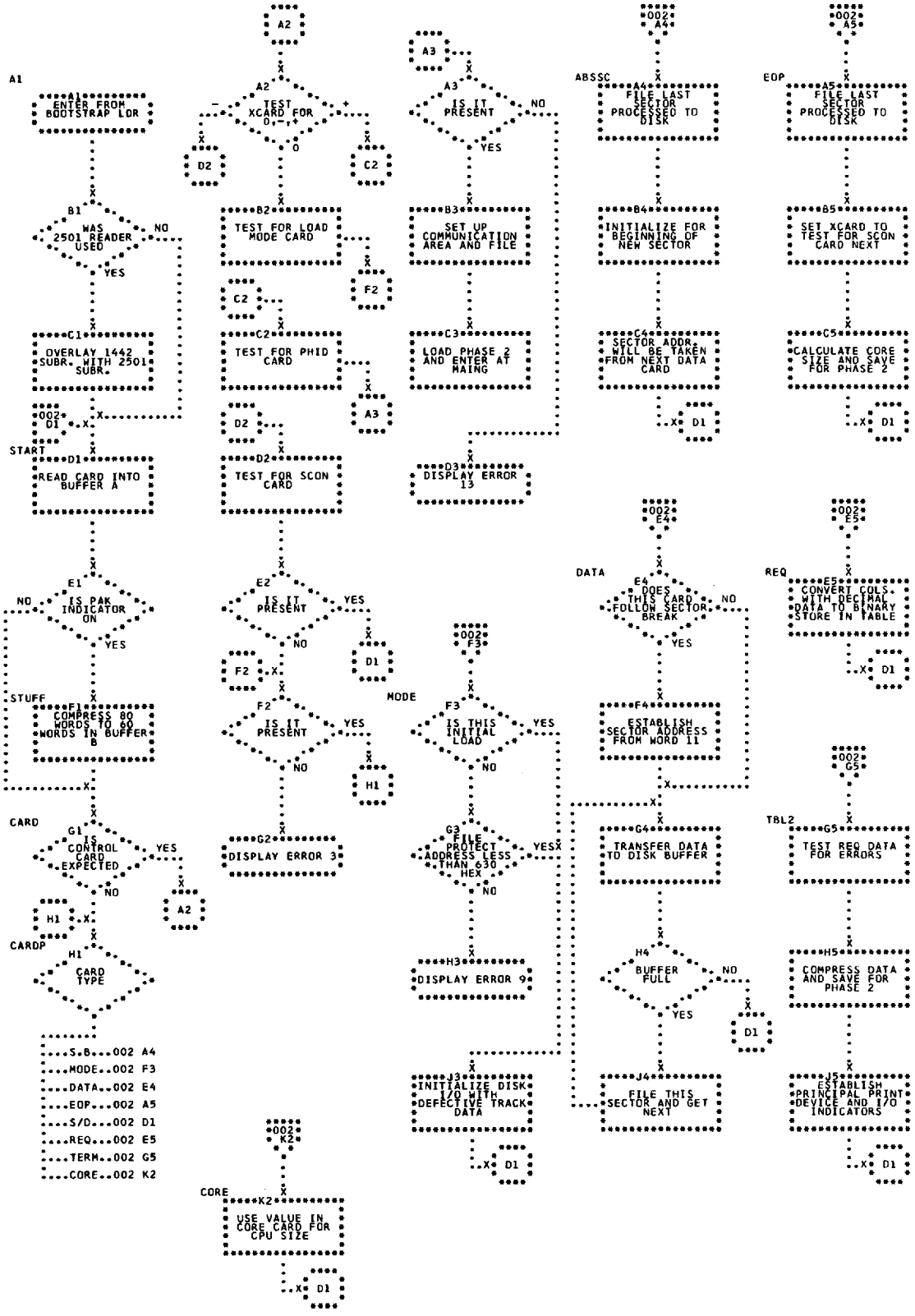
Symbolic location	Contents/Use
DN992+XR1	Remaining word count to be read or written.
DN993+XR1	Word count to be used in next I/O operation.
DN994+XR1	Current I/O area address.
DN995+XR1	Read-back-check counter: 100_{10} = C(DN995+XR1) = total errors occurred attempting to perform RBC.
$DBSY	Current status 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.



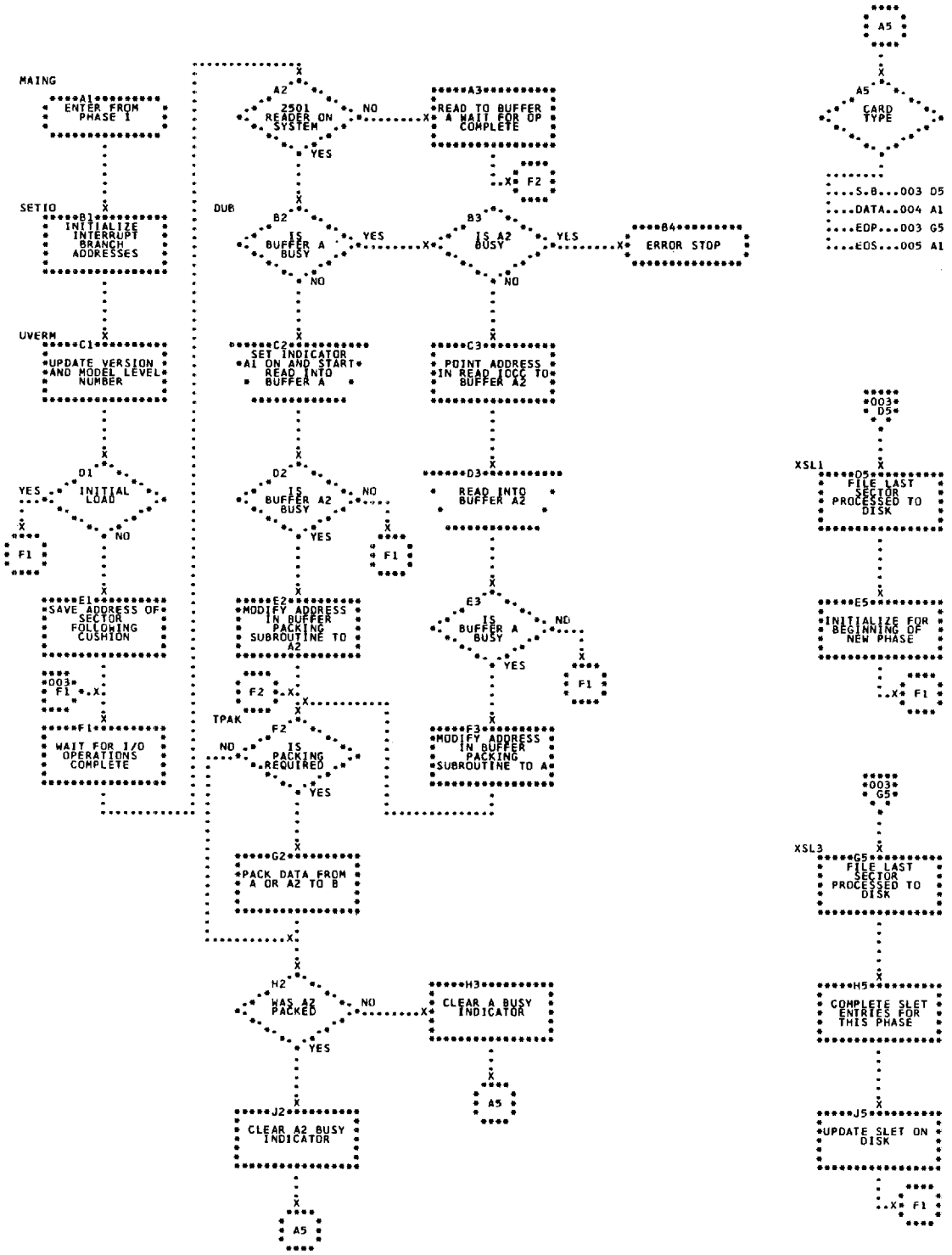
Flowchart DMS01. Disk Monitor System, System Overview



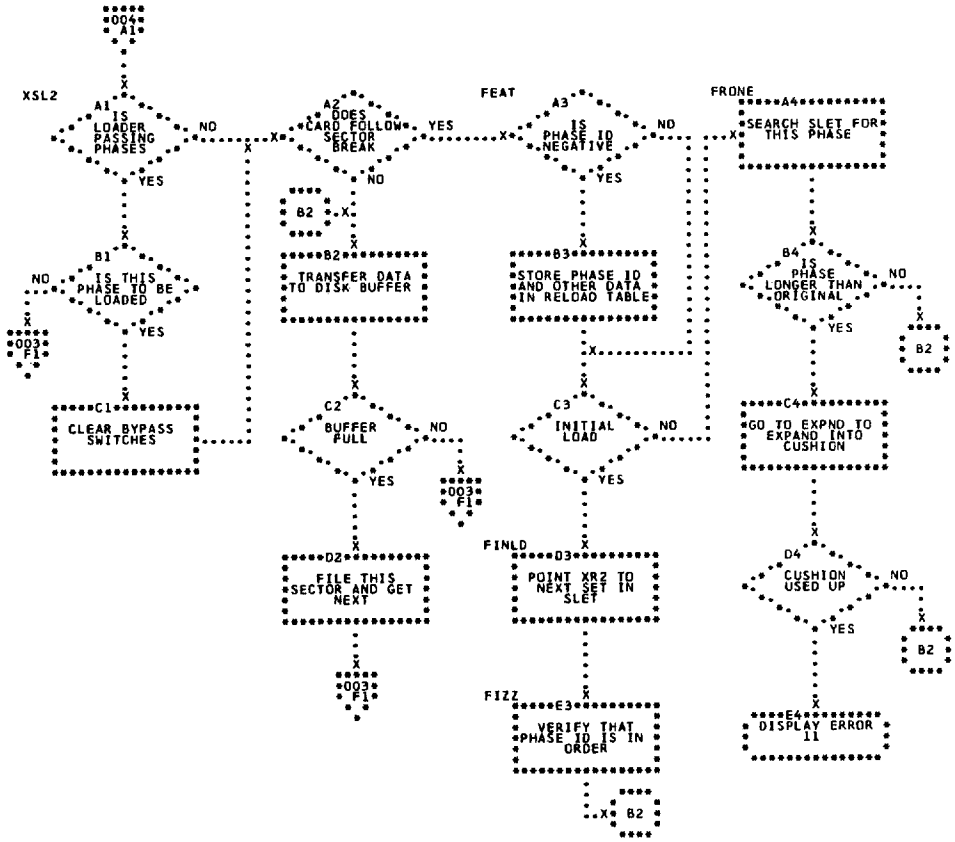
Flowchart SYL01. System Loader, General Flow



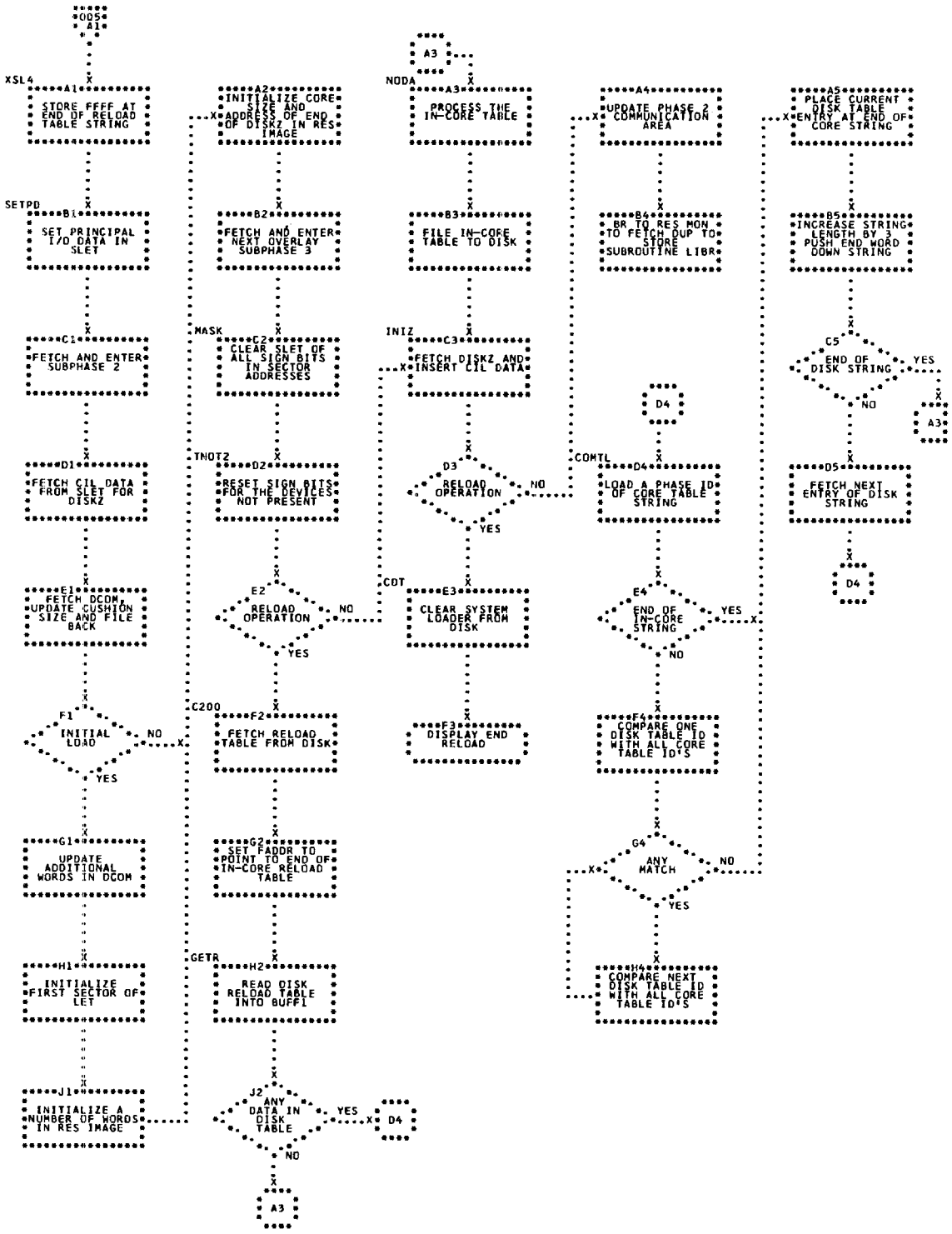
Flowchart SYL02. System Loader, Phase 1



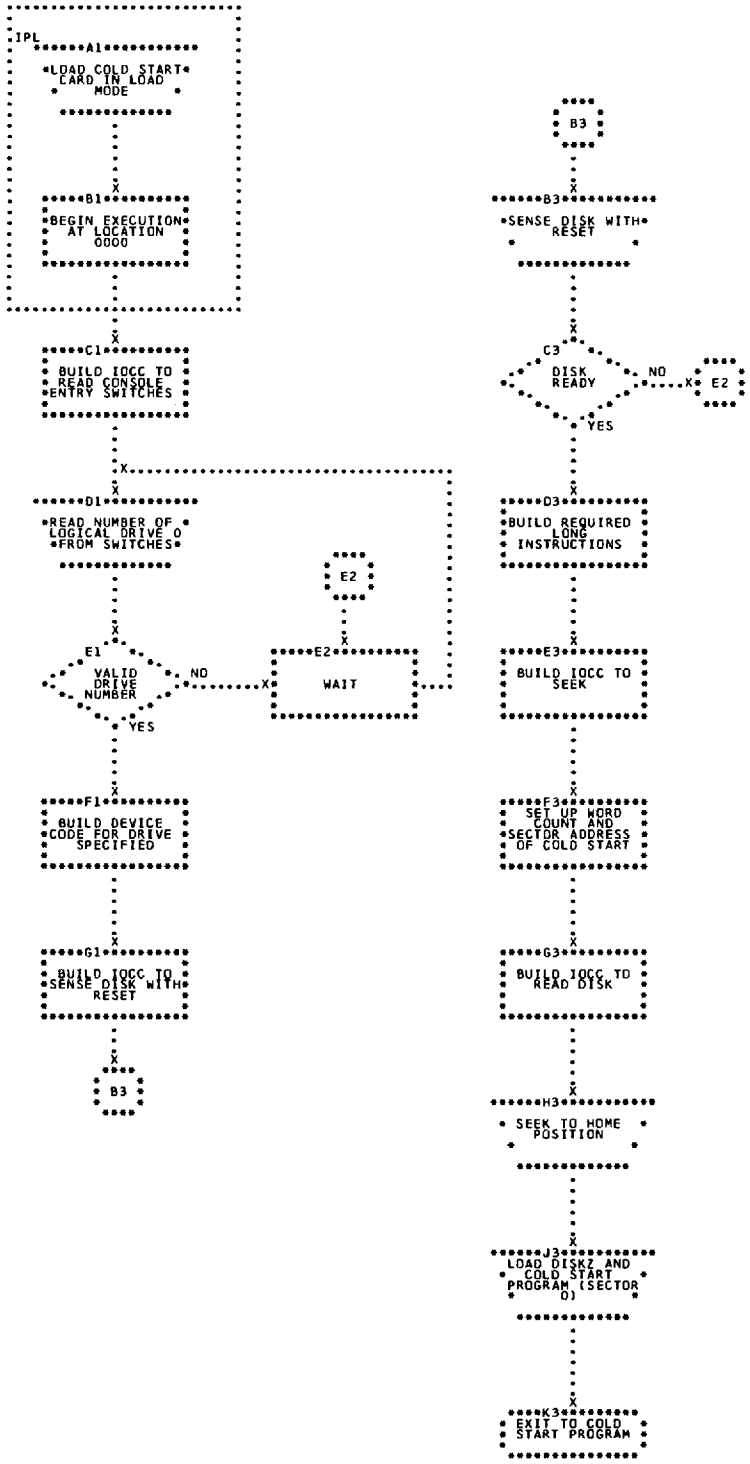
Flowchart SYL03. System Loader, Phase 2



Flowchart SYL04. System Loader, Phase 2

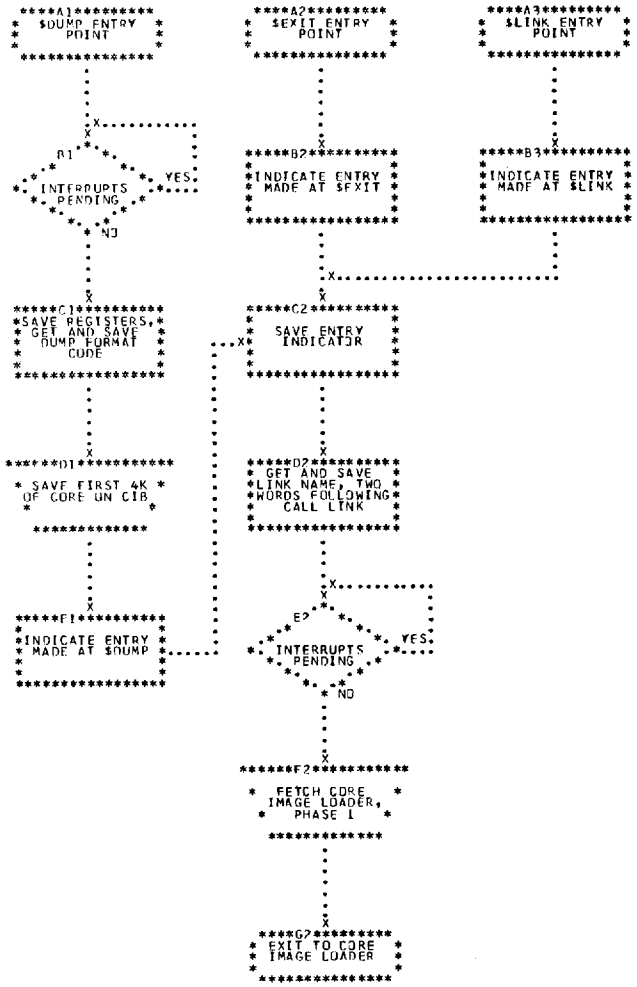


Flowchart SYL05. System Loader, Phase 2

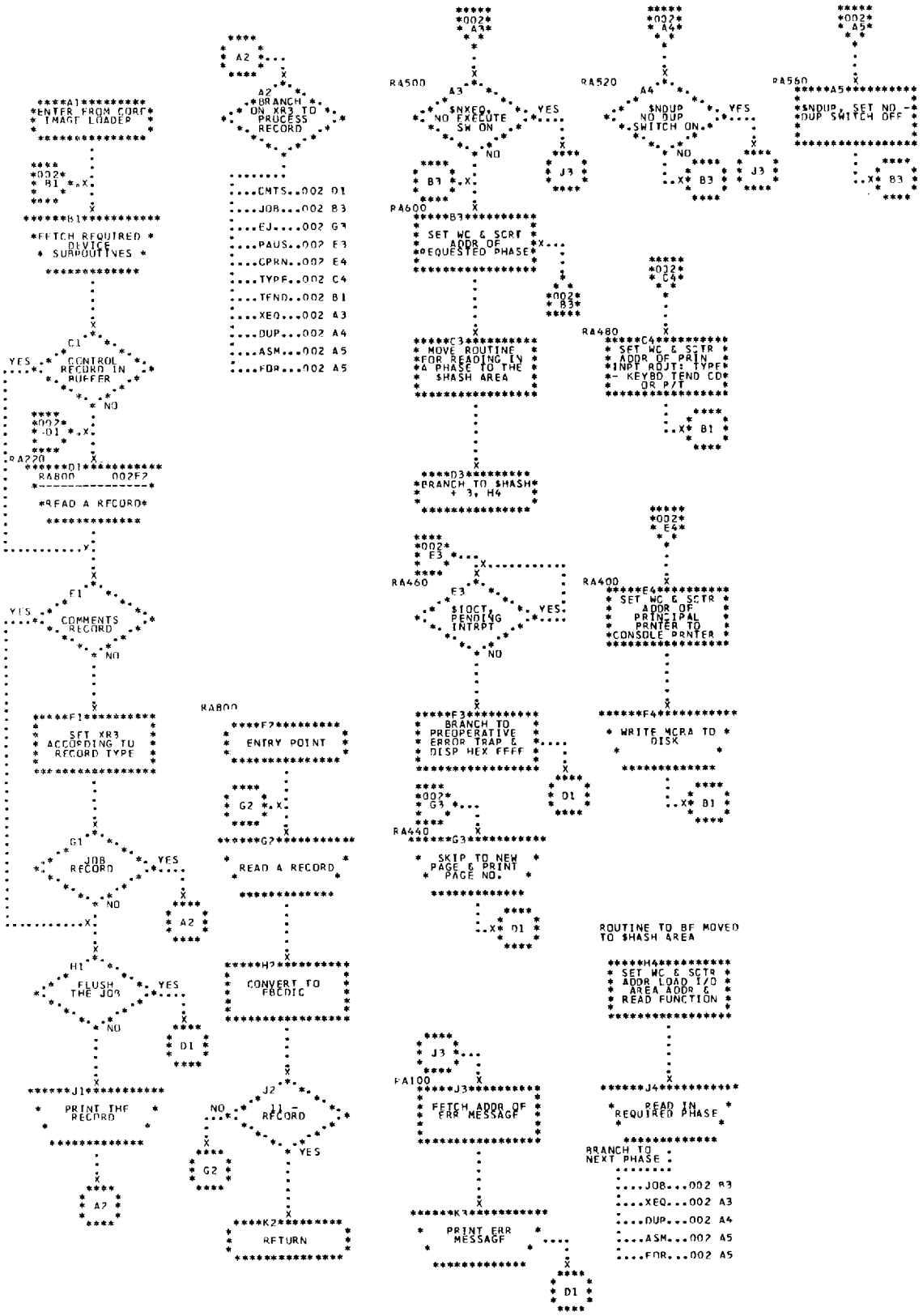


Flowchart CST01. Cold Start Loader

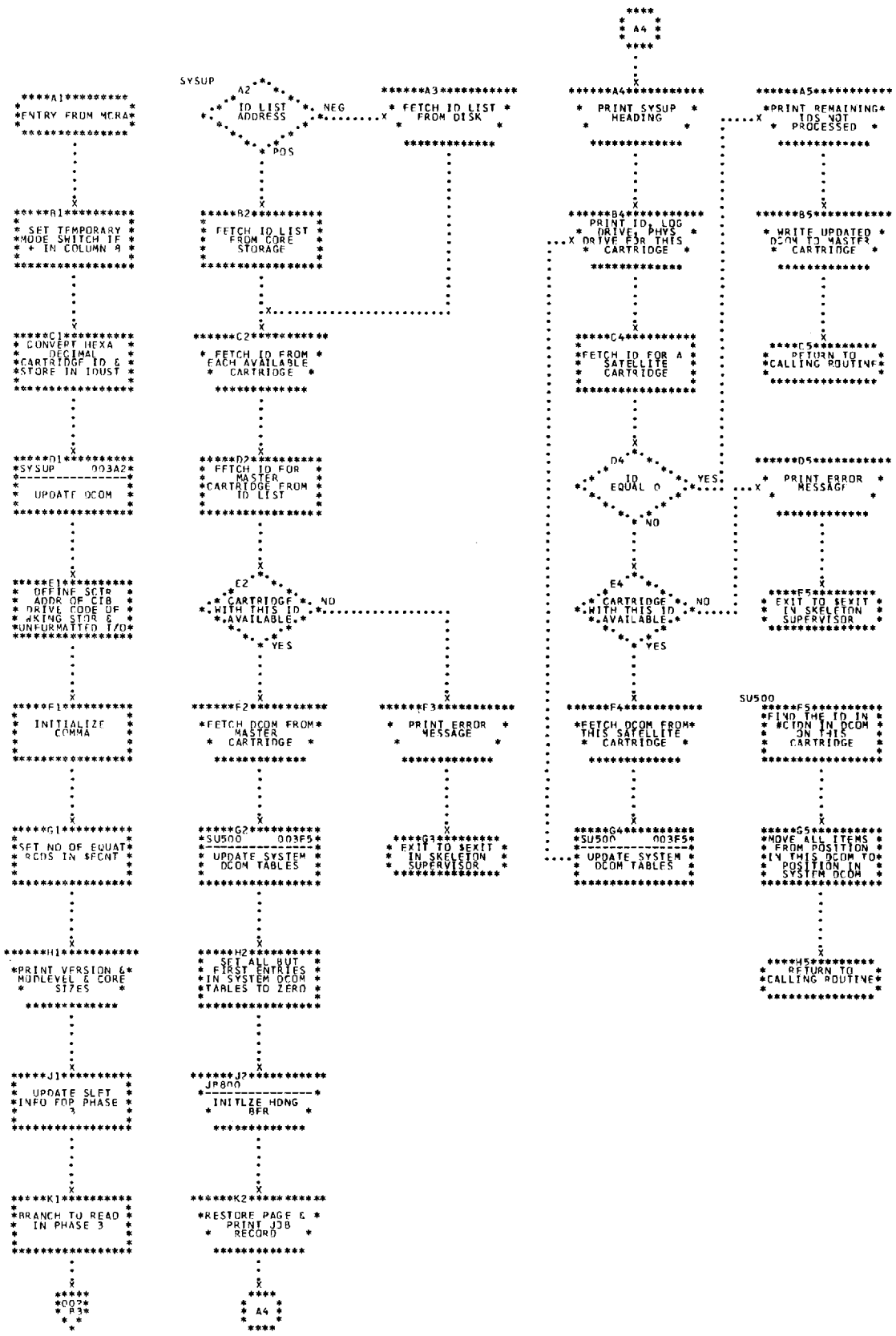
Flowchart CST02. Cold Start Program



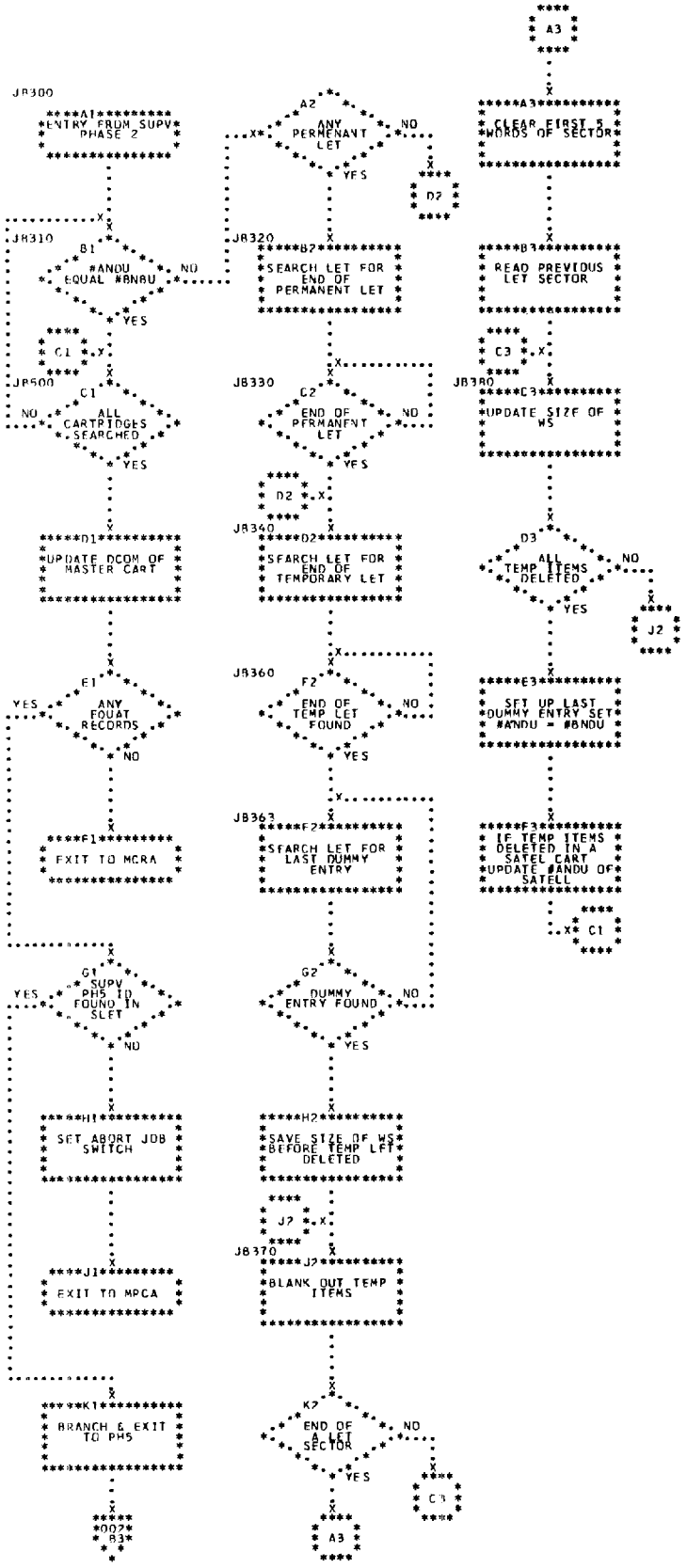
•Flowchart SUP01. Supervisor, Skeleton Supervisor



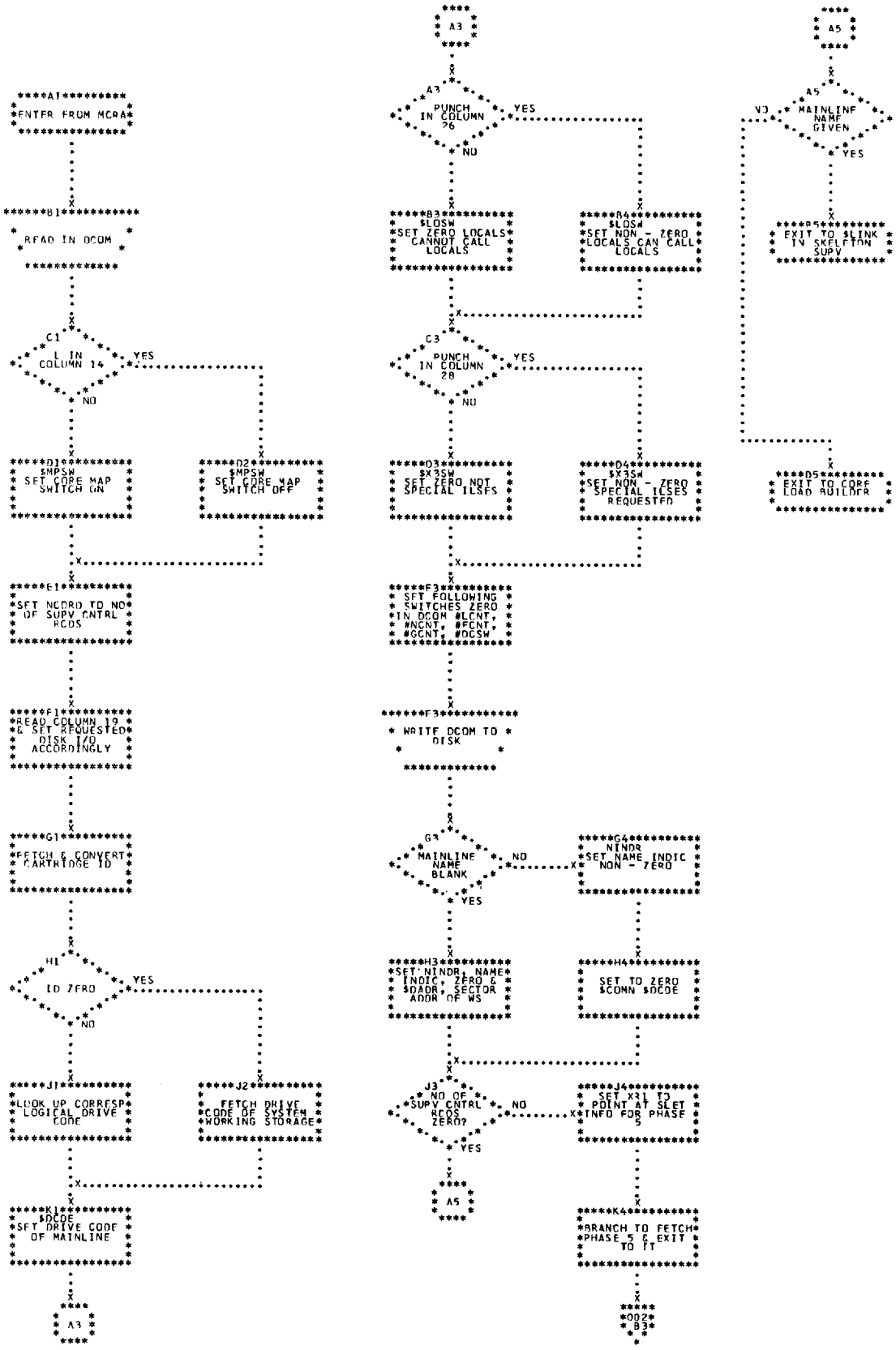
• Flowchart SUP02. Supervisor, Monitor Control Record Analyzer



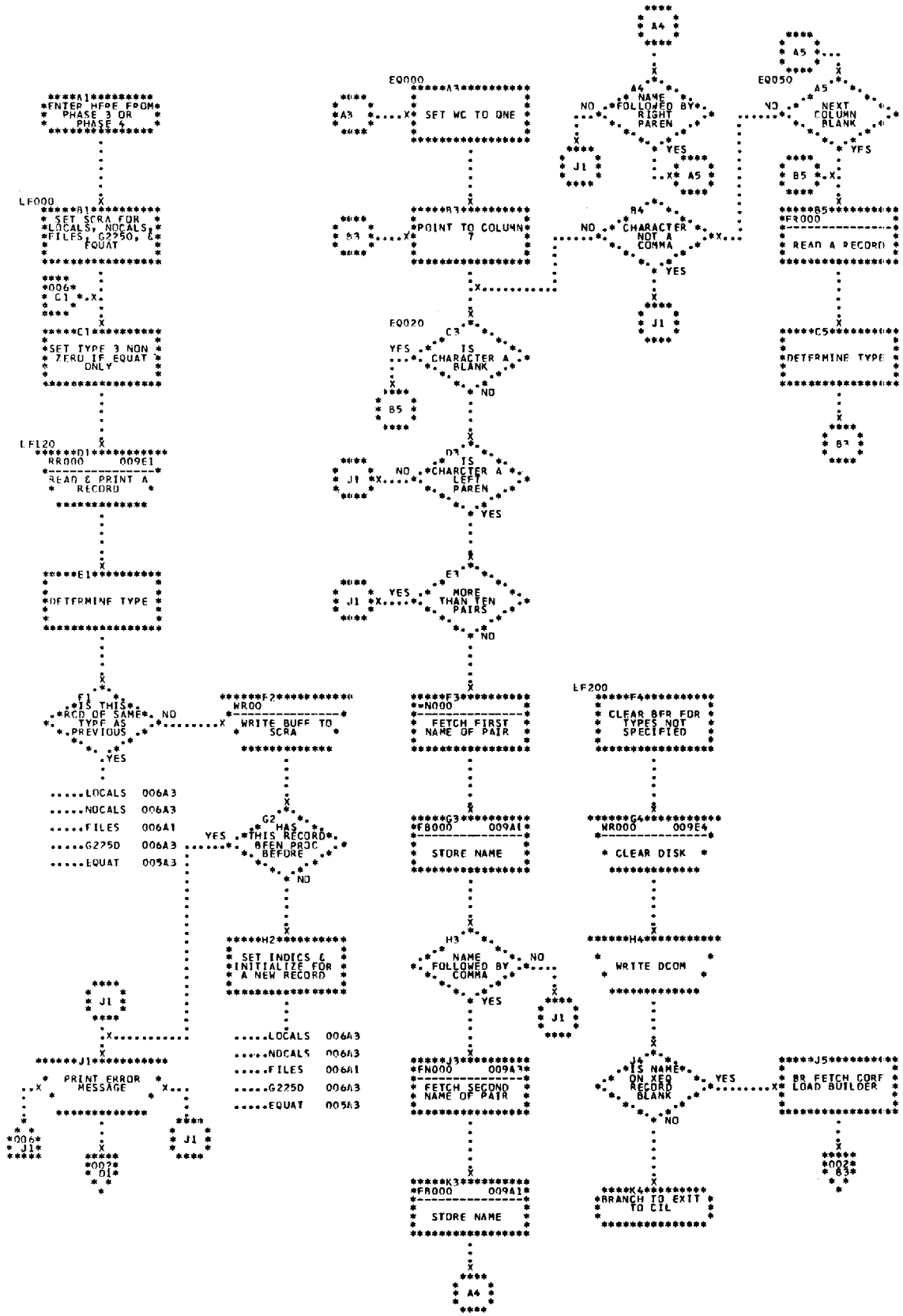
•Flowchart SUP03. Supervisor, Job Processor



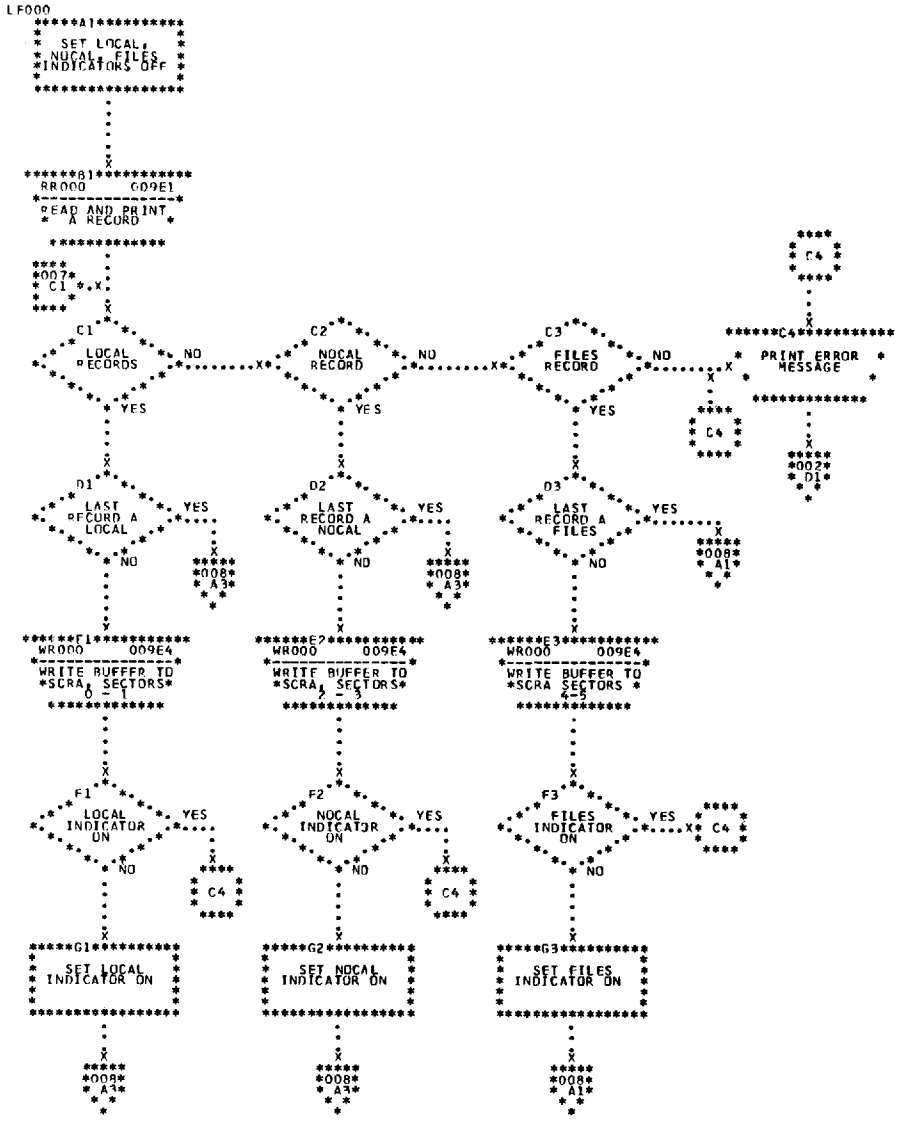
•Flowchart SUP04. Supervisor, Delete (Temporary Items) Let



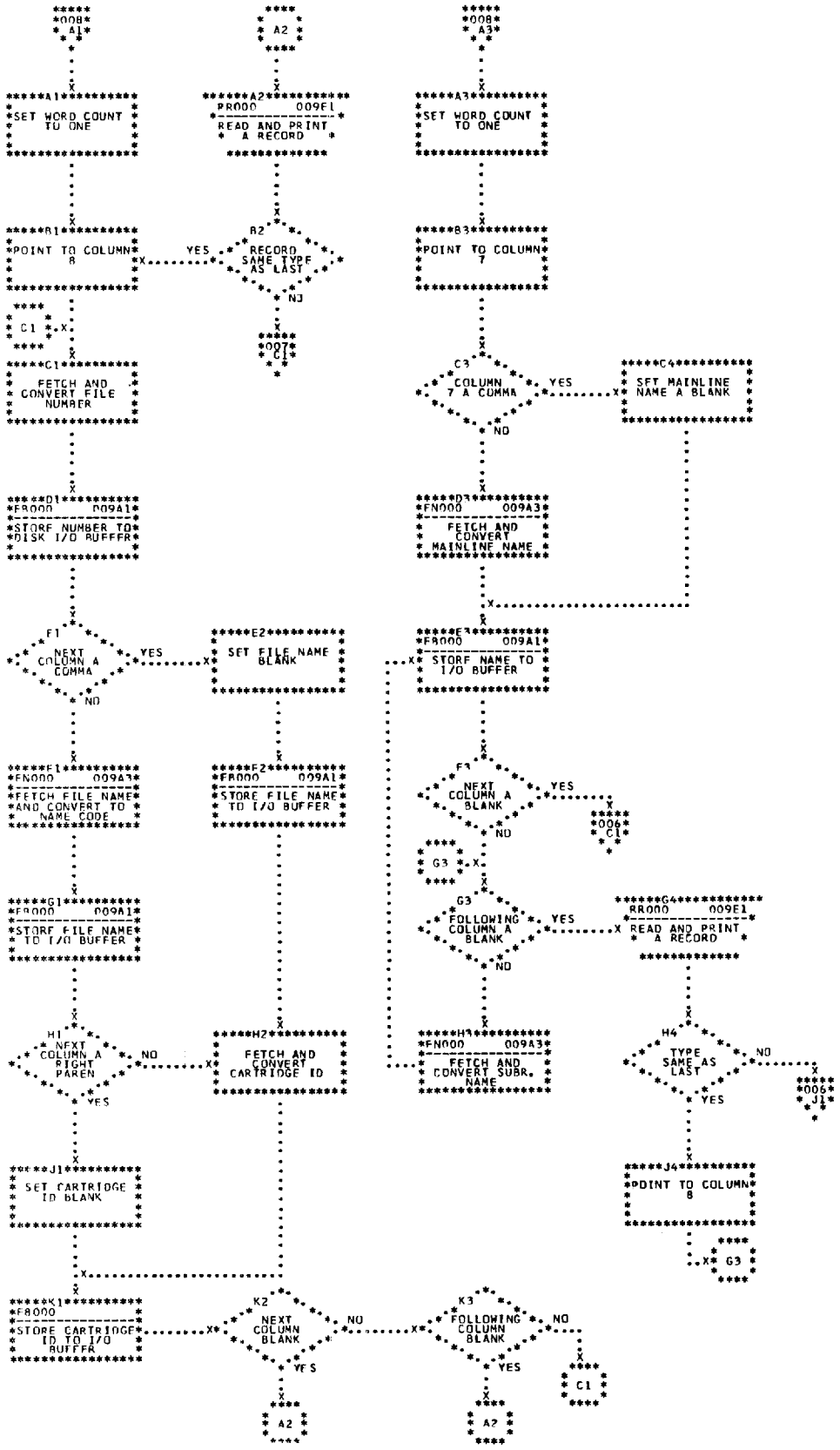
•Flowchart SUP05. Supervisor, XEQ Processing



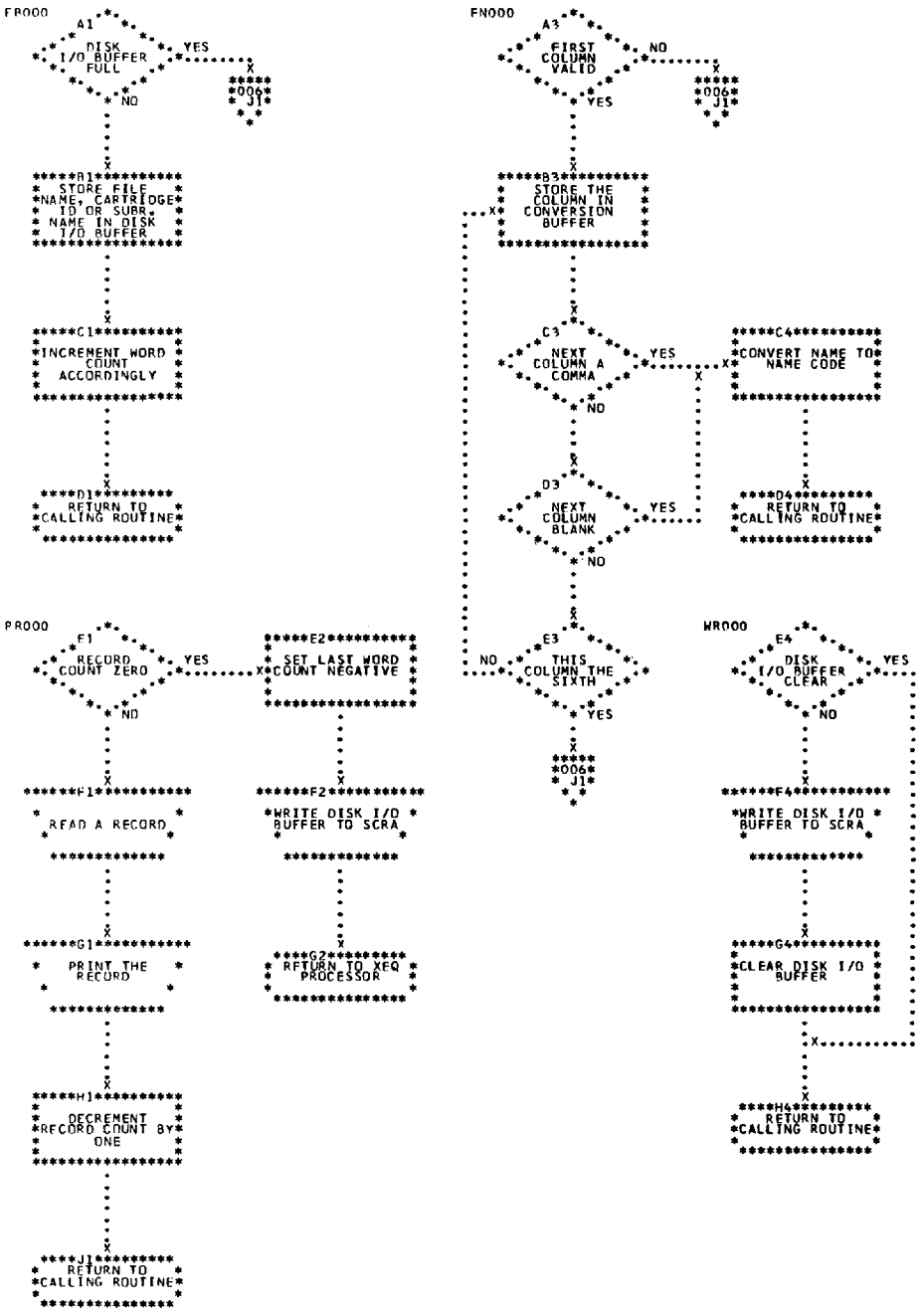
•Flowchart SUP06. Supervisor, Control Record Analyzer



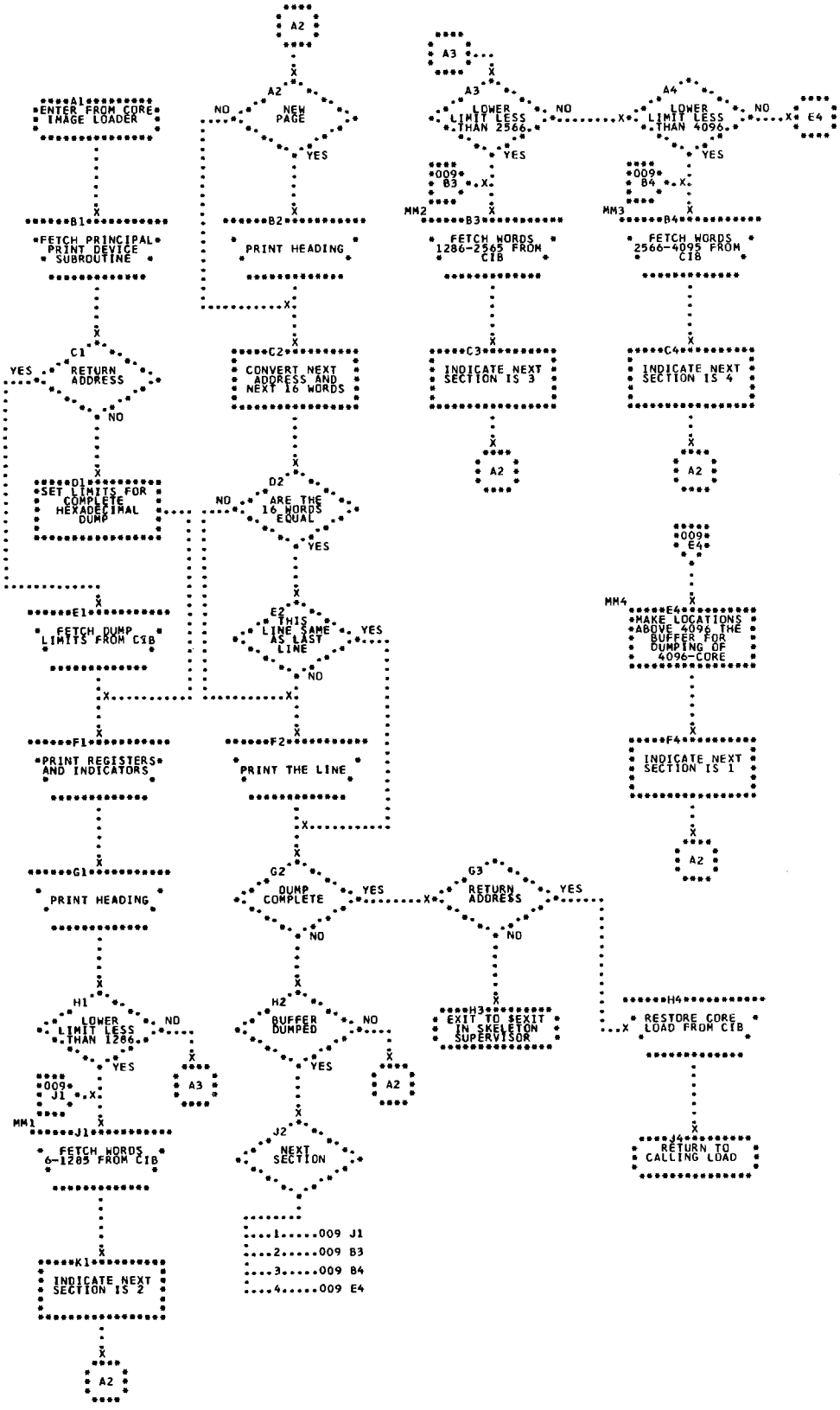
•Flowchart SUP07. Supervisor, Control Record Processor



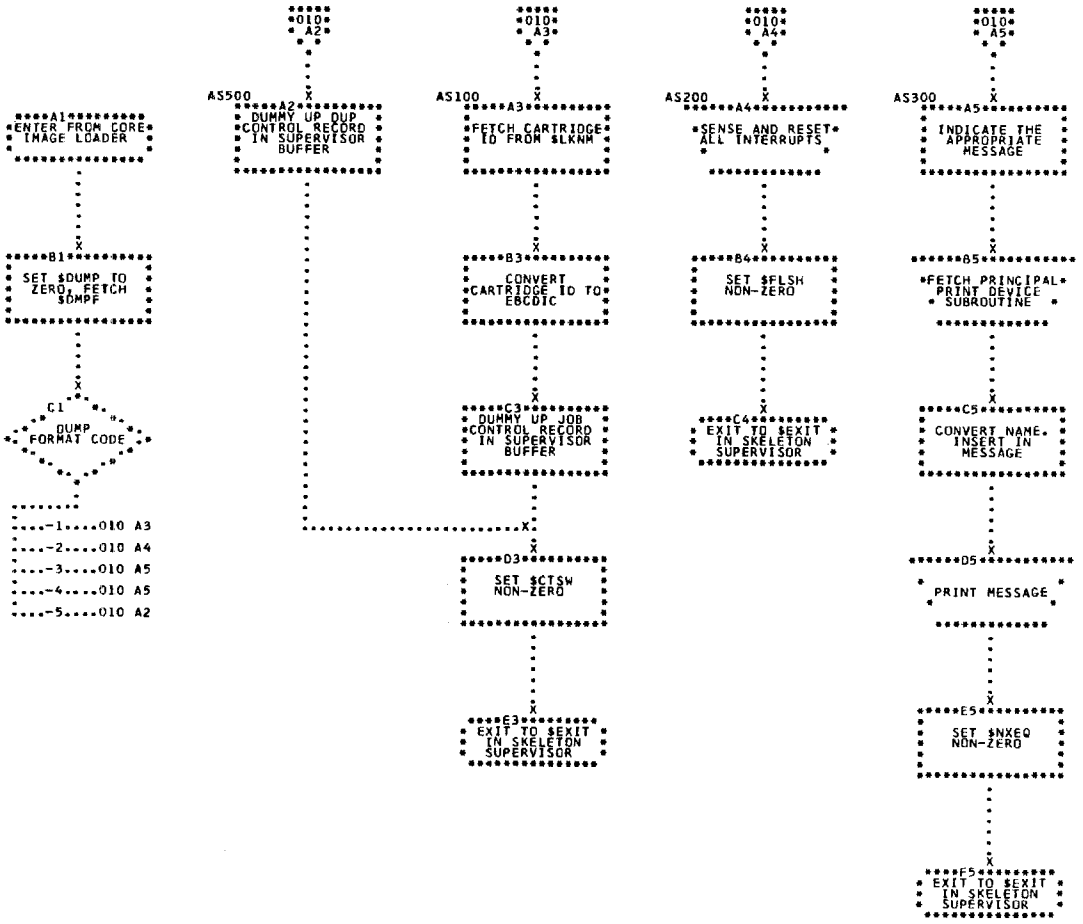
•Flowchart SUP08. Supervisor, Control Record Processor



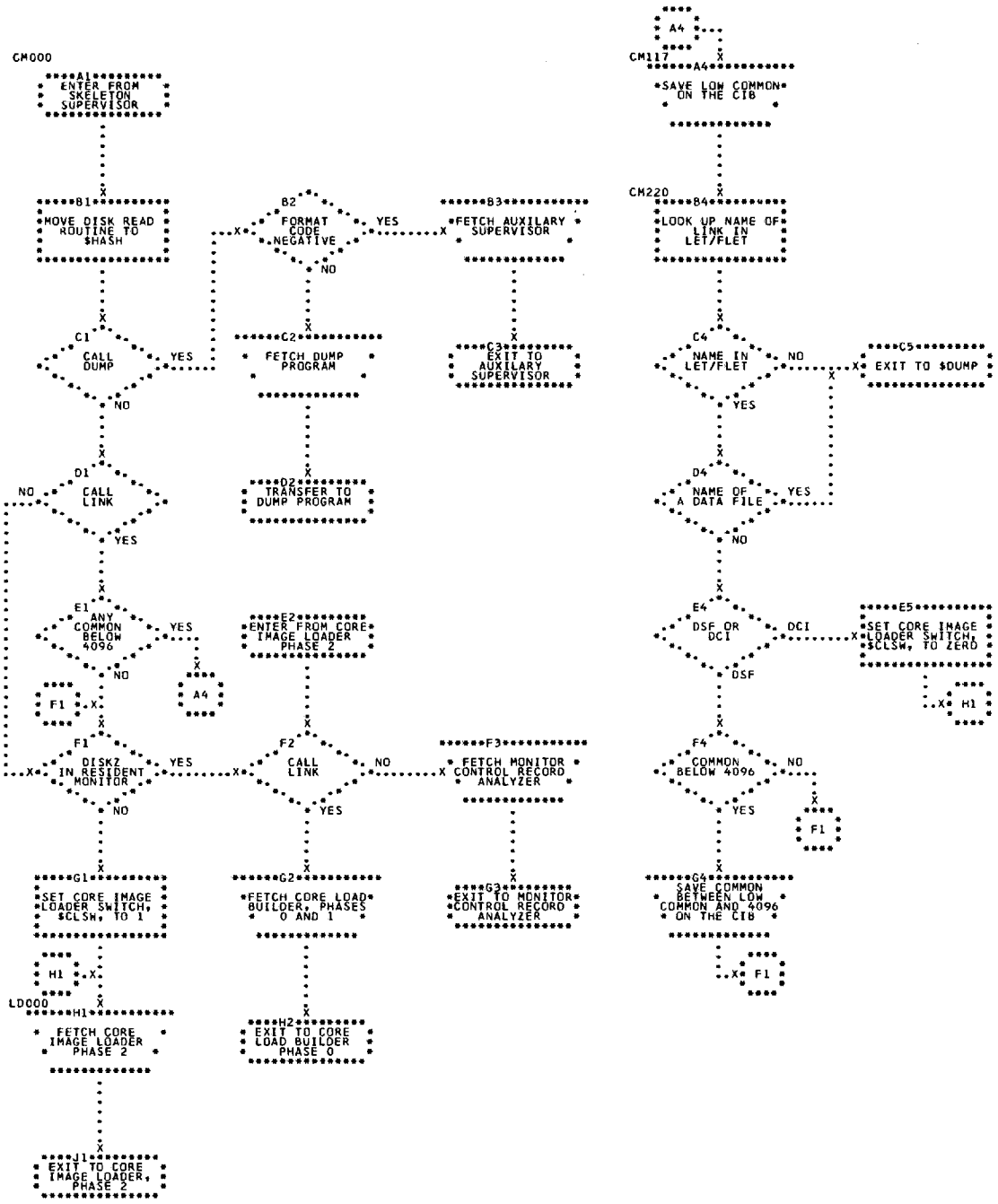
•Flowchart SUP09. Supervisor, Control Record Processor



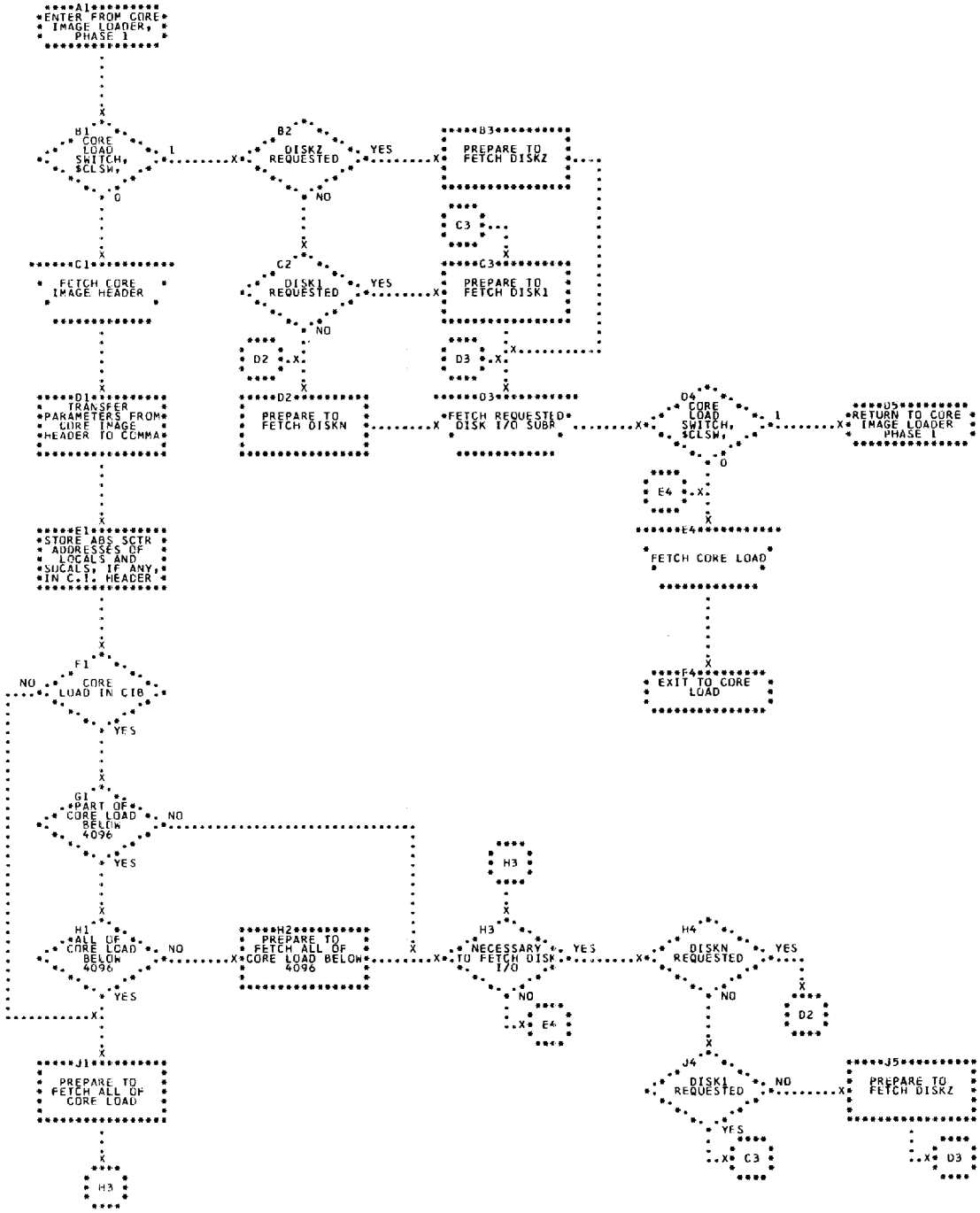
Flowchart SUP10. Supervisor, Auxiliary Supervisor



Flowchart SUP11. Supervisor, System Core Dump Program

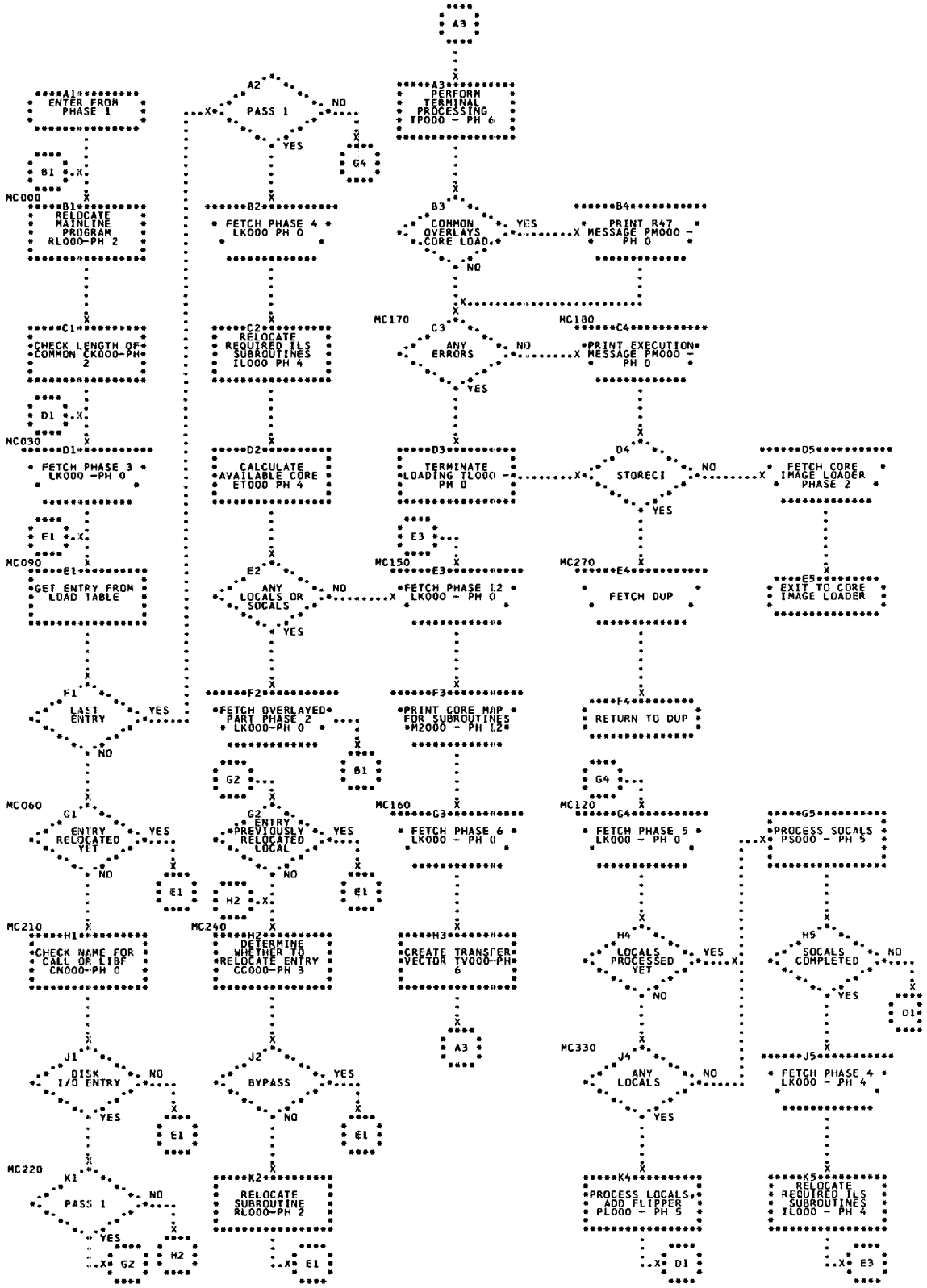


Flowchart CIL01. Core Image Loader, Phase 1

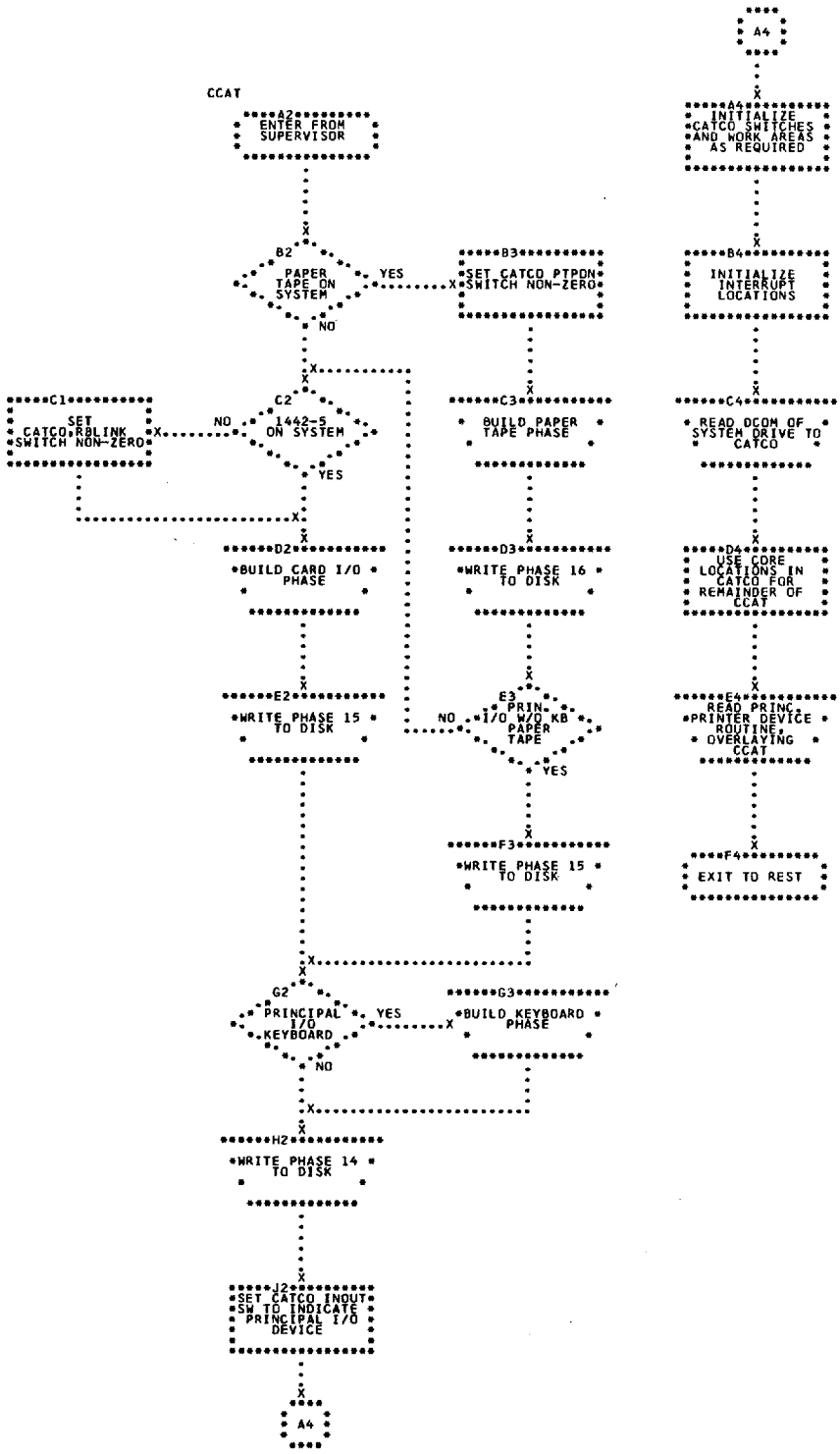


Flowchart CIL02. Core Image Loader, Phase 2

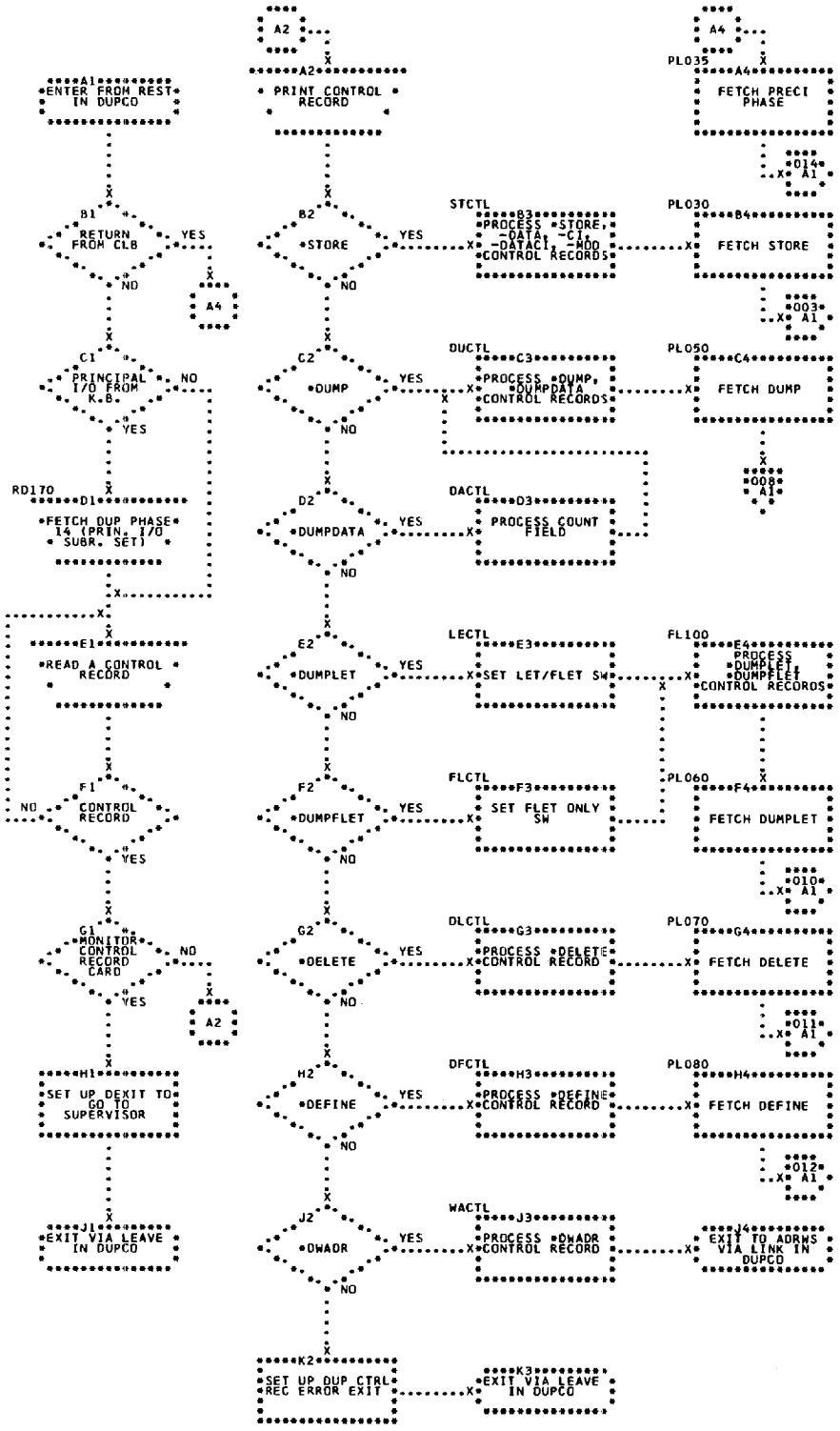
Flowchart CLB01. Core Load Builder, Initialization



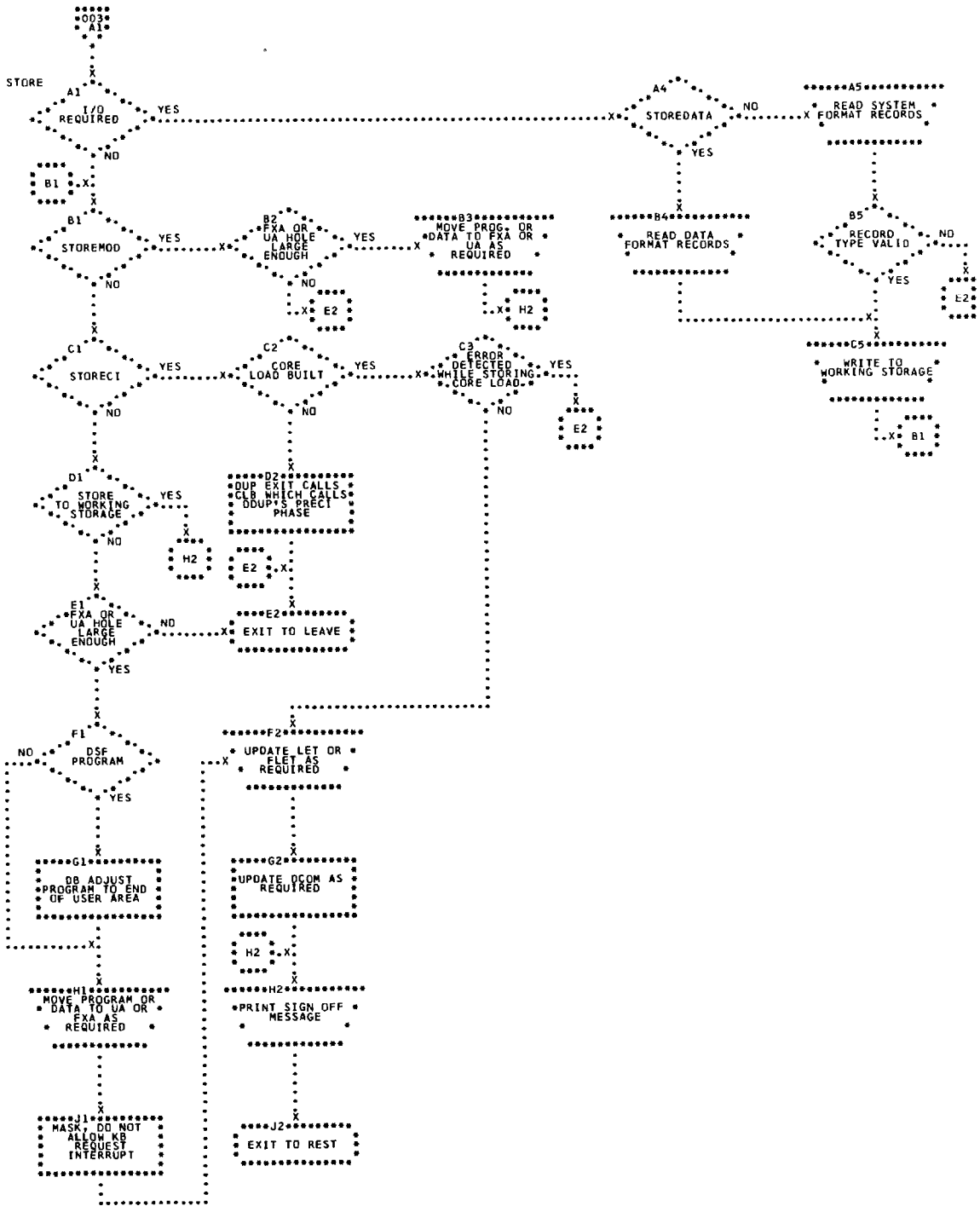
Flowchart CLB02. Core Load Builder, Master Control



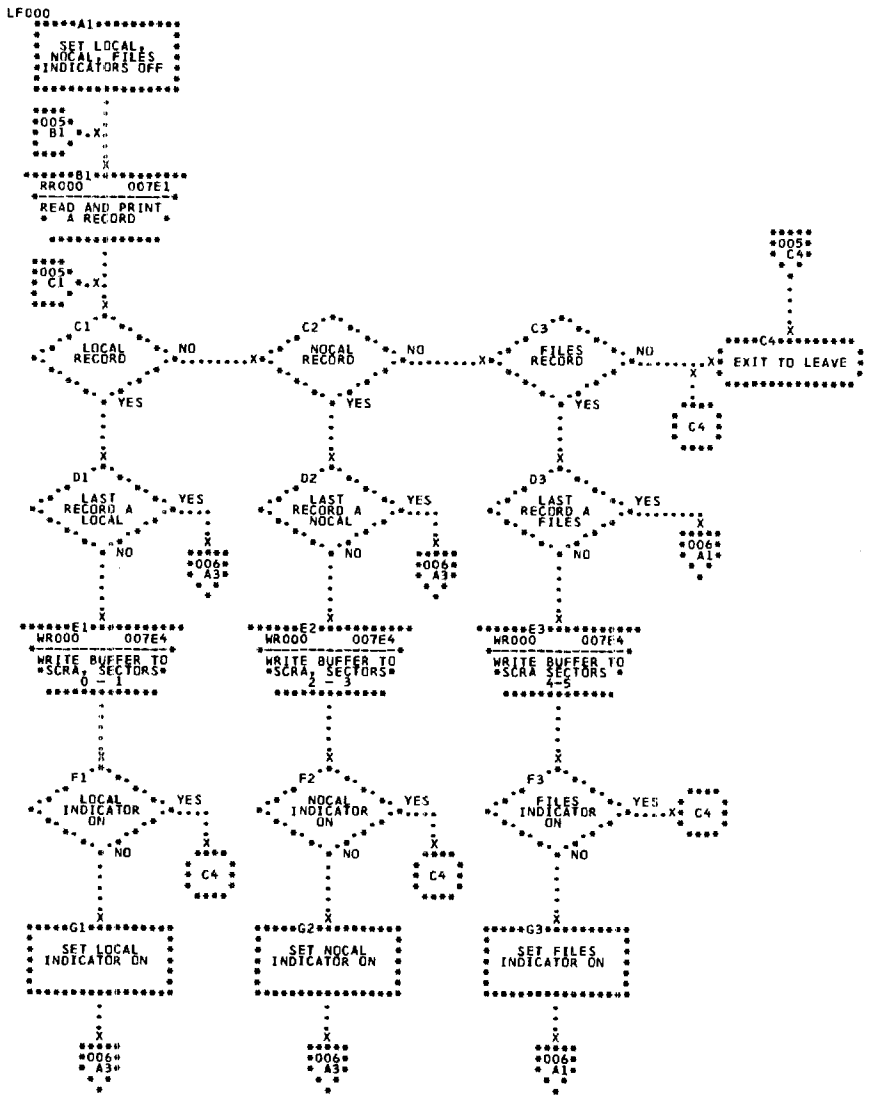
Flowchart DUP01. Disk Utility Program, CCAT



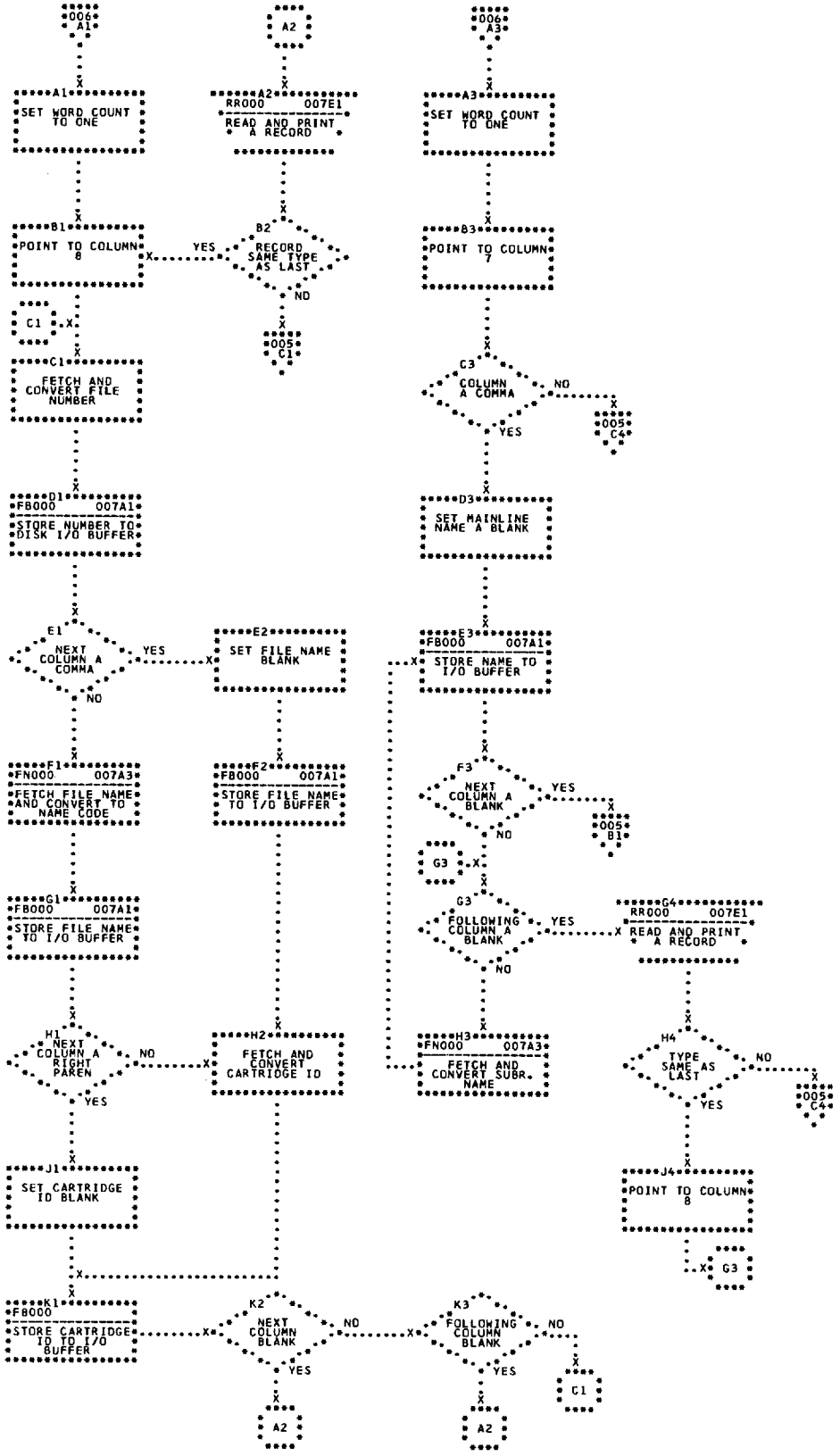
Flowchart DUP02. Disk Utility Program, DCTL



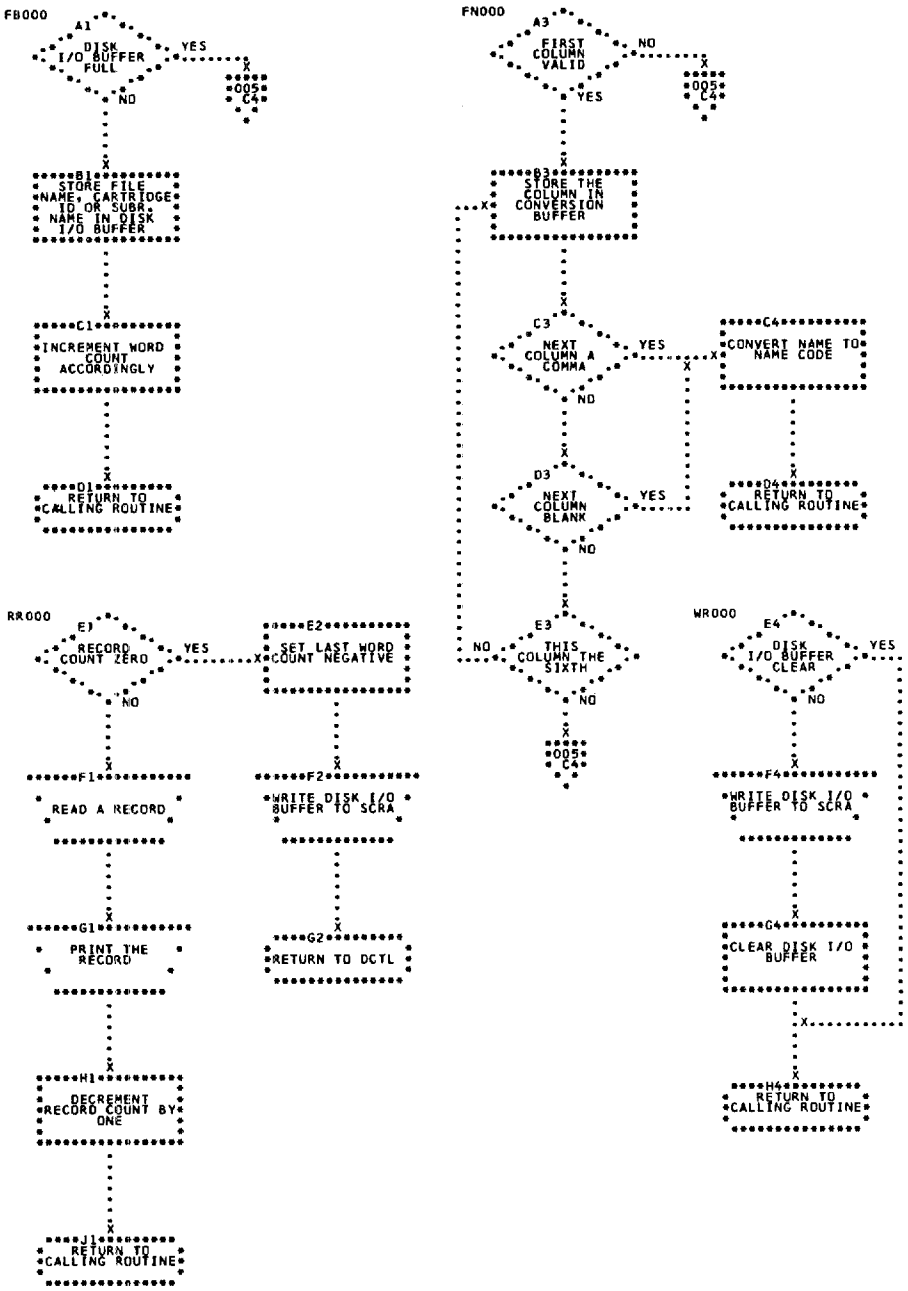
Flowchart DUP03. Disk Utility Program, STORE



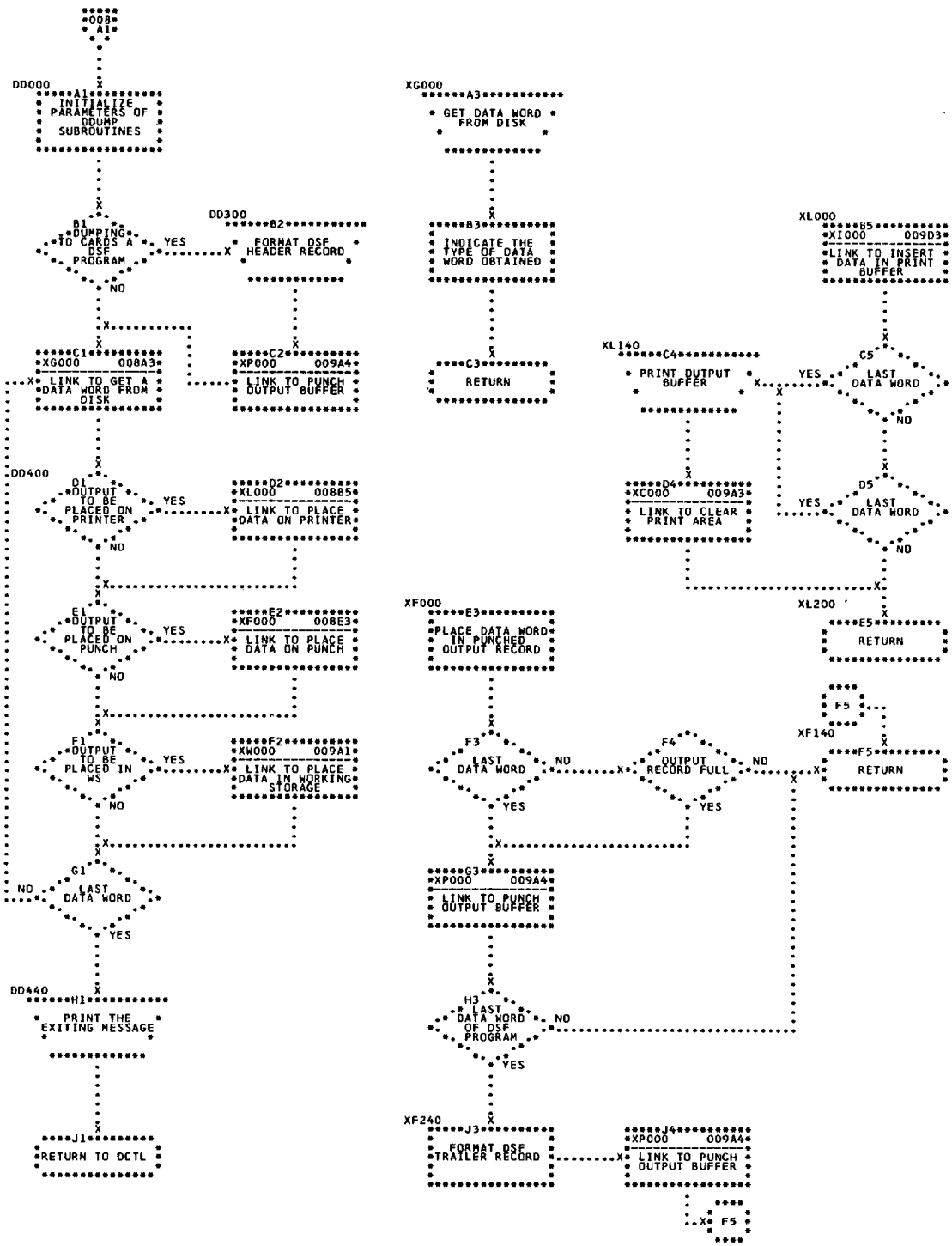
Flowchart DUP04. Disk Utility Program, FILEQ



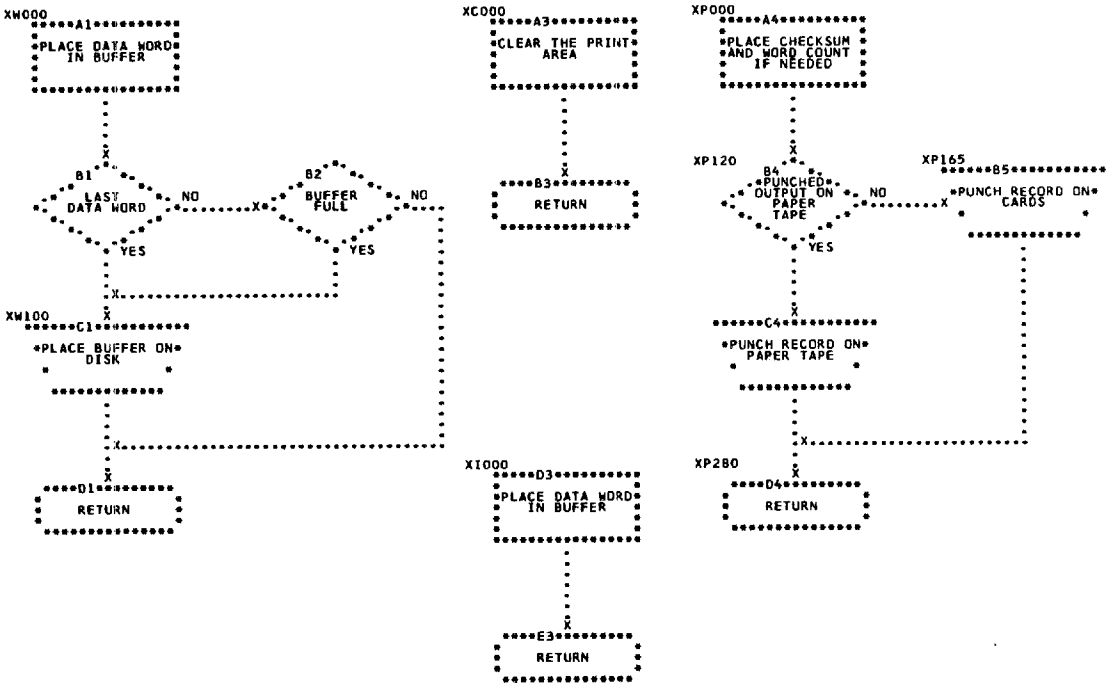
Flowchart DUP05. Disk Utility Program, FILEQ



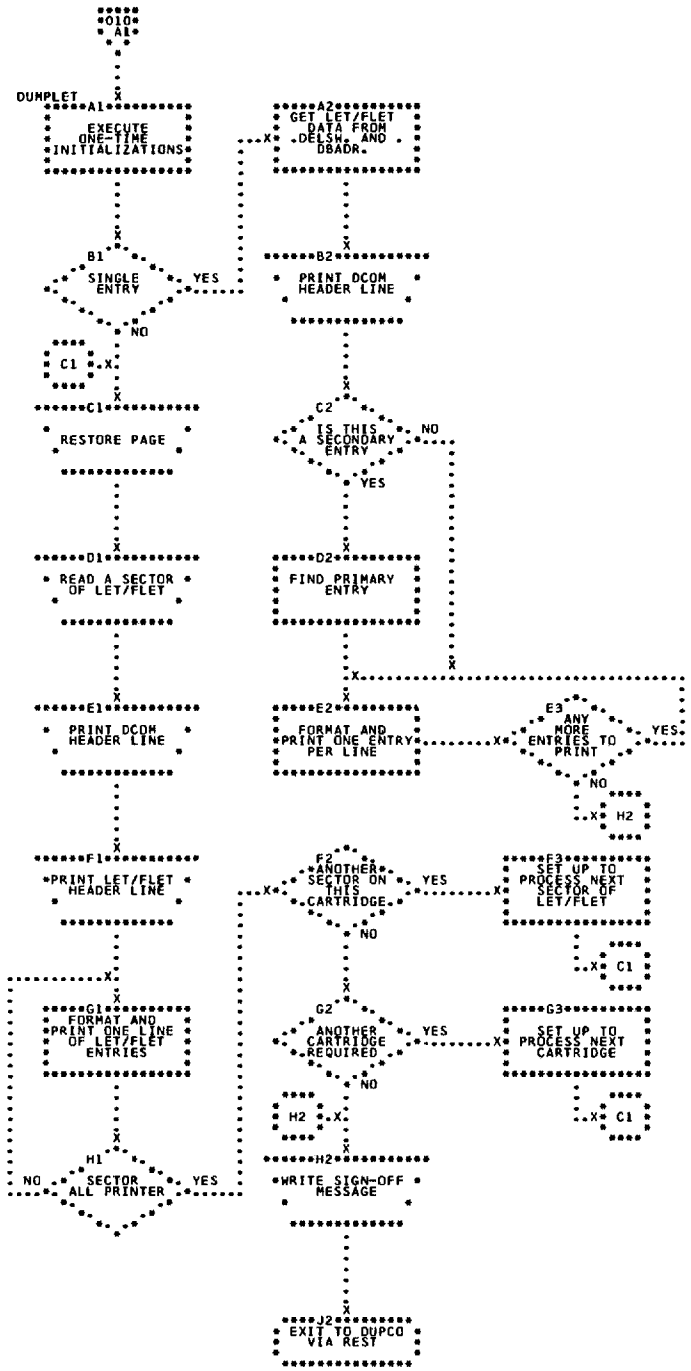
Flowchart DUP06. Disk Utility Program, FILEQ



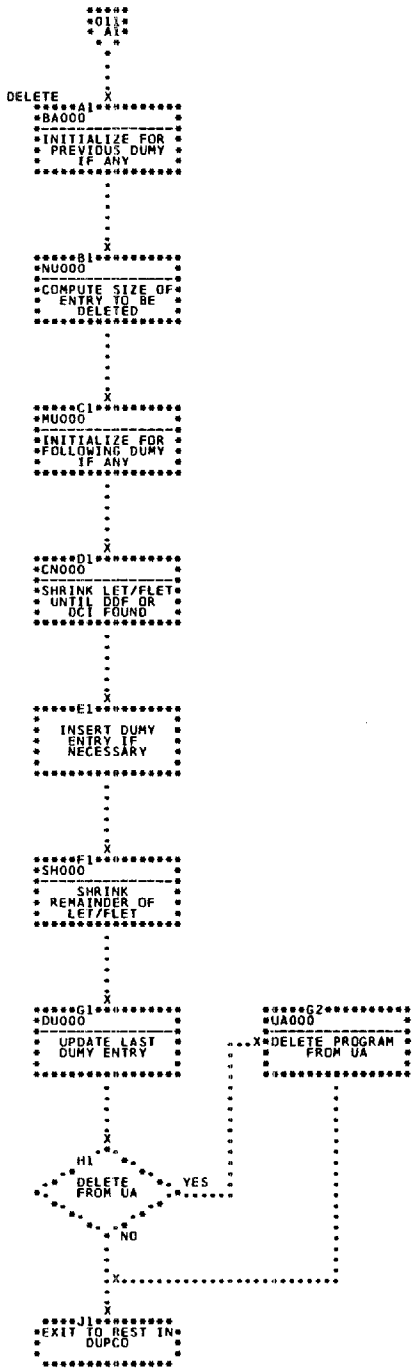
Flowchart DUP07. Disk Utility Program, DDUMP



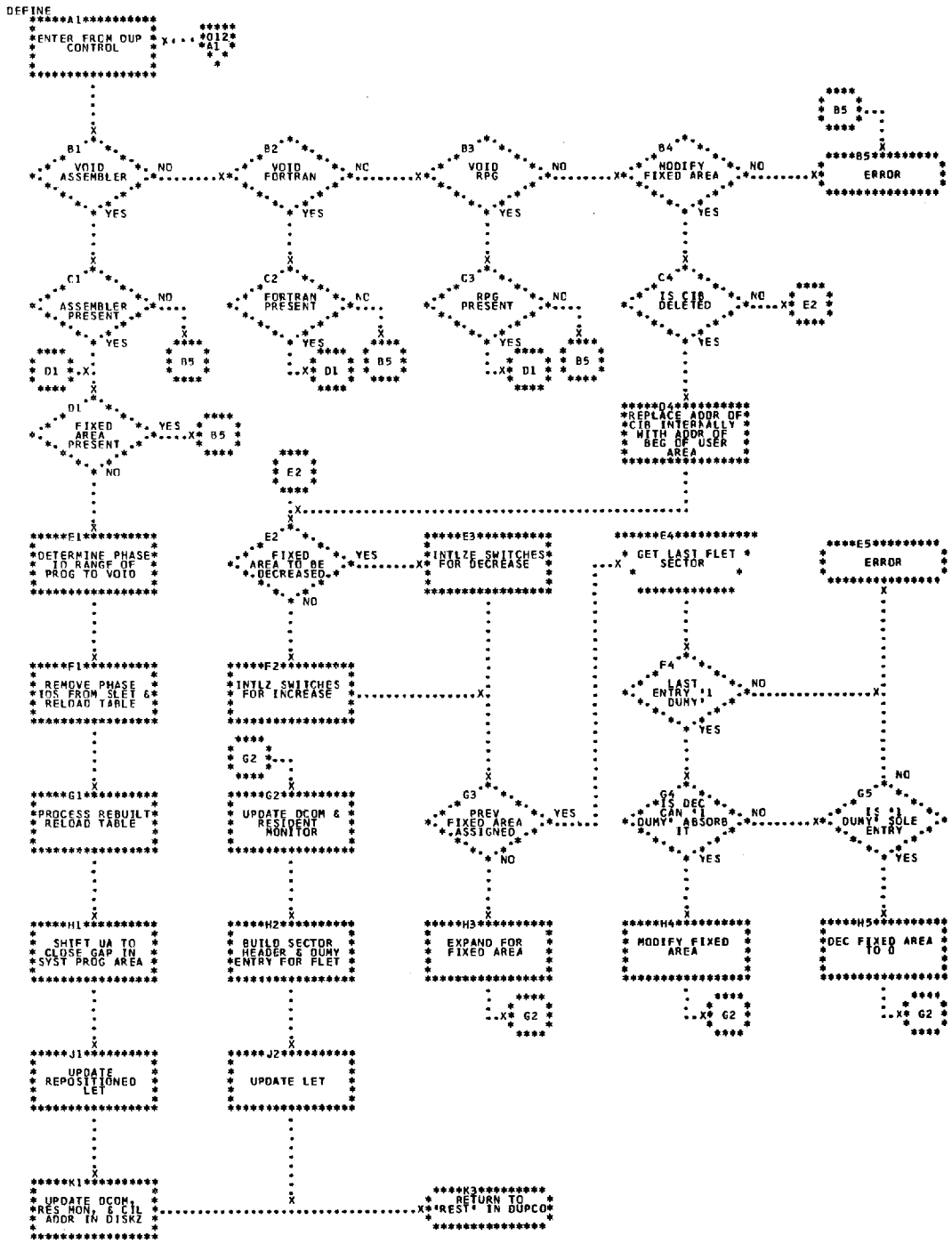
Flowchart DUP08. Disk Utility Program, DDUMP



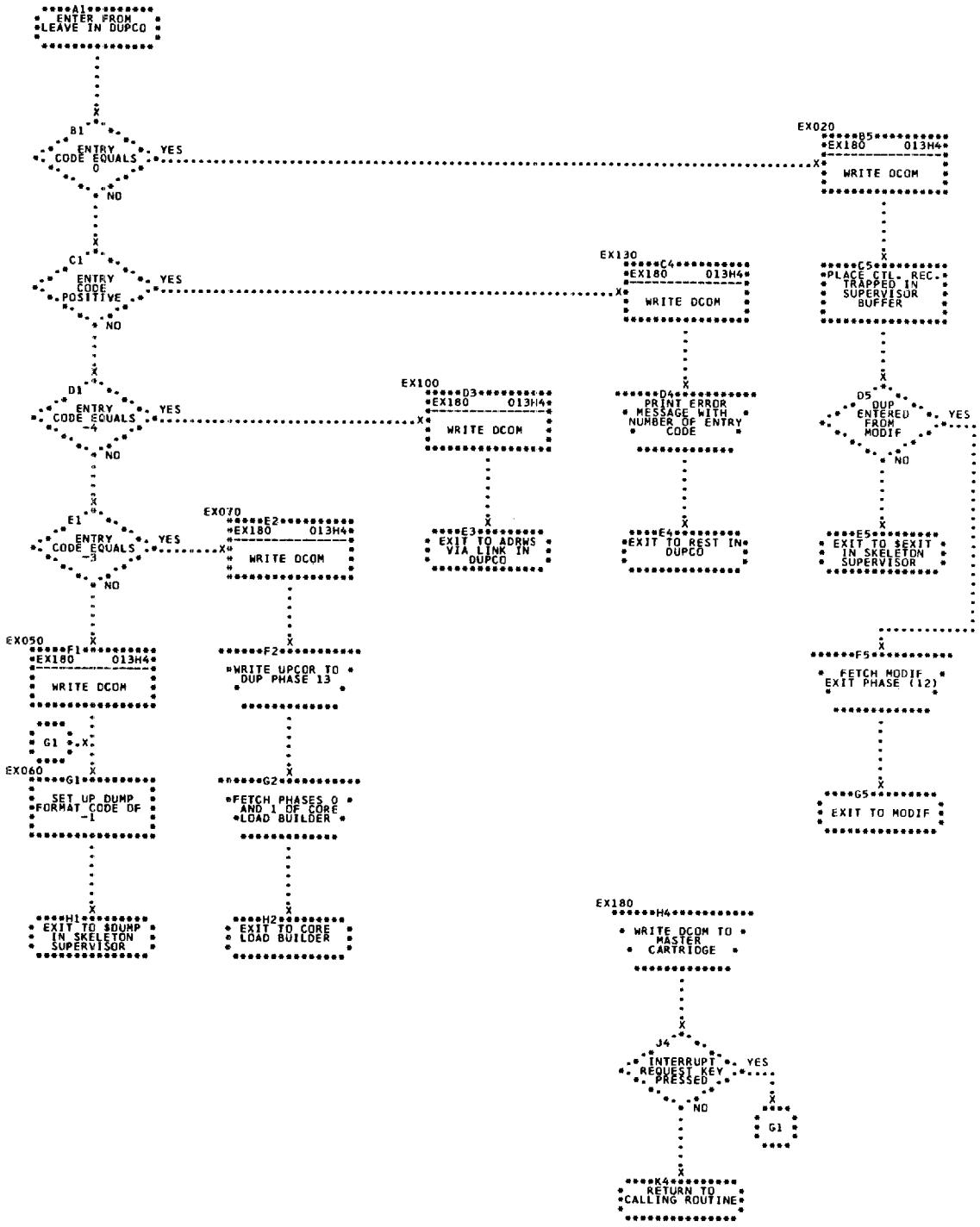
Flowchart DUP09. Disk Utility Program, DUMPLET/DUMPFLET



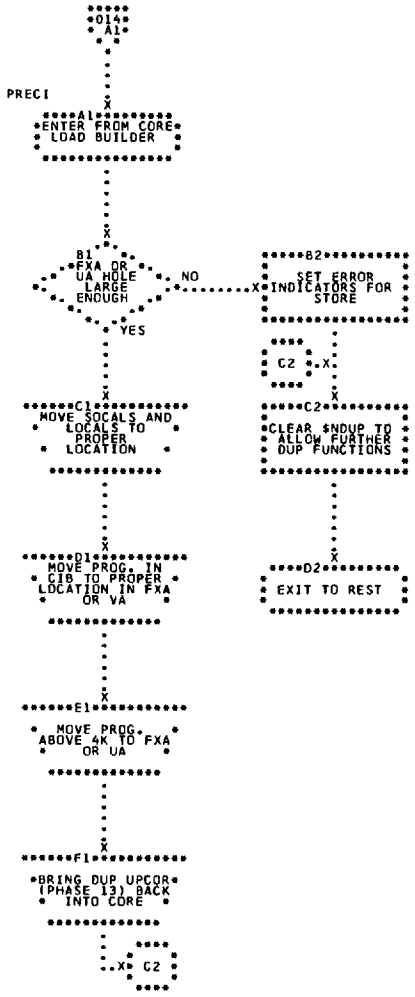
Flowchart DUP10. Disk Utility Program, DELETE



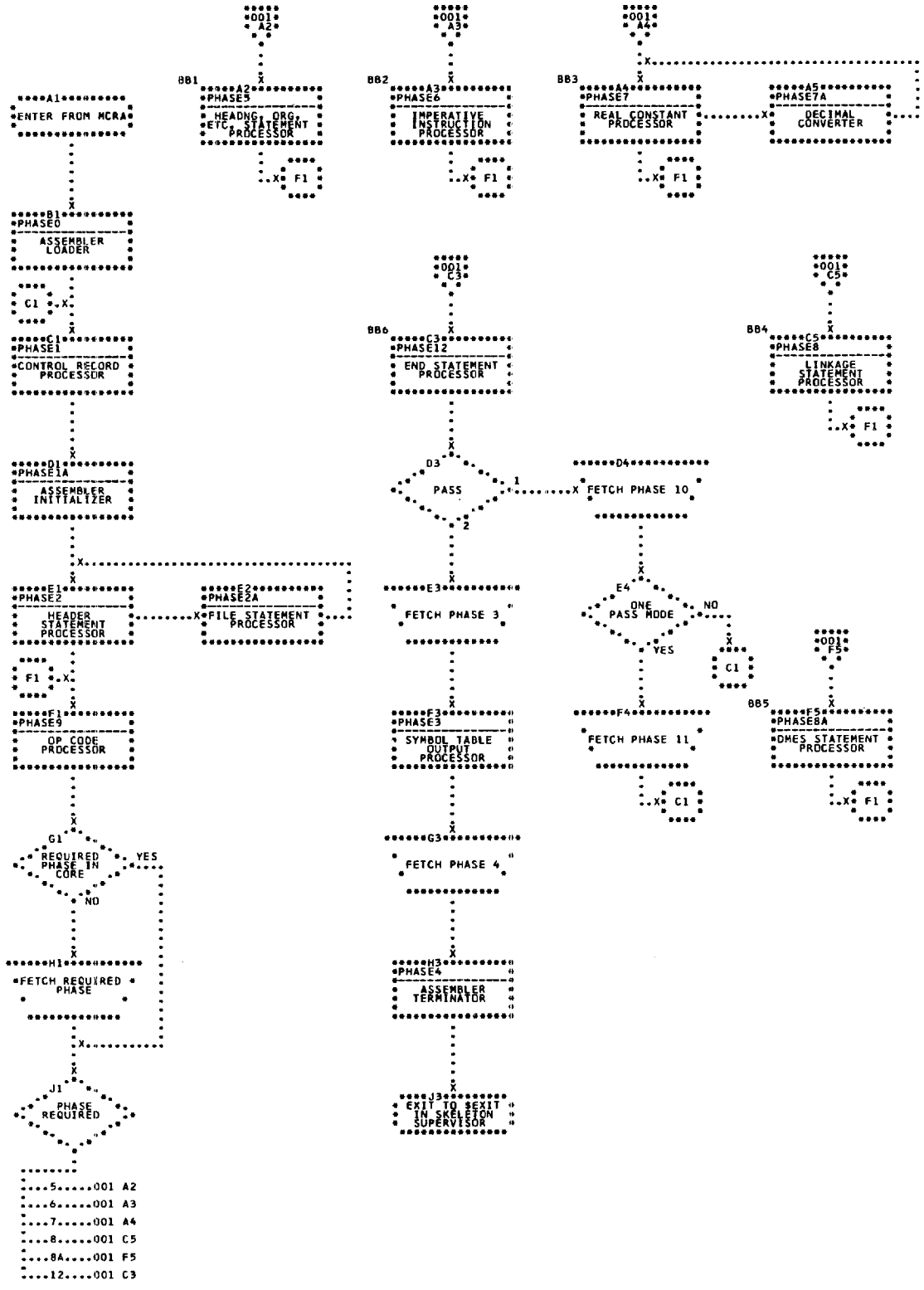
Flowchart DUP11. Disk Utility Program, DEFINE



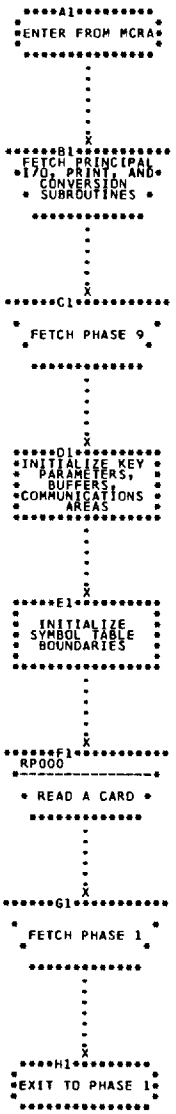
Flowchart DUP12. Disk Utility Program, DEXIT



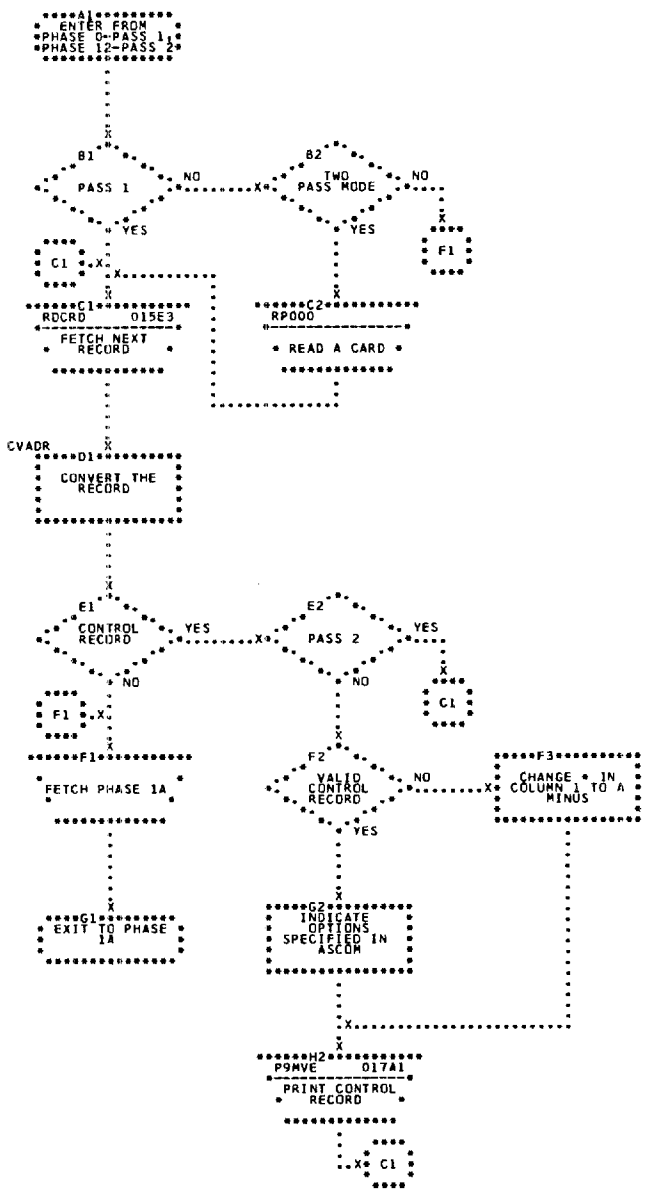
Flowchart DUP13. Disk Utility Program, PRECI



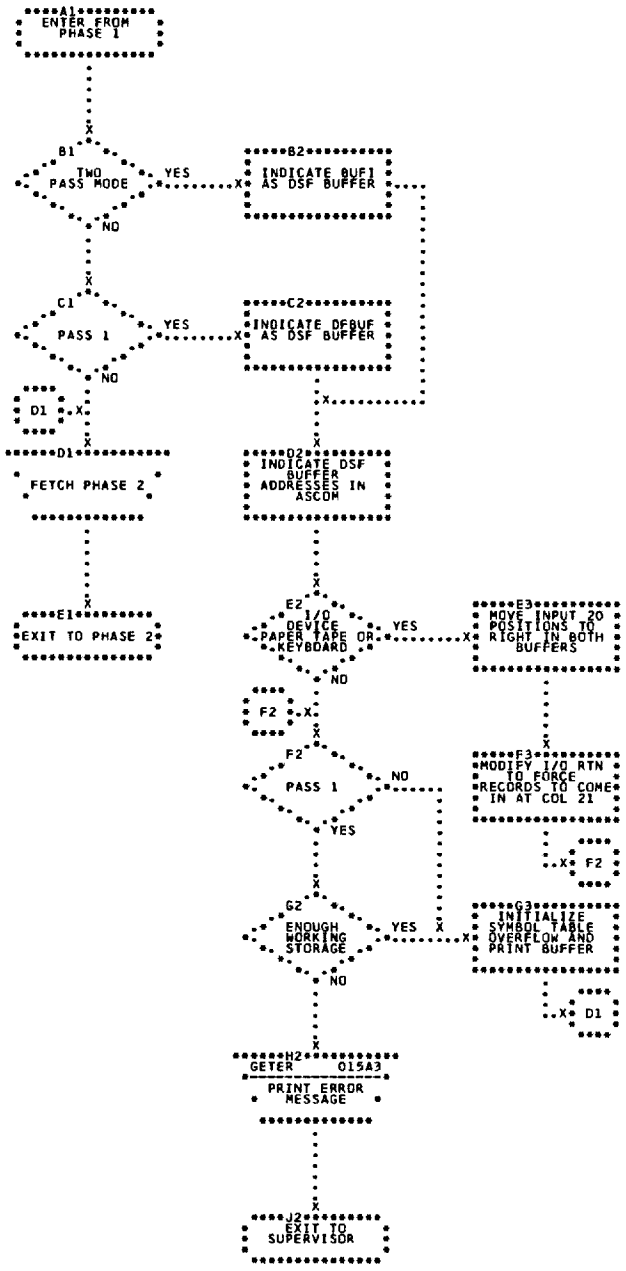
Flowchart ASM01. Assembler Program, General Flow



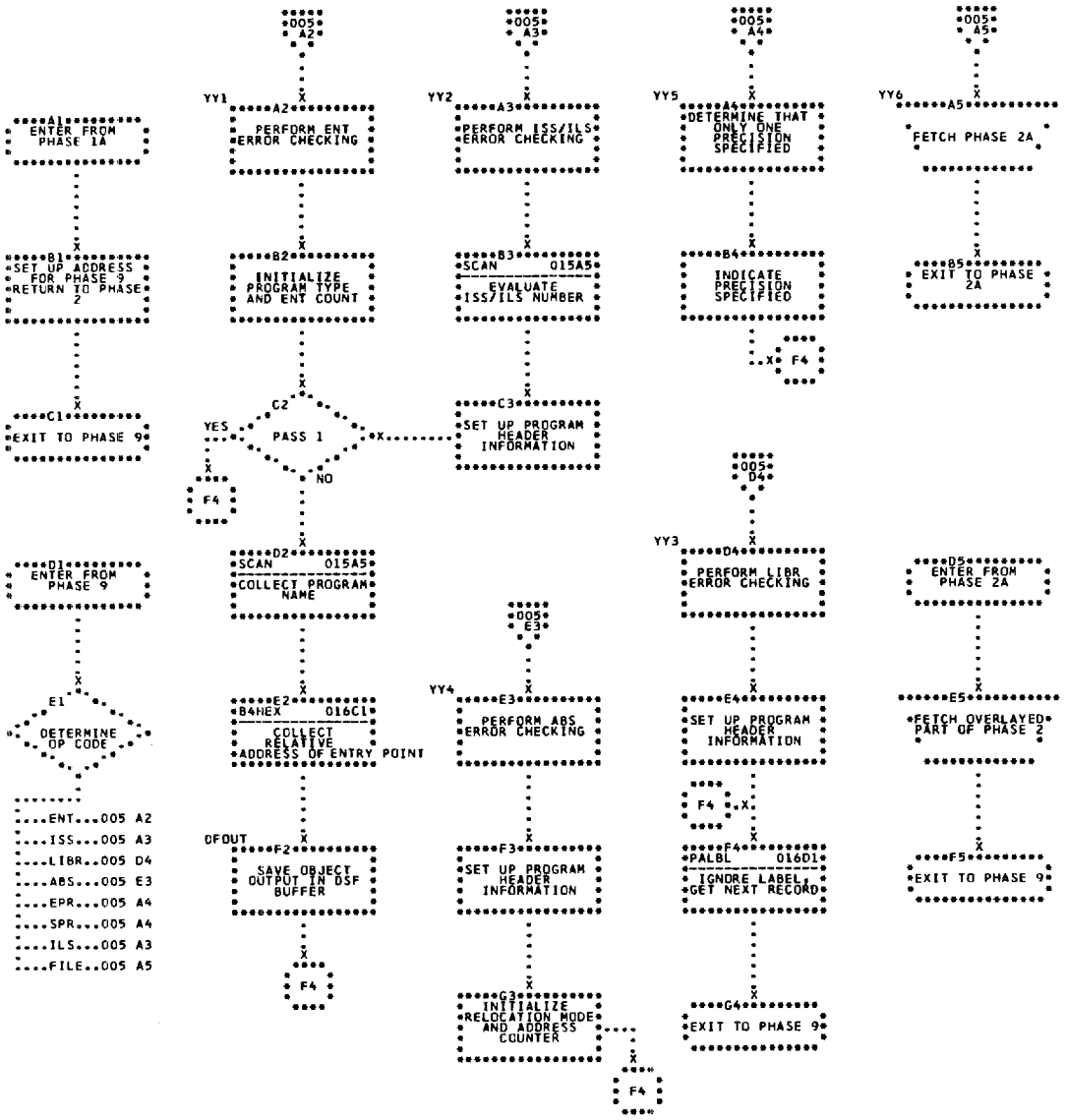
Flowchart ASM02. Assembler Program, Phase 0



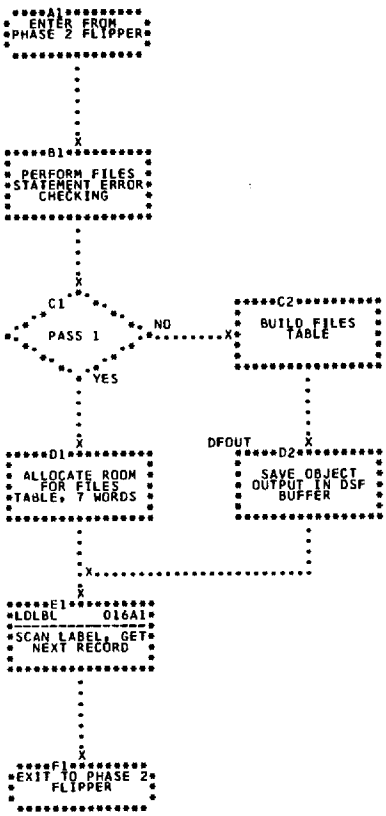
Flowchart ASM03. Assembler Program, Phase 1



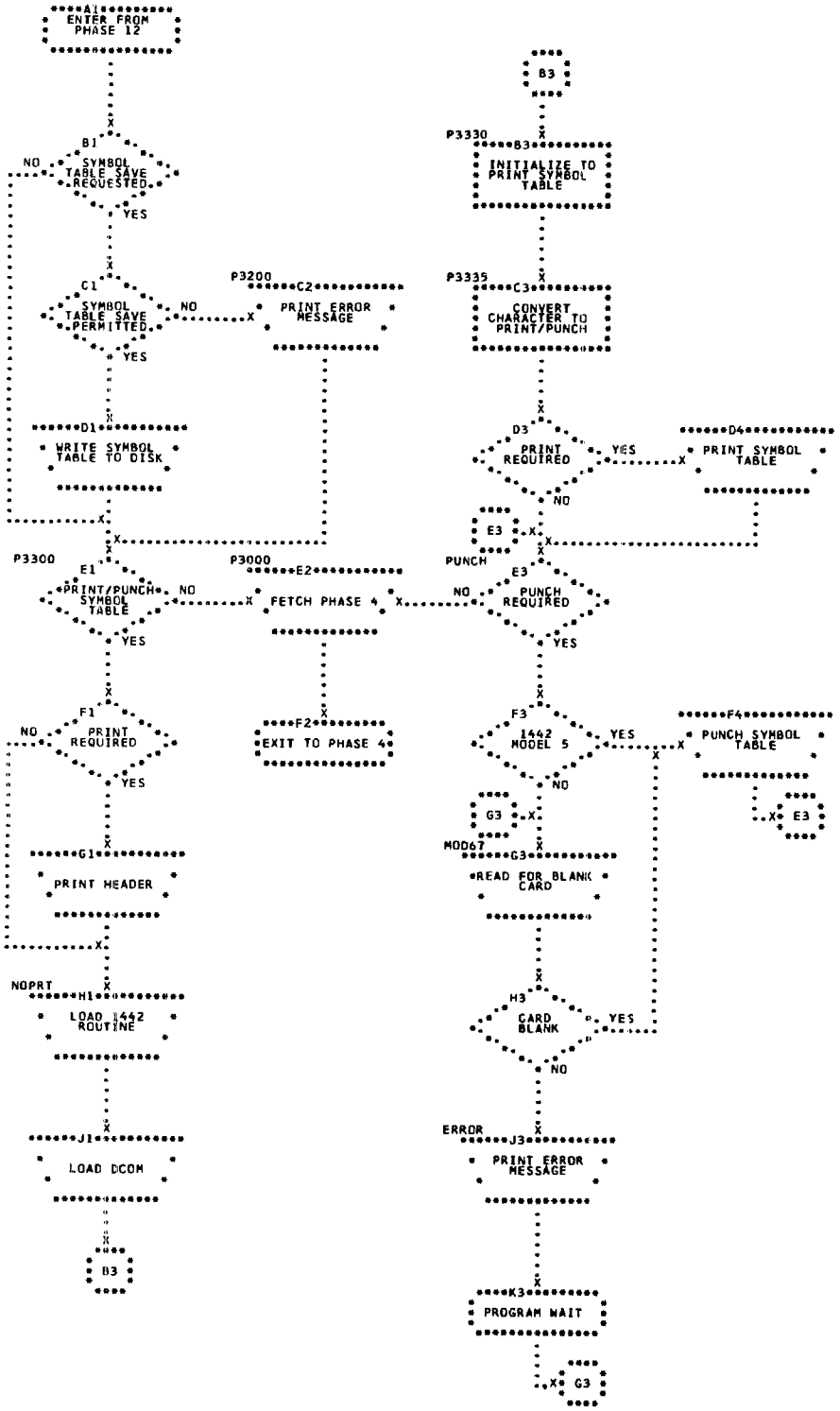
Flowchart ASM04. Assembler Program, Phase 1A



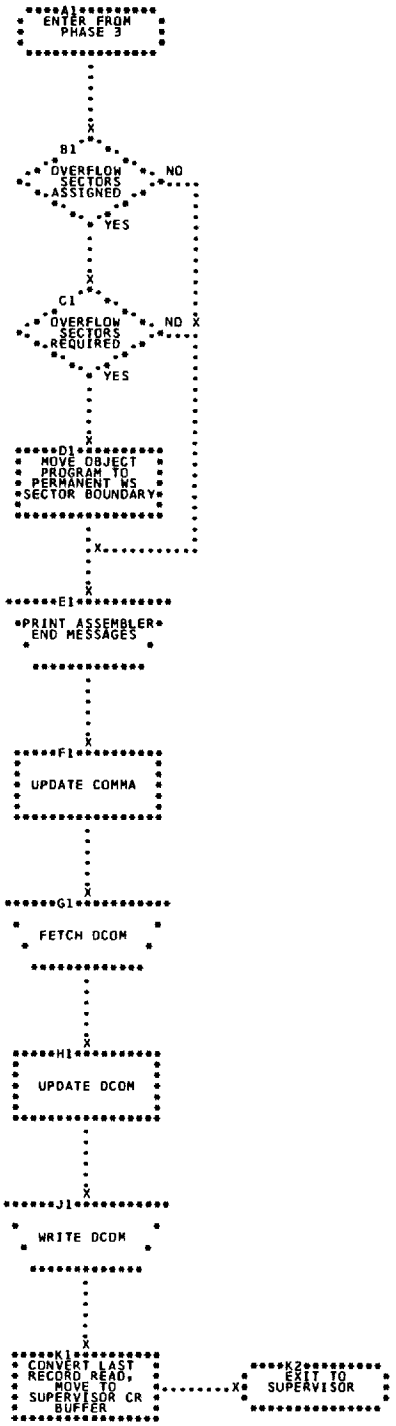
Flowchart ASM05. Assembler Program, Phase 2



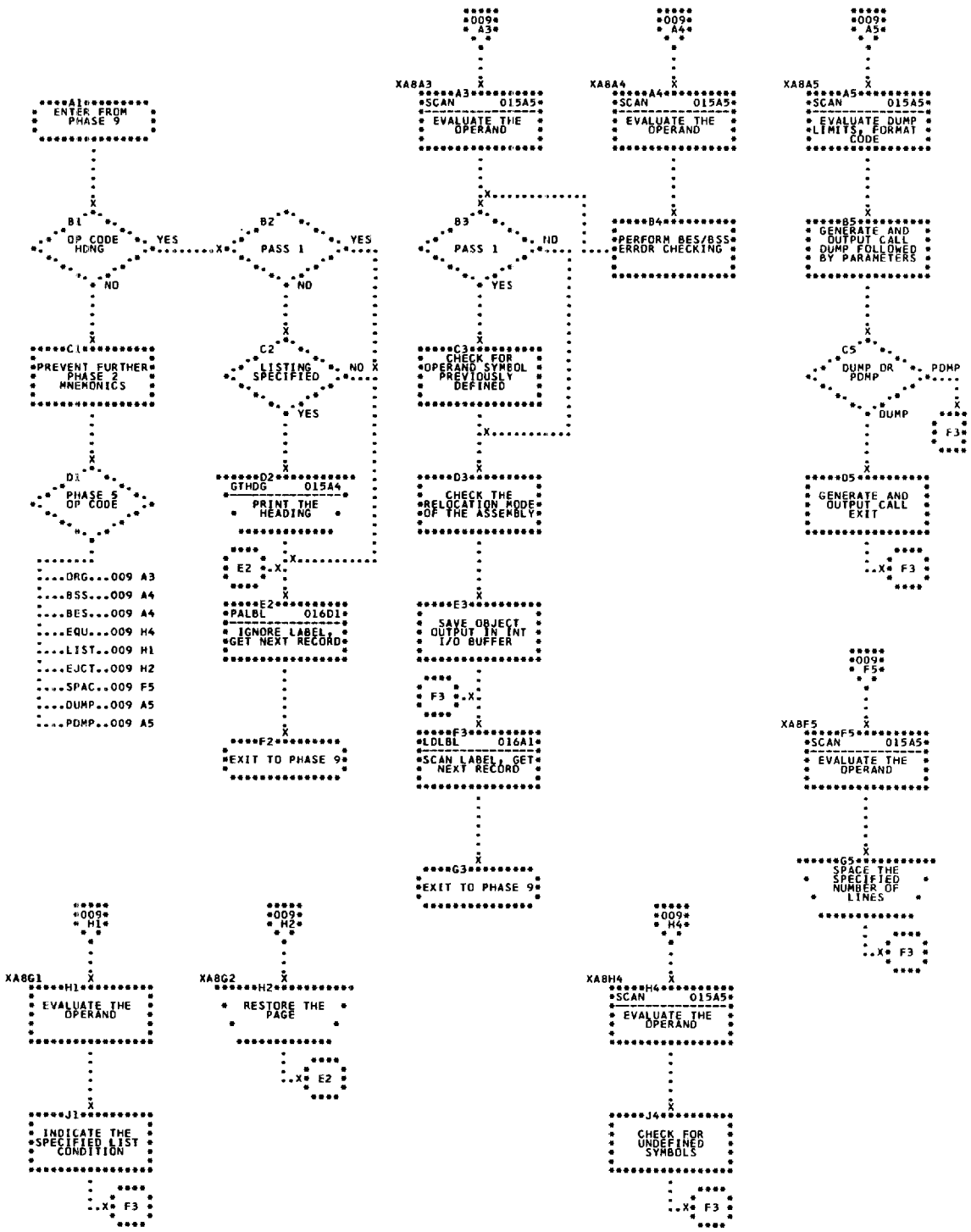
Flowchart ASM06. Assembler Program, Phase 2A



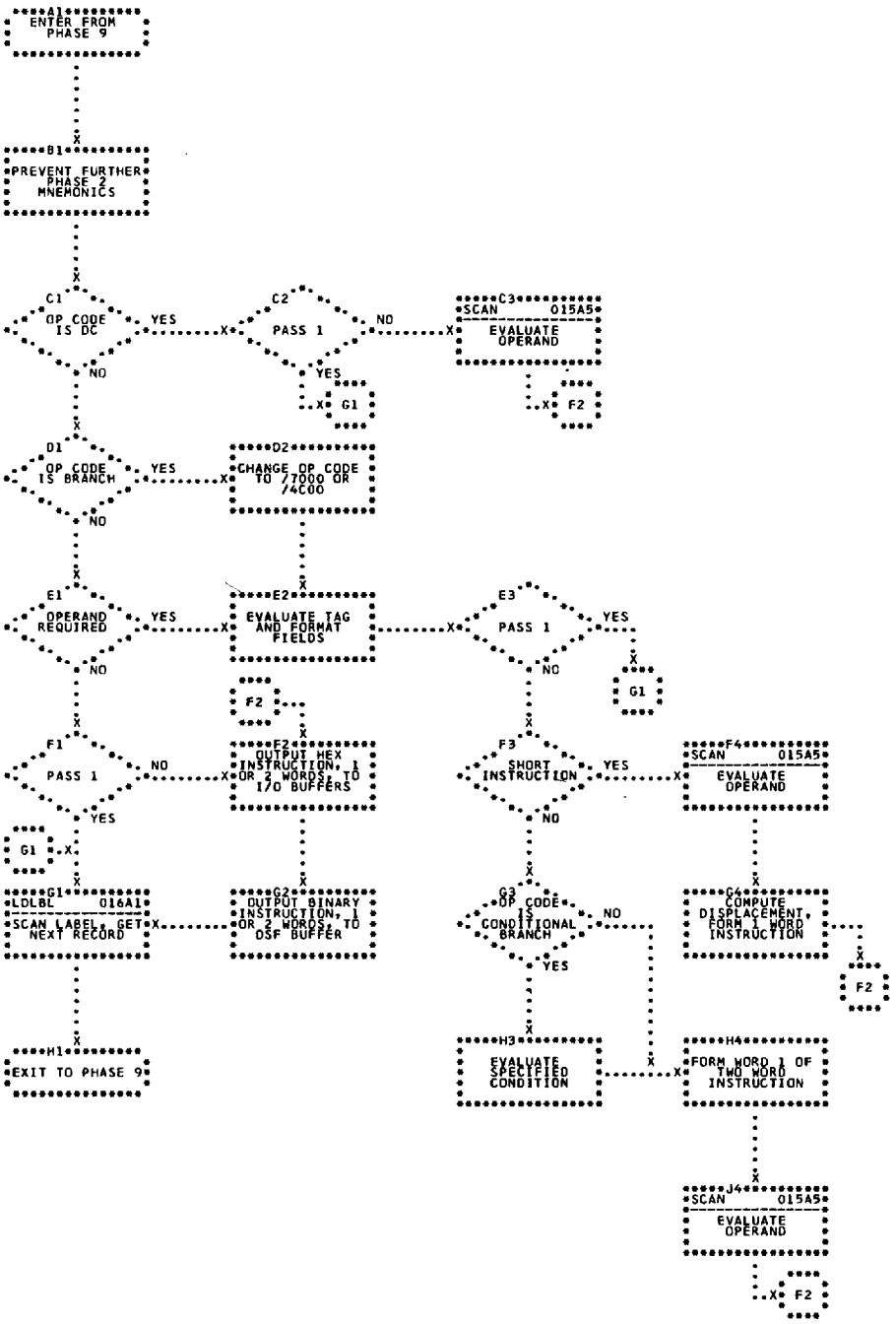
Flowchart ASM07. Assembler Program, Phase 3



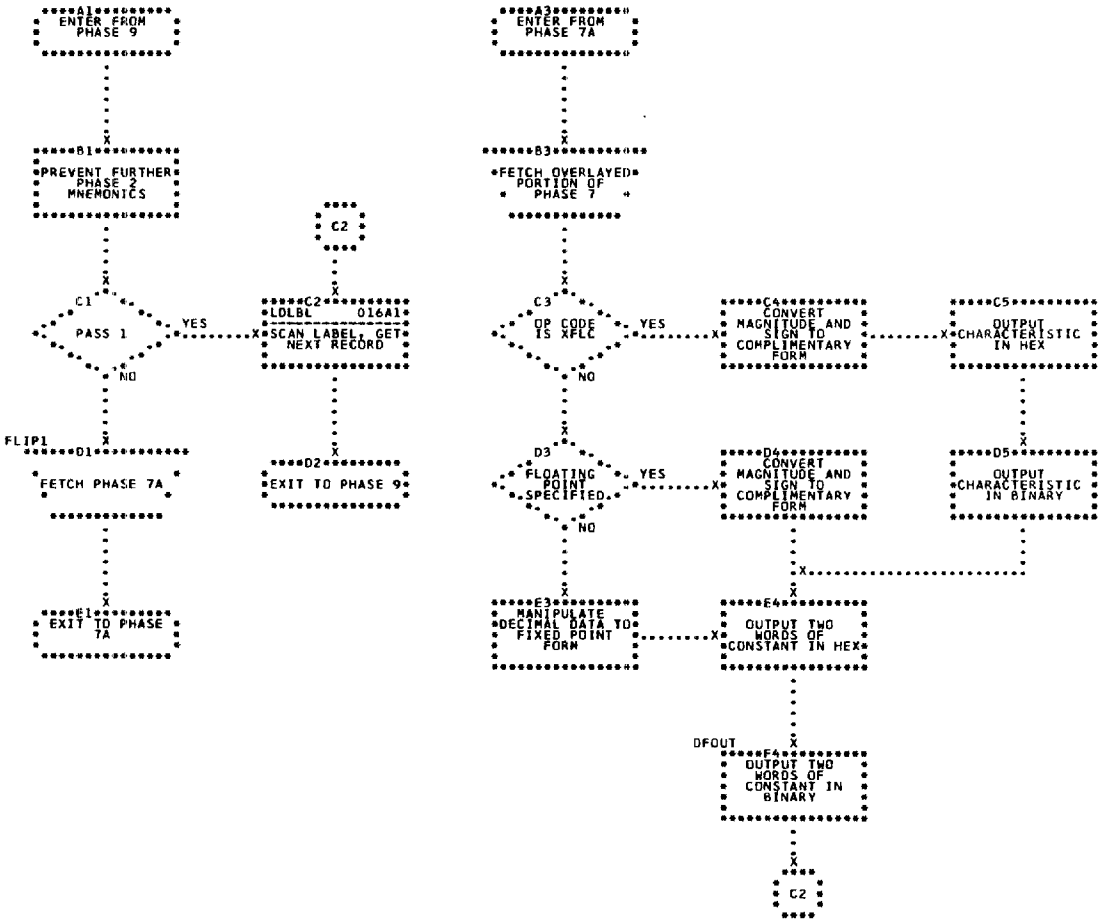
Flowchart ASM08. Assembler Program, Phase 4



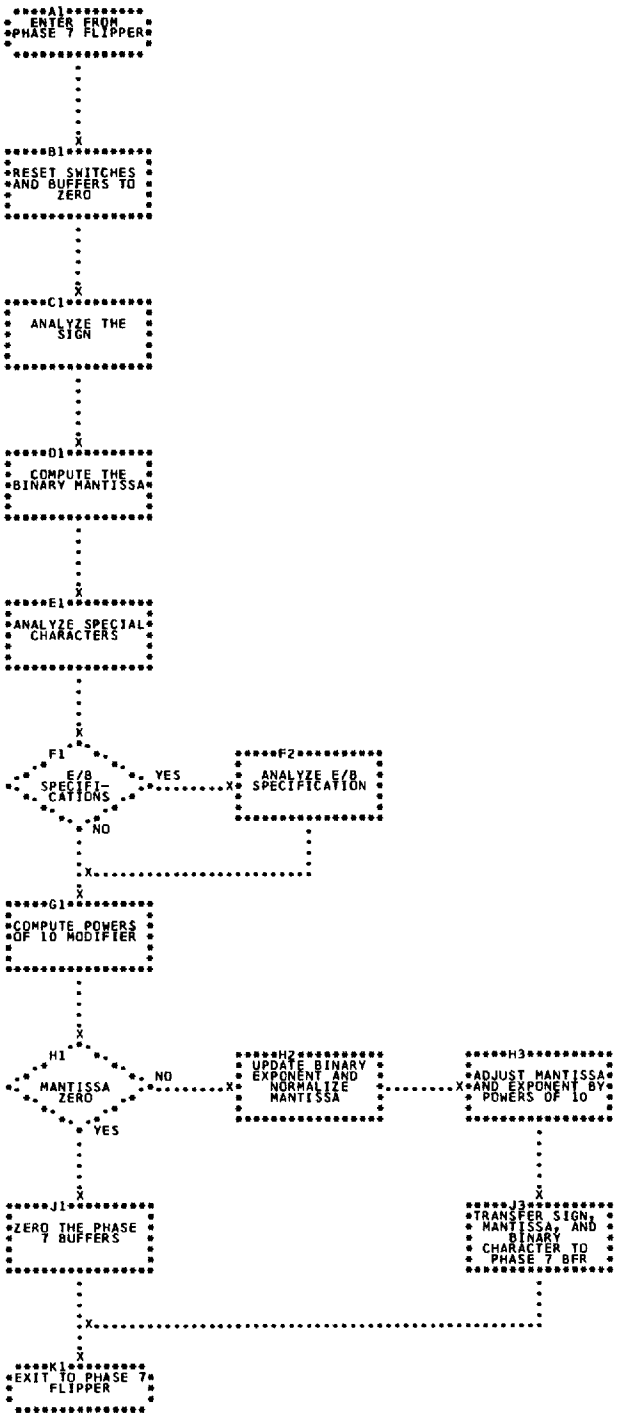
Flowchart ASM09. Assembler Program, Phase 5



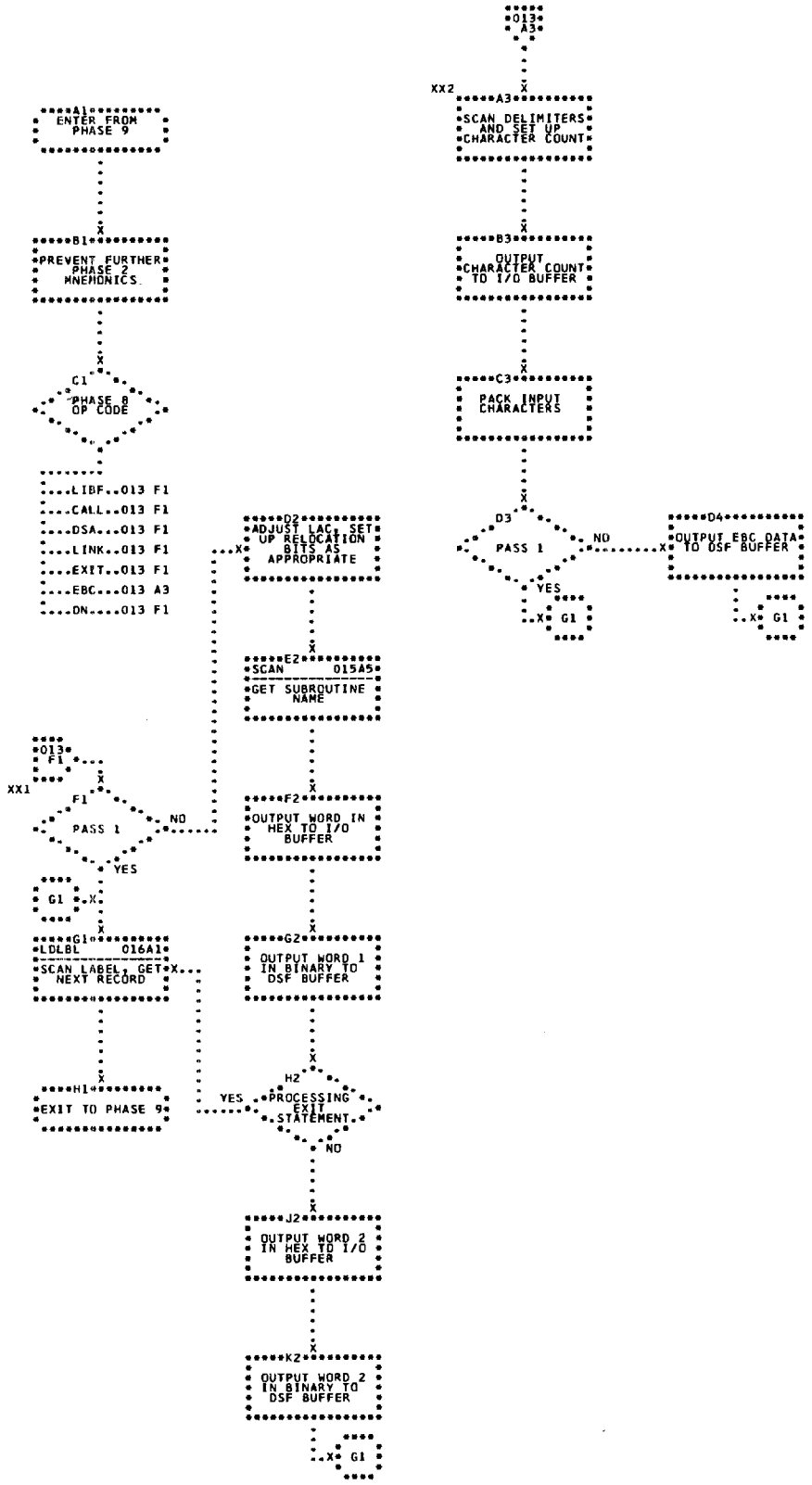
Flowchart ASM10. Assembler Program, Phase 6



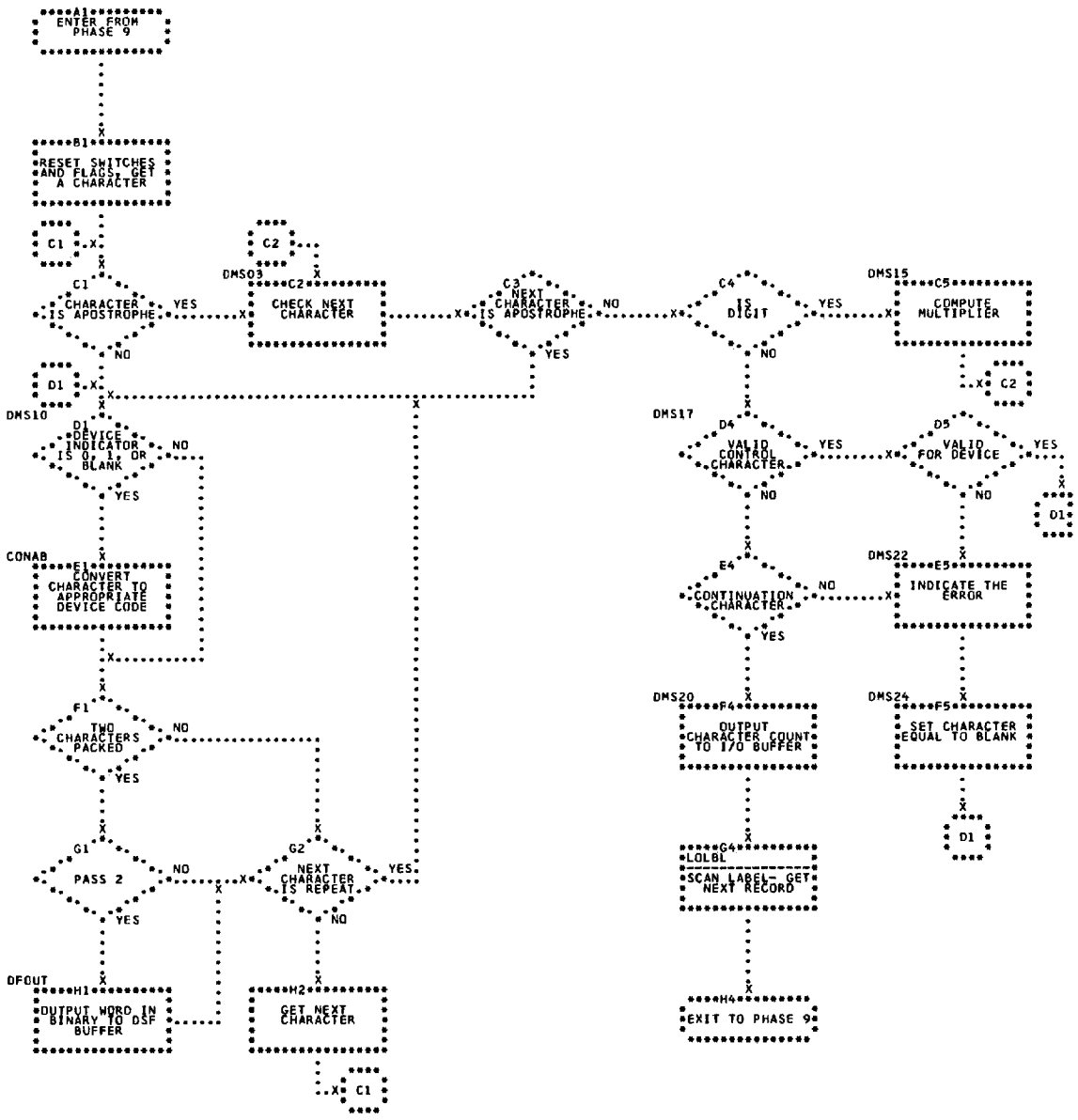
Flowchart ASM11. Assembler Program, Phase 7



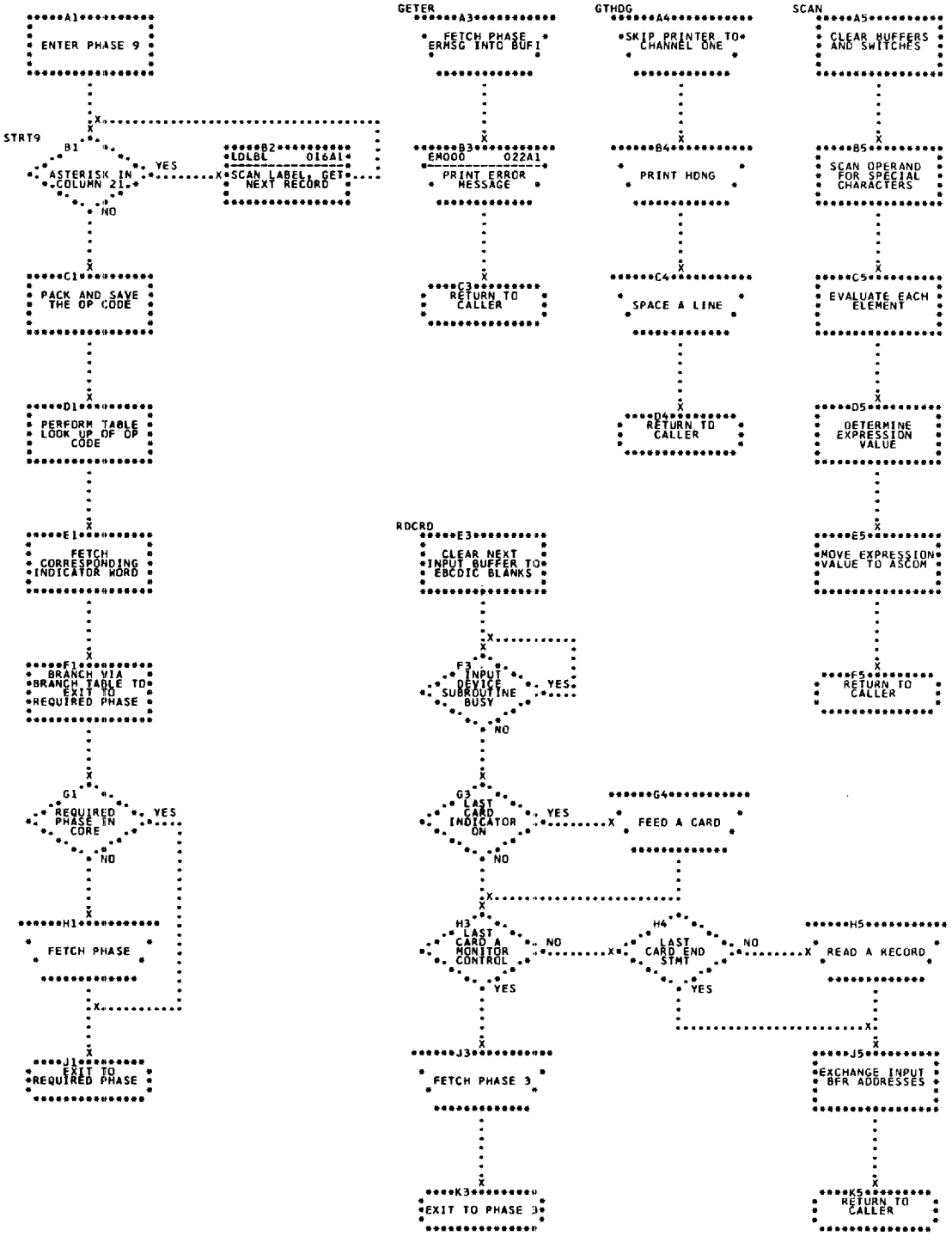
Flowchart ASM12. Assembler Program, Phase 7A



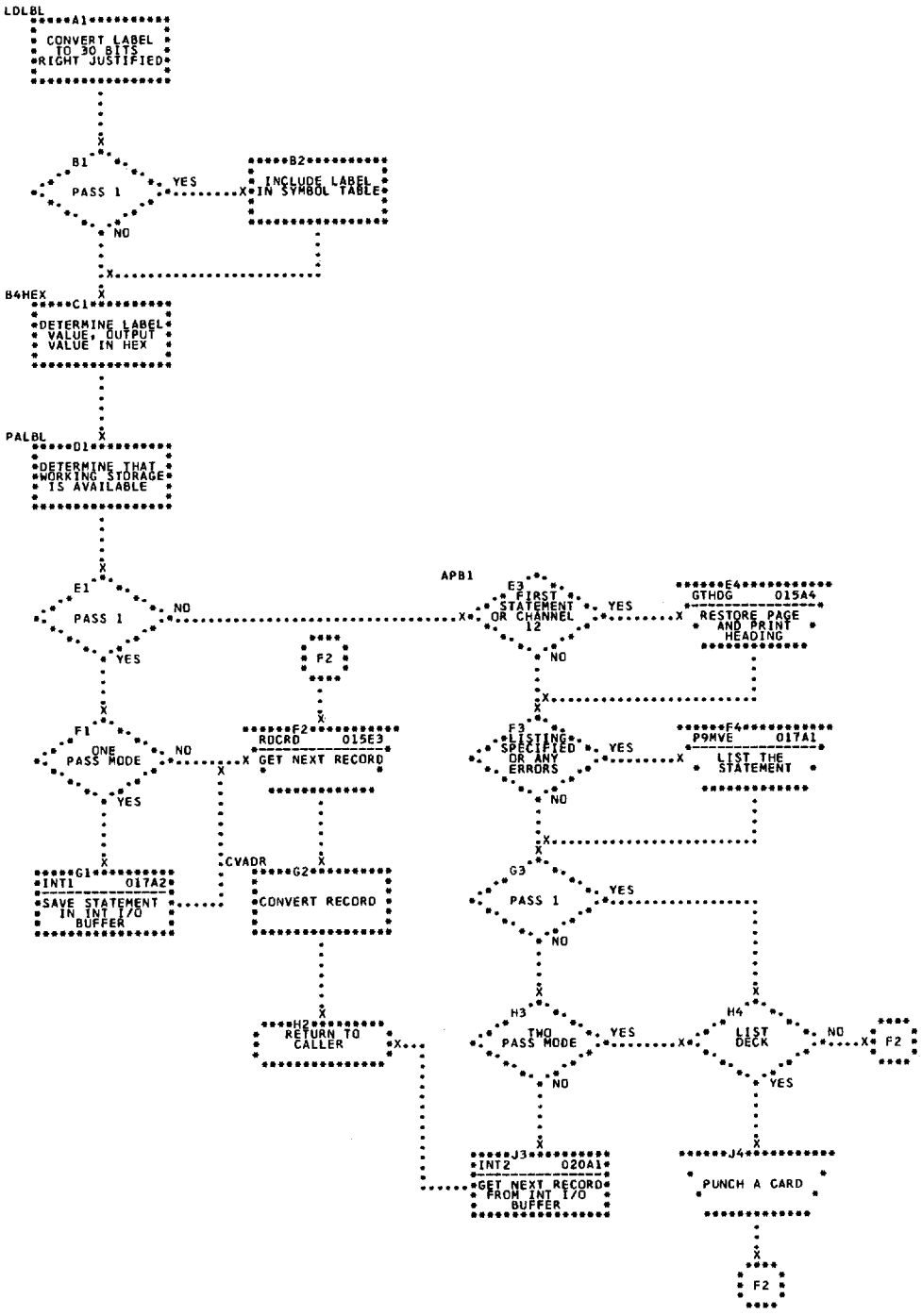
Flowchart ASM13. Assembler Program, Phase 8



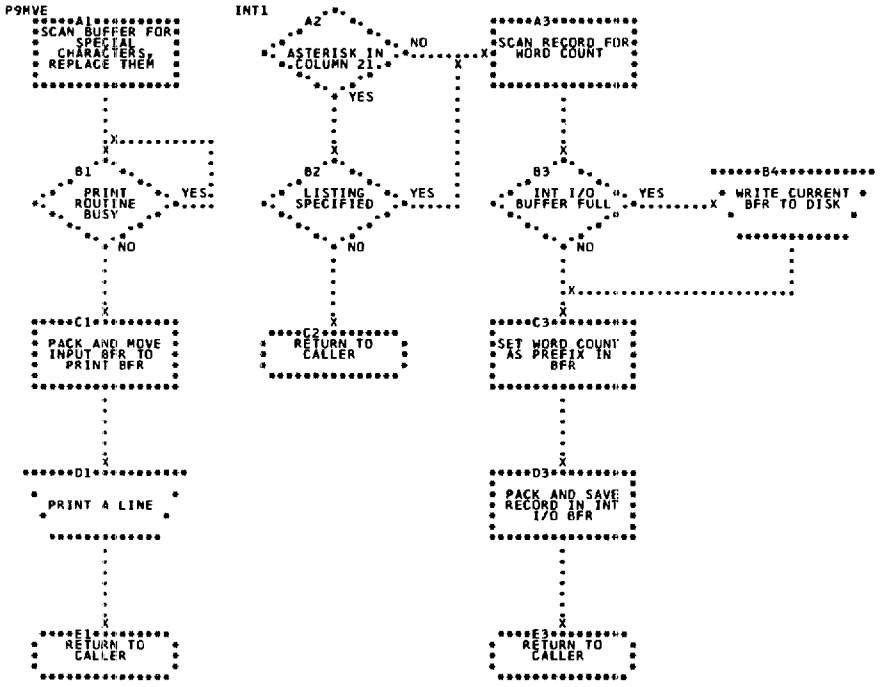
Flowchart ASM14. Assembler Program, Phase 8A



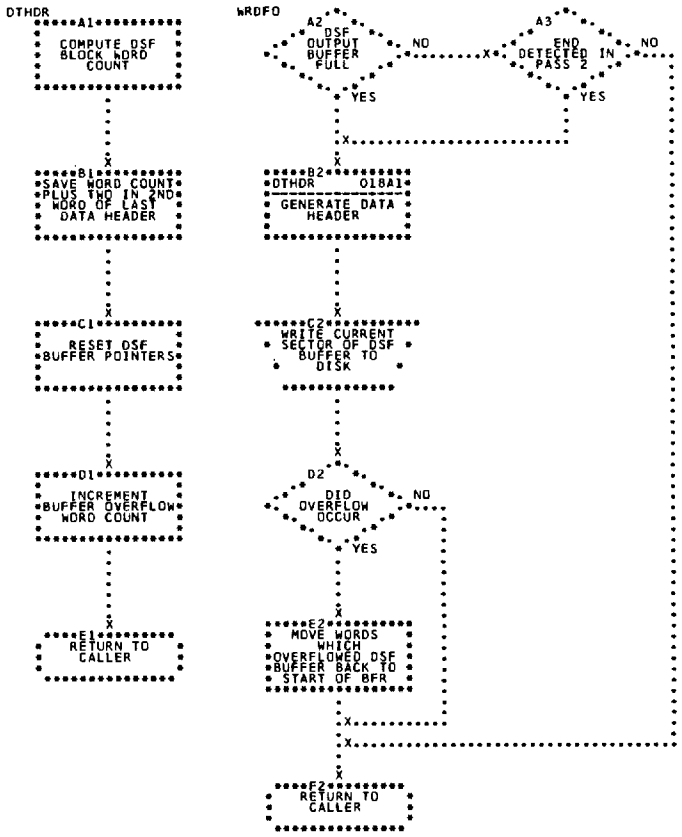
Flowchart ASM15, Assembler Program, Phase 9



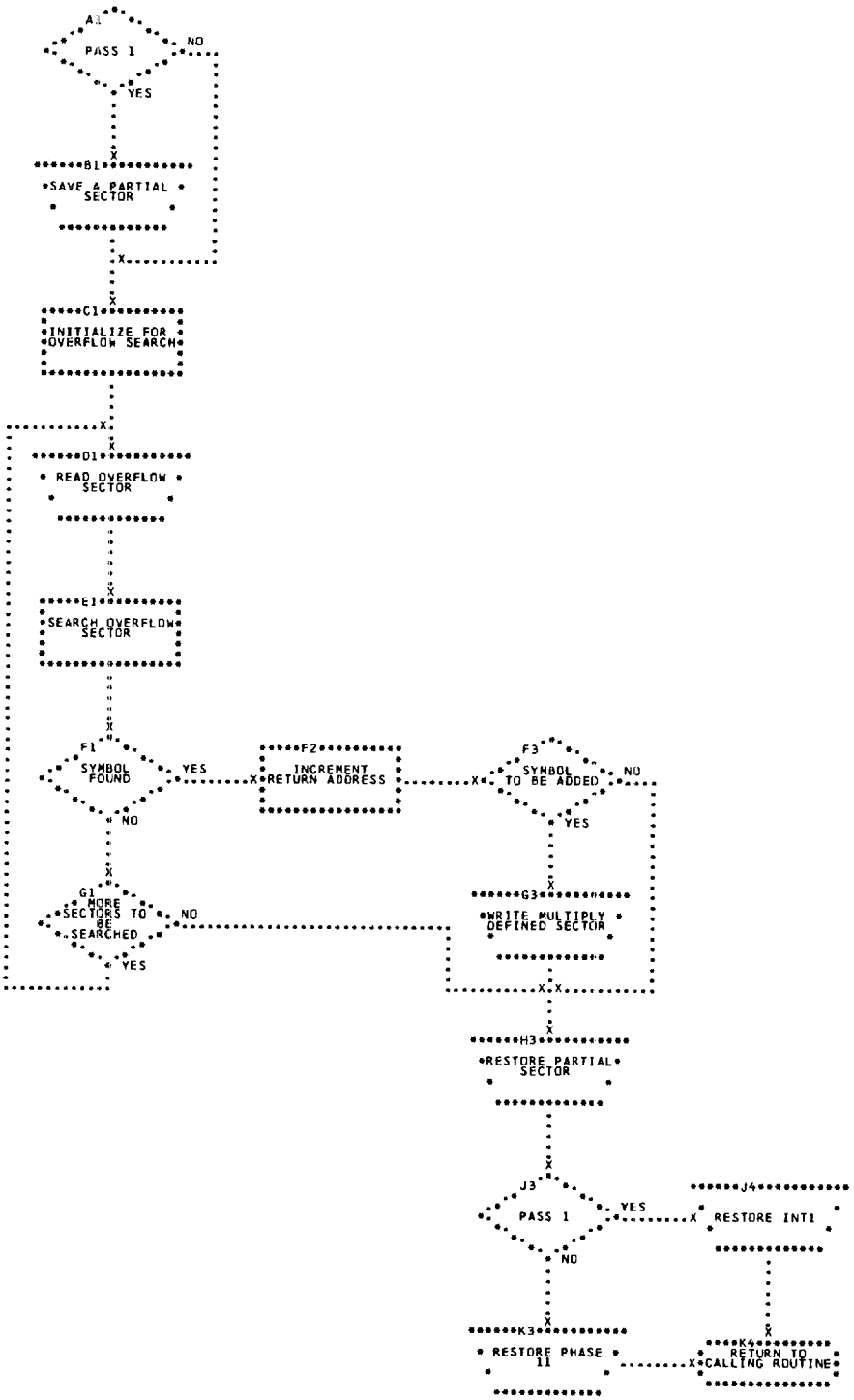
Flowchart ASM16. Assembler Program, Phase 9



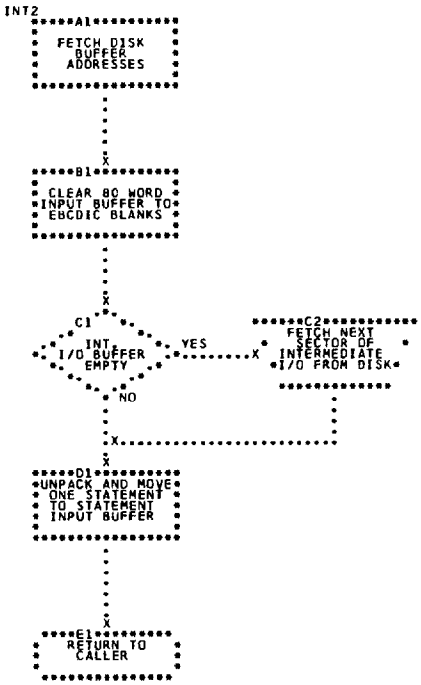
Flowchart ASM17. Assembler Program, Phase 9



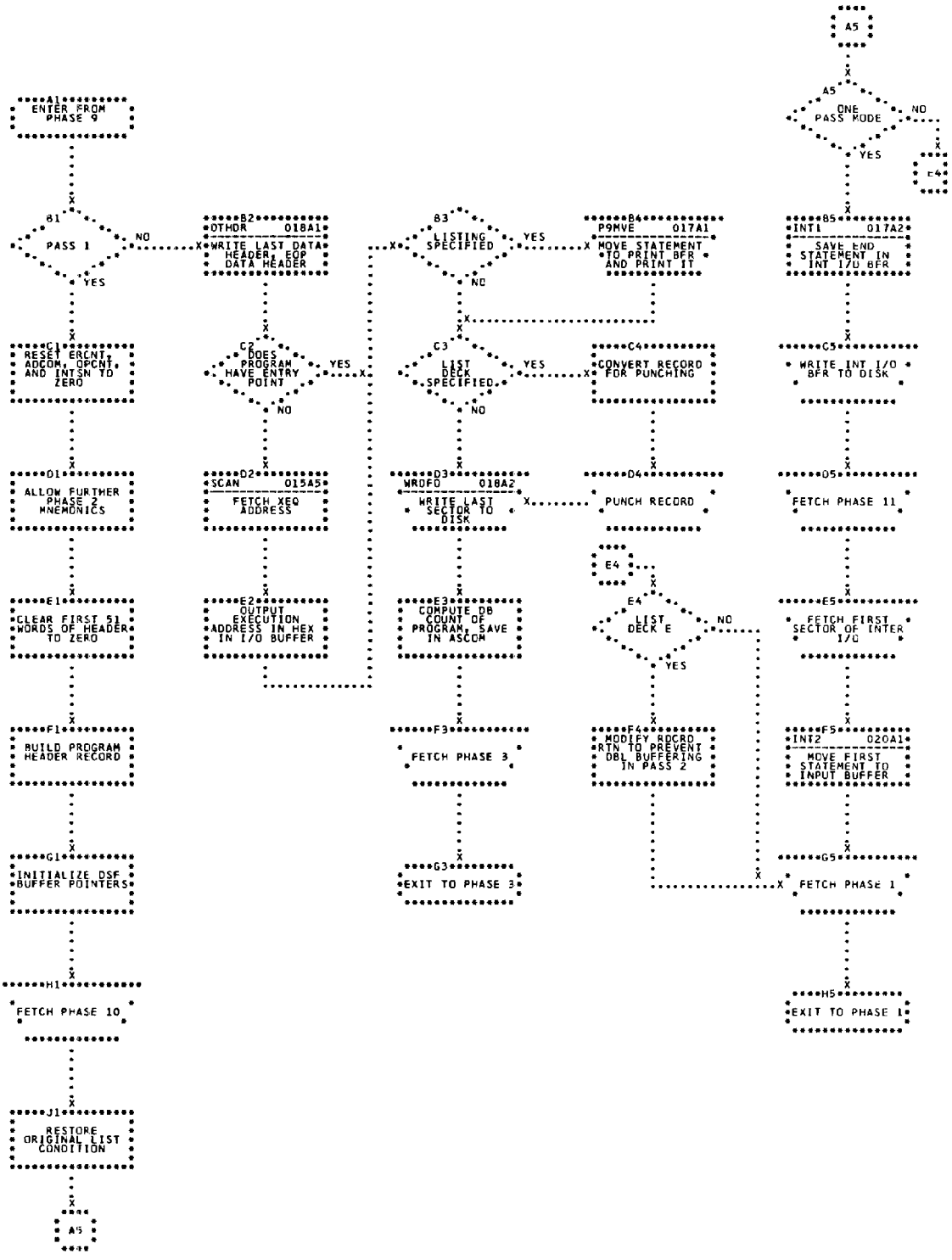
Flowchart ASM18. Assembler Program, Phase 10



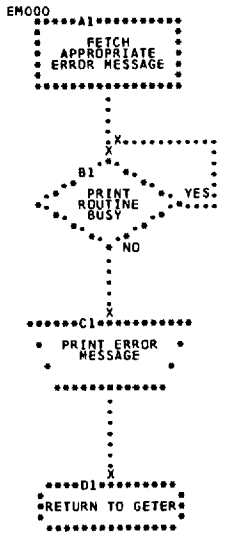
Flowchart ASM19. Assembler Program, Phase 10A



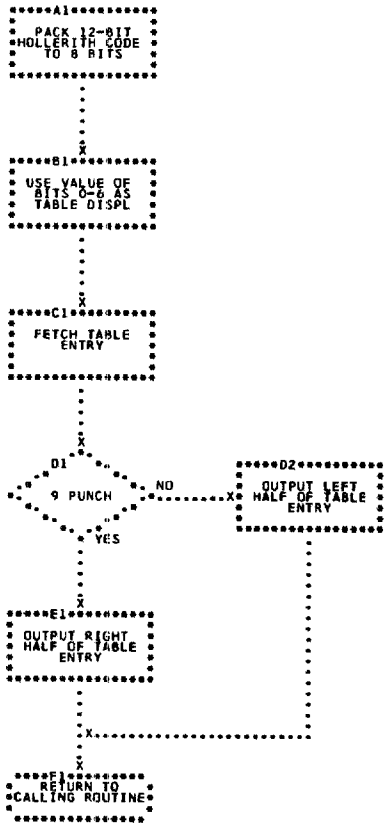
Flowchart ASM20. Assembler Program, Phase 11



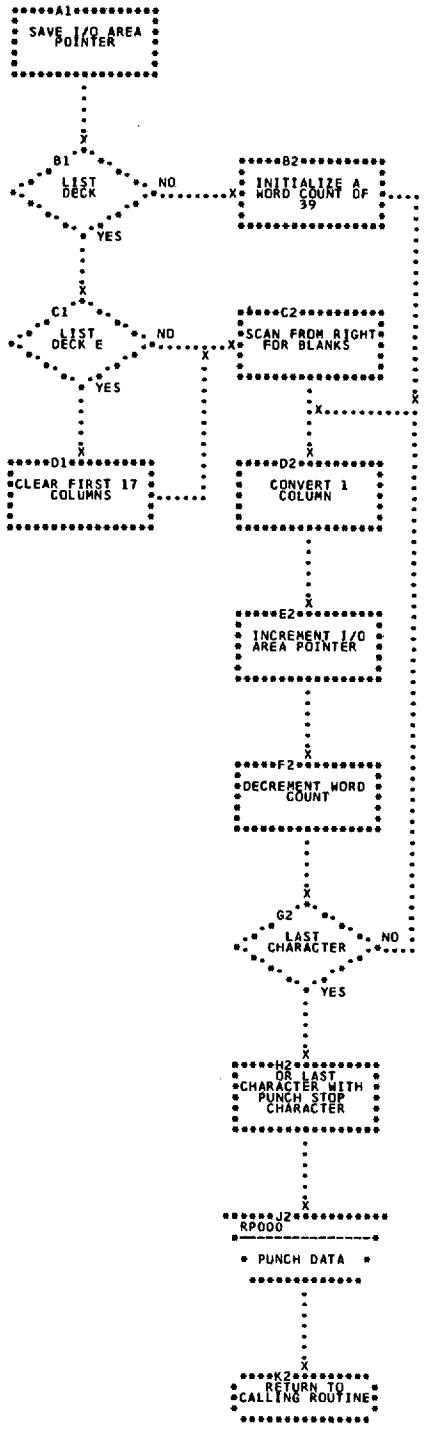
Flowchart ASM21. Assembler Program, Phase 12



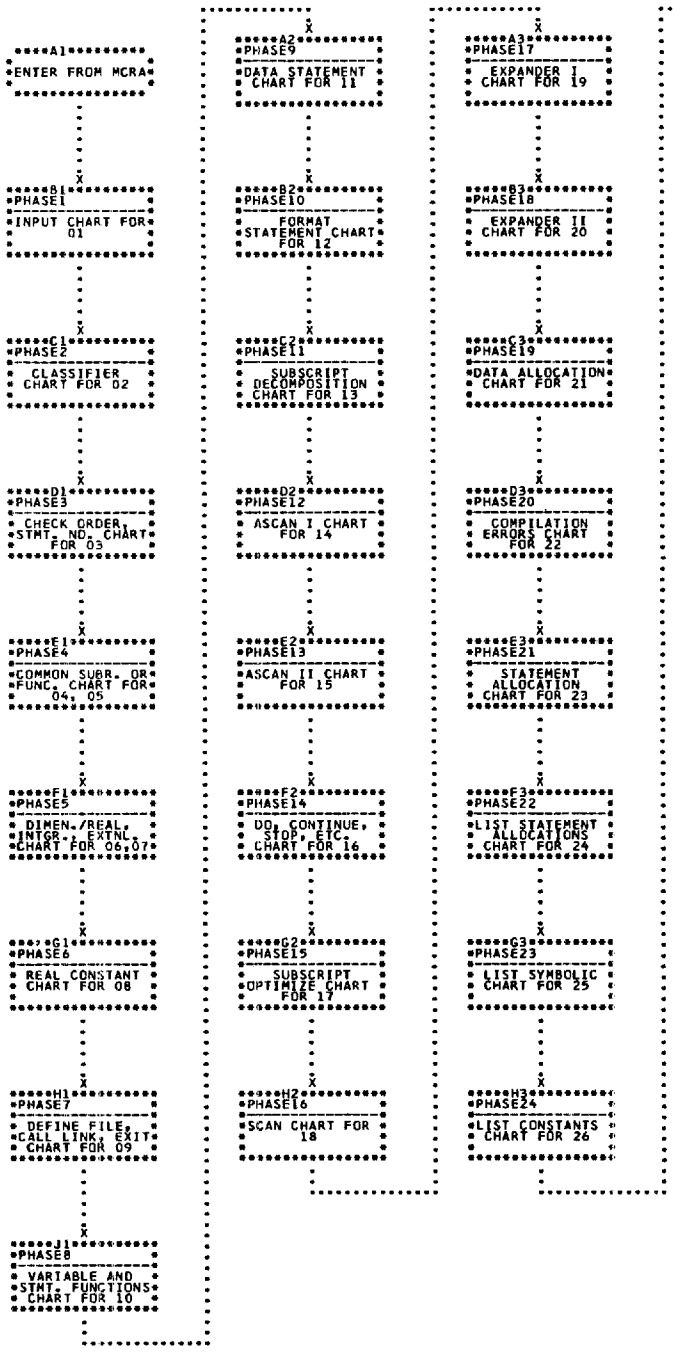
Flowchart ASM22. Assembler Program, Error Message Phase



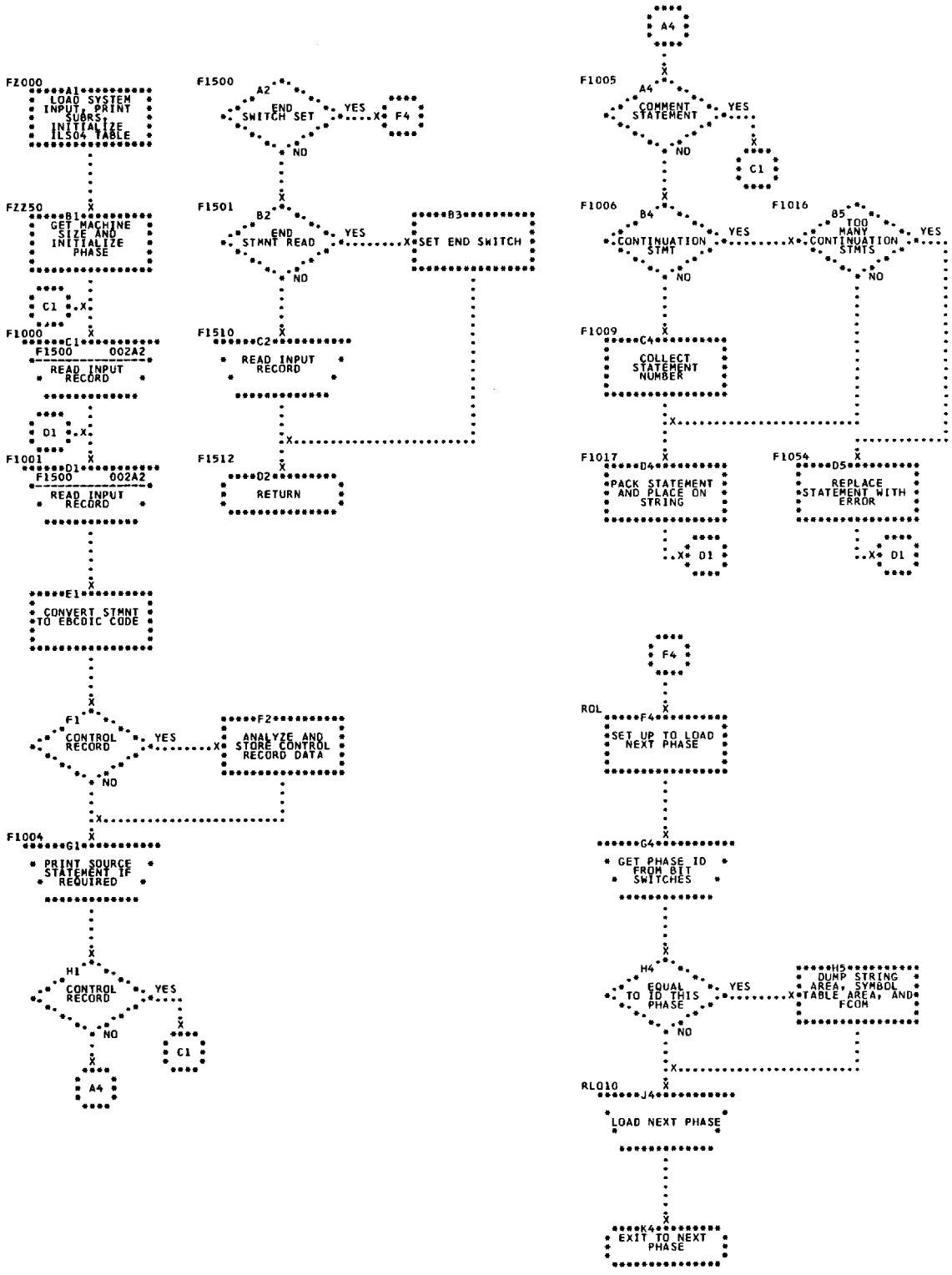
Flowchart ASM23. Assembler Program, Read Conversion Phase



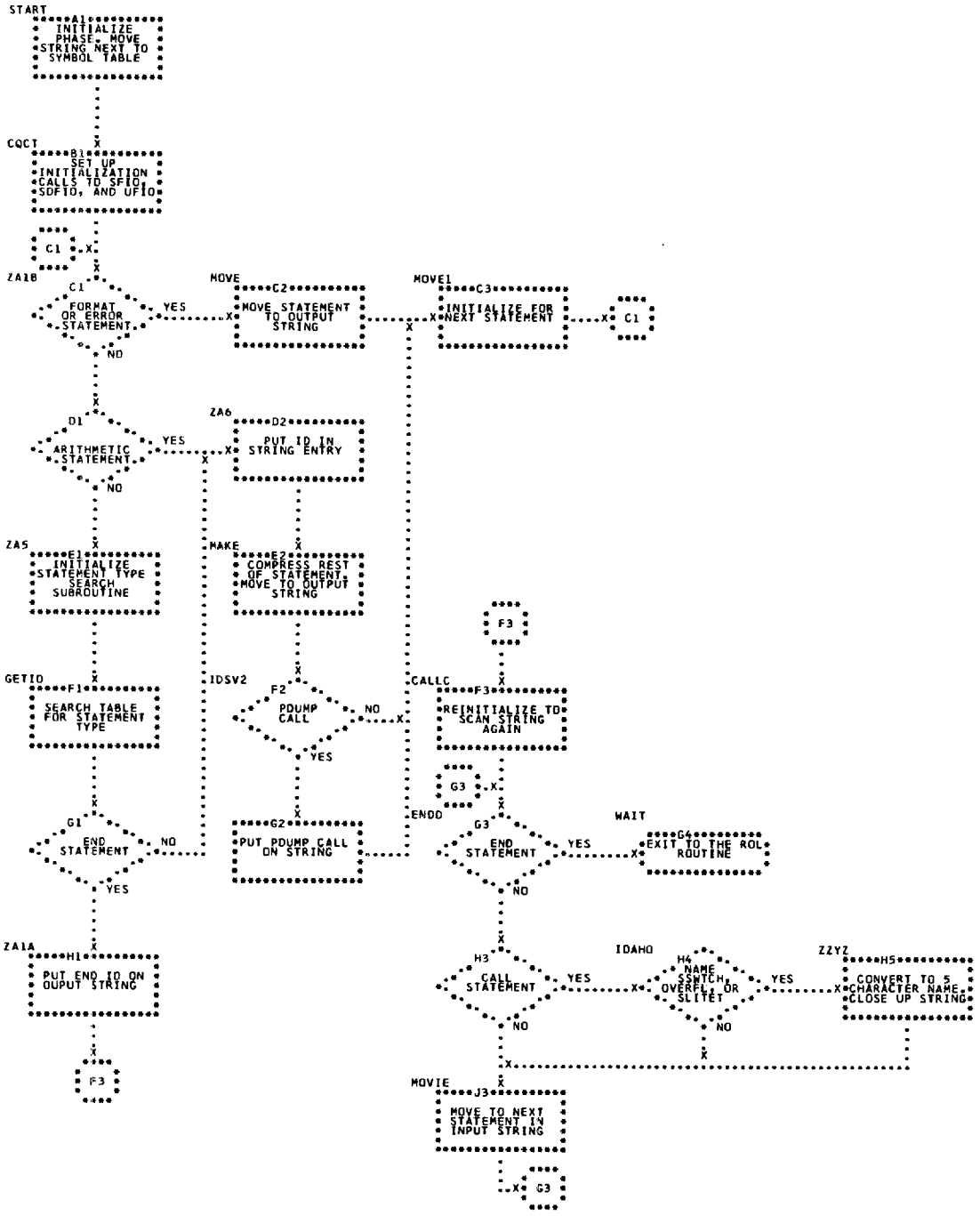
Flowchart ASM24. Assembler Program, Punch Conversion Phase



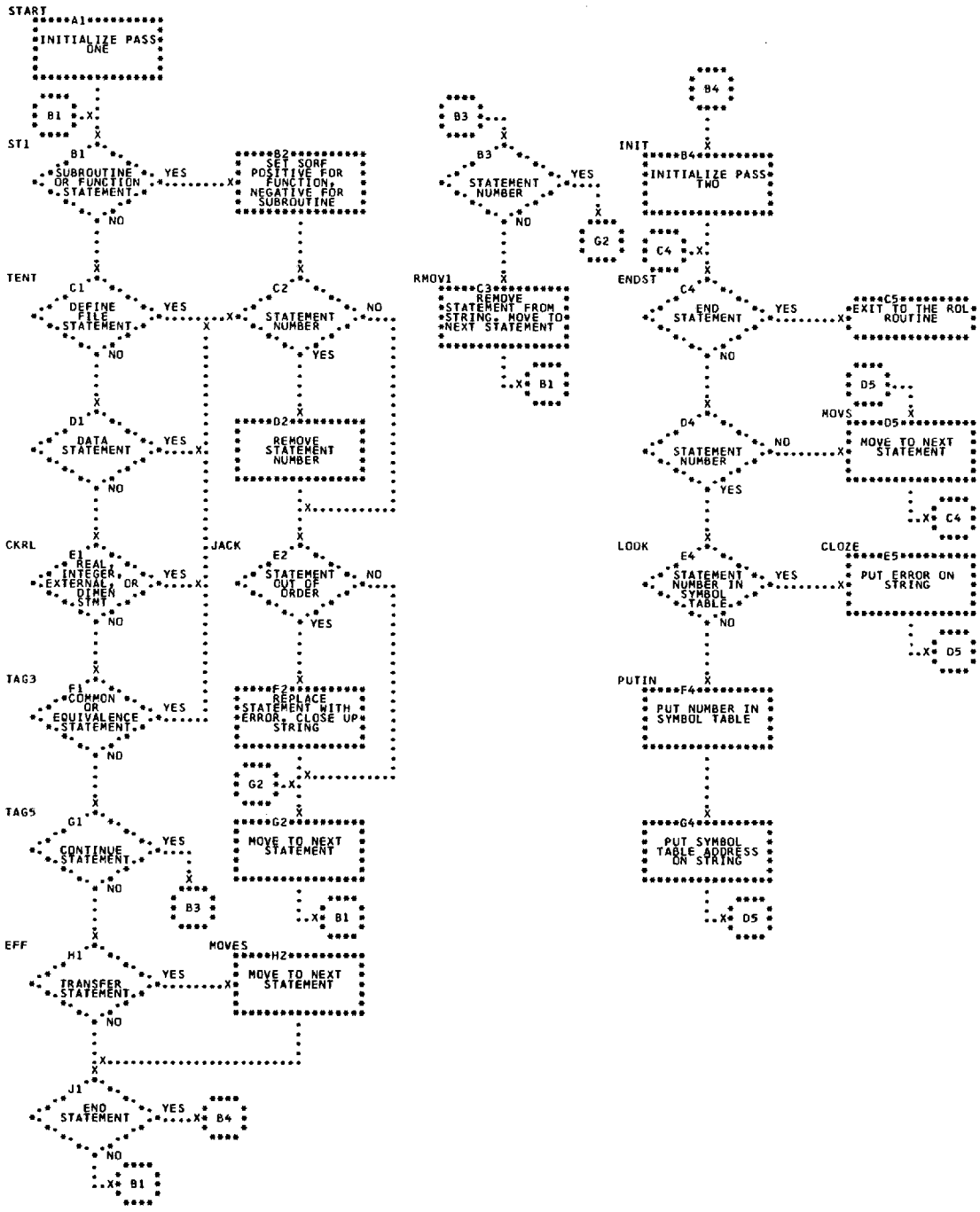
Flowchart FOR01. FORTRAN Compiler, General Flow



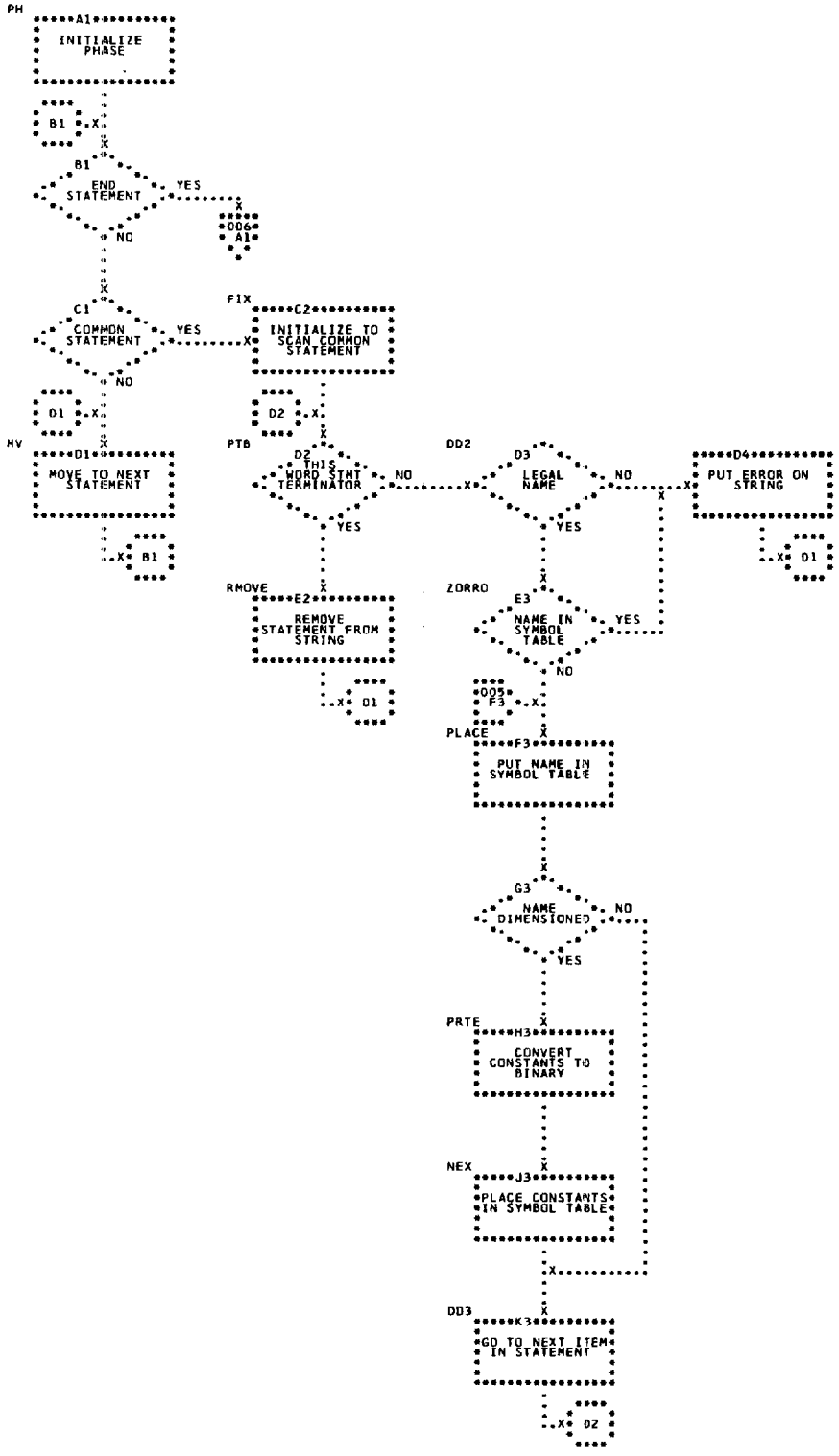
Flowchart FOR02. FORTRAN Compiler, Phase 1



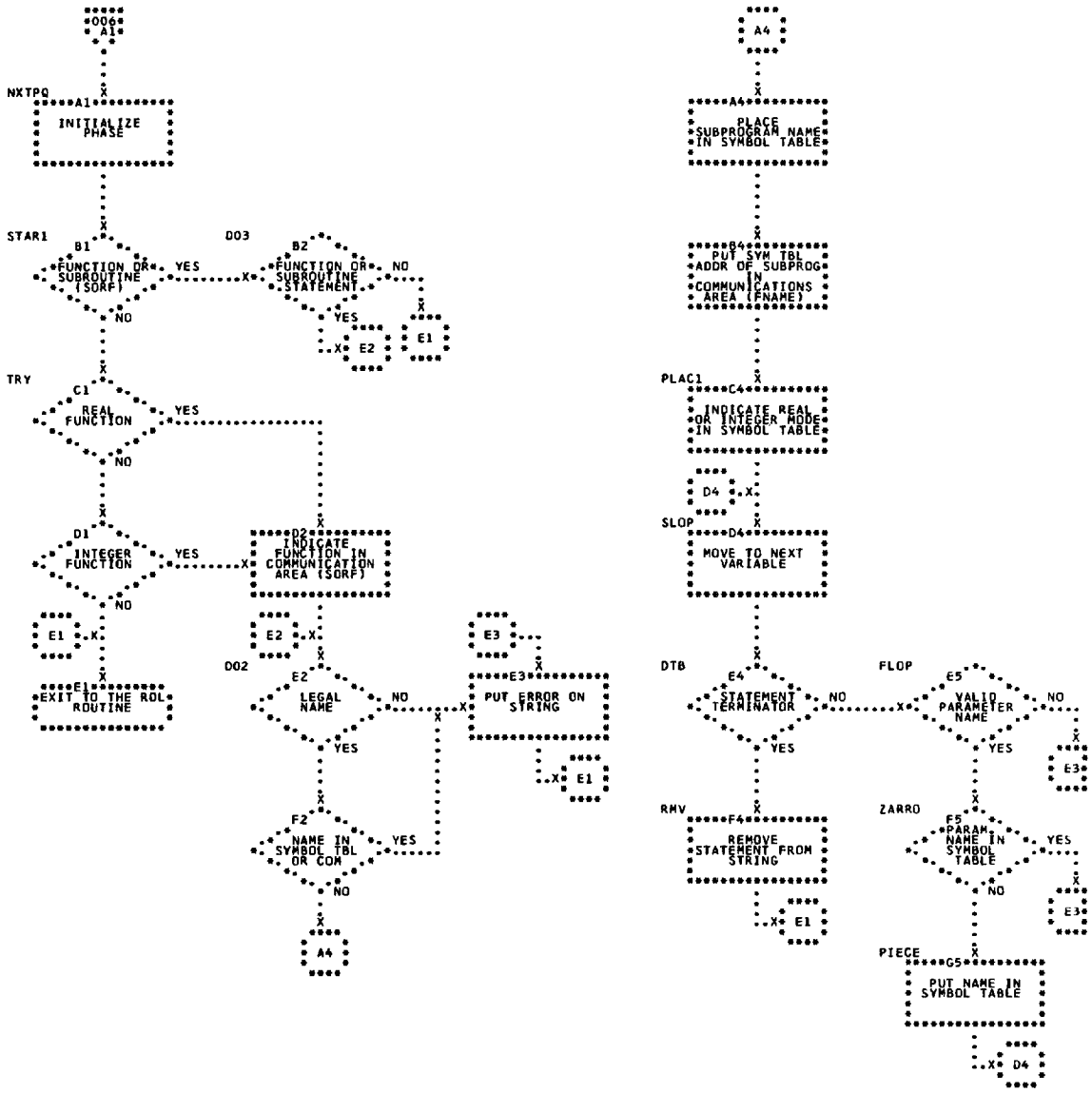
Flowchart FOR03. FORTRAN Compiler, Phase 2



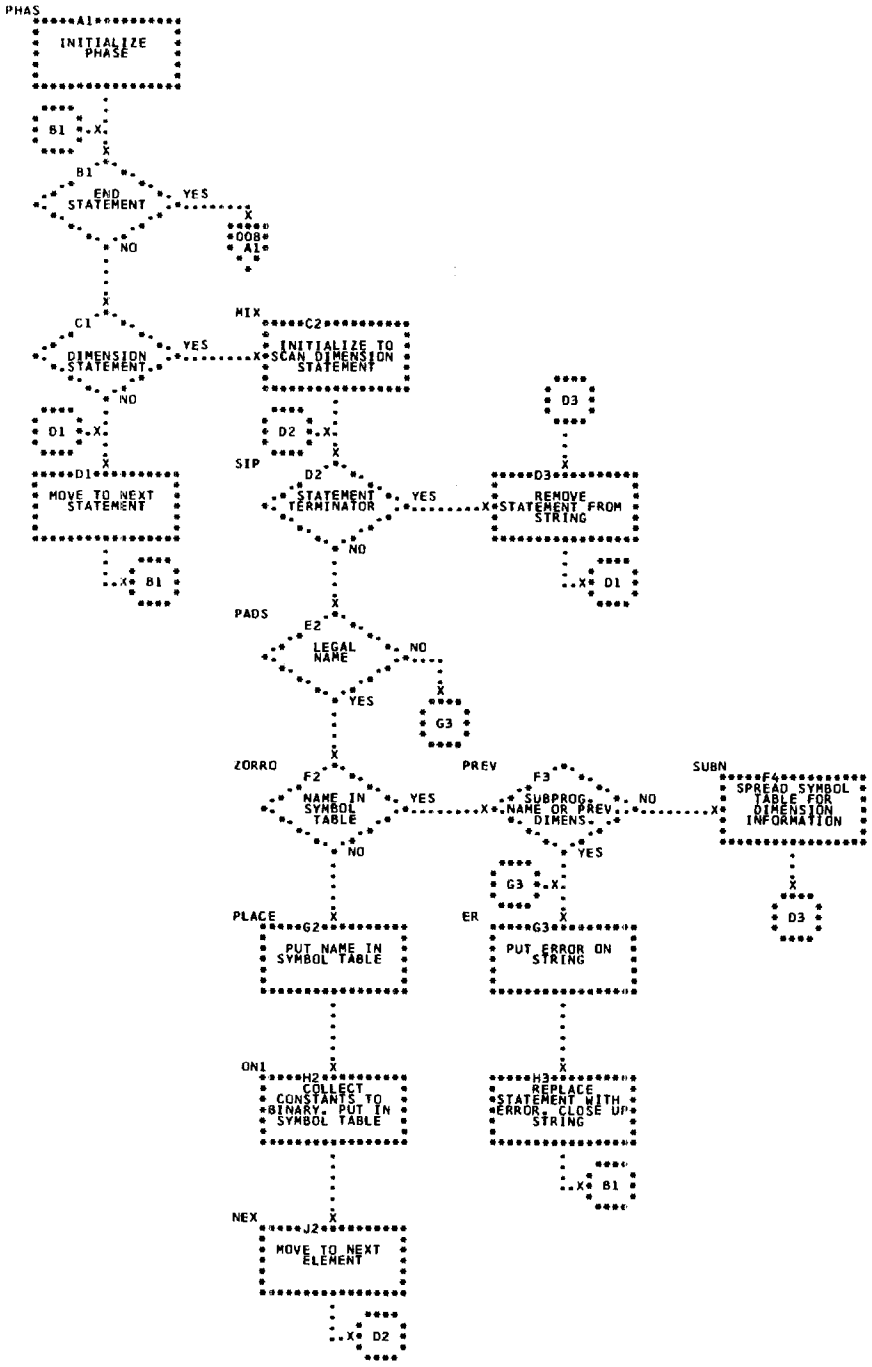
Flowchart FOR04. FORTRAN Compiler, Phase 3



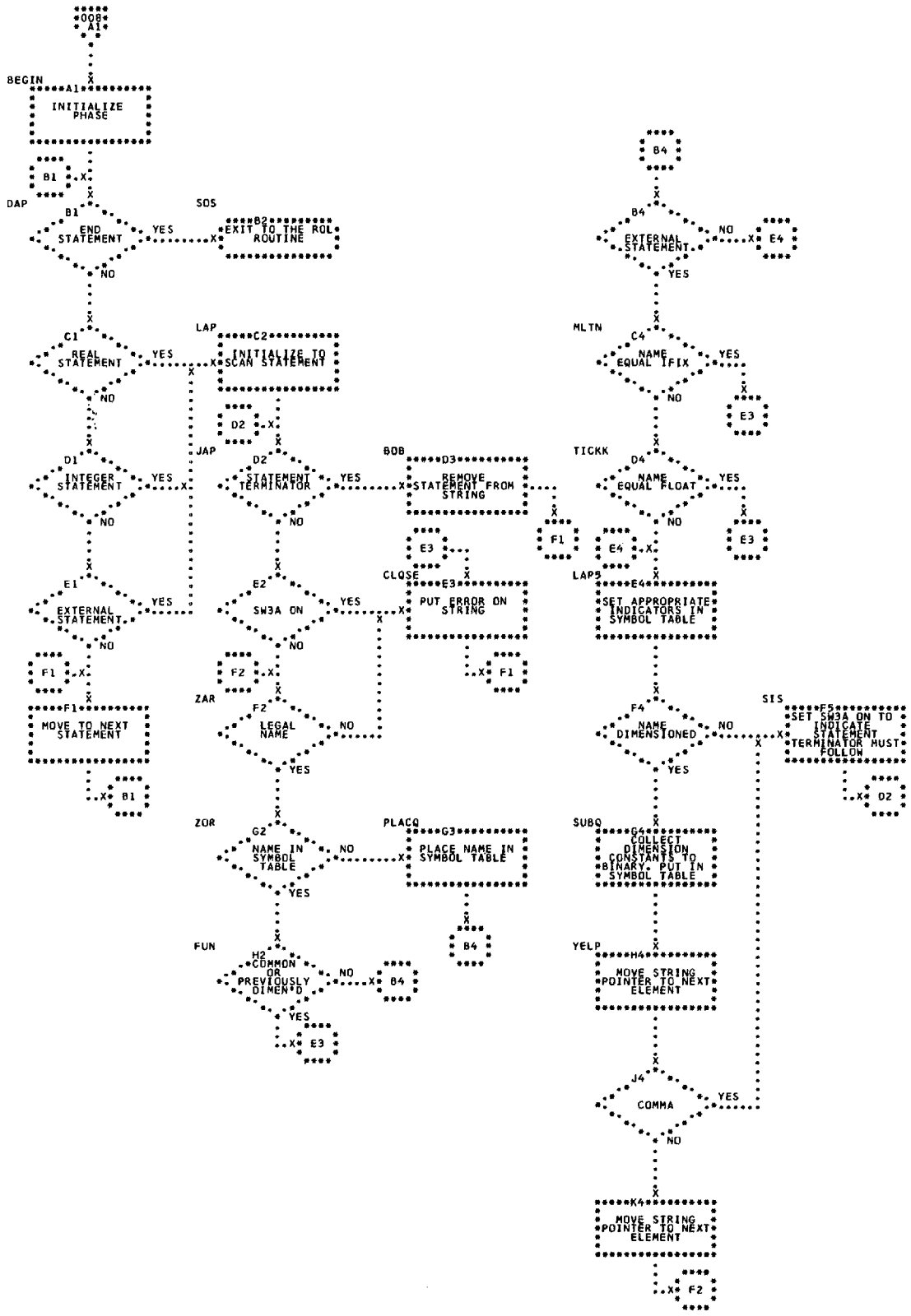
Flowchart FOR05. FORTRAN Compiler, Phase 4



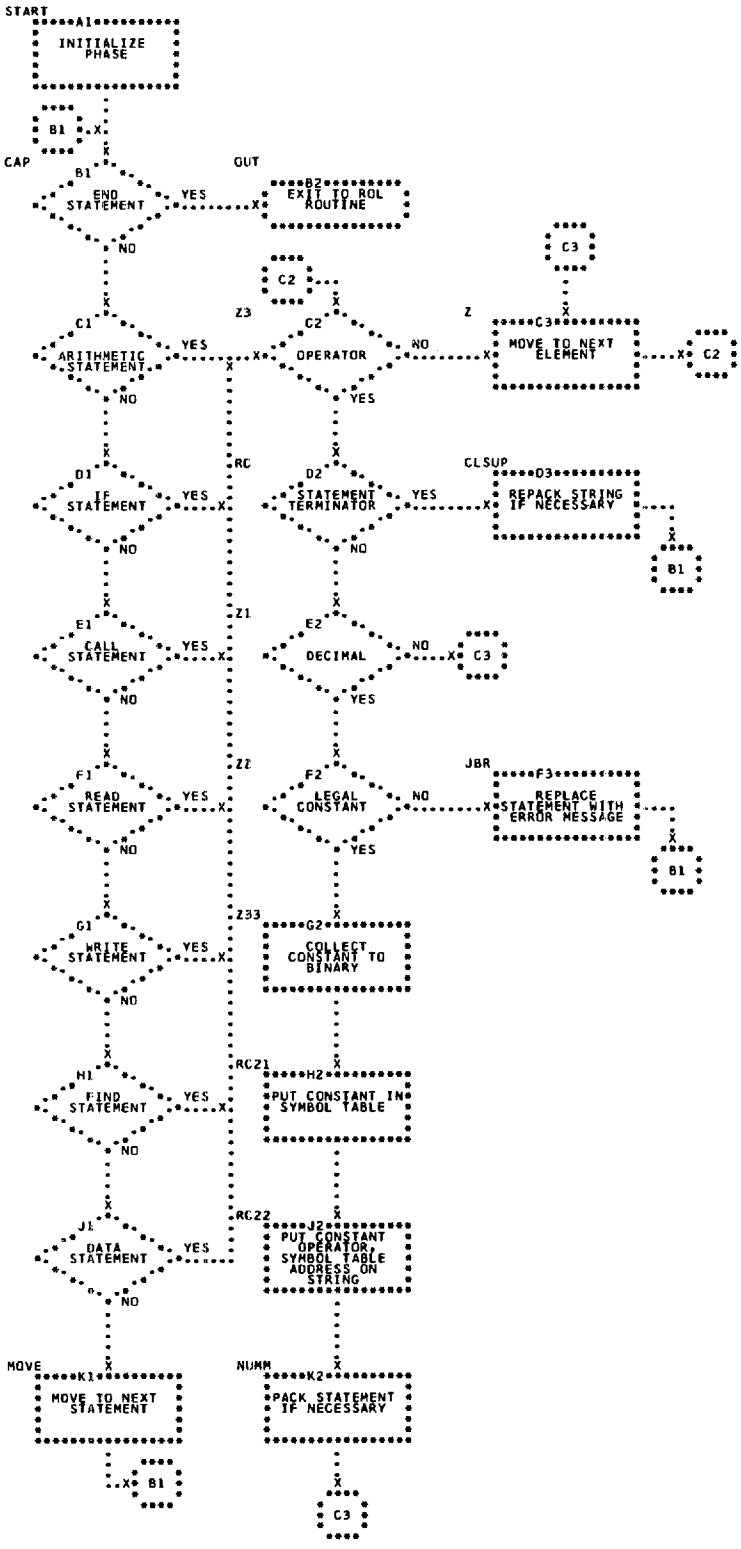
Flowchart FOR06. FORTRAN Compiler, Phase 4



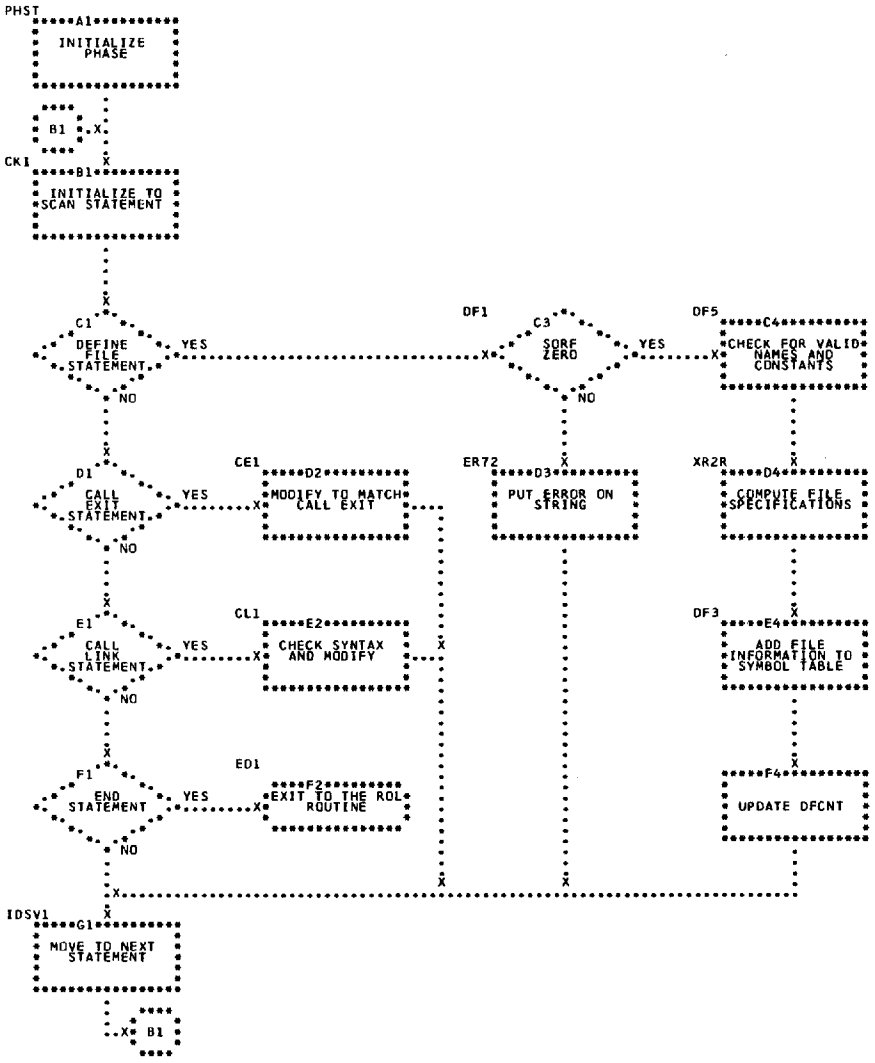
Flowchart FOR07. FORTRAN Compiler, Phase 5



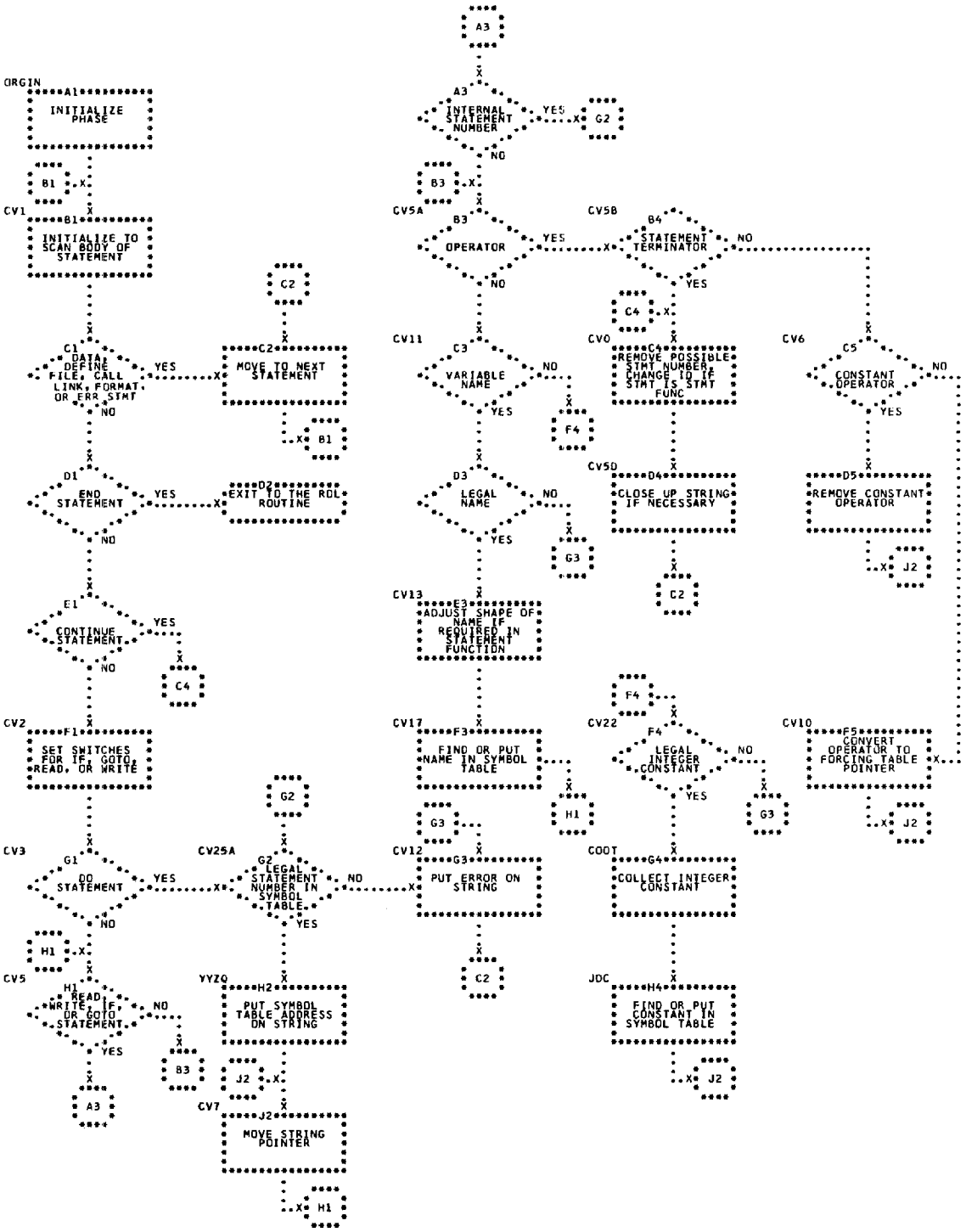
Flowchart FOR08. FORTRAN Compiler, Phase 5



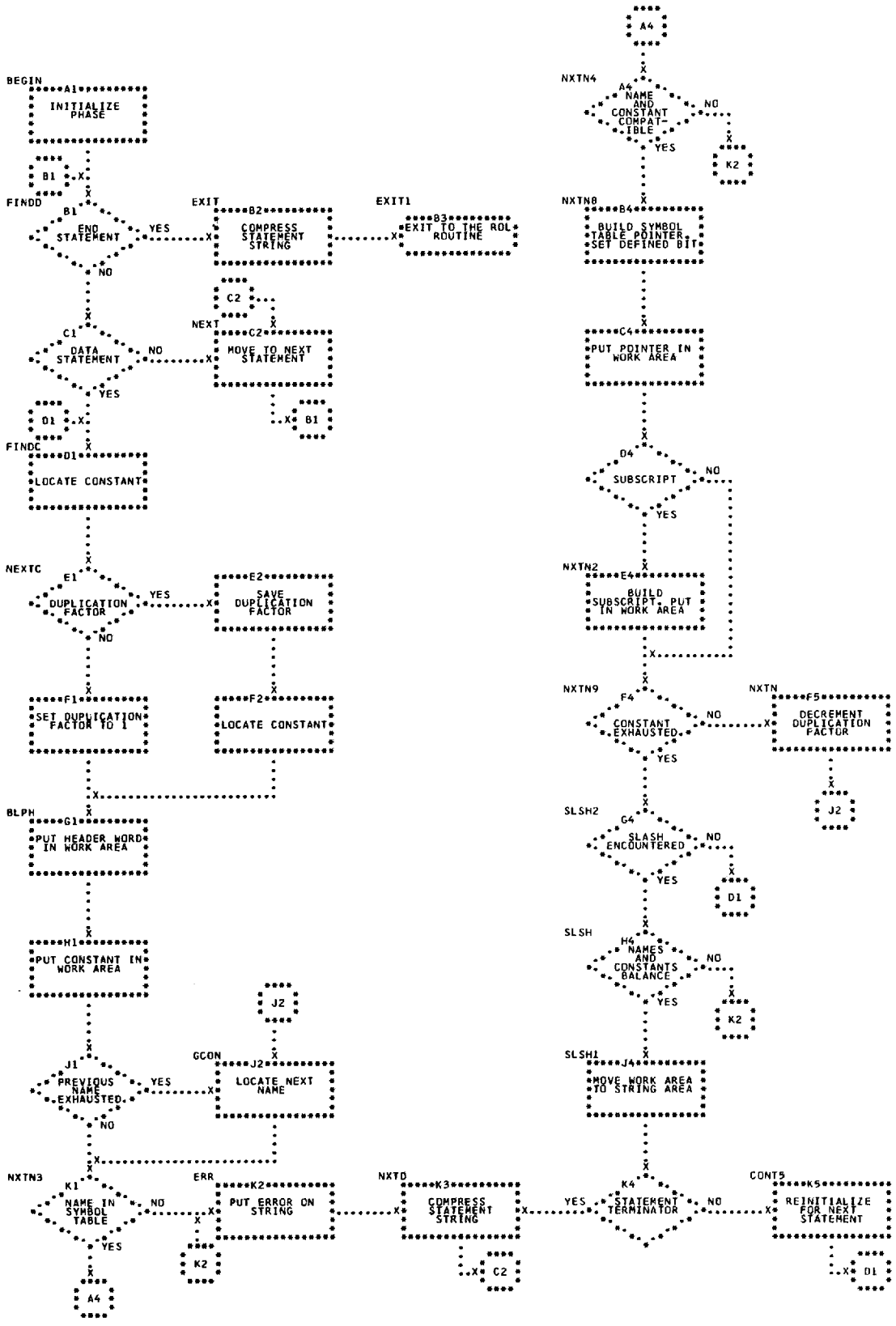
Flowchart FOR09. FORTRAN Compiler, Phase 6



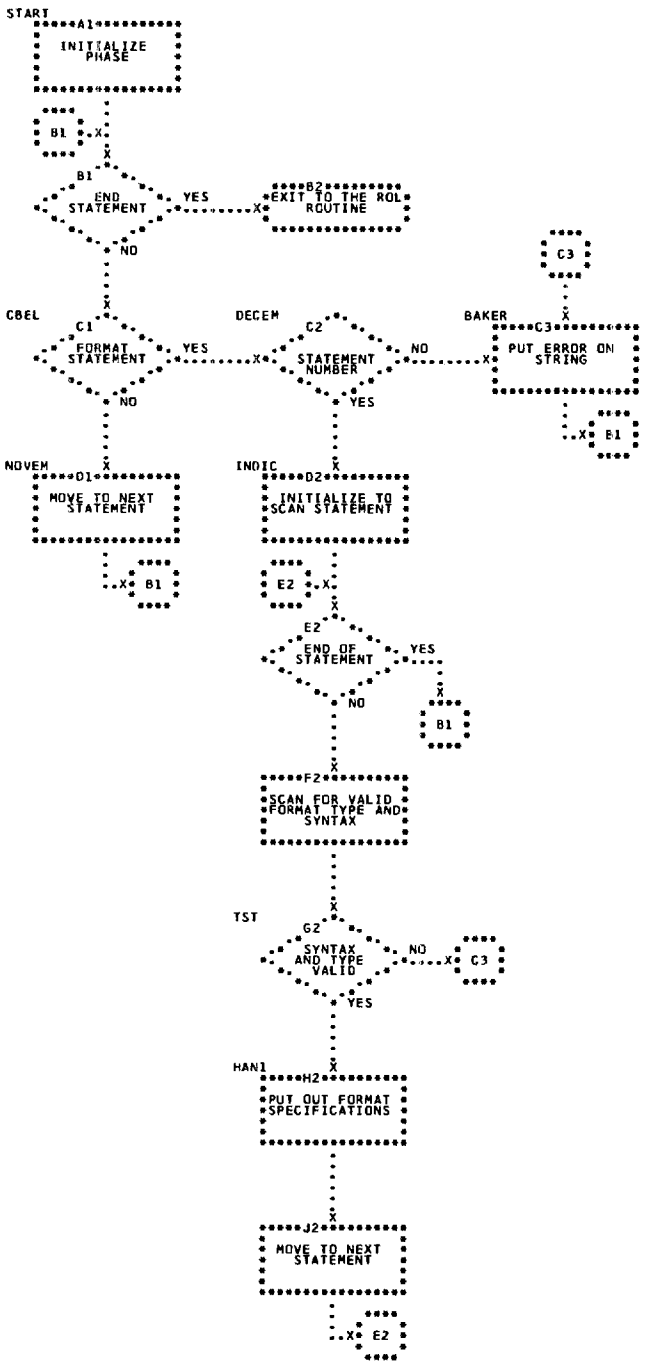
Flowchart FOR10. FORTRAN Compiler, Phase 7



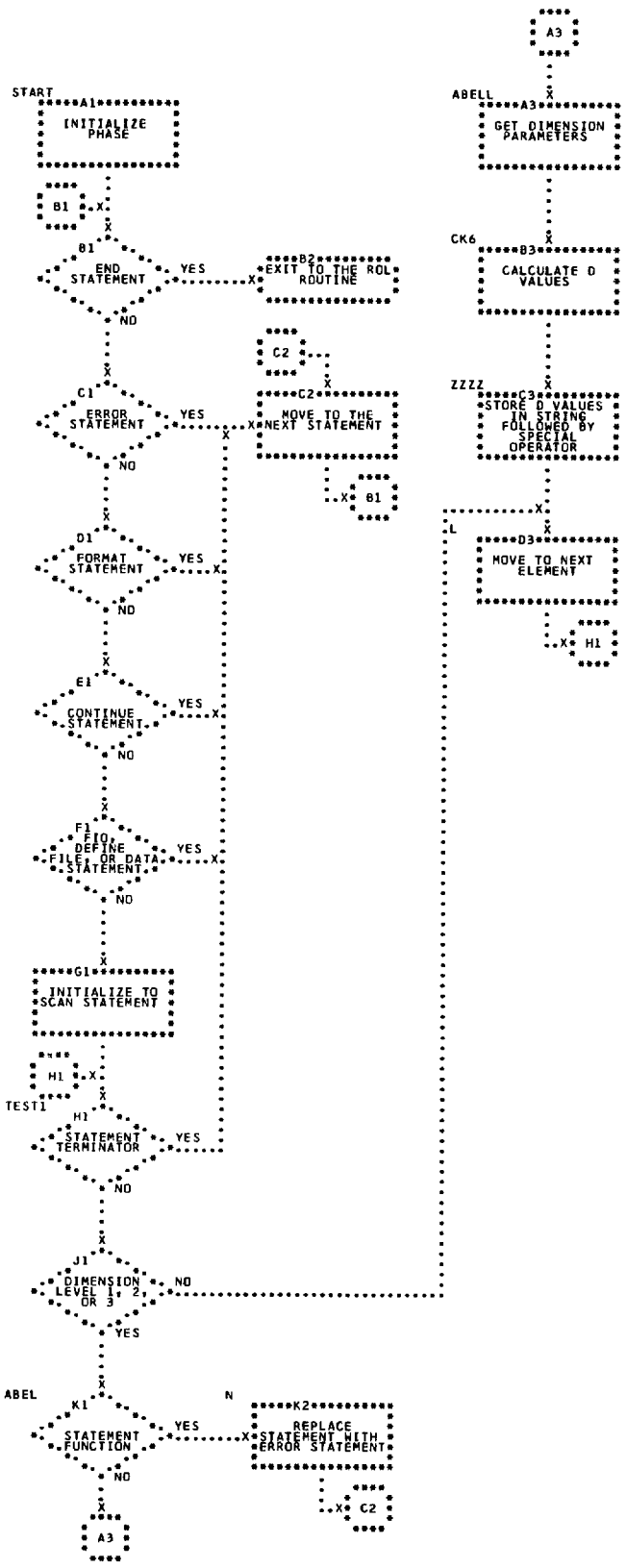
Flowchart FOR11. FORTRAN Compiler, Phase 8



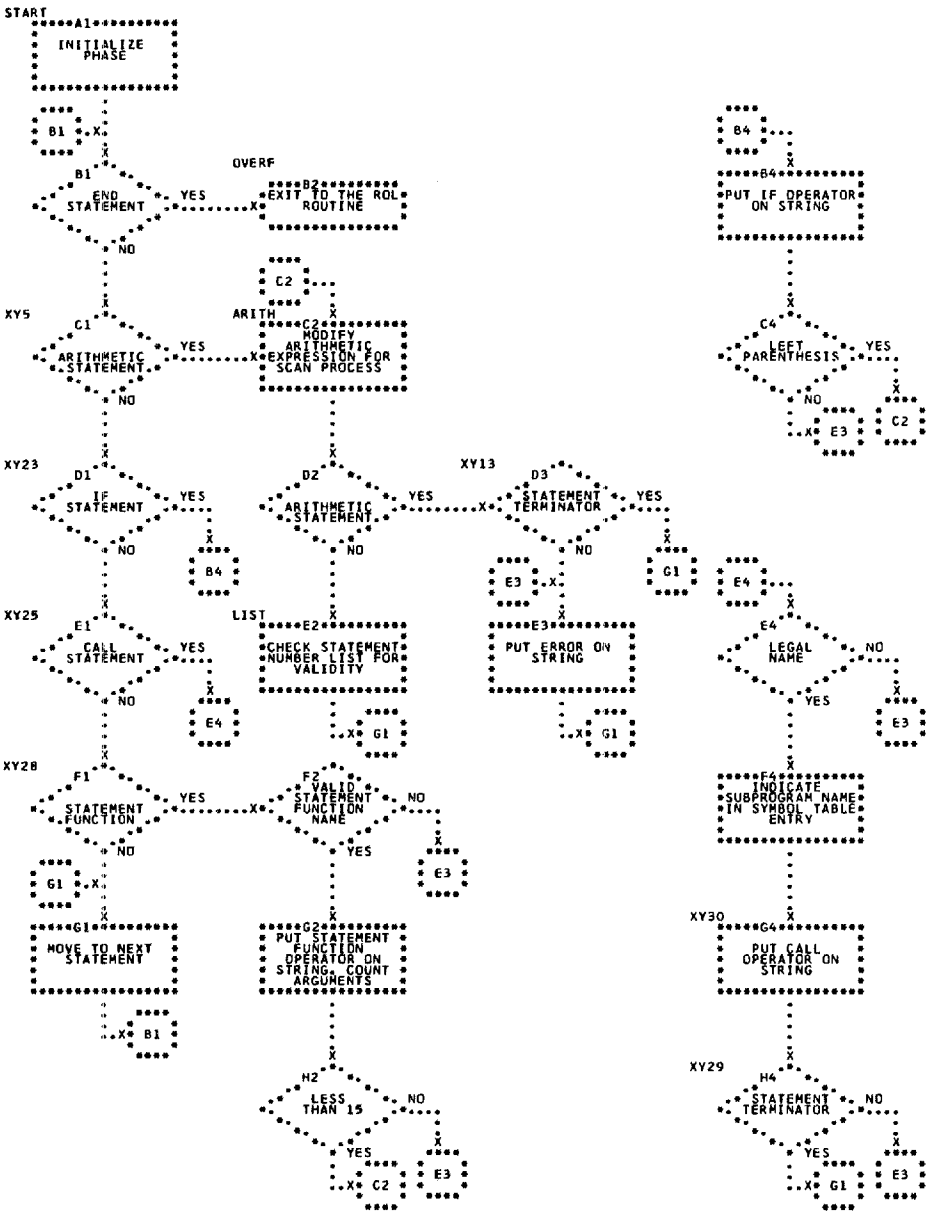
Flowchart FOR12. FORTRAN Compiler, Phase 9



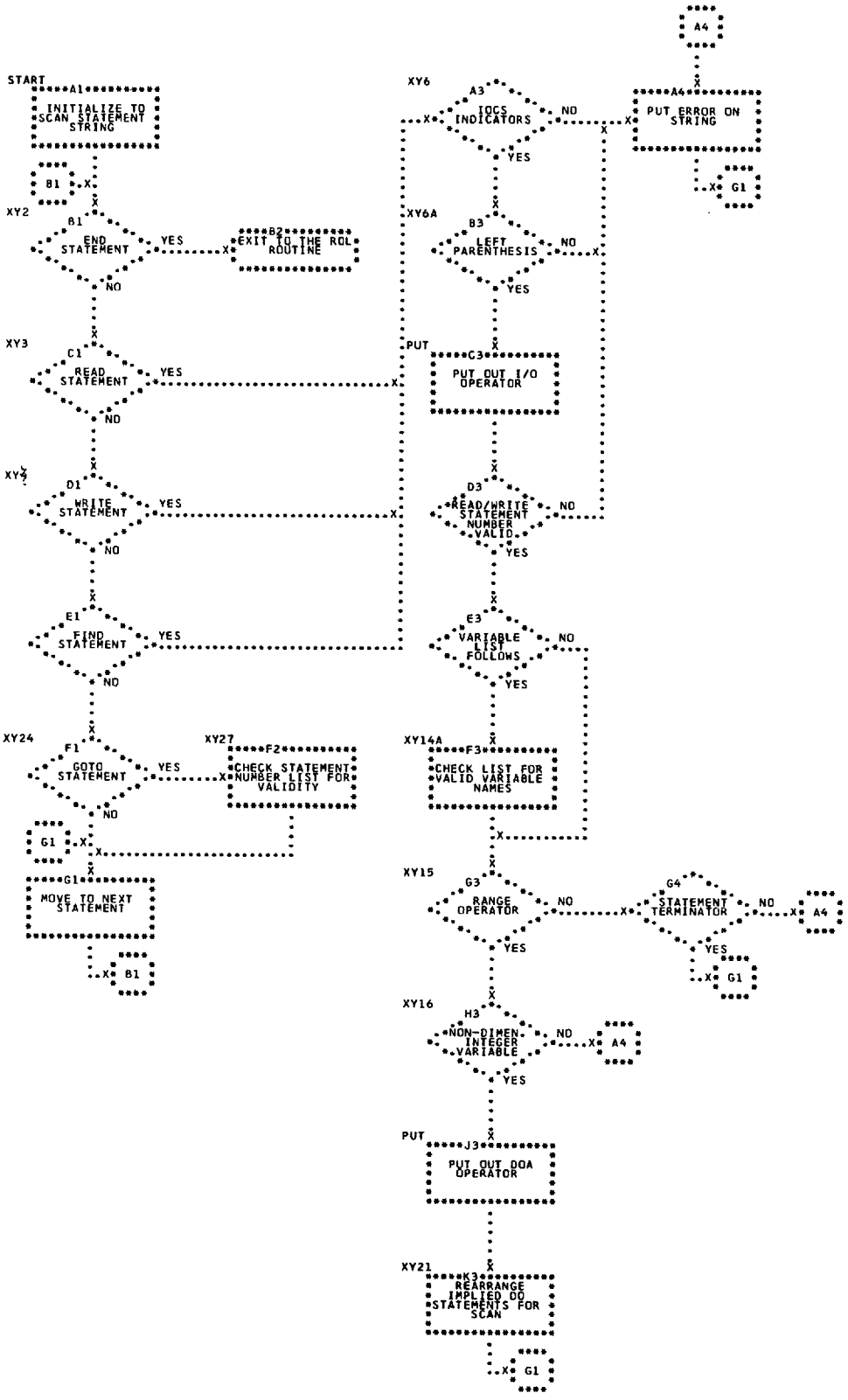
Flowchart FOR13. FORTRAN Compiler, Phase 10



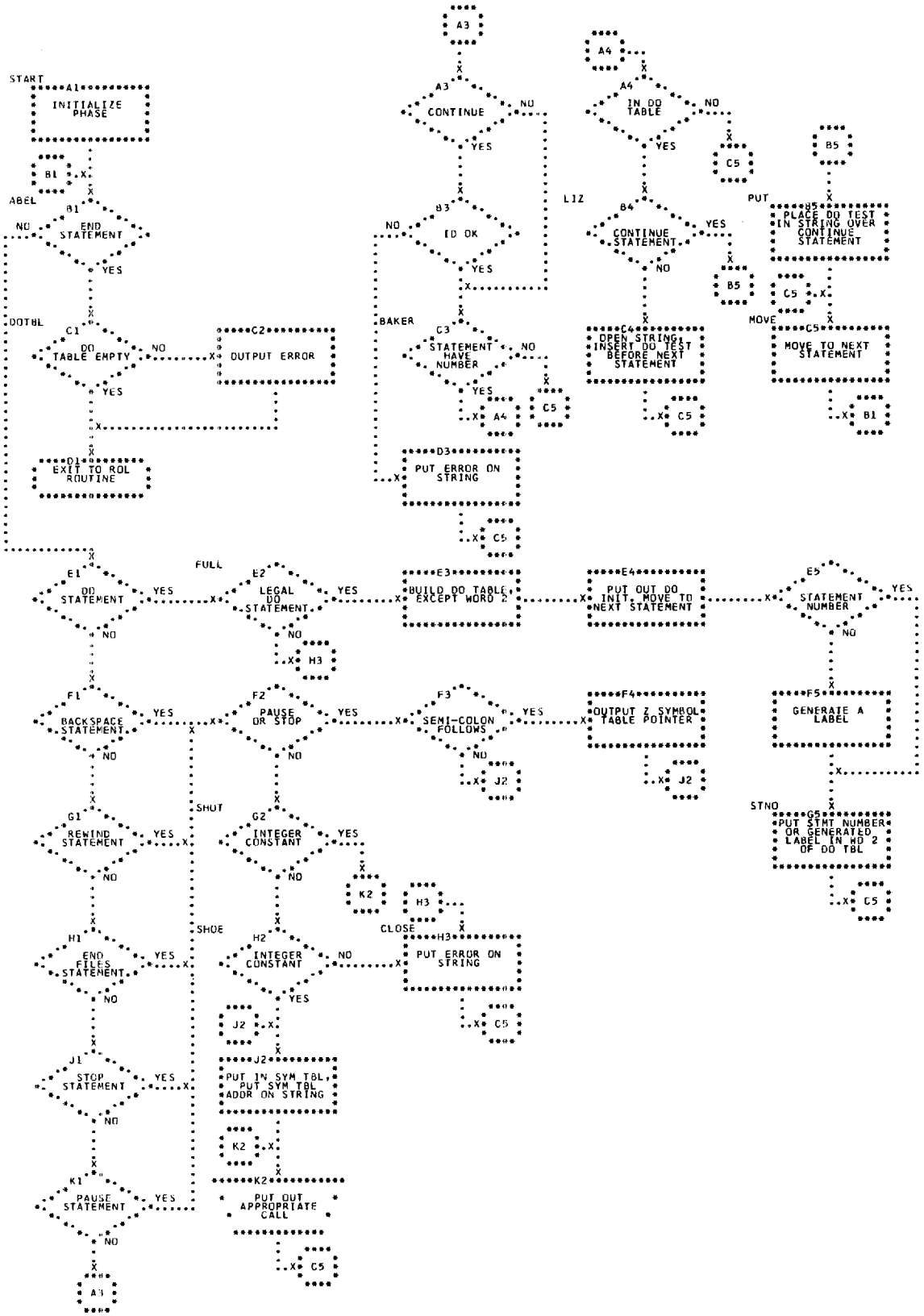
Flowchart FOR14. FORTRAN Compiler, Phase 11



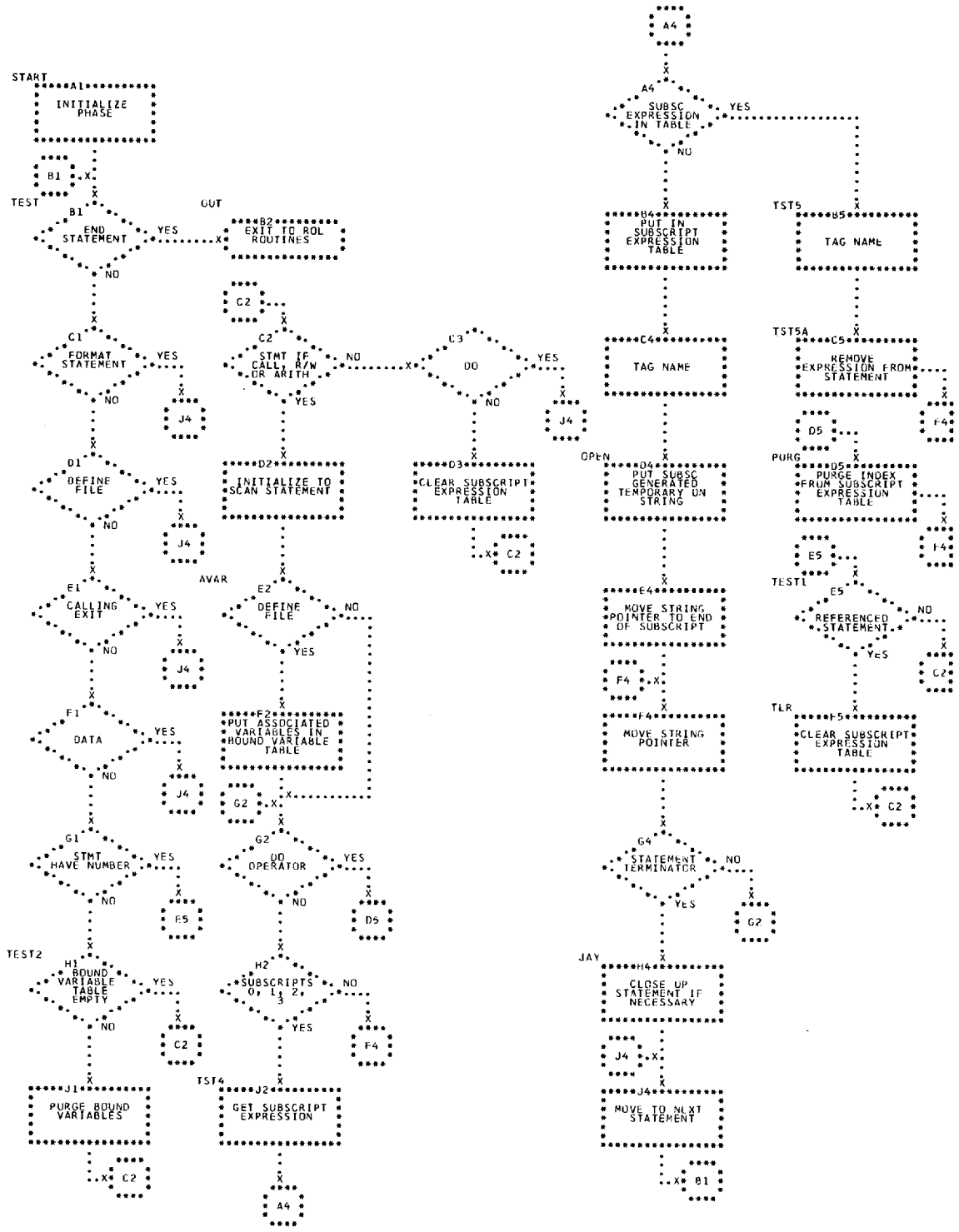
Flowchart FOR15. FORTRAN Compiler, Phase 12



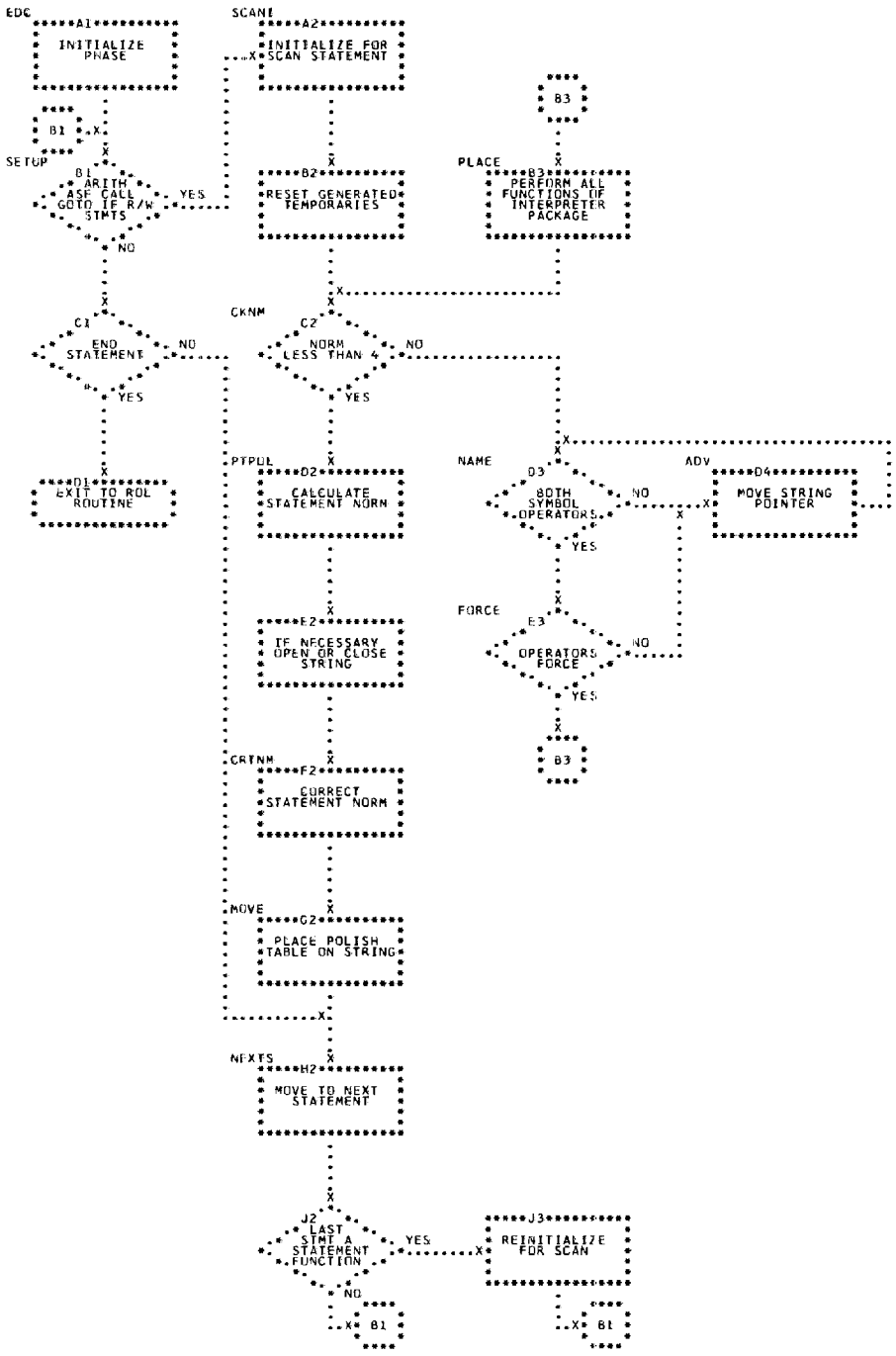
Flowchart FOR16. FORTRAN Compiler, Phase 13



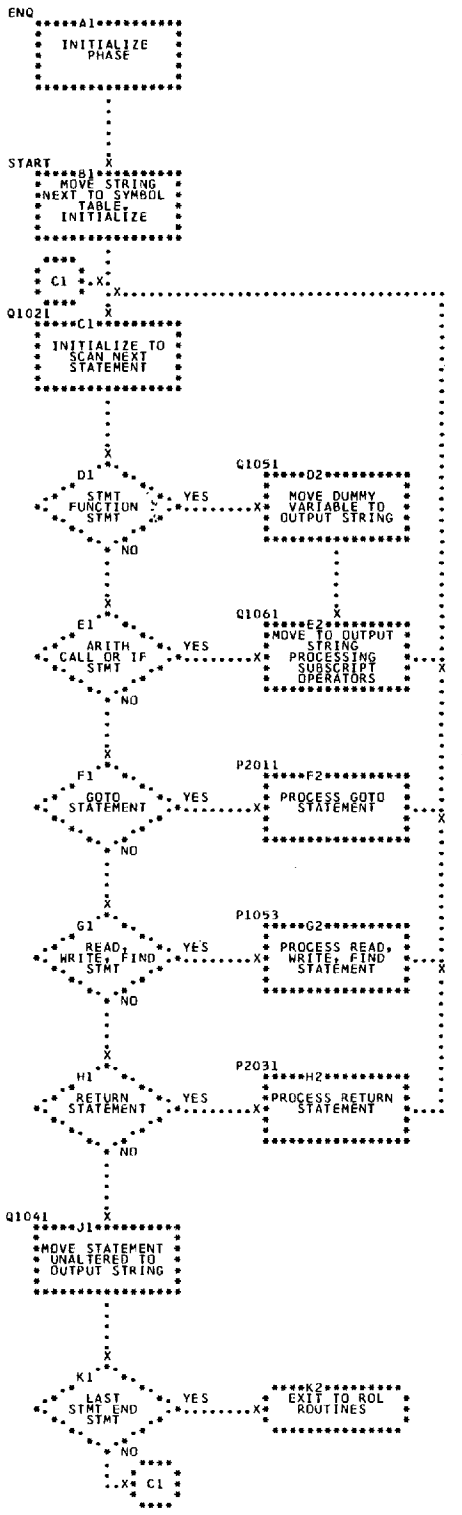
Flowchart FOR17. FORTRAN Compiler, Phase 14



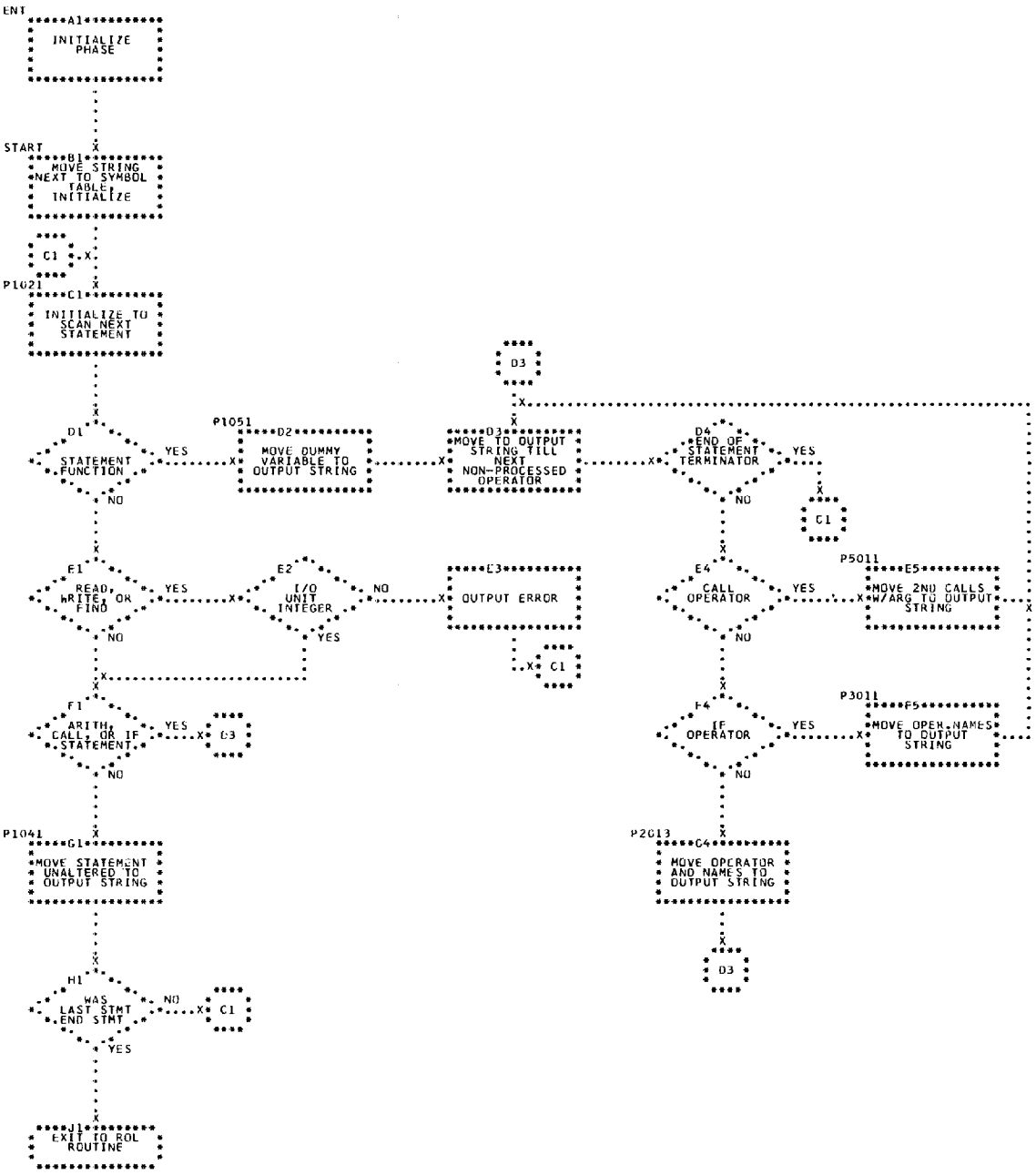
Flowchart FOR18. FORTRAN Compiler, Phase 15



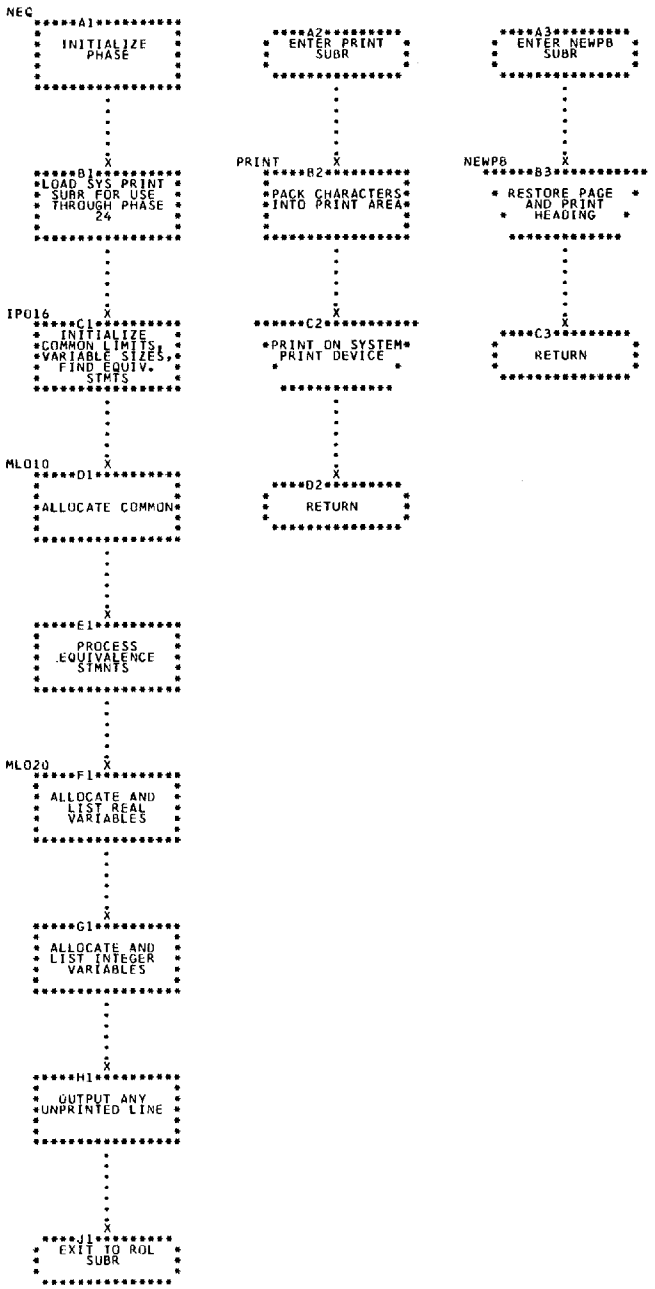
Flowchart FOR19. FORTRAN Compiler, Phase 16



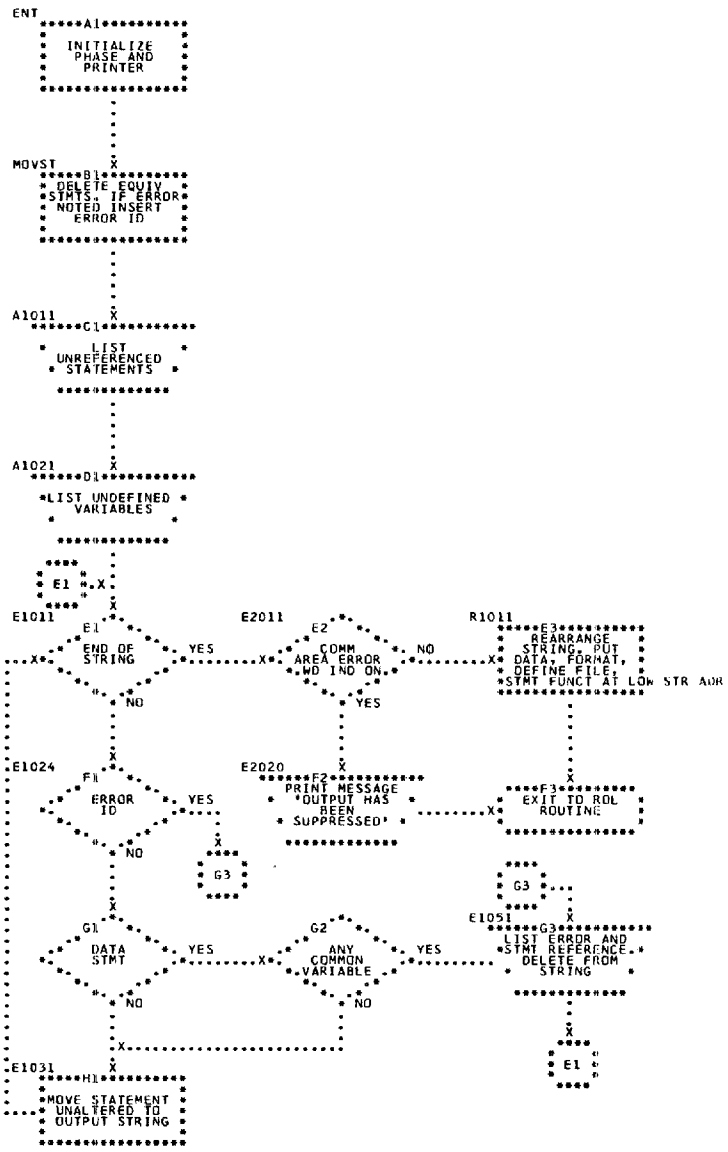
Flowchart FOR20. FORTRAN Compiler, Phase 17



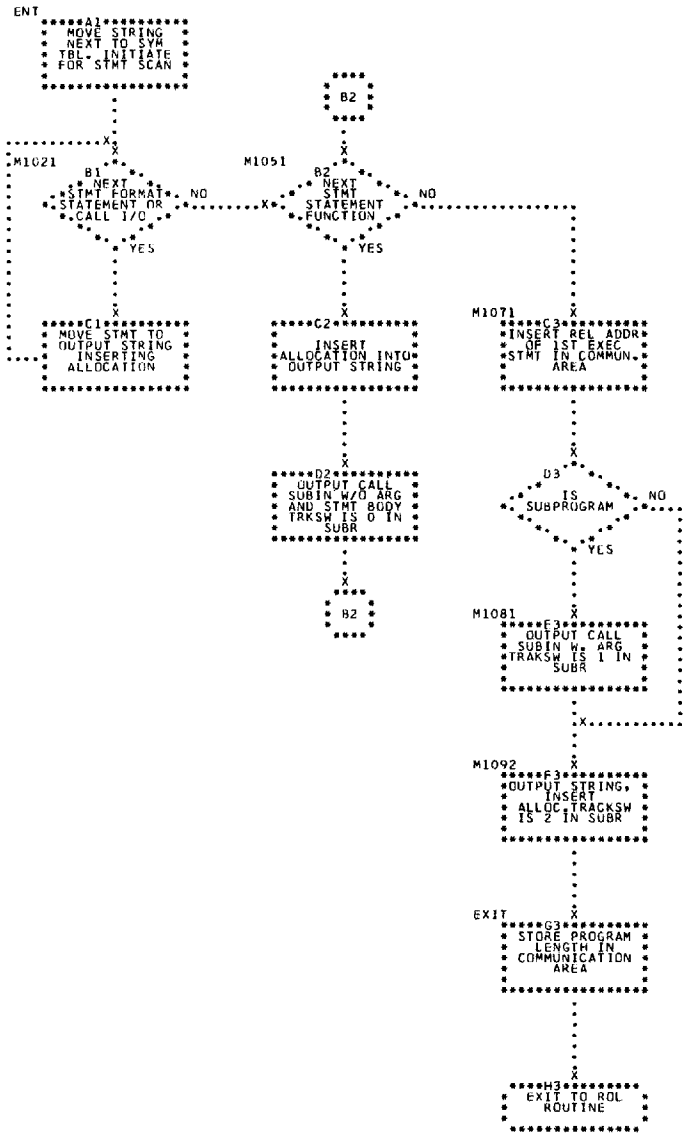
Flowchart FOR21. FORTRAN Compiler, Phase 18



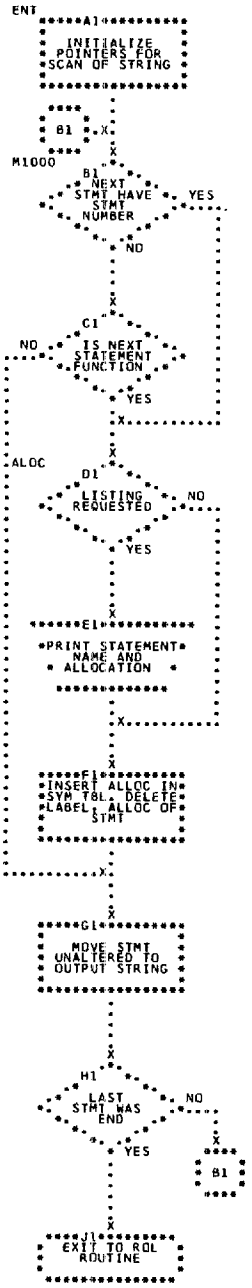
Flowchart FOR22. FORTRAN Compiler, Phase 19



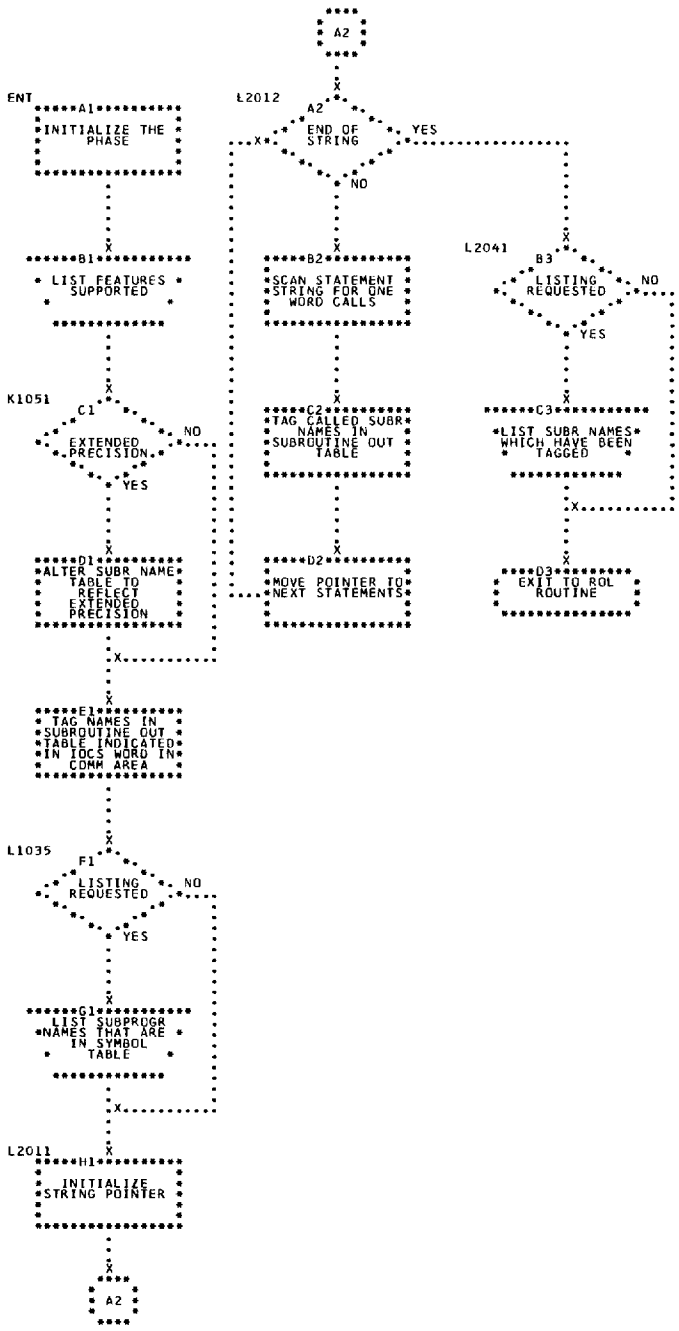
Flowchart FOR23. FORTRAN Compiler, Phase 20



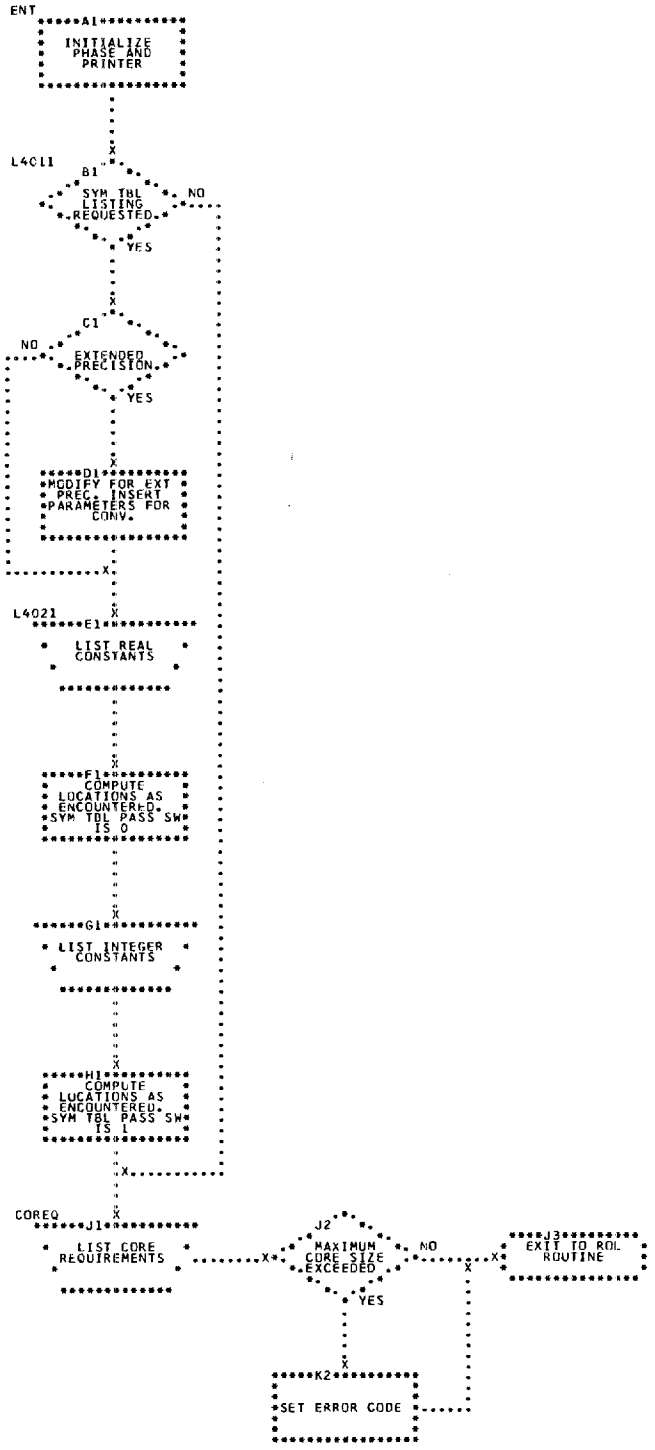
Flowchart FOR24. FORTRAN Compiler, Phase 21



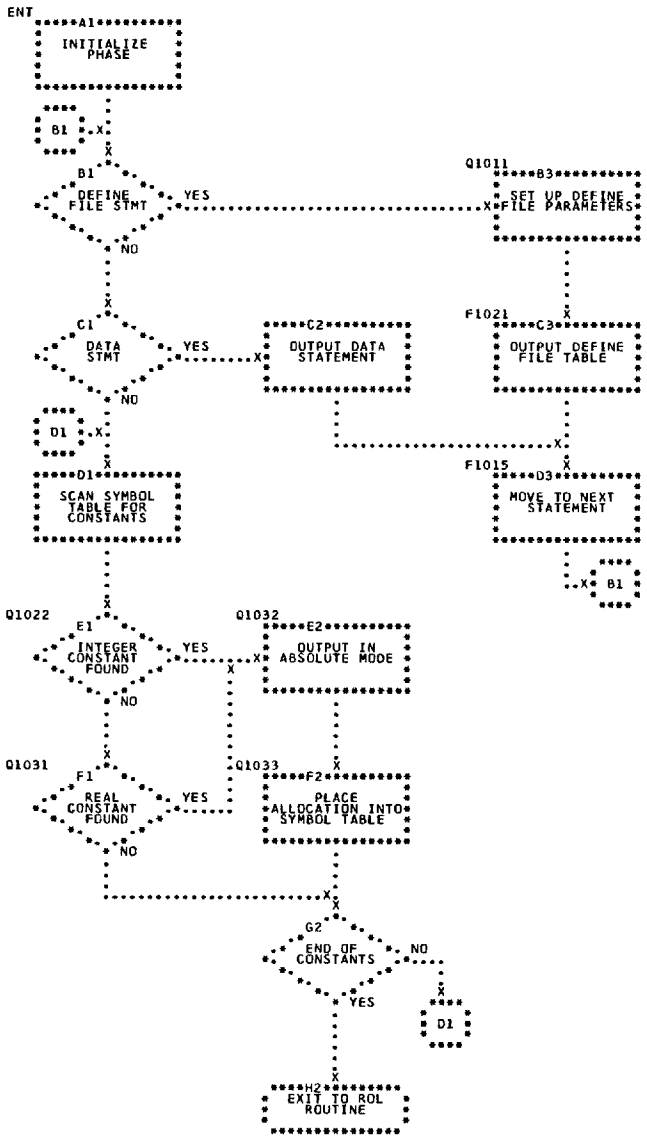
Flowchart FOR25. FORTRAN Compiler, Phase 22



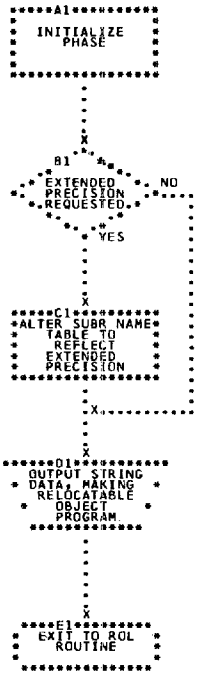
Flowchart FOR26. FORTRAN Compiler, Phase 23



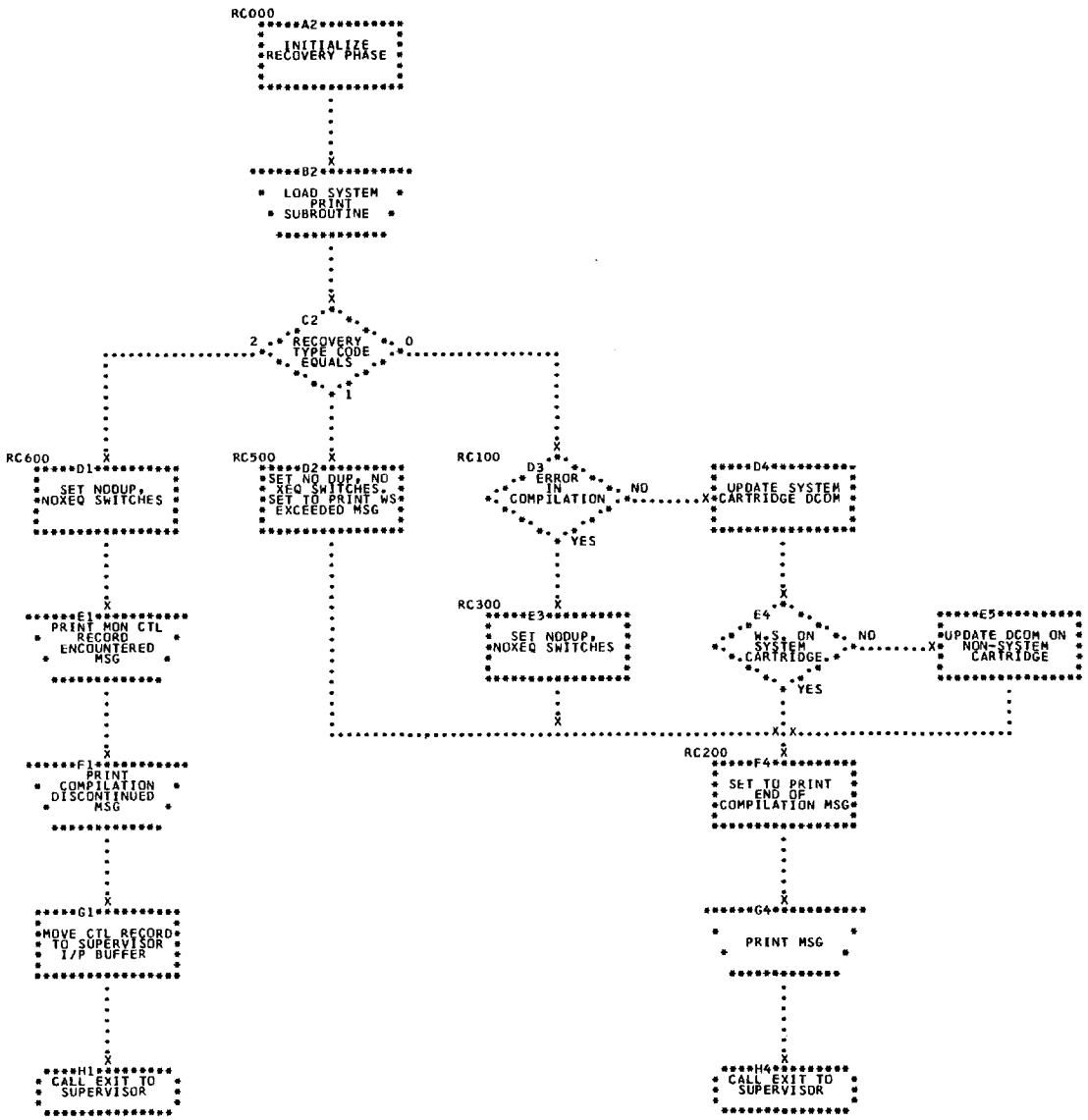
Flowchart FOR27. FORTRAN Compiler, Phase 24



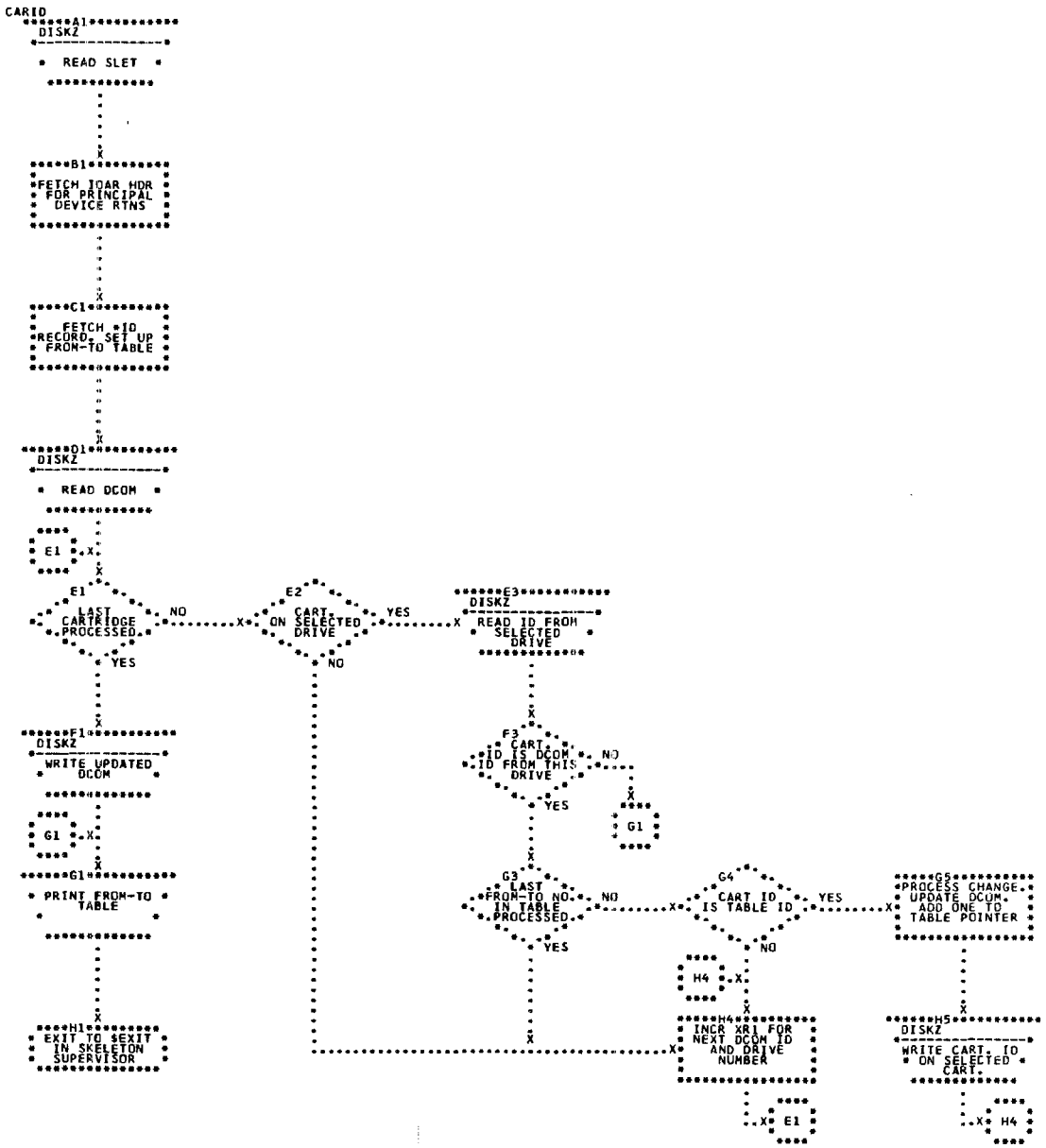
Flowchart FOR28. FORTRAN Compiler, Phase 25



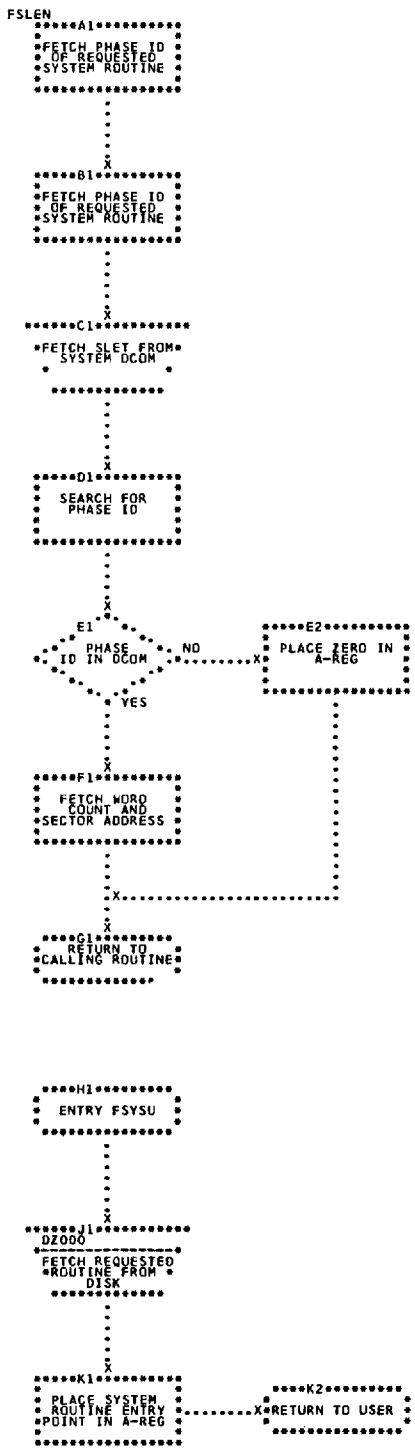
Flowchart FOR29. FORTRAN Compiler, Phase 26



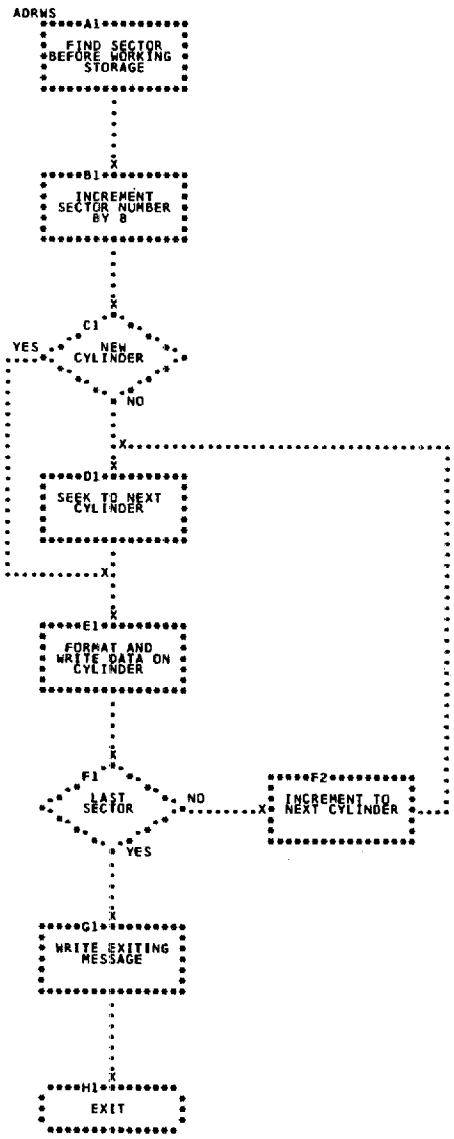
Flowchart FOR30. FORTRAN Compiler, Phase 27



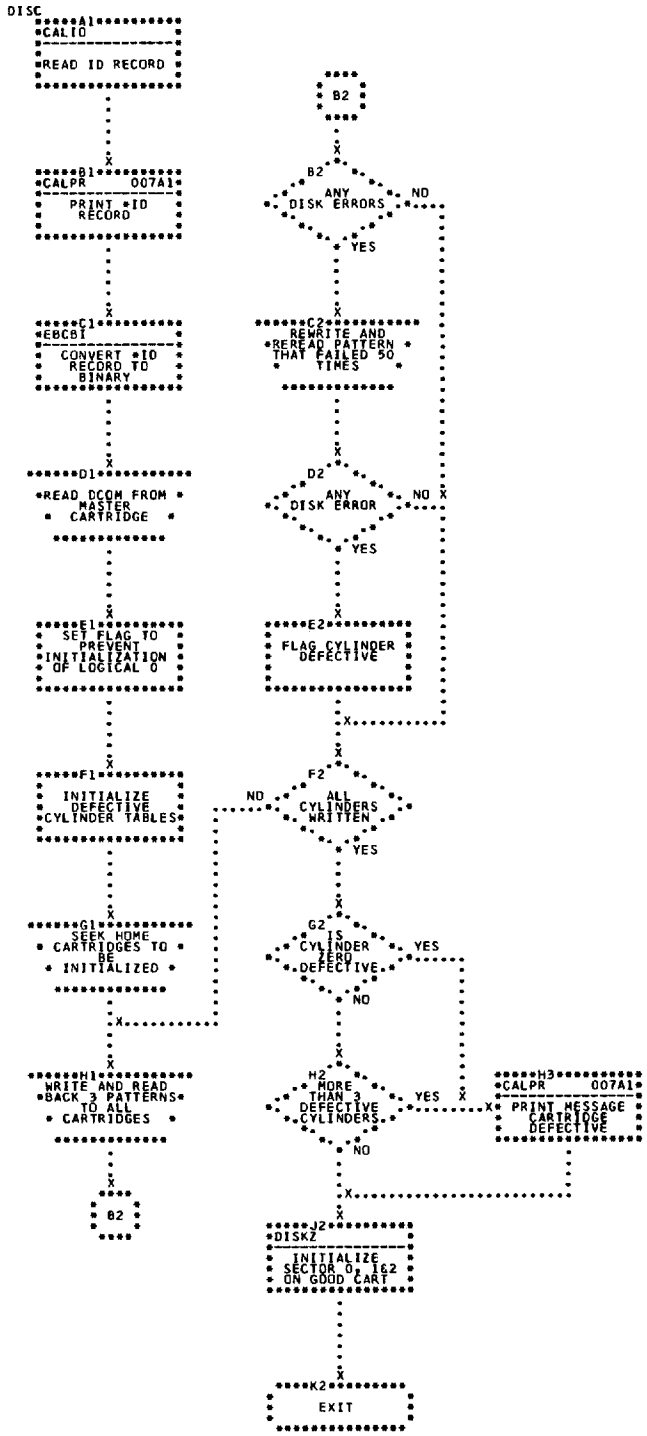
Flowchart UTL01. System Library, ID



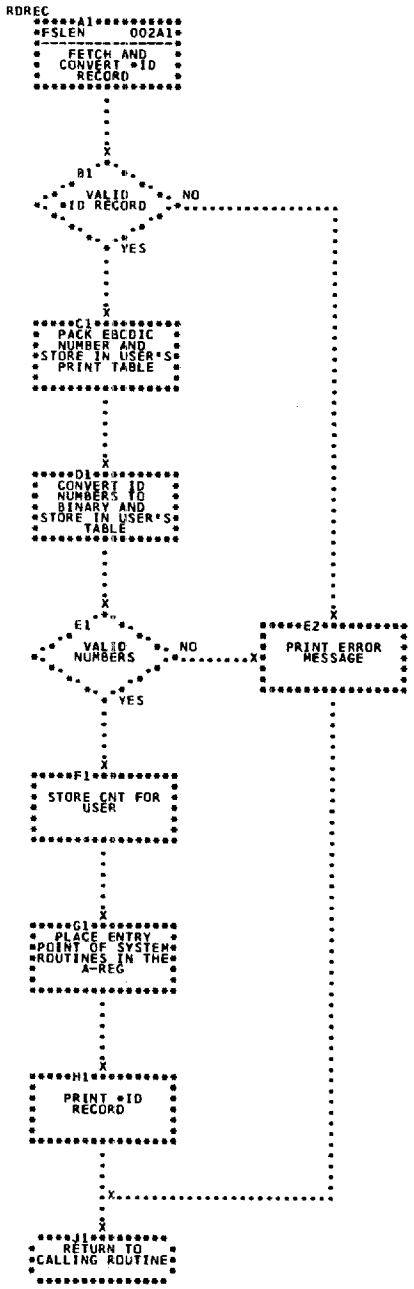
Flowchart UTL02. System Library, FSLEN/FSYSU



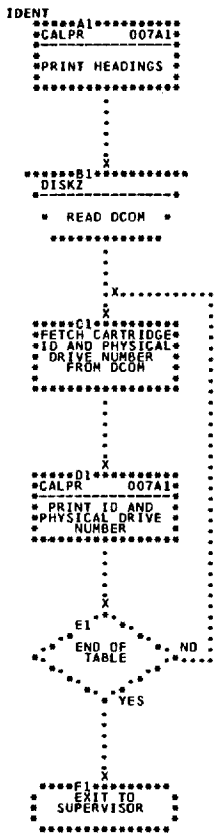
Flowchart UTL03. System Library, ADRWS



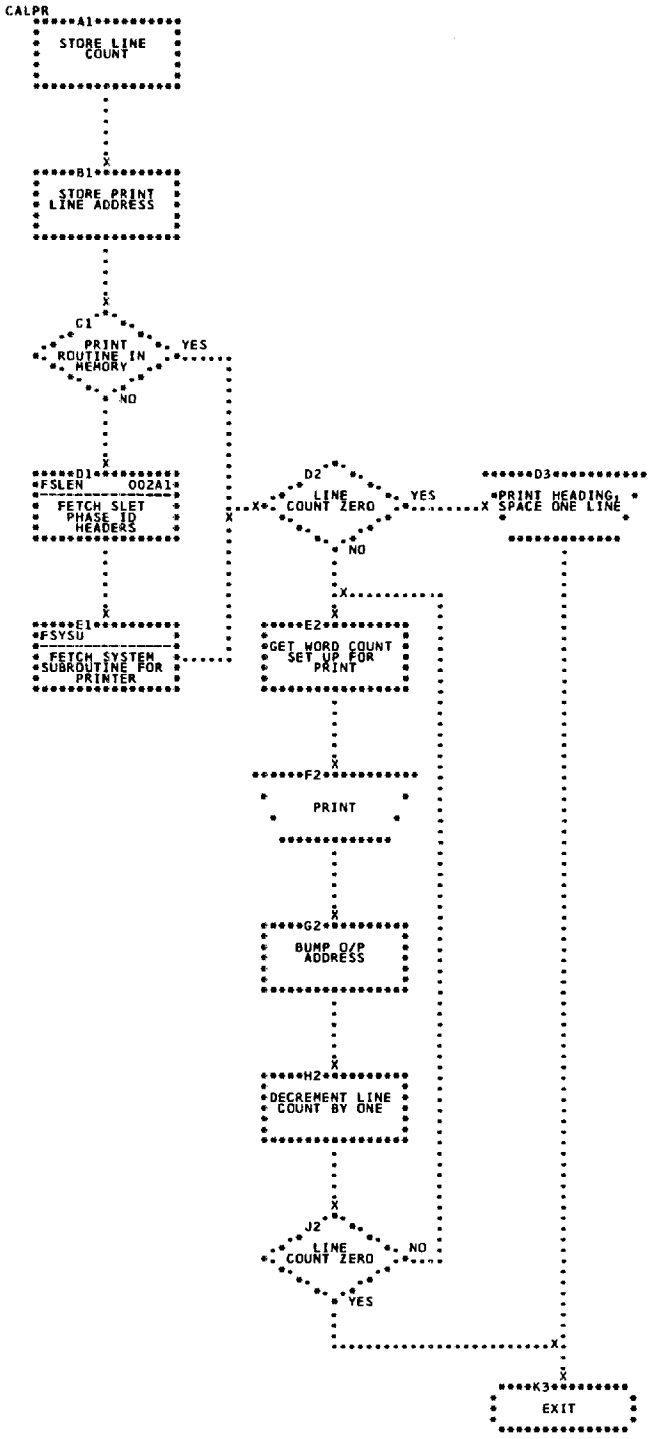
Flowchart UTL04. System Library, DISC



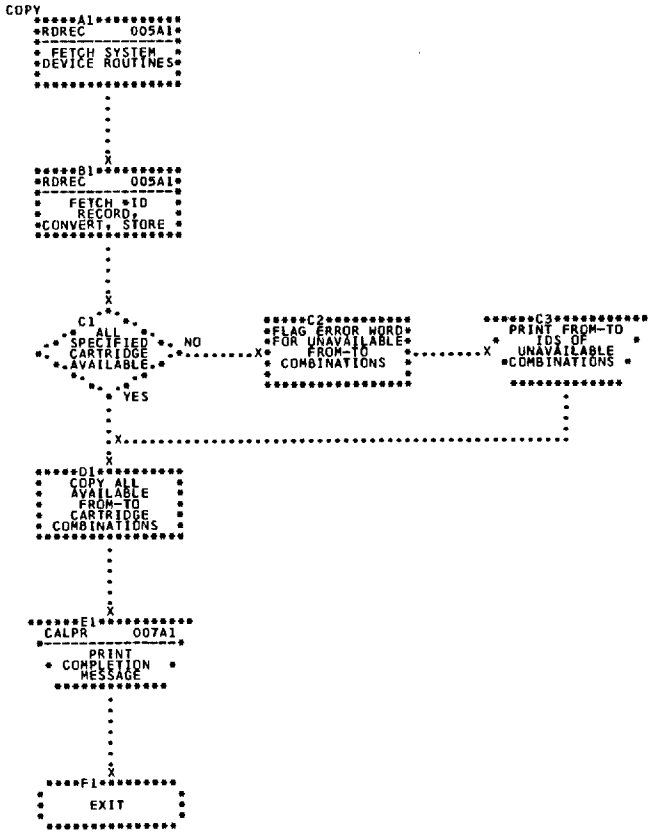
Flowchart UTL05. System Library, RDREC



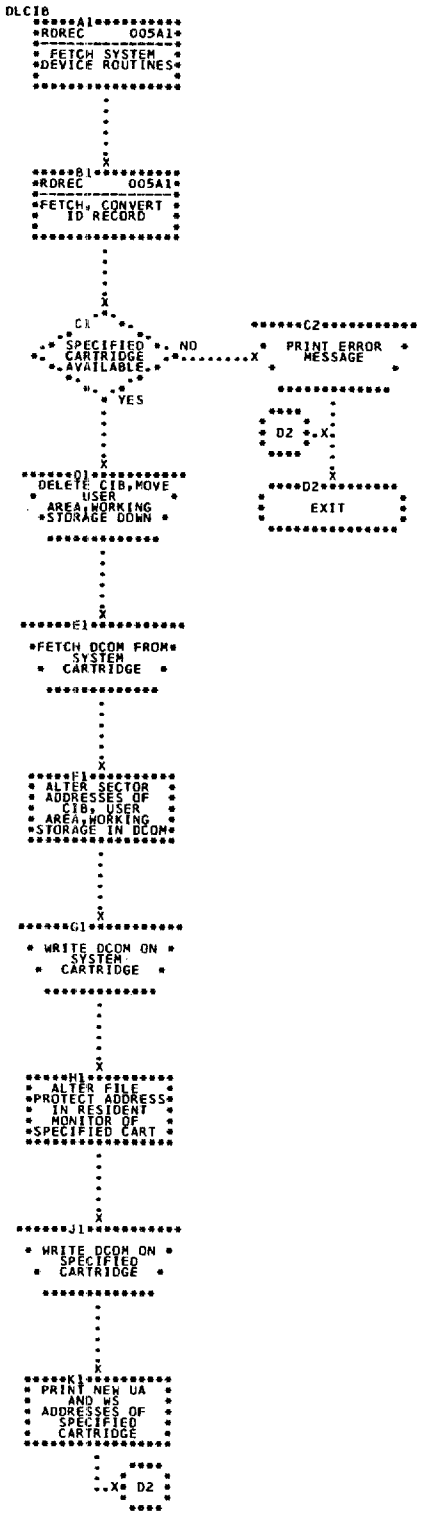
Flowchart UTL06. System Library, IDENT



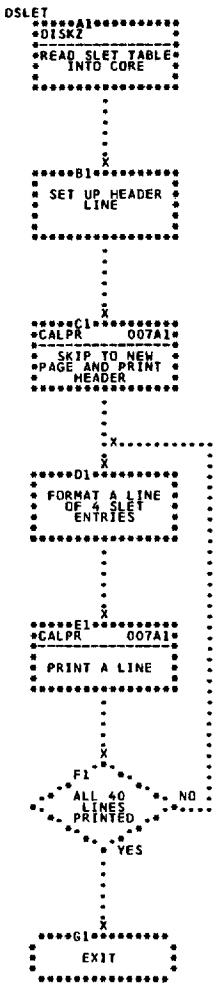
Flowchart UTL07. System Library, CALPR



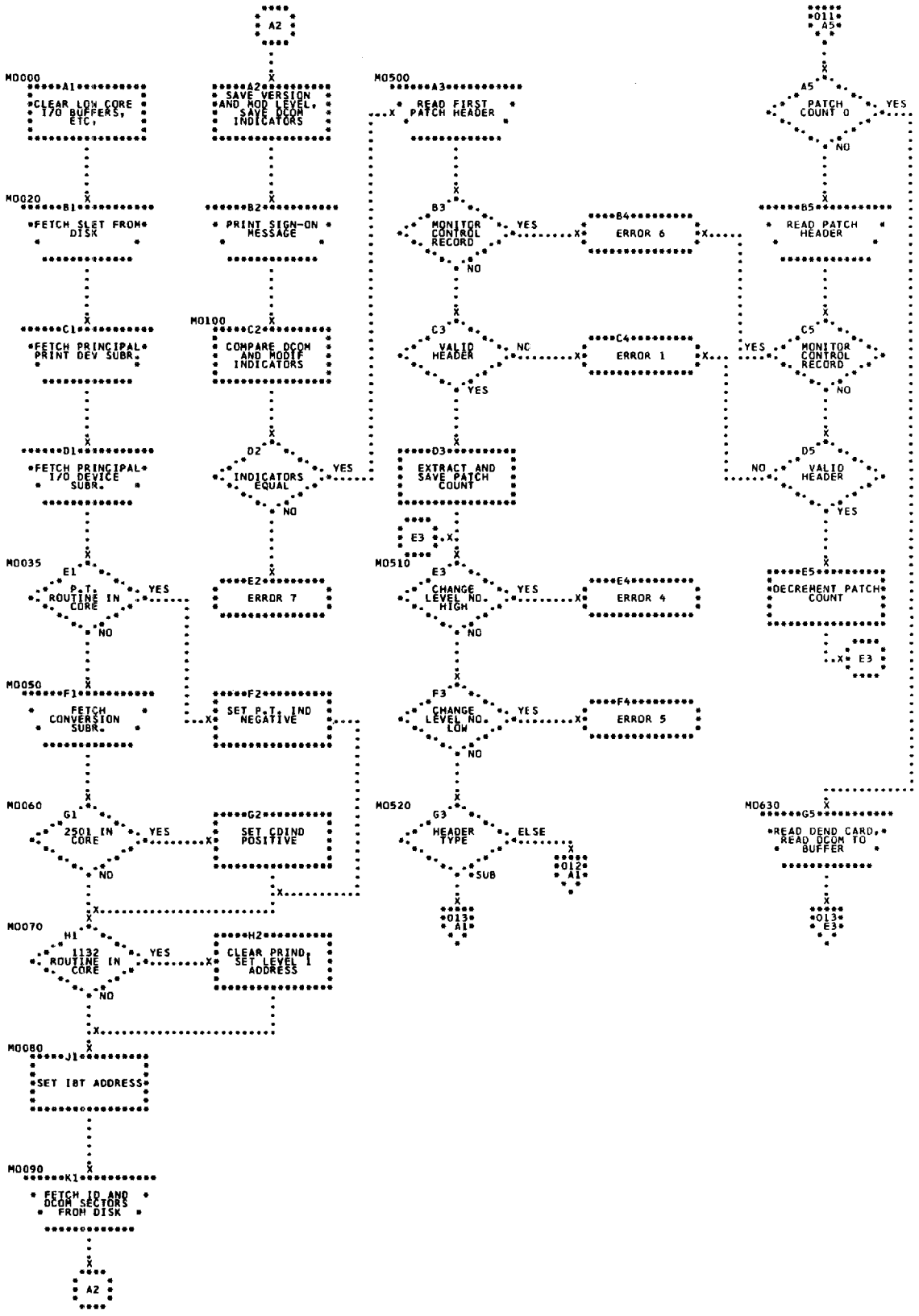
Flowchart UTL08. System Library, COPY



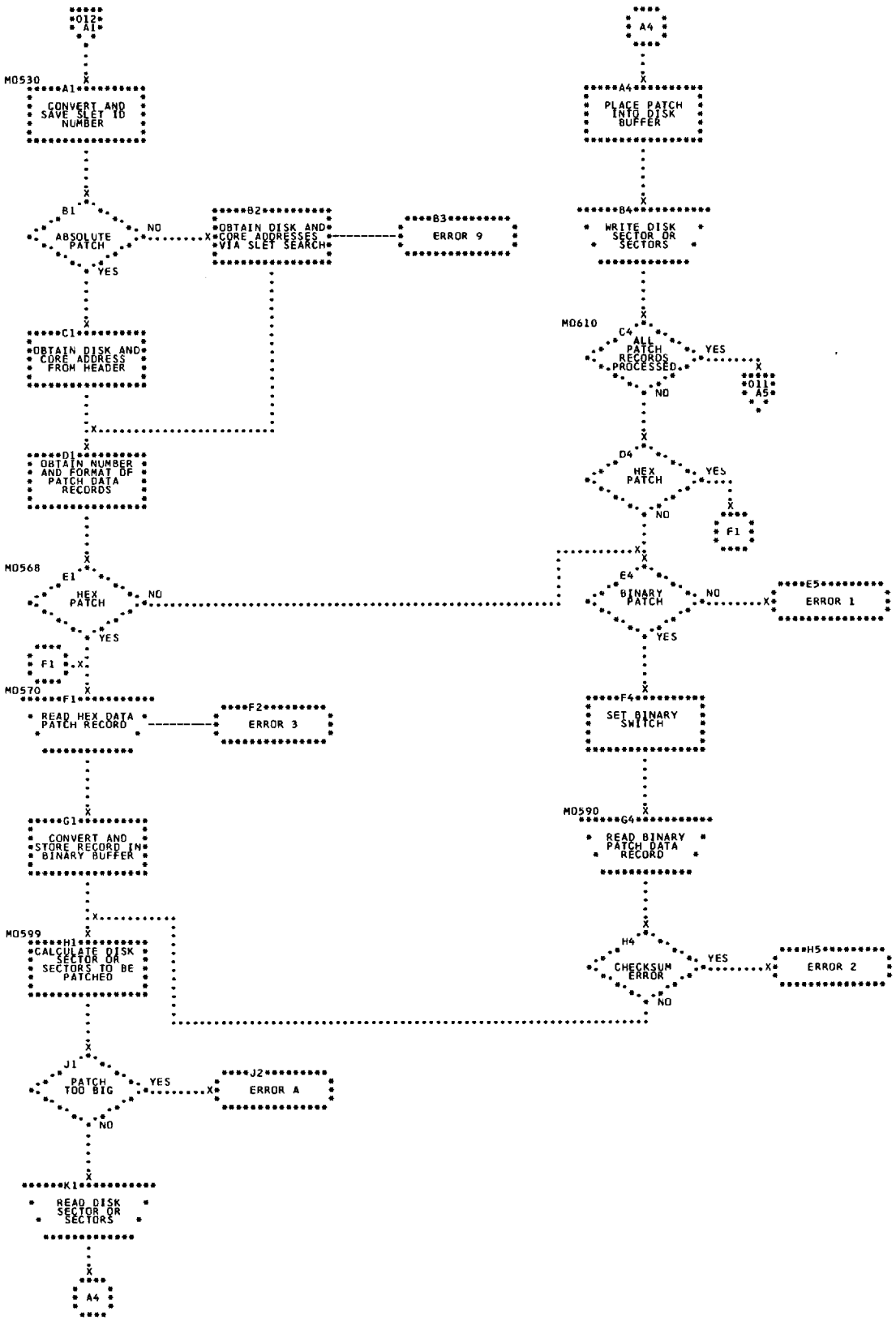
Flowchart UTL09. System Library, DLCIB



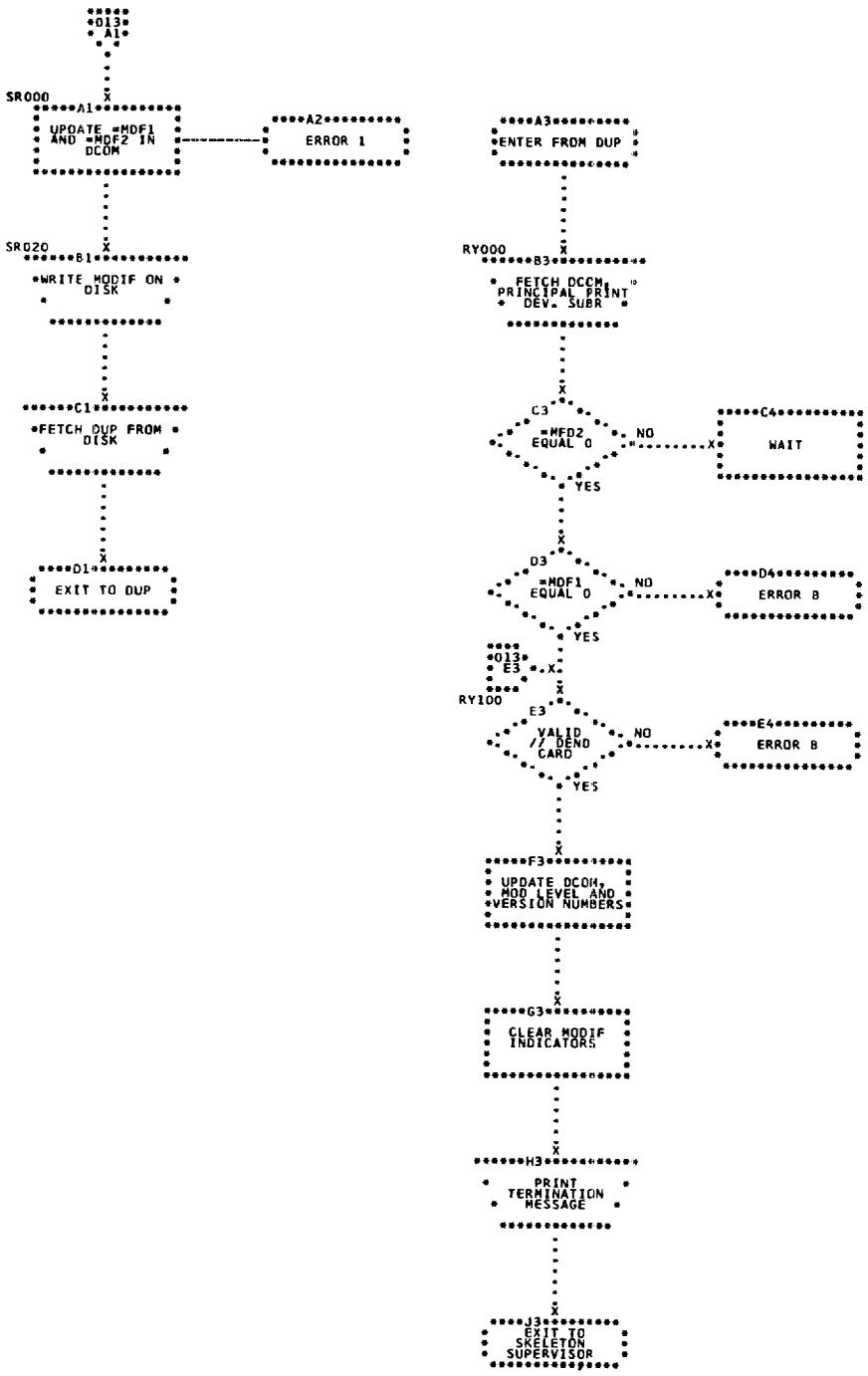
Flowchart UTL10. System Library, DSLET



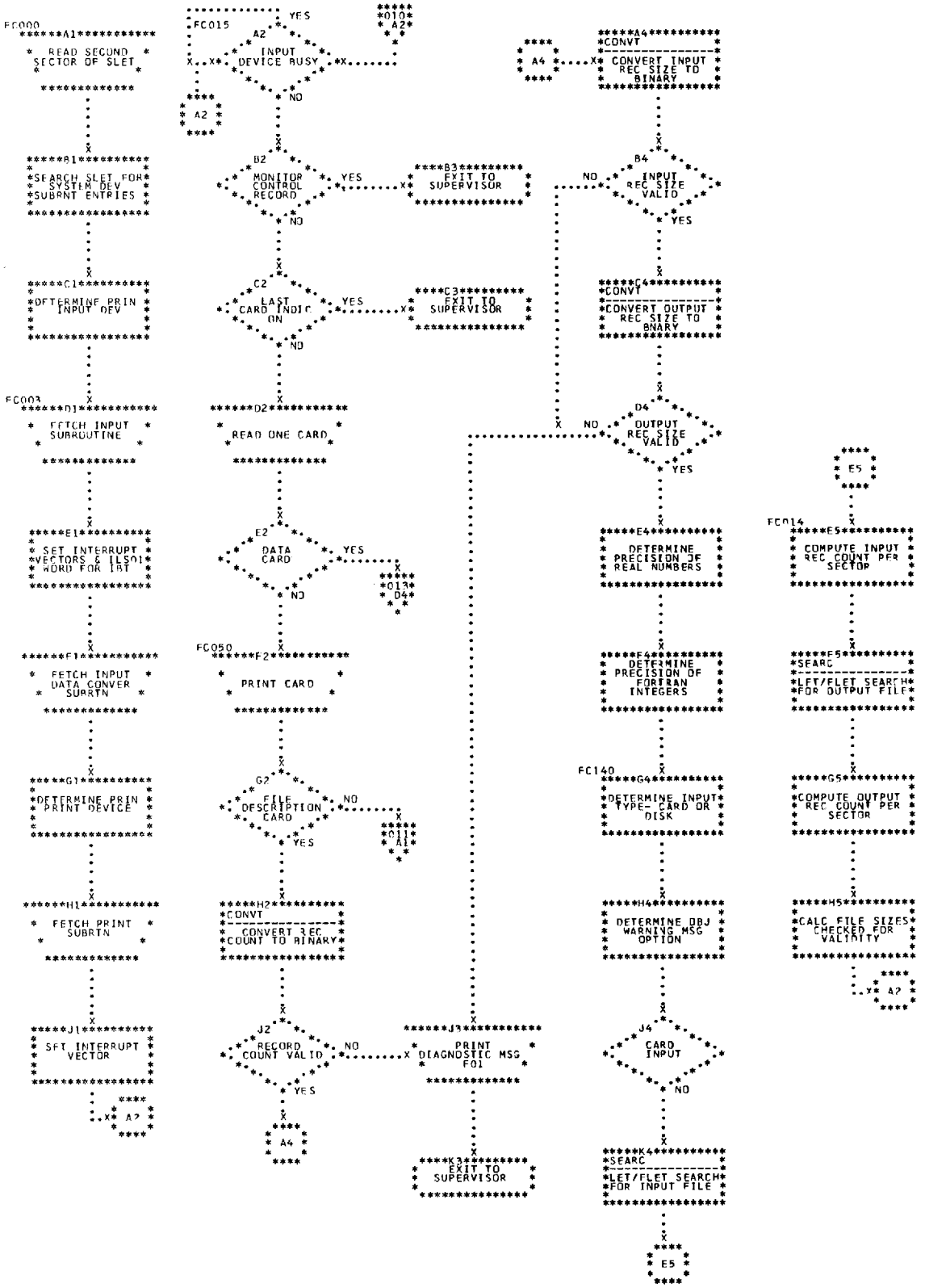
Flowchart UTL11. System Library, MODIF



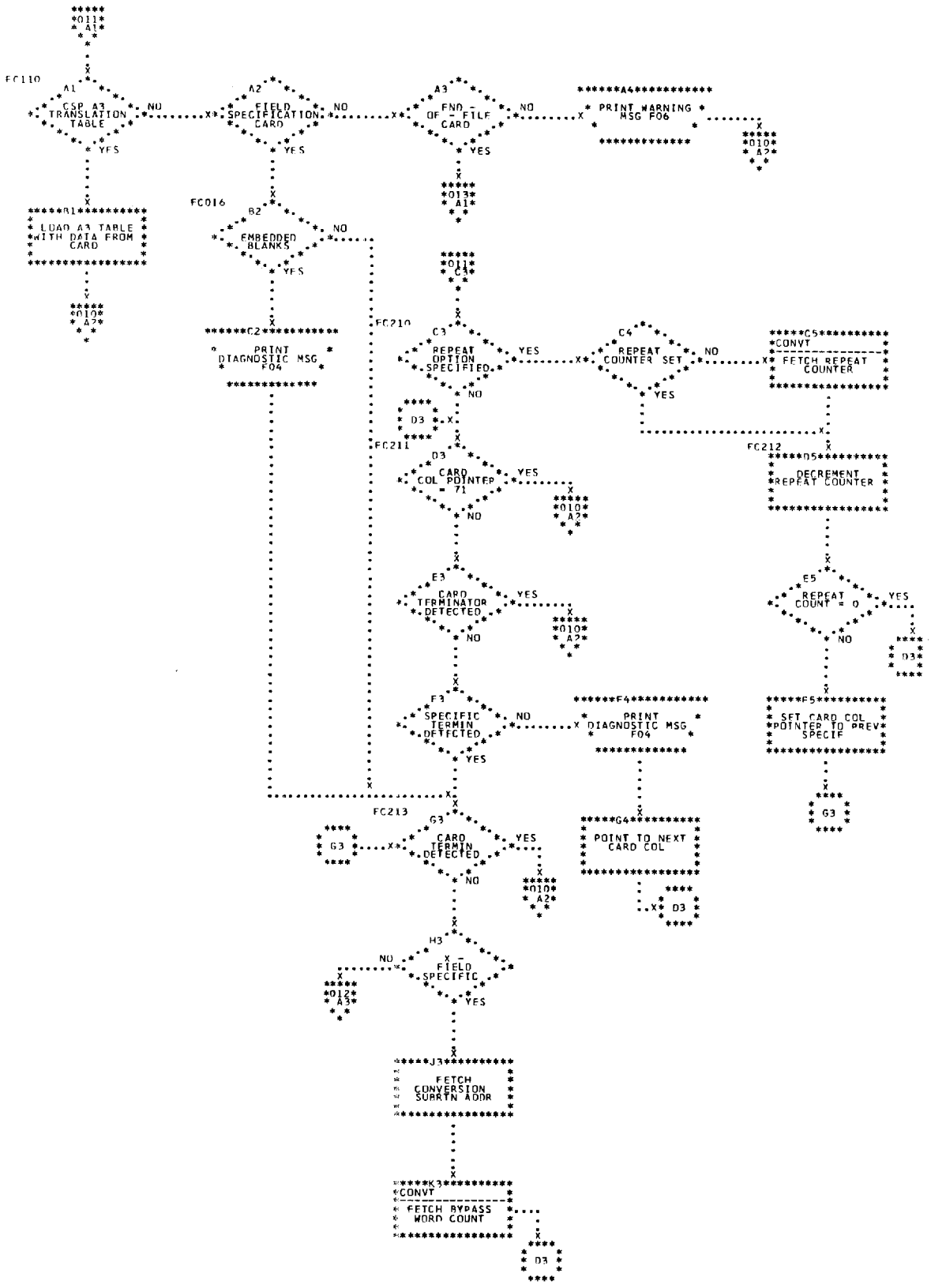
Flowchart UTL12. System Library, MODIF



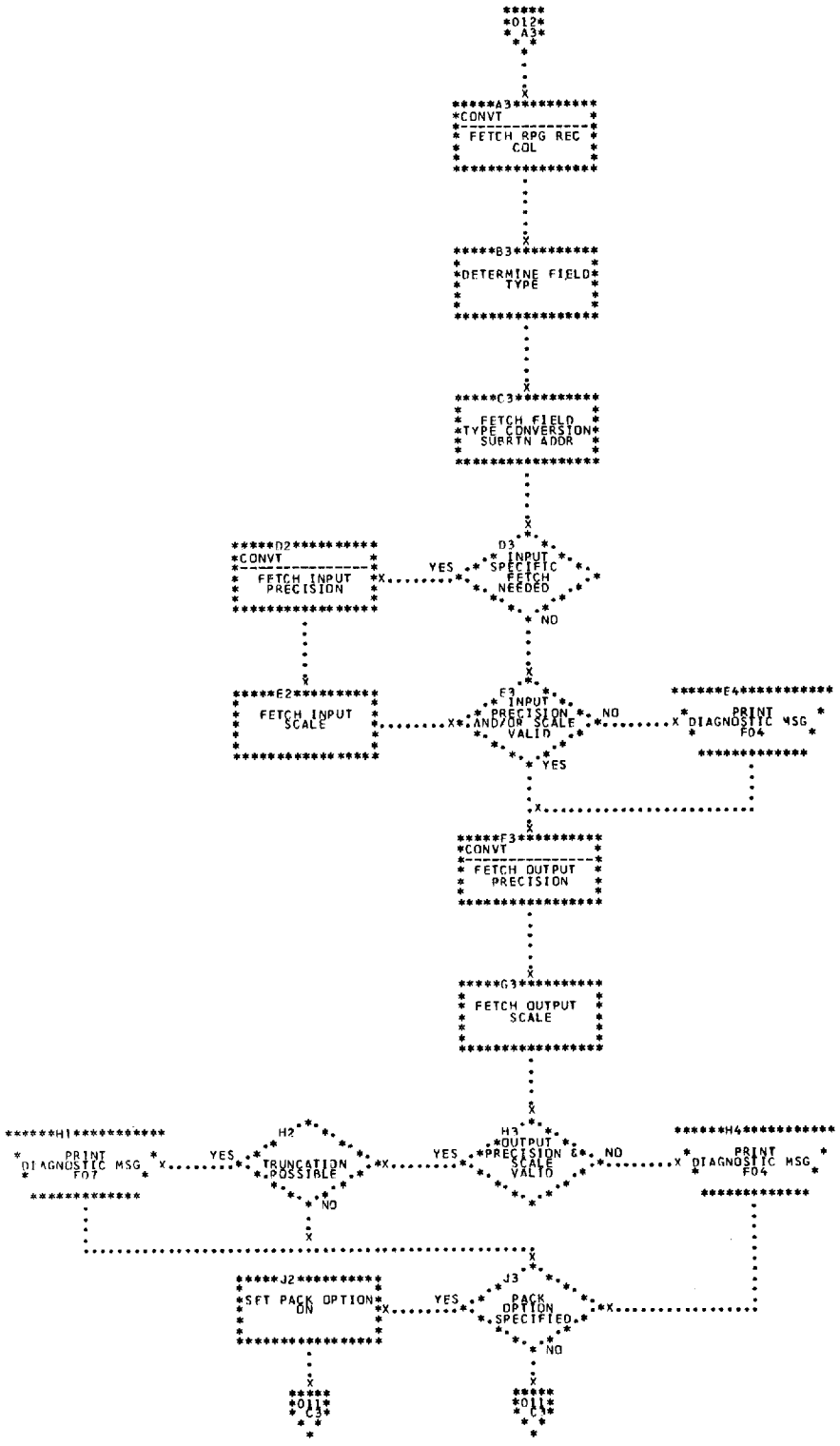
Flowchart UTL13. System Library, MODIF



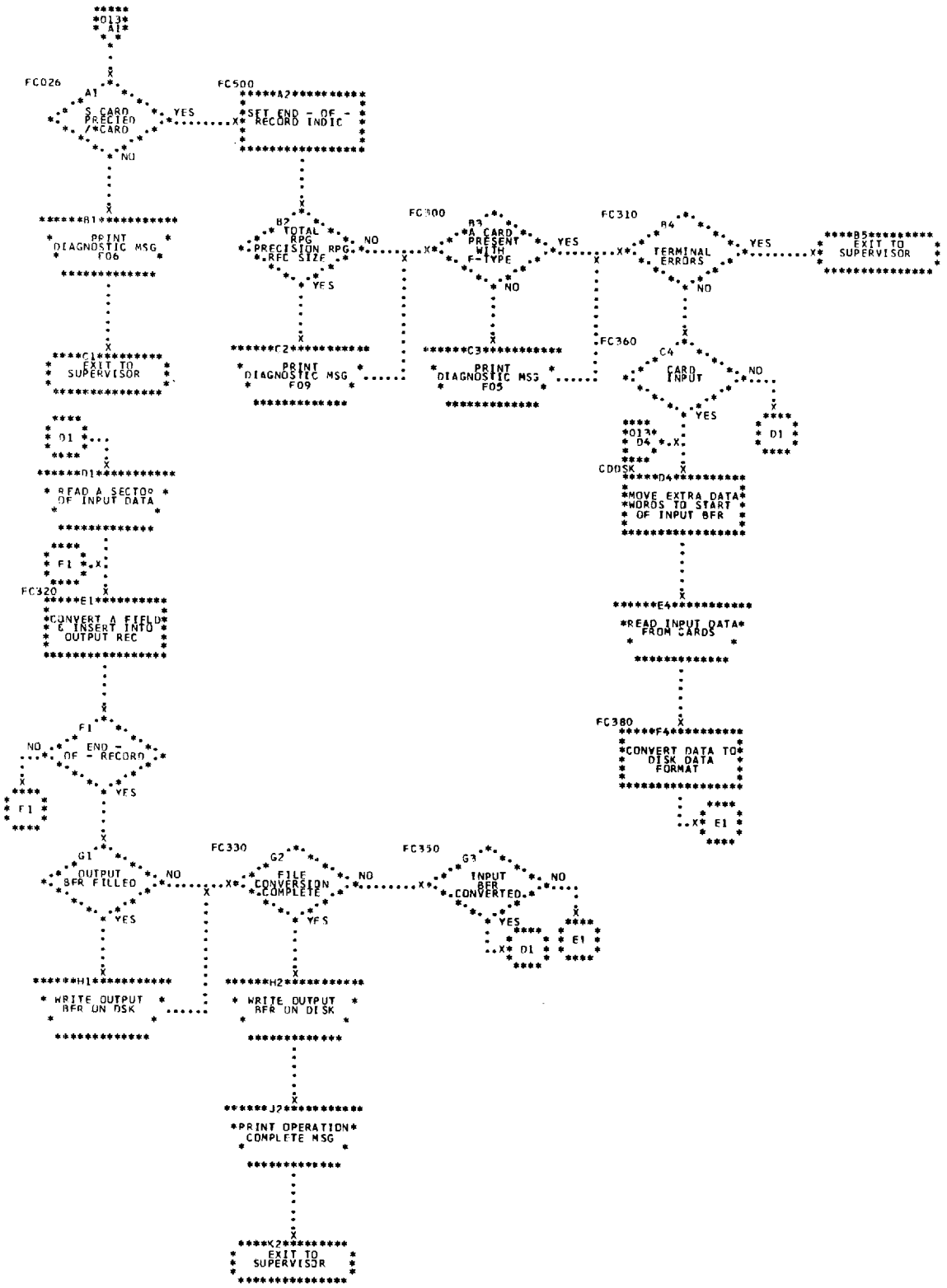
●Flowchart DCN 10, CHART A. System Library Disk Data File Conversion Program (DFCNV)



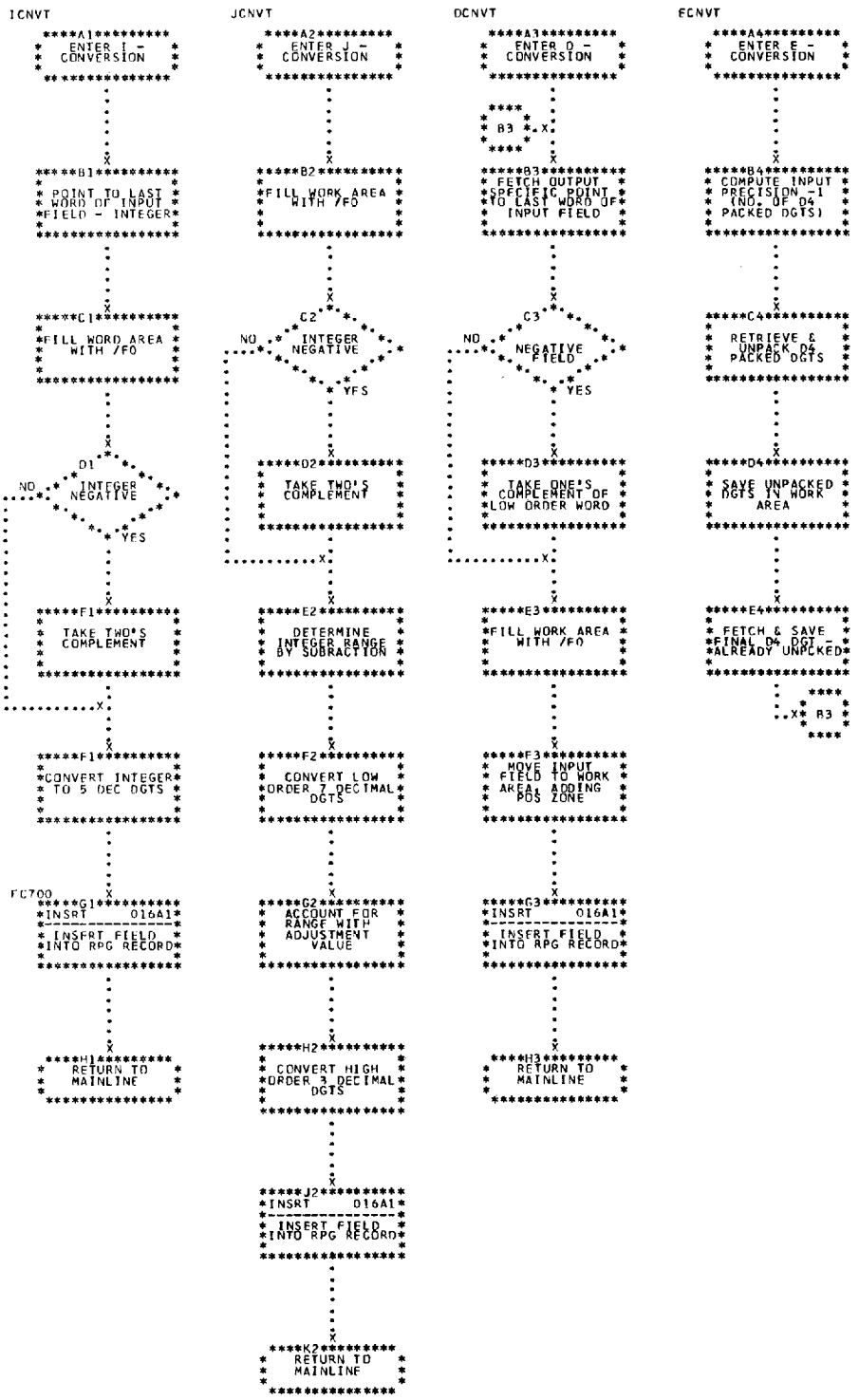
• Flowchart DCN 11, CHART B, System Library Disk Data File Conversion Program (DFCNV)



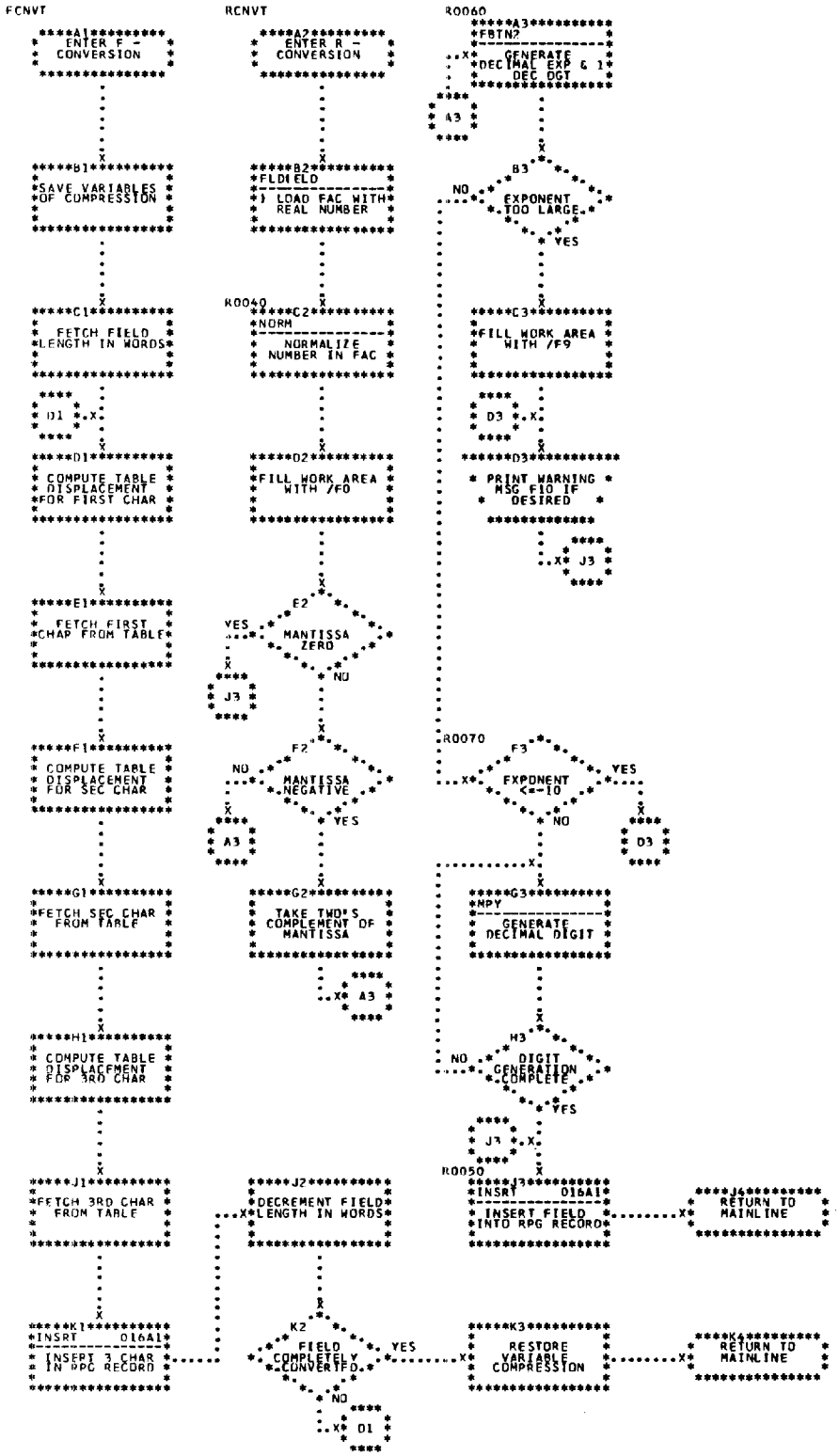
•Flowchart DCN 12. CHART C. System Library Disk Data File Conversion Program (DFCNV)



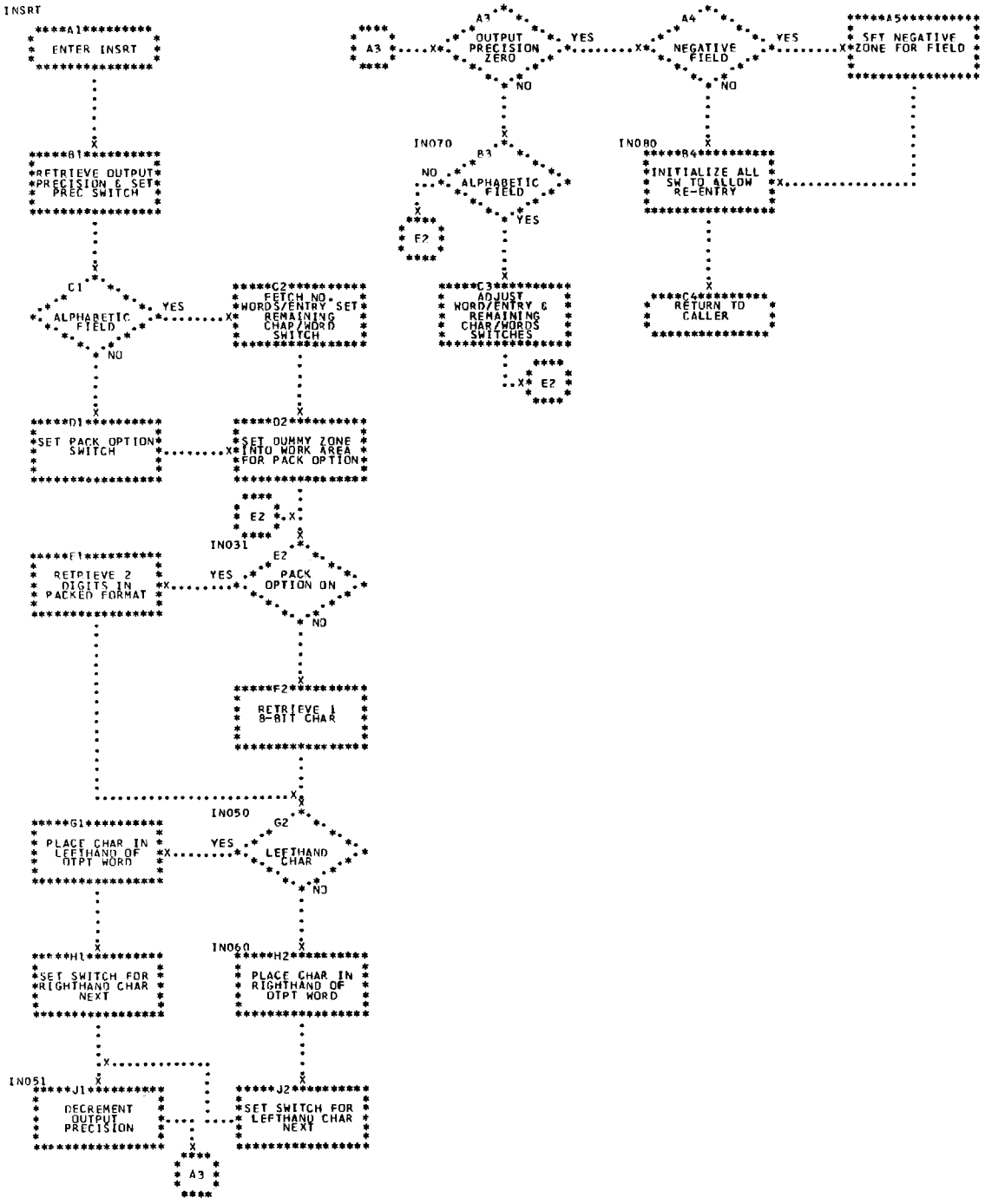
•Flowchart DCN 13. CHART D. System Library Disk Data File Conversion Program (DFCNV)



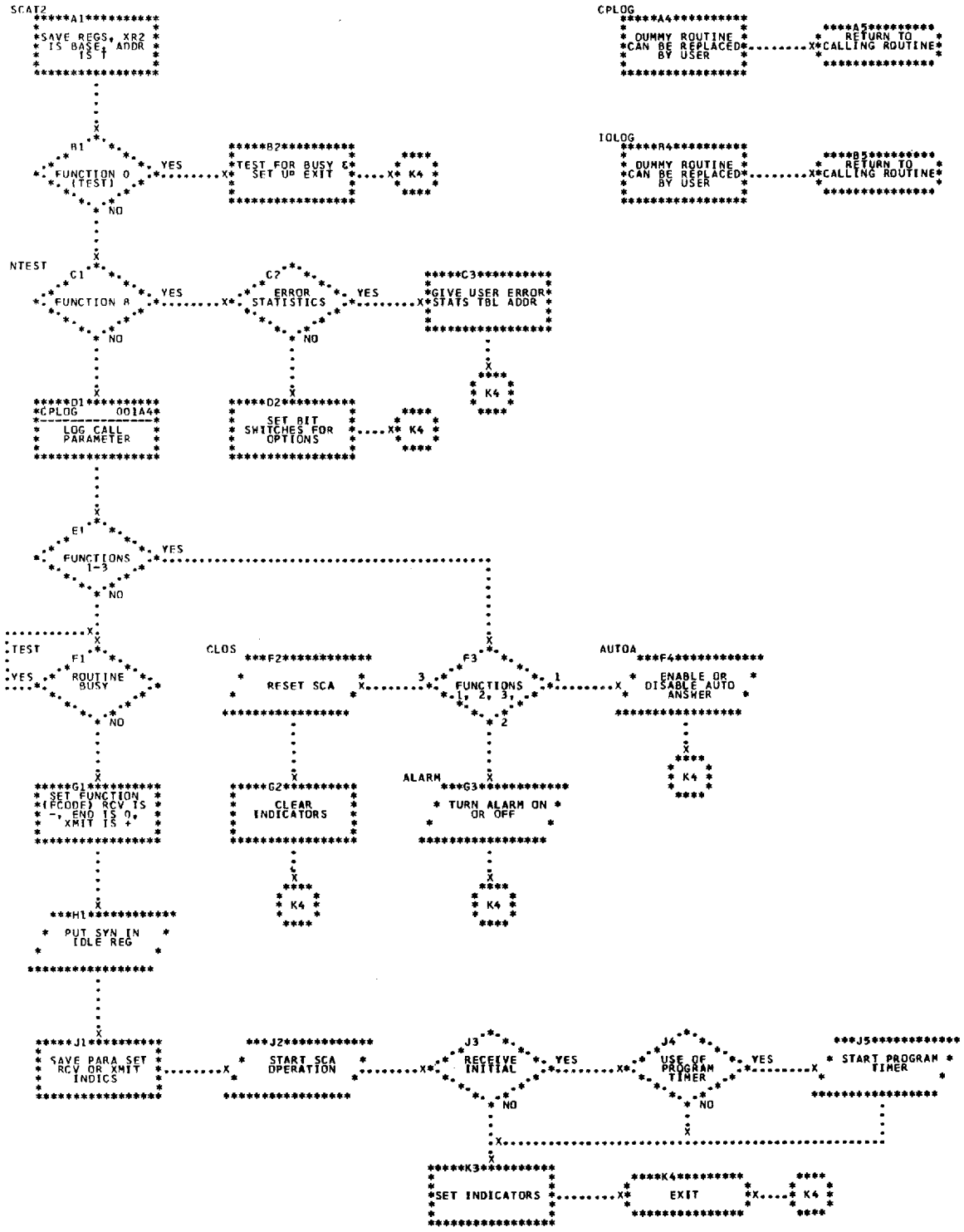
•Flowchart DCN 14. CHART E. System Library Disk Data File Conversion Program (DFCNV)



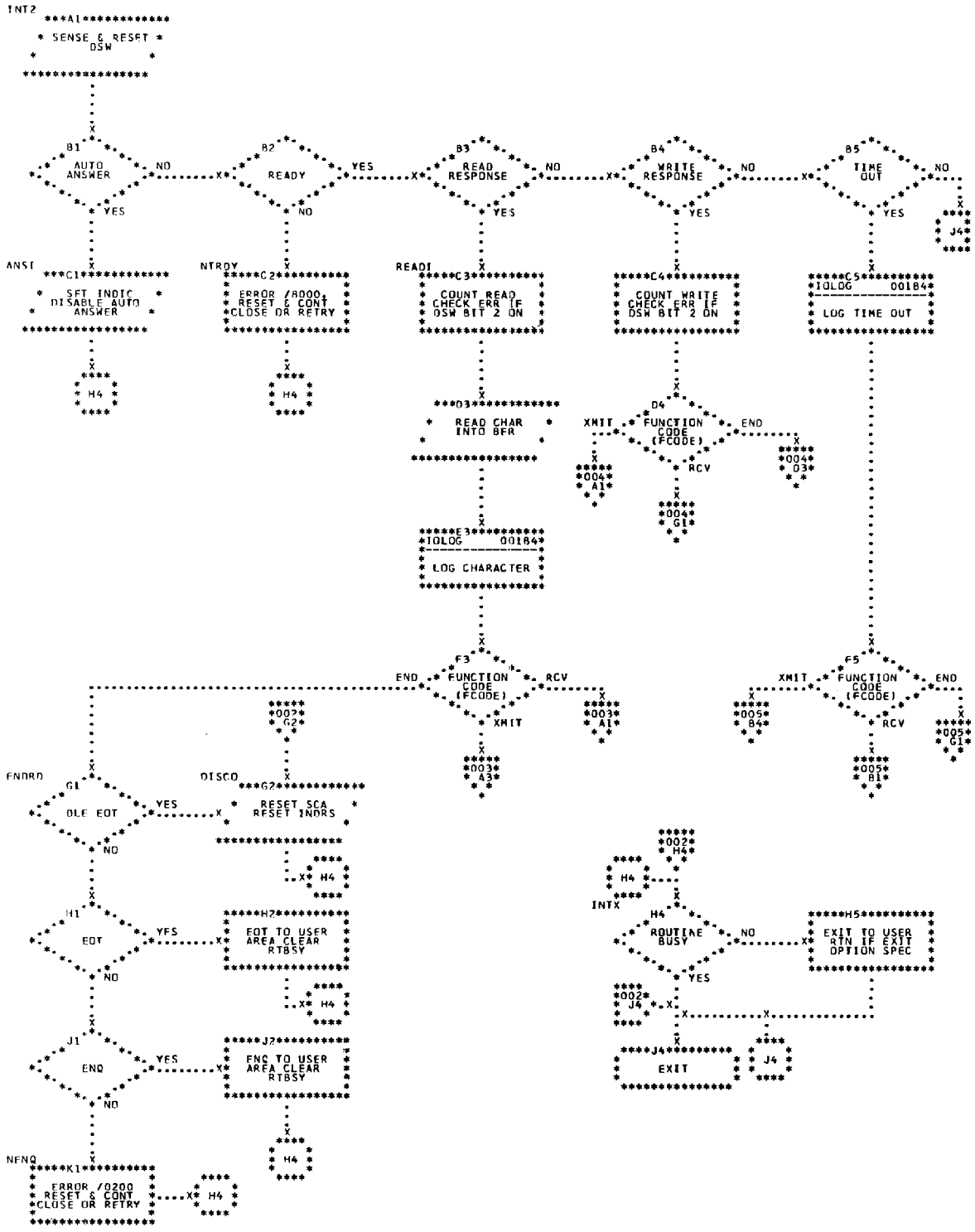
•Flowchart DCN 15. CHART F. System Library Disk Data File Conversion Program (DFCNV)



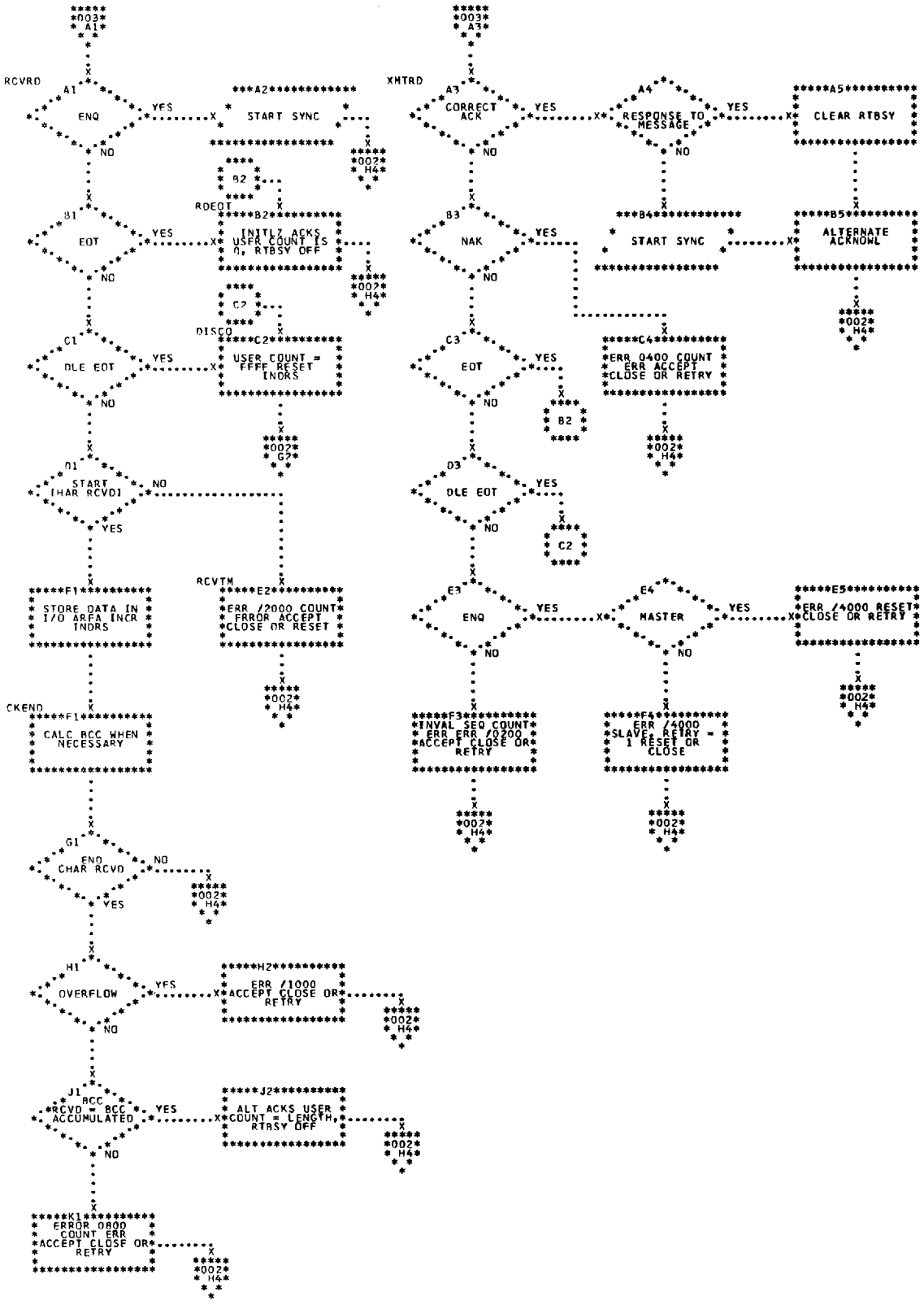
•Flowchart DCN 16. CHART G. System Library Disk Data File Conversion Program (DFCNV)



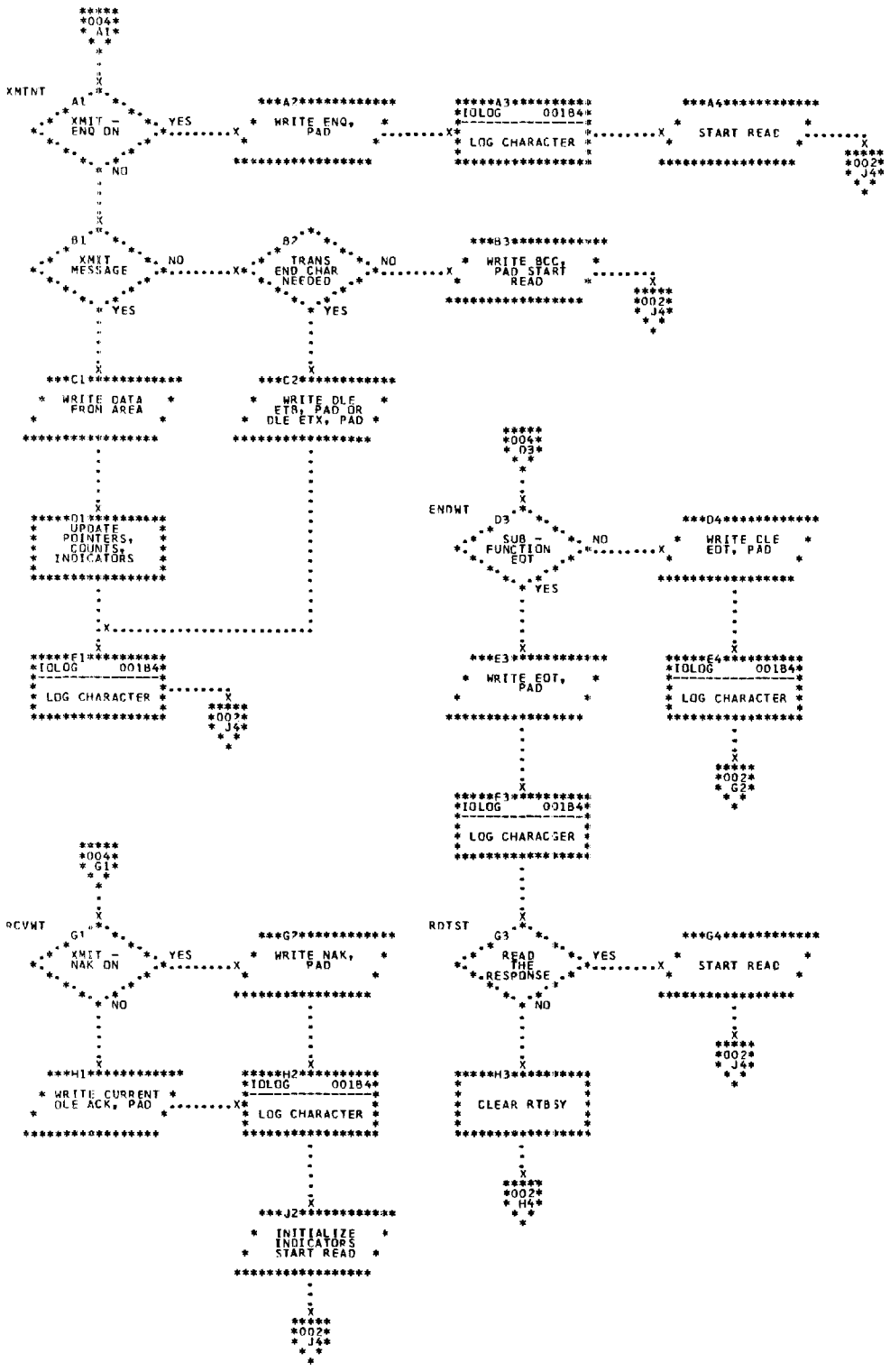
Flowchart SCA01. System Library, SCAT2 Call Processing



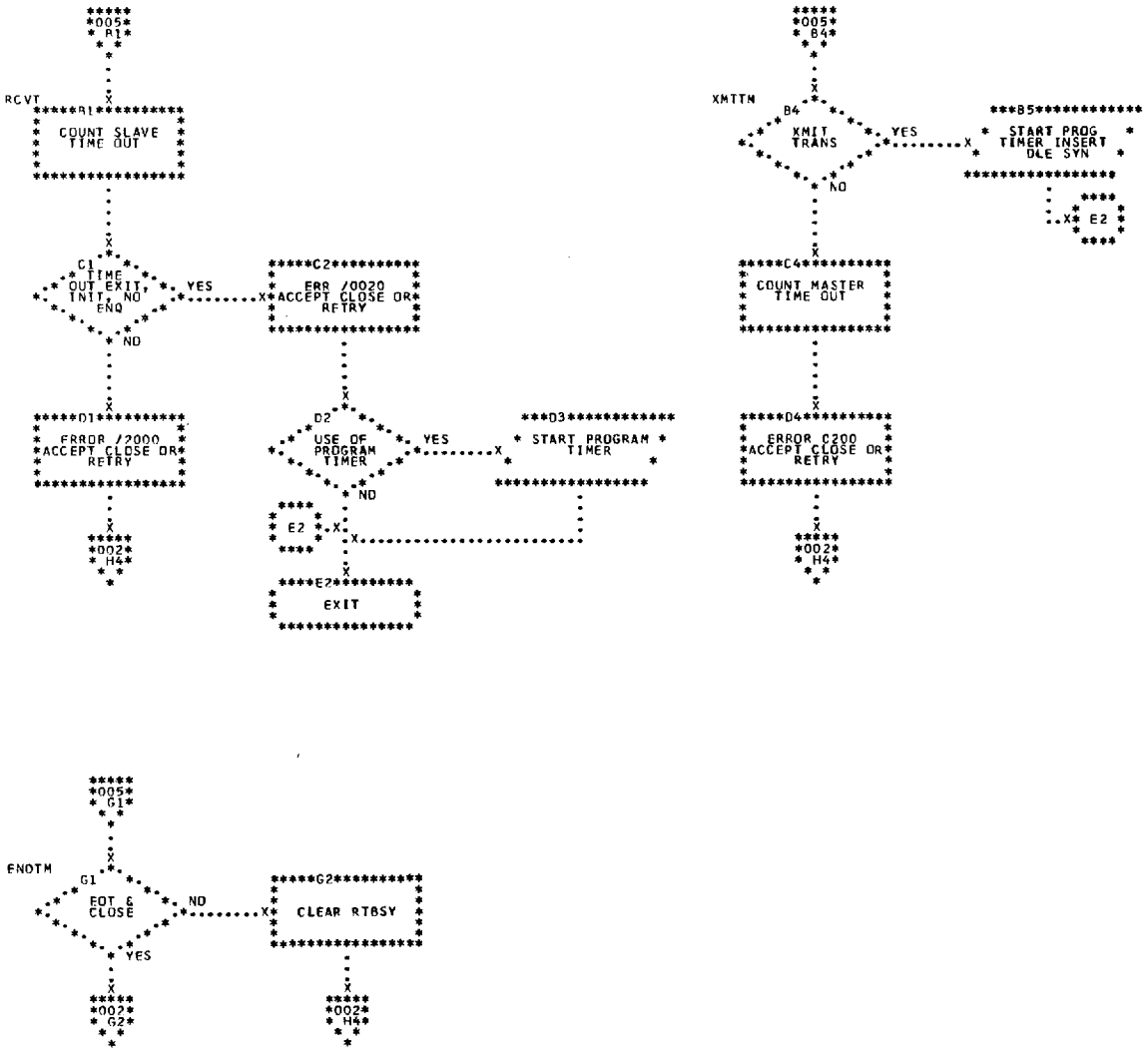
Flowchart SCA02. System Library, SCAT2 Interrupt Processing



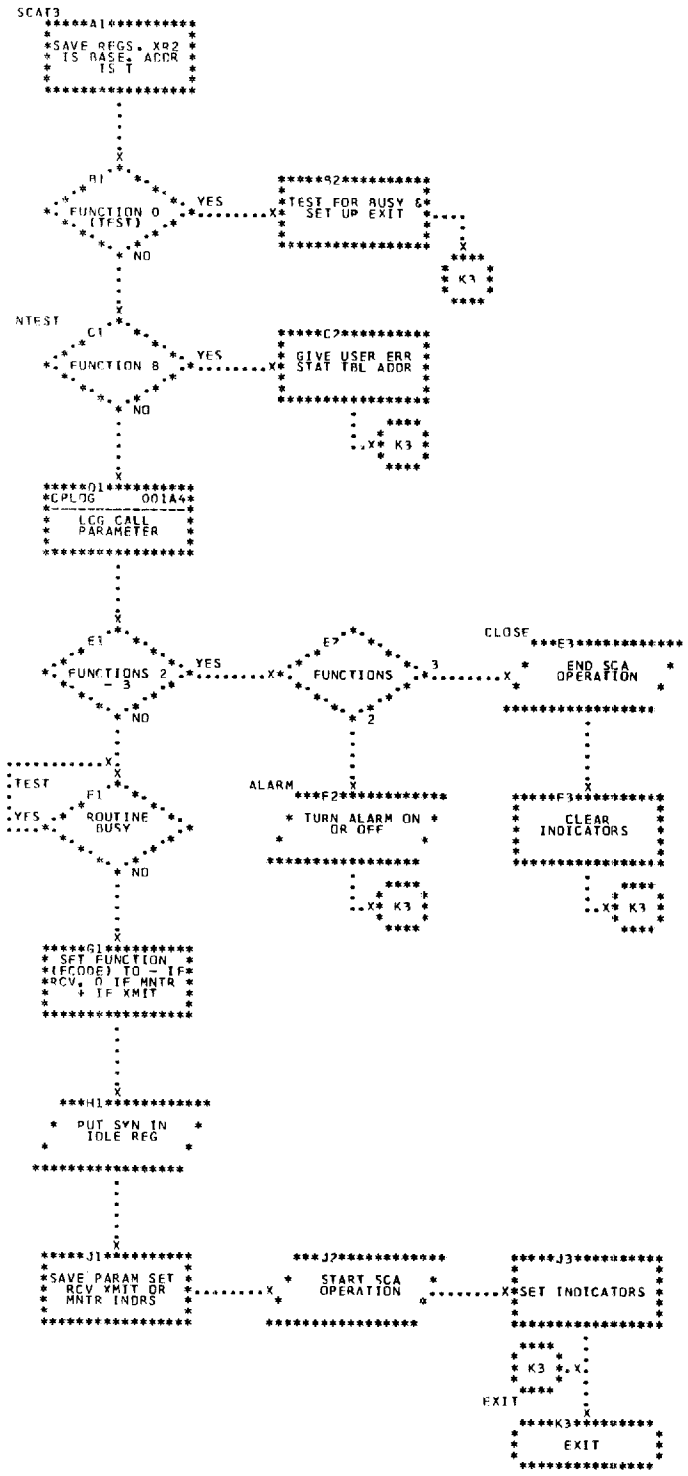
Flowchart SCA03. System Library, SCAT2 Interrupt Processing



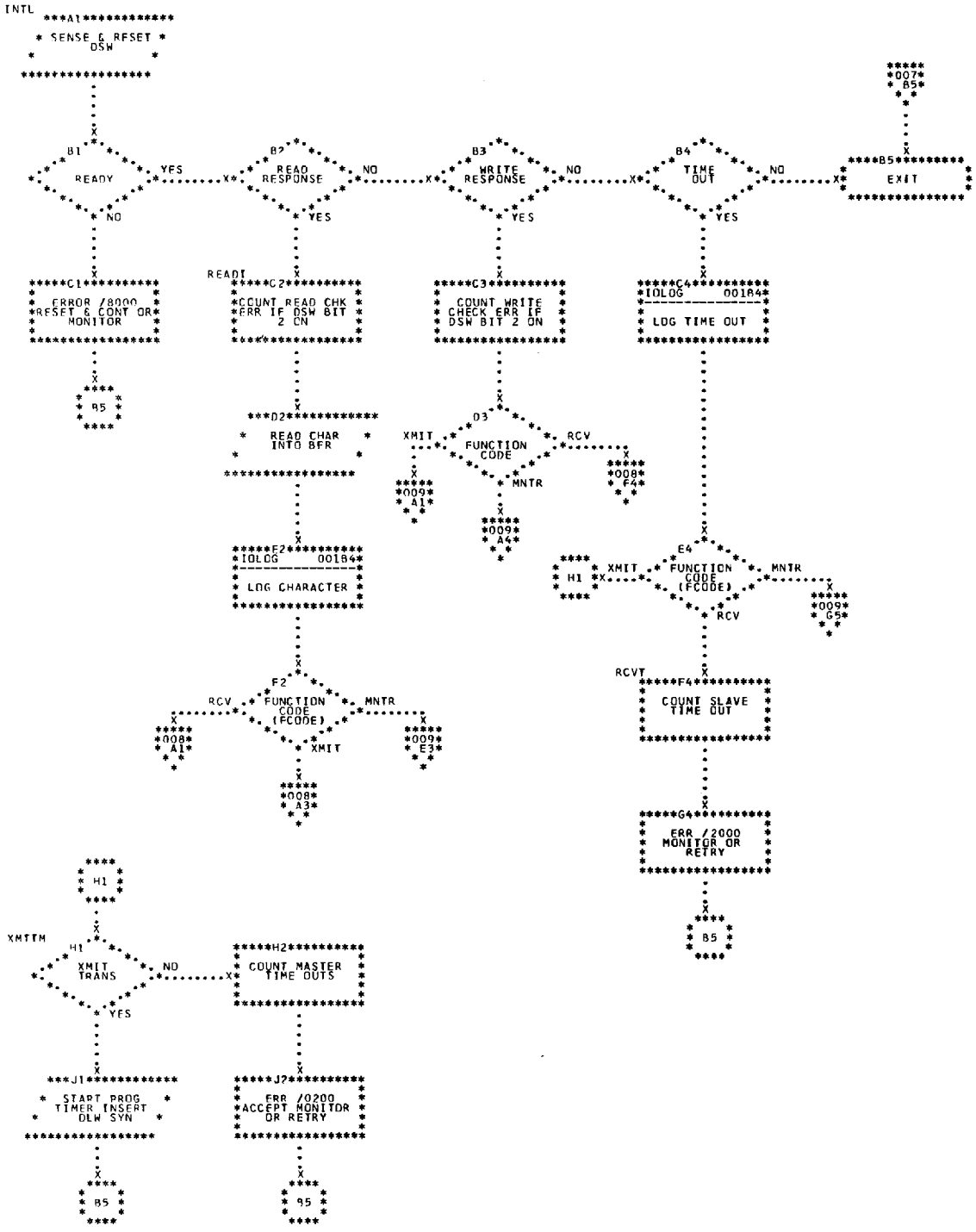
Flowchart SCA04. System Library, SCAT2 Interrupt Processing



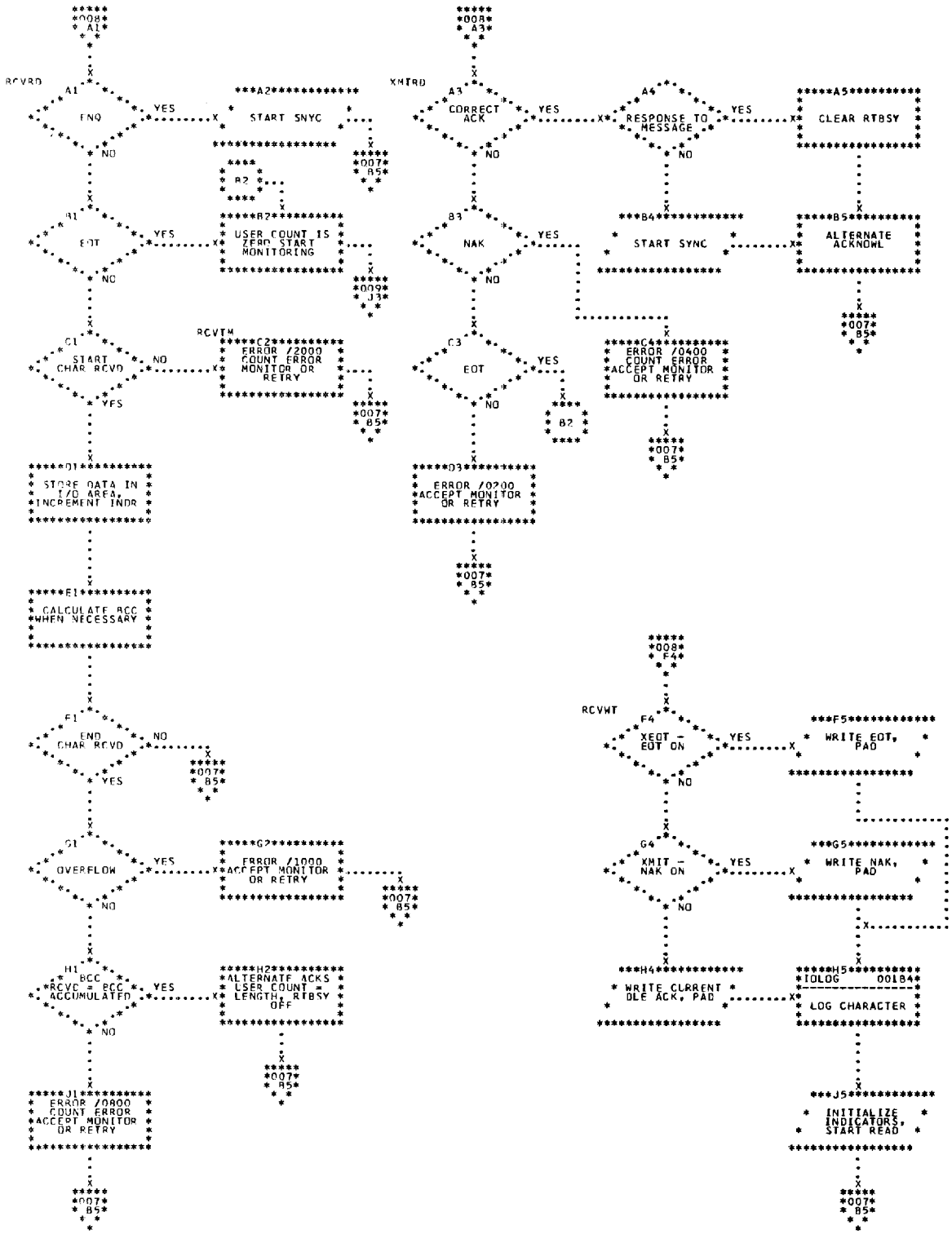
Flowchart SCA05. System Library, SCAT2 Interrupt Processing



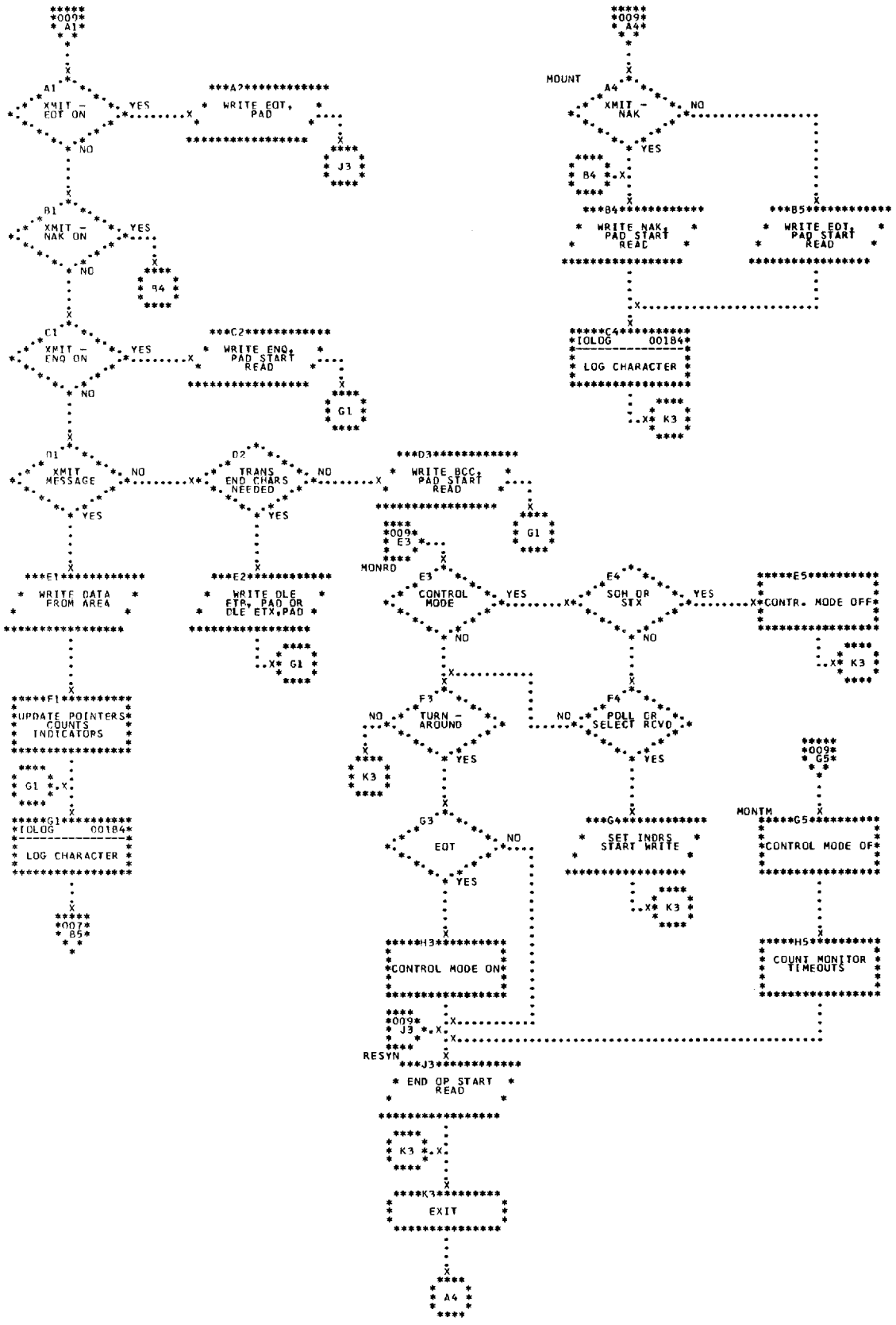
Flowchart SCA06. System Library, SCAT3 Call Processing



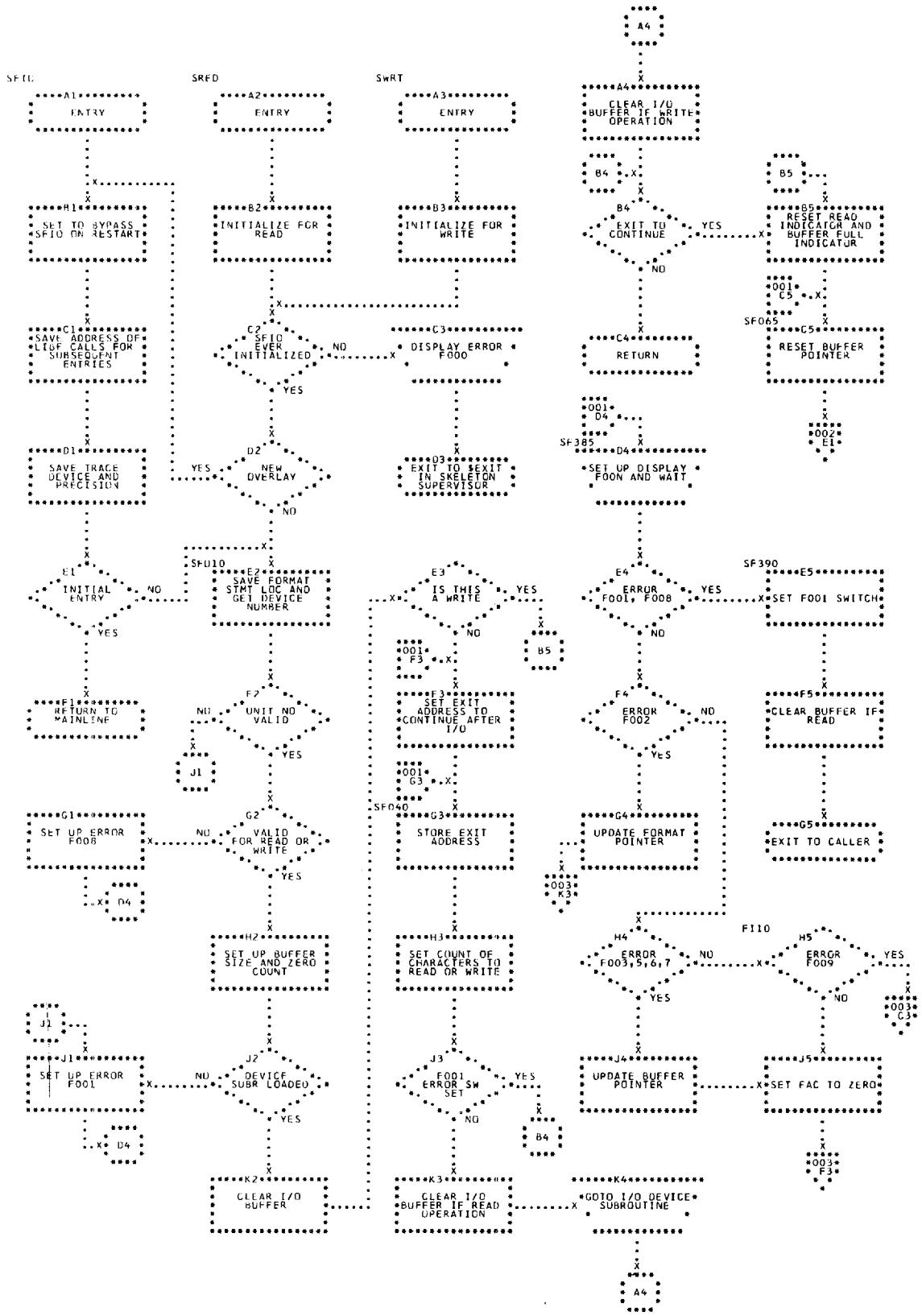
Flowchart SCA07. System Library, SCAT3 Interrupt Processing



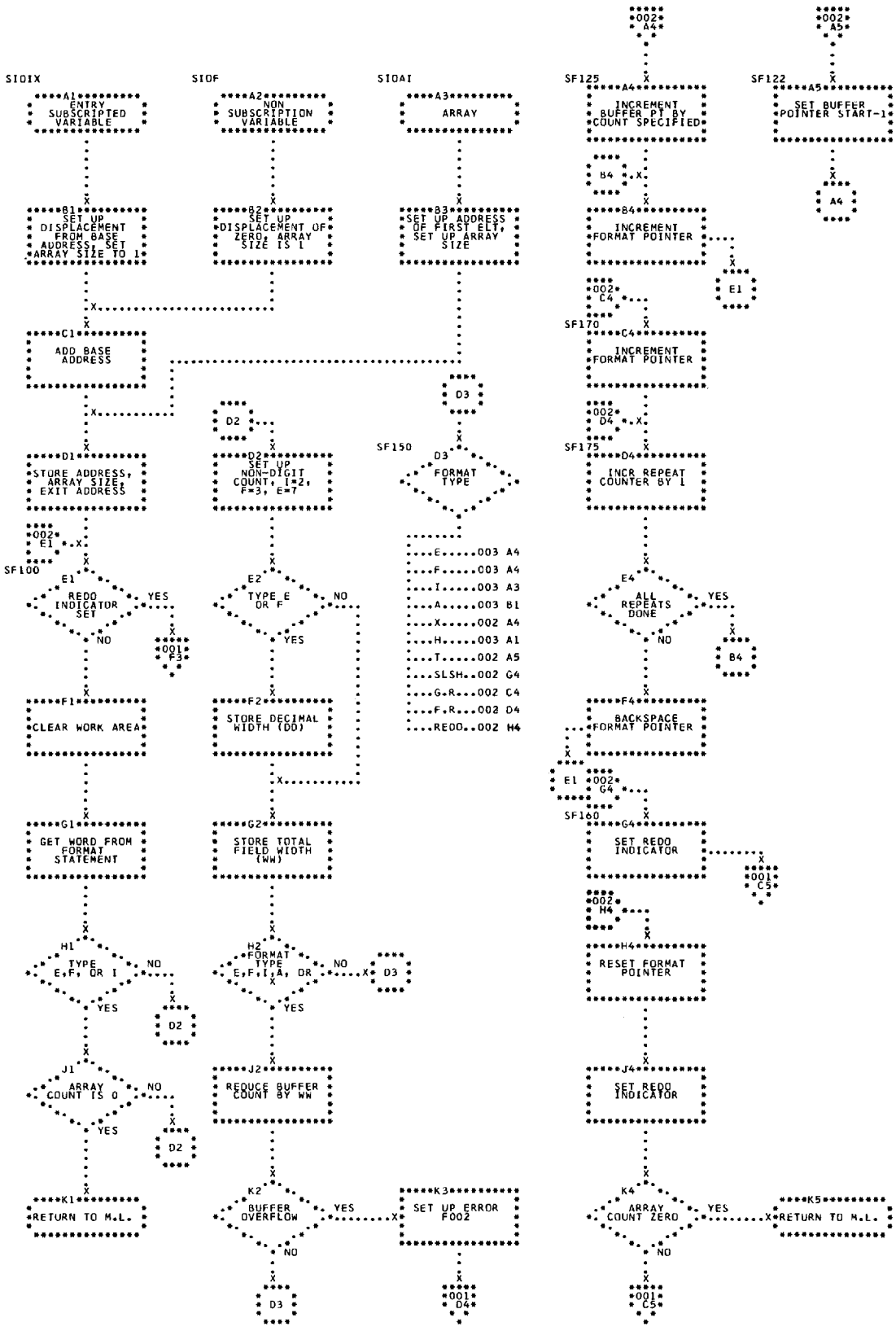
Flowchart SCA08. System Library, SCAT3 Interrupt Processing



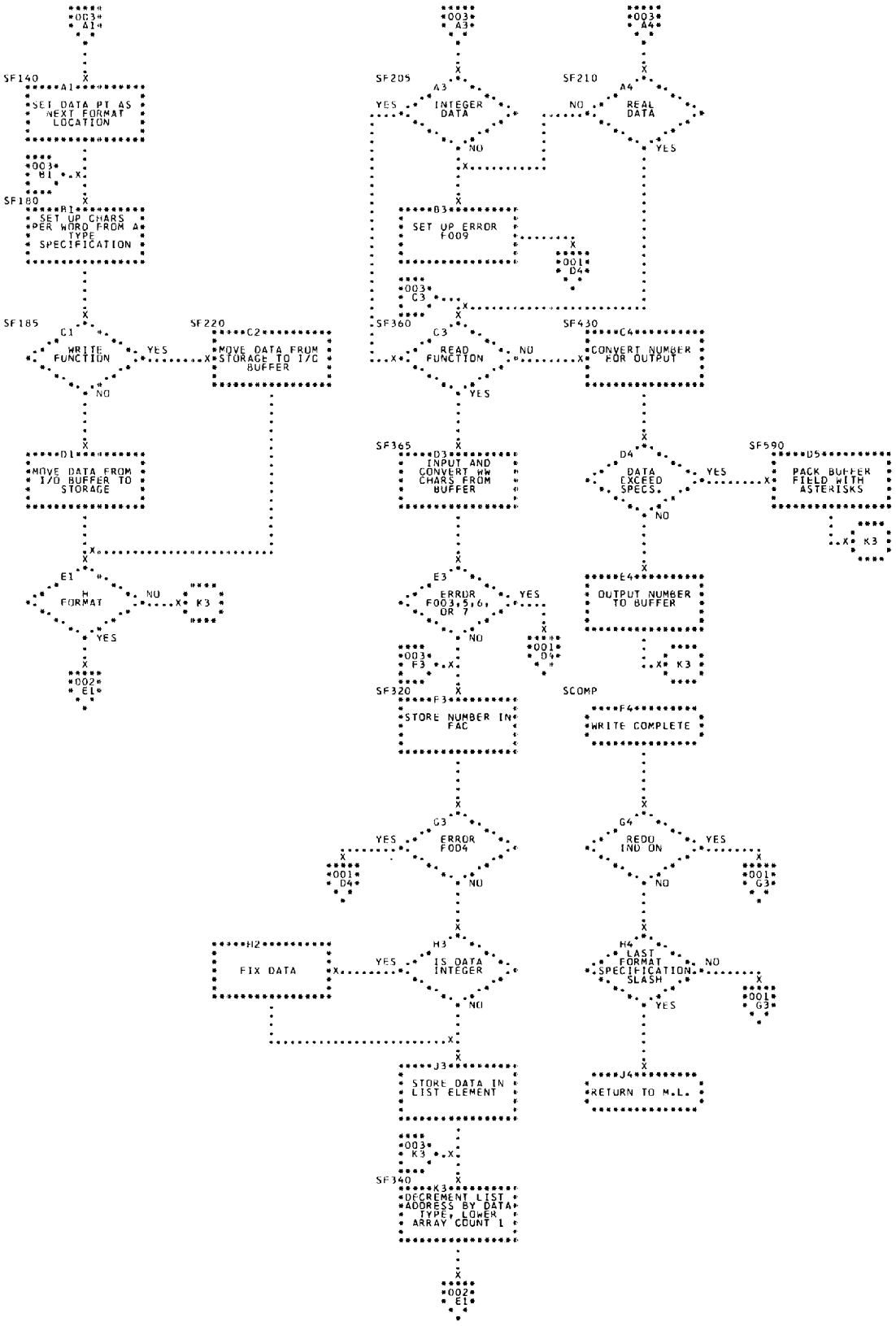
Flowchart SCA09. System Library, SCAT3 Interrupt Processing



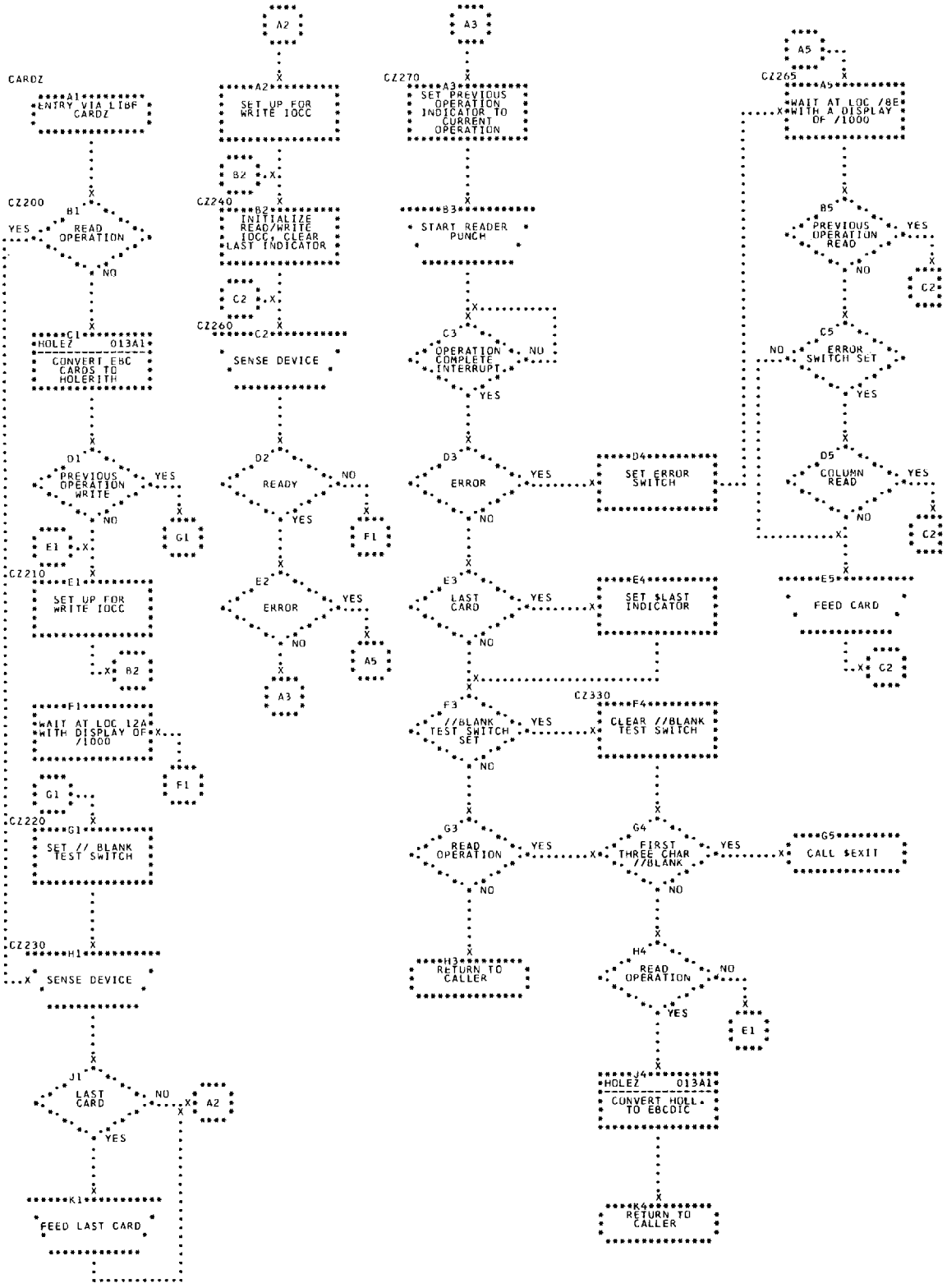
Flowchart FIO01. System Library, FORTRAN Non-disk I/O



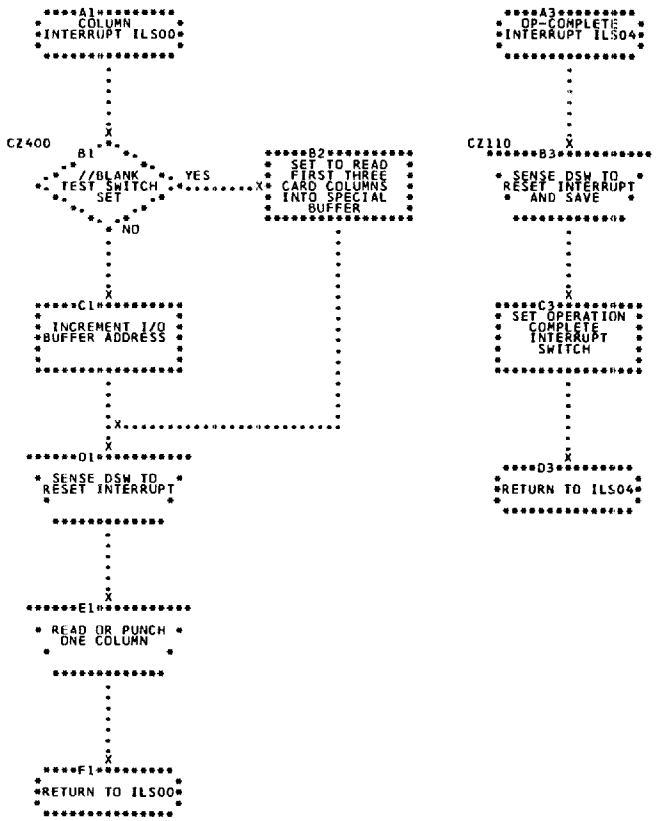
Flowchart FIO02. System Library, FORTRAN Non-disk I/O



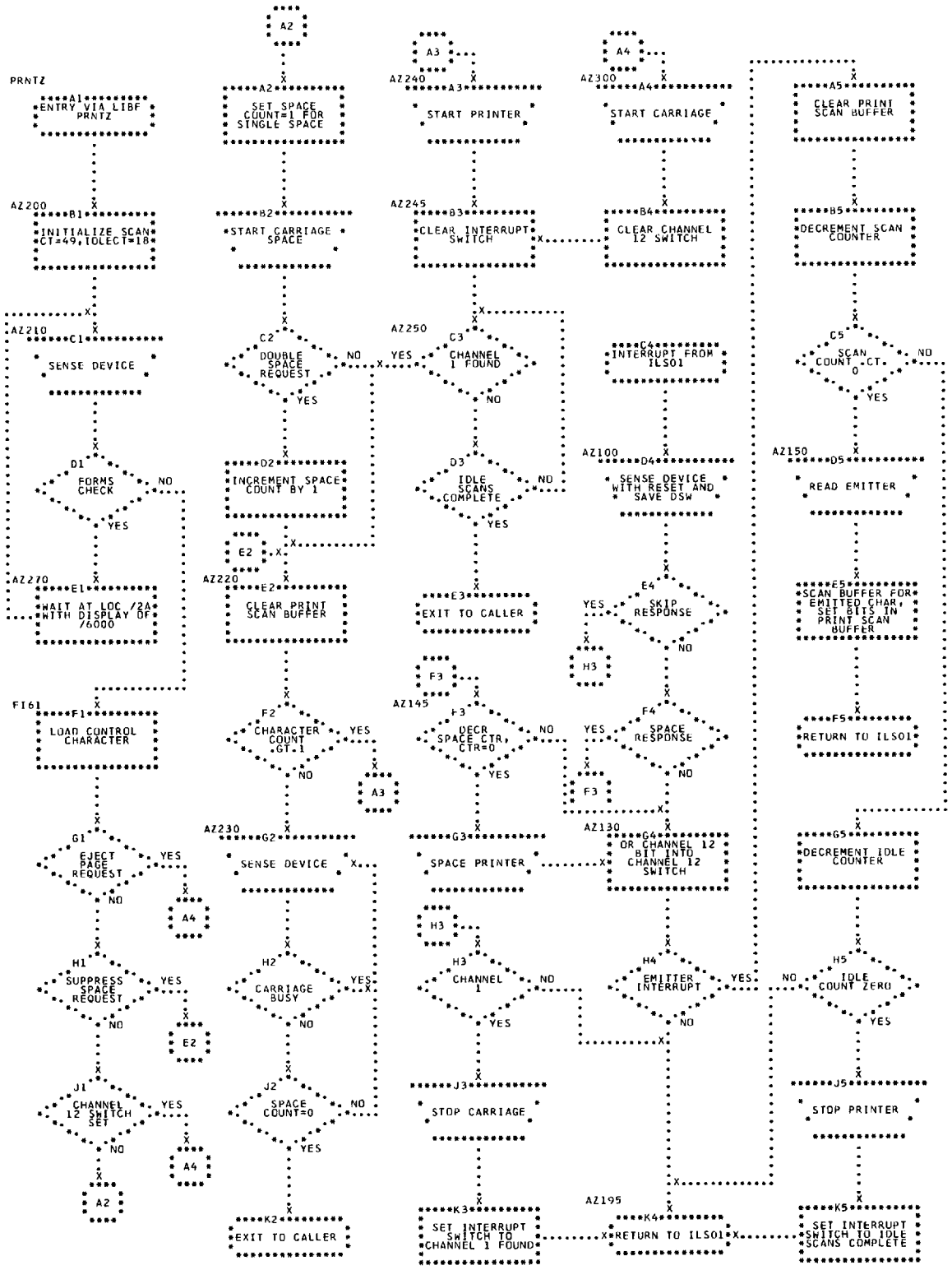
Flowchart FIO03. System Library, FORTRAN Non-disk I/O



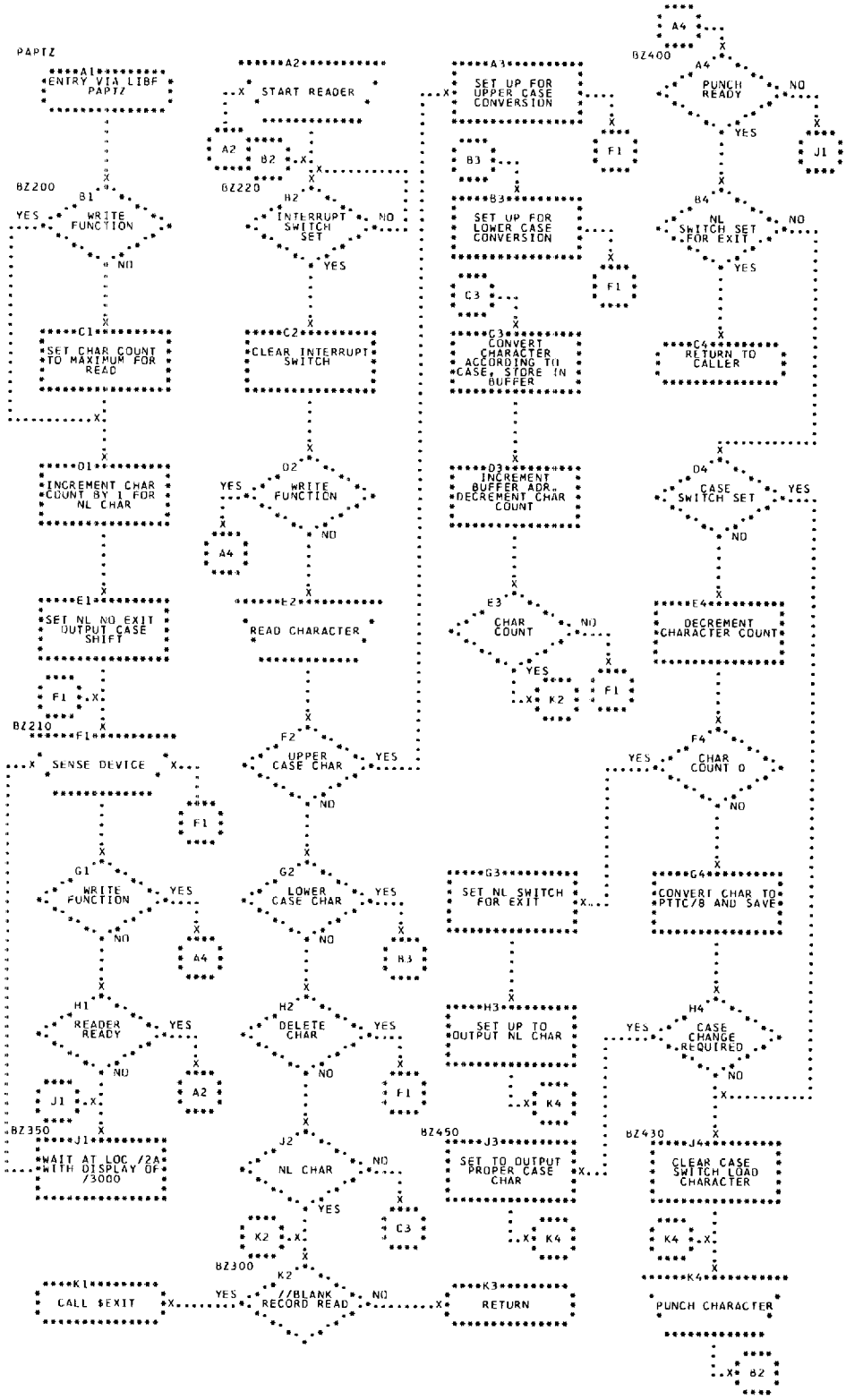
Flowchart FIO04. System Library, CARDZ



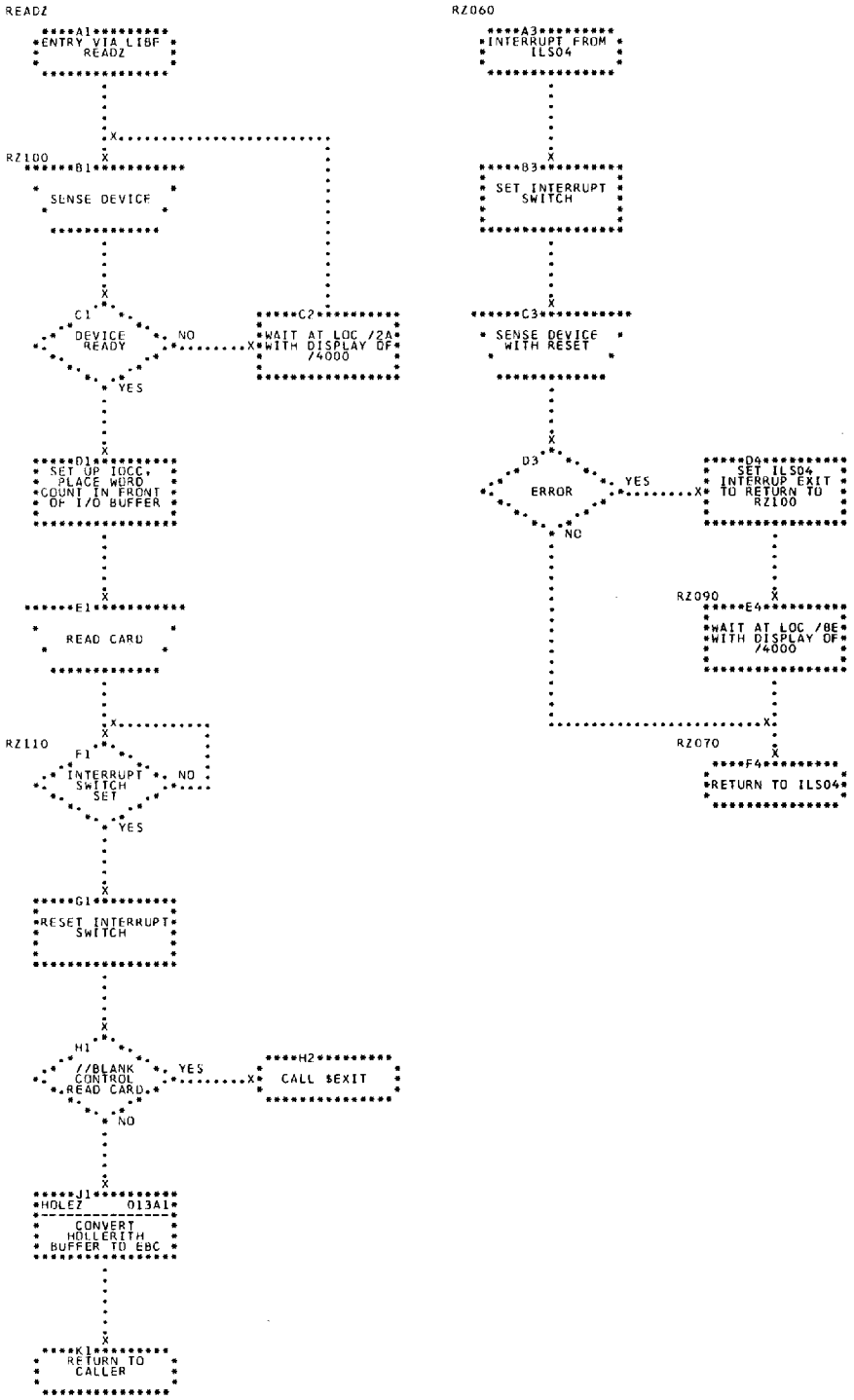
Flowchart FIO05. System Library, CARDZ



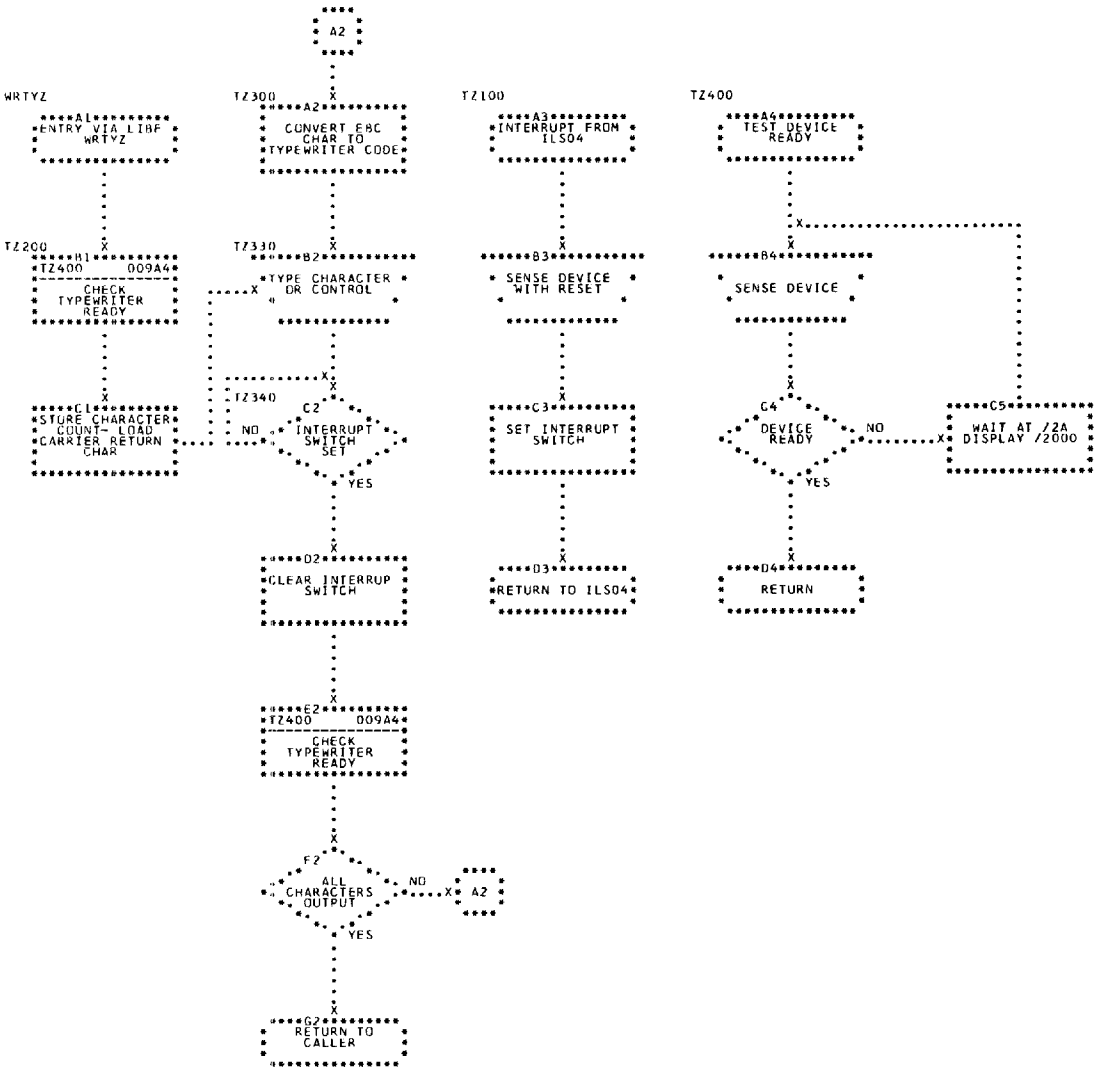
Flowchart FIO06. System Library, PRNTZ



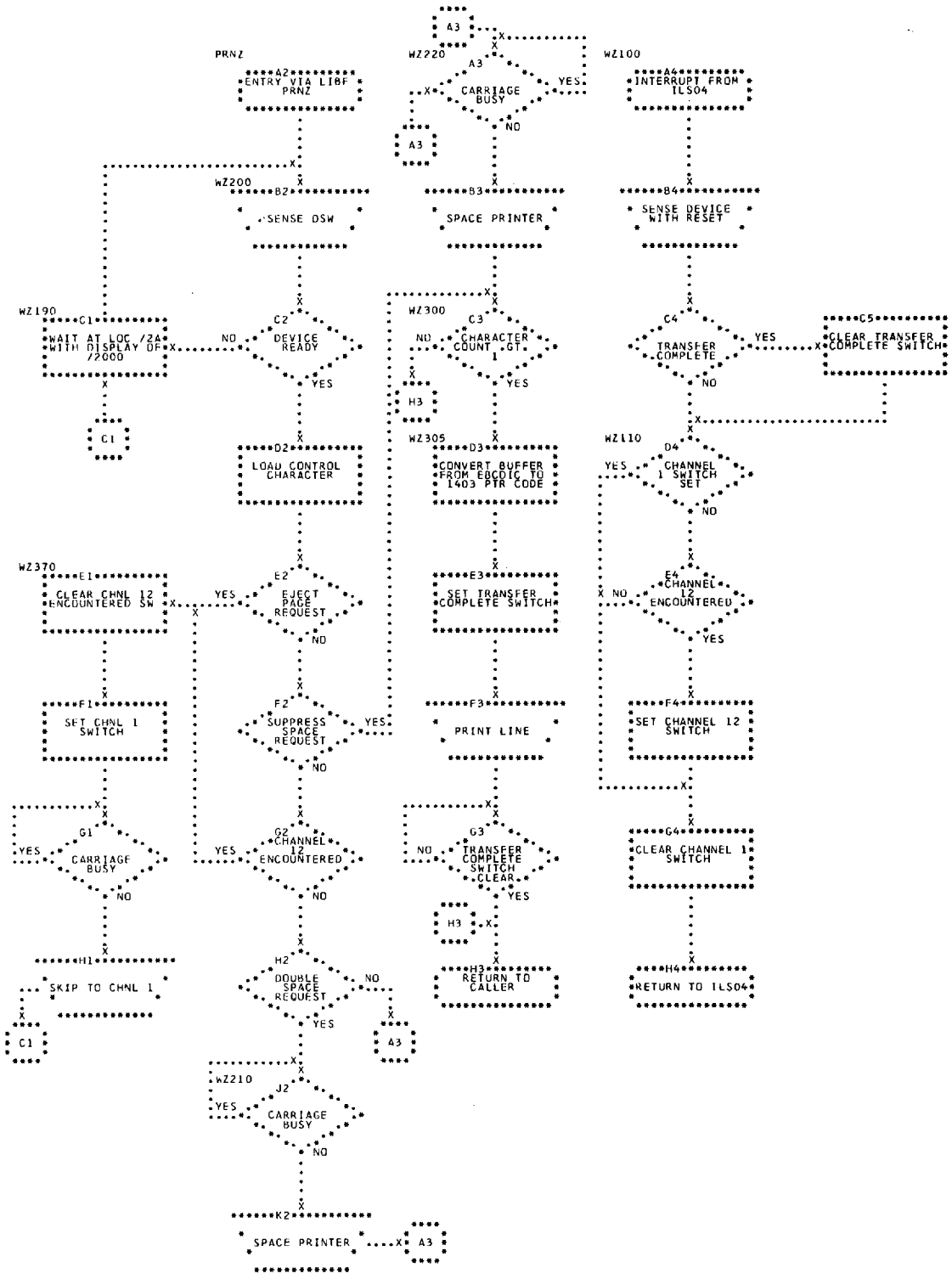
Flowchart FIO07. System Library, PAPTZ



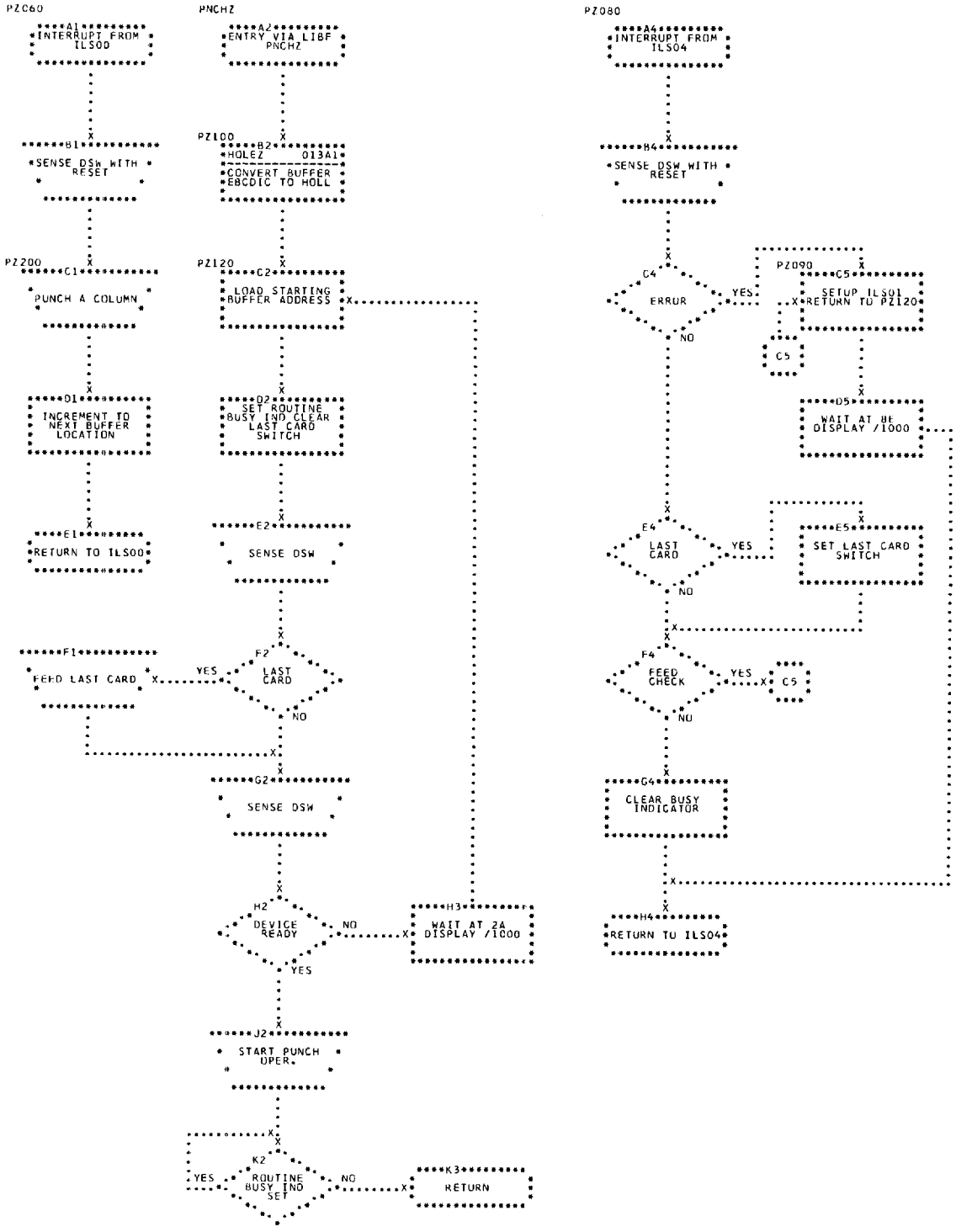
Flowchart FIO08. System Library, READZ



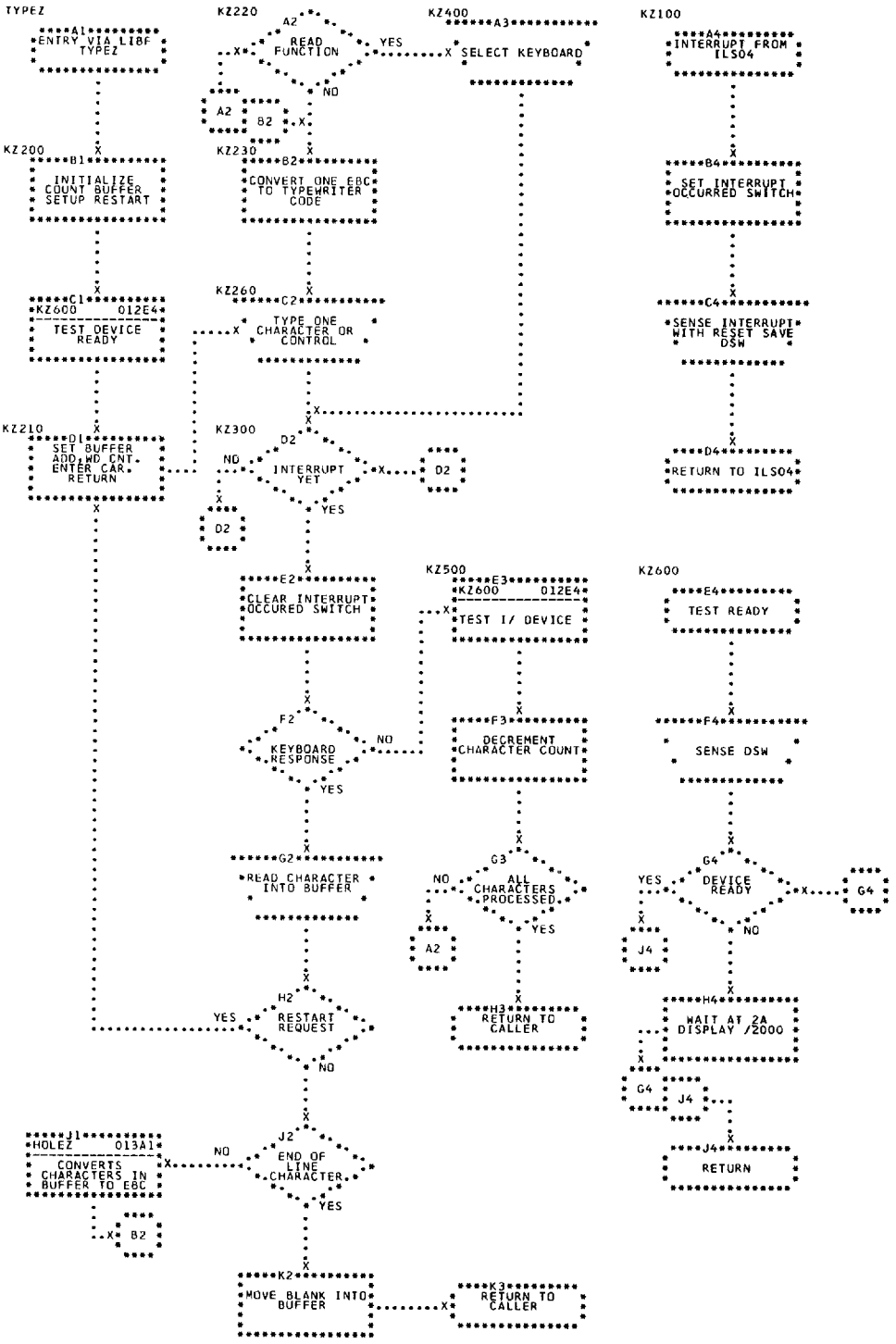
Flowchart FIO09. System Library, WRTYZ



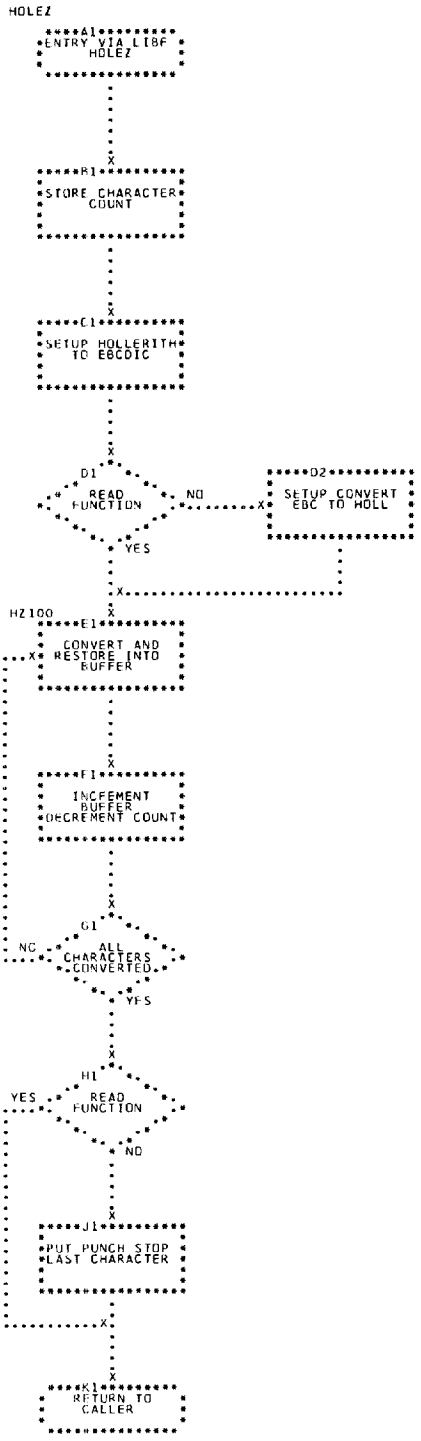
Flowchart FIO10. System Library, PRNZ



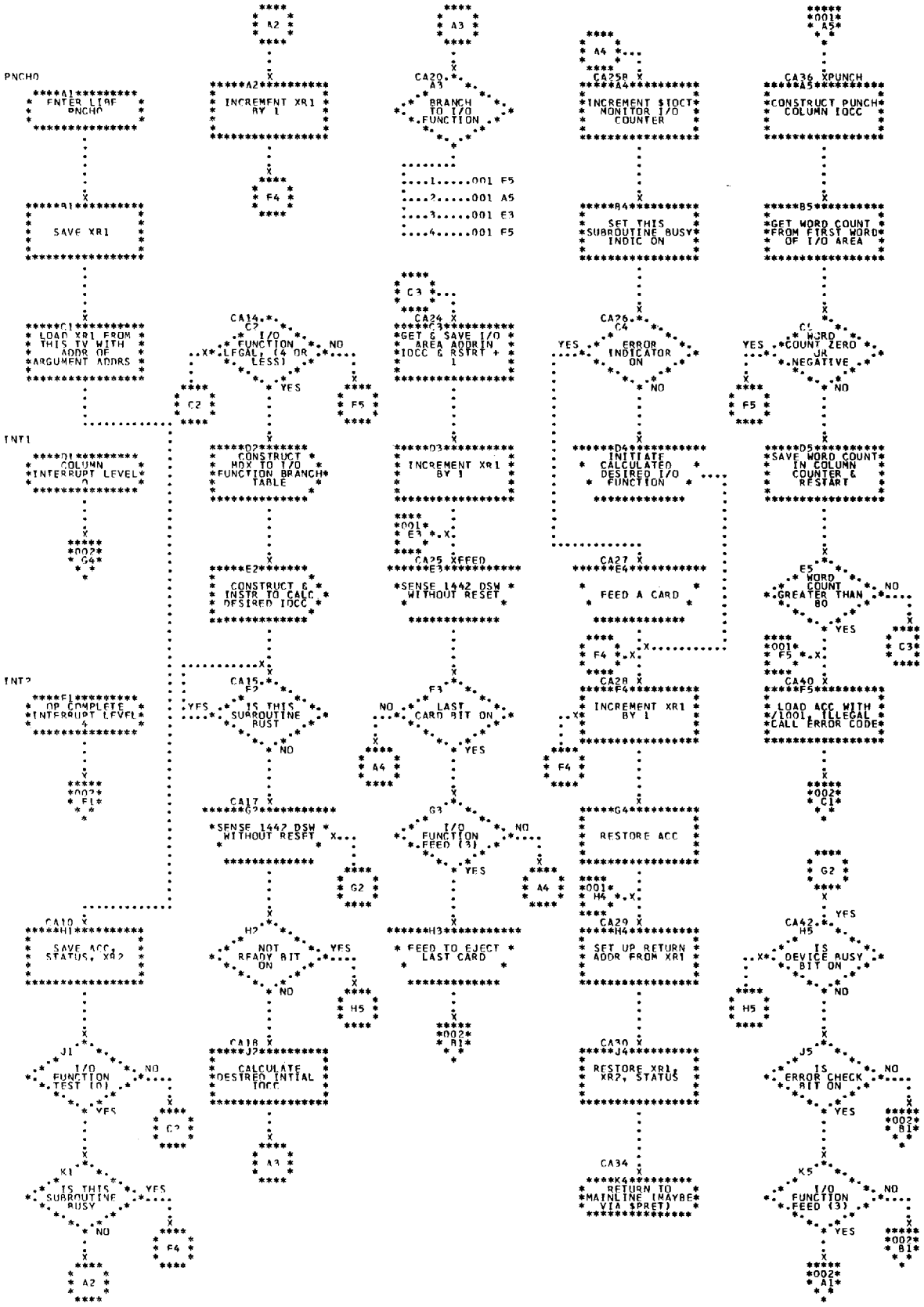
Flowchart FIO11. System Library, PNCHZ



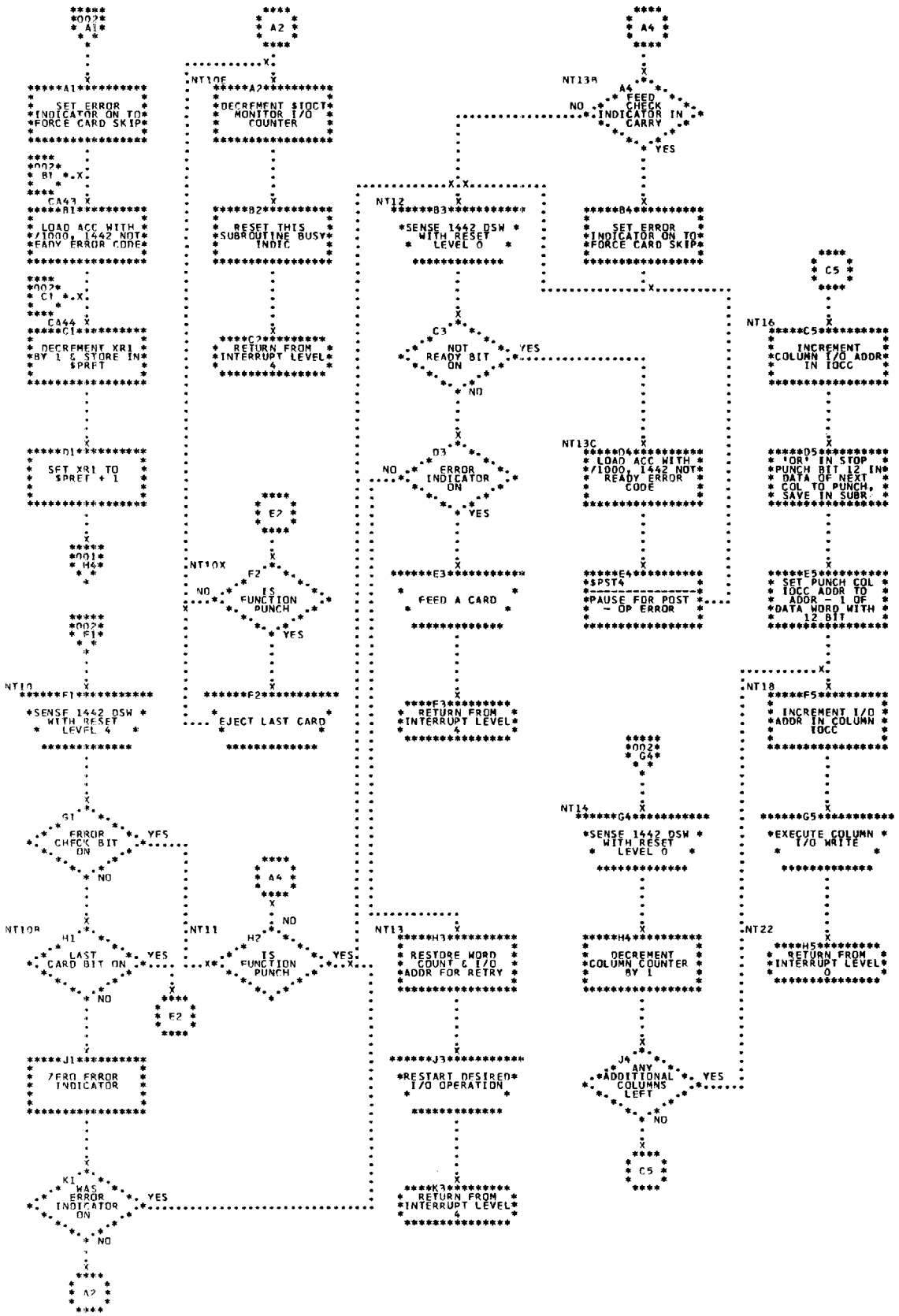
Flowchart FIO12. System Library, TYPEZ



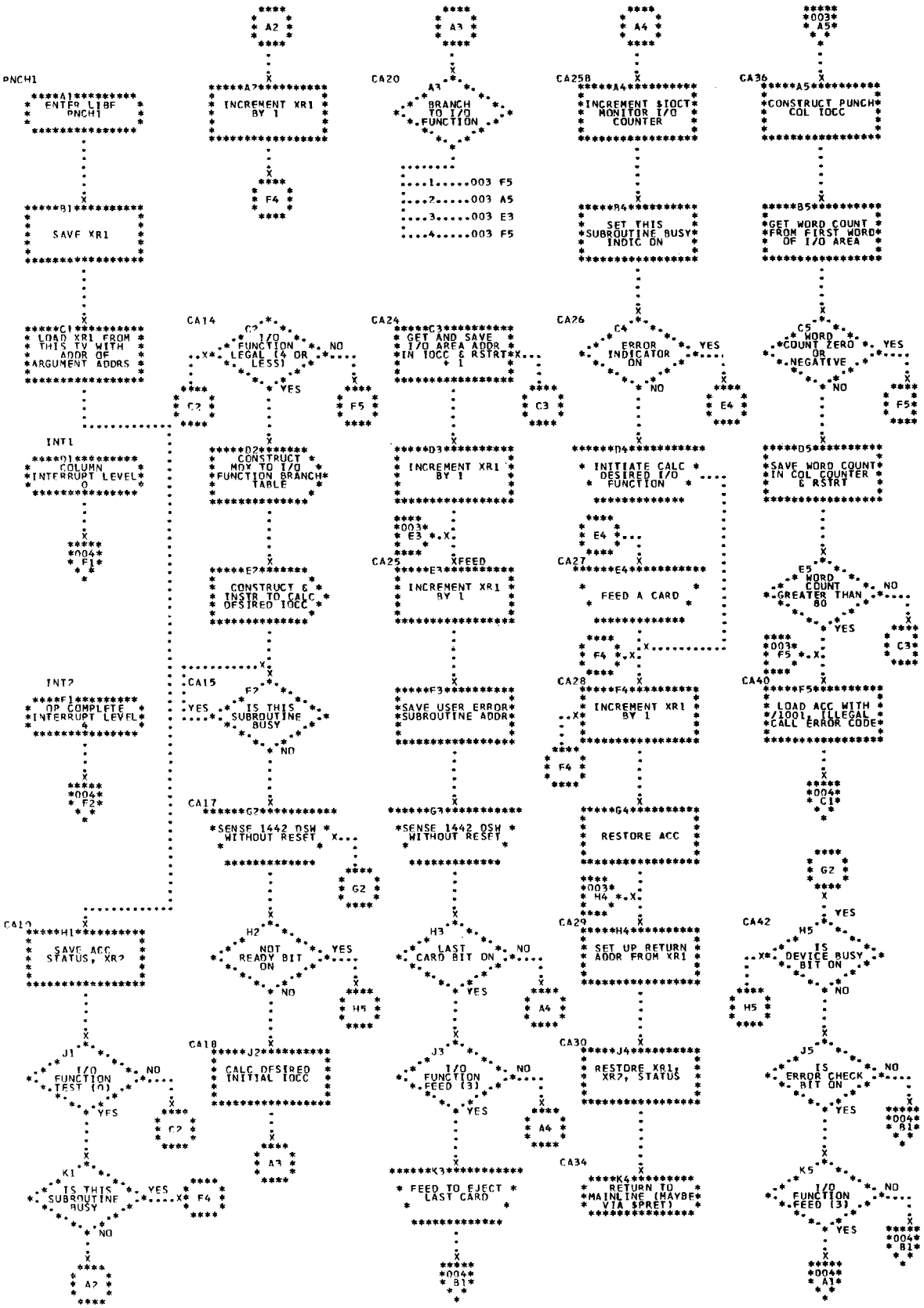
Flowchart FIO13. System Library, HOLEZ



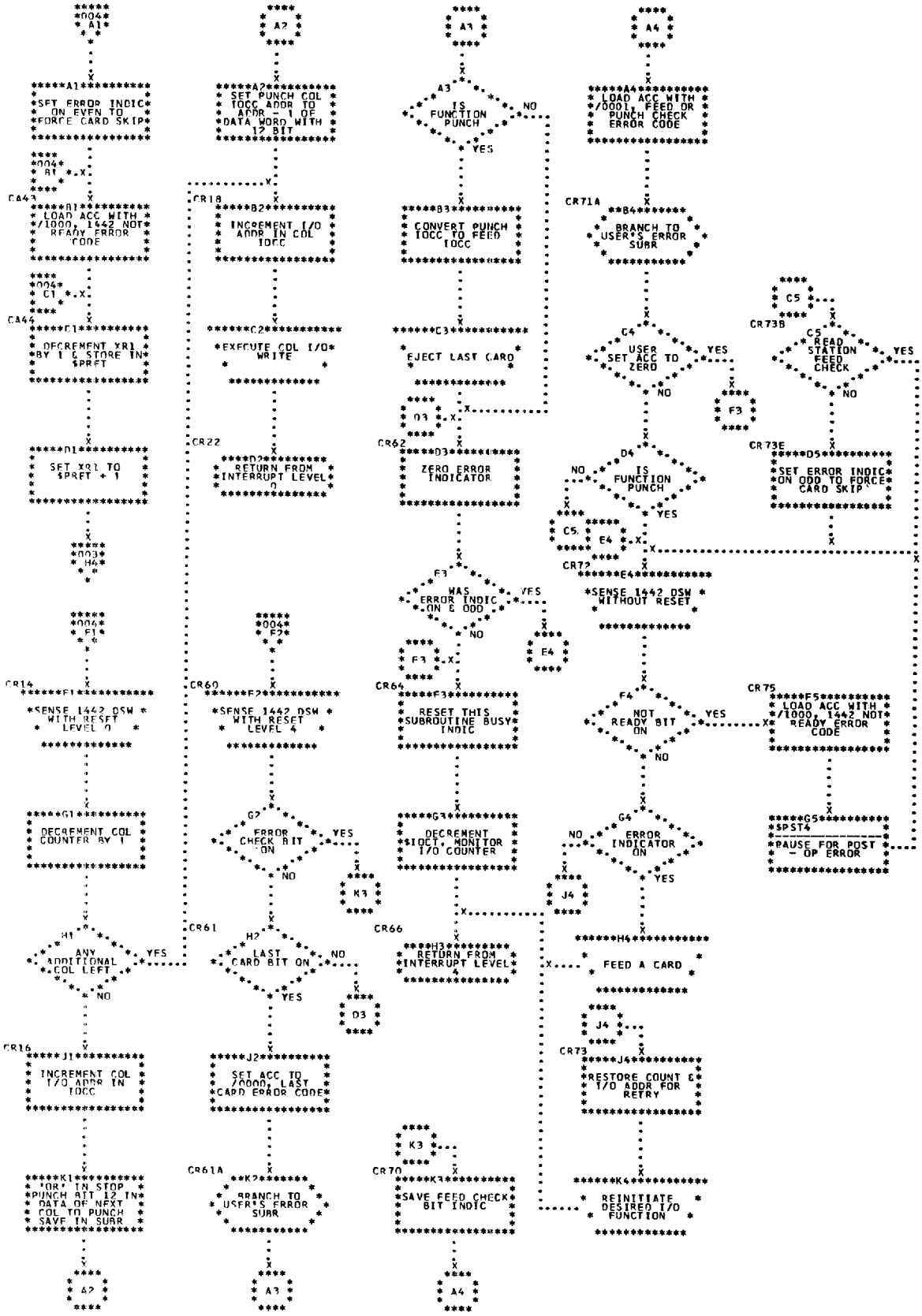
Flowchart FIO14. System Library, PNCH0



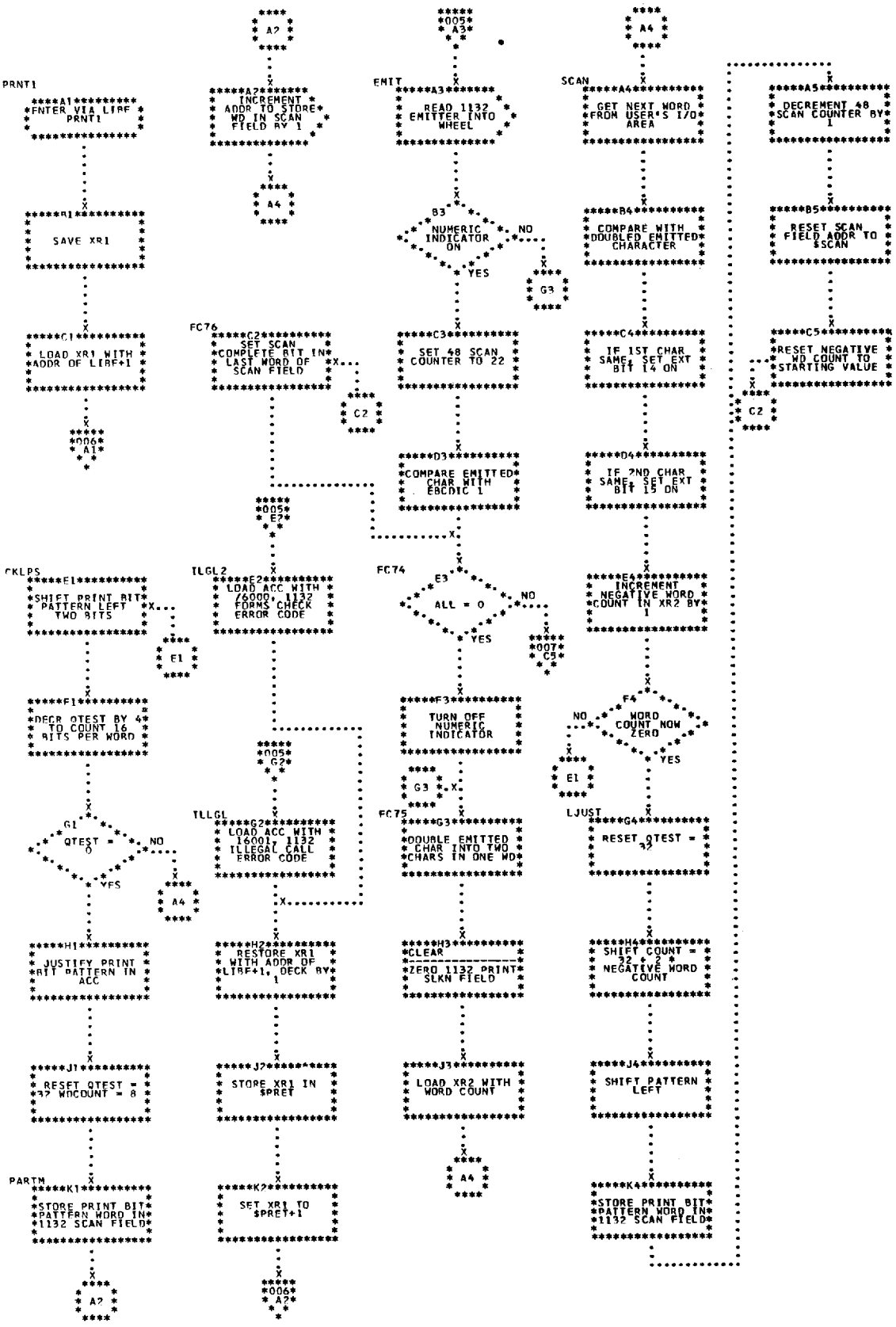
Flowchart FIO15. System Library, PNCH0



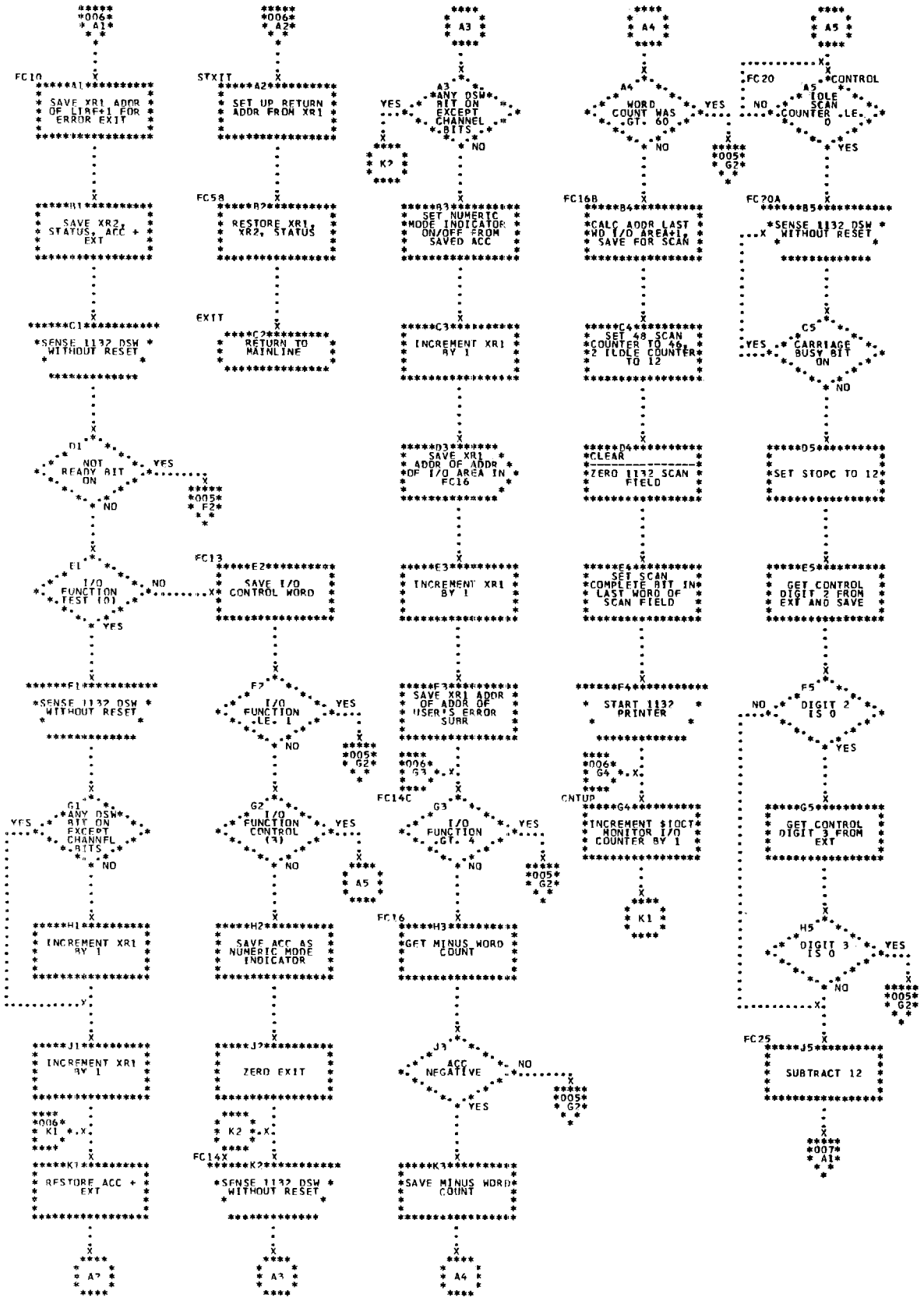
Flowchart FIO16. System Library, PNCH0



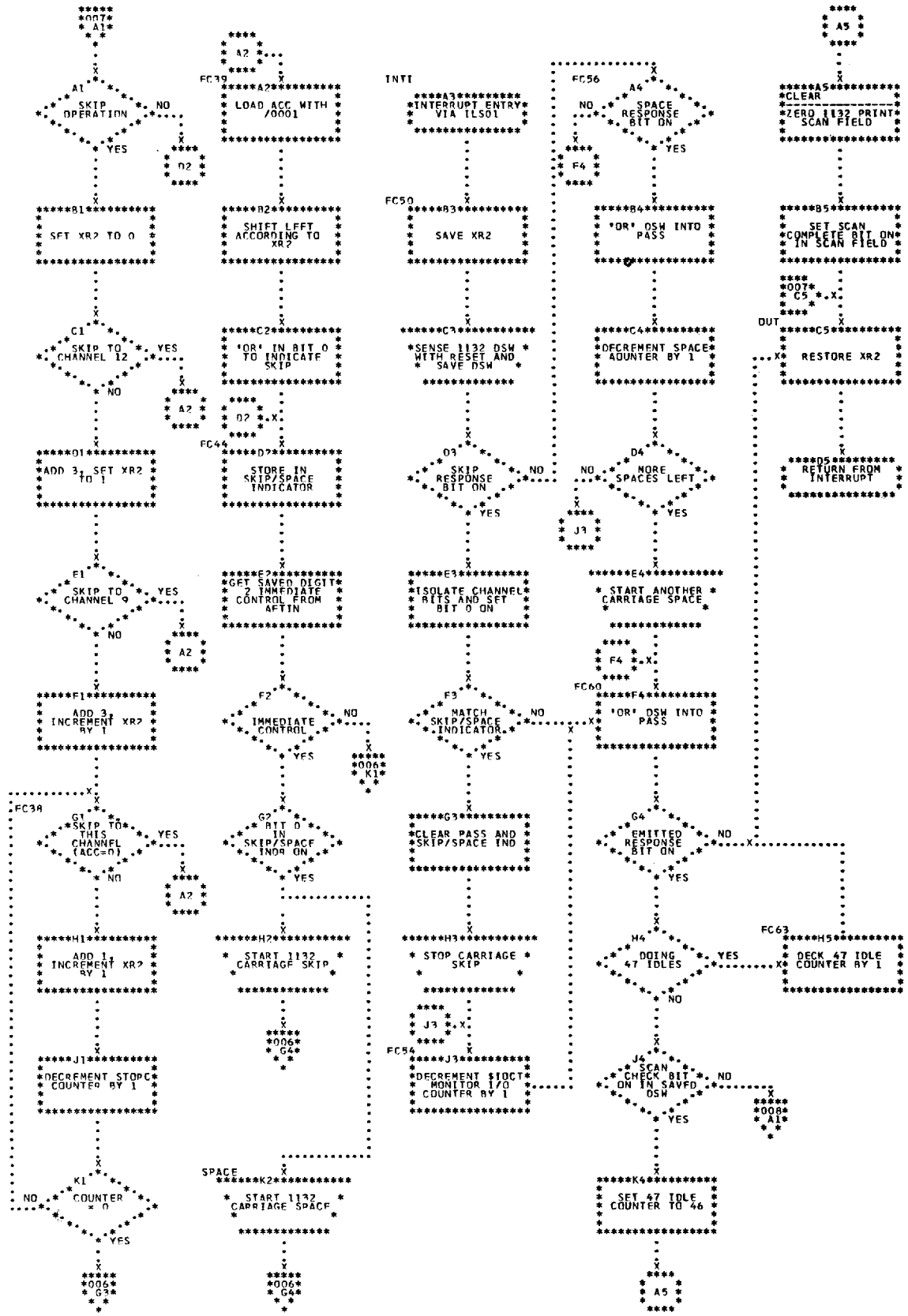
Flowchart FIO17. System Library, PNCH1



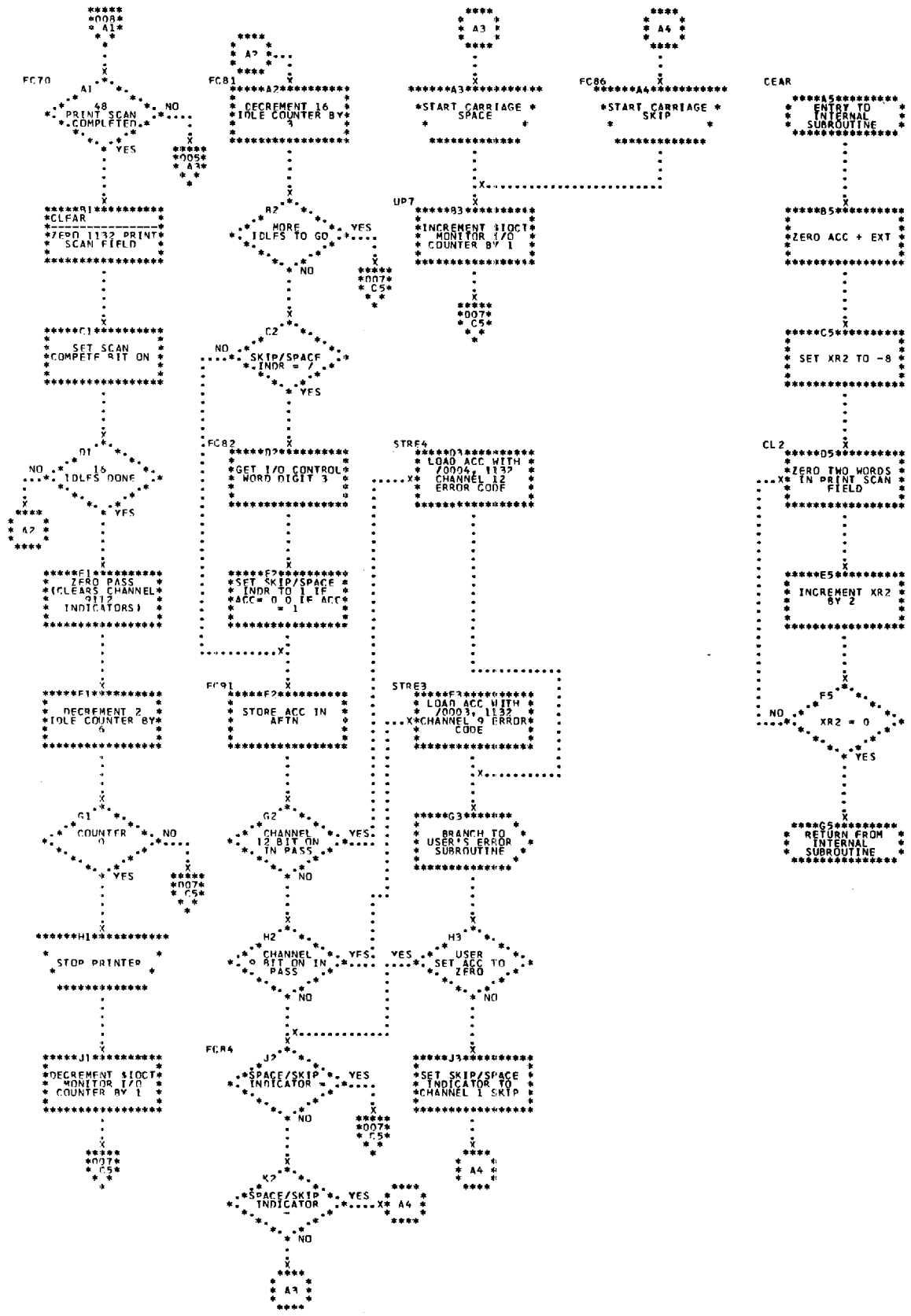
Flowchart FIO18. System Library, PRNT1



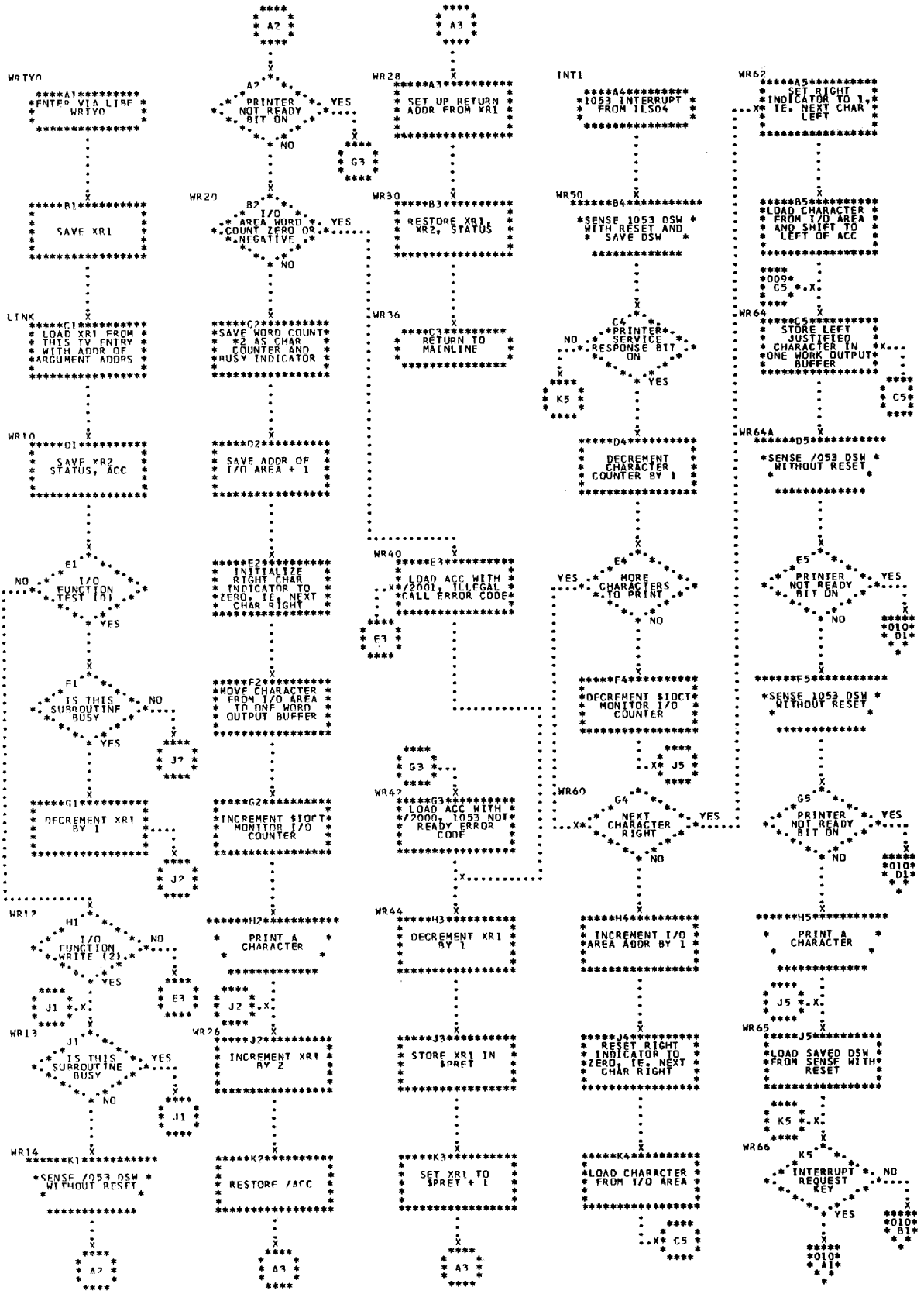
Flowchart FIO19. System Library, PRNT1



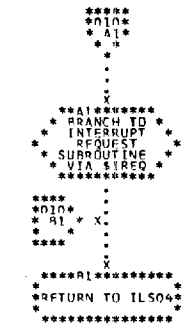
Flowchart FIO20. System Library, PRNT1

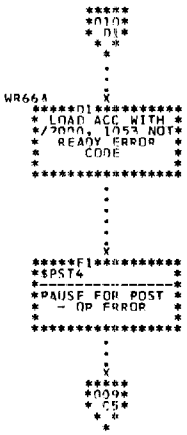


Flowchart FIO21. System Library, PRNT1

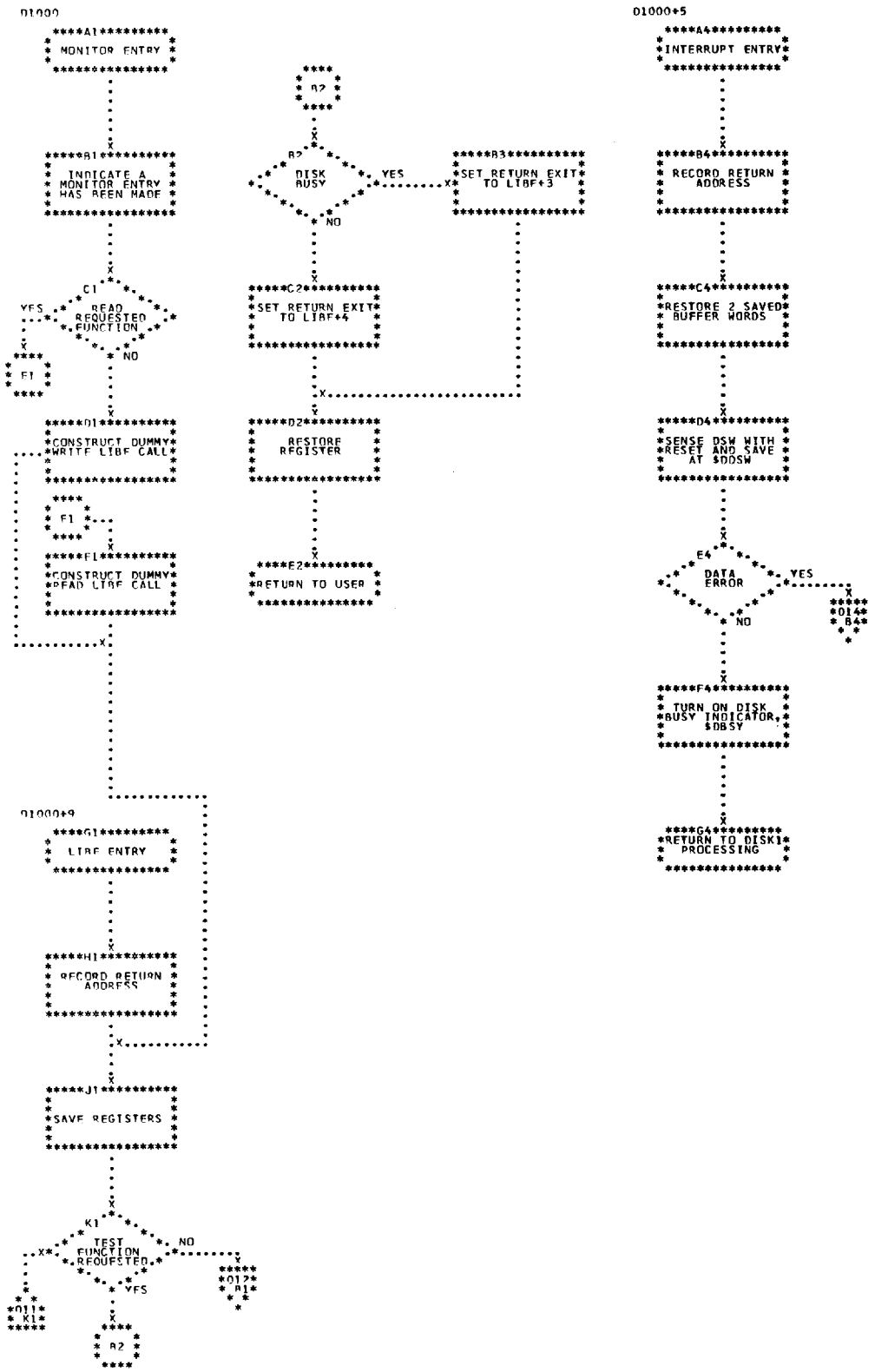


Flowcharts FIO22. System Library, WRTYO

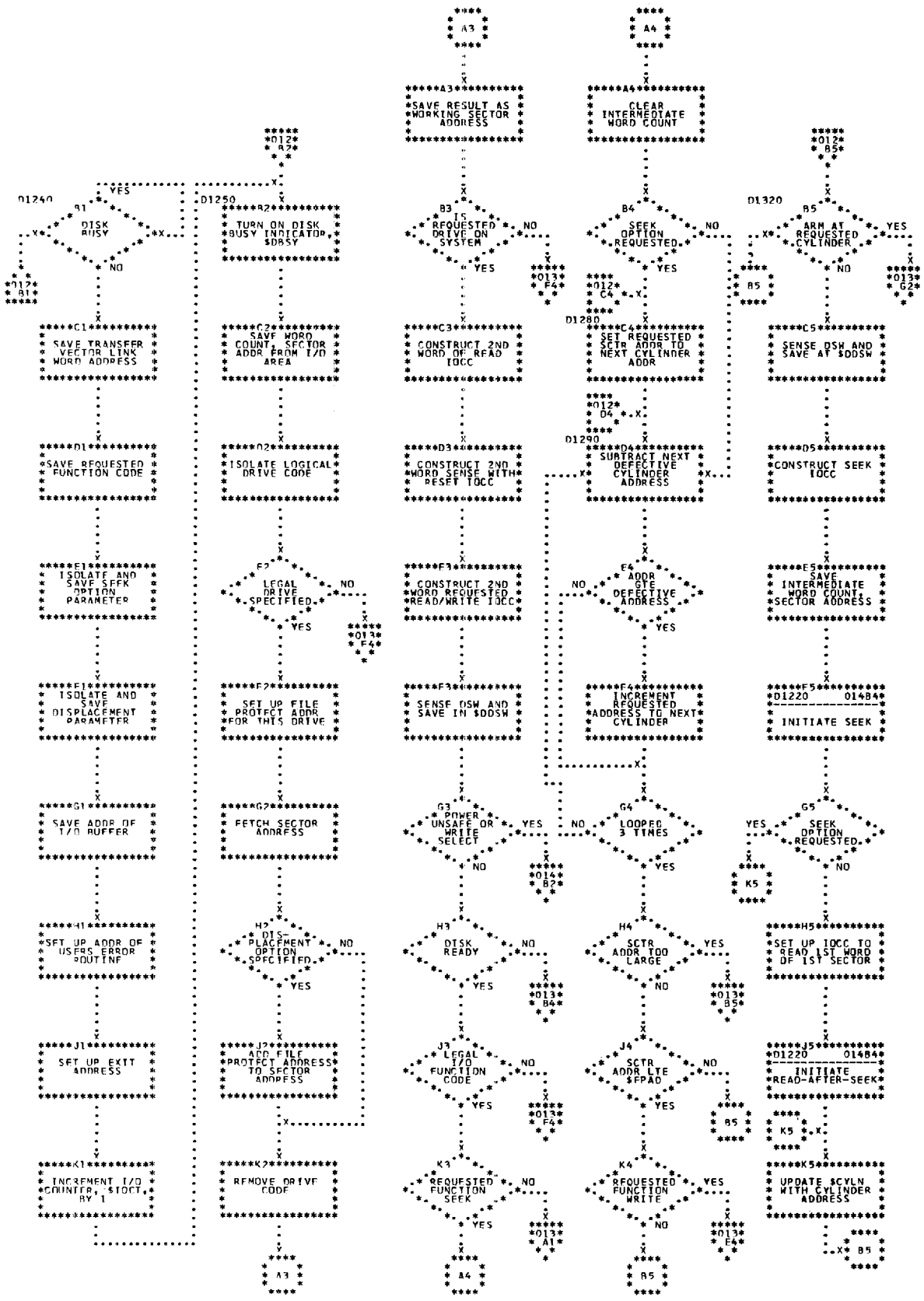




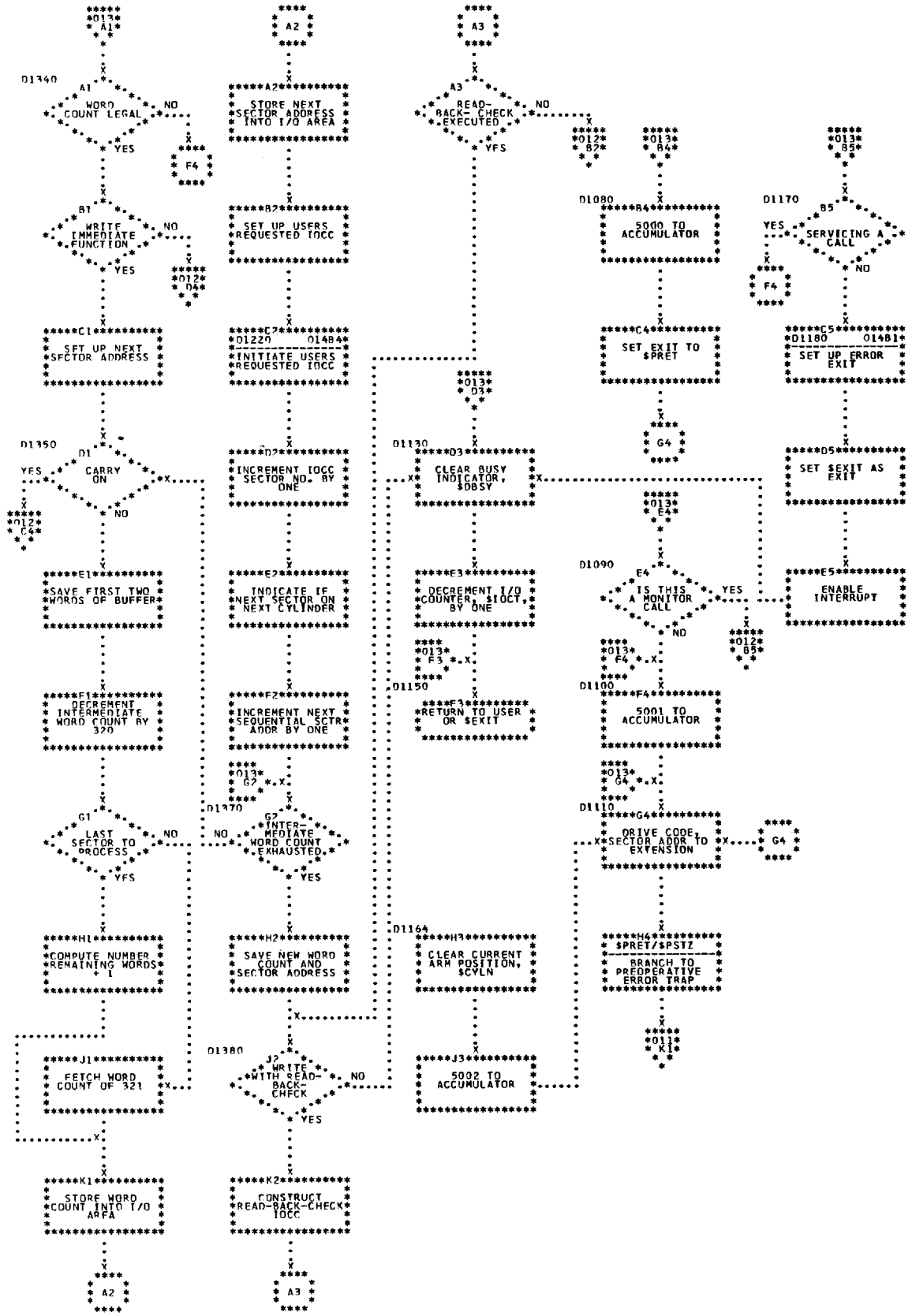
Flowchart FIO23. System Library, WRTY0



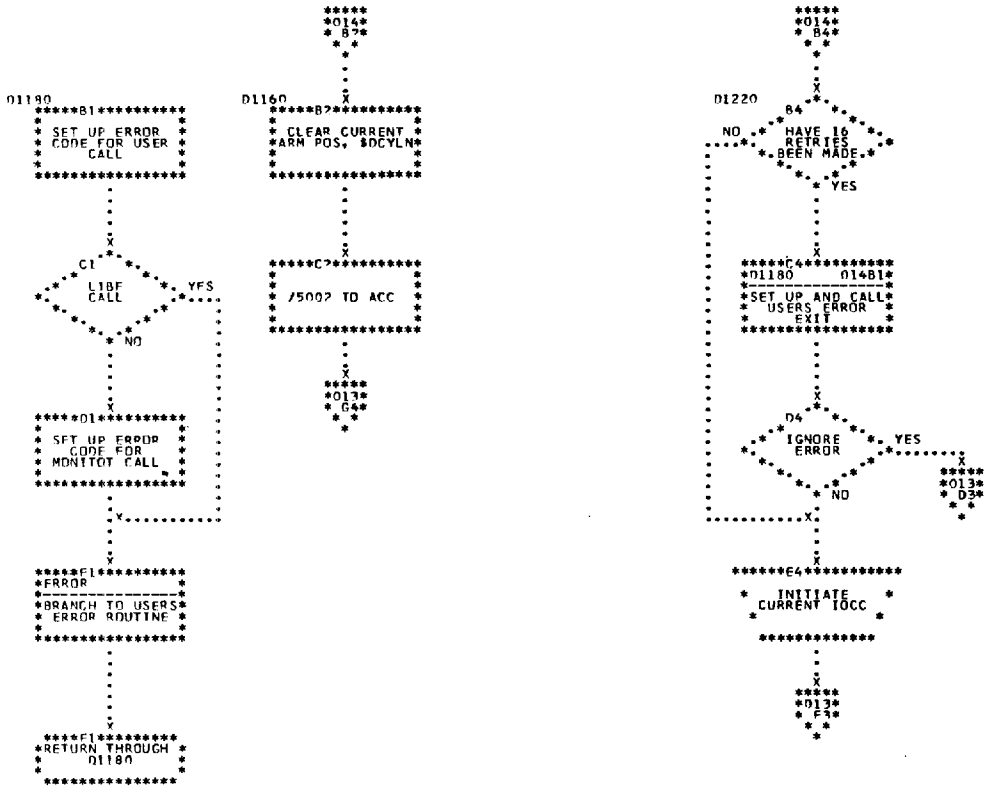
Flowchart FIO24. System Library, DISK1



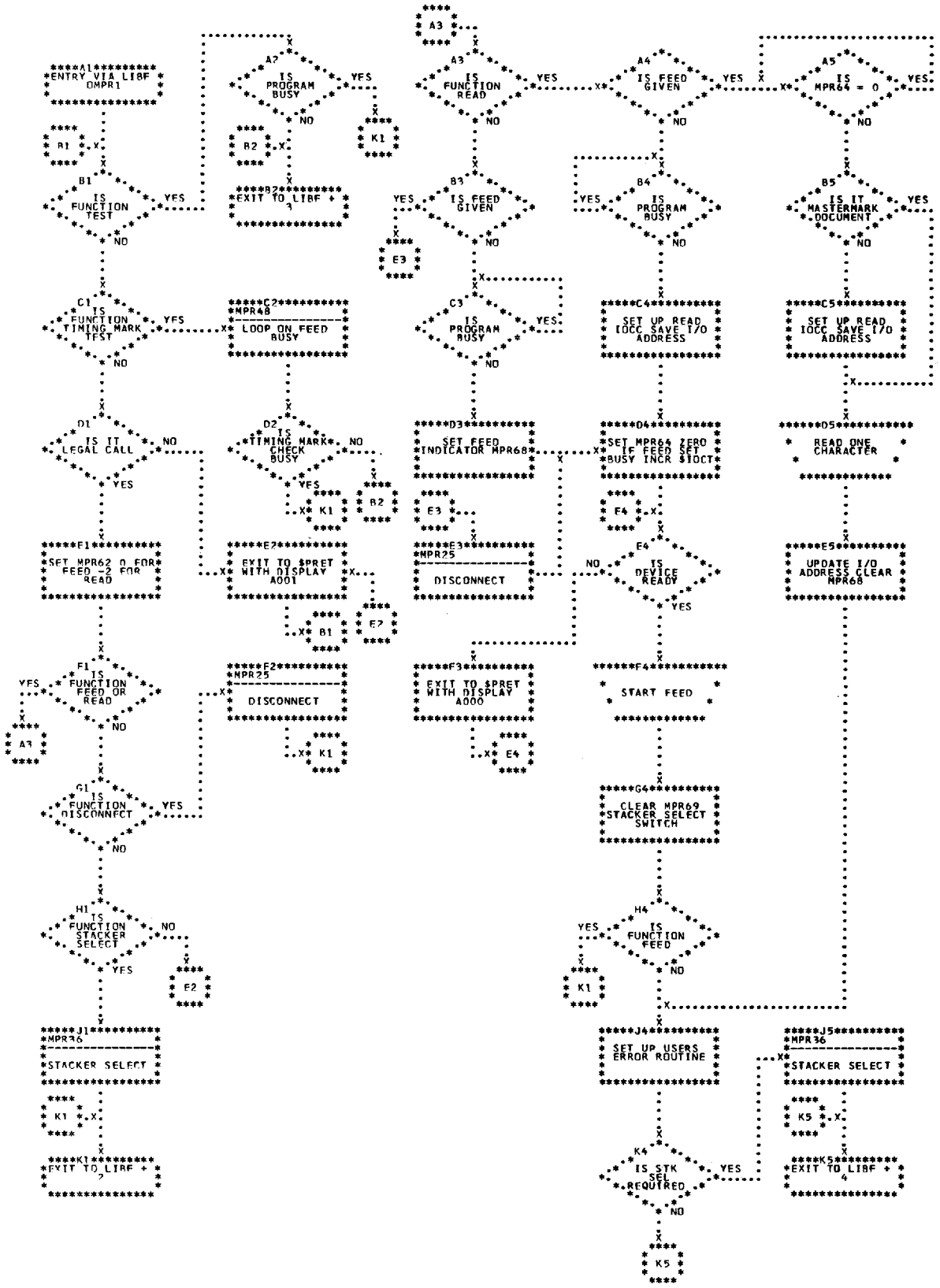
Flowchart FIO25. System Library, DISK1



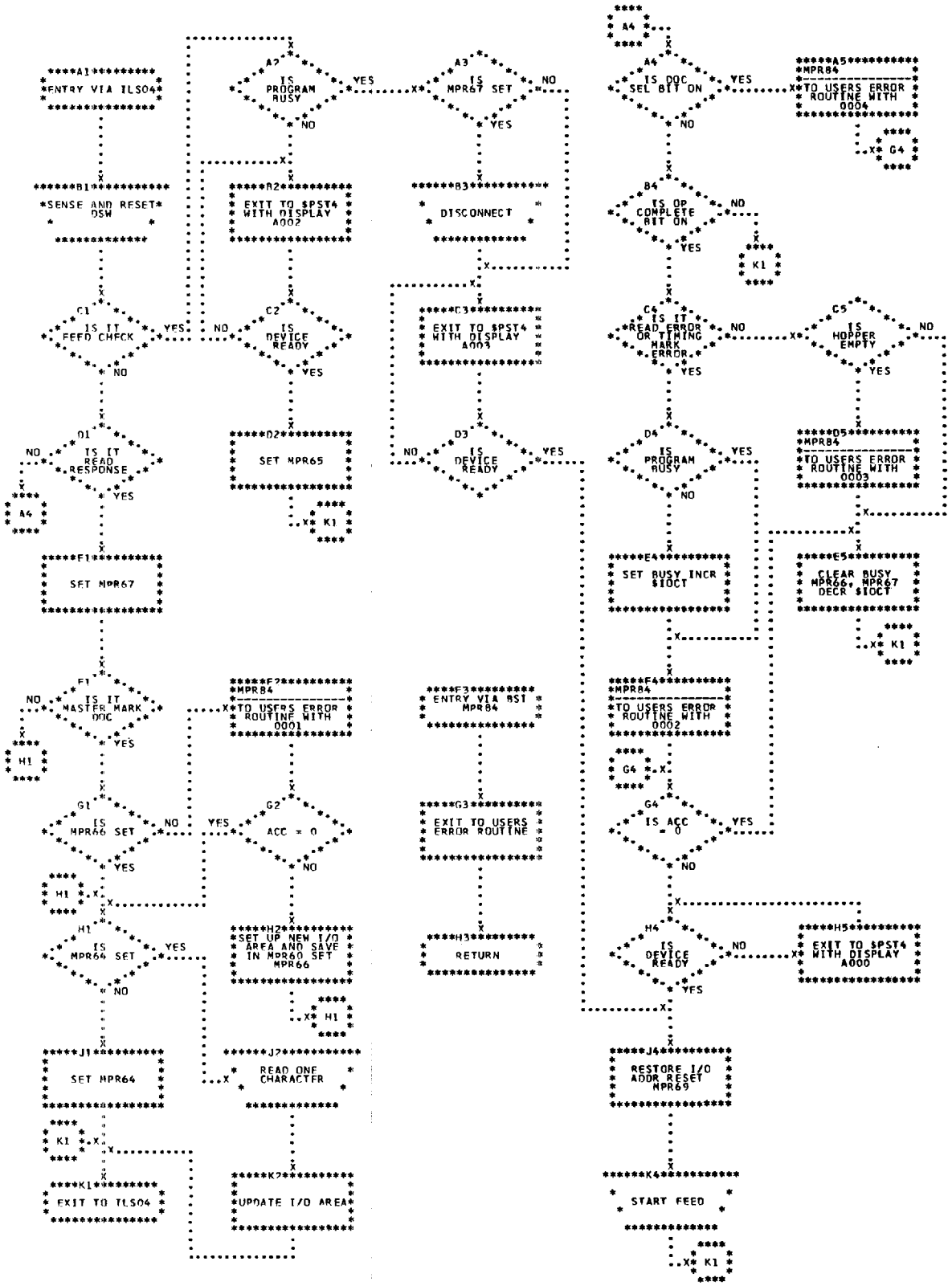
Flowchart FIO26. System Library, DISK1



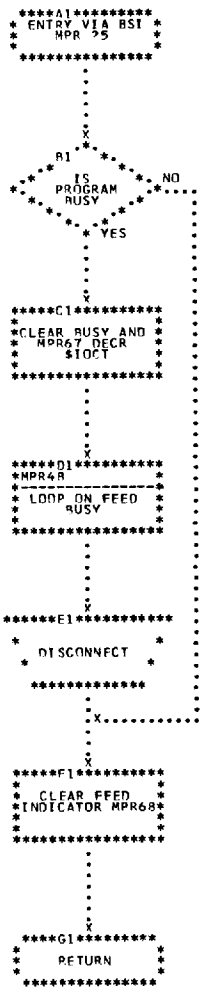
Flowchart F1027. System Library, DISK1

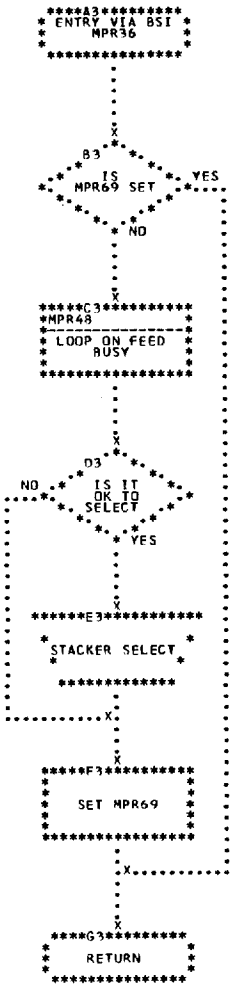


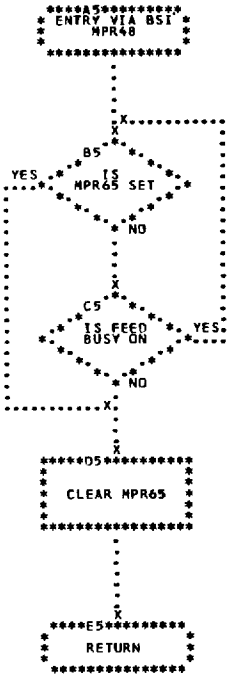
Flowchart FIO28. System Library, OMPR1 -CALL



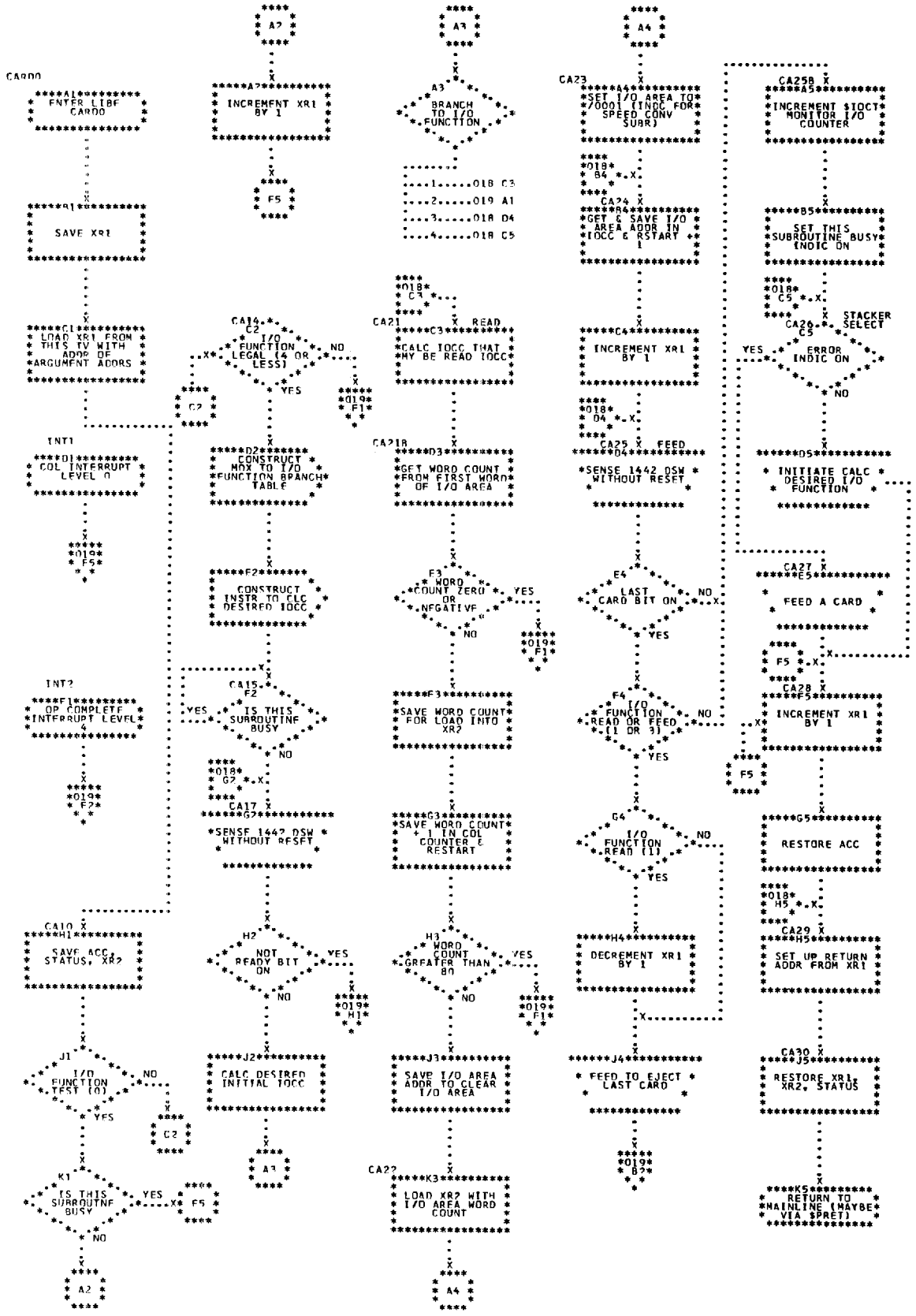
Flowchart FIO29. System Library, OMPR1-INTERRUPT



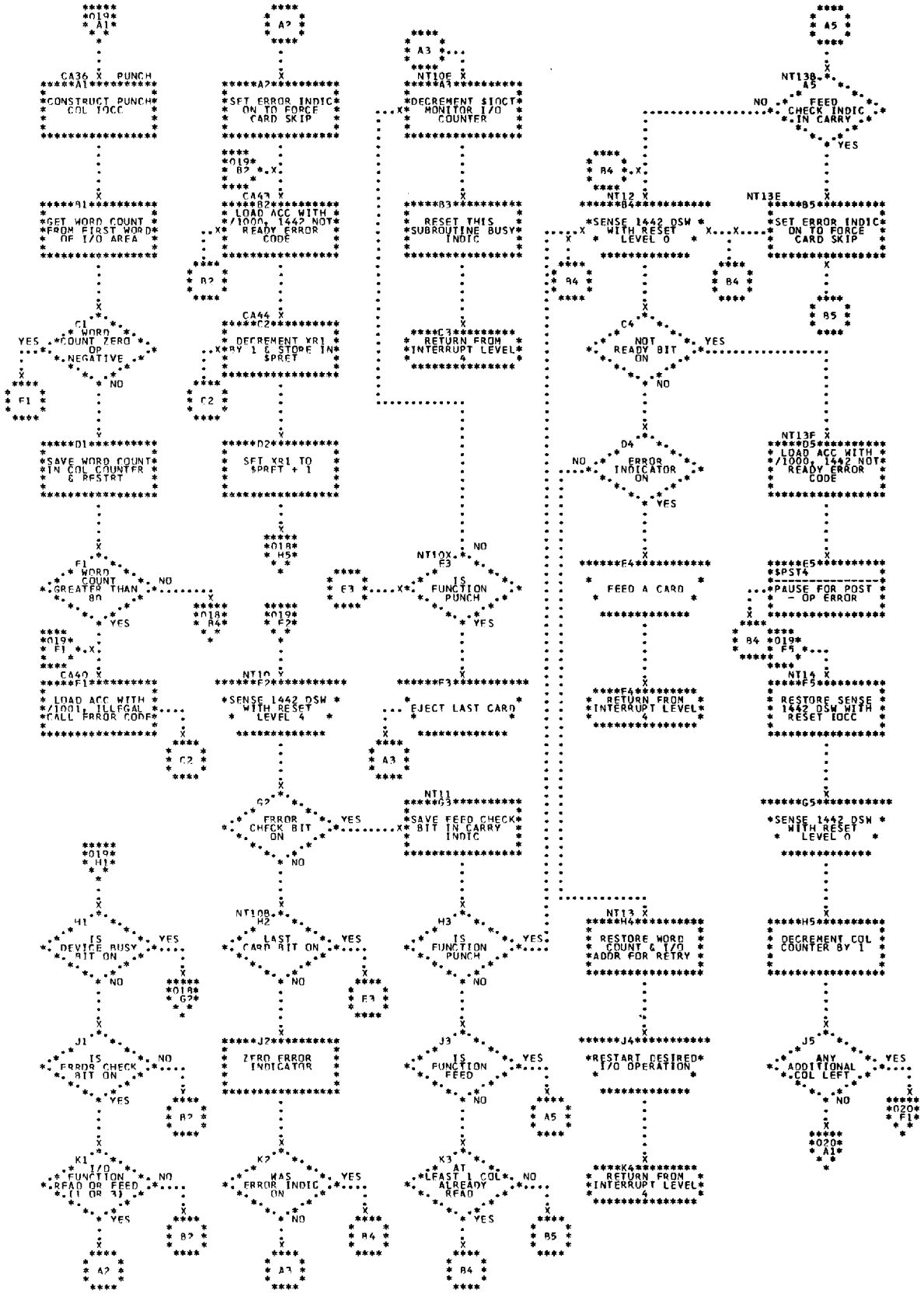




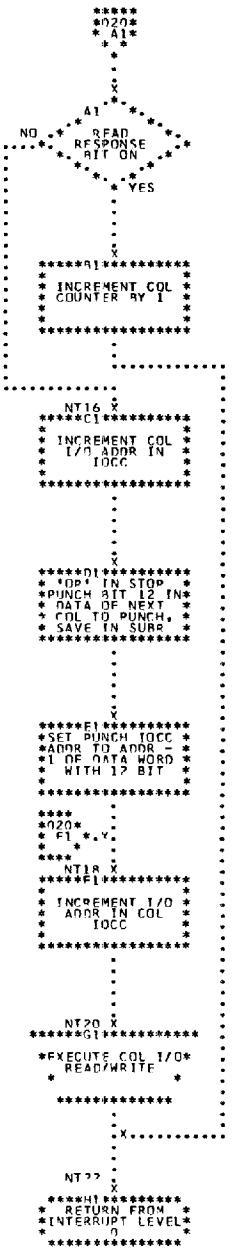
Flowchart FIO30. System Library, OMPR1



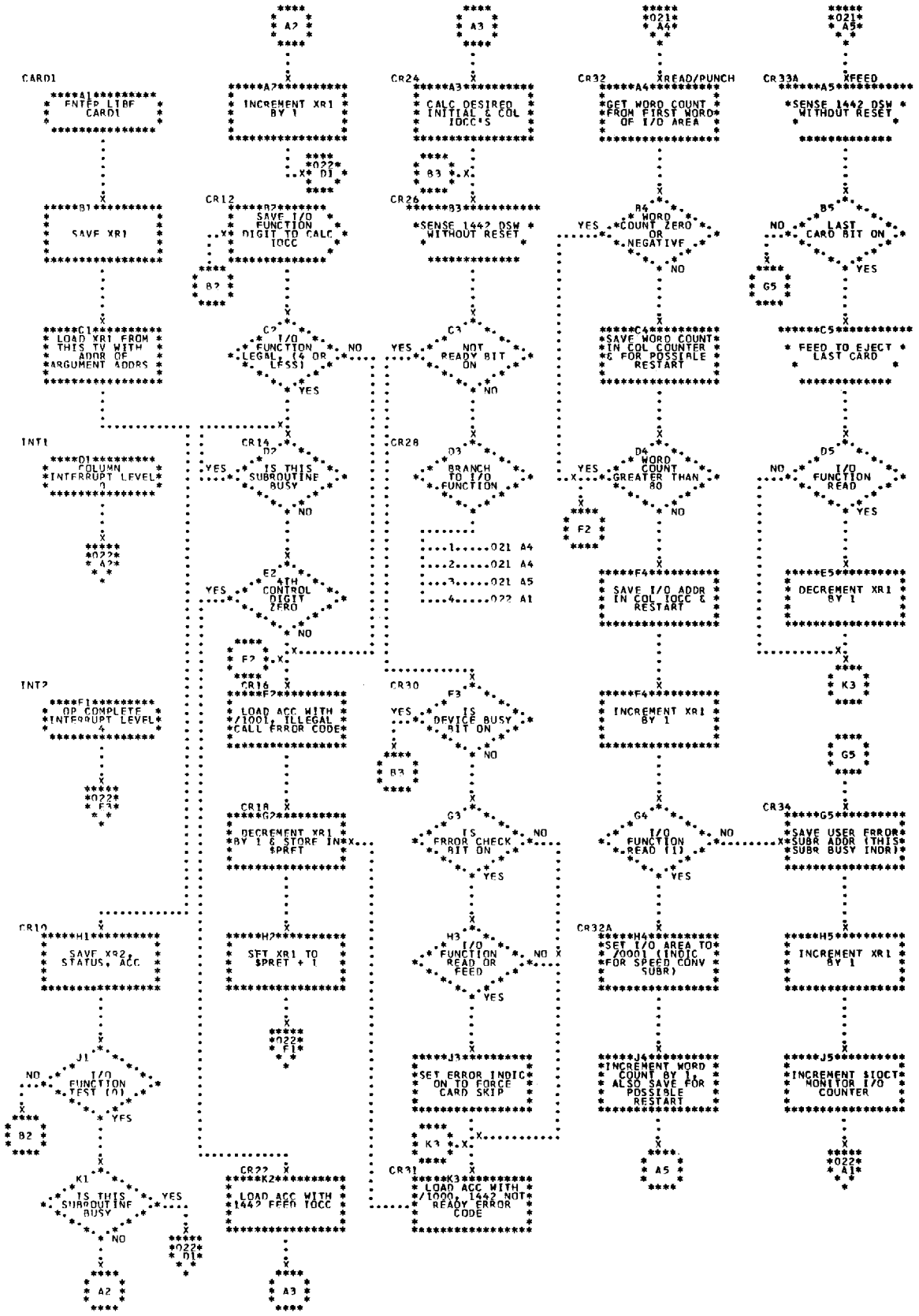
Flowchart FIO31. System Library, CARD0



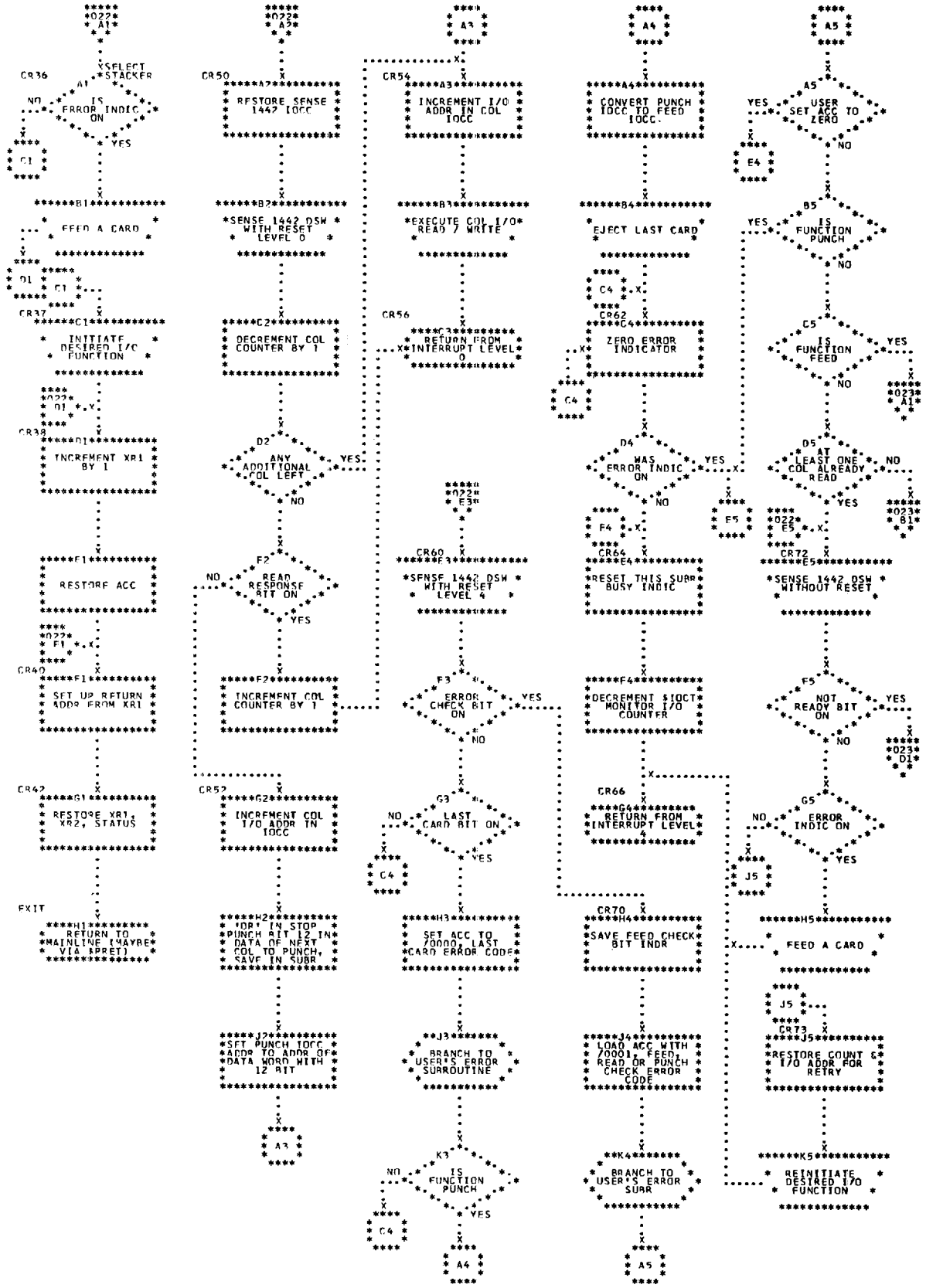
Flowchart FIO32. System Library, CARD0



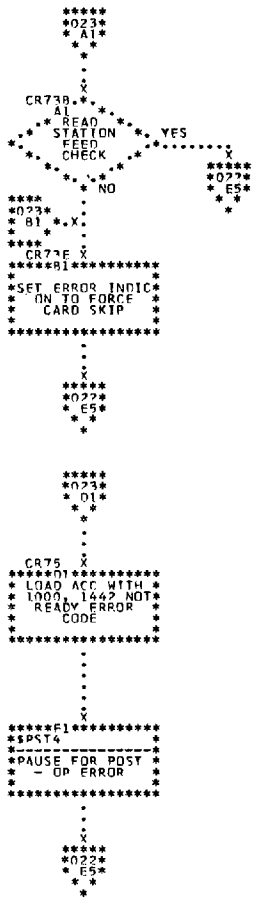
Flowchart FIO33. System Library, CARD0



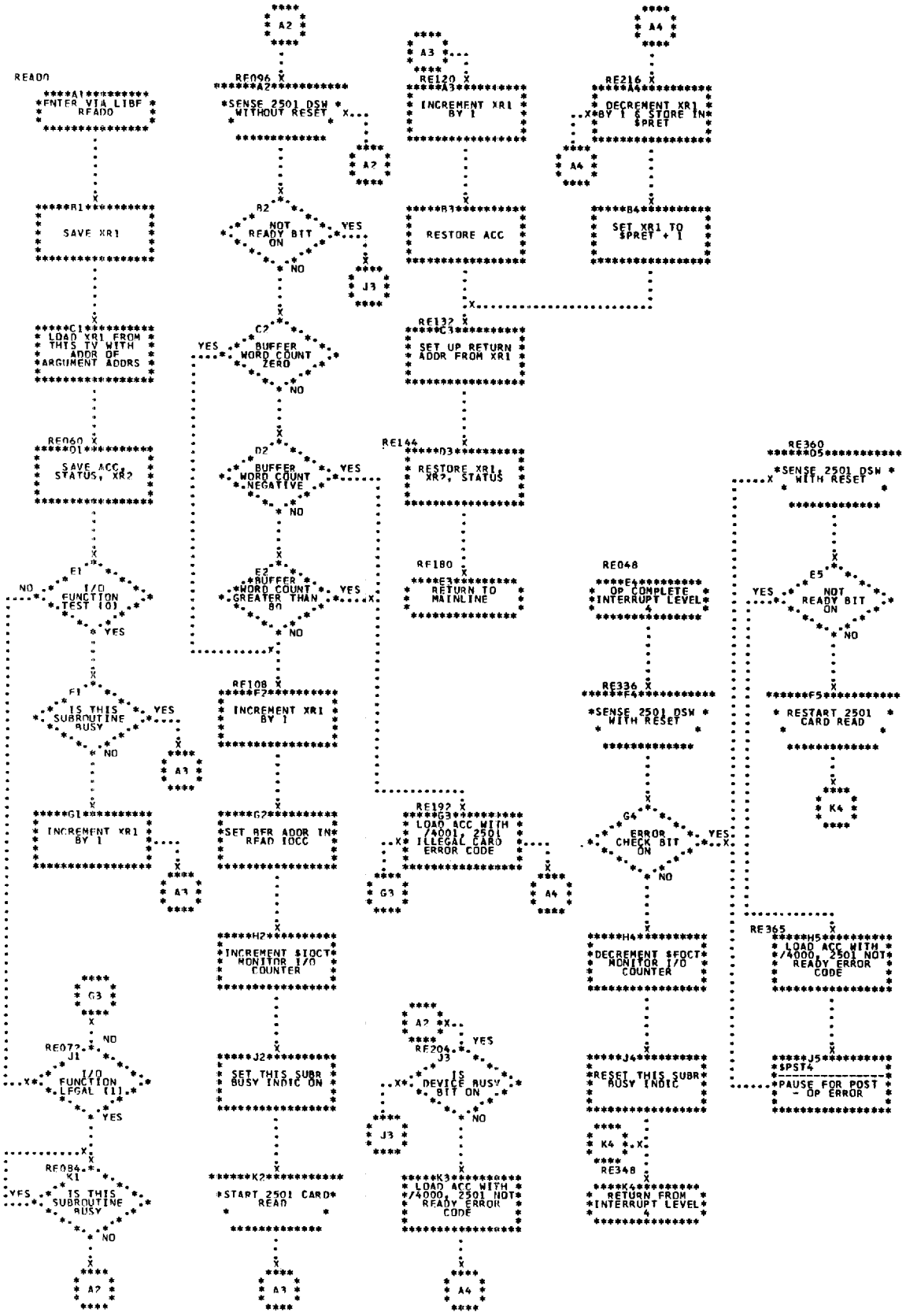
Flowchart FIO34. System Library, CARD1



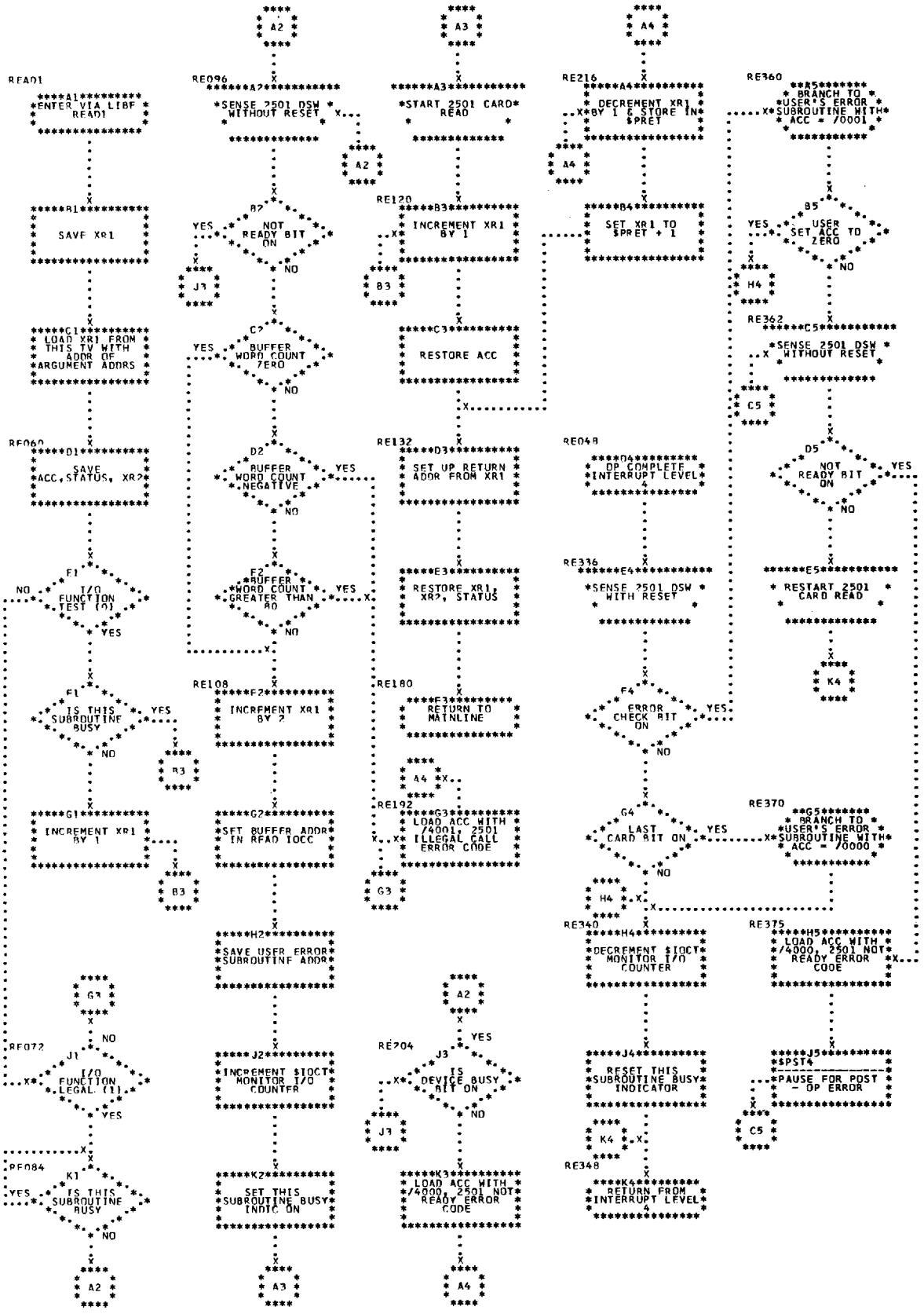
Flowchart FIO35. System Library, CARD1



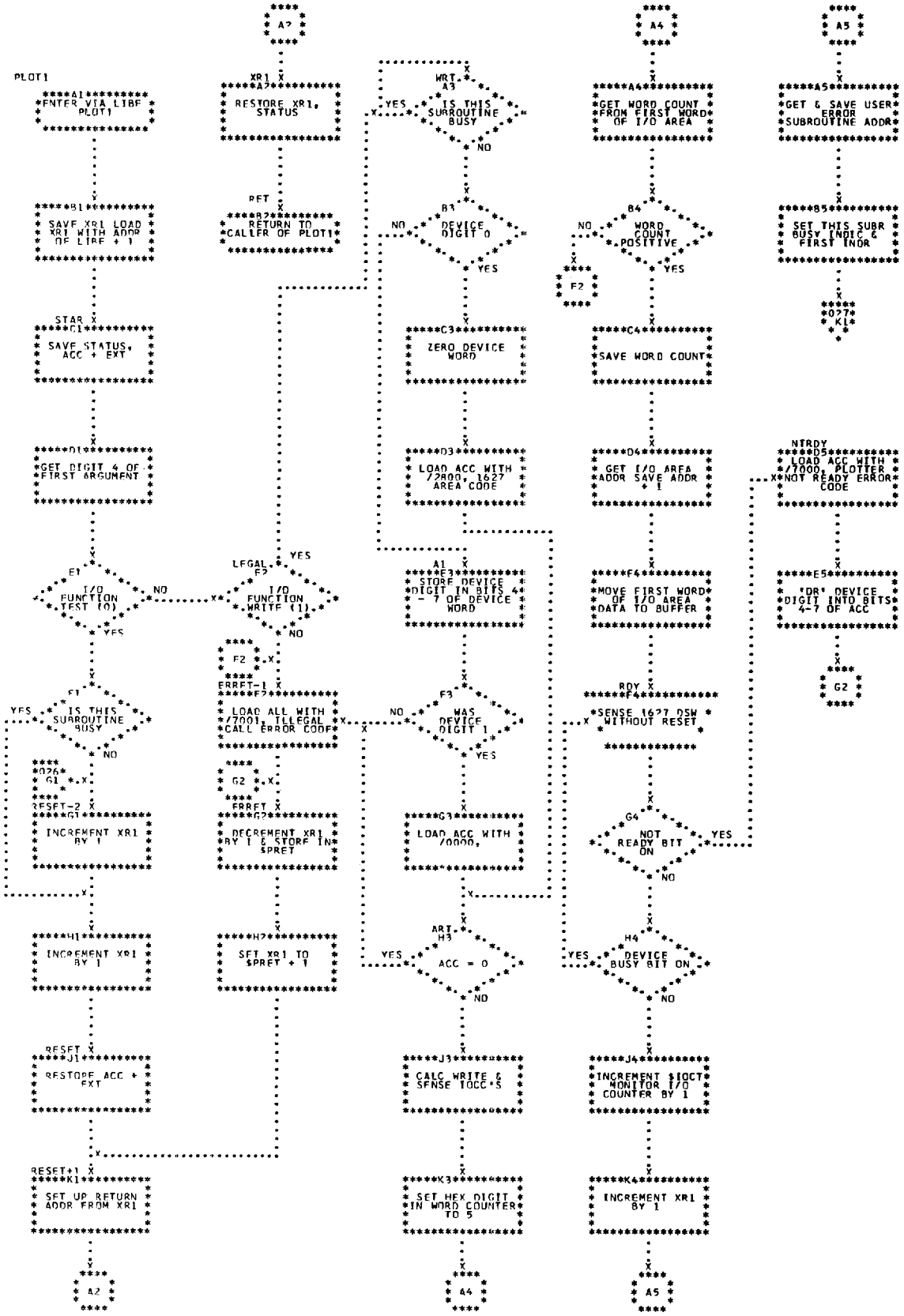
Flowchart FIO26. System Library, CARD1



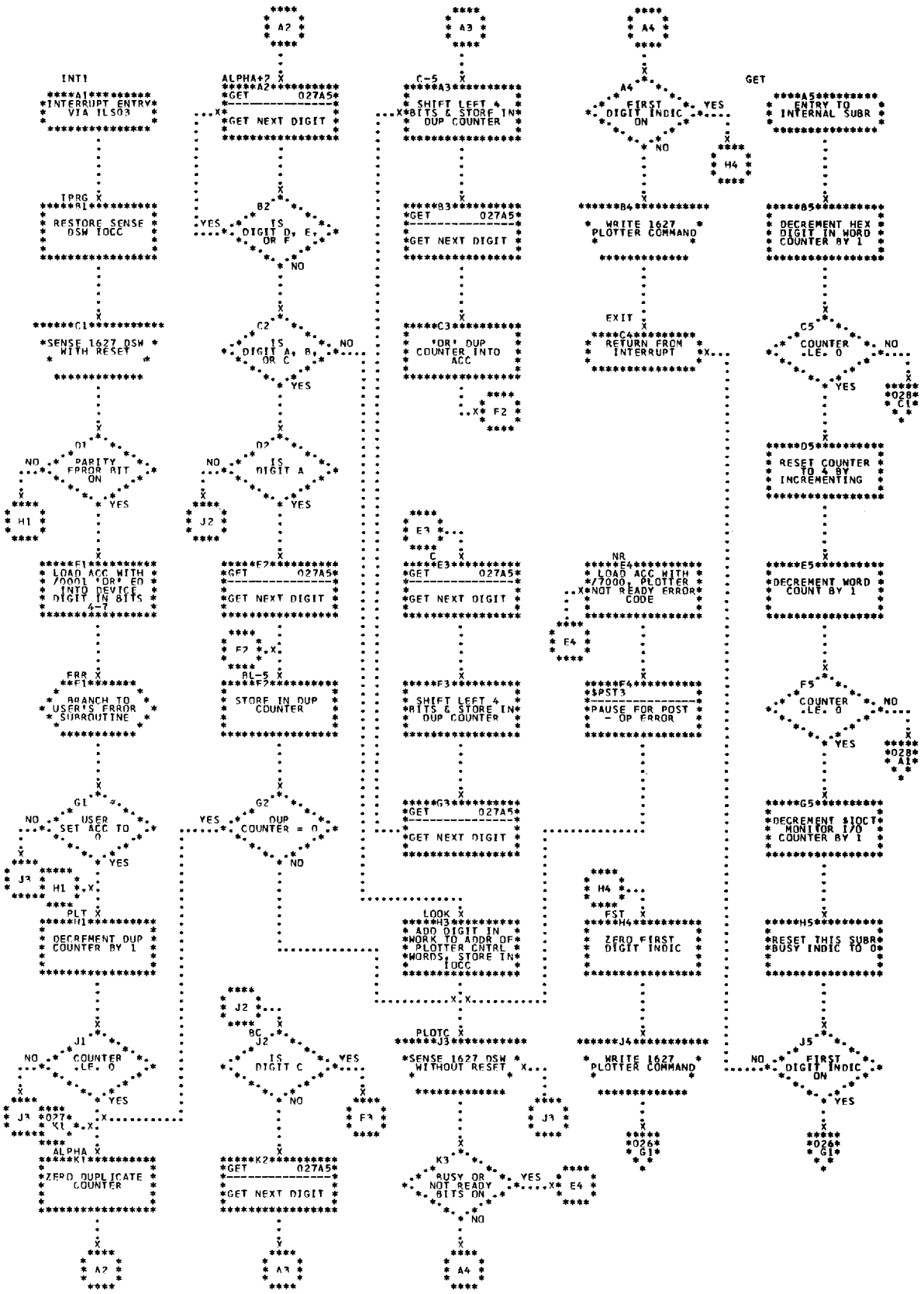
Flowchart FIO37. System Library, READ0



Flowchart FIO38. System Library, READ1

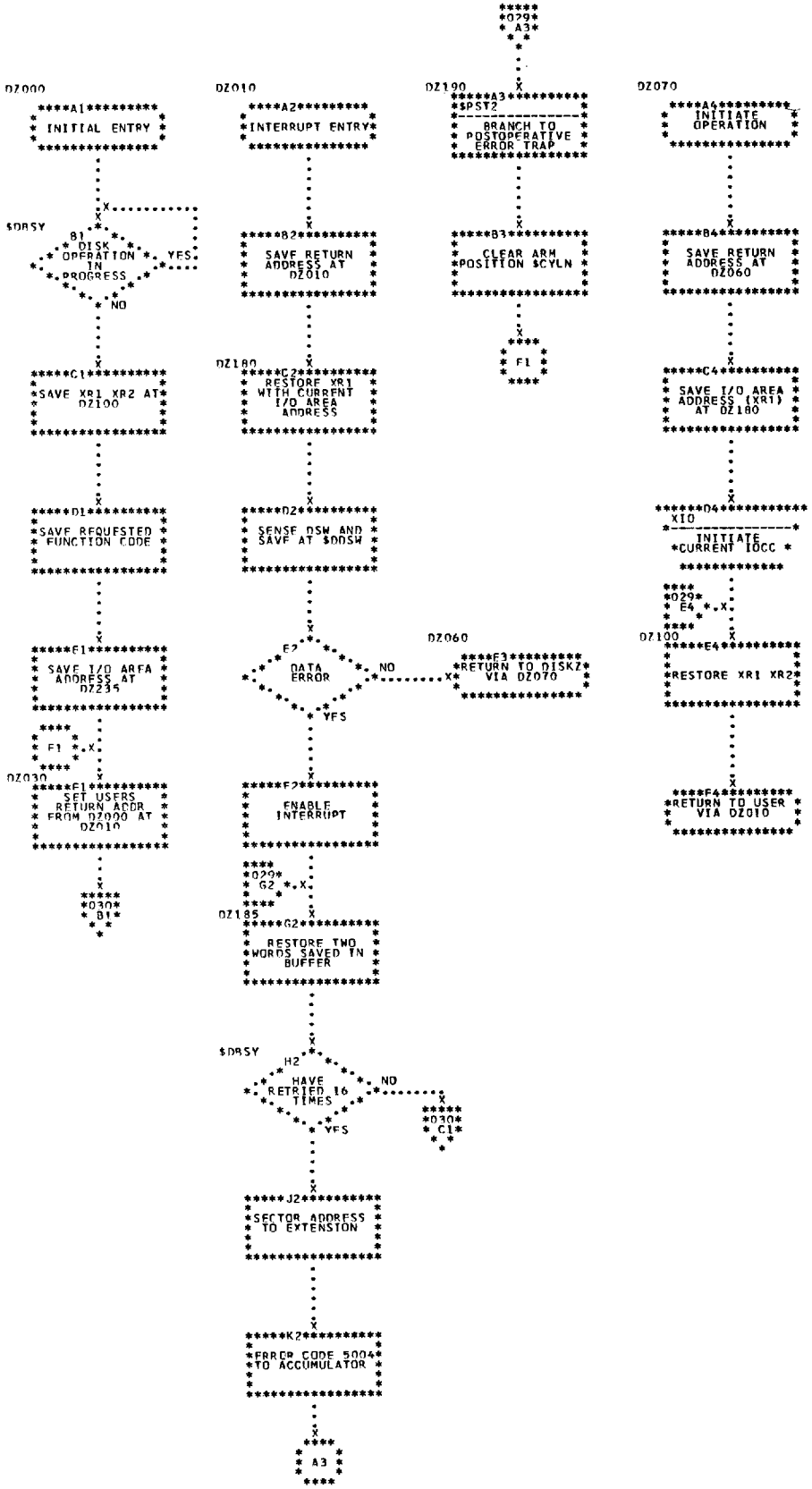


Flowchart FIO39. System Library, PLOT1

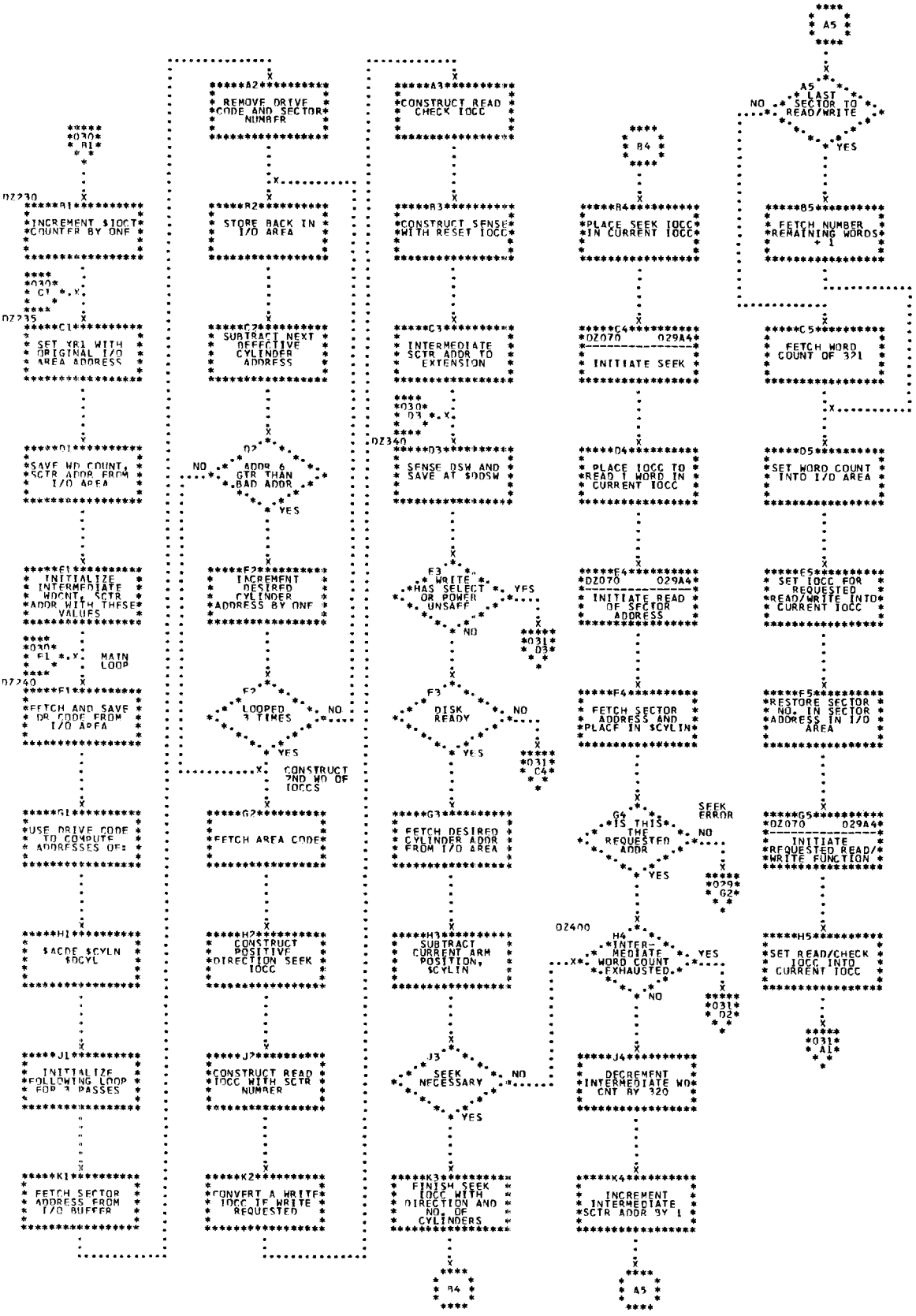


Flowchart FIO40. System Library, PLOT1

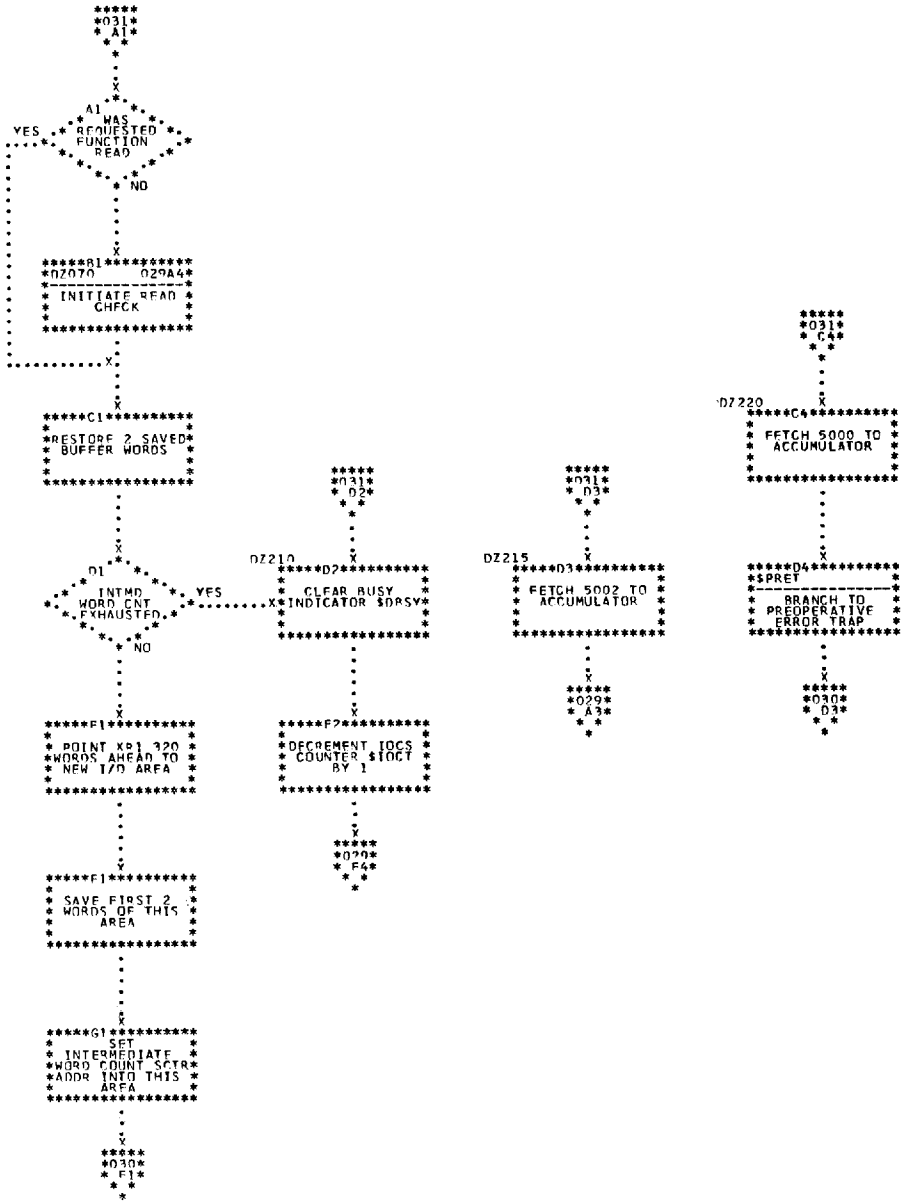
Flowchart FIO41. System Library, PLOT1



Flowchart FIO42. System Library, DISKZ



Flowchart FIO43. System Library, DISKZ



Flowchart FIO44. System Library, DISKZ

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.

Source Coding	Object Coding			Object Coding With Trace*	
Arithmetic Statements - real, integer, and mixed modes					
I=J	LD	L	J	.	
	STO	L	I	LIBF	SIAR
				DC	I
A=B	LIBF		FLD	.	
	DC		B	.	
	LIBF		FSTO	LIBF	SFAR
	DC		A	.	
A=I	LD	L	I	.	
	LIBF		FLOAT	.	
	LIBF		FSTO	LIBF	SFAR
	DC		A	.	
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	LIBF	SFAR
	DC		A	.	

* 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 Coding	Object Coding			Object 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	LD	L	I	.	
or	LIBF		FLOAT	.	
(A=I+B-J)	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	.	
	STO	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	.	

Source Coding	Object Coding			Object Coding With Trace	
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

Source Coding	Object Coding			Object Coding With Trace	
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=I**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	GT1	.	
	LIBF		FLD	.	
	DC		B	.	
	LIBF		FAXI	.	
	DC		GT1	.	
	LIBF		FSTO	LIBF	SFAR
	DC		A	.	
A=B**(C+D)	LIBF		FLD	.	
	DC		C	.	
	LIBF		FADD	.	
	DC		D	.	
	LIBF		FSTO	.	
	DC		GT1	.	
	LIBF		FLD	.	
	DC		B	.	

Source Coding		Object Coding			Object Coding With Trace	
		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 Coding	Object Coding			Object 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)	.	

Source Coding	Object Coding			Object Coding With Trace	
	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
Statement Function Statements					
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	.	

Source Coding		Object Coding			Object Coding With Trace	
A=C(B, .5)+E		CALL		C	.	
		DC		B	.	
		DC		ADR1	.	
		LIBF		FADD	.	
		DC		E	.	
		LIBF		FSTO	LIBF	SFAR
		DC		A	.	
		:			:	
	ADR1	DC		.5	.	

Call Statements

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	.	

Source Coding		Object Coding			Object Coding With Trace
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	.
		NOP			.
		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		:			:
		:			:
		:			:

Source Coding		Object Coding			Object Coding With Trace		
		LD	L	I	.		
		A	L	M	.		
		STO	L	I	.		
		S	L	K	.		
		BSC	L	ADR1,+			
GO TO Statement							
GO TO 111		BSC	L	ADR1	.		
		:			:		
	ADR1	(coding generated from statement 111)			.		
Computed GO TO Statement							
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)			.		
IF Statements							
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)			.		

Source Coding		Object Coding			Object Coding With Trace		
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)			.		

PAUSE Statement

PAUSE 11		LIBF		PAUSE
		DC		ADR1
		:		
	ADR1	DC		/11

STOP Statement

STOP 21		LIBF		STOP
		DC		ADR1
		:		
	ADR1	DC		/21

Source Coding	Object Coding		
RETURN 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	SDFIC (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
LIBF	FLD (or ELD)
LIBF	FSTO (or ESTO)

Source Coding | Object Coding

Non-Disk I/O Statements

Source Coding		Object Coding		
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		LIBF		SRED
where X is dimensioned		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 D_4 for array X
		DC		I
		DC		value D_1 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 Coding	Object 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 Coding		Object 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		TWO
		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
		LIBF		SDCOM
FIND (J,K)		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.

The contents of this appendix are not to be construed as an external specification, i.e., the locations in these listings may be changed. $PRET, $OREQ, $EXIT, $LINK, and $DUMP are the only guaranteed locations.

Note that in the listings the character is printed as ', and the character is printed as =.

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                   0001    * RLTV ADDR* SYMBOL* DESCRIPTION                         PMN00010
                   0002    *          *        *                                    PMN00020
                   0003    * 0-3      *        * RESERVED FOR EVEN BOUNDARIES       PMN00030
                   0004    * 4-5      * #NAME  * NAME OF PROGRAM/CORE LOAD          PMN00040
                   0005    * 6        * #DBCT  * BLOCK COUNT OF PROG/CORE LOAD      PMN00050
                   0006    * 7        * #FCNT  * *FILES SWITCH--ZERO MEANS NO       PMN00060
                   0007    *          *        * FILES HAVE BEEN EQUATED            PMN00070
                   0008    * 8        * #SYSC  * SYS/NON-SYS CARTRIDGE INDR        PMN00080
                   0009    * 9        * #JBSW  * JOBT SWITCH-- NON-ZERO MEANS      PMN00090
                   0010    *          *        * TEMPORARY MODE                     PMN00100
                   0011    * 10       * #CBSW  * CLB-RETURN-TO-DUP SWITCH--        PMN00110
                   0012    *          *        * ZERO=CLB RETURN TO SUPV            PMN00120
                   0013    * 11       * #LCNT  * NO. OF LOCALS                     PMN00130
                   0014    * 12       * #MPSW  * CORE MAP SWITCH--ZERO MEANS       PMN00140
                   0015    *          *        * DO NOT PRINT A CORE MAP            PMN00150
                   0016    * 13       * #MDF1  * NO. DUP CTRL RECDS (MODIF)        PMN00160
                   0017    * 14       * #MDF2  * ADDR OF MODIF BUFFER              PMN00170
                   0018    * 15       * #NCNT  * NO. OF NOCALS                     PMN00180
                   0019    * 16       * #ENTY  * RLTV ENTRY ADDR OF PROGRAM        PMN00190
                   0020    * 17       * #RP67  * 1442-5 SW (0#1442-5 ON SYSTEM     PMN00200
                   0021    * 18       * #TODR  * 'TO' WORKING STG DRIVE CODE       PMN00210
                   0022    * 19       * #FRDR  * 'FROM' WORKING STG DRIVE CODE     PMN00220
                   0023    * 20       * #FHOL  * ADDR OF LARGEST HOLE IN FXA       PMN00230
                   0024    * 21       * #FSZE  * BLK CNT OF LARGEST HOLE IN FXA    PMN00240
                   0025    * 22       * #UHOL  * ADDR OF LARGEST HOLE IN UA        PMN00250
                   0026    * 23       * #USZE  * BLK CNT OF LARGEST HOLE IN UA     PMN00260
                   0027    * 24       * #DCSW  * DUP CALL SW--NON-ZERO#DUP CALL    PMN00270
                   0028    * 25       * #PIOD  * PRINCIPAL I/O DEVICE INDICATOR    PMN00280
                   0029    * 26       * #PPTR  * PRINC. PRINT DEVICE INDICATOR     PMN00290
                   0030    * 27       * #CIAD  * RLTV ADDR IN 'STRT OF CIL ADDR    PMN00300
                   0031    * 28       * #ACIN  * AVAILABLE CARTRIDGE INDICAT2-2    PMN00310
                   0032    * 29       * #GRPH  * 2250 INDICATOR              2G2   PMN00320
                   0033    * 30       * #GCNT  * NO. G2250 RECORDS           2G2   PMN00330
                   0034    * 31       * #LOSW  * LOCAL-CANNOT-CALL-LOCAL SW  2-2   PMN00340
                   0035    * 32       * #X3SW  * SPECIAL ILS SWITCH          2-2   PMN00350
                   0036    * 33       * #ECNT  * NO. OF *EQUAT RCDS          2-4   PMN00355
                   0037    * 33-34    *        * RESERVED FOR FUTURE USE     2-2   PMN00360
                   0038    * 35       * #ANDU  * 1+BLOCK ADDR OF END OF USER       PMN00370
                   0039    *          *        * AREA (ADJUSTED) LOGICAL DR 0       PMN00380
                   0040    * 36       *        * 1+BLOCK ADDR OF END OF USER       PMN00390
                   0041    *          *        * AREA (ADJUSTED) LOGICAL DR 1       PMN00400
                   0042    * 37       *        * 1+BLOCK ADDR OF END OF USER       PMN00410
                   0043    *          *        * AREA (ADJUSTED) LOGICAL DR 2       PMN00420
                   0044    * 38       *        * 1+BLOCK ADDR OF END OF USER       PMN00430
                   0045    *          *        * AREA (ADJUSTED) LOGICAL DR 3       PMN00440
                   0046    * 39       *        * 1+BLOCK ADDR OF END OF USER       PMN00450
                   0047    *          *        * AREA (ADJUSTED) LOGICAL DR 4       PMN00460
                   0048    * 40       * #BNDU  * 1+BLOCK ADDR OF END OF USER       PMN00470
                   0049    *          *        * AREA (BASE) LOGICAL DRIVE  0       PMN00480
                   0050    * 41       *        * 1+BLOCK ADDR OF END OF USER       PMN00490
                   0051    *          *        * AREA (BASE) LOGICAL DRIVE  1       PMN00500
                   0052    * 42       *        * 1+BLOCK ADDR OF END OF USER       PMN00510
                   0053    *          *        * AREA (BASE) LOGICAL DRIVE  2       PMN00520
                   0054    * 43       *        * 1+BLOCK ADDR OF END OF USER       PMN00530
                   0055    *          *        * AREA (BASE) LOGICAL DRIVE  3       PMN00540
                   0056    * 44       *        * 1+BLOCK ADDR OF END OF USER       PMN00550
                   0057    *          *        * AREA (BASE) LOGICAL DRIVE  4       PMN00560
                   0058    * 45       * #FPAD  * FILE PROTECT ADDR, LOGICAL        PMN00570
                   0059    *          *        * DRIVE 0 (BASE)                     PMN00580
                   0060    * 46       *        * FILE PROTECT ADDR, LOGICAL        PMN00590
                   0061    *          *        * DRIVE 1 (BASE)                     PMN00600
                   0062    * 47       *        * FILE PROTECT ADDR, LOGICAL        PMN00610
                   0063    *          *        * DRIVE 2 (BASE)                     PMN00620
                   0064    * 48       *        * FILE PROTECT ADDR, LOGICAL        PMN00630
                   0065    *          *        * DRIVE 3 (BASE)                     PMN00640
                   0066    * 49       *        * FILE PROTECT ADDR,LOGICAL         PMN00650
                   0067    *          *        * DRIVE 4 (BASE)                     PMN00660
                   0068    * 50       * #PCID  * CARTRIDGE ID, PHYSICAL DRIVE 0    PMN00670
                   0069    * 51       *        * CARTRIDGE ID, PHYSICAL DRIVE 1    PMN00680
                   0070    * 52       *        * CARTRIDGE ID, PHYSICAL DRIVE 2    PMN00690
                   0071    * 53       *        * CARTRIDGE ID, PHYSICAL DRIVE 3    PMN00700
                   0072    * 54       *        * CARTRIDGE ID, PHYSICAL DRIVE 4    PMN00710
                   0073    * 55       * #CIDN  * CARTRIDGE ID, LOGICAL DRIVE 0     PMN00720
                   0074    * 56       *        * CARTRIDGE ID, LOGICAL DRIVE 1     PMN00730
```

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO
                  0075     * 57       *       * CARTRIDGE ID, LOGICAL DRIVE 2      PMN00740
                  0076     * 58       *       * CARTRIDGE ID, LOGICAL DRIVE 3      PMN00750
                  0077     * 59       *       * CARTRIDGE ID, LOGICAL DRIVE 4      PMN00760
                  0078     * 60       * #CIBA * SCTR ADDR OF CIB, LOGICAL DR 0     PMN00770
                  0079     * 61       *       * SCTR ADDR OF CIB, LOGICAL DR 1     PMN00780
                  0080     * 62       *       * SCTR ADDR OF CIB, LOGICAL DR 2     PMN00790
                  0081     * 63       *       * SCTR ADDR OF CIB, LOGICAL DR 3     PMN00800
                  0082     * 64       *       * SCTR ADDR OF CIB, LOGICAL DR 4     PMN00810
                  0083     * 65       * #SCRA * SCRA, LOGICAL DRIVE 0              PMN00820
                  0084     * 66       *       * SCRA, LOGICAL DRIVE 1              PMN00830
                  0085     * 67       *       * SCRA, LOGICAL DRIVE 2              PMN00840
                  0086     * 68       *       * SCRA, LOGICAL DRIVE 3              PMNC0850
                  0087     * 69       *       * SCRA, LOGICAL DRIVE 4              PMN00860
                  0088     * 70       * #FMAT * FORMAT OF PROG IN WS, DRIVE 0      PMN00870
                  0089     * 71       *       * FORMAT OF PROG IN WS, DRIVE 1      PMN00880
                  0090     * 72       *       * FORMAT OF PROG IN WS, DRIVE 2      PMN00890
                  0091     * 73       *       * FORMAT OF PROG IN WS, DRIVE 3      PMN00900
                  0092     * 74       *       * FORMAT OF PROG IN WS, DRIVE 4      PMN00910
                  0093     * 75       * #FLET * FLET SCTR ADDR, LOGICAL DR 0       PMN00920
                  0094     * 76       *       * FLET SCTR ADDR, LOGICAL DR 1       PMN00930
                  0095     * 77       *       * FLET SCTR ADDR, LOGICAL DR 2       PMN00940
                  0096     * 78       *       * FLET SCTR ADDR, LOGICAL DR 3       PMN00950
                  0097     * 79       *       * FLET SCTR ADDR, LOGICAL DR 4       PMN00960
                  0098     * 80       * #ULET * LET SCTR ADDR, LOGICAL DR 0        PMN00970
                  0099     * 81       *       * LET SCTR ADDR, LOGICAL DR 1        PMN00980
                  0100     * 82       *       * LET SCTR ADDR, LOGICAL DR 2        PMN00990
                  0101     * 83       *       * LET SCTR ADDR, LOGICAL DR 3        PMN01000
                  0102     * 84       *       * LET SCTR ADDR, LOGICAL DR 4        PMN01010
                  0103     * 85       * #WSCT * BLK CNT OF PROG IN WS, DRIVE 0     PMN01020
                  0104     * 86       *       * BLK CNT OF PROG IN WS, DRIVE 1     PMN01030
                  0105     * 87       *       * BLK CNT OF PROG IN WS, DRIVE 2     PMN01040
                  0106     * 88       *       * BLK CNT OF PROG IN WS, DRIVE 3     PMN01050
                  0107     * 89       *       * BLK CNT OF PROG IN WS, DRIVE 4     PMN01060
                  0108     * 90       * #CSHN * SCTR CNT CUSHION,LOGICAL DR 0      PMN01070
                  0109     * 91       *       * SCTR CNT CUSHION,LOGICAL DR 1      PMN01080
                  0110     * 92       *       * SCTR CNT CUSHION,LOGICAL DR 2      PMN01090
                  0111     * 93       *       * SCTR CNT CUSHION,LOGICAL DR 3      PMN01100
                  0112     * 94       *       * SCTR CNT CUSHION,LOGICAL DR 4      PMN01110
                  0113     * 95-319   *       * RESERVED FOR FUTURE USE            PMN01120

RESIDENT MONITOR

ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  0115     ************************************************** PMN01140
                  0116     *                                                * PMN01150
                  0117     *STATUS-VERSION 2, MODIFICATION 4                * PMN01160
                  0118     *                                                * PMN01170
                  0119     *FUNCTION/OPERATION-                             * PMN01180
                  0120     *   THIS SECTION ALWAYS REMAINS IN CORE. IT      * PMN01190
                  0121     *   IS COMPRISED OF THE COMMUNICATIONS           * PMN01200
                  0122     *   AREA (COMMA), THE SKELETON SUPERVISOR, AND   * PMN01210
                  0123     *   A DISK I/O SUBROUTINE, NOMINALLY DISK2. (THE * PMN01220
                  0124     *   FIRST TWO OF THESE SECTIONS ARE INTERMIXED.) * PMN01230
                  0125     *   COMMA CONTAINS THE SYSTEM PARAMETERS REQUIR- * PMN01240
                  0126     *   ED TO FETCH A CORE LOAD IN CORE IMAGE FOR-   * PMN01250
                  0127     *   MAT. THE SKELETON SUPERVISOR PROVIDES IN-    * PMN01260
                  0128     *   STRUCTIONS FOR INITIATING A CALL EXIT, A     * PMN01270
                  0129     *   CALL LINK, A DUMP-TO-PRINTER OR A CALL TO THE * PMN01280
                  0130     *   AUXILIARY SUPERVISOR.  IN ADDITION, THE SKELE-* PMN01290
                  0131     *   TON SUPERVISOR CONTAINS SEVERAL TRAPS FOR CER-* PMN01300
                  0132     *   TAIN I/O FUNCTIONS/CONDITIONS. THE DISK I/O  * PMN01310
                  0133     *   SECTION CONSISTS OF A SUBROUTINE FOR READING * PMN01320
                  0134     *   FROM OR WRITING ON A DISK CARTRIDGE ON A     * PMN01330
                  0135     *   GIVEN LOGICAL DISK DRIVE.                    * PMN01340
                  0136     *                                                * PMN01350
                  0137     *ENTRY POINTS-                                   * PMN01360
                  0138     *   * $PRET-A TRAP FOR PREOPERATIVE I/O ERRORS.  * PMN01370
                  0139     *           THE CALLING SEQUENCE IS              * PMN01380
                  0140     *              BSI   L  $PRET                    * PMN01390
                  0141     *   * $PSTX-A POSTOPERATIVE ERROR TRAP FOR I/O   * PMN01400
                  0142     *           DEVICES ON LEVEL X (X#1,2,3,OR 4).   * PMN01410
                  0143     *           THE CALLING SEQUENCE IS              * PMN01420
                  0144     *              BSI   L  $PSTX                    * PMN01430
                  0145     *   * $STOP-THE PROGRAM STOP KEY TRAP.           * PMN01440
                  0146     *   * $EXIT-THE ENTRY POINT FOR THE EXIT/CALL    * PMN01450
                  0147     *         EXIT STATEMENT. THE CALLING SEQUENCE IS* PMN01460
                  0148     *            LDX   0 $EXIT                      * PMN01470
                  0149     *   * $LINK-THE ENTRY POINT FOR THE LINK/CALL    * PMN01480
                  0150     *         LINK STATEMENT. THE CALLING SEQUENCE IS* PMN01490
                  0151     *            BSI   L  $LINK                     * PMN01500
                  0152     *   * $DUMP-THE ENTRY POINT FOR THE DUMP/PDMP    * PMN01510
                  0153     *         STATEMENT. THE CALLING SEQUENCE IS     * PMN01520
                  0154     *            BSI   L  $DUMP                     * PMN01530
                  0155     *            DC       FORMAT                    * PMN01540
                  0156     *            DC       LIMIT1                    * PMN01550
                  0157     *            DC       LIMIT2                    * PMN01560
                  0158     *         WHERE LIMIT1 AND LIMIT2 ARE THE LIMITS * PMN01570
                  0159     *         BETWEEN WHICH THE DUMP IS TO OCCUR, AND* PMN01580
                  0160     *         FORMAT IS A CODE INDICATING THE FORMAT * PMN01590
                  0161     *         OF THE DUMP. IF FORMAT IS NEGATIVE,    * PMN01600
                  0162     *         THE AUXILIARY SUPERVISOR IS FETCHED    * PMN01610
                  0163     *         AND CONTROL PASSED TO IT.              * PMN01620
```

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  0164     *    * DZ000-ENTERED WHEN THE CALLER WISHES TO      * PMN01630
                  0165     *           PERFORM A DISK I/O OPERATION. THE       * PMN01640
                  0166     *           CALLING SEQUENCE VARIES WITH THE        * PMN01650
                  0167     *           VERSION OF THE DISK I/O SUBROUTINE.     * PMN01660
                  0168     *    * $I200/$I400-ENTERED WHEN THE OPERATION-      * PMN01670
                  0169     *                COMPLETE INTERRUPT OCCURS ON       * PMN01680
                  0170     *                LEVEL 2/4.                         * PMN01690
                  0171     *                                                   * PMN01700
                  0172     *INPUT-N/A                                          * PMN01710
                  0173     *                                                   * PMN01720
                  0174     *OUTPUT-WORDS 6-4090 SAVED ON THE CIB ON A CALL     * PMN01730
                  0175     *       DUMP                                        * PMN01740
                  0176     *                                                   * PMN01750
                  0177     *EXTERNAL REFERENCES-N/A                            * PMN01760
                  0178     *                                                   * PMN01770
                  0179     *EXITS-                                             * PMN01780
                  0180     *    * NORMAL                                       * PMN01790
                  0181     *        *THE EXITS FROM THE SUBROUTINES AT $PRET   * PMN01800
                  0182     *         $PST1, $PST2, $PST3, $PST4, AND $STOP     * PMN01810
                  0183     *         ARE BRANCH INSTRUCTIONS FOLLOWING A       * PMN01820
                  0184     *         WAIT INSTRUCTION. $STOP TURNS OFF IN-     * PMN01830
                  0185     *         TERRUPT LEVEL 5 AFTER THE START KEY IS    * PMN01840
                  0186     *         DEPRESSED.                                * PMN01850
                  0187     *        *THE EXITS FROM $EXIT,$LINK,AND $DUMP ARE  * PMN01860
                  0188     *         TO THE CORE IMAGE LOADER, PHASE 1,        * PMN01870
                  0189     *         AFTER THAT PHASE HAS BEEN FETCHED.        * PMN01880
                  0190     *        *THE EXIT FROM DZ000 IS BACK TO THE        * PMN01890
                  0191     *         CALLER AFTER THE REQUESTED DISK OPERA-    * PMN01900
                  0192     *         TION HAS BEEN INITIATED.                  * PMN01910
                  0193     *        *THE EXITS FROM $I200/$I400 ARE BACK TO    * PMN01920
                  0194     *         THE ADDRESSES FROM WHICH THE DISK OP-     * PMN01930
                  0195     *         ERATION COMPLETE INTERRUPT OCCURED        * PMN01940
                  0196     *         AFTER THE INTERRUPT HAS BEEN SERVICED     * PMN01950
                  0197     *         BY THE APPROPRIATE ISS.                   * PMN01960
                  0198     *    * ERROR-N/A                                    * PMN01970
                  0199     *                                                   * PMN01980
                  0200     *TABLES/WORK AREAS-                                 * PMN01990
                  0201     *    * $ACDE                                        * PMN02000
                  0202     *    * $CH12                                        * PMN02010
                  0203     *    * $CILA                                        * PMN02020
                  0204     *    * $CLSW                                        * PMN02030
                  0205     *    * $COMN                                        * PMN02040
                  0206     *    * $CORE                                        * PMN02050
                  0207     *    * $CTSW                                        * PMN02060
                  0208     *    * $CXR1                                        * PMN02070
                  0209     *    * $CYLN                                        * PMN02080
                  0210     *    * $DADR                                        * PMN02100
                  0211     *    * $DBSY                                        * PMN02110
                  0212     *    * $DCYL                                        * PMN02120
                  0213     *    * $DMPF                                        * PMN02130
                  0214     *    * $DREQ                                        * PMN02140
                  0215     *    * $FPAD                                        * PMN02150
                  0216     *    * $GCOM                                 2G2    * PMN02160
                  0217     *    * $GRIN                                 2G2    * PMN02170
                  0218     *    * $HASH                                        * PMN02180
                  0219     *    * $IBT2                                        * PMN02190
                  0220     *    * $IBT4                                        * PMN02200
                  0221     *    * $IBSY                                        * PMN02210
                  0222     *    * $IOCT                                        * PMN02220
                  0223     *    * $KCSW                                        * PMN02230
                  0224     *    * $LAST                                        * PMN02240
                  0225     *    * $NDUP                                        * PMN02250
                  0226     *    * $NXEQ                                        * PMN02260
                  0227     *    * $PBSY                                        * PMN02270
                  0228     *    * $PGCT                                        * PMN02280
                  0229     *    * $PHSE                                        * PMN02290
                  0230     *    * $RMSW                                        * PMN02300
                  0231     *    * $SCAT                                 2-4    * PMN02305
                  0232     *    * $SNLT                                        * PMN02310
                  0233     *    * $UFIO                                        * PMN02320
                  0234     *    * $ULET                                        * PMN02330
                  0235     *    * $WRD1                                        * PMN02340
                  0236     *    * $WSDR                                        * PMN02350
                  0237     *    * $XR3X                                 2-2    * PMN02360
                  0238     *                                                   * PMN02370
                  0239     *ATTRIBUTES-REUSABLE                               * PMN02380
                  0240     *                                                   * PMN02390
                  0241     *NOTES-                                            * PMN02400
                  0242     *    THERE ARE WAIT INSTRUCTIONS AT $PRET+1,        * PMN02410
                  0243     *    $STOP+1, AND $PSTX+1. DEPRESSING THE START     * PMN02420
                  0244     *    KEY WILL RETURN CONTROL TO THE CALLER IN ALL   * PMN02430
                  0245     *    CASES.                                         * PMN02440
                  0246     ************************************************** PMN02450
```

```
ADDR REL OBJECT    ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                    0248    * PROVIDE PARAMETERS FOR SYSTEM LOADER                   PMN02470
                    0249    *                                                        PMN02480
                    0250          ABS                                                PMN02490
0280                0251          ORG     4                                          PMN02500
0004 0  0FFA        0252          DC      4095-* WD CNT FOR WRITING CORE ON CIB      PMN02510
0005 0  0000        0253    $CIBA DC      *-*  SCTR ADDR OF THE CIB                  PMN02520
0006 0  0000        0254    $CH12 DC      *-*  ADDR OF CHANNEL 12 INDICATOR          PMN02530
0007 0  0000        0255    $COMN DC      *-*  LENGTH OF COMMON (IN WORDS)           PMN02540
                    0256    *                                                        PMN02550
                    0257    * ULTIMATE RESIDENCE OF THE INTERRUPT TV                 PMN02560
                    0258    *                                                        PMN02570
0008 0  0000        0259    $LEV0 DC      *-*   LEVEL 0 BRANCH ADDRESS             PMN02580
0009 0  0000        0260    $LEV1 DC      *-*   LEVEL 1 BRANCH ADDRESS             PMN02590
000A 0  0083        0261    $LEV2 DC      $I200 LEVEL 2 BRANCH ADDR                PMN02600
000B 0  0000        0262    $LEV3 DC      *-*   LEVEL 3 BRANCH ADDRESS             PMN02610
000C 0  00C4        0263    $LEV4 DC      $I400 LEVEL 4 BRANCH ADDR                PMN02620
000D 0  0091        0264    $LEV5 DC      $STOP LEVEL 5 BRANCH ADDR                PMN02630
                    0265    *                                                        PMN02640
                    0266    *                                                        PMN02650
000E 0  0000        0267    $CORE DC      *-*  SIZE OF CORE, E.G., 4096=4K         PMN02660
000F 0  0000        0268    $CTSW DC      *-*  CONTROL RECORD TRAP SWITCH           PMN02670
0010 0  0000        0269    $DADR DC      *-*  SCTR ADDR OF PROG TO BE LOADED       PMN02680
0011 0  0000        0270    $SCAT DC      *-*  NON ZERO=SCA INTRPT PNDNG     2-4    PMN02690
0012 0  0000        0271    $DREQ DC      *-*  IND. FOR REQUESTED VERSION DKI/O    PMN02700
0013 0  0000        0272    $IBSY DC      *-*  NON-ZERO IF CD/PAP TP DEV. BUSY     PMN02710
0014      000C      0273    $HASH BSS  E  12   WORK AREA                           PMN02720
                    0274    *                                                        PMN02730
                    0275    *                                                        PMN02740
0020      0008      0276    $SCAN BSS     8    1132 SCAN AREA                 32     PMN02750
                    0277    *                                                        PMN02760
                    0278    *                                                        PMN02770
                    0279    *                                                        PMN02780
                    0280    * TRAP FOR PREOPERATIVE I/O ERRORS                       PMN02790
                    0281    *                                                        PMN02800
0028 0  0000        0282    $PRET DC      *-*        ENTRY POINT                     PMN02810
0029 0  3000        0283          WAIT              WAIT TIL START KEY PUSHED       PMN02820
002A 00 4C800028    0284          BSC   I  $PRET    RETURN TO CALLER                PMN02830
                    0285    *                                                        PMN02840
                    0286    *                                                        PMN02850
002C 0  0000        0287    $IREQ DC      *-*  ADDR OF INT REQUEST SUBROUTINE      PMN02860
002D 0  0000        0288    $ULET DC      *-*  ADDR OF LET, LOGICAL DR 0           PMN02870
002E 0  0000        0289          DC      *-*  ADDR OF LET, LOGICAL DR 1           PMN02880
002F 0  0000        0290          DC      *-*  ADDR OF LET, LOGICAL DR 2           PMN02890
0030 0  0000        0291          DC      *-*  ADDR OF LET, LOGICAL DR 3           PMN02900
0031 0  0000        0292          DC      *-*  ADDR OF LET, LOGICAL DR 4           PMN02910
0032 0  0000        0293    $IOCT DC      *-*  ZERO IF NO I/O IN PROGRESS     50    PMN02920
0033 0  0000        0294    $LAST DC      *-*  NON-ZERO WHEN LAST CARD SENSED      PMN02930
0034 0  0000        0295    $NDUP DC      *-*  DO NOT DUP IF NON-ZERO              PMN02940
0035 0  0000        0296    $NXFQ DC      *-*  DO NOT EXECUTE IF NON-ZERO          PMN02950
0036 0  0000        0297    $PBSY DC      *-*  NON-ZERO WHEN PRINTER BUSY          PMN02960
0037 0  0000        0298    $PGCT DC      *-*  PAGE NO. FOR HEADINGS               PMN02970
                    0299    *                                                        PMN02980
                    0300    * CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR          PMN02990
                    0301    *                                                        PMN03000
0038 0  7019        0302    $EXIT MDX     $S000     BR TO FETCH CIL, PHASE 1  56     PMN03010
                    0303    *                                                        PMN03020
                    0304    *** CALL LINK ENTRY POINT                              PMN03030
                    0305    *                                                        PMN03040
0039 0  0000        0306    $LINK DC      *-*        ENTRY POINT               57    PMN03050
003A 0  1810        0307          SRA     16                                         PMN03060
003B 0  7017        0308          MDX     $S100     BR TO FETCH CIL, PHASE 1         PMN03070
003C      0000      0309          BSS  E  0                                          PMN03080
003C 0  0001        0310    $S900 DC      1    DISK PARAMETERS FOR SAVING CORE     PMN03090
003D 0  0004        0311          DC      $CIBA-1 *IN CONNECTION WITH DUMP         PMN03100
003E 0  FFFF        0312    $S910 DC      -1   CALL EXIT INDICATOR                 PMN03110
                    0313    *                                                        PMN03120
                    0314    *** SAVE 1ST 4K OF CORE ON THE CIB                      PMN03130
                    0315    *                                                        PMN03140
003F 0  0000        0316    $DUMP DC      *-*        ENTRY POINT               63    PMN03150
0040 0  0809        0317          STD     $ACEX     SAVE ACCUMULATOR, EXTENSION      PMN03180
0041 0  4009        0318          BSI     $S250     CHK PNDNG INTRPT          2-4    PMN03185
0042 0  69D4        0319          STX   1 $CXR1     SAVE XR1                         PMN03190
0043 00 C480003F    0320          LD    I $DUMP                                    PMN03200
0045 0  D003        0321          STO     $DMPF     SAVE DUMP FORMAT CODE            PMN03210
0046 0  C8F5        0322          LDD     $S900                                      PMN03220
0047 00 440000F2    0323          BSI   L DZ000     SAVE WDS 6-4095 ON CIB           PMN03230
0049 0  C0F2        0324          LD      $S900                                      PMN03240
004A 0  7008        0325          MDX     $S100     BR TO FETCH CIL, PHASE 1         PMN03250
                    0326    *                                                        PMN03251
                    0327    *** SUBR TO  CHECK IF ANY INTRPT IS PENDING             PMN03252
                    0328    *                                                        PMN03253
```

```
ADDR REL OBJECT   ST.NO.  LABEL OPCD FT OPERANDS                                  ID/SEQNO

004B 0  0000      0329    $S250 DC      *-*       ENTRY POINT                     PMN03254
004C 0  C0E5      0330    $S300 LD      $IOCT     IS THERE INTRPT PNDNG           PMN03255
004D 0  E8C3      0331          OR      $SCAT     *OR SCA INTRPT PNDNG            PMN03256
004E 00 4C20004C  0332          BSC  L  $S300,Z   *THEN BR, IF ALL INTRPT         PMN03257
0050 00 4C800048  0333          RSC  I  $S250     *IS SERVICED-RETURN             PMN03258
                  0335    *                                                       PMN03270
                  0336    *** FETCH CORE IMAGE LOADER, PHASE 1                    PMN03280
                  0337    *                                                       PMN03290
0052 0  C0E8      0338    $S000 LD      $S910                                     PMN03300
0053 0  D0C2      0339    $S100 STD     $RMSW     SAVE EXIT-LINK-DUMP SWITCH      PMN03310
0054 00 65800039  0340          LDX   I1 $LINK     LINK ADDR TO XR1                PMN03350
0056 0  C101      0341          LD    1  1        FETCH 2ND WD OF LINK NAME       PMN03360
0057 0  1BD0      0342          RTE      16                                       PMN03370
0058 0  C100      0343          LD    1  0        FETCH 1ST WD OF LINK NAME       PMN03380
                  0344    * $S150+1 CONTAINS ADDR LAST WD OF DISK I/O MINUS 3     PMN03400
0059 00 65000000  0345    $S150 LDX  L1 *-*       ADDR END OF DKI/O-1 TO XR1      PMN03410
005B 0  D888      0346          STD     $LKNM     SAVE LINK NAME                  PMN03415
005C 0  40EE      0347          BSI     $S250     CHK ANY PNONG INTRPT     2-4    PMN03417
005D 0  C0FC      0348          LD      $CILA                                     PMN03420
005E 0  1890      0349    $S200 SRT     16                                        PMN03430
005F 00 440000F2  0350          BSI  L  DZ000     FETCH CI LOADER, PHASE 1        PMN03440
0061 0  40E9      0351          BSI     $S250     CHK DISK OP FINISHED     2-4    PMN03460
0062 0  4102      0352          BSI   1 2         BR TO CI LOADER, PHASE 1        PMN03470
                  0353    *                                                       PMN03480
0063 0  0000      0354    $GCOM DC      *-*  GRAPHIC SUBR PACKAGE INDR     2G2   PMN03490
0064 0  0000      0355    $GRIN DC      *-*  GRAPHIC INITLZN PROGRAM INDR  2G2   PMN03500
                  0356    *                                                  2G2  PMN03510
0065      0003    0357          BSS     3         RESERVED FOR 2250       2G2   PMN03520
0068      0009    0358          BSS     9         PATCH AREA                      PMN03530
                  0359    *                                                       PMN03540
0071 0  0000      0360    $FLSH DC      *-*  FLUSH-TO-NEXT-JOB SWITCH 1#FLUSH    PMN03550
0072      0000    0361          BSS  E  0                                        PMN03560
0072 0  0000      0362    $CWCT DC      *-*  WORD COUNT AND SECTOR ADDRESS       PMN03570
0073 0  0000      0363          DC      *-*  *FOR SAVING/RESTORING COMMON        PMN03580
0074 0  0000      0364    $CCAD DC      *-*  ADDR FOR SAVING/RESTORING COMMON    PMN03590
0075 0  0000      0365    $LSAD DC      *-*  SCTR ADDR OF 1ST LOCAL/SOCAL        PMN03600
0076 0  0000      0366    $DZIN DC      *-*  DISKZ/1/N INDICATOR (-1,0,+1)       PMN03610
0077 0  0000      0367    $DCDE DC      *-*  LOGICAL DRIVE CODE FOR PROGRAM      PMN03620
0078 0  0000      0368    $PHSE DC      *-*  NO. OF PHASE NOW IN CORE            PMN03630
0079 0  0000      0369    $UFIO DC      *-*  UNFORMATTED I/O RECORD NO.          PMN03640
007A 0  0000      0370    $WSDR DC      *-*  WORKING STORAGE DRIVE CODE          PMN03650
007B 0  0000      0371    $WRD1 DC      *-*  LOADING ADDR OF THE CORE LOAD       PMN03660
007C 0  0000      0372    $KCSW DC      *-*  1 IF KB,CP BOTH UTILIZED            PMN03670
007D 0  0000      0373    $UFDR DC      *-*  UNFORMATTED I/O DRIVE CODE          PMN03680
007E 0  0000      0374    $CPTR DC      *-*  CHANNEL 12 INDICATOR FOR CP         PMN03690
007F 0  0000      0375    $1132 DC      *-*  CHANNEL 12 INDICATOR FOR 1132       PMN03700
0080 0  0000      0376    $1403 DC      *-*  CHANNEL 12 INDICATOR FOR 1403       PMN03710
                  0378    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 1          PMN03730
                  0379    *                                                       PMN03740
0081 0  0000      0380    $PST1 DC      *-*       ENTRY POINT                     PMN03750
0082 0  3000      0381          WAIT                                              PMN03760
0083 00 4C800081  0382          BSC  I  $PST1     RETURN TO DEVICE SUBROUTINE     PMN03770
                  0383    *                                                       PMN03780
                  0384    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 2          PMN03790
                  0385    *                                                       PMN03800
0085 0  0000      0386    $PST2 DC      *-*       ENTRY POINT                     PMN03810
0086 0  3000      0387          WAIT                                              PMN03820
0087 00 4C800085  0388          BSC  I  $PST2     RETURN TO DEVICE SUBROUTINE     PMN03830
                  0389    *                                                       PMN03840
                  0390    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 3          PMN03850
                  0391    *                                                       PMN03860
0089 0  0000      0392    $PST3 DC      *-*       ENTRY POINT                     PMN03870
008A 0  3000      0393          WAIT                                              PMN03880
008B 00 4C800089  0394          BSC  I  $PST3     RETURN TO DEVICE SUBROUTINE     PMN03890
                  0395    *                                                       PMN03900
                  0396    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 4          PMN03910
                  0397    *                                                       PMN03920
008D 0  0000      0398    $PST4 DC      *-*       ENTRY POINT                     PMN03930
008E 0  3000      0399          WAIT                                              PMN03940
008F 00 4C80008D  0400          BSC  I  $PST4     RETURN TO DEVICE SUBROUTINE     PMN03950
                  0401    *                                                       PMN03960
                  0402    *                                                       PMN03970
                  0403    * PROGRAM STOP KEY TRAP                                 PMN03980
                  0404    *                                                       PMN03990
0091 0  0000      0405    $STOP DC      *-*       ENTRY POINT                     PMN04000
0092 0  3000      0406          WAIT              WAIT TIL START KEY PUSHED       PMN04010
0093 00 4CC00091  0407          BOSC I  $STOP     RETURN TO CALLER                PMN04020
```

```
ADDR REL OBJECT    ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                   0409     *                                                         PMN04040
                   0410     * PARAMETERS USED BY THE DISK I/O SUBROUTINES. THE       PMN04050
                   0411     * LOGICAL DRIVE CODE IS FOUND IN BITS 1-3 FOR ALL       PMN04060
                   0412     * BUT THE AREA CODE. BIT 0 WILL ALWAYS BE ZERO.         PMN04070
                   0413     *                                                         PMN04080
                   0414     *                                                         PMN04090
                   0415     *** DISK1 AND DISKN WILL NOT WRITE BELOW THE            PMN04100
                   0416     *** FOLLOWING SCTR ADDRESSES (EXCEPT WRITE IMMED).      PMN04110
                   0417     *                                                         PMN04120
0095 0  0000       0418     $FPAD DC      *-*  FILE PROTECT ADDR, LOGICAL DR 0      PMN04130
0096 0  0000       0419           DC      *-*  FILE PROTECT ADDR, LOGICAL DR 1      PMN04140
0097 0  0000       0420           DC      *-*  FILE PROTECT ADDR, LOGICAL DR 2      PMN04150
0098 0  0000       0421           DC      *-*  FILE PROTECT ADDR, LOGICAL DR 3      PMN04160
0099 0  0000       0422           DC      *-*  FILE PROTECT ADDR, LOGICAL DR 4      PMN04170
                   0423     *                                                         PMN04180
                   0424     *** THE ARM POSITION IS UPDATED WHENEVER A SEEK         PMN04190
                   0425     *** OCCURS.                                             PMN04200
                   0426     *                                                         PMN04210
009A 0  0000       0427     $CYLN DC      0    ARM POSITION FOR LOGICAL DRIVE 0    PMN04220
009B 0  0000       0428           DC      0    ARM POSITION FOR LOGICAL DRIVE 1    PMN04230
009C 0  0000       0429           DC      0    ARM POSITION FOR LOGICAL DRIVE 2    PMN04240
009D 0  0000       0430           DC      0    ARM POSITION FOR LOGICAL DRIVE 3    PMN04250
009E 0  0000       0431           DC      0    ARM POSITION FOR LOGICAL DRIVE 4    PMN04260
                   0432     *                                                         PMN04270
                   0433     *** BELOW ARE THE DISK AREA CODES. A ZERO               PMN04280
                   0434     *** INDICATES THE CORRESPONDING DRIVE IS NOT            PMN04290
                   0435     *** ON THE SYSTEM                                       PMN04300
                   0436     *                                                         PMN04310
009F 0  0000       0437     $ACDE DC      *-*  AREA  CODE FOR LOGICAL DRIVE 0      PMN04320
00A0 0  0000       0438           DC      *-*  AREA  CODE FOR LOGICAL DRIVE 1      PMN04330
00A1 0  0000       0439           DC      *-*  AREA  CODE FOR LOGICAL DRIVE 2      PMN04340
00A2 0  0000       0440           DC      *-*  AREA  CODE FOR LOGICAL DRIVE 3      PMN04350
00A3 0  0000       0441           DC      *-*  AREA  CODE FOR LOGICAL DRIVE 4      PMN04360
                   0442     *                                                         PMN04370
                   0443     *** THE ADR OF THE CYLINDER IN WHICH A DEFECT OC-       PMN04380
                   0444     *** CURS, IF ANY, IS STORED IN THE 1ST, 2ND, OR 3RD     PMN04390
                   0445     *** WORD BELOW, DEPENDING ON WHETHER IT IS THE 1ST,     PMN04400
                   0446     *** 2ND, OR 3RD DEFECT ON THE CARTRIDGE.                PMN04410
                   0447     *                                                         PMN04420
00A4 0  0000       0448     $DCYL DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1    PMN04430
00A5 0  0000       0449           DC      *-*  *FOR LOGICAL DRIVE 0            2    PMN04440
00A6 0  0000       0450           DC      *-*                                 3    PMN04450
00A7 0  0000       0451           DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1    PMN04460
00A8 0  0000       0452           DC      *-*  *FOR LOGICAL DRIVE 1            2    PMN04470
00A9 0  0000       0453           DC      *-*                                 3    PMN04480
00AA 0  0000       0454           DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1    PMN04490
00AB 0  0000       0455           DC      *-*  *FOR LOGICAL DRIVE 2            2    PMN04500
00AC 0  0000       0456           DC      *-*                                 3    PMN04510
00AD 0  0000       0457           DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1    PMN04520
00AE 0  0000       0458           DC      *-*  *FOR LOGICAL DRIVE 3            2    PMN04530
00AF 0  0000       0459           DC      *-*                                 3    PMN04540
00B0 0  0000       0460           DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1    PMN04550
00B1 0  0000       0461           DC      *-*  *FOR LOGICAL DRIVE 4            2    PMN04560
00B2 0  0000       0462           DC      *-*                                 3    PMN04570
                   0464     *                                                         PMN04590
                   0465     * ILS02--THIS SUBROUTINE SAVES XR1, XR2, STATUS,        PMN04600
                   0466     *         AND THE ACCUMULATOR AND ITS EXTENSION.        PMN04610
                   0467     *         THE ADDRESS OF THE INTERRUPT SERVICE ROU-     PMN04620
                   0468     *         TINE IS STORED IN $I205 BY PHASE 2 OF         PMN04630
                   0469     *         THE CORE IMAGE LOADER. WORD 10 ALWAYS         PMN04640
                   0470     *         CONTAINS THE ADDRESS OF $I200.                PMN04650
                   0471     *                                                         PMN04660
                   0472     *                                                         PMN04670
                   0473     *                                                         PMN04680
00B3 0  0000       0474     $I200 DC      *-*        ENTRY PT (LEVEL 2 INTRUPT)   PMN04690
00B4 0  6906       0475           STX   1 $I210+1    SAVE XR1                     PMN04700
00B5 0  6A07       0476           STX   2 $I210+3    SAVE XR2                     PMN04710
00B6 0  2807       0477           STS     $I210+4    STORE STATUS                 PMN04720
00B7 0  D80A       0478           STD     $I290      SAVE ACCUMULATOR,EXTENSION   PMN04730
                   0479     * $I205+1 CONTAINS ADDR INTERRUPT ENTRY PT TO DKI/O    PMN04740
00B8 00 44000000   0480     $I205 BSI   L  *-*        BR TO SERVICE THE INTERRUPT  PMN04750
00BA 00 65000000   0481     $I210 LDX   L1 *-*        RESTORE XR1                  PMN04760
00BC 00 66000000   0482           LDX   L2 *-*        RESTORE XR2                  PMN04770
00BE 0  2000       0483           LDS     0          RESTORE STATUS               PMN04780
00BF 0  C802       0484           LDD     $I290      RESTORE ACCUMULATOR,EXT      PMN04790
00C0 00 4CC00083   0485           BOSC I  $I200      RETURN FROM INTERRUPT        PMN04800
00C2       0000    0486     $I290 BSS   E  0                                       PMN04810
00C2 0  0000       0487           DC      *-*        CONTENTS OF ACCUMULATOR AND  PMN04820
00C3 0  0000       0488           DC      *-*        *EXTENTION                   PMN04830
```

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  0490     *                                                        PMN04850
                  0491     * ILS04--THIS SUBROUTINE SAVES XR1, XR2, STATUS,        PMN04860
                  0492     *         AND THE ACCUMULATOR AND ITS EXTENSION.         PMN04870
                  0493     *         IF THE INTERRUPT IS FOR A KEYBOARD REQ-  *     PMN04880
                  0494     *         UEST,AND IF A MONITOR PROGRAM IS IN CON- *     PMN04890
                  0495     *         TROL, CONTROL IS PASSED TO DUMP. OTHER-  *     PMN04900
                  0496     *         WISE, CONTROL IS PASSED TO THE KEYBOARD/ *     PMN04910
                  0497     *         CONSOLE PRINTER SUBROUTINE. WORD 12 AL-  *     PMN04920
                  0498     *         WAYS CONTAINS THE ADDRESS OF $I400.      *     PMN04930
                  0499     *                                                        PMN04940
                  0500     * THE TABLE BELOW CONTAINS THE ADDRESSES OF THE         PMN04950
                  0501     * INTERRUPT SERVICE ROUTINES FOR ALL THE DEVICES       PMN04960
                  0502     * ON LEVEL 4.                                          PMN04970
                  0503     *                                                        PMN04980
                  0504     *                                                        PMN04990
                  0505     *                                                        PMN05000
00C4 0  0000      0506     $I400 DC      *-*        ENTRY POINT                   PMN05010
00C5 0  D818      0507           STD     $I490      SAVE ACCUMULATOR, EXTENSION   PMN05020
00C6 0  280E      0508           STS     $I410      SAVE STATUS                   PMN05030
00C7 0  690F      0509           STX   1 $I410+2    SAVE XR1                      PMN05040
00C8 0  6A10      0510           STX   2 $I410+4    SAVE XR2                      PMN05050
00C9 0  0816      0511           XIO     $I492      SENSE DSW                     PMN05060
00CA 0  1002      0512           SLA     2          IS THIS INTERRUPT REQUEST     PMN05070
00CB 00 4C100000  0513           BSC   L $I403,-    BR IF NOT INTERRUPT REQUEST   PMN05080
00CD 00 4480002C  0514           BSI   I $IREQ      BR IF INTERRUPT REQUEST       PMN05090
00CF 0  FFFE      0515           DC      -2         ERROR CODE                    PMN05100
00D0 0  6109      0516     $I403 LDX   1 9          NO. DEVICES ON LEVEL TO XR1   PMN05110
00D1 0  0810      0517           XIO     $I494      SENSE ILSW                    PMN05120
00D2 0  1140      0518           SLCA  1            FIND CAUSE OF INTERRUPT       PMN05130
                  0519     * $I405+1 CONTAINS ADDR OF LEVEL 4 IBT MINUS 1         PMN05140
00D3 00 45800000  0520     $I405 BSI   I1 *-*       BR TO SERVICE THE INTERRUPT   PMN05150
00D5 0  2000      0521     $I410 LDS     0          RESTORE STATUS                PMN05160
00D6 00 65000000  0522           LDX   L1 *-*       RESTORE XR1                   PMN05170
00D8 00 66000000  0523           LDX   L2 *-*       RESTORE XR2                   PMN05180
00DA 0  C803      0524           LDD     $I490      RESTORE ACCUMULATOR, EXT.     PMN05190
00DB 00 4CC000C4  0525           BOSC I  $I400      RETURN                        PMN05200
                  0526     *                                                        PMN05210
                  0527     * CONSTANTS AND WORK AREAS                             PMN05220
                  0528     * EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES          PMN05230
                  0529     *                                                        PMN05240
00DD 0  0000      0530     $DDSW DC      *-*        DSW FOR THE DISK              PMN05250
00DF    0002      0531     $I490 BSS   E 2     CONTENTS OF ACCUMULATOR, EXT.      PMN05260
00E0 0  0000      0532     $I492 DC      *-*                                      PMN05270
00E0 0            0533     $SYSC EQU     *-1  VERSION AND MOD NO.                 PMN05280
00E1 0  0F00      0534           DC      /0F00 IOCC FOR SENSE IOCC FOR KB/CP      PMN05290
00E2    0001      0535     $I494 BSS     1     PATCH AREA                         PMN05300
00E3 0  0300      0536           DC      /0300 IOCC FOR SENSING ILSW04            PMN05310
                  0538     *                                                2-2 PMN05330
                  0539     *     PATCH AREA                                 2-2 PMN05340
                  0540     *     FIX FOR APAR N5044                         2-2 PMN05350
                  0541     *                                                2-2 PMN05360
00E4 0  0000      0542     $I496 DC      *-*        XR3 SETTING DURING XEQ  2-2 PMN05370
00E5 0  0F01      0543           DC      /0F01      SENSE KEY BOARD W RESET 2-2 PMN05380
                  0544     *                                                2-2 PMN05390
00E6 0  0000      0545     $I420 DC      *-*        ENTRY POINT FLUSH JOB   2-2 PMN05400
00E7 0  08FC      0546           XIO     $I496      SENSE KEY BOARD W RESET 2-2 PMN05410
00E8 00 4C4000EA  0547           BOSC L  $I425      TURN OF INTERRUPT       2-2 PMN05420
00EA 00 4400003F  0548     $I425 BSI   L  $DUMP     BRANCH TO WAIT OUT PEND 2-2 PMN05430
00EC 0  FFFE      0549           DC      -2         *INTER AND GET AUX SUP  2-2 PMN05440
                  0550     *                                                2-2 PMN05450
00ED    0001      0551           BSS     1          PATCH AREA              2-2 PMN05460
00EE 0  0000      0552     $DBSY DC      *-*  NON-ZERO WHEN DISK I/O BUSY          PMN05470
```

DISKZ

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  0554     ************************************************** PMN05490
                  0555     *                                                * PMN05500
                  0556     *STATUS-VERSION 2, MODIFICATION 1                * PMN05510
                  0557     *                                                * PMN05520
                  0558     *PROGRAM NAME-                                   * PMN05530
                  0559     *   *FULL NAME-FORTRAN/SYSTEM DISK I/O SUBROUTINE * PMN05540
                  0560     *   *CALLING SEQUENCE-                           * PMN05550
                  0561     *      LDD     PARAM                             * PMN05560
                  0562     *      BSI  L  DZ000                             * PMN05570
                  0563     *    WHERE PARAM IS THE LABEL OF A DOUBLE-WORD   * PMN05580
                  0564     *    CELL CONTAINING THE FUNCTION CODE AND THE   * PMN05590
                  0565     *    ADDR OF THE I/O BUFFER,I.E., ADDR OF WD CNT. * PMN05600
                  0566     *    SEE 'CAPABILITIES' FOR DISCUSSION OF PARAM-  * PMN05610
                  0567     *    ETERS.                                      * PMN05620
                  0568     *                                                PMN05630
```

```
ADDR REL OBJECT   ST.NO.  LABEL OPCD FT OPERANDS                                    ID/SEQNO
                  0569    *PURPOSE-                                                * PMN05640
                  0570    *   TO PROVIDE A SUBROUTINE TO PERFORM DISK OPERA-*      PMN05650
                  0571    *   TIONS. THIS SUBROUTINE IS INTENDED FOR USE BY *      PMN05660
                  0572    *   MONITOR PROGRAMS AND USER PROGRAMS WRITTEN IN *      PMN05670
                  0573    *   FORTRAN. THUS,IT IS INTENDED FOR USE IN AN    *      PMN05680
                  0574    *   ERROR-FREE ENVIRONMENT.                       *      PMN05690
                  0575    *                                                 *      PMN05700
                  0576    *METHOD-                                          *      PMN05710
                  0577    *   DISKZ REQUIRES A BUFFER,THE LENGTH OF WHICH IS*      PMN05720
                  0578    *   2 GREATER THAN THE NO. WORDS TO BE READ/WRIT- *      PMN05730
                  0579    *   TEN.                                          *      PMN05740
                  0580    *                                                 *      PMN05750
                  0581    *CAPABILITIES AND LIMITATIONS-                    *      PMN05760
                  0582    *   THE WD CNT,AS WELL AS DZ000,MUST BE ON AN EVEN*      PMN05770
                  0583    *   BOUNDARY,MUST BE IN THE RANGE 0-32767. THE    *      PMN05780
                  0584    *   DRIVE CODE MUST BE IN BITS 1-3 OF THE SECTOR  *      PMN05790
                  0585    *   ADDR,WHICH FOLLOWS THE WD CNT. THE FUNCTION   *      PMN05800
                  0586    *   INDICATOR MUST BE XX00 FOR A READ OR XX01 FOR *      PMN05810
                  0587    *   A WRITE,WHERE 'XX' MEANS ANY 2 HEXADECIMAL    *      PMN05820
                  0588    *   CHARACTERS. A WD CNT OF ZERO INDICATES A SEEK.*      PMN05830
                  0589    *   (READ OR WRITE MAY BE INDICATED.) AUTOMATIC   *      PMN05840
                  0590    *   SEEKING IS PROVIDED AS A PART OF READ/WRITE.  *      PMN05850
                  0591    *   A WRITE IS ALWAYS WITH A READ-BACK-CHECK.     *      PMN05860
                  0592    *   DISKZ MAKES NO PREOPERATIVE PARAMETER CHECKS. *      PMN05870
                  0593    *                                                 *      PMN05880
                  0594    *SPECIAL FEATURES-                                *      PMN05890
                  0595    *   DISKZ PROVIDES ONLY THOSE FUNCTIONS MENTIONED *      PMN05900
                  0596    *   ABOVE. DISK1 AND DISKN OFFER THIS BASIC SET OF*      PMN05910
                  0597    *   FUNCTIONS PLUS OTHERS.                        *      PMN05920
                  0598    *                                                 *      PMN05930
                  0599    ***************************************************      PMN05940
                  0601    * PROVIDE PARAMETERS FOR SYSTEM LOADER                   PMN05960
                  0602    *                                                       PMN05970
00F0      0000    0603          BSS  E  0                                         PMN05980
00F0 0    00EF    0604          DC      $ZEND-*     DISKZ WORD COUNT              PMN05990
00F1 0    FF6A    0605          DC      -'DZID      PHASE ID                      PMN06000
00F2 0    00E8    0606          DC      $ZEND-6-*+1 ADDR OF SLET EXTRACT          PMN06010
00F3 0    0001    0607          DC      1           NO. ENTRIES IN SLET EXTRACT   PMN06020
00F4              0608          ORG     *-2                                       PMN06030
00F2 0    0000    0610    DZ000 DC      *-*         ENTRY POINT                   PMN06050
00F3 00 740000EE  0611          MDX  L  $DBSY,0     LOOP UNTIL OPERATION IN       PMN06060
00F5 0    70FD    0612          MDX     *-3         *PROGRESS IS COMPLETE         PMN06070
00F6 0    7002    0613          MDX     DZ020       BR AROUND INT ENTRY POINT     PMN06080
                  0614    *                                                       PMN06090
                  0615    * INTERRUPT ENTRY POINT                                 PMN06100
                  0616    *                                                       PMN06110
00F7 0    0000    0617    DZ010 DC      *-*         INTERRUPT ADDRESS             PMN06120
00F8 0    7015    0618          MDX     DZ180       BR TO SERVICE INTERRUPT       PMN06130
00F9 0    690F    0619    DZ020 STX   1 DZ100+1     SAVE XR1                      PMN06140
00FA 0    6A10    0620          STX   2 DZ100+3     SAVE XR2                      PMN06150
00FB 0    1008    0621          SLA     8           SHIFT INDICATOR 8 BITS        PMN06160
00FC 0    D03C    0622          STO     DZ945       SAVE FUNCTION INDICATOR       PMN06170
00FD 0    1800    0623          RTE     16                                        PMN06180
00FE 0    0056    0624          STO     DZ235+1     SAVE ADDR OF THE I/O AREA     PMN06190
00FF 0    6211    0625    DZ030 LDX   2 'TCNT       TURN BUSY INDICATOR ON AND    PMN06200
0100 0    6AED    0626          STX   2 $DBSY       *SET RETRY COUNT              PMN06210
0101 0    C0F0    0627          LD      DZ000                                     PMN06220
0102 0    D0F4    0628          STO     DZ010                                     PMN06230
0103 0    704E    0629          MDX     DZ230       BR TO CONTINUE                PMN06240
0104 00 4C000000  0630    DZ060 BSC  L  *-*         BR TO SERVICE THE INTERRUPT   PMN06250
                  0631    *                                                       PMN06260
                  0632    * START ALL DISK OPERATIONS                             PMN06270
                  0633    *                                                       PMN06280
0106 0    6908    0634    DZ070 STX   1 DZ180+1     SAVE ADDR OF THE I/O AREA     PMN06290
0107 0    081E    0635          XIO     DZ904       START AN OPERATION            PMN06300
                  0636    *                                                       PMN06310
                  0637    * RETURN TO USER                                        PMN06320
                  0638    *                                                       PMN06330
0108 00 65000000  0639    DZ100 LDX  L1 *-*         RESTORE XR1                   PMN06340
010A 00 66000000  0640          LDX  L2 *-*         RESTORE XR2                   PMN06350
010C 00 4C8000F7  0641          BSC  I  DZ010       RETURN                        PMN06360
                  0642    *                                                       PMN06370
                  0643    * SERVICE ALL INTERRUPTS                                PMN06380
                  0644    *                                                       PMN06390
010E 00 65000000  0645    DZ180 LDX  L1 *-*         ADDR OF I/O AREA TO XR1       PMN06400
0110 00 660000F2  0646          LDX  L2 DZ000       ADDR OF DZ000 TO XR2          PMN06410
0112 0    0819    0647          XIO     DZ910       SENSE THE DSW                 PMN06420
0113 0    00C9    0648          STO     $DDSW       SAVE THE DSW                  PMN06430
0114 0    4850    0649          BOSC    -           SKIP IF ERROR BIT SET         PMN06440
0115 0    70EE    0650          MDX     DZ060       BRANCH IF ERROR BIT NOT SET   PMN06450
0116 0    C800    0651    DZ185 LCD     DZ902       RESTORE WORD COUNT            PMN06460
0117 0    0900    0652          STD   1 0           *AND SECTOR ADDRESS           PMN06470
```

```
ADDR REL OBJECT   ST.NO.  LABEL OPCD FT OPERANDS                              ID/SEQNO
0118 00 74FF00EE  0653          MDX  L  $DBSY,-1  SKIP IF 16 RETRIES DONE     PMN06480
011A 0  7039      0654          MDX     DZ235     BRANCH IF LESS THAN 16     PMN06490
                  0655    *                                                  PMN06500
                  0656    * TRAP OUT TO POSTOPERATIVE TRAP                   PMN06510
                  0657    *                                                  PMN06520
011B 0  C812      0658          LDD     DZ912     1+SCTR ADDR TO EXTENSION   PMN06530
011C 0  C014      0659          LD      DZ915                                PMN06540
011D 0  4293      0660    DZ190 BSI   2 $PST2-X2  BR TO POSTOPERATIVE ER TRAP PMN06550
011E 0  1810      0661          SRA     16        CLEAR                      PMN06560
011F 00 04800191  0662          STO  I  DZ350+1   *ARM POSITION              PMN06570
0121 0  70DD      0663          MDX     DZ030     RETRY OPERATION            PMN06580
                  0664    *                                                  PMN06590
                  0665    * CONSTANTS AND WORK AREAS                         PMN06600
                  0666    *                                                  PMN06610
0122      0000    0667          BSS  F  0                                    PMN06620
                  0668    * EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES      PMN06630
0122 0  0001      0669    DZ900 DC      1     CONSTANT,READ-AFTER-SEEK WD CNT PMN06640
0123 0  0000      0670    DZ901 DC      0     CURRENT ARM POSITION           PMN06650
0124 0  0000      0671    DZ902 DC      *-*   LAST TWO WORDS OF SECTOR       PMN06660
0125 0  0000      0672          DC      *-*   *PREVIOUSLY READ               PMN06670
0126 0  0000      0673    DZ904 DC      *-*   IOCC FOR OPERATION CURRENTLY   PMN06680
0127 0  0000      0674    DZ905 DC      *-*   *BEING PERFORMED               PMN06690
0128 0  0000      0675    DZ906 DC      *-*   SAVE AREA FOR IOCC FOR         PMN06700
0129 0  0000      0676    DZ907 DC      *-*   *USER-REQUESTED OPERATION      PMN06710
012A 0  0122      0677    DZ908 DC      DZ900 IOCC FOR READ                  PMN06720
012B 0  0000      0678    DZ909 DC      *-*   *AFTER SEEK                    PMN06730
012C 0  0000      0679    DZ910 DC      *-*   2ND WORD OF SEEK IOCC          PMN06740
012D 0  0000      0680    DZ911 DC      *-*   SENSE IOCC                     PMN06750
012E 0  0000      0681    DZ912 DC      *-*   INTERMEDIATE WORD COUNT        PMN06760
012F 0  0000      0682    DZ913 DC      *-*   ADDR OF NEXT SEQUENTIAL SECTOR PMN06770
0130 0  5002      0683    DZ914 DC      /5002 WRITE SELECT/POWER UNSAFE INDR PMN06780
0131 0  5004      0684    DZ915 DC      /5004 READ/WRITE/SEEK ERROR INDICATOR PMN06790
0132 0  FEC0      0685    DZ916 DC      -320 TO BE USED TO SIMULTANEOUSLY    PMN06800
0133 0  0001      0686          DC      1     *DECR WD CNT, INCR SCTR ADDR   PMN06810
0134 0  0080      0687    DZ920 DC      /0080 READ CHECK BIT FOR IOCC        PMN06820
0135 0  0600      0688    DZ925 DC      /0600 2ND WD OF READ IOCC W/O AREA CD PMN06830
0136 0  0008      0689    DZ930 DC      8     NO. SECTORS PER CYLINDER       PMN06840
0137 0  5000      0690    DZ935 DC      /5000 NOT READY DISPLAY CODE         PMN06850
0138 0  0FF8      0691    DZ940 DC      /0FF8 'AND' OUT DR CODE, SCTR ADDR   PMN06860
0139 0  0000      0692    DZ945 DC      *-*   FUNC INDICATOR (0#READ,1#WRITE) PMN06870
013A 0  0701      0693    DZ950 DC      /0701 SENSE IOCC W/O AREA CODE       PMN06880
013B 0  0007      0694    DZ955 DC      /0007 'AND' OUT ALL BUT SCTR NO.     PMN06890
013C 0  000A      0695    DZ960 DC      $DCYL-$CYLN BASE DEFECTIVE CYL ADDR  PMN06900
013D 0  009F      0696    DZ965 DC      $ACDE BASE AREA CODE ADDR            PMN06910
013E 0  FFF8      0697    DZ970 DC      $CYLN-$ACDE BASE ARM POSITION ADDR   PMN06920
013F 0  0000      0698    DZ975 DC      *-*   2ND WORD OF READ CHECK IOCC    PMN06930
0140 0  0400      0699    DZ980 DC      /0400 2ND WD OF SEEK IOCC W/O AREA CD PMN06940
0141 0  0141      0700    DZ985 DC      321   NO. WORDS PER SECTOR (W/ ADDR) PMN06950
0142 0  0000      0701    DZ990 DC      *-*   CURRENT SECTOR NO.             PMN06960
0143 0  FFFF      0702    DZ995 DC      -1    MASK FOR COMPLEMENTING         PMN06970
                  0703    *                                                  PMN06980
                  0704    * RESERVED FOR SAVING CORE ON A DUMP ENTRY TO SKEL PMN06990
                  0705    *                                                  PMN07000
0144      0007    0706          BSS     7     THIS AREA MUST BE AT $CIBA+319 PMN07010
00F2 0            0707    X2    EQU     DZ000                                PMN07020
                  0708    *                                                  PMN07030
                  0709    *                                                  PMN07040
                  0710    *                                                  PMN07050
0146 0  1810      0711    DZ210 SRA     16                                   PMN07060
0147 0  00A6      0712          STO     $DBSY     CLEAR BUSY INDICATOR       PMN07070
0148 00 74FF0032  0713          MDX  L  $IOCT,-1  DECREMENT IOCS COUNTER     PMN07080
014A 0  1000      0714          NOP                                          PMN07090
014B 0  70BC      0715          MDX     DZ100     TO EXIT                    PMN07100
                  0716    *                                                  PMN07110
                  0717    * PREPARE TO TRAP OUT ON 'POWER UNSAFE' CONDITION  PMN07120
                  0718    *                                                  PMN07130
014C 0  C0E3      0719    DZ215 LD      DZ914                                PMN07140
014D 0  70CF      0720          MDX     DZ190     BR TO TPAP OUT             PMN07150
                  0721    *                                                  PMN07160
                  0722    * PREPARE TO TRAP OUT ON 'NOT READY' CONDITION      PMN07170
                  0723    *                                                  PMN07180
014E 0  C0E8      0724    DZ220 LD      DZ935     FETCH ERROR CODE           PMN07190
014F 00 44000028  0725          BSI  L  $PRET     BR TO PREOPERATIVE ERR TRAP PMN07200
0151 0  7036      0726          MDX     DZ340     RETRY THE OPERATION        PMN07210
                  0727    *                                                  PMN07220
                  0728    *                         STATEMENTS MOVED      2-1 PMN07230
                  0729    *                                                  PMN07240
0152 00 74010032  0730    DZ230 MDX  L  $IOCT,1   INCREMENT IOCS COUNTER     PMN07250
0154 00 65000000  0731    DZ235 LDX  L1 *-*       ADDR I/O AREA TO XR1       PMN07260
0156 0  C900      0732          LDD   1 0                                    PMN07270
0157 0  D8CC      0733          STD     DZ902     SAVE WORD COUNT, SCTR ADDR PMN07280
0158 0  D8D5      0734          STD     DZ912                                PMN07290
```

```
ADDR REL OBJECT   ST.NO.  LABEL OPCD FT OPERANDS                                  ID/SEQNO

0159 0  1810      0735    DZ240 SRA     16                                        PMN07300
015A 0  1084      0736          SLT     4         DRIVE CODE IN BITS 12-15        PMN07310
015B 0  D00E      0737          STO     DZ280+1                                   PMN07320
015C 0  80E0      0738          A       DZ965     COMPUTE AND STORE THE           PMN07330
015D 0  D01C      0739          STO     DZ330+1   *ADDR OF THE AREA CODE          PMN07340
015E 0  80DF      0740          A       DZ970     COMPUTE AND STORE THE           PMN07350
015F 0  0031      0741          STO     DZ350+1   *ADDR OF THE ARM POSITION       PMN07360
0160 0  8008      0742          A       DZ960     ADD IN BASE OT ADDR             PMN07370
0161 0  8008      0743          A       DZ280+1   ADD IN THE DRIVE                PMN07380
0162 0  8007      0744          A       DZ280+1   *CODE TWICE MORE                PMN07390
0163 0  D006      0745          STO     DZ280+1                                   PMN07400
0164 0  62FD      0746          LDX   2 -3        INITIALIZE COUNTER FOR LOOP     PMN07410
0165 0  69C2      0747          STX   1 DZ906                                     PMN07420
0166 0  C101      0748          LD    1 1         FETCH DESIRED SECTOR ADDR       PMN07430
0167 0  E0D0      0749          AND     DZ940     'AND' OUT SECTOR NO.            PMN07440
0168 0  D101      0750    DZ250 STO   1 1         *AND DRIVE CODE                 PMN07450
0169 00 94000000  0751    DZ280 S     L *-*       SUB DEFECTIVE CYLINDER ADDR     PMN07460
016B 0  4828      0752          BSC     Z+        SKIP IF BAD CYLINDER            PMN07470
016C 0  7007      0753          MDX     DZ300     BR TO CONTINUE PROCESSING       PMN07480
016D 0  C101      0754          LD    1 1                                       PMN07490
016E 0  80C7      0755          A       DZ930     INCREMENT SCTR ADDR BY 8        PMN07500
016F 00 7401016A  0756          MDX   L DZ280+1,1 POINT TO NEXT DEFECTIVE CYL     PMN07510
0171 0  7201      0757          MDX   2 1         SKIP AFTER 3RD PASS             PMN07520
0172 0  70F5      0758          MDX     DZ250     COMPARE W/ NEXT DEF CYL ADR     PMN07530
0173 0  D101      0759          STO   1 1         SCTR ADDR WITH 3 DEF CYL2-4     PMN07535
                  0760    *                                                       PMN07540
                  0761    * CONSTRUCT THE 2ND WORD OF ALL IOCC'S                  PMN07550
                  0762    *                                                       PMN07560
0174 00 660000F2  0763    DZ300 LDX   L2 DZ000      ADDR OF DZ000 TO XR2          PMN07570
0176 0  C23D      0764          LD    2 DZ913-X2    FETCH SECTOR ADDRESS          PMN07580
0177 0  E249      0765          AND   2 DZ955-X2    'AND' OUT ALL BUT SECTOR NO   PMN07590
0178 0  D250      0766          STO   2 DZ990-X2    SAVE SECTOR NO.               PMN07600
0179 00 C4000000  0767    DZ330 LD    L *-*         FETCH AREA CODE               PMN07610
017B 0  EA4F      0768          OR    2 DZ980-X2    'OR' IN SEEK FUNCTION CODE    PMN07620
017C 0  D23A      0769          STO   2 DZ910-X2    SEEK IOCC MINUS DIRECTION     PMN07630
017D 0  EA43      0770          OR    2 DZ925-X2    'OR' IN READ FUNCTION CODE    PMN07640
017E 0  D239      0771          STO   2 DZ909-X2    IOCC FOR READ-AFTER-SEEK      PMN07650
017F 0  EA50      0772          OR    2 DZ990-X2    'OR' IN SECTOR NO.            PMN07660
0180 0  9247      0773          S     2 DZ945-X2    COMPLETE READ/WRITE CODE      PMN07670
0181 0  0237      0774          STO   2 DZ907-X2    2ND WD OF READ/WRITE IOCC     PMN07680
0182 0  EA42      0775          OR    2 DZ920-X2    'OR' IN READ CHECK BIT        PMN07690
0183 0  8247      0776          A     2 DZ945-X2                                  PMN07700
0184 0  D24D      0777          STO   2 DZ975-X2    2ND WD OF READ CHECK IOCC     PMN07710
0185 0  EA48      0778          OR    2 DZ950-X2    'OR' IN SENSE IOCC BITS       PMN07720
0186 0  D23B      0779          STO   2 DZ911-X2    COMPLETED SENSE IOCC          PMN07730
0187 0  CA3C      0780          LOD   2 DZ912-X2    1+SCTR ADDR TO EXTENSION      PMN07740
0188 0  0A3A      0781    DZ340 XIO   2 DZ910-X2    SENSE FOR DISK READY          PMN07750
0189 0  D2EB      0782          STO   2 $DDSW-X2    SAVE THE DSW                  PMN07760
018A 0  4828      0783          BSC     Z+        SKIP UNLESS POWER UNSAFE OR   PMN07770
018B 0  70C0      0784          MDX     DZ215     *WRITE SELECT, BR OTHERWISE   PMN07780
018C 0  1002      0785          SLA     2         BR TO PREOPERATIVE ERR TRAP   PMN07790
018D 0  4828      0786          BSC     Z+        *IF DISK NOT READY, SKIP      PMN07800
018E 0  708E      0787          MDX     DZ220     *OTHERWISE                    PMN07810
                  0788    *                       STATEMENTS REMOVED      2-1   PMN07820
018F 0  C101      0789          LD    1 1         FETCH DESIRED CYLINDER ADDR   PMN07830
0190 00 94000000  0790    DZ350 S     L *-*       SUBTRACT ARM POSITION         PMN07840
0192 0  4818      0791          BSC     +-        SKIP IF SEEK NECESSARY        PMN07850
0193 0  7018      0792          MDX     DZ400     BRANCH TO PERFORM OPERATION   PMN07860
                  0793    *                                                       PMN07870
                  0794    * SEEK                                                  PMN07880
                  0795    *                                                       PMN07890
0194 0  1893      0796          SRT     19        PUT NO. CYLINDERS IN EXT      PMN07900
0195 0  180F      0797          SRA     15        + OR - SIGN TO BIT 15         PMN07910
0196 0  1002      0798          SLA     2         SHIFT SIGN TO BIT 13          PMN07920
0197 0  EA3A      0799          OR    2 DZ910-X2  "OR" IN REMAINDER OF IOCC     PMN07930
0198 0  1800      0800          RTE     16                                      PMN07940
0199 0  4810      0801          BSC     -         SKIP IF SEEK TOWARD HOME      PMN07950
019A 0  7002      0802          MDX     DZ380     BRANCH IF SEEK TOWARD CENTR   PMN07960
019B 0  F251      0803          EOR   2 DZ995-X2  COMPLEMENT NO. CYLS TO BE     PMN07970
019C 0  8230      0804          A     2 DZ900-X2  *SOUGHT TO GET POSITIVE NO.   PMN07980
019D 0  DA34      0805    DZ380 STD   2 DZ904-X2                                PMN07990
019E 0  C2EB      0806          LD    2 $DDSW-X2  FETCH THE DSW             2-1 PMN08000
019F 0  1000      0807          SLA     13                                  2-1 PMN08010
01A0 0  4810      0808          BSC     -                                   2-1 PMN08020
01A1 0  7003      0809          MDX     DZ390                               2-1 PMN08030
01A2 0  C101      0810    DZ385 LD    1 1         FETCH SECTOR ADDR         2-1 PMN08040
01A3 0  1803      0811          SRA     3         CONVERT TO CYLINDER ADDR  2-1 PMN08050
01A4 0  D234      0812          STO   2 DZ904-X2  *AND STORE IN IOCC         2-1 PMN08060
01A5 0  4213      0813    DZ390 BSI   2 DZ070-1-X2 START SEEK                 2-1 PMN08070
```

```
ADDR REL OBJECT   ST.NO.  LABEL OPCD FT OPERANDS                                  ID/SEQNO
                  0814    *                                                       PMN08080
                  0815    * SEEK COMPLETE INTERRUPT PROCESSING                   PMN08090
                  0816    *                                                       PMN08100
01A6 0  CA38      0817          LDD   2 DZ908-X2  SET UP IOCC FOR               PMN08110
01A7 0  DA34      0818          STD   2 DZ904-X2  *READ AFTER SEEK               PMN08120
01A8 0  4213      0819          BSI   2 DZ070-1-X2 START READ-AFTER-SEEK         PMN08130
                  0820    *                                                       PMN08140
                  0821    * READ-AFTER-SEEK COMPLETE INTERRUPT PROCESSING        PMN08150
                  0822    *                                                       PMN08160
01A9 0  C231      0823          LD    2 DZ901-X2  FETCH ADR OF SCTR JUST READ   PMN08170
01AA 00 04800191  0824          STO  I  DZ350+1   UPDATE ARM POSITION            PMN08180
01AC 0  9101      0825          S     1 1         SUB DESIRED SCTR ADDR         PMN08190
01AD 00 4C200116  0826          BSC  L  DZ185,Z   BR IF SEEK UNSUCCESSFUL        PMN08200
                  0827    *                                                       PMN08210
                  0828    *                                                       PMN08220
                  0829    * READ/WRITE                                           PMN08230
                  0830    *                                                       PMN08240
01AF 0  CA3C      0831    DZ400 LDD   2 DZ912-X2  FETCH INTERMEDIATE WD CNT       PMN08250
01B0 0  4808      0832          BSC     +         SKIP, WD CNT NOT EXHAUSTED    PMN08260
01B1 0  7094      0833    DZ410 MDX     DZ210     BRANCH IF READ/WRITE DONE     PMN08270
01B2 0  8A40      0834          AD    2 DZ916-X2  DECREMENT WORD COUNT AND       PMN08280
01B3 0  DA3C      0835          STD   2 DZ912-X2  *INCREMENT SECTOR ADDRESS      PMN08290
01B4 0  4830      0836          BSC     Z-        SKIP IF THIS IS LAST SECTOR   PMN08300
01B5 0  1810      0837          SRA     16        CLEAR ACCUMULATOR             PMN08310
01B6 0  824F      0838          A     2 DZ985-X2  ADD BACK 321 TO WD CNT         PMN08320
01B7 0  D100      0839          STO   1 0         STORE RESULT IN I/O AREA      PMN08330
01B8 0  CA36      0840          LDD   2 DZ906-X2  RESTORE IOCC FOR ORIGINALLY    PMN08340
01B9 0  DA34      0841          STD   2 DZ904-X2  *REQUESTED OPERATION           PMN08350
01BA 0  C101      0842          LD    1 1         ADD SECTOR NO. TO SECTOR      PMN08360
01BB 0  EA50      0843          OR    2 DZ990-X2  *ADDRESS                      PMN08370
01BC 0  D101      0844          STO   1 1                                       PMN08380
01BD 0  4213      0845          BSI   2 DZ070-1-X2 START READ/WRITE OPERATION   PMN08390
                  0846    *                                                       PMN08400
                  0847    * READ/WRITE COMPLETE INTERRUPT PROCESSING             PMN08410
                  0848    *                                                       PMN08420
01BE 0  C24D      0849          LD    2 DZ975-X2  SET UP FOR READ CHECK         PMN08430
01BF 0  D235      0850          STO   2 DZ905-X2                                PMN08440
01C0 0  C247      0851          LD    2 DZ945-X2  FETCH FUNCTION INDICATOR      PMN08450
01C1 0  4820      0852          BSC     Z         SKIP IF READ REQUESTED        PMN08460
01C2 0  4213      0853          BSI   2 DZ070-1-X2 START READ CHECK OPERATION   PMN08470
01C3 0  CA32      0854          LDD   2 DZ902-X2  RESTORE LAST 2 WDS OF SEC-    PMN08480
01C4 0  D900      0855          STD   1 0         *TOR PREVIOUSLY READ          PMN08490
01C5 0  C23C      0856          LD    2 DZ912-X2  FETCH INTERMEDIATE WD CNT     PMN08500
01C6 0  4808      0857          BSC     +         SKIP IF MORE READING/WRTING   PMN08510
01C7 0  70E9      0858          MDX     DZ410     BRANCH IF FINISHED            PMN08520
01C8 00 75000140  0859          MDX   L1 320      POINT XR1 TO NEW I/O AREA     PMN08530
01CA 0  C900      0860          LDD   1 0         SAVE LAST 2 WDS OF SECTOR     PMN08540
01CB 0  DA32      0861          STD   2 DZ902-X2  *JUST READ/WRITTEN            PMN08550
01CC 0  CA3C      0862          LDD   2 DZ912-X2  WD CNT, SCTR ADDR NEXT OP     PMN08560
01CD 0  D900      0863          STD   1 0         STORE BOTH IN NEW I/O AREA    PMN08570
01CE 0  708A      0864          MDX     DZ240     BACK TO SET UP NEXT OPERATN   PMN08580
                  0865    *                                                       PMN08590
                  0866    *                                                       PMN08600
01CF      000B    0867          BSS     11        PATCH AREA              2-4   PMN08610
                  0868    *                                                       PMN08620
                  0869    *                                                       PMN08630
01DA 0  00A0      0870          DC      'CIL1     ID NO. OF CORE IMAGE LDR,P1   PMN08640
01DB 0  0000      0871    $CIDN DC      *-*       CORE ADDR/CID NO.             PMN08650
01DC 0  0000      0872          DC      *-*       WORD COUNT                    PMN08660
01DD 0  0000      0873          DC      *-*       SCTR ADDR                     PMN08670
01DE      0002    0874          BSS     2         WD CNT, SCTR ADDR CORE LDS    PMN08680
01E0 0            0875    $ZEND EQU     *         1 + END OF DISKZ              PMN08690
```

EQUIVALENCES

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  0877     *                                                        PMN08710
                  0878     * EQUIVALENCES FOR OCOM PARAMETERS                      PMN08720
                  0879     *                                                        PMN08730
0004 0            0880     #NAME EQU     4     NAME OF PROGRAM/CORE LOAD            PMN08740
0006 0            0881     #DBCT EQU     6     BLOCK CT OF PROGRAM/CORE LOAD        PMN08750
0007 0            0882     #FCNT EQU     7     FILES SWITCH                         PMN08760
0008 0            0883     #SYSC EQU     8     SYSTEM/NON-SYSTEM CARTRIDGE INDR     PMN08770
0009 0            0884     #JBSW EQU     9     JOBT SWITCH                          PMN08780
000A 0            0885     #CBSW EQU     10    CLB-RETURN SWITCH                    PMN08790
000B 0            0886     #LCNT EQU     11    NO. OF LOCALS                        PMN08800
000C 0            0887     #MPSW EQU     12    CORE MAP SWITCH                      PMN08810
000D 0            0888     #MDF1 EQU     13    NO. DUP CTRL RECORDS (MODIF)         PMN08820
000E 0            0889     #MDF2 EQU     14    ADDR OF MODIF BUFFER                 PMN08830
000F 0            0890     #NCNT EQU     15    NO. OF NOCALS                        PMN08840
0010 0            0891     #ENTY EQU     16    RLTV ENTRY ADDR OF PROGRAM           PMN08850
0011 0            0892     #RP67 EQU     17    1442-5 SWITCH                        PMN08860
0012 0            0893     #TODR EQU     18    OBJECT WORK STORAGE DRIVE CODE       PMN08870
0014 0            0894     #FHOL EQU     20    ADDR LARGEST HOLE IN FIXED AREA      PMN08880
0015 0            0895     #FSZE EQU     21    BLK CNT LARGEST HOLE IN FXA          PMN08890
0016 0            0896     #UHOL EQU     22    ADDR LARGEST HOLE IN USER AREA       PMN08900
0017 0            0897     #USZE EQU     23    BLK CNT LARGEST HOLE IN UA.          PMN08910
0018 0            0898     #DCSW EQU     24    DUP CALL SWITCH                      PMN08920
0019 0            0899     #PIOD EQU     25    PRINCIPAL I/O DEVICE INDICATOR       PMN08930
001A 0            0900     #PPTR EQU     26    PRINCIPAL PRINT DEVICE INDICATOR     PMN08940
001B 0            0901     #CIAD EQU     27    RLTV ADDR IN *STRT OF CIL ADDR       PMN08950
001C 0            0902     #ACIN EQU     28    AVAILABLE CARTRIDGE INDICATOR        PMN08960
001D 0            0903     #GRPH EQU     29    2250 INDICATOR              2G2      PMN08970
001E 0            0904     #GCNT EQU     30    NO. G2250 RECORDS           2G2      PMN08980
001F 0            0905     #LOSW EQU     31    LOCAL-CALLS-LOCAL SWITCH    2-2      PMN08990
0020 0            0906     #X3SW EQU     32    SPECIAL ILS SWITCH          2-2      PMN09000
0021 0            0907     #ECNT EQU     33    NO. OF *EQUAT RCDS          2-4      PMN09005
0023 0            0908     #ANDU EQU     35    1+BLK ADDR END OF UA (ADJUSTED)      PMN09010
0028 0            0909     #BNDU EQU     40    1+BLK ADDR END OF UA (BASE)          PMN09020
002D 0            0910     #FPAD EQU     45    FILE PROTECT ADDR                    PMN09030
0032 0            0911     #PCID EQU     50    CARTRIDGE ID, PHYSICAL DRIVE         PMN09040
0037 0            0912     #CIDN EQU     55    CARTRIDGE ID, LOGICAL DRIVE          PMN09050
003C 0            0913     #CIBA EQU     60    SCTR ADDR OF CIB                     PMN09060
0041 0            0914     #SCRA EQU     65    SCTR ADDR OF SCRA                    PMN09070
0046 0            0915     #FMAT EQU     70    FORMAT OF PROG IN WORKING STG        PMN09080
0048 0            0916     #FLET EQU     75    SCTR ADDR 1ST SCTR OF FLET           PMN09090
0050 0            0917     #ULET EQU     80    SCTR ADDR 1ST SCTR OF LET            PMN09100
0055 0            0918     #WSCT EQU     85    BLK CNT OF PROG IN WORKING STG       PMN09110
005A 0            0919     #CSHN EQU     90    NO. SCTRS IN CUSHION AREA            PMN09120
                  0920     *                                                        PMN09130
                  0921     * EQUIVALENCES FOR PHASE ID NUMBERS                     PMN09140
                  0922     *                                                        PMN09150
006E 0            0923     'MCRA EQU     110       PHASE ID FOR MCRA                PMN09160
0073 0            0924     'SUP6 EQU     115       PHASE ID FOR DUMP PROG  2-4      PMN09170
0074 0            0925     'SUP7 EQU     116       PHASE ID FOR AUX SUPV   2-4      PMN09180
0078 0            0926     'CLB0 EQU     120       PHASE ID FOR CLB, PHASE 0/1      PMN09190
008C 0            0927     '1403 EQU     140       PHASE ID FOR SYS 1403 SUBR       PMN09200
008D 0            0928     '1132 EQU     141       PHASE ID FOR SYS 1132 SUBR       PMN09210
008E 0            0929     'CPTR EQU     142       PHASE ID FOR SYS CP SUBR         PMN09220
008F 0            0930     '2501 EQU     143       PHASE ID FOR SYS 2501 SUBR       PMN09230
0090 0            0931     '1442 EQU     144       PHASE ID FOR SYS 1442 SUBR       PMN09240
0091 0            0932     '1134 EQU     145       PHASE ID FOR SYS 1134 SUBR       PMN09250
0092 0            0933     'KBCP EQU     146       PHASE ID FOR SYS KB/CP SUBR      PMN09260
0093 0            0934     'CDCV EQU     147       PHASE ID FOR SYS CD   CONV       PMN09270
0094 0            0935     'PTCV EQU     148       PHASE ID FOR SYS 1134 CONV       PMN09280
0095 0            0936     'KBCV EQU     149       PHASE ID FOR SYS KB   CONV       PMN09290
0096 0            0937     'DZID EQU     150       PHASE ID FOR DISKZ               PMN09300
0097 0            0938     'DIID EQU     151       PHASE ID FOR DISKI               PMN09310
0098 0            0939     'DNID EQU     152       PHASE ID FOR DISKN               PMN09320
00A0 0            0940     'CIL1 EQU     160       PHASE ID FOR CI LOADER,PH 1      PMN09330
00A1 0            0941     'CIL2 EQU     161       PHASE ID FOR CI LOADER,PH 2      PMN09340
                  0942     *                                                        PMN09350
                  0943     * EQUIVALENCES FOR RESIDENT MONITOR                     PMN09360
                  0944     *                                                        PMN09370
0014 0            0945     $LKNM EQU     $HASH     SAVE AREA FOR NAME OF LINK     PMN09380
0016 0            0946     $RMSW EQU     $HASH+2   EXIT-LINK-DUMP SW(-1,0,+1)     PMN09390
0017 0            0947     $CXR1 EQU     $HASH+3   SAVE AREA FOR XR1              PMN09400
0018 0            0948     $CLSW EQU     $HASH+4   SW FOR CORE IMAGE LDR,PH 2     PMN09410
0019 0            0949     $DMPF EQU     $HASH+5   DUMP FORMAT CODE               PMN09420
001A 0            0950     $ACEX EQU     $HASH+6   ACC AND EXT WHEN ENTER DUMP    PMN09430
005A 0            0951     $CILA EQU     $S150+1   ADDR OF END OF DK I/O - 3      PMN09440
0089 0            0952     $IBT2 EQU     $I205+1   ADR OF SERVICE PART OF DKIO    PMN09450
0004 0            0953     $IBT4 EQU     $I405+1   ADDR OF THE IBT                 PMN09460
00FF 0            0954     $SNLT EQU     $DBSY+1   SENSE LIGHT INDICATOR          PMN09470
00F0 0            0955     $PAUS EQU     DZ000-2   PAUSE,INTERRUPT INDICATOR     PMN09480
00F1 0            0956     $RWCZ EQU     DZ000-1   READ/WRITE SWITCH (CARDZ)     PMN09490
00E4 0            0957     $XR3X EQU     $I496     XR3 SETTING DURING XEQ  2-2    PMN09500
                  0958     *                                                        PMN09510
```

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                  ID/SEQNO

                  0959     * EQUIVALENCES FOR ABSOLUTE SECTOR ADDRESSES           PMN09520
                  0960     *                                                      PMN09530
0000 0            0961     'IOAD EQU     0     ADDR OF SCTR WITH ID,DEF CYL ADR   PMN09540
0001 0            0962     'DCOM EQU     1     ADDR OF SCTR CONTAINING DCOM       PMN09550
0002 0            0963     'RIAD EQU     2     ADDROOF SCTR CONTAINING RES IMGE   PMN09560
0003 0            0964     'SLET EQU     3     ADDR OF SCTR CONTAINING SLET       PMN09570
0006 0            0965     'RTBL EQU     6     ADDR OF SCTR CONTAINING RELD TBL   PMN09580
0007 0            0966     'HDNG EQU     7     ADDR OF SCTR CONTAINING PAGE HDR   PMN09590
0000 0            0967     'STRT EQU     0     ADDR OF SCTR W/ COLD START PROG    PMN09600
                  0968     *                                                      PMN09610
                  0969     * EQUIVALENCES FOR THE CORE IMAGE HEADER              PMN09620
                  0970     *                                                      PMN09630
0000 0            0971     'XEQA EQU     0     RLTV ADDR OF CORE LOAD EXEC ADDR   PMN09640
0001 0            0972     'CMON EQU     1     RLTV ADDR OF WD CNT OF COMMON      PMN09650
0002 0            0973     'DREQ EQU     2     RLTV ADDR OF DISK I/O INDICATOR    PMN09660
0003 0            0974     'FILE EQU     3     RLTV ADDR OF NO. FILES DEFINED     PMN09670
0004 0            0975     'HWCT EQU     4     RLTV ADDR OF WD CNT OF CI HEADER   PMN09680
0005 0            0976     'LSCT EQU     5     SCTR CNT OF FILES IN WK STORAGE    PMN09690
0006 0            0977     'LOAD EQU     6     RLTV ADDR OF LOAD ADDR CORE LOAD   PMN09700
0007 0            0978     'XCTL EQU     7     RLTV ADDR DISK1/DISKN EXIT CTRL    PMN09710
0008 0            0979     'TVWC EQU     8     RLTV ADDR OF WD CNT OF TV          PMN09720
0009 0            0980     'WCNT EQU     9     RLTV ADDR OF WD CNT OF CORE LOAD   PMN09730
000A 0            0981     'XR3X EQU     10    RLTV ADDR OF EXEC SETTING OF XR3   PMN09740
000B 0            0982     'ITVX EQU     11    RLTV ADDR OF 1ST WD OF ITV         PMN09750
0011 0            0983     'ILS4 EQU     17    RLTV ADDR OF 1ST WD OF IBT4        PMN09760
001A 0            0984     'OVSW EQU     26    RLTV ADDR OF LOCAL/SOCAL SWITCH    PMN09770
001C 0            0985     'CORE EQU     28    CORE SIZE OF BUILDING SYSTEM       PMN09780
001D 0            0986     'HEND EQU     29    RLTV ADDR OF LAST WD OF CI HDR     PMN09790
                  0987     *                                                      PMN09800
                  0988     * EQUIVALENCES FOR LET/FLET                           PMN09810
                  0989     *                                                      PMN09820
0005 0            0990     'LFHD EQU     5     WORD COUNT OF LET/FLET HEADER      PMN09830
0003 0            0991     'LFEN EQU     3     NO OF WDS PER LET/FLET ENTRY       PMN09840
0000 0            0992     'SCTN EQU     0     RLTV ADDR OF LET/FLET SCTR NO.     PMN09850
0001 0            0993     'UAFX EQU     1     RLTV ADDR OF SCTR ADDR OF UA/FXA   PMN09860
0003 0            0994     'WDSA EQU     3     RLTV ADDR OF WDS AVAIL IN SCTR     PMN09870
0004 0            0995     'NEXT EQU     4     RLTV ADDR OF ADDR NEXT SCTR        PMN09880
0000 0            0996     'LFNM EQU     0     RLTV ADDR OF LET/FLET ENTRY NAME   PMN09890
0002 0            0997     'BLCT EQU     2     RLTV ADDR OF LET/FLET ENTRY DBCT   PMN09900
                  0998     *                                                      PMN09910
                  0999     * MISCELLANEOUS EQUIVALENCES                          PMN09920
                  1000     *                                                      PMN09930
0033 0            1001     'ISTV EQU     51    ISS NO. ADJUSTMENT FACTOR    2-1   PMN09940
0005 0            1002     'MXDR EQU     5     MAX NO. DRIVES SUPPORTED           PMN09950
0380 0            1003     'COMZ EQU     896   LOW COMMON LIMIT FOR DISKZ         PMN09960
04C0 0            1004     'COM1 EQU     1216  LOW COMMON LIMIT FOR DISK1         PMN09970
0600 0            1005     'COM2 EQU     1536  LOW COMMON LIMIT OF DISKN          PMN09980
0011 0            1006     'TCNT EQU     17    NO. TRIES BEFORE DISK ERROR        PMN09990
00F9 0            1007     'DKEP EQU     DZ000+7 LIBF ENTRY TO DISK1/N            PMN10000
00F7 0            1008     'DKIP EQU     DZ000+5 DISK I/O INTERRUPT ENTRY PT      PMN10010
0010 0            1009     'SCIB EQU     16    CIB SECTOR COUNT             2-2   PMN10020
0003 0            1010     'HCIB EQU     3     HIGH COMMON SECTOR COUNT     2-2   PMN10030
1000 0            1011     'MCOR EQU     4096        SIZE OF MINIMUM CORE  2-2   PMN10040
007F 0            1012     Y     EQU     127                                      PMN10050
                  1013     *                                                      PMN10060
0004 0            1014     'CIDN EQU     4     RLTV ADDR CARTRIDGE ID       2-2   PMN10070
0005 0            1015     'COPY EQU     5     RLTV ADDR COPY INDICATOR     2-2   PMN10080
0001 0            1016     'DCTB EQU     1     RLTV ADDR DEFECTIV CYL TBL   2-2   PMN10090
0008 0            1017     'DTYP EQU     8     RLTV ADDR DISK TYPE INDR     2-2   PMN10100
```

COLD START PROGRAM

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                   1019    ************************************************** PMN10120
                   1020    *                                                * PMN10130
                   1021    *STATUS - VERSION 2, MODIFICATION LEVEL 5.       * PMN10140
                   1022    *                                                * PMN10150
                   1023    *FUNCTION/OPERATION -                            * PMN10160
                   1024    *   THIS PROGRAM IS READ INTO CORE FROM SECTOR 0  * PMN10170
                   1025    *   OF THE SYSTEM CARTRIDGE AND TRANSFERRED TO BY * PMN10180
                   1026    *   THE COLD START CARD.  DEFECTIVE CYLINDER     * PMN10190
                   1027    *   ADDRESSES, CARTRIDGE ID AND DISKZ ARE ALSO ON * PMN10200
                   1028    *   SECTOR 0 AND ARE READ IN AT THE SAME TIME.   * PMN10210
                   1029    *   ALL THAT REMAINS FOR THE COLD START PROGRAM IS* PMN10220
                   1030    *   TO READ IN THE RESIDENT IMAGE, SAVE THE      * PMN10230
                   1031    *   CARTRIDGE ID AND TRANSFER TO THE AUXILIARY   * PMN10240
                   1032    *   SUPERVISOR THROUGH $DUMP IN THE RESIDENT     * PMN10250
                   1033    *   MONITOR.                                     * PMN10260
                   1034    *                                                * PMN10270
                   1035    *ENTRY - CR010-2                                 * PMN10280
                   1036    *   ENTER PROGRAM BY TRANSFER FROM COLD START CARD* PMN10290
                   1037    *                                                * PMN10300
                   1038    *INPUT -                                         * PMN10310
                   1039    *   THE CARTRIDGE ID OF LOGICAL DRIVE ZERO (THE  * PMN10320
                   1040    *   SYSTEM CARTRIDGE) IS READ IN FROM SECTOR 0   * PMN10330
                   1041    *   WITH THE COLD START PROGRAM.                 * PMN10340
                   1042    *                                                * PMN10350
                   1043    *OUTPUT -                                        * PMN10360
                   1044    *   * THE RESIDENT IMAGE IS READ INTO CORE FROM  * PMN10370
                   1045    *     THE DISK.                                  * PMN10380
                   1046    *   * IN COMMA-                                  * PMN10390
                   1047    *     $ACDE                                      * PMN10400
                   1048    *     $CIBA-1                                    * PMN10410
                   1049    *     $CION                                      * PMN10420
                   1050    *     $CYLN                                      * PMN10430
                   1051    *     $DBSY                                      * PMN10440
                   1052    *     $IOCT                                      * PMN10450
                   1053    *                                                * PMN10460
                   1054    *EXTERNAL REFERENCES -                           * PMN10470
                   1055    *   DZ000  SUBROUTINE TO PERFORM DISK I/O.       * PMN10480
                   1056    *                                                * PMN10490
                   1057    *EXITS -                                         * PMN10500
                   1058    *   THE ONLY EXIT IS TO THE AUXILIARY SUPERVISOR * PMN10510
                   1059    *   AS FOLLOWS-                                  * PMN10520
                   1060    *     BSI     $DUMP                              * PMN10530
                   1061    *     DC      -1                                 * PMN10540
                   1062    *                                                * PMN10550
                   1063    *TABLES/WORK AREAS - N/A                         * PMN10560
                   1064    *                                                * PMN10570
                   1065    *ATTRIBUTES -                                    * PMN10580
                   1066    *   THIS PROGRAM IS NOT NATURALLY RELOCATABLE.   * PMN10590
                   1067    *                                                * PMN10600
                   1068    *NOTES -                                         * PMN10610
                   1069    *   DISK ERRORS RESULT IN A WAIT AT $PST2.       * PMN10620
                   1070    ************************************************** PMN10630
                   1072    *                                                  PMN10650
                   1073    * READ THE RESIDENT IMAGE INTO CORE                PMN10660
                   1074    *                                                  PMN10670
01E0 0  617F       1075          LDX   1 Y                                    PMN10680
01E1 0  C82E       1076          LDD     CR920      SET UP WORD COUNT AND SCTR PMN10690
01E2 00 0C000004   1077    CR010 STD   L $CIBA-1    *ADDR OF RESIDENT IMAGE    PMN10700
01E4 0  0125       1078          STO   1 $DCYL-Y    *INITIALIZE DEF CYL NO. 1  PMN10710
01E5 0  C184       1079          LD    1 3-Y        FETCH LOG DRIVE 0 AREA CODE PMN10720
01E6 0  0120       1080          STO   1 $ACDE-Y    *AND STORE IT IN COMMA     PMN10730
01E7 0  0029       1081          STO     CR920+1    SAVE THE AREA CODE         PMN10740
01E8 0  C156       1082          LD    1 DZ000-2-27-Y FETCH AND SAVE THE       PMN10750
01E9 0  00F1       1083          STO     $CION      *CARTRIDGE ID              PMN10760
01EA 0  C0F8       1084          LD      CR010+1    FETCH CORE ADDR OF RESIDENT PMN10770
01EB 0  1890       1085          SRT     16         *IMAGE AND PUT IN EXTENSION PMN10780
01EC 0  016F       1086          STO   1 $DBSY-Y    CLEAR DISK BUSY INDICATOR  PMN10790
01ED 0  0118       1087          STO   1 $CYLN-Y    INITIALIZE ARM POSITION    PMN10800
01EE 0  4173       1088          BSI   1 DZ000-Y    FETCH RESIDENT IMAGE       PMN10810
01EF 0  3000       1089          WAIT               WAIT OUT THE INTERRUPT     PMN10820
                   1090    *                                                  PMN10830
                   1091    * INITIALIZE ITEMS IN COMMA                       PMN10840
                   1092    *                                                  PMN10850
01F0 0  1810       1093          SRA     16                                   PMN10860
01F1 0  0183       1094          STO   1 $IOCT-Y    CLEAR IOCS COUNTER         PMN10870
01F2 0  C818       1095          LDD     CR910                                PMN10880
01F3 0  0985       1096          STD   1 $CIBA-1-Y *FOR SAVING CORE ON THE CIB PMN10890
01F4 0  C01C       1097          LD      CR920+1    FETCH AREA CODE            PMN10900
01F5 0  0120       1098          STO   1 $ACDE-Y    RESET AREA CODE            PMN10910
01F6 0  C016       1099          LD      CR905      INITIALIZE WD ZERO TO BR TO PMN10920
01F7 0  0181       1100          STO   1 0-Y        *DUMP ENTRY POINT PLUS 1   PMN10930
                   1101    *                                                  PMN10940
                   1102    * TRANSFER TO THE AUXILIARY SUPERVISOR            PMN10950
```

```
ADDR REL OBJECT   ST.NO.   LABEL OPCD FT OPERANDS                                    ID/SEQNO

                  1103     * TO COMPLETE INITIALIZATION                             PMN10960
                  1104     *                                                        PMN10970
01F8 0  41C0      1105           BSI    1 $DUMP-Y    BR TO AUXILLIARY SUPERVISOR  PMN10980
01F9 0  FFFF      1106           DC      -1          *FOR JOB PROCESSING          PMN10990
                  1107     *                                                        PMN11000
01FA    0013      1108           BSS     19          PATCH AREA                   PMN11010
                  1109     *                                                        PMN11020
                  1110     * CONSTANTS AND WORK AREAS                               PMN11030
                  1111     *                                                        PMN11040
020D 0  703F      1112     CR905 MDX   X  $DUMP+1-1 TO BE STORED IN LOCN ZERO       PMN11050
020E 0  0000      1113     CR910 DC       0         WC CNT,SCTR ADDR OF      2-5  PMN11060
020F 0  0007      1114           DC       'HDNG     *HARMLESS WRITE TO DISK      PMN11070
0210 0  00F8      1115     CR920 DC       $DBSY-$CH12 WD CNT AND SCTR             PMN11080
0211 0  0002      1116           DC       'RIAD     *ADDR OF RESIDENT IMAGE      PMN11090
0212    0212      1117           END      *                                        PMN11100
```

CROSS-REFERENCE

SYMBOL	VALUE	REL	DEFN	REFERENCES
CR010	01E2	0	1077	1084
CR905	020D	0	1112	1099
CR910	020E	0	1113	1095
CR920	0210	0	1115	1076 1081 1097
DZ000	00F2	0	0610	0323 0350 0627 0646 0707 0763 0955 0956 1007 1008 1082 1088
DZ010	00F7	0	0617	0628 0641
DZ020	00F9	0	0619	0613
DZ030	00FF	0	0625	0663
DZ060	0104	0	0630	0650
DZ070	0106	0	0634	0813 0819 0845 0853
DZ100	0108	0	0639	0619 0620 0715
DZ180	010E	0	0645	0618 0634
DZ185	0116	0	0651	0826
DZ190	0110	0	0660	0720
DZ210	0146	0	0711	0833
DZ215	014C	0	0719	0784
DZ220	014E	0	0724	0787
DZ230	0152	0	0730	0629
DZ235	0154	0	0731	0674 0654
DZ240	0159	0	0735	0864
DZ250	0168	0	0750	0758
DZ280	0169	0	0751	0737 0743 0744 0745 0756
DZ300	0174	0	0763	0753
DZ330	0179	0	0767	0739
DZ340	0188	0	0781	0726
DZ350	0190	0	0790	0662 0741 0824
DZ380	019D	0	0805	0802
DZ385	01A2	0	0810	
DZ390	01A5	0	0813	0809
DZ400	01AF	0	0831	0792
DZ410	0181	0	0833	0858
DZ900	0122	0	0669	0677 0804
DZ901	0123	0	0670	0823
DZ902	0124	0	0671	0651 0733 0854 0861
DZ904	0126	0	0673	0635 0805 0812 0818 0841
DZ905	0127	0	0674	0850
DZ906	0128	0	0675	0747 0840
DZ907	0129	0	0676	0774
DZ908	012A	0	0677	0817
DZ909	012B	0	0678	0771
DZ910	012C	0	0679	0647 0769 0781 0799
DZ911	012D	0	0680	0779
DZ912	012E	0	0681	0658 0734 0780 0831 0835 0856 0862
DZ913	012F	0	0682	0764
DZ914	0130	0	0683	0719
DZ915	0131	0	0684	0659
DZ916	0132	0	0685	0834
DZ920	0134	0	0687	0775
DZ925	0135	0	0688	0770
DZ930	0136	0	0689	0755
DZ935	0137	0	0690	0724
DZ940	0138	0	0691	0749
DZ945	0139	0	0692	0622 0773 0776 0851
DZ950	013A	0	0693	0778
DZ955	013B	0	0694	0765
DZ960	013C	0	0695	0742
DZ965	013D	0	0696	0738

SYMBOL	VALUE	REL	DEFN	REFERENCES						
DZ970	013E	0	0697	0740						
DZ975	013F	0	0698	0777	C849					
DZ980	0140	0	0699	0768						
DZ985	0141	0	0700	0838						
DZ990	0142	0	0701	0766	0772	0843				
DZ995	0143	0	0702	0803						
$ACDE	009F	0	0437	0696	C697	1080	1098			
$ACEX	001A	0	0950	0317						
$CCAD	0074	0	0364							
$CH12	0006	0	0254	1115						
$CIBA	0005	0	0253	0311	1077	1096				
$CIDN	010B	0	0871	1083						
$CILA	005A	0	0951	0348						
$CLSW	0018	0	0948							
$COMN	0007	0	0255							
$CORE	000E	0	0267							
$CPTR	007E	0	0374							
$CTSW	000F	0	0268							
$CWCT	0072	0	0362							
$CXR1	0017	0	0947	0319						
$CYLN	009A	0	0427	0695	0697	1087				
$DADR	0010	0	0269							
$DBSY	00EF	0	0552	0611	0626	0653	0712	0954	1086	1115
$DCDE	0077	0	0367							
$DCYL	00A4	0	0448	0695	1078					
$DDSW	000D	0	0530	0648	0782	0806				
$DMPF	0019	0	0949	0321						
$DREQ	0012	0	0271							
$DUMP	003F	0	0316	0320	0548	1105	1112			
$DZLN	0076	0	0366							
$EXIT	0038	0	0302							
$FLSH	0071	0	0360							
$FPAD	0095	0	0418							
$GCOM	0063	0	0354							
$GRIN	0064	0	0355							
$HASH	0014	0	0273	0945	0946	0947	0948	0949	0950	
$IBSY	0013	0	0272							
$IBT2	00B9	0	0952							
$IBT4	00D4	0	0953							
$IOCT	0032	0	0293	0330	0713	0730	1094			
$IREQ	002C	0	0287	0514						
$I200	0083	0	0474	0261	0485					
$I205	00B8	0	0480	0952						
$I210	00BA	0	0481	0475	0476	0477				
$I290	00C2	0	0486	0478	0484					
$I400	00C4	0	0506	0263	0525					
$I403	00D0	0	0516	0513						
$I405	00D3	0	0520	0953						
$I410	00D5	0	0521	0508	0509	0510				
$I420	00E6	0	0545							
$I425	00EA	0	0548	0547						
$I490	00DE	0	0531	0507	0524					
$I492	00E0	0	0532	0511						
$I494	00E2	0	0535	0517						
$I496	00E4	0	0542	0546	0957					
$KCSW	007C	0	0372							
$LAST	0033	0	0294							
$LEV0	0008	0	0259							
$LEV1	0009	0	0260							
$LEV2	000A	0	0261							
$LEV3	000B	0	0262							
$LEV4	000C	0	0263							
$LEV5	000D	0	0264							
$LINK	0039	0	0306	0340						
$LKNM	0014	0	0945	0346						
$LSAD	0075	0	0365							
$NDUP	0034	0	0295							
$NXEQ	0035	0	0296							
$PAUS	00F0	0	0955							
$PBSY	0036	0	0297							
$PGCT	0037	0	0298							
$PHSE	0078	0	0368							
$PRET	0028	0	0282	0284	0725					
$PST1	0081	0	0380	0382						
$PST2	0085	0	0386	0388	C660					
$PST3	0089	0	0392	0394						
$PST4	008D	0	0398	0400						
$RMSW	0016	0	0946	0339						
$RWCZ	00F1	0	0956							

SYMBOL	VALUE	REL	DEFN	REFERENCES
$SCAN	0020	0	0276	
$SCAT	0011	0	0270	0331
$SNLT	00EF	0	0954	
$STOP	0091	0	0405	0264 0407
$SYSC	00E0	0	0533	
$S000	0052	0	0338	0302
$S100	0053	0	0339	0308 0325
$S150	0059	0	0345	0951
$S200	005E	0	0349	
$S250	004B	0	0329	0318 0333 0347 0351
$S300	004C	0	0330	0332
$S900	003C	0	0310	0322 0324
$S910	003E	0	0312	0338
$UFDR	007D	0	0373	
$UFIO	0079	0	0369	
$ULET	002D	0	0288	
$WRD1	007B	0	0371	
$WSDR	007A	0	0370	
$XR3X	00F4	0	0957	
$ZEND	01E0	0	0875	0604 0606
$1132	007F	0	0375	
$1403	0080	0	0376	
X2	00F2	0	0707	0660 0764 0765 0766 0768 0769 0770 0771 0772 0773 0774 0775 0776 0777 0778 0779 0780 0781 0782 0799 0803 0804 0805 0806 0812 0813 0817 0818 0819 0823 0831 0834 0835 0838 0840 0841 0843 0845 0849 0850 0851 0853 0854 0856 0861 0862
Y	007E	0	1012	1075 1078 1079 1080 1082 1086 1087 1088 1094 1096 1098 1100 1105
#ACIN	001C	0	0902	
#ANDU	0023	0	0908	
#BNDU	0028	0	0909	
#CBSW	000A	0	0885	
#CIAD	001B	0	0901	
#CIBA	003C	0	0913	
#CIDN	0037	0	0912	
#CSHN	005A	0	0919	
#DBCT	0006	0	0881	
#DCSW	0018	0	0898	
#ECNT	0021	0	0907	
#ENTY	0010	0	0891	
#FCNT	0007	0	0882	
#FHOL	0014	0	0894	
#FLET	0048	0	0916	
#FMAT	0046	0	0915	
#FPAD	002D	0	0910	
#FSZE	0015	0	0895	
#GCNT	001E	0	0904	
#GRPH	0010	0	0903	
#JBSW	0009	0	0884	
#LCNT	000B	0	0886	
#LOSW	001F	0	0905	
#MDF1	000D	0	0888	
#MDF2	000E	0	0889	
#MPSW	000C	0	0887	
#NAME	0004	0	0880	
#NCNT	000F	0	0890	
#PCID	0032	0	0911	
#PIOD	0019	0	0899	
#PPTR	001A	0	0900	
#RP67	0011	0	0892	
#SCRA	0041	0	0914	
#SYSC	0008	0	0883	
#TODR	0012	0	0893	
#UHOL	0016	0	0896	
#ULET	0050	0	0917	
#USZE	0017	0	0897	
#WSCT	0055	0	0918	
#XRSW	0020	0	0906	
*BLCT	0002	0	0997	
*CDCV	0093	0	0934	
*CIDN	0004	0	1014	
*CIL1	00A0	0	0940	0870
*CIL2	00A1	0	0941	
*CLR0	0078	0	0926	
*CMON	0001	0	0972	
*COMZ	0380	0	1003	
*COM1	04C0	0	1004	
*COM2	0600	0	1005	
*COPY	0005	0	1015	

SYMBOL	VALUE	REL	DEFN	REFERENCES
*CORE	001C	0	0985	
*CPTR	008E	0	0929	
*DCOM	0001	0	0962	
*DCTR	0001	0	1016	
*DKEP	00F9	0	1007	
*DKIP	00F7	0	1008	
*DNID	0098	0	0939	
*DREQ	0002	0	0973	
*DTYP	0008	0	1017	
*DZID	0096	0	0937	0605
*DIID	0097	0	0938	
*FILE	0003	0	0974	
*HCIB	0003	0	1010	
*HDNG	0007	0	0966	1114
*HEND	001D	0	0986	
*HWCT	0004	0	0975	
*IDAD	0000	0	0961	
*ILS4	0011	0	0983	
*ISTV	0033	0	1001	
*ITVX	0008	0	0982	
*KBCP	0092	0	0933	
*KBCV	0095	0	0936	
*LDAD	0006	0	0977	
*LFEN	0003	0	0991	
*LFHD	0005	0	0990	
*LFNM	0000	0	0996	
*LSCT	0005	0	0976	
*MCOR	1000	0	1011	
*MCRA	006E	0	0923	
*MXDR	0005	0	1002	
*NEXT	0004	0	0995	
*OVSW	001A	0	0984	
*PTCV	0094	0	0935	
*RIAD	0002	0	0963	1116
*RTBL	0006	0	0965	
*SCIB	0010	0	1009	
*SCTN	0000	0	0992	
*SLFT	0003	0	0964	
*STRT	0000	0	0967	
*SUP6	0073	0	0924	
*SUP7	0074	0	0925	
*TCNT	0011	0	1006	0625
*TVWC	0008	0	0979	
*UAFX	0001	0	0993	
*WCNT	0009	0	0980	
*WDSA	0003	0	0994	
*XCTL	0007	0	0978	
*XEQA	0000	0	0971	
*XR3X	000A	0	0981	
*1132	008D	0	0928	
*1134	0091	0	0932	
*1403	008C	0	0927	
*1442	0090	0	0931	
*2501	008F	0	0930	

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.

Abbreviation	Meaning
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
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

Abbreviation	Meaning
DETM	Determine
DEVC	Device
DGT	Digit
DI	Digital Input
DICT	Dictionary
DIM	Dimension
DIRCTY	Directory
DISP	Displacement
DISPCHG	Dispatching
DK	Disk
DLMTER	Delimeter
DPC	Direct Program Control
DR	Drive
DSW	Device Status Word
DT	Defective Track
DUP	Disk Utility Program
DUPCTN	Duplication
EBC	EBCDIC
ELIM	Eliminate
ELT	Element
ENT	Entry
EOF	End Of File
EOJ	End Of Job
EOR	End Of Reel
EP	Extended Precision
EQ	Equal, Equate
EQU	Equate
ERP	Error Parameter
ERR	Error
ES	Electronic Switch
ETV	Executive Transfer Vector
EVAL	Evaluate
EXCH	Exchange
EXEC	Execute
EXP	Exponent
EXPR	Expression
EXTYP	Exit Type
EXTR	Extract
FAC	Floating Accumulator
FOR	FORTRAN
FIO	FORTRAN I/O
FLD	Field
FLDL	Field Length
FIG	Figure
FLET	The Location Equivalence Table for the Fixed Area
FLT	Floating
FMT	Format
FR	From
FREQ	Frequency
FUNC	Function
FWD	Forward
FXA	Fixed Area
FXD	Fixed
GEN	Generator
GENL	General
GM	Group Mark
GT	Greater Than
GTE	Greater Than or Equal To
HDLER	Handler
HDR	Header
HEX	Hexadecimal
HI	High
HLT	Halt
HSK	Housekeeping
HYPER	Hypertape
IAR	Instruction Address Register
IC	Instruction Counter
ID	Identification
IDX	Index
ILS	Interrupt Level Subroutine
ILSW	Interrupt Level Status Word
INCR	Increment
IND	Indicate
INDN	Indication
INDR	Indicator
INFO	Information
INITLZ	Initialize
INQ	Inquire
INT	Initial
INTFCE	Interface
INTLD	Interlude
INTM	Interim
INTMD	Intermediate
INTNL	Internal
INTRPT	Interrupt
I/O	Input/Output
IOAP	I/O Area Parameter
IOCC	I/O Control Command
IOCR	I/O Control Routine
INST	Instruction
INTERP	Interpret
INVAL	Invalid
ISS	Interrupt Service Subroutine
ITER	Iterate, Iteration
ITG	Integer
I/P	Input

Abbreviation	Meaning
KB	Keyboard
KP	Keypunch
LBL	Label
LCT	List Control Table
LD	Load
LET	Location Equivalence Table for User Area
LFT	Left
LH	Left-Hand, Leftmost
LINKB	Link/Busy Word
LIT	Literal
LN	Line
LNG	Length
LO	Low
LOC	Location
LOCAL	Load-on-Call Subroutine
LT	Less Than
LTE	Less Than or Equal To
LTR	Letter
LVL	Level
MACH	Machine
MAGT	Magnetic Tape
MAINT	Maintain, Maintenance
MALF	Malfunction
MAX	Maximum
MEM	Memory
MIN	Minimum
MISC	Miscellaneous
ML	Mainline
MN	Mnemonic
MOD	Modification
MON	Monitor
MPXR	Multiplexor
MPY	Multiply
MRGE	Merge
MSG	Message
MSTR	Master
NEC	Necessary
NEG	Negative
NO.	Number
NORM	Normalize, Normalized
NUM	Numeric
NXT	Next
OBJ	Object
OP	Operation
O/P	Output
OPN	Open
OPND	Operand
OPTN	Option
OPTR	Operator
ORG	Origin
OVFLO	Overflow
OVLP	Overlap
OVLY	Overlay
PAPT	Paper Tape
PARAM	Parameter
PARTL	Partial
PERF	Perforate, Perforated, Perforation
PERPHL	Peripheral
PFM	Perform
PG	Page
PGLIN	Page and Line
PH	Phase
PHYS	Physical
PK	Pack
PKD	Packed
PNCH	Punch
PNDG	Pending
POS	Position
PR	Print
PREC	Precision
PREV	Previous
PRGE	Purge
PRI	Priority
PRINC	Principal
PROC	Process
PROG	Program
PROT	Protect
PRTN	Partition
PRVNT	Prevent
PT	Pointer, Point
PTR	Printer
PTV	Positive
QUALFD	Qualified
QUANT	Quantity
QUE	Queue
RAND	Random
R + S	Reset and Start
R/W	Read/Write
RCD	Record
RCV	Receive
RD	Read
RDY	Ready

Abbreviation	Meaning
REF	Reference
REG	Register
REL	Release
RELOC	Relocate, Relocatable
REQ	Request, Require
RET	Return
RH	Right-Hand, Rightmost
RI	Read in
RLS	Reels
RLTV	Relative
RM	Record Mark
RO	Read Out
RPT	Report
RSLT	Result
RST	Reset
RSTRT	Restart
RT	Right
RTE	Route
RTN	Routine
RWD	Rewind
SAD	Sector Address
SAT	Satellite
SAR	System Action Required
SCHED	Schedule, Scheduler
SCN	Scan
SCTR	Sector
SECT	Section
SEL	Select
SEN	Sense
SEQ	Sequence
SEQNO	Sequence Number
SER	Serial
SEG	Segment
SIG	Signal
SIM	Simulator
SK	Skeleton
SLET	System Location Equivalence Table
SM	Storage Mask
SNGL	Single
SOCAL	System Load-On-Call Subroutine
SP	Space
SRCH	Search
SPEC	Specification, Specify
ST	Store
STA	Station
STD	Standard
STG	Storage, Storing
STMNT	Statement
STP	Standard Precision
SUB	Subtract
SUBP	Subprogram
SUBR	Subroutine
SUBSC	Subscript
SUMM	Summarize
SUP	Suppress
SUP	Supervisor
SYNC	Synchronize, Synchronizer
SYM	Symbol
SYSRx	System Reserved Word (x is a digit)
SYST	System
SW	Switch
TBL	Table
TECHNQE	Technique
TEMP	Temporary
TERM	Terminal, Terminate
TM	Tapemark
TMN	Transmission
TMT	Transmit
TOT	Total
TP	Tape
TR	Transfer
TRK	Track
TRLR	Trailer
TRUNC	Truncate, Truncation
TST	Test
TU	Tape Unit
TV	Transfer Vector
TW	Typewriter
UA	User Area
UAR	User Action Required
UFLO	Underflow
UNC	Unconditional
UNLD	Unload
UNPKD	Unpacked
UTIL	Utility
V	Version
VAL	Value
VAR	Variable
VIOL	Violation
VOL	Volume
WD	Word
WM	Word Mark
WR	Write
WRK	Work
WS	Working Storage

Abbreviation	Meaning
W/	With
W/O	Without
XPL	Explain, Explanation
XR1	Index Register 1
XR2	Index Register 2
XR3	Index Register 3
XTR	Extra
Z	Zero
ZN	Zone

Abbreviation	Meaning
1st	First
2nd	Second
3rd	Third
4th	Fourth
5th	Fifth
6th	Sixth
7th	Seventh
8th	Eighth
9th	Ninth

APPENDIX D. MICROFICHE REFERENCE TABLE

Disk Monitor System Component	Microfiche Identification
System Loader (Paper Tape)	
Bootstrap	BAA.001.00
Phase 1	BAA.002.00
Phase 2	BAA.003.00
System Loader (Card)	
Bootstrap	BAA.004.00
Core Image Loader	BAA.005.00
Phase 1	BAA.006.00
Phase 2	BAA.007.00
Disk Utility Program	
DUPCO	JAA.001.00
DCTL	JAA.002.00
STORE	JAA.003.00
FILEQ	JAA.004.00
DDUMP	JAA.005.00
DMPLT	JAA.006.00
DELET	JAA.007.00
DFINE	JAA.008.00
DEXIT	JAA.009.00
CDFAC	JAA.010.00
KBFAC	JAA.011.00
PTFAC	JAA.012.00
PRECI	JAA.013.00
FORTRAN Compiler	
Phase 1	KAA.001.00
Phase 2	KAA.002.00
Phase 3	KAA.003.00
Phase 4	KAA.004.00
Phase 5	KAA.005.00
Phase 6	KAA.006.00
Phase 7	KAA.007.00
Phase 8	KAA.008.00
Phase 9	KAA.009.00
Phase 10	KAA.010.00
Phase 11	KAA.011.00
Phase 12	KAA.012.00
Phase 13	KAA.013.00
Phase 14	KAA.014.00
Phase 15	KAA.015.00
Phase 16	KAA.016.00
Phase 17	KAA.017.00
Phase 18	KAA.018.00
Phase 19	KAA.019.00
Phase 20	KAA.020.00
Phase 21	KAA.021.00
Phase 22	KAA.022.00
Phase 23	KAA.023.00
Phase 24	KAA.024.00
Phase 25	KAA.025.00
Phase 26	KAA.026.00
Phase 27	KAA.027.00
Assembler Program	
Phase 0	MAA.001.00
Phase 1	MAA.007.00
Phase 1A	MAA.008.00
Phase 2	MAA.012.00
Phase 2A	MAA.013.00
Phase 3	MAA.010.00
Phase 4	MAA.011.00
Phase 5	MAA.015.00
Phase 6	MAA.016.00
Phase 7	MAA.017.00
Phase 7A	MAA.018.00
Phase 8	MAA.019.00
Phase 8A	MAA.020.00
Phase 9	MAA.014.00
Phase 10	MAA.003.00
Phase 10A	MAA.009.00
Phase 11	MAA.004.00
Phase 12	MAA.005.00
Error Message Phase	MAA.006.00
Punch Conversion Phase	MAA.021.00
Read Conversion Phase	MAA.002.00
Supervisor	
Monitor Control Record Analyzer	NAA.001.00
System Core Dump Program	NAA.002.00
Auxiliary Supervisor	NAA.003.00
Core Load Builder	OAA.001.00

Disk Monitor System Component	Microfiche Identification
System Device Subroutines	PAA.001.00
Standard Precision Arithmetic and Function Subroutines	
FADD	RAA.001.00
FAXI	RAA.001.00
FDIV	RAA.001.00
FDVR	RAA.001.00
FGETP	RAA.001.00
FLD	RAA.002.00
FMPY	RAA.002.00
FSBR	RAA.002.00
FABS	RAA.002.00
FATN	RAA.002.00
FAXB	RAA.003.00
FEXP	RAA.003.00
FLN	RAA.003.00
FSIGN	RAA.003.00
FSIN	RAA.003.00
FSQR	RAA.004.00
FTANH	RAA.004.00
SFAR	RAA.004.00
SFIF	RAA.004.00
Extended Precision Arithmetic and Function Subroutines	
EADD	SAA.001.00
EAXI	SAA.001.00
EDIV	SAA.001.00
EDVR	SAA.001.00
EGETP	SAA.001.00
ELD	SAA.002.00
EMPY	SAA.002.00
ESBR	SAA.002.00
EABS	SAA.002.00
EATN	SAA.002.00
EAXB	SAA.003.00
EEXP	SAA.003.00
ELN	SAA.003.00
ESIGN	SAA.003.00
ESIN	SAA.003.00
ESQR	SAA.004.00
ETANH	SAA.004.00
SEAR	SAA.004.00
SEIF	SAA.004.00
FARC	SAA.004.00
FIXI	SAA.005.00
FLOAT	SAA.005.00
IFIX	SAA.005.00
NORM	SAA.005.00
SNR	SAA.005.00
XDD	SAA.006.00
XMD	SAA.006.00
XMDS	SAA.006.00
FORTRAN Common Subroutines (No Precision)	
FBTD	TAA.001.00
IABS	TAA.001.00
XSQR	TAA.001.00
PAUSE	TAA.001.00
STOP	TAA.002.00
SUBIN	TAA.002.00
SUBSC	TAA.002.00
TTEST	TAA.002.00
DATSW	TAA.002.00
DVCHK	TAA.003.00
FCTST	TAA.003.00
ISIGN	TAA.003.00
OVERF	TAA.003.00
PDUMP	TAA.003.00
SLITE	TAA.004.00
TSTOP	TAA.004.00
TSTRT	TAA.004.00
FORTRAN Trace Subroutines	
SGOTO	TAA.009.00
SIAR	TAA.009.00
SIIF	TAA.009.00
FORTRAN Conversion Subroutines and Tables	
EBCTB	TAA.006.00
GETAD	TAA.006.00
HOLEZ	TAA.007.00
HOLTB	TAA.007.00
FORTRAN I/O Subroutines	
SDFIO	TAA.004.00
SDFND	TAA.005.00
SFIO	TAA.005.00
UFIO	TAA.006.00

Disk Monitor System Component	Microfiche Identification
FORTRAN Device Subroutines	
CARDZ	TAA.006.00
PAPTZ	TAA.007.00
PNCHZ	TAA.007.00
PRNTZ	TAA.008.00
PRNZ	TAA.008.00
READZ	TAA.008.00
TYPEZ	TAA.008.00
WRTYZ	TAA.008.00
Interrupt Level Subroutines (ILSs)	
ILS00	UAA.001.00
ILS01	UAA.001.00
ILS02	UAA.001.00
ILS03	UAA.001.00
ILS04	UAA.001.00
Interrupt Service Subroutines (ISSs)	
CARD0	UAA.002.00
CARD1	UAA.002.00
OMPR1	UAA.002.00
PAPT1	UAA.002.00
PAPTN	UAA.002.00
PAPTX	UAA.003.00
PLOT1	UAA.003.00
PNCH0	UAA.003.00
PNCH1	UAA.003.00
PRNT1	UAA.004.00
PRNT3	UAA.004.00
READ0	UAA.004.00
READ1	UAA.004.00
TYPE0	UAA.005.00
WRTY0	UAA.005.00
Conversion Subroutines and Tables	
EBPRT	UAA.005.00
HOLEB	UAA.005.00
HOLPR	UAA.006.00
HXBIN	UAA.006.00
PAPEB	UAA.006.00
PAPHL	UAA.006.00
PAPPR	UAA.006.00
SPEED	UAA.007.00
ZIPCO	UAA.007.00
BIDEC	UAA.007.00
BINDC	UAA.007.00
BINHX	UAA.007.00
CPEBC	UAA.008.00
CPHOL	UAA.008.00
CPPT3	UAA.008.00
DCBIN	UAA.008.00
DECBI	UAA.008.00
EBCCP	UAA.009.00
EBHOL	UAA.009.00
EBPA	UAA.009.00
EBPT3	UAA.009.00
HLEBC	UAA.009.00
HLPT3	UAA.010.00
HOLCP	UAA.010.00
HOLL	UAA.010.00
PRTY	UAA.010.00
PT3CP	UAA.010.00
PT3EB	UAA.011.00
PTHOL	UAA.011.00
Utility Dump Subroutines	
DMP80	UAA.011.00
DMTD0	UAA.011.00
DMPD1	UAA.012.00
System Subroutines	
SYSUP	UAA.013.00
FLIPR	UAA.012.00
Mainline Programs	
ADRWS	UAA.014.00
COPY	UAA.014.00
DISC	UAA.015.00
DLCIB	UAA.016.00
DSLET	UAA.016.00
IDENT	UAA.017.00
ID	UAA.017.00
MODIF	UAA.018.00
PTUTL	UAA.019.00
CALPR	UAA.019.00
FSLEN	UAA.019.00
RDREC	UAA.020.00
Plotter Subroutines	
ECHAR	VAA.001.00

Disk Monitor System Component	Microfiche Identification
ECHRX	VAA.001.00
EGRID	VAA.001.00
EPLOT	VAA.001.00
ERULE	VAA.002.00
FCHAR	VAA.002.00
FCHRX	VAA.002.00
FGRID	VAA.002.00
FPLOT	VAA.003.00
FRULE	VAA.003.00
PLOTI	VAA.003.00
PLOTX	VAA.003.00
POINT	VAA.003.00
SCALE	VAA.004.00
SCALF	VAA.004.00
XYPLT	VAA.004.00
SCA Interrupt Service Subroutines (ISSs), Conversion Subroutines, and Conversion Tables	
EBC48	WAA.001.00
HOL48	WAA.001.00
HOLCA	WAA.001.00
HXCV	WAA.001.00
PRNT2	WAA.002.00
SCAT1	WAA.003.00
STRTB	WAA.004.00
Stand-alone Programs	
Cold Start Card	ZAA.001.00
Console Printer Core Dump (Paper Tape)	ZAA.003.00
Console Printer Core Dump (Card)	ZAA.002.00
DCIP	ZAA.004.00
1132/1403 Printer Core Dump (Paper Tape)	
Phase 1	ZAA.007.00
Phase 2	ZAA.008.00
1132/1403 Printer Core Dump (Card)	
Phase 1	ZAA.005.00
Phase 2	ZAA.006.00
PTREP	ZAA.009.00
UCART	ZAA.010.00

FORTRAN Integer Data:

A FORTRAN integer can always be contained in one word (16 bits) regardless of the amount of disk storage allocated for the integer. If the *ONE WORD INTEGERS control card is used in the job which creates the FORTRAN disk data file, one data word is allocated for each integer; otherwise, disk storage allocation for each integer is that specified for real variables (2 or 3 words).

Example: The integer field +32767 appears on a disk file beginning * at record word n as described below.

With one word integers and standard or extended precision specified:

Record Word:	n
Content:	/7FFF

With standard precision only specified:

Record Word:	n-1	n
Content:	/7FFF	/XXXX

where /XXXX represents a fill word (meaningless data).
With extended precision only specified:

Record Word:	n-2	n-1	n
Content:	/7FFF	/XXXX	/XXXX

where /XXXX represents a fill word.

*FORTRAN disk file build proceeds backwards from the 320th word of each disk sector.

Two Word Integer Data:

A two-word integer is an integer which can be represented in two or less words (32 bits). Two words of disk storage are allocated for each two-word integer. One word integers may appear in the same disk data file as two-word integers, but extended precision real variables may not be used in a file which contains two-word integers.

Example: Thetwo-word integer field +500 appears on a disk file beginning at record word n as

Record Word:	n-1	n
Content:	/0000	/01F4

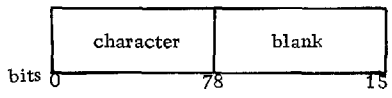
FORTRAN REAL DATA:

A complete description of FORTRAN real data appears in the IBM 1130 Subroutine Library, Form C26-5929.

FORTRAN A-FORMATTED DATA (integer and real): This data format is described in the IBM 1130/1800 Basic FORTRAN IV Language, Form C26-3715.

Commercial Subroutine Package (CSP) A1 FORMAT:

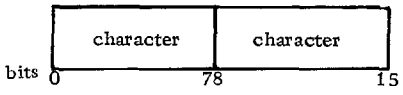
CSP A1 format consists of 1 character per 16-bit word, left-justified:



The righthand 8 bit always contain the blank character /40. This blank is inserted by the CSP subroutines.

CSP A2 FORMAT:

CSP A2 format consists of two characters per word:



CSP A3 FORMAT

CSP A3 format represents 3 characters as a one-word integer. The user supplies a 40 character translation table and the A3 integer is formed from the relative (to card column 40) positions of each character in this table.

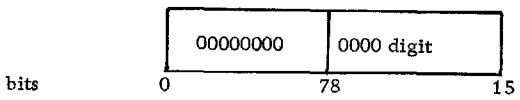
A3 integer = (N1-20)*1600+(N2*40)+N3

where N1, N2 and N3 are the relative translation table positions of the first, second and third chara-

cters respectively.

CSP D1 FORMAT:

CSP D1 format consists of one digit per word, right-justified, representation as described below:



Example: The six-digit field 001968 appears on a disk file beginning at record word n as follows:

Record Word:	n-5	n-4	n-3	n-2	n-1	n
Content:	/0008	/0006	/0009	/0001	/0000	/0000

The sign of the field is reflected in the low order (digit) word of the field, this word being the ones' complement of the digit if the field is negative.

Example: The six-digit field -001968 appears on a disk file beginning at record word n as follows:

Record Word:	n-5	n-4	n-3	n-2	n-1	n
Content:	/FFF7	/0006	/0009	/0001	/0000	/0000

CSP D4 FORMAT:

CSP D4 format consists in general of four decimal digits per word, with each digit occupying four bits of the word. However, since the low order digit carries the sign of the field, it is handled separately, and is placed by itself in the last word of the D4 field. Any 4-bit blocks not used in the field are filled with /F. This format is best illustrated with examples.

Example 1: The five-digit field +12345 appears on a disk file beginning at record word n as follows:

Record Word:	n-1	n
Content:	/0005	/1234

Example 2: The six-digit field +123456 appears on a disk file beginning at record word n as follows:

Record Word:	n-2	n-1	n
Content:	/0006	/5FFF	/1234

Example 3: The seven-digit field -1234567 appears on a disk file beginning at record word n as follows:

Record Word:	n-2	n-1	n
Content:	/FFF8	/56FF	/1234

All DFCNV error messages and their identifying numbers (listed below) are printed on the principal printer. All diagnostics are printed before data conversion begins. Any diagnostic except warning messages (indicated below) causes program termination.

No.	Message	Cause
F01	INVALID DESCRIPTION CARD FIELD-COL. XX	1) 1. Numeric field at card column XX outside allowable field range. 1) 2. Unrecognizable character in field at card column XX.
F02	FILE NAME NOT IN LET/FLET-Y	1) 1. LET/FLET entry not found for file named on File Description card. 1) 2. File name given on File Description card invalid. Y=I, input file error =O, output file error
F03	FILE SIZE INVALID-Y	1) File size calculated from File Description data exceeds actual file size.
4) F04	INVALID FIELD SPECIFICATION SYNTAX-COL. XX	1. Numeric field of specification starting at card column XX outside allowable field range. 2. Unrecognizable character in field of specification starting at card column XX. 3) 3. Embedded or intervening blanks on Field Specification card. 4. J-field type specification detected starting at card column XX when extended precision was specified.
F05	CSP A3 TABLE MISSING	No A (col 72) card precedes/* card when F-field specified.
F06	INVALID CARD SEQUENCE	2) 1. Unrecognizable card precedes /* card (column 72 not D, S, or A). 1) 2. Multiple File Description cards read. 1) 3. File Description card out of order. 1) 4. No Field S ecification card precedes /* card.

No.	Message	Cause
5) F07	TRUNCATION OCCURS AT COL. XXX.	2) High order truncation occurs in output field at column XXX.
F08	CARD INPUT INVALID	1) Card input is specified when principal input device is console keyboard.
F09	OUTPUT RECORD LENGTH INVALID	Sum of individual field lengths exceeds specified record length for output.
F10	FIELD OUT OF RANGE AT COL. XXX OF RECORD YYYYY	2) RPG real number field starting at column XXX has been set to zeroes or nines in record YYYYY.

1) Program termination immediate.
2) Warning only.
3) No columns indication.
4) It is noted that certain invalid field specification formats and/or invalid continuation marks (comma or blank allowable) which produce the F04 error message offset the "next" specification checked for validity. That is, the field specification immediately following is performed and no compression is built.

Example: The Field Specification card

```
Col.  1                    72
      6-16,1-15,X4         S
```

produces the following error message. F04 INVALID FIELD SPECIFICATION SYNTAX - COL. 01.
The invalid specification 1-15 will not be diagnosed until the specification 6-16 is corrected to read 6-16.0.

5) Extraneous truncation warnings may result from certain invalid field specifications.

Compression for I-, J- and R-field type:

Word	Contents
1	Address of I-, J- or R-conversion subroutine.
2	The displacement in characters to the start of the RPG output field in binary (equivalent to the specified RPG column -1).
3 bit 0-7	Precision (field length) of output field in binary
bit 8	1 if output field is to be packed, 0 if not.
bits 9-15	Scale of output field in binary

Compression for B- and C-field type:

Word	Contents
1	Address of B (C) - conversion subroutine
2	The displacement in characters to the start of the RPG output field in binary (equivalent to the specified RPG column -1).
3 bits 0-7	Precision of output field in binary.
bits 8-9	Number of words per entry.
bits 10-15	Number of characters per entry.

Compression for D- and E-field type:

Word	Contents
1	Address of D- or E-conversion subroutine
2	The displacement in characters to the start of the RPG output field in binary (equivalent to the specified RPG column -1).
3 bits 0-7	Precision of output field in binary
bit 8	1 if output field is to be packed, 0 if not
bits 9-15	Scale of output field in binary
4 bits 0-7	CSP precision in binary
bits 8-15	CSP scale in binary

Compression for F-field type:

Word	Contents
1	Address of F-conversion subroutine
2	The displacement in characters to the start of the RPG output field in binary (equivalent to the specified RPG column -1)
3	Hexadecimal constant /0341
4	Number of characters to be converted

Compression for X-field type:

Word	Contents
1	Address of X-conversion subroutine
2	Number of words to be bypassed

INDEX

IBM 1130 Printed in U.S.A. Y26-3714-2

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130 Disk Monitor System Version 2,
Program Logic Manual

Form Y26-3714-2

- How did you use this publication?

  As a reference source ..... ☐
  As a classroom text ..... ☐
  As a self-study text ..... ☐

- Based on your own experience, rate this publication . . .

	.....	.....	.....	.....	.....
As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

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

YOUR COMMENTS, PLEASE . . .

This SRL manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, 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. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Department 813 L

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

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

CUT ALONG THIS LINE

IBM 1130 Printed in U. S. A. Y26-3714-2