File No. 1130-36
Order No. GC26-3717-9

IBM Systems Reference Library

IBM 1130 Disk Monitor System, Version 2, Programmer's and Operator's Guide

Program Numbers: 1130-OS-005
1130-OS-006

Tenth Edition (May 1972)

This is a major revision of, and obsoletes, GC26-3717-8. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the changes.

This edition applies to version 2, modification 11, of the IBM 1130 Disk Monitor Programming System; to version 1, modification 5, of the IBM 1130 Remote Job Entry Work Station Program, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein. Before using this publication in connection with the operation of IBM systems, consult the latest SRL Newsletter, GN20-1130, for the editions that are applicable and current.

Text for this manual has been prepared with the IBM Selectric ® Composer.

Some illustrations in this manual have a code number in the lower corner. This is a publishing con number and is not related to the subject matter.

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

A form for readers' comments is provided at the back of this publication. If the form ha send your comments to IBM Corporation, Systems Publications, Department 27T, P. O. Boca Raton, Florida 33432.

This publication contains reference information for controlling and operating the 1130 Disk Monitor System, Version 2. The publication assumes you are familiar with the programming language needed to do your jobs.

Chapter 1 of this publication describes how you use this book. The rest of the chapters:

- Describe the disk monitor system (DM2) programs and disk areas
- Describe the control records for controlling the functions of the disk monitor system
- Provide tips and techniques for more efficient use of DM2
- Provide sample operating procedures for loading, reloading, and using DM2
- Describe the 1130 RJE Work Station Program

The minimum system configuration required to operate the IBM 1130 Disk Monitor System, Version 2, Program Number 1130-OS-005 (card input/output) is:

- An IBM 1131 Central Processing Unit, Model 2A or 4A (with an internal single disk storage drive and 4096 words of core storage)
- An IBM 1442 Card Read Punch, Model 6 or 7, or an IBM 1442 Card Reader, in combination with an IBM 1442 Card Punch, Model 5

or

- An IBM 1131 Central Processing Unit, Model 1B (with 8192 words of core storage)
- An IBM 1133 Multiplex Control Enclosure
- An IBM 2311 Disk Storage Drive, Model 12
- An IBM 1442 Card Read Punch, Model 6 or 7, or an IBM 2501 Card Reader, in combination with an IBM 1442 Card Punch, Model 5

The minimum system configuration required to operate the IBM 1130 Disk Monitor System, Version 2, Program Number 1130-OS-006 (paper tape input/output) is:

- An IBM 1131 Central Processing Unit, Model 2A (with an internal single disk storage drive and 4096 words of core storage)
- An IBM 1134 Paper Tape Reader
- An IBM 1055 Paper Tape Punch

The following publications provide further information about the 1130 computing system:

IBM 1130 Functional Characteristics, GA26-5881
IBM 1130 Operating Procedures, GA26-5717
IBM 1130/1800 Assembler Language, GC26-3778
IBM 1130/1800 Basic FORTRAN IV Language, GC26-3715
IBM 1130 RPG Language, GC21-5002
IBM 1130 Subroutine Library, GC26-5929
IBM 1130 MTCA IOCS Subroutines, GC33-3002
IBM 1130 Synchronous Communications Adapter Subroutines, GC26-3706
IBM 1130/1800 Plotter Subroutines, GC26-3755
IBM System/360 Operating System and 1130 Disk Monitor System: System/360 1130 Data Transmission for FORTRAN, GC27-6937
IBM System/360 Operating System and 1130 Disk Monitor System: User's Guide for Job Control from an IBM 2250 Display Unit Attached to an IBM 1130 System, GC27-6938
IBM System/360 Operating System: Remote Job Entry, GC30-2006

Publications that provide information about IBM 1130 COBOL, a program product, are:

IBM 1130 COBOL General Information Manual, GH20-0799
IBM 1130 COBOL Language Specifications Manual, SH20-0816

Contents

Summary of Amendments . . . vii

Chapter 1. How to Use This Publication . . . 1-1

Chapter 2. Disk Organization . . . 2-1

System Cartridge . . . 2-4
  Cylinder 0 on a System Cartridge . . . 2-4
  IBM System Area on a System Cartridge . . . 2-6
  Fixed Area . . . 2-9
  User Area and Working Storage . . . 2-9
Nonsystem Cartridge . . . 2-12
  Cylinder 0 on a Nonsystem Cartridge . . . 2-12
  IBM System Area on a Nonsystem Cartridge . . . 2-13
Summary of the Contents of Disk Cartridges . . . 2-14

Chapter 3. Monitor System Programs . . . 3-1

Supervisor . . . 3-2
  Resident Monitor . . . 3-2
  Disk-resident Supervisor Programs . . . 3-3
Disk Utility Program . . . 3-4
  General Functions of DUP . . . 3-4
Assembler . . . 3-5
FORTRAN Compiler . . . 3-6
RPG Compiler . . . 3-6
Core Load Builder . . . 3-7
  Construction of a Core Load . . . 3-7
Core Image Loader . . . 3-13

Chapter 4. Monitor System Library . . . 4-1

System Library ISS Subroutines . . . 4-2
System Library Utility Subroutines . . . 4-4
System Library Mainline Programs . . . 4-5
  IDENT . . . 4-5
  DISC . . . 4-6
  DSLET . . . 4-7
  ID . . . 4-7
  COPY . . . 4-7
  ADRWS . . . 4-8
  DLCIB . . . 4-8
  MODIF . . . 4-8
  MODSF . . . 4-14
  DFCNV . . . 4-20
  PTUTL . . . 4-25

Chapter 5. Control Records . . . 5-1

Monitor Control Records . . . 5-1
  // JOB . . . 5-2
  // ASM . . . 5-5
  // FOR . . . 5-6
  // RPG . . . 5-6
  // COBOL . . . 5-6
  // DUP . . . 5-6
  // XEQ . . . 5-7
  // * (Comments) . . . 5-9
  // PAUS . . . 5-10
  // TYP . . . 5-10
  // TEND . . . 5-10
  // EJECT . . . 5-11
  // CPRNT . . . 5-11
  // CEND . . . 5-11
Supervisor Control Records . . . 5-12
  *LOCAL . . . 5-13
  *NOCAL . . . 5-14
  *FILES . . . 5-15
  *G2250 . . . 5-16
  *EQUAT . . . 5-17
DUP Control Records . . . 5-18
  Altering LET and FLET . . . 5-20
  Information Transfer and Format Conversion . . . 5-20
  Restrictions Caused by Temporary Mode . . . 5-22
  *DUMP . . . 5-22
  *DUMPDATA . . . 5-24
  *DUMPDATA E . . . 5-26
  *DUMPLET . . . 5-28
  *DUMPFLET . . . 5-29
  *STORE . . . 5-30
  *STOREDATA . . . 5-33
  *STOREDATAE . . . 5-34
  *STOREDATACI . . . 5-37
  *STORECI . . . 5-38
  *STOREMOD . . . 5-42
  *DELETE . . . 5-44
  *DEFINE . . . 5-45
  *DWADR . . . 5-47
  *DFILE . . . 5-48
  *MACRO UPDATE . . . 5-49
Assembler Control Records . . . 5-50
  *TWO PASS MODE . . . 5-52
  *LIST . . . 5-53
  *XREF . . . 5-56
  *LIST DECK . . . 5-57
  *LIST DECK E . . . 5-59
  *PRINT SYMBOL TABLE . . . 5-59
  *PUNCH SYMBOL TABLE . . . 5-59
  *SAVE SYMBOL TABLE . . . 5-60
  *SYSTEM SYMBOL TABLE . . . 5-60
  *LEVEL . . . 5-61
  *OVERFLOW SECTORS . . . 5-61
  *COMMON . . . 5-63
  *MACLIB . . . 5-63
FORTRAN Control Records . . . 5-64
  *IOCS . . . 5-65
  *LIST SOURCE PROGRAM . . . 5-66
  *LIST SUBPROGRAM NAMES . . . 5-66
  *LIST SYMBOL TABLE . . . 5-67
  *LIST ALL . . . 5-67
  *EXTENDED PRECISION . . . 5-68
  *ONE WORD INTEGERS . . . 5-68
  *NAME . . . 5-69
  ** (Header Information) . . . 5-69
  *ARITHMETIC TRACE . . . 5-70
  *TRANSFER TRACE . . . 5-70
  *ORIGIN . . . 5-71
  RPG Control Card . . . 5-74
  End-of-File Control Card . . . 5-74

Chapter 6. Programming Tips and Techniques 6-1

Tips on Monitor Control and Usage 6-1
Stacked Job Input Arrangement 6-1
How to Use Temporary Job Mode 6-4
Using the Disk I/O Subroutines 6-4
Restoring Destroyed Cartridges 6-5
How to Avoid Overprinting When Using // CPRNT . 6-5
How to Avoid Overprinting When Linking Between Programs 6-5
Usage of the EJECT Monitor Control Record . . . 6-5
Duplicate Program and Data File Names 6-6
Disadvantages of Storing a Program in DCI Format . 6-7
Size Discrepancies in Stored Programs 6-7
Dumping and Restoring Data Files 6-8
Use of Defined Files 6-9
Mainline Programs that Use All of Core 6-9
The Use of LOCALs 6-9
LOCAL-Calls-a-LOCAL 6-10
LOCAL and NOCAL Control Record Usage . . . 6-10
The Use of NOCALs 6-11
The Use of SOCALs 6-13
Reading a Core Map and a File Map 6-13
Locating FORTRAN Allocation Addresses 6-18
Reading the Transfer Vector 6-19
SYSUP 6-20
Data File Processing 6-23
FORTRAN Disk File Organization and Processing . . 6-23
Assembler and RPG Disk File Organization and Processing 6-27
Calculating Sequentially Organized and ISAM File Sizes 6-29
Contents of an ISAM File 6-31
Deleting Duplicate Records Caused by a Disk Error During an ISAM Add Operation 6-34
Tips for Assembler Language Programmers 6-35
Grouping of Assembler Mnemonics 6-35
Assembler Program Use of Index Register 3 . . . 6-35
Double Buffering in Assembler Programs 6-35
Assembler Program Use of 1403 Conversion Subroutines 6-37
Writing ISSs and ILSs 6-37
Assembler INT REQ Service Subroutine 6-45
Tips for FORTRAN Programmers 6-48
Tips for Use of the EQUAT Control Record . . . 6-48
Invalid Characters in FORTRAN Source Cards . . . 6-49
FORTRAN Object Program Paper Tape Data Record Format 6-49
Keyboard Input of Data Records During FORTRAN Program Execution 6-50
FORTRAN Program Control of the Console Printer . 6-50
Length of FORTRAN DATA Statement 6-51
// Records Read During FORTRAN Program Execution 6-51
FORTRAN I/O Errors 6-51
Dumping FORTRAN DSF Programs to Cards . . . 6-52
RPG Object Program Considerations 6-52

Chapter 7. Operating the 1130 Disk Monitor System . . 7-1

Readying the 1131 Central Processing Unit 7-2
Readying the 2310 Disk Storage Drive 7-3
Readying the 2311 Disk Storage Drive 7-3
Readying the 1132 Printer 7-4
Readying the 1403 Printer 7-4
Readying the 1442 Model 6 and 7 Card Read Punch . . 7-4
Readying the 1442 Model 5 Card Punch 7-5
Readying the 2501 Card Reader 7-5
Readying the 1134 Paper Tape Reader 7-5
Readying the 1055 Paper Tape Punch 7-6
Readying the 1627 Plotter 7-7
Readying the 1231 Optical Mark Page Reader 7-8
Cold Start Procedure 7-9
Card System Cold Start Procedure 7-10
Paper Tape System Cold Start Procedure 7-10
Using the 1130 with the Monitor System 7-11
Entering Jobs from the Card Reader 7-11
Entering Jobs from the Paper Tape Reader 7-11
Entering Jobs from the Console Keyboard 7-11
Functions of Console Operator Keys During Monitor System Control 7-12
Displaying or Altering the Contents of a Selected Core Location 7-13
Manual Dump of Core Storage 7-13

Chapter 8. Monitor System Initial Load and System Reload 8-1

IBM-Supplied System Loader Control Records 8-2
SCON and TERM Control Records 8-2
Phase Identification (PHID) Control Records . . . 8-3
Type 81 Control Record 8-7
System Loader Control Records that you Punch . . . 8-7
Load Mode Control Record 8-8
System Configuration Control Records 8-9
CORE Control Record 8-10
Preparation of Load Mode and System Configuration Control Tapes 8-10
Card System Initial Load Operating Procedure 8-15
Card System Reload Operating Procedure 8-19
Card System Preload Operating Procedure 8-25
Paper Tape System Initial Load Operating Procedure . . 8-28
Paper Tape System Reload Operating Procedure . . . 8-33

Chapter 9. Stand-alone Utility Programs 9-1

Console Printer Core Dump 9-1
Printer Core Dump Program 9-4
Disk Cartridge Initialization Program (DCIP) 9-8
Disk Initialization Subroutine 9-8
Disk Copy Subroutine 9-8
Disk Dump Subroutine 9-9
Disk Patch Subroutine 9-9
Disk Analysis Subroutine 9-9
Disk Compare Subroutine 9-9
DCIP Operating Procedures 9-9
Paper Tape Reproducing Program 9-42
Stand-alone Paper Tape Utility Program (PTUTL) . . . 9-46

Chapter 10. Remote Job Entry Program 10-1

Machine and Device Requirements 10-1
Communication Considerations 10-1
Communication Considerations for Switched Lines . 10-2
Input at the Work Station 10-2
Generation of the 1130 RJE Work Station Program . 10-3
JECL for the 1130 Work Station 10-5
End-of-File Indicators 10-6
Output to the Work Station 10-6
Discontinuing and Continuing Output 10-7
User-Exit Subroutine 10-8
Operating Procedures 10-9
Work Station Startup 10-9
The Null Command 10-10
Console Keyboard Procedures 10-10
Error Recovery Procedures 10-11
Restart Procedures 10-11
Messages Sent to Work Stations 10-12
RJE Program Console Entry Switches 10-12
Error Statistics 10-12

Appendix A. Monitor System Operational and Error Messages A-1

Assembler Error Codes and Messages A-2
FORTRAN Messages and Error Codes A-7
DUP and MUP Messages and Error Messages A-13
System Loader Messages and Error Messages A-22
Satellite Graphic Job Processor Error Messages A-26
RJE Messages and Error Messages A-27
Supervisor Messages and Error Messages A-35
RPG Compiler Messages and Error Notes A-38
Core Load Builder Messages A-54
Auxiliary Supervisor Error Messages A-58
Monitor System Library Mainline Programs Messages and Error Messages A-59
IDENT Messages A-59
DISC Messages and Error Messages A-59
ID Messages and Error Messages A-60
COPY Messages and Error Messages A-60
DLCIB Messages and Error Messages A-61
MODIF Messages and Error Messages A-61
MODSF Messages and Error Messages A-65
DFCNV Messages and Error Messages A-67

Appendix B. Monitor System Error Wait Codes . . . B-1

Cold Start Program Error Waits B-1
ISS Subroutine Preoperative Error Waits B-2
I/O Device Subroutine Errors B-5
1442 Card Subroutine Errors B-5
2501 Card Subroutine Errors B-8
Console Printer Subroutine Errors B-8
Paper Tape Subroutine Errors B-9
Card Core Image Loader Wait Code B-9
Paper Tape Utility Program (PTUTL) Error Wait Codes . B-9
FORTRAN I/O Wait Codes B-10
RPG Object Program Wait Codes B-12

Appendix C. Monitor System Library Listing C-1

Appendix D. LET/FLET D-1

LET/FLET Disk Format D-1
LET/FLET Dump Format D-2

Appendix E. System Location Equivalence Table (SLET) . E-1

Appendix F. Core Dump F-1

Appendix G. Resident Monitor (Including Table of Equivalences) G-1

Appendix H. Monitor System Sample Programs . . . H-1

1. FORTRAN Sample Program H-1
   FORTRAN Sample Program Run on 4K H-2
   FORTRAN Sample Program Run on 8K H-5
2. Assembler Sample Program H-7
3. RPG Sample Program H-9
4. Using FORTRAN Unformatted I/O H-12
5. Processing on One Disk Drive a File that Extends over Two Cartridges H-13
6. Processing on Two Disk Drives a File that Extends over Two Cartridges H-16
7. Calculating ISAM File Parameters H-17

Appendix I. Formats I-1

Disk Formats I-2
Card Formats I-6
Paper Tape Formats I-11
Print Formats I-13
Data Formats I-15

Appendix J. Field Type Examples for DFCNV J-1

I-Field Type J-1
J-Field Type J-2
R-Field Type J-2
B-Field Type J-4
C-Field Type J-4
D-Field Type J-5
E-Field Type J-6
F-Field Type J-6
X-Field Type J-7

Appendix K. Decimal and Hexadecimal Disk Addresses . K-1

Appendix L. Disk Storage Unit Conversion Factors . . L-1

Appendix M. Character Code Set M-1

Glossary-Index X-1

GC26-3717-9 UPDATED BY VERSION 2 MODIFICATION 11

2311 Disk Storage Drive

New Hardware Feature. The 2311 Disk Storage Drive is a new feature that adds a larger online storage capacity and quicker online storage retrieval.

DCIP Function

New Programming Feature. The DCIP initialize and copy functions now have a wait for verifying that the console entry switches you turn on for the physical drive number and cartridge ID are correct before initialization and copying begins.

FORTRAN Messages

New Programming Feature. Messages describing errors in FORTRAN statements now indicate which statement is in error.

Chapter 1. How to Use This Publication

Chapters 2, 3, and 4 include information for the systems planner who is interested in the contents and organization of disks, core storage, and the functions of the programs and storage areas that comprise the IBM 1130 Disk Monitor System, Version 2. The information in these chapters assists you in planning the contents of your disks, as well as maintaining them. The disk maintenance programs are described in Chapter 4.

Chapters 5 and 6 contain information that is frequently referenced by programmers. Chapter 5 contains descriptions of all control records that control the functions of the disk monitor system (DM2). Use the programming tips and techniques in Chapter 6 for more efficient use of DM2.

Chapters 7, 8, and 9 include operating information for using the disk monitor system. Chapter 7 contains procedures for readying the devices that are a part of your computing system, for performing a cold start of the monitor system, for entering jobs and for displaying, altering, and dumping core storage.

Sample procedures for loading and reloading the system are shown in Chapter 8. You may use these operating procedures as they are presented, or modify them to meet the needs of your computing system.

Chapter 9 describes stand-alone utility programs. These programs provide for dumping core storage to a print device, for initializing, copying, patching, analyzing, dumping and comparing disks, and for punching paper tapes. Operating procedures for using the utility programs are listed.

The functions of the flowchart blocks that are used in the sample procedures in Chapters 7, 8, and 9 are:

The steps of the procedure that you perform. Each block contains a heading that describes the purpose of the block.

A system action that occurs during a procedure.

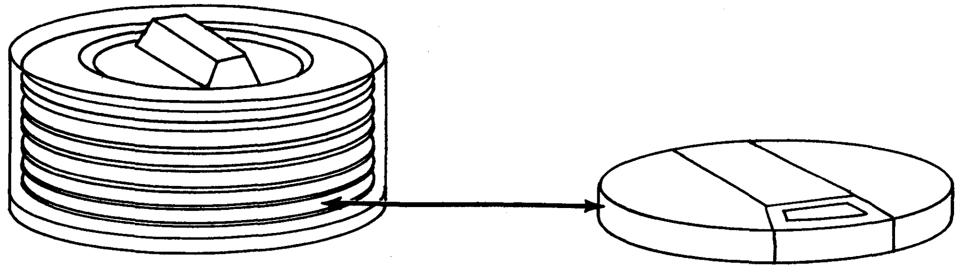
References procedures that are described elsewhere in this publication.

Chapter 10 describes the 1130 RJE Work Station Program.

When errors occur during monitor system processing, refer to Appendix A for error messages and codes, and to Appendix B for wait codes displayed on the console display panel.

The remaining appendixes contain information that you will need to reference at various times, such as, the names of the programs and subroutines in the system library and listings of LET, FLET, SLET, the resident monitor, and sample programs.

The terms *disk, disk cartridge,* and *cartridge* are used in this publication to refer to the single disk in an IBM 2315 Disk Cartridge or to any one of the 3 or 5 usable disks in an IBM 1316 Disk Pack, Model 12 or 11, respectively. Each usable disk in a 1316 Disk Pack is treated by DM2 as one 2315 disk, thus:



A disk in an IBM 1316 Disk Pack is the same as one IBM 2315 Disk Cartridge.

Each disk in the 1131 CPU and 2310 Disk Storage or 2311 Disk Storage Drive is assigned a physical drive number when the devices of an 1130 computing system are installed. Physical drive numbers are assigned in this order:

	Disk locations		
Physical drive number	*1131 CPU*	*2310 Disk Storage* or *2311 Disk Storage Drive*	
0	Internal disk		
1	———	First 2310, first disk	First 2311, first disk
2	———	First 2310, second disk	First 2311, second disk
3*	———	Second 2310, first disk	First 2311, third disk*
4*	———	Second 2310, second disk	First 2311, fourth disk*
5	———	———	First 2311, fifth disk
6	———	———	Second 2311, first disk
7	———	———	Second 2311, second disk
8*	———	———	Second 2311, third disk*
9*	———	———	Second 2311, fourth disk*
10	———	———	Second 2311, fifth disk

*Not used when a 2311 Disk Storage Drive is a Model 12

From one to 5 of these disks, depending on the configuration of your computing system, can be specified for use by assigning logical drive numbers to them. You assign logical drive numbers to disks with a // JOB monitor control record or when you code your program to call SYSUP (see "// JOB" in Chapter 5 and "SYSUP" in Chapter 6). The logical drive numbers do not have to be assigned in the same order as the physical drive numbers. The organization of disks is discussed in Chapter 2.

All hexadecimal addresses in this manual are shown in the form /xxxx.

Symbolic addresses rather than absolute addresses are used throughout this publication. Certain constants are also denoted symbolically. Appendix G contains a listing of the resident monitor.

$xxxx	All symbolic labels whose first character is a dollar sign ($) are found in the core communications area (COMMA).
#xxxx	All symbolic labels whose first character is a number sign (#) are found in the disk communications area (DCOM).
@xxxx	All symbolic labels whose first character is a commercial at sign (@) are considered to have absolute values (such as @HDNG refers to the page heading sector, sector 7, and thus has a value of 7).

Note. The number sign and commercial at sign are not included in the 1403 Printer or 1132 Printer character set; therefore, an equal sign (=) replaces the # and an apostrophe (') replaces the @ in printer listings.

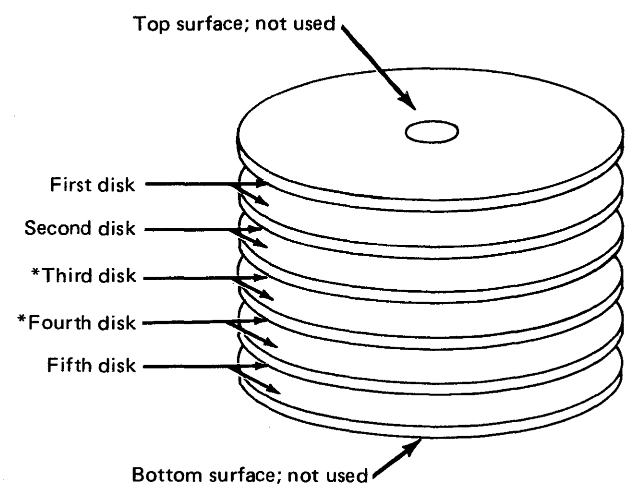
Chapter 2. Disk Organization

Two disk devices are used by the IBM 1130 Disk Monitor System, Version 2 (DM2):

- The IBM 2315 Disk Cartridge in an IBM 1131 Central Processing Unit internal disk drive and in IBM 2310 Disk Storage drives
- The IBM 1316 Disk Pack in IBM 2311 Disk Storage Drives, Models 11 and 12

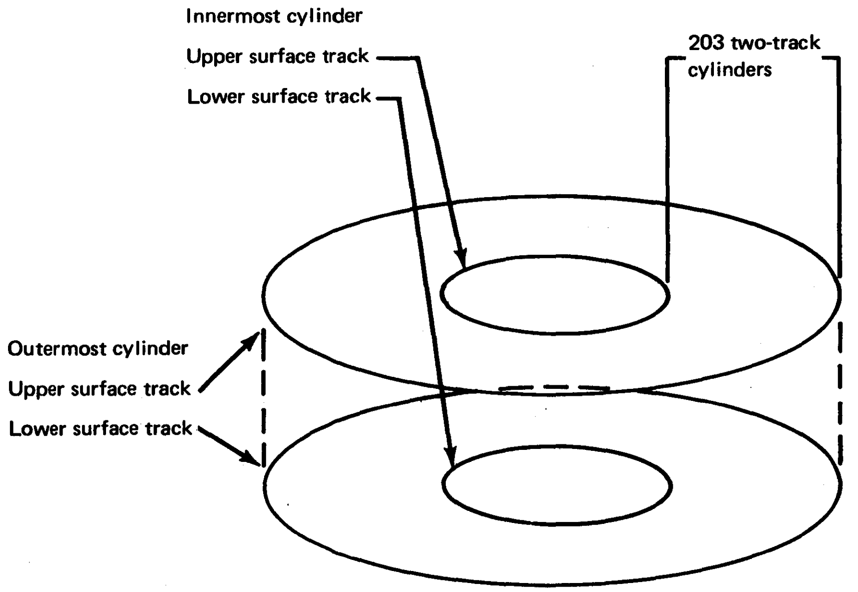
An IBM 2315 Disk Cartridge contains a single disk on which DM2 stores information on the top and bottom surfaces.

An IBM 1316 Disk Pack contains 6 disks mounted on a vertical shaft. The top surface of the top disk and the bottom surface of the bottom disk cannot be used for recording data, which leaves 10 possible recording surfaces. The monitor system programs consider the lower surface of one disk and the top surface of the disk immediately below as a *disk* (disk cartridge or cartridge). The arrangement of disks in a 1316 Disk Pack is illustrated by:



*The third and fourth disks are not used if the 2311 Disk Storage Drive is a Model 12.

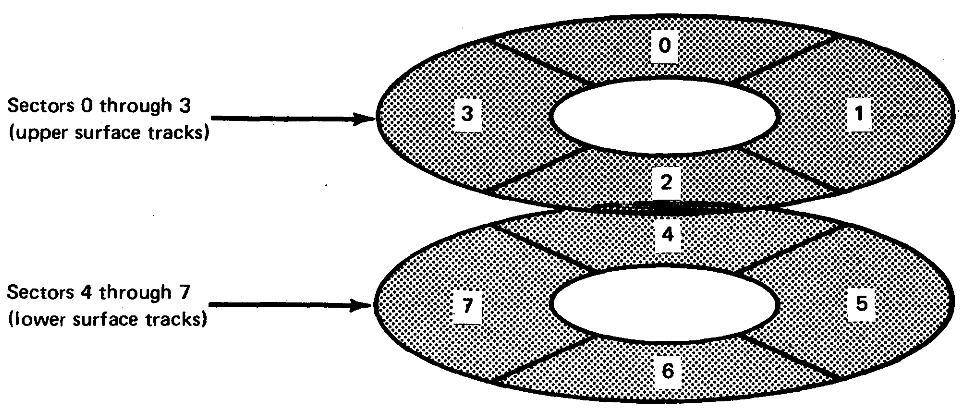
The storage area of all disks used by DM2 is arranged into circular patterns called *tracks.* Two tracks one above the other constitute a *cylinder.* A disk contains 203 concentric cylinders; 200 of these are available to the monitor system. The 3 remaining are reserved for use if defective cylinders are detected. The following illustrates the innermost and the outermost cylinders on a disk.



Note. The thickness of the disk has been greatly exaggerated in order to show the relative positions of the upper and lower surface tracks.

To complete the picture, the 201 intermediate cylinders, or pairs of tracks, should be visualized; they are omitted for the sake of clarity of the diagram.

For convenience in transferring data between core storage and disk storage, each track is divided into 4 equal segments. These segments are called *sectors.* Thus, each cylinder consists of eight sectors. Sectors 0 through 3 divide the upper surface track and 4 through 7 divide the lower. The following illustrates how sectors are numbered.



A sector contains 321 data words. The first data word is used for the sector address. This address is the number of that sector, counted in sequence from sector 0 on cylinder 0. Another unit of storage within a sector is the *disk block*. Each sector is divided into 16 disk blocks, each 20 words long. A disk storage word contains 16 data bits. The organizational components of disk storage are shown by the following chart.

No. of ↓ Per →	Word	Disk block	Sector	Track	Cylinder	Disk
Bits	16	320	5,112	20,480	40,960	8,192,000
Data words		20	320 [1]	1,280	2,560	512,000
Disk blocks			16	64	128	25,600
Sectors				4	8	1,600
Tracks					2	400
Cylinders						200

[1] **These follow the first actual word of each sector, which is used for the address.**

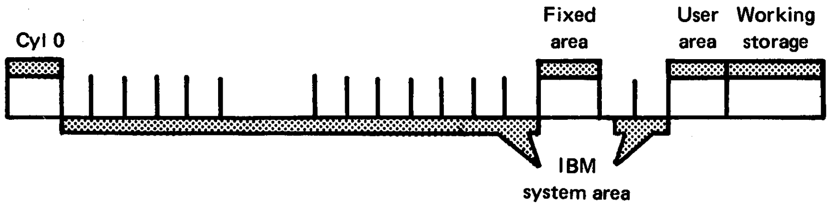
Before continuing with the descriptions of the contents of disk cartridges used by the monitor system, several terms must be defined.

- *System cartridge.* An initialized cartridge that contains the IBM 1130 Disk Monitor System. If your 1130 has only one disk (the internal disk in the 1131 CPU), all cartridges must be system cartridges.
- *Nonsystem cartridge.* An initialized cartridge that does not contain the monitor system.
- *Master cartridge.* A system cartridge that is designated as logical drive 0 by the *cold start* program, or by a monitor // JOB control record. This cartridge continues in use until another cold start, another // JOB control record, or a CALL instruction to SYSUP switches control to a different system cartridge. The disk on an 1130 with only one disk drive (the internal disk in the 1131 CPU) is both a system and a master cartridge.
- *Note:* If your system has only one disk drive (the internal disk in the 1131 CPU, or one 2311), you should cold start after changing cartridges, or packs, to avoid possible errors in the location of disk areas on system cartridges.
- *Satellite cartridge.* On an 1130 with more than one disk drive, this is any cartridge that *is not* the master cartridge. This cartridge can be either a system or a nonsystem cartridge.

The organization of programs and areas on system and nonsystem cartridges is described and illustrated in the following text.

SYSTEM CARTRIDGE

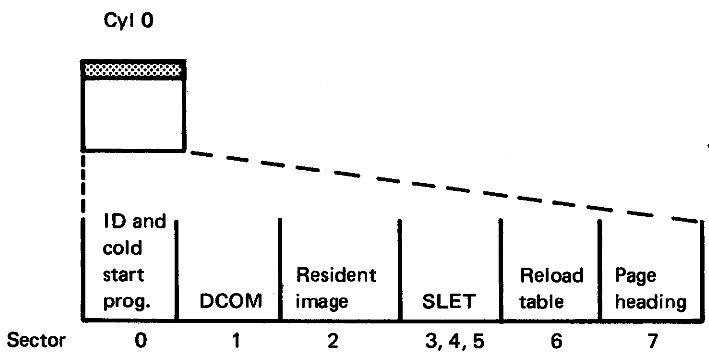
A system cartridge is divided into 5 logical areas as illustrated by the following:



Each area is described in the following text. The last section of this chapter, "Summary of the Contents of Disk Cartridges," contains a chart that indicates when these areas are present, or can be removed, on system cartridges.

Cylinder 0 on a System Cartridge

The contents of cylinder 0 on a system cartridge are defined during disk initialization and system load. The contents of cylinder 0 are as follows:



Sector	Label
0	@ IDAD
1	@ DCOM
2	@ RIAD
3, 4, 5	@ SLET
6	@ RTBL
7	@ HDNG

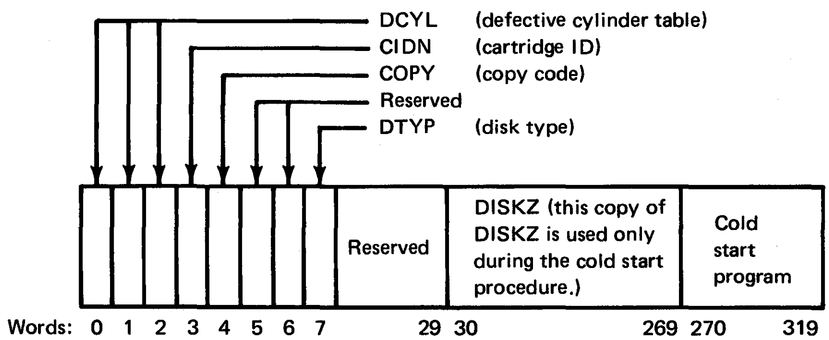
The following is a discussion of each sector.

sector @IDAD

Sector @IDAD on a system cartridge consists of:

- The defective cylinder table
- The cartridge ID
- The cartridge copy code
- The disk type
- A reserved area
- The DISKZ system device subroutine
- The cold start program

The contents of sector @IDAD on a system cartridge are shown in the following illustration.



The *defective cylinder table* (DCYL) contains the addresses of the first sector of any cylinders that are not capable of accurately storing data. This table is defined during disk initialization. If no defective cylinders are found, each of the 3 words of DCYL contains /0658 (hexadecimal). A cartridge with a maximum of 3 defective cylinders can be used by the monitor system.

The *cartridge ID* (CIDN) is a hexadecimal number in the range /0001 through /7FFF that uniquely identifies the cartridge. The ID is placed on a cartridge when the cartride is initialized.

The *cartridge copy code* (COPY) identifies the copy number of a cartridge that has been copied from another cartridge. When a disk is initialized, this word is zero. Each time the disk is copied, word 5 of the cartridge being copied to is incremented by one; that is, the copy code of the receiving disk is one greater than the copy code of the source cartridge.

The *reserved areas* of sector @IDAD are for possible future expansion.

The *disk type* (DTYP) is a code that indicates whether or not the disk is a system cartridge. The appropriate code is placed in DTYP when the cartridge is initialized by DCIP or DISC and when the monitor system is loaded onto the disk.

The *DISKZ subroutine* is stored in sector @IDAD and in the system device subroutine area in the IBM system area (see "IBM System Area on a System Cartridge" in this chapter) when the monitor system is loaded on the disk. The cold start program uses DISKZ stored in sector @IDAD. All other times that DISKZ is called, the copy stored in the system device subroutine area is used.

The *cold start program* is placed in sector @IDAD when the monitor system is loaded onto the disk.

sector @DCOM

Sector 1 contains the *disk communications area* (@DCOM). This area contains parameters that are passed from one monitor program to another. These parameters contain information such as:

- The number of LOCALs associated with the program in working storage
- The temporary job indicator switch
- The cartridge IDs for cartridges on the system
- The format of programs in working storage for all cartridges on the system
- The block count of the programs in working storage for all cartridges on the system

These parameters are listed in Appendix I. They are set and reset during the processing of JOB monitor control records or during the DCOM update operation called SYSUP. The parameters obtained from nonsystem disks are merged into DCOM on the master cartridge during one of the previous operations. The parameter table entries for the nonsystem disks are cleared to zero.

sector @RIAD

Sector 2 contains the *resident image* (@RIAD). The resident image is a copy of the skeleton supervisor and the COMMA portion of the resident monitor. (A description of the resident monitor is in Chapter 3, "Monitor System Programs.") The resident image is used to initialize the resident monitor during a cold start.

SLET

Sectors 3, 4, and 5 are the *system location equivalence table* (@SLET). SLET is composed of an identification number, core loading address, word count, and sector address for every phase of every monitor program. Chapter 4 contains information about obtaining a listing of SLET, and a sample of a SLET printout is in Appendix E.

sector @RTBL

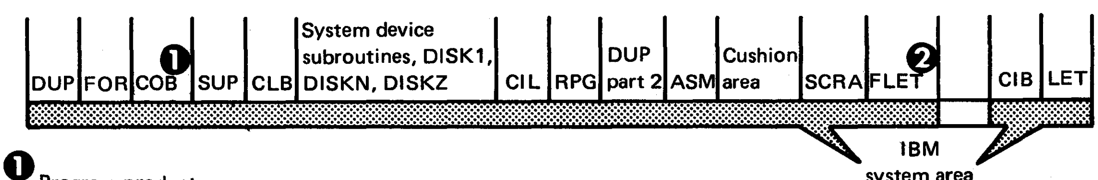
Sector 6 is the *reload table* (@RTBL). This table is established during an initial system load. @RTBL contains a 3-word entry for each monitor system program phase that requests SLET information during a load or reload operation. Each entry consists of the ID number of the requesting phase, the location in the requesting phase where the SLET information is to be placed, and the number of SLET entries to be inserted. The reload table is updated during a system reload when phases that request SLET information are added or modified. The last entry in the reload table is followed by the hexadecimal word /FFFF.

sector @HDNG

Sector 7 (@HDNG) is used to store the heading that appears at the top of each page printed by monitor programs other than RPG.

IBM System Area on a System Cartridge

Monitor programs and disk areas are loaded onto a disk during a system load. This entire area is called the IBM system area, and is illustrated by the following:



❶ Program product

❷ FLET is contained on a disk only if a fixed area is defined on the disk. See "Fixed Area" in this chapter.

The monitor programs in this area are described in Chapter 3. These programs are:

- Disk utility program (DUP)
- FORTRAN compiler (FOR)
- COBOL compiler (COB) program product
- Supervisor (SUP)
- Core load builder (CLB)
- Core image loader (CIL)
- RPG compiler (RPG)
- Assembler (ASM)

The disk areas of the IBM system area are described in the following text.

system device subroutine area

The *system device subroutine area* consists of the following:

- The subroutines used by the monitor programs to operate these print devices

  1403 Printer
  1132 Printer
  Console Printer

- The subroutines used by the monitor programs to operate these I/O devices

  2501 Card Reader/1442 Card Punch, Model 5, 6, or 7
  1442 Card Read/Punch, Model 6 or 7
  1134 Paper Tape Reader/1055 Paper Tape Punch
  Console Keyboard/Printer

- The I/O character code conversion subroutines used in conjunction with the I/O subroutines for these devices

  2501 Card Reader/1442 Card Punch
  1134 Paper Tape Reader/1055 Paper Tape Punch
  Console Keyboard/Printer

- The disk I/O subroutines

  DISKZ
  DISK1
  DISKN

All of the subroutines in the system device subroutine area, except the disk I/O subroutines, are naturally relocatable and are intended for use only by monitor programs. The disk I/O subroutines are located in this area rather than in the monitor system library because they are processed by the core load builder differently from subroutines stored in the monitor system library.

DISKZ is stored twice on a system cartridge; once in sector @IDAD with the cold start program, and once in the system device subroutine area with DISK1 and DISKN. Cold start uses DISKZ in sector @IDAD; all other times that DISKZ is called, the copy that is stored in the system device subroutine area is used.

cushion area

The *cushion area* immediately follows the system programs and provides for the possible expansion of the monitor system programs in a reload operation. This area occupies the remaining sectors of the last cylinder occupied by the system programs, plus the next complete cylinder.

SCRA

The *supervisor control record area* (SCRA) is the area in which supervisor control records (LOCAL, NOCAL, FILES, G2250, and EQUAT) are saved. These records, except the EQUAT record, are read from the input stream (following an XEQ or STORECI control record) and are stored in the SCRA for subsequent processing by the core load builder. The processing of the EQUAT record is similar to that of the other supervisor control records, but it is read from the input stream following a JOB control record.

FLET

The *fixed location equivalence table* (FLET) is a directory to the contents of the fixed area for the cartridge on which it appears. There is one FLET entry for:

- Each program stored in disk core image (DCI) format
- Each data file stored in disk data format (DDF)
- The padding required to permit a DCI program or data file to be stored beginning on a sector boundary

Each FLET entry includes:

- The name of the DCI program or the data file
- The format of the program or data file
- The size, in disk blocks, of the program or data file
- The disk block address of the program or data file

Each cartridge on which you define a fixed area has a FLET (see "Fixed Area" in this chapter). Regardless of the fixed area sizes FLET occupies the cylinder preceding the beginning of the fixed area.

The sector address of the first sector of FLET on a given cartridge is obtained from the location equivalence table (LET). The last item (#FLET) in the first header line of a LET dump contains this sector address. A listing of a LET/FLET dump is in Appendix D.

CIB

The *core image buffer* (CIB) is the disk area in which the portion of a core load that is to reside in core storage below decimal location 4096 in a 4K system (decimal location 5056 in larger systems) is built by the core load builder. The CIB is also used by the core image loader during the transfer of control from one link to the next to save any COMMON defined below decimal location 4096 or 5056.

LET

The *location equivalence table* (LET) is a directory to the contents of the *user area* on the cartridge. On a system cartridge, LET occupies the cylinder preceding the user area. There is one LET entry for:

- Each program stored in disk system format (DSF)
- Each program stored in disk core image (DCI) format
- Each data file stored in disk data format (DDF)
- The padding required to permit a DCI program or data file to be stored beginning on a sector boundary

Each LET entry includes:

- The name of the program or data file
- The format of the program (DSF or DCI) or data file
- The size in disk blocks of the program or data file
- The disk block address of the program or data file

A listing of a LET/FLET dump is contained in Appendix D. The starting location of the beginning of LET on each disk on the system is included in the resident monitor.

Fixed Area

The *fixed area* (FX) is the area in which you store programs and data files when you want them to occupy the same sectors at all times. Programs stored in this area must be in disk core image (DCI) format. This is an optional area and is defined on any 1130 cartridge by the use of the DEFINE FIXED AREA operation of the *Disk Utility Program* (DUP). This DUP operation is also used to increase or decrease the size of the fixed area. (See Chapter 3, "Monitor System Programs" for a description of DUP operations.) The contents of the fixed area are illustrated by the following:

Fixed area
Your programs and data files

A program or data file stored in the fixed area starts at the beginning of a sector. When a program or a data file is deleted from this area, the fixed area is not packed. Programs and data files stored in this area reside at fixed sector addresses and can be referred to by sector address.

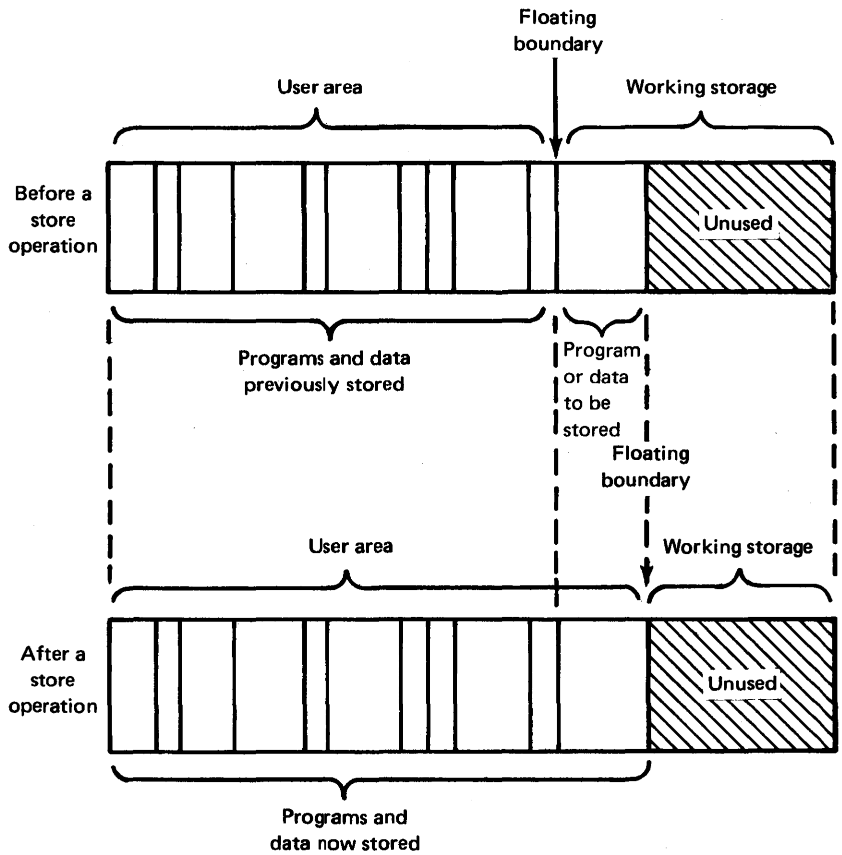
User Area and Working Storage

UA The user area (UA) on a system cartridge contains the monitor system library and programs and data files that you write and store there. Programs are stored in this area in disk system format (DSF) or in disk core image (DCI) format. Data files are stored in disk data format (DDF). The following illustrates the user area and working storage.

User area	Working storage
Monitor system library Your programs and data files	

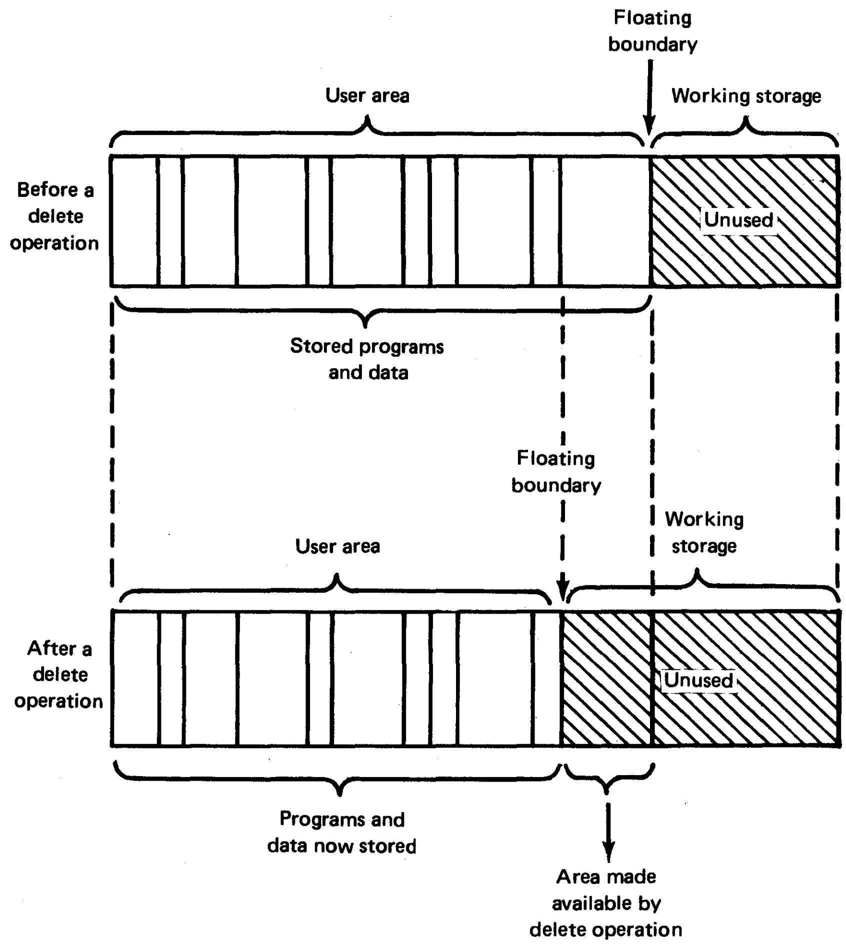
The user area is defined on any 1130 cartridge during disk initialization. The monitor system library is placed in this area during an initial system load. This area occupies as many sectors as are required to contain the system library plus any user programs and/or data files that are stored there.

When a program or a data file is entered, it is placed at the beginning of working storage; that is, immediately following the end of the user area. The area occupied by the new program or data file is then incorporated into the user area during a store operation. Working storage is decreased by the size of the program or data file. The following illustrates the contents of the user area and working storage before and after a store operation.



DSF programs are stored in the user area starting at the beginnning of a disk block; DCI programs and data files are stored starting at the beginning of a sector.

The user area is packed when a program or data file is deleted from this area; that is, the programs and data files are moved so as to occupy the area formerly occupied by the deleted program or data file. During packing, DSF programs are moved to the first disk block boundary in the vacancy; DCI programs and data files are moved to the first sector boundary. All remaining programs and data files are similarly packed. The area gained by packing the user area is returned to working storage as illustrated by:

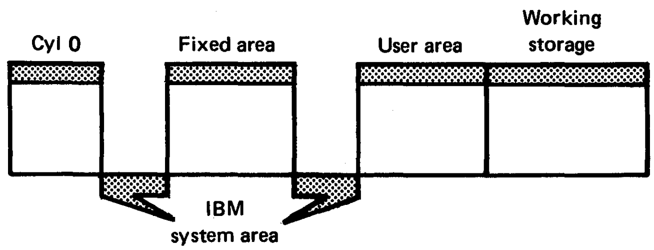


WS

On all cartridges, *working storage* (WS) is the area that is not defined as cylinder 0, the IBM system area, the fixed area, or the user area. Working storage is available to monitor programs and user programs alike as temporary disk storage. This area extends from the sector boundary immediately following the user area to the end of the cartridge.

NONSYSTEM CARTRIDGE

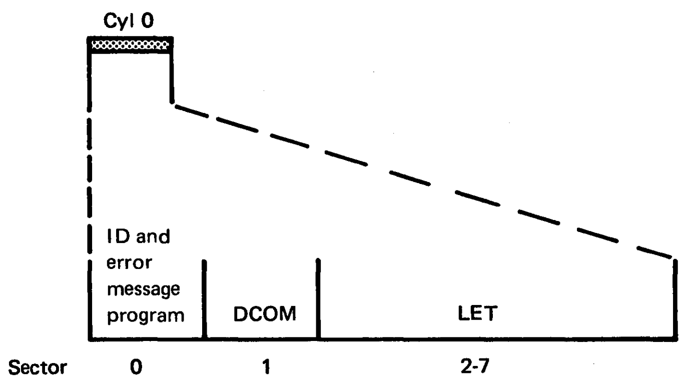
A nonsystem cartridge on an 1130 that has more than one disk drive can be used exclusively for the storage of data and/or programs, and is called a satellite cartridge. The 5 logical areas of a nonsystem cartridge are:



The contents of cylinder 0 and the IBM system area are described in the following sections. The contents of the fixed area, the user area, and working storage are the same as described for system cartridges, except that the user area does not contain the monitor system library. The last section of this chapter, "Summary of the Contents of Disk Cartridges," contains a chart that indicates when these areas are present or can be removed.

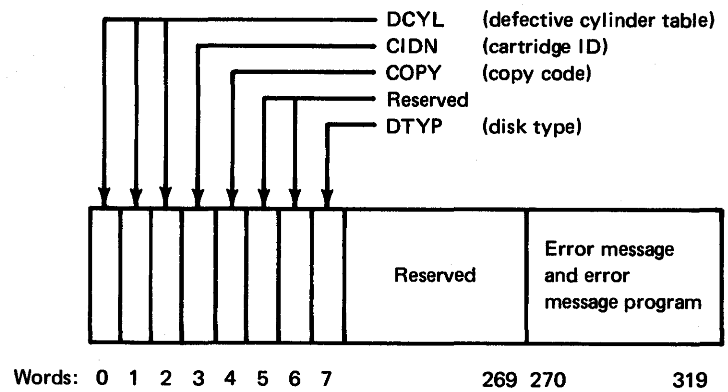
Cylinder 0 on a Nonsystem Cartridge

The contents of cylinder 0 on a nonsystem cartridge are established when the cartridge is initialized, and are illustrated by:



sector @IDAD

The first 8 words of sector @IDAD on a nonsystem cartridge are the same as described for a system cartridge. The remaining words of this sector are a reserved area, an error message program, and an error message. The error message is printed if an attempt is made to cold start a nonsystem cartridge. This message and the program that prints it plus part of the reserved area are overlaid by the cold start program and the DISKZ subroutine when the monitor system is loaded onto a cartridge. Sector @IDAD on a nonsystem cartridge consists of:



sector @DCOM

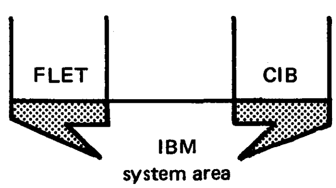
The information in sector @DCOM of cylinder 0 on a nonsystem cartridge is similar to a system cartridge. The difference is that the information on a nonsystem cartridge applies only to that cartridge.

LET

The remaining sectors of cylinder 0 are the *location equivalence table* (LET) for the cartridge. The contents of LET are described under the description of the IBM system area on a system cartridge.

IBM System Area on a Nonsystem Cartridge

The IBM system area of a nonsystem cartridge can contain the *fixed location equivalence table* (FLET) and the *core image buffer* (CIB). This area is illustrated by:



FLET

FLET is described under the description of the IBM system area on a system cartridge. This table is on a nonsystem cartridge only if you define a fixed area on the cartridge.

CIB

The CIB is described under the description of the IBM system area on a system cartridge. This area is optional on a nonsystem cartridge, and can be deleted with the disk maintenance program called DLCIB (see Chapter 4).

SUMMARY OF THE CONTENTS OF DISK CARTRIDGES

Figure 2-1 is a chart of the contents of the 5 logical areas of system and nonsystem cartridges. This chart indicates when these areas are present on system and nonsystem cartridges, and when it can be removed if the area is optional.

Logical area	Subareas	Present
Cylinder 0		On system and nonsystem cartridges
IBM system area	DUP SUP CLB System device subroutines CIL Cushion area SCRA	Only on system cartridges
	CIB	On system and nonsystem cartridges; can be removed from nonsystem cartridges
	Assembler	Only on system cartridges; can be removed
	FORTRAN compiler	Only on system cartridges; can be removed
	RPG compiler	Only on system cartridges; can be removed
	COBOL compiler (program product)	Only on system cartridges; can be removed
	LET	On system and nonsystem cartridges
	FLET	Only if a fixed area is defined by user
Fixed area (FX)	User programs User data files	Only if defined by user
User area (UA)	Monitor system library (only on system cartridges) User programs User data files	On system and nonsystem cartridges. As the result of a system load, the UA contains the monitor system library.
Working storage (WS)		On system and nonsystem cartridges

Figure 2-1. The 5 logical areas of disk cartridges

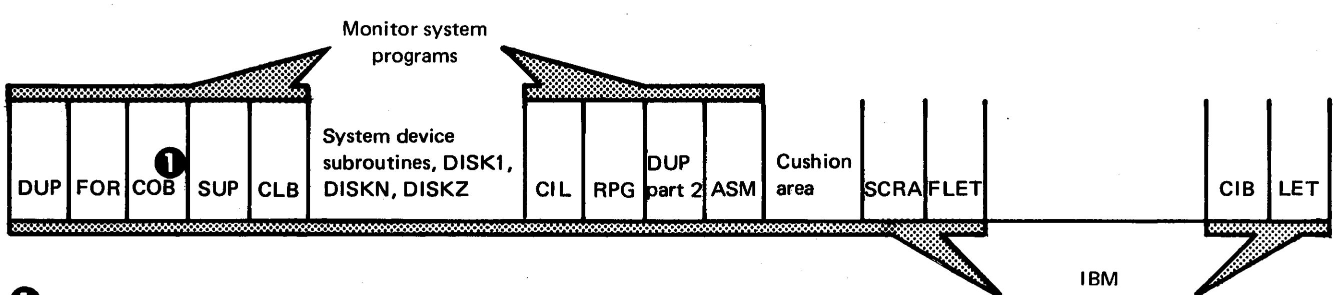
Chapter 3. Monitor System Programs

The IBM 1130 Disk Monitor System provides continuous operation of the 1130 computing system with minimal setup time and operator intervention. The monitor system consists of a system library and 7 interdependent system programs. The monitor system programs perform monitor control functions and include:

- The supervisor (SUP), which performs the control functions of the monitor system and provides the linkage between user programs and monitor programs.
- The Disk Utility Program (DUP), which performs operations that involve the disk, such as storing, moving, deleting, and dumping programs or data files or both.
- The assembler (ASM), which translates source programs written in 1130 Assembler language into object programs.
- The FORTRAN compiler (FOR), which translates source programs written in 1130 basic FORTRAN IV language into object programs.
- The RPG compiler, which translates programs written in 1130 RPG language into object programs.
- The core load builder (CLB), which constructs an executable core load from programs in disk system format (DSF). The DSF program and all associated subprograms are converted into disk core image (DCI) format, and the resultant core load is ready for immediate execution or for storing as a core image program.
- The core image loader (CIL), which transfers core loads into core storage for execution and serves as an interface between some monitor programs.

Although the COBOL compiler (COB) resides in the IBM system area when the monitor system is loaded onto a cartridge, the COBOL compiler is not a monitor program. It is an IBM program product.

A flowchart of the general logic flow of the monitor system programs is included under "Logic Flow of the Monitor System" at the end of this chapter. The monitor system library is a group of disk resident programs that performs I/O functions, data conversion, arithmetic functions, disk initialization, and maintenance functions. This library is discussed in Chapter 4, and the monitor system programs are discussed in the following text. The disk placement of these programs is shown by the following.



❶ Program product

SUPERVISOR

The *supervisor* is 2 groups of programs that control the monitor system and link the user and monitor programs. One portion of the supervisor, the skeleton supervisor, is stored in sector @RIAD of cylinder 0. The other portion of the supervisor is stored in the IBM system area.

The skeleton supervisor initially gains control of the monitor system through the cold start program. During a cold start, the skeleton supervisor is loaded from sector @RIAD into the resident monitor section of core storage.

Resident Monitor

The *resident monitor* resides at the beginning of core storage and contains (1) the core communications area (COMMA), (2) the skeleton supervisor, and (3) a disk I/O subroutine (DISKZ, DISK1, or DISKN). Appendix G is a listing of the resident monitor.

COMMA

The *core communications area* (COMMA) consists of parameters required by the core image loader to link from one core image program to another. These parameters are interspersed with parts of the skeleton supervisor in the resident monitor.

skeleton supervisor

The *skeleton supervisor* is interspersed with COMMA in the resident monitor and is composed of:

- Entry points for linking from one core load to another ($LINK), for linking from a core load to monitor system programs ($EXIT), and for dumping core storage ($DUMP).
- Interrupt level subroutines (ILS02 and ILS04) for handling interrupts on levels 2 and 4. Disk devices interrupt on level 2, and since disks are used in all operations of the monitor system, ILS02 is included. Since the console keyboard INT REQ key interrupts on level 4 and can be pressed at any time, the ILS04 subroutine for handling level 4 interrupts is included.
- A preoperative error trap that is entered by all interrupt service subroutines (ISS) when an error is detected before an operation is performed. The trap consists of a WAIT instruction and a branch instruction. (The address of $PRET+1 is displayed in the INSTRUCTION ADDRESS indicator on the console display panel during the wait.) Pressing PROGRAM START causes the branch to be taken, and execution resumes. (Under certain conditions, such as a FORTRAN PAUSE statement, this trap is entered when an error has not occurred.)
- Postoperative error traps (one for each interrupt level) that are entered by all ISS subroutines when an error is detected after an I/O operation has been started. Each trap consists of a WAIT instruction and a branch instruction. (The address of $PST1, $PST2, $PST3, or $PST4 plus one is displayed in the INSTRUCTION ADDRESS indicator on the console display panel during the wait.) Pressing PROGRAM START returns control to the ISS subroutine, which may retry the operation in error.
- The PROGRAM STOP key error trap that is entered when the PROGRAM STOP key is pressed (unless a user-written subroutine associated with interrupt level 5 is in core). If a higher level interrupt level is being serviced when PROGRAM STOP is pressed, the PROGRAM STOP interrupt is masked until the current operation is complete. This trap consists of a WAIT instruction and a branch instruction. (The address of $STOP+1 is displayed in the INSTRUCTION ADDRESS indicator on the console display panel during the wait.) Pressing PROGRAM START continues execution of the monitor system.

disk I/O subroutine

The *disk I/O subroutine* (DISKZ, DISK1, or DISKN) required by the program in control resides in core storage immediately following the skeleton supervisor. DISKZ is the subroutine used by all system programs. DISKZ is initially loaded into core storage with the resident image during a cold start.

Prior to the execution of a core load that requires DISK1 or DISKN, the core image loader overlays DISKZ with the required disk I/O subroutine. When control is returned to the supervisor, the core image loader overlays the disk I/O subroutine currently in core (if DISK1 or DISKN) with DISKZ. Source programs written in assembler, FORTRAN, RPG, or COBOL can call any of the 3 I/O subroutines; however, only one disk I/O subroutine can be referenced in a given core load. The entry in column 19 of an XEQ monitor control record specifies the version of the subroutine to be used during execution of the core load. (Monitor control records are described in Chapter 5.)

Disk-resident Supervisor Programs

The portion of the supervisor that resides in the IBM system area includes programs that analyze monitor and supervisor control records and perform the functions specified, the auxiliary supervisor, and the System Core Dump Program.

monitor control record analyzer

The *monitor control record analyzer* (1) reads a monitor control record from the input stream, (2) prints the control record on the principal print device, and (3) calls the required monitor system program and transfers control to it.

supervisor control record analyzer

The *supervisor control record analyzer* reads a supervisor control record from the input stream, and stores the information in the control record in the supervisor control record area (SCRA) on disk.

auxiliary supervisor

The *auxiliary supervisor* is used by the Cold Start Program, ILS04 subroutine, core image loader, and system loader as a pre-entry to the monitor control record analyzer. The auxiliary supervisor is entered via the $DUMP entry point in the skeleton supervisor. This program sets appropriate parameters in COMMA, writes dummy monitor control records (such as the JOB monitor control record printed during a cold start), and prints error messages for errors detected by the core image loader. Control is then transferred to the monitor control record analyzer through the $EXIT entry point in the skeleton supervisor.

Supervisor Core Dump Program

The *Supervisor Core Dump Program* provides a hexadecimal printout of the contents of core storage. (A portion of a core dump is shown in Appendix F.) This program is entered through the $DUMP entry point in the skeleton supervisor in 2 ways.

- A special calling sequence during execution of an Assembler or FORTRAN program (see the publications *IBM 1130 Assembler Language,* GC26-3778, and *IBM 1130/1800 Basic FORTRAN IV Language,* GC26-3715). The portion of core storage specified in the assembler or FORTRAN statements, or all of core storage if limits are not specified, is dumped. Execution of the core load in process then continues with the statement following the one that called the dump.
- A manual dump of core storage through $DUMP+1 (see "Manual Dump of Core Storage" in Chapter 7). The contents of core storage are dumped, and the dump program executes a CALL EXIT, which terminates the execution of the core load in progress.

DISK UTILITY PROGRAM

The Disk Utility Program (DUP) allows you to perform the following operations through the use of DUP control records:

- Store programs and data files on disks
- Make programs and data files on a disk available as printed, punched card, or punched paper tape output
- Delete programs and data files from a disk
- Determine the status of disk storage areas through a printed copy of LET and FLET
- Define a fixed area on a disk, and delete monitor system programs from a disk
- Maintain disk macro libraries
- Reassign sector addresses on a disk
- Reserve space for a data file or macro library

DUP control records are described in Chapter 6. DUP error messages are listed in Appendix A.

General Functions of DUP

DUP is called into operation when a DUP monitor control record (// DUP) is recognized by the supervisor. The control portion of DUP is brought into core to read the next DUP control record from the input stream. The DUP control record is printed and analyzed.

The DUP program required to perform the operation specified in the control record is read into core storage from the disk and assumes control. The DUP program performs the functions specified in the control record, and when complete, a message is printed on the principal printer, and control is returned to the control portion of DUP. The next control record is read from the input stream.

If the next record is a monitor control record, other than a comments control record (// *), system control is returned to the supervisor to process the record. Comments monitor control records are printed; blank records are passed. If the record is a DUP control record, DUP maintains control and reads the next record.

ASSEMBLER

The source language and macro capabilities for the assembler are described in the publication *IBM 1130/1800 Assembler Language,* GC26-3778. This section of this chapter contains only a general description of the Monitor System Assembler Program. Assembler control records are described in Chapter 6. Assembler error detection codes and error messages are listed in Appendix A.

The assembler can be deleted from the monitor system if desired (see "*DEFINE" under "DUP Control Records" in Chapter 5). The assembler cannot, however, be operated independently of the monitor system.

A monitor control record, // ASM, is used to call the assembler into operation. The assembler reads assembler control records and the source deck from the principal input device. The assembler interprets and performs the functions specified in the control records and translates the source program into an object program. Control records cause the assembler to:

- Pass the source deck through the assembler twice
- List the source deck and cross-reference symbol table on the principal printer
- Punch object decks into cards
- Print the symbol table on the principal printer, or punch the symbol table into cards
- Save and add to the symbol table on disk
- Specify the interrupt level for assembly of ISS subroutines
- Specify additional sectors for overflow of the symbol table
- Specify the length of COMMON used when linking between FORTRAN and assembler programs
- Specify the use of the macro library during assembly

After assembly is complete, the object program resides in working storage. The program can now be (1) called for execution, (2) stored in either the user area or the fixed area, or (3) punched as a binary deck or tape.

FORTRAN COMPILER

The source language for the FORTRAN compiler is described in the publication *IBM 1130/1800 Basic FORTRAN IV Language,* GC26-3715. This section of this chapter contains only a general description of the monitor system FORTRAN compiler. FORTRAN compiler control records are described in Chapter 6. FORTRAN error codes and error messages are listed in Appendix A.

The FORTRAN compiler can be deleted from the monitor system if desired (see "*DEFINE" under "DUP Control Records" in Chapter 5). The FORTRAN compiler, however, cannot be operated independently of the monitor system.

A monitor control record, // FOR, is used to call the FORTRAN compiler into operation. The compiler reads FORTRAN compiler control records and the source program from the principal input device. The compiler interprets and performs the functions specified in the control records and translates the source program into an object program. Control records cause the compiler to:

- Specify the I/O devices to be used during program execution
- List the source program, the names of all subprograms associated with the source program, and symbol table information on the principal print device
- Specify that all variables and real constants are stored in 3 words instead of 2
- Specify that all integer variables are stored in one word instead of the standard 2 words
- Print header information at the top of each printed page, and print the program name at the end of a listing
- Trace the values of variables, IF expressions, and computed GO TO statements during program execution
- Specify the origin of an absolute program

After compilation is complete, the program resides in working storage in disk system format (DSF). The program can now be (1) called for execution, (2) stored in the user area or fixed area, or (3) punched in binary form into cards or paper tape.

RPG COMPILER

The source language specifications for the RPG compiler are described in the publication *IBM 1130 RPG Language,* GC21-5002. This section of this chapter contains a general description of the monitor system RPG compiler. RPG compiler control cards are described in Chapter 6. RPG error messages and error notes are described in Appendix A.

The RPG compiler can be deleted from the monitor system if desired (see "*DEFINE" under "DUP Control Records" in Chapter 5). The compiler, however, cannot be operated independently of the monitor system.

A monitor control record, // RPG, is used to call the compiler into operation. The compiler reads the RPG compiler control card and the source program from the principal input device. The compiler interprets and performs the functions specified in the control card and translates the source program into an object program. After compilation is complete, the object program, in disk system format (DSF), resides in working storage. The program can now be (1) called for execution, (2) stored in the user area or the fixed area, or (3) punched in binary form into cards.

CORE LOAD BUILDER

The core load builder constructs an executable core load from a program in disk system format (DSF). The DSF program and all required subroutines (including any LOCALs, SOCALs, and NOCALs) are converted from disk system format into disk core image (DCI) format. The resultant core load is ready for immediate execution or for storing.

The core load builder is called by any of the following programs.

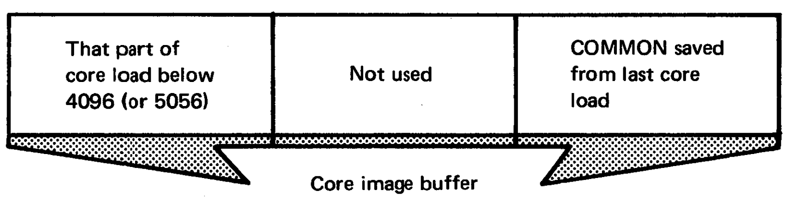
- *Supervisor.* When an XEQ monitor control record is read by the supervisor, the information specified in any supervisor control records that follow is written in the supervisor control record area (SCRA). Then, the core load builder is called to begin construction of the core load. When the core load is complete, the core image loader transfers the core load into core for execution.
- *Disk Utility Program.* When a STORECI control record is read by the Disk Utility Program (DUP), information specified in any supervisor control records that follow are written in the supervisor control record area (SCRA). Then, if the specified program is not in working storage, the program is loaded into working storage, and the core load builder is called to begin construction of the core load. When the core load is complete, DUP stores it as a core image program in the user area or fixed area as specified in the STORECI control record.
- *Core Image Loader.* When a core load calls for a link to another, the core image loader determines the format of the program from its LET or FLET entry. If the format is DSF, the core load builder is called to begin construction of the core image program. When the core load is complete, the core image loader transfers the core load for execution.

Construction of a Core Load

When the core load builder (CLB) is called by one of the previous monitor programs, the core load is constructed by the functions described in this section. The core load builder uses 3 storage areas while constructing a core load. These areas are the core image buffer (CIB), working storage (WS), and core storage.

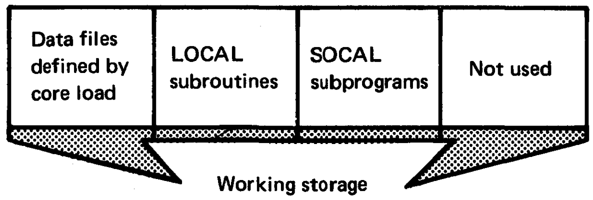
CLB use of the CIB

The core load builder places in the core image buffer the parts of a core load that are to reside below core location 4096 (decimal) for a 4K system, or 5056 for larger systems, during execution. These parts can be the core image header, the main-line program, and subroutines. The contents of the CIB during core load construction are illustrated by:



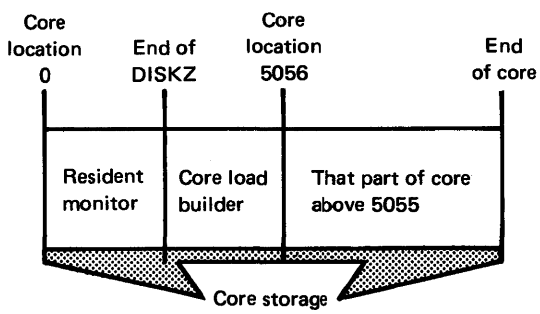
CLB use of WS

The core load builder reserves enough space in working storage for any data files that are specified for use by the core load, as well as any LOCAL and/or SOCAL subroutines that are referenced by the core load (see "Processing Data Files" and "Incorporating Subroutines" in this section). The contents of working storage during core load construction are shown by:

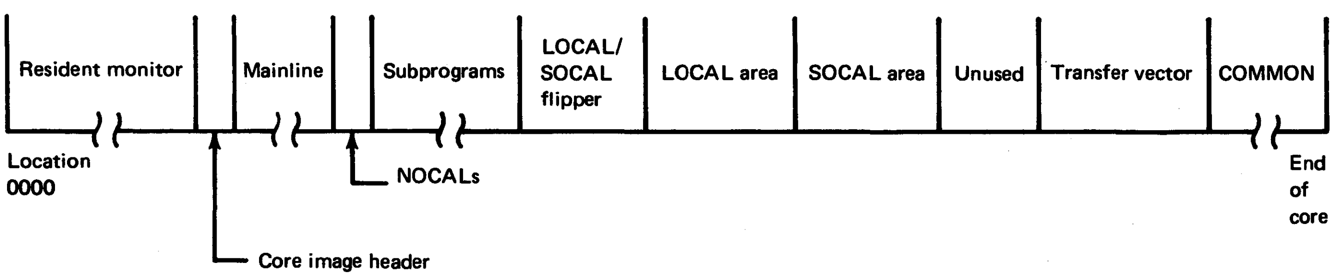


CLB use of core storage

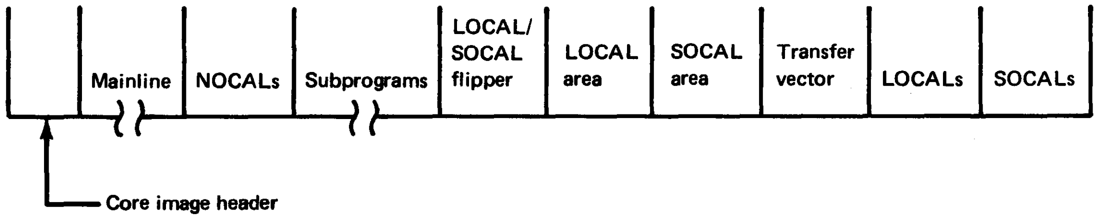
In systems larger than 4K, the core load builder places in core storage the parts of a core load that are to reside above core location 5055 during execution. These parts of a core load can be subroutines and the transfer vector. The contents of core storage during construction of a core load are illustrated by:



When construction of a core load is finished and is executed immediately, the core image loader is called to transfer it into core storage. The layout of a core load in core that is ready for execution is illustrated by:



When a core load is stored immediately following construction, it is placed in the user area or the fixed area as follows:



When the core load builder is called, the core load is built by the following functions, but not necessarily in the order described.

Construction of the Core Image Header

The core image header is established at the beginning of the construction of a core load. Throughout the building of a core load, information is placed in this header. The information placed in the header is used by the core image loader to transfer the core load into core storage and start program execution. The core image header is a part of the core load and resides in core storage during execution.

Note. The area of core storage occupied by the core image header should not be considered as a work area, because FORTRAN subroutines access information in the header during execution.

Assignment of the Origin of a Core Load

The core location where the core image loader begins loading a relocatable core image program is assigned by the core load builder. This loading address is placed in the core image header, and is called the origin. The origin is determined by adding decimal 30 to the next higher-addressed word above the end of the disk I/O subroutine used by the core load. The following chart lists the origin locations (in decimal and hexadecimal) used by the core load builder.

Disk I/O subroutine in core	Core load origin	
	Decimal	Hexadecimal
DISKZ	510	/01FE
DISK1	690	/02B2
DISKN	960	/03C0

The origin of absolute programs is assigned by the assembler or FORTRAN programmer, not by the core load builder. The assembler programmer assigns the origin of a program with the ORG statement in his program. The FORTRAN programmer defines the origin of his program with an *ORIGIN control record. The origin that you define must not be less than those in the preceding chart, depending on the disk I/O subroutine used by the core load. When the programmer assigns an origin, the addresses printed in a program listing are absolute; thus, he can see exactly where his statements and constants are in core during execution.

Note. When DISKZ is in core, the assembler programmer must specify an *even* address in an ORG statement. Also, an ORG statement specifying an even address must not be followed by a BSS or BES statement of an odd number of locations.

Processing the Contents of the SCRA

The core load builder analyzes the LOCAL, NOCAL, FILES, G2250, and EQUAT control records stored in the SCRA on disk, and builds tables for the respective control record types from the information specified. The information placed in these tables is used in later phases of the construction of the core load.

Processing Data Files

The core load builder uses the information in the FILES control records stored in the supervisor control record area (SCRA) to equate data files defined in the mainline program to data files stored on disk. The mainline program statements that define these files are the FORTRAN DEFINE FILE statement and the assembler FILE statement. During compilation or assembly, a define file table is built from the DEFINE FILE statements or FILE statements.

The core load builder compares a file number from a define file table entry with the file numbers specified in the FILES supervisor control records stored in the SCRA. If a match occurs, the name of the disk area associated with the file number on the FILES control record is found in LET or FLET, and the sector address of that disk area (including the logical drive code) is placed in the corresponding define file table entry. If the number in the define file table entry does not match any of the file numbers for FILES control records or if a name is not specified on the FILES control record, the core load builder assigns an area in working storage for the data file. The sector address of the data file, relative to the start of working storage, is placed in the define file table entry. This procedure is repeated for each define file table entry in the mainline program.

Conversion of the Mainline Program

The mainline program is converted from disk system format into disk core image format. The mainline is always converted before any of the other portions of the core load.

Incorporating Subroutines

Subroutines in general

All the subroutines called by other subroutines, by the mainline program and all subroutines specified as NOCALs are included in the core load, except for (1) the disk I/O subroutine, (2) any LOCAL subroutines specified, and (3) SOCAL subroutines employed.

EQUAT subroutines or symbolic names

Subroutines called by the core load that is being built can be replaced if indicated in EQUAT monitor control records stored in the SCRA. Symbolic names in assembler DSA statements are replaced by other symbolic names if so indicated in EQUAT control records.

FLIPR

The LOCAL/SOCAL flipper, FLIPR, is included in each core load in which LOCAL subroutines are specified or in which SOCAL subroutines are employed. FLIPR is entered by special LOCAL/SOCAL linkage through the transfer vector. FLIPR checks to determine if the required LOCAL or SOCAL is already in core. If not, FLIPR reads the required LOCAL or SOCAL into the LOCAL or SOCAL area in core. If the subroutine or subprogram is already in the LOCAL or SOCAL area of core, FLIPR transfers execution control to them.

When execution immediately follows the building of a core load, FLIPR reads a LOCAL or SOCAL, as it is called, from working storage into the LOCAL or SOCAL area of core. If the core image program was stored following the building of a core load, FLIPR reads a LOCAL or SOCAL, as it is called, from the user area or the fixed area (where it was stored following construction of the core load) into the LOCAL or SOCAL area of core.

CLB provision for LOCALs

LOCALs (load-on-call) are subroutines that you specify as overlays with LOCAL supervisor control records when error messages indicate that a core load is too large to fit into core.

If LOCALs are specified for use by a core load, the core load builder reserves an area in the core load as large as the largest LOCAL subroutine specified. LOCAL subroutines will be read by FLIPR into this area as required during execution. LOCAL subroutines are stored in working storage following any data files stored there. If the core load is executed immediately, each LOCAL subroutine is read as it is called from working storage into the LOCAL area by FLIPR. If the core load is stored in disk core image format before it is executed, LOCAL subroutines are stored following the core load, and will be read from the storage area, user area, or fixed area, during execution.

CLB provision for SOCALs

SOCALs (system-overlays-to-be-loaded-on-call) are groups of subroutines (by class, type, and subtype) that are made into overlays by the core load builder. SOCALs make it possible for FORTRAN core loads that are too large to fit into core to be loaded and executed. (SOCALs are not built for mainline programs written in assembler or RPG language.)

If, in constructing a core image program from a FORTRAN mainline program, the core load builder determines that the core load will not fit into core, SOCALs are created. An area as large as the largest SOCAL overlay (usually SOCAL 2) is reserved in the core load. SOCAL overlays will be read by flipper into this area as required during execution. The SOCAL overlays are placed in working storage following any data files and LOCALs stored there. If the core load is executed immediately, each SOCAL overlay is read, as it is called, from working storage into the SOCAL area by flipper. If the core load is stored in disk core image format before it is executed, SOCALs are stored following the core load and any LOCALs. SOCALs are then read from the storage area, user area, or fixed area, during execution.

The core load builder creates SOCAL overlays by subroutine class, type, and subtype (program types and subtypes are described under "Disk System Format" in Appendix I.) SOCAL overlays are numbered 1, 2, and 3. The classes of subroutines, their types and subtypes, that can be included in each SOCAL overlay are:

SOCAL overlay number includes	Subroutine class	Type	Sub-type
1	Arithmetic	3	2
	Function	4	8
2	Nondisk FORTRAN I/O and "Z" conversion subroutines	3	3
	"Z" device subroutines	5	3
3	Disk FORTRAN I/O	3	1

Each SOCAL overlay does not contain all the subroutines of the specified classes, types, and subtypes that are available in the monitor system library; only those subroutines required by the core load are included in the SOCAL. The names of the subroutines included in the SOCALs associated with a program are listed in a core map. A printout of the core map is obtained by placing an L in column 14 of an XEQ monitor control record (see "Reading a Core Map and File Map" in Chapter 6).

Two options are used by the core load builder in creating SOCAL overlays.

- *SOCAL Option 1.* An attempt is made to make the core load fit into core by using SOCAL overlays 1 and 2. This option reserves enough space in the core load for the largest of the 2 SOCALs (usually SOCAL 2) and approximately 115 additional words that are required for the special SOCAL linkage. SOCALs 1 and 2 are placed in working storage. When this option has been tried and the core load still does not fit into core, the second option is used.
- *SOCAL Option 2.* An attempt is made to make the core load fit into core by using SOCAL overlays 1, 2, and 3. This option reserves enough space in the core load for the largest of the 3 SOCALs (usually SOCAL 2) and approximately 120 additional words that are required for the special SOCAL linkage. If, after both SOCAL options have been tried, the core load still does not fit into core, an error message is printed.

If you specify as a LOCAL subroutine a subroutine that would usually be included in a SOCAL, the core load builder makes that subroutine a LOCAL and does not include it in the SOCAL in which it would ordinarily be placed. Further information is contained in "The Use of SOCALs" in Chapter 6.

Transfer Vector

The transfer vector (TV) is a table included in each core load that provides linkage to subroutines. This table is composed of:

- *CALL TV*—the transfer vector for subroutines referenced by CALL statements
- *LIBF TV*—the transfer vector for subroutines referenced by LIBF statements

Each CALL TV entry is a single word containing the absolute address of an entry point in a subroutine included in the core load that is referenced by a CALL statement. In the case of a subroutine referenced by a CALL statement but specified as a LOCAL, the CALL TV entry contains the address of the special LOCAL linkage instead of the subroutine entry point address. If SOCALs are required, the CALL TV entries for function subroutines contain the address of the special SOCAL linkage instead of the subroutine entry point address.

Each LIBF TV entry consists of 3 words. Word 1 is the link word in which the return address is stored; words 2 and 3 contain a branch to the subroutine entry point. In the case of a subroutine referenced by a LIBF statement but specified as a LOCAL, the LIBF TV entry contains a branch to the special LOCAL linkage instead of to the subroutine entry point address. The core load builder inserts the address in word 1 of the transfer vector entry (link word) into the entry point+2 of the associated LIBF subroutine. If SOCALs are required, the LIBF TV entry for a SOCAL subroutine contains a branch to a special entry in the LIBF TV for the SOCAL of which the subroutine is a part. This special entry provides the linkage to the desired SOCAL.

The core load builder can build a core load that references up to approximately 375 different LIBF *and* CALL entry points; 80 LIBFs plus 295 CALLs (the maximum number of LIBFs allowable is 83 due to the size of the LIBF TV). If the core load is built on an 1130 system with core size of 4K, the maximum number of different LIBF and CALL entry points is approximately 110.

See "Reading the Transfer Vector" in Chapter 6 for more information.

CORE IMAGE LOADER

The core image loader (CIL) has 2 functions:

- Transfer control between some monitor programs
- Transfer core loads into core for execution

On an entry to the skeleton supervisor at $EXIT, $DUMP, or $LINK, the core image loader is called and control transferred to it. The core image loader determines where the skeleton supervisor was entered and calls the appropriate monitor or mainline program.

$EXIT entry

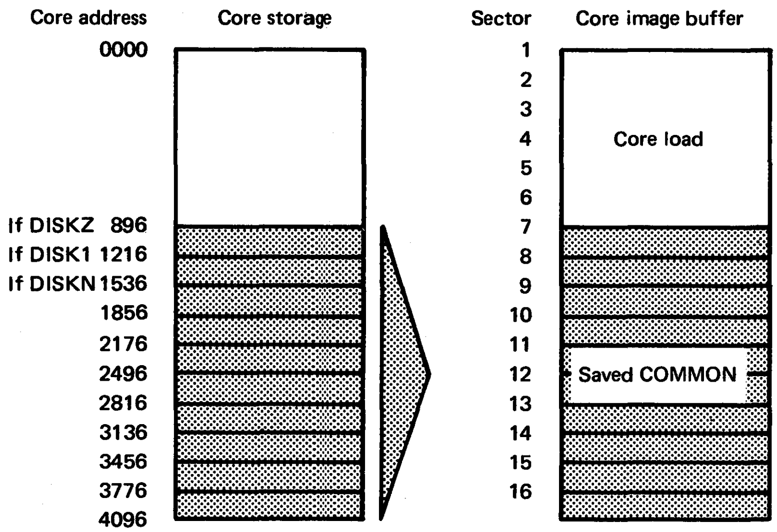
When the skeleton supervisor is entered at the $EXIT entry point, the core image loader calls the DISKZ I/O subroutine if DISKZ is not already in core. Then, the CIL calls and transfers control to the monitor control record analyzer to read monitor control records from the input stream.

$DUMP entry

When the skeleton supervisor is entered at the $DUMP entry point, the core image loader saves words 6 through 4095 (decimal) in the core image buffer. Then the CIL calls and transfers control to the Supervisor Core Dump Program. When the dump is complete, the dump program either restores core from the CIB and transfers control back to the core load in process or terminates execution with a CALL EXIT (see "Disk Resident Supervisor Programs" in this chapter).

$LINK entry

When an entry is made to the skeleton supervisor at the $LINK entry point, the core image loader saves the sector of core referred to as low COMMON. The sector saved depends on the disk I/O subroutine that is in core; locations (in decimal) 896 through 1215 if DISKZ, 1216 through 1535 if DISK1, or 1536 through 1855 if DISKN. Then the CIL determines from COMMA the lowest-addressed word of COMMON if any was defined by the core load just executed. Any COMMON in core below location 4096 (4K system) or 5056 in larger systems is saved in the CIB. The following illustrates the saving of COMMON.

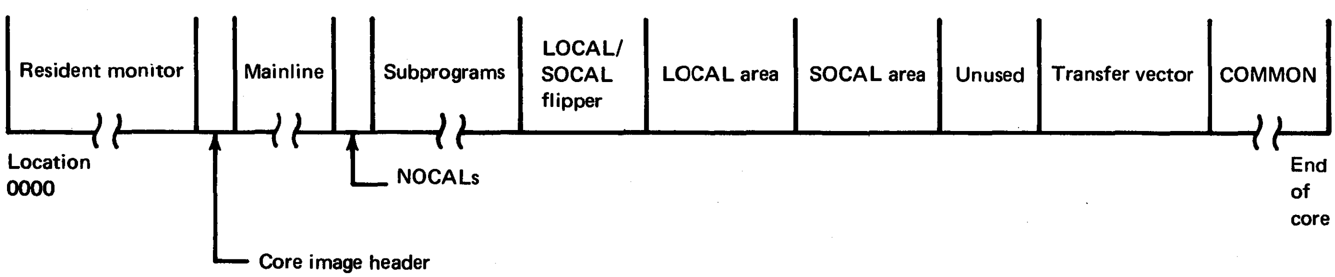


Next, the CIL determines from the LET or FLET entry for the program being called whether the program is in disk system format or in disk core image format.

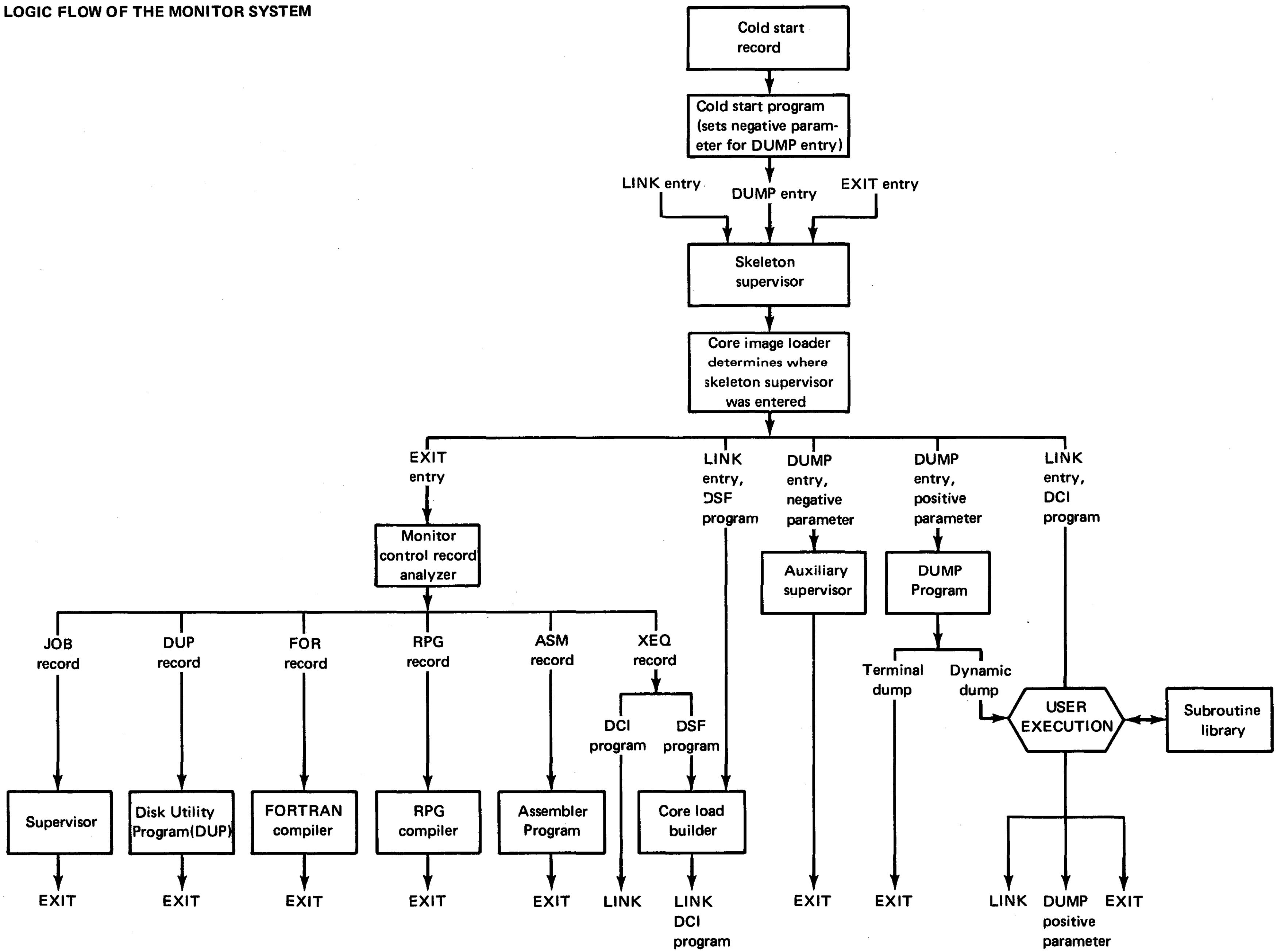
If the called program is in disk system format, the core load builder is called to construct a core load from the mainline program. After the core load is built, the core image loader is called to transfer the core load into core for execution.

If the called mainline program is stored in disk core image format, the disk I/O subroutine required by the core load is called, if it is not already in core. Any COMMON defined by the core load just executed and saved in the CIB is restored, and the called core load is transferred into core for execution.

The following illustration is the layout of a core load in core ready for execution.



LOGIC FLOW OF THE MONITOR SYSTEM



The monitor system library is a group of mainline programs and subroutines that performs the following functions for the monitor system:

- Input/output
- Data conversion
- Arithmetic functions
- Disk initialization
- Disk maintenance
- Paper tape utility

Appendix C is a listing of the names, types and subtypes, required subroutines, and ID fields for the programs and subroutines in the monitor system library.

Monitor system subroutines can be added to or deleted from the monitor system library. You add or delete them with Disk Utility Program (DUP) store and delete functions (see "*STORE" and "*DELETE" under "DUP Control Records" in Chapter 5). Each program in the IBM-supplied system deck used in an initial load is preceded by a DUP *STORE control record.

This chapter contains general information about:

- System library ISS subroutines
- System library utility subroutines
- System library mainline programs

Additional and more detailed information about the system library is contained in the publication *IBM 1130 Subroutine Library,* GC26-5929.

SYSTEM LIBRARY ISS SUBROUTINES

The interrupt service subroutines (ISS), in the monitor system library, manipulate the I/O devices that are part of the computer configuration. Each subroutine has a symbolic name that must be used when the subroutine is available, although only one for each I/O device can be selected for use in any one program (including subroutines). The following is a list of the devices available on the 1130 and the names of the ISS subroutines that are available for each device.

I/O device	I/O device subroutine
1442 Card Read Punch	CARDZ, CARD0, or CARD1
2501 Card Reader	READZ, READ0, or READ1
1442 Card Punch	PNCHZ, PNCH0, or PNCH1
Disk	DISKZ, DISK1, or DISKN
1132 Printer	PRNTZ, PRNT1, PRNT2
1403 Printer	PRNZ, or PRNT3
Console keyboard/printer	TYPEZ, or TYPE0
Console printer	WRTYZ, or WRTY0
1134/1055 Paper Tape Reader Punch	PAPTZ, PAPT1, PAPTN, or PAPTX
1627 Plotter	PLOT1, or PLOTX
1231 Optical Mark Page Reader	OMPR1
Synchronous Communications Adapter	SCAT1, SCAT2, or SCAT3

The last character or digit (Z, 0, 1, or N) of an ISS name indicates the general characteristics of the subroutine:

nameZ — The *nameZ* versions are designed for use in an error-free environment; preoperative error checking is not provided. FORTRAN and RPG use the nameZ versions of the ISS subroutines.

name0 — The *name0* versions are shorter and less complicated than the name1 or nameN versions. The name0 versions handle error conditions automatically.

name1 — Use the name1 versions rather than the name0 versions when you write an error exit. The name0 versions handle error conditions automatically.

nameN

The *nameN* versions are available to operate the 1134/1055 Paper Tape Reader/Punch simultaneously and to minimize extra disk revolutions when transferring more than 320 words to or from the disk. DISKN offers more options than DISK1. Depending on your computer configuration, it also offers simultaneous operation of any one of the following disk combinations.

- Up to five 2315 Disk Cartridges
- One 2315 Disk Cartridge (the 1131 CPU internal disk) and one disk in each of one or two 1316 Disk Packs
- One disk in each of two 1316 Disk Packs

Preoperative and postoperative errors that occur during the operations of the I/O device subroutines are included in Appendix B.

Extra space on a system cartridge can be gained by deleting the I/O device subroutines that are in the system library for devices that are not a part of your computer configuration. The following is a list of the subroutines that can be deleted for each device:

Device not in configuration	**I/O device subroutines that can be deleted**	**Disk blocks gained (hexadecimal)**
1442 Card Read Punch (input/output)	CARD0, CARD1, CARDZ	/4E
2501 Card Reader	READ0, READ1, READZ	/62
1442 Card Punch	PNCH0, PNCH1, PNCHZ	/22
1134/1055 Paper Tape Reader/Punch	PAPT1, PAPTN, PAPTX, PAPTZ, PAPEB, PAPPR, PAPHL,	/75
1132 Printer	PRNT1, PRNT2, PRTZ2, PRNTZ, DMPD1	/69
1403 Printer	PRNT3, PRNZ, EPTP3, CPPT3, HLPT3, PT3EB, PT3CP, PTHOL	/40
1627 Plotter	PLOT1, PLOTI, PLOTX, FCHRX, ECHRX, SCALF, SCALE, FGRID, EGRID, FCHAR, ECHAR, FPLOT, EPLOT, FRULE, ERULE, POINT, XYPLT	/B0
Synchronous Communications Adapter	SCAT1, SCAT2, SCAT3, PRNT2, PRTZ2, IOLOG, EBC48, HOL48, HXCV, STRTB, HOLCA	/FA
1231 Optical Mark Page Reader	OMPR1	/15
MTCA	MTCA0, MTCAZ, TSM41, TSTTY, FEB41	/9A

You should not delete subroutines that are called by subroutines left in the monitor system library (see Appendix C for lists of the subroutines called by each subroutine in the monitor system library).

The mainline programs required for devices not on the system that can be deleted from the system library are:

Device not in configuration	Mainline programs that can be deleted	Disk blocks gained (hexadecimal)
1134/1055 Paper Tape Reader/Punch	PTUTL	/0A
2310 Disk Storage or 2311 Disk Storage Drive	DLCIB, ID, COPY, DISC, IDENT	/9D

SYSTEM LIBRARY UTILITY SUBROUTINES

A group of subroutines that perform utility functions for the monitor system are included in the monitor system library. These subroutines are:

- SYSUP, disk communications area (DCOM) update subroutine, that you call in an assembler or FORTRAN program when you need to change disk cartridges or packs during execution of a core load. This subroutine updates DCOM on the master cartridge with the IDs and DCOM information from all satellite cartridges that are mounted on the system and that are specified in the special SYSUP calling sequence. Uses and calling sequences of SYSUP are discussed in Chapter 6.
- CALPR, call system print subroutine, that calls the print subroutines into core storage for printing information on the principal printer.
- FLIPR, LOCAL/SOCAL flipper overlay subroutine, that calls LOCAL (load-on-call) and SOCAL (system-load-on-call) subroutines into core storage during execution of a core load. LOCALs, SOCALs, and FLIPR are discussed under "Incorporating Subroutines" in Chapter 3 and in Chapter 6, "Programming Tips and Techniques".
- FSLEN, fetch phase IDs and fetch system subroutines, that performs 2 functions. The first function obtains system program phase ID headers from SLET as requested by monitor system programs. The second function calls system subroutines into core storage as needed.
- RDREC, Read *ID Record, that is called by the disk maintenance programs, discussed in this chapter, to read *ID control records.

Note. SYSUP is the only one of these utility subroutines that can be called by FORTRAN programs. The other subroutines are called as needed by monitor system programs or by assembler language programs.

SYSTEM LIBRARY MAINLINE PROGRAMS

The 1130 system library mainline programs provide for disk maintenance and paper tape utility functions. These programs (except the disk maintenance program, ADRWS) are called for execution with a monitor XEQ control record, and are described in the following sections of this chapter. These programs can be executed in a stacked job stream.

disk maintenance programs

The disk maintenance programs reinitialize cartridges, modify the contents of cartridges, and print information from cartridges. The disk maintenance programs are:

- IDENT that prints cartridge IDs
- DISC that reinitializes satellite cartridges
- DSLET that prints the contents of the system location equivalence table
- ID that changes cartridge IDs
- COPY that copies the contents of one cartridge onto another
- ADRWS that writes sector address in working storage
- DLCIB that deletes the core image buffer from a nonsystem cartridge
- MODIF that modifies the monitor system programs
- MODSF that modifies programs and subroutines in the system library
- DFCNV that converts 1130 FORTRAN and/or commercial subroutine package (1130-SE-25X) disk data files to disk files acceptable to 1130 RPG programs.

For execution, some disk maintenance programs require in addition to the monitor XEQ control record, special control records. The fields and uses of these special control records are described when required in the descriptions of these programs in this chapter.

PTUTL program

The Paper Tape Utility (PTUTL) Program accepts input from the paper tape reader or console keyboard and provides output to the console printer and/or the paper tape punch.

messages and halt codes

Messages printed by the disk maintenance programs are described in Appendix A. Halt codes displayed in the console ACCUMULATOR are described in Appendix B.

The following sections of this chapter describe the functions and calling sequences of the system library mainline programs.

IDENT

The Print Cartridge ID (IDENT) mainline program prints the cartridge ID and physical drive number of each disk cartridge that is mounted on the system and is ready, not just the cartridges that are specified in the current JOB monitor control record (see "Monitor Control Records" in Chapter 5). Invalid cartridge IDs, including negative numbers, are printed.

The IDENT program is called for execution with a monitor XEQ control record:

```
1   5    10   15   20   25   30   35   40   45   50
// XEQ IDENT
```

DISC

The Satellite Disk Initialization (DISC) mainline program requires at least 8K of core storage to run. DISC reinitializes from one to four satellite cartridges; all but the master cartridge. (All new cartridges must be initialized with the stand-alone DCIP utility program, see Chapter 9). On each cartridge being reinitialized, the DISC program:

- Tests disk sectors to determine which, if any, are defective, and fills in the defective cylinder table accordingly
- Writes a sector address on every sector, including defective sectors
- Establishes a file-protected area for the cartridge
- Places an ID on the cartridge
- Establishes a disk communications area, sector @DCOM, a location equivalence table (LET), and a core image buffer (CIB)

If an error occurs during testing, the cylinder on which the error occurred is retested. If the error occurs again, the address of the first sector on that cylinder is written in the defective cylinder table. The monitor system I/O subroutines operate with up to 3 defective cylinders on a cartridge. That is, 3 cylinders that contain one or more defective sectors. A cartridge cannot be initialized if cylinder 0 is defective, or if a sector address cannot be written on every sector.

A message and the program that prints it are written in sector @RIAD. The message is:

NONSYST. CART. ERROR

This message is printed when an attempt is made to cold start a nonsystem cartridge that is initialized with DISC.

The DISC program is called for execution with a monitor XEQ control record followed by an *ID control record:

```
1   5    10   15   20   25   30   35   40   45   50
// XEQ DISC
*IDFID1,TID1,FID2,TID2,....,FIDn,TIDn
```

***ID fields**

FID1 Through FIDn. Replace FID1 through FIDn with the current IDs on the satellite cartridges that are being reinitialized. This program overrides the cartridges that are specified in the current JOB monitor control record.

TID1 Through TIDn. Replace TID1 through TIDn with the new IDs to be placed on the satellite cartridges during initialization. A valid cartridge ID is a hexadecimal number from /0001 to /7FFF.

DSLET

The Dump System Location Equivalence Table (DSLET) mainline program prints the contents of SLET on the principal printer. Each SLET entry printed includes a symbolic name, phase ID, core address, word count, and disk sector address. Appendix E is a printout of a SLET dump.

The DSLET program is called for execution with a monitor XEQ control record:

```
// XEQ DSLET
```

ID

The Change Cartridge ID (ID) mainline program changes the ID on from one to four satellite cartridges. The ID program is called for execution with a monitor XEQ control record followed by an *ID control record:

```
// XEQ ID
*IDFID1,TID1,FID2,TID2,....,FIDn,TIDn
```

*ID fields

FID1 Through FIDn. Replace FID1 through FIDn with the IDs currently on the satellite cartridges that are to be changed. These IDs must be coded in the same logical order as those coded in the current JOB monitor control record.

TID1 Through TIDn. Replace TID1 through TIDn with new IDs that you want placed on the satellite cartridges. A valid cartridge ID is a hexadecimal number between /0001 and /7FFF.

COPY

The Disk Copy (COPY) mainline program requires at least 8K of core storage to run. COPY copies the contents from one cartridge (source) onto another (object cartridge). The defective cylinder data and cartridge ID are not copied. The copy code (word 5 of sector @IDAD) on the object cartridge is incremented to one greater than the copy code on the source cartridge. (The stand-alone DCIP program described in Chapter 9 provides a similar disk copy function.)

If a copy is made of a system cartridge from a system with a different configuration, the object cartridge must be reconfigured before a cold start can be performed (see Chapter 8 for information about reconfiguration).

The COPY program is called for execution with a monitor XEQ control record followed by an *ID control record:

```
// XEQ COPY
*IDFID1,TID1,FID2,TID2,....,FIDn,TIDn
```

*ID fields

FID1 Through FIDn. Replace FID1 through FIDn with the IDs of the cartridges that are being copied. When multiple copies are being made from a single cartridge, replace FID1 through FIDn with the same cartridge ID. This program overrides the cartridges that are specified on the current JOB monitor control record.

TID1 Through TIDn. Replace TID1 through TIDn with the IDs of the object cartridges.

ADRWS

The Write Sector Addresses in Working Storage (ADRWS) mainline program writes a sector address on every sector of working storage of a cartridge. This program is not executed with an XEQ monitor control record as the other disk maintenance mainline programs are. ADRWS is linked to from the Disk Utility Program (DUP) when a DWADR DUP control record is read from the job stream. (The DWADR control record is described under "DUP Control Records" in Chapter 5.)

DLCIB

The Delete Core Image Buffer (DLCIB) mainline program deletes the CIB from a nonsystem cartridge. The areas on the cartridge that followed the CIB before it was deleted are moved back 2 cylinders closer to cylinder 0. The new addresses of the areas moved are placed in DCOM on the master cartridge and in COMMA on the cartridge from which the CIB was deleted.

The DLCIB program is called for execution with a monitor XEQ control record followed by an *ID control record:

```
1   5    10   15   20   25   30   35   40   45   50
// XEQ DLCIB
*IDCART
```

*ID field

CART. Replace CART with the cartridge ID of the nonsystem cartridge from which the CIB is being deleted.

MODIF

The System Maintenance (MODIF) mainline program allows you to make updates to the monitor system programs and/or the system library. This program changes the word of the disk communications area (DCOM) that contains the version and modification level of the monitor system. (Information stored in the user area in disk system format can also be changed with the MODSF disk maintenance program described later in this chapter.)

A card deck or paper tape containing corrections to update the monitor system to the latest version and modification level is supplied by IBM. All modifications included must be run, even if an affected program has been deleted from the system, to update the version and modification level.

The MODIF program is called for execution with a monitor XEQ control record:

```
// XEQ MODIF
```

Note. A system program phase that contains reload table entries (references to other entries in SLET generated by the system loader during an initial load or reload operation) cannot be replaced with MODIF; a system reload must be used (see Chapter 8 for reload information). MODIF cannot be used if temporary mode is indicated in the current monitor JOB control record. A cold start procedure is recommended prior to a system reload if the reload precedes the execution of MODIF, as in a system modification update.

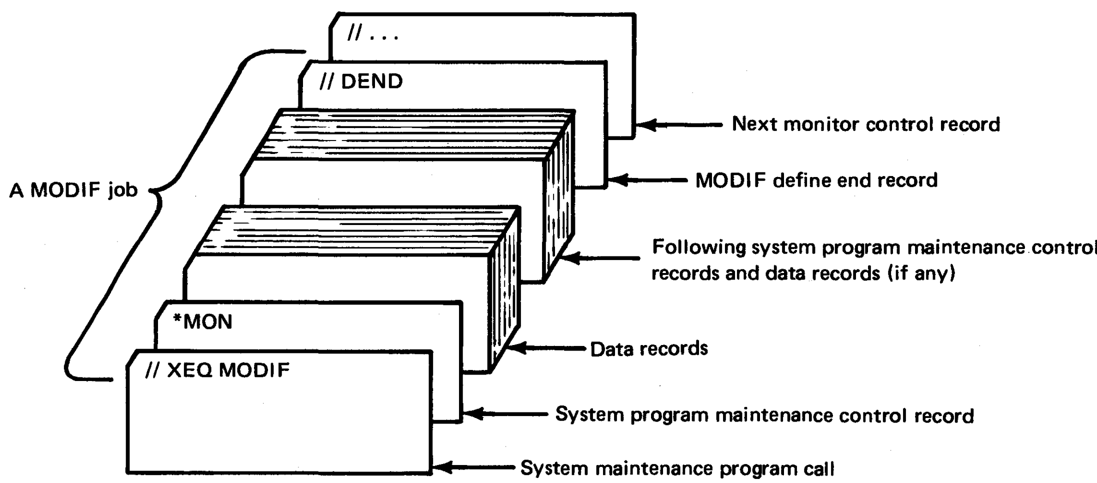
MODIF Patch Control and Data Records

The MODIF patch control records that can follow the monitor XEQ control record are:

- *MON that identifies a monitor program phase that is being modified
- *SUB that identifies a change to the system library
- // DEND that specifies the end of MODIF execution

*MON patch control record

The *MON patch control record, patch data records, and a // DEND control record modify monitor program phases. A typical input card deck for system program maintenance is:



Each program phase that is changed requires a *MON control record and patch data records that specify the changes. If MODIF determines from SLET that the FORTRAN compiler or the assembler has been deleted from the disk, any modifications that are included for these programs cannot be made; however, the version and modification levels for these programs are updated in DCOM.

*MON patch control record format

Card column	Contents	Explanation
1 through 4	*MON	These characters identify a patch to any of the monitor system programs and/or the system device subroutines.
5	Blank	
6 through 8	vmm	A hexadecimal number; *v* is the monitor version, and *mm* is the monitor modification level.
9	0 or G or R	*0* indicates system modification update. *G* indicates general temporary fix. *R* indicates restricted temporary fix.
10	Blank	
11 through 14	xxxx	The SLET ID (in hexadecimal) of the monitor program phase to which the patch is being made. 0000 indicates an absolute patch (see columns 28 through 31 and 33 through 36).
15	Blank	
16 through 19	nnnn	The numbers (in hexadecimal) of *patch data records* that follow this control record.
20	Blank	
21	B or H	This character identifies the format of the patch data records that follow. *B* indicates binary system format. *H* indicates hexadecimal patch format.
22	Blank	
23 through 26	pppp	A hexadecimal number that specifies the total number of patch control records to be processed. This field is required only on the first patch control record.
27	Blank	
28 through 31	dsss	A hexadecimal number; *d* is the disk drive code, and *sss* is the sector address of the program being patched. Use this field only when columns 11 through 14 contain 0000.

Card column	Contents	Explanation
32	Blank	
33 through 36	cccc	A hexadecimal number that specifies the core address of the sector specified in columns 28 through 31. Use this field only when columns 11 through 14 contain 0000.
37 through 80	Not used	

additional field information

**MON.* The programs that can be patched are: the FORTRAN compiler, RPG compiler, COBOL compiler (program product), assembler, Disk Utility Program, supervisor, core load builder, core image loader, and the system device subroutines. Modifications to the system device subroutines must be made with a *MON patch, not a *SUB, *DELETE, and *STORE patch.

0 or G or R. A system modification update (0) can be made only on a system of one level lower than the level indicated in columns 6 through 8. A general temporary fix (G) can be made only on a system of the same or one higher level than the level indicated in columns 6 through 8. A general temporary fix does not change the level of the system.

A restricted fix (R) can be made only on a system of the same level as the level indicated in columns 6 through 8.

pppp. A MODIF job can modify more than one system program and can modify both system programs *and* the system library.

In the latter case, the specified count in columns 23 through 26 must include the *SUB patch control record. The // DEND control record is not included in this count.

cccc. Core addresses can be obtained from the microfiche listings.

patch data records

Patch data records are in either hexadecimal patch format or binary system format. These data records specify the beginning address of the patch, and the new data for the patch. Patch data records cannot contain CALLs or LIBFs, and the relocation indicators will not be used.

hexadecimal patch data record format

Card column	Contents	Explanation
1 through 4	aaaa	The beginning core address (in hexadecimal) of the patch. Each patch data record must contain the core address.
5	Blank	
6 through 9, 11 through 14, 16 through 19, . . . 66 through 69		Each 4-column field is one word of patch data (in hexadecimal). Up to 13 words of patch data can be included in one data record. A blank must separate each word of data.
70 through 72	Blank	
73 through 80	Not used	

Hexadecimal patch records can contain ID/sequence numbers in columns 73 through 80. Zeros must be punched; leading blanks are not assumed.

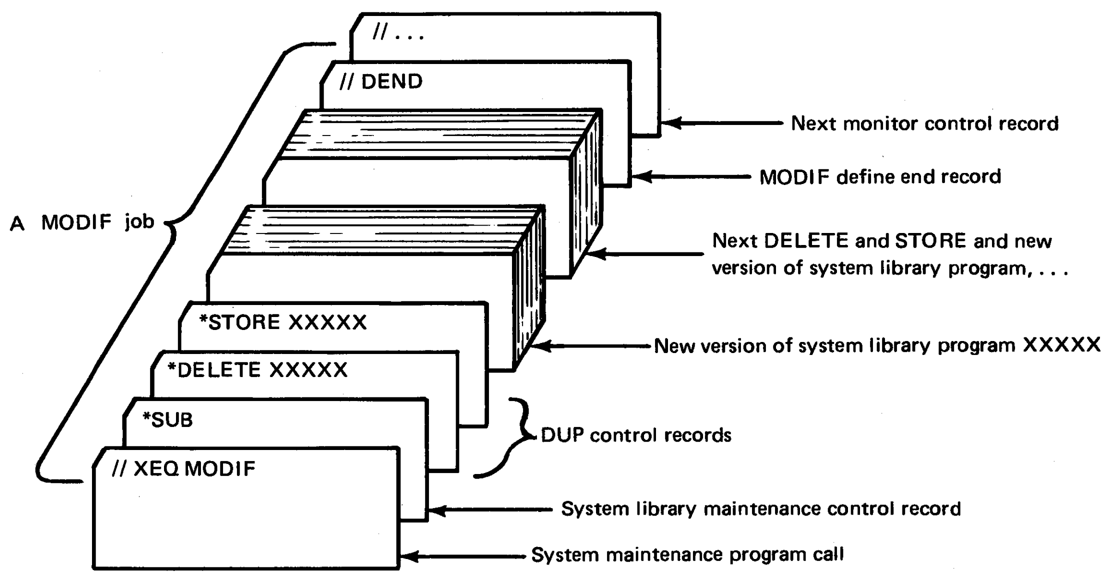
binary system patch data record format

Word	Contents
1	Location
2	Checksum
3	Type code (first 8 bits) 00001010
4 through 9	Relocation indicators
10 through 54	Data words 1 through 45
55 through 60	ID and sequence number or blanks

Note: Checksum verification is not made if word 2 is blank.

*SUB patch control record

The *SUB patch control record, DUP *DELETE and *STORE functions, new versions of system library programs and subroutines, and a // DEND control record are used to modify the system library. A typical input card deck for system library maintenance is:



Only one *SUB control record is used in a MODIF job; however, any number of deletes and stores can be included after a *SUB control record. When a MODIF job is used to modify system programs *and* the system library, the *SUB control record must be the last patch control record before // DEND in the MODIF job. The *SUB control record is also included in the count of MODIF patch control records coded in columns 23 through 26 of the *MON control record.

*SUB patch control record format

Card column	Contents	Explanation
1 through 4	*SUB	These characters identify a patch to the monitor system library.
5	Blank	
6 through 8	vmm	A hexadecimal number; *v* is the monitor version, and *mm* is the monitor modification level.
9	0 or G or R	*0* indicates system modification update. *G* indicates general temporary fix. *R* indicates restricted temporary fix.
10 through 15	Blanks	
16 through 19	nnnn	The number (in hexadecimal) of delete and store control records that follow this control record.
20 through 80	Not used	

additional field information

0 or G or R. A system modification update (0) can be made only on a system of one level lower than the level indicated in columns 6 through 8.

A general temporary fix (G) can be made only on a system of the same or one higher level than the level indicated in columns 6 through 8. A general temporary fix does not change the level of the system.

A restricted fix (R) can be made only on a system of the same level as the level indicated in columns 6 through 8.

// DEND patch control record

All MODIF jobs must end with a define end control record (// DEND). This record terminates MODIF execution and passes control to the supervisor.

// DEND patch control record format

Card column	Contents	Explanation
1 through 7	//bDEND	b indicates blank.
8 through 80	Not used	

MODIF Example

This example illustrates how to change an instruction in the 1134/1055 system subroutine. The following data is used to make the change:

- The SLET phase ID of the subroutine is /0091.
- Hexadecimal patch format is used.
- The instruction address (from an assembly listing) is /0023.
- The instruction is /0000.
- The instruction is to be changed to /7002.
- The new modification level is 8.
- One patch data record is required.
- Only one patch control record (// DEND) follows the *MON control record.

The coding sequence for making this change is:

```
1   5    10   15   20   25   30   35   40   45   50
// JOB
// XEQ MODIF
*MON 2080 0091 0001 H 0001
0023 7002
// DEND
```

When execution is complete, the following messages are printed on the principal printer:

MODIF EXECUTION 0207	The execution of MODIF is started on Disk Monitor System Version 2, level 7.
MODIF TERMINATION 0208	The patch is installed; the new level is 8.

MODSF

The Library Maintenance (MODSF) mainline program allows you to update programs that are stored in the user area in disk system format. (Monitor system programs are modified or replaced with the MODIF program discussed in the previous section of this chapter.)

MODSF updates a program by replacing existing code and/or inserting additional code at the end of the program. Existing code is replaced in the program as it resides in the user area. The existing code of several programs can be updated in one MODSF job, but code can only be added to the last program included in the MODSF job. When additional code is added to a program, MODSF moves the program into working storage before inserting the new code. The modified program is still in working storage when MODSF execution is finished and can be transferred back to the user area with DUP *DELETE and *STORE functions.

On the basis of where the addresses you specify are in the program being modified, MODSF determines whether a particular update is a replacement or an addition of code. A maximum of 31 words can be updated in one MODSF job.

The MODSF program is called for execution with a monitor XEQ control record:

```
1   5    10   15   20   25   30   35   40   45   50
// XEQ MODSF
```

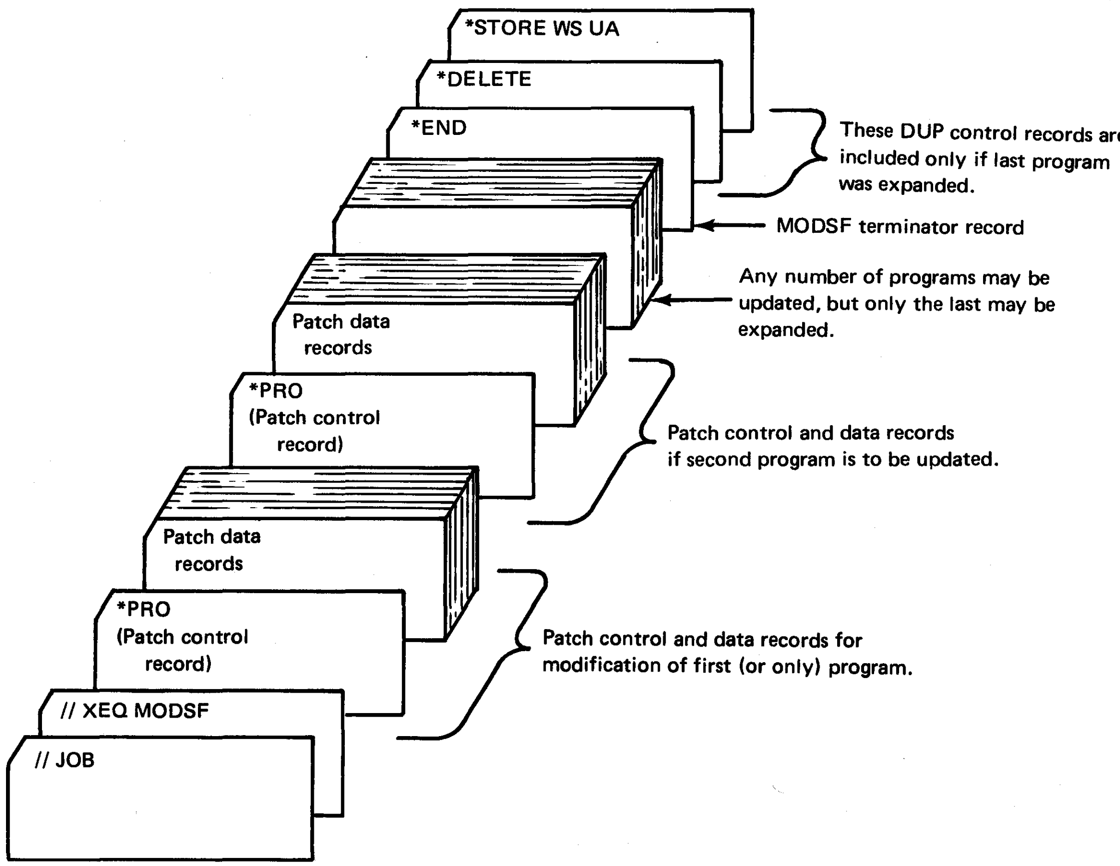
MODSF Patch Control and Data Records

The MODSF patch control records that can follow the monitor XEQ control record are:

- *PRO that identifies the program that is being modified.
- *END that specifies the end of MODSF execution.

*PRO patch control record

The *PRO patch control record, patch data records, and an *END control record are used to modify programs and subroutines stored in the user area. A typical input card deck for library program maintenance is:



Each program or subroutine that is being modified requires a *PRO control record and patch data records that specify the changes being made.

*PRO patch control record format

Card column	Contents	Explanation
1 through 4	*PRO	These characters identify a MODSF patch control record.
5	Blank	
6 through 8	vmm	A hexadecimal number; *v* is the current monitor version, and *mm* is the current monitor modification level.
9	Blank	
10 through 14	pname	The name of the DSF program being updated. (If the program has secondary entry points, this must be the name of the primary entry point.)
15	Blank	
16 through 19	nnnn	The number (in hexadecimal) of *patch data records* that follow this control record.
20	Blank	
21	m	Indicates addressing mode, where *m* is: *P* for program-address mode, or *D* for disk-displacement mode.
22	Blank	
23 through 26	xxxx	Cartridge ID of the cartridge on which the program being modified is stored. (A cartridge ID is not necessary if the program is stored on the master cartridge.)
27, 37, 47, 57	Blanks	
28 through 31 38 through 41 48 through 51 58 through 61	aaaa aaaa aaaa aaaa	Each of these optional fields specifies an address (in hexadecimal) at which the current content of the program is compared with the values specified beginning in column 33.
32, 42, 52, 62	Blanks	
33 through 36 43 through 46 53 through 56 63 through 66	vvvv vvvv vvvv vvvv	The value (in hexadecimal) that is being compared with the program content at the addresses specified beginning in column 28. These optional fields are used when the aaaa fields are used.
67 through 72	Reserved	
73 through 80	Not used	

additional field information

m. Addresses at which modifications are being made to the program are expressed as either P for P-mode (program-address) or D for D-mode (disk-displacement). In P-mode, each address represents a relative address within the program such as is printed on the left of an assembly listing.

In D-mode, each address represents a relative location on a disk; a location that the number of words indicated by the displacement beyond word 0 of the DSF program header. Each D-mode address corresponds to an address on a DUP *DUMP of the program to the printer.

Note. D-mode should be used if the program or subroutine being updated contains a backward origin. If P-mode is used when a program contains a backward origin, the results of MODSF execution are unpredictable.

aaaa . . . and vvvv . . . These optional fields allow you to verify whether or not a specific update has been made by checking the contents of the program at specified addresses (aaaa . . .) with specified values (vvvv . . .). If the contents of the words checked are not exactly as specified, the MODSF job is terminated. The addresses (aaaa . . .) are interpreted by MODSF as P-mode or D-mode according to the addressing mode specified in column 21 of this control record.

Note. The second word of a LIBF or CALL cannot be verified.

patch data records

Code can be replaced or added in either P-mode or D-mode. You specify the addressing mode in column 21 of the *PRO control record. The patch data records for MODSF are in either P-mode or D-mode format. For the patch data records, choose the format according to the addressing mode you specify in the *PRO control record.

In P-mode, you can update any word in a program, including the relocation code for that word. (You cannot update the program header or any data header in the program text because these are not a part of the program.) You can add words to the end of a program; a relocation code must be specified for each new word. The program length and the disk block count in the program header are automatically updated by MODSF when an addition is made.

Because the object code of a LIBF occupies 2 words as stored on disk but only one word in a subsequent core load of the program, you can only replace a LIBF with another LIBF.

P-mode patch data record format

Card column	Contents	Explanation
1 through 4	aaaa	The address (in hexadecimal) in the program of the first word being changed.
5	Blank	
6	r	Relocation code of the first word being changed; enter: *A* for an absolute expression or the second word of an LIBF or a CALL (relocation code 0), *R* for a relocatable expression or the second word of a DSA statement (relocation code 1), *L* for the first word of an LIBF (relocation code 2)—an update with an *L* relocation code *must* be immediately followed (on the same patch data record) by a second update word with an *A* relocation code, *C* for the first word of a CALL or DSA statement (relocation code 3).
7	Blank	
8 through 11	xxxx	The value (in hexadecimal) that is being inserted in the first location.
12	Blank	
13	r	Relocation code of the second word being changed (see column 6).
14	Blank	
15 through 18 . . . 64 through 67	xxxx	The value that is being inserted in the next location. As many as 9 consecutive words can be updated with one data record. A relocation code must precede each value specified, and a blank must separate a relocation code from a value.
68 through 72	Reserved	
73 through 80	Not used	

In D-mode, you can change any word in a program. You can also change the program header or any data headers in the program text. You must update the program length and the disk block count in the program header when you add code to the end of a program. You must also modify any data headers and indicator data words affected by your changes or additions. Be careful to change only the required information in headers.

D-mode data control record format

Card column	Contents	Explanation
1 through 4	aaaa	Disk displacement (in hexadecimal) of the first word being changed with this data record.
5	Blank	
6 through 9	xxxx	The value (in hexadecimal) that is being inserted in the location specified by columns 1 through 4.
10	Blank	
11 through 14 . . . 66 through 69	xxxx	The next value that is being inserted in the next location. As many as 13 consecutive words can be updated with one data control record. Each value specified must be separated from the next with a blank.
70 through 72	Reserved	
73 through 80	Not used	

*END patch control record

All MODSF jobs must end with a MODSF terminator record (*END). This record terminates MODSF execution and passes control the the supervisor.

*END control record format

Card column	Contents	Explanation
1 through 4	*END	These characters signify the end of input for MODSF.
5 through 72	Reserved	
73 through 80	Not used	

MODSF Example

This example illustrates how to change three instructions to NOP instructions. The following data is used to make the changes:

- The name of the program is FADD.
- The instruction addresses (from an assembly listing) are 001B, 001C, and 001D (hexadecimal).
- The values that are compared with the contents at these locations are C900, D839, and 18D0, respectively.
- The instructions are all changed to 1000.
- The addressing mode is P.
- One P-mode patch data record is used.
- The modification level is 9.

The coding sequence for making these changes is:

```
1   5    10   15   20   25   30   35   40   45   50   55   60   65      72
// JOB
// XEQ MODSF
*PRO 2Ø9 FADD ØØØ1 P      ØØ1B C9ØØ ØØ1C D839 ØØ1D 18DØ
ØØ1B A 1ØØØ A 1ØØØ A 1ØØØ
*END
```

When execution is complete, the following messages are printed on the principal printer:

MODIFICATIONS MADE	The changes are made and did not expand the program.
SUCCESSFUL COMPLETION	This message is printed when the *END record is read and the program is not expanded.

DFCNV

The Disk Data File Conversion (DFCNV) mainline program converts 1130 FORTRAN and/or commercial subroutine package (1130-SE-25X) disk data files to disk files acceptable to 1130 RPG. The program operates in a minimum 8K core DM2 system and uses DISK1 and the system device subroutines for the principal input device and principal printer.

DFCNV accepts all FORTRAN and commercial subroutine package (CSP) disk data formats for conversion to acceptable RPG disk data format. FORTRAN or CSP input to DFCNV can be a disk file created with or without 2-word integers, or a deck of cards produced by a DUP *DUMPDATA operation.

Prior to executing DFCNV, use a DUP *STOREDATA or *DFILE operation to reserve an output file in the user or fixed area and to enter its file name in LET or FLET. The DFCNV output file can be defined on the same disk as the input file or on a cartridge residing on another drive. DFCNV converts one input file to one output file; subsequent DFCNV program executions must be performed to convert more than one file.

RPG programs can process the converted files sequentially or randomly, but not as indexed sequential access method (ISAM) files.

Note. The disk file protection indicators $FPAD-$FPAD+4 in COMMA are modified during the conversion portion of DFCNV. These modified indicators must be restored prior to further monitor processing if unforseen problems, such as accidentally pressing IMM STOP, cause abnormal ending of DFCNV. Normally, these indicators are restored by DFCNV after a successful file conversion.

The DFCNV program is called for execution with a monitor XEQ control record:

```
// XEQ DFCNV       1
```

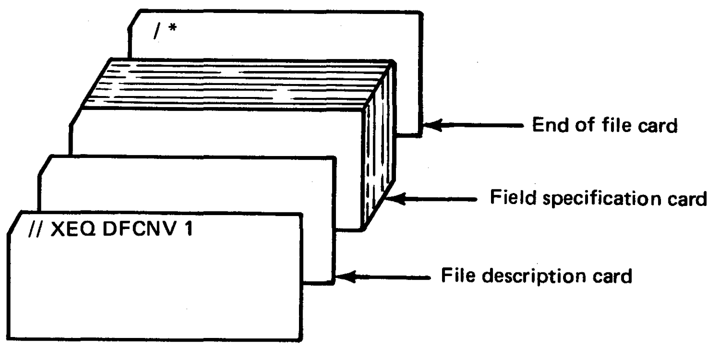
DFCNV Control Records

Three types of control records are required by the conversion program:

- File description
- Field specification
- End-of-file

file description control record

A file description control record is required and must immediately follow the XEQ control record. Only one file description record is used. A typical input card deck for the conversion program is:



The file description control record contains the following information.

file description control record format

Card column	Contents	Explanation
1 through 5	Name	The file name (left-justified) of the file whose data is being converted. This field is ignored if card input is specified in column 31.
6	Blank	
7 through 11	RPG name	The file name (left-justified) of the file where the RPG data is to be placed.
12	Blank	
13 through 17	Number of input records	A right-justified decimal number with leading zeros or blanks and in the range 1 through 32767.
18	Blank	
19 through 21	Input-file record size, in words	A right-justified decimal number with leading zeros or blanks and in the range 1 through 320.
22	Blank	
23 through 25	RPG file record size, in characters	A right-justified decimal number with leading zeros or blanks and in the range 1 through 640.
26	Blank	
27	S or E	*S* indicates standard precision. *E* indicates extended precision.
28	Blank	
29	1 or blank	*1* indicates one-word integers are used.
30	Blank	
31	C or blank	*C* indicates input from cards. Blank indicates that input is from disk.
32	Blank	
33	W or blank	*W* indicates that an object time warning message is to be printed if a real number (see "R-Field Type" in Appendix J) is out of range upon conversion. Blank indicates that the object time warning message is not printed.
34 through 71	Blanks	
72	D	This character identifies this record as a file description record.
73 through 80	Not used	

additional field information

Name. Use the exact name of the FORTRAN or CSP file that is being converted.

RPG name. The RPG file name cannot contain any special characters, although the input file name can contain the character $. DFCNV does not check the RPG file name for $.

computing file sizes

Both the input and RPG file sizes are calculated from the information that you specify in the file description control record. These computed sizes are checked against their corresponding LET or FLET entries for correct size. The following formulas are used to calculate the input and output file sizes.

1. Compute the number of words (*L*) in a record:

$$L = \frac{C}{2}$$

   where

   C is the record size in characters. Round the answer to the next higher number if the answer has a remainder.

2. Compute the number of records (*N*) that can be contained in one sector:

$$N = \frac{320}{L}$$

   where

   L is the length in words of each record computed in Step 1, and *320* is the number of words in a sector. Disregard the remainder, if any.

3. Compute the input file size (*I*) in sectors:

$$I = \frac{R}{N}$$

   where

   R is the number of records in the file, and *N* is the number of records per sector computed in Step 2. Round the answer to the next higher number if the answer has a remainder.

4. Compute the output file size (*O*) in sectors:

$$O = \frac{R+1}{N}$$

   where

   R is the number of records in the file, and *N* is the number of records per sector computed in Step 2. Round the answer to the next higher number if the answer has a remainder.

These are the same formulas that you use to calculate record and file sizes of sequentially organized files, see "File Processing" in Chapter 6.

field specification control record

The second required control record, field specification, describes the RPG fields for the converted data. Descriptions and examples of each field type supported by the program are in Appendix J.

Caution: DFCNV does not check data format; therefore, you must know *in detail* the format of the fields of your FORTRAN or CSP input file.

You can use as many *complete* field specifications on a field specification control record as can be placed in columns 1 through 71. Column 72 of each record must contain an S. Field specifications must be placed on the control records in the same order as the corresponding fields of the input record. Each field specification must be separated from the next with a comma. Blanks embedded in specifications or blanks between specifications are not allowed. The following is an example of a field specification control record:

```
1   5    10   15   20   25   30   35   40   45   50   55   60   65     72
1-R3.Ø,5-I4.1,12-R7.5,21-B8.2,3Ø-I8.2                                   S
```

Selected field conversion can be done by using the X-field type. See Appendix J for a description of this field type. Data can be rearranged and field size can be modified with the *m* term of field types. When data is rearranged or fields are expanded, you must prevent data overlay in the converted field.

repeat specification option

Identical fields that are sequentially repeated can be specified with only one field specification for any field type except the X-field type. You specify the repeat option by immediately following the specification being repeated with the character R and the total number of identical fields. Each repeat field begins in the first vacant output column after the previous field; that is, columns are not skipped when the repeat specification is used.

For example, the following field specification describes three integer fields, the first beginning in column 15 of the RPG record. Each field is packed and is five characters long with 2 places to the right of the decimal point:

15–I5.2(P)R3

The 3 resulting output fields start in the eighth word of the output record as:

Word:	8	9	10	11	12
Contents:	XXX0	0FXY	Y00F	ZZZ0	0F40

where

XXX, YYY, and ZZZ represent the three integer fields.

optional control record

When any F-field type conversions are specified on the field specification control record, an optional control record is required. This control record must contain the 40 character translation table for CSP A3 format and the character A in column 72. This control record immediately precedes the first field specification control record that specifies F-field type conversion. Only one conversion table is allowed per file; if more than one is included in the control records, the additional tables are ignored. The conversion table must correspond to the original table used to convert to CSP A3 format.

end-of-file control record

The third required control record for DFCNV is the end-of-file control record. All other DFCNV control records must precede the end-of-file (/*) control record.

DFCNV Example

This example illustrates how to convert the FORTRAN file named FORFL to an RPG file named RPGFL. The FORTRAN file contains 1,000 records, each 10 words long. The file is standard precision with one-word integers. One such FORTRAN record is as follows:

Word:	1	2	3	4	5	6	7
Content:	3A7E	D64B	40D5	D540	D4C1	BC00	0080
Word:	8	9	10				
Content:	03C8	C000	0083				

The RPG file consists of records 40 characters long. The coding for converting the FORTRAN file is:

```
// JOB
// XEQ DFCNV      1
FORFL RPGFL Ø1ØØØ Ø1Ø Ø4Ø S 1                                          D
1-R3.Ø,5-I4.1,12-R7.5,21-B8.2,3Ø-I8.2                                  S
/*
```

After conversion, the RPG record that corresponds to the previous FORTRAN record is stored on disk as:

Word:	1	2	3	4	5	6	7	8
Content:	F0F0	D440	F9F6	F8F0	4040	40F0	F0F5	F3F1
Word:	9	10	11	12	13	14	15	16
Content:	F2D5	4040	D4C1	D540	40D5	D64B	40F0	F1F4
Word:	17	18	19	20				
Content:	F9F7	F4F0	F040	4040				

PTUTL

The Paper Tape Utility (PTUTL) mainline program accepts input from the keyboard or the 1134 Paper Tape Reader and provides output on the console printer and/or the 1055 Paper Tape Punch. You can make changes and/or additions to FORTRAN and assembler language source records and monitor control records with PTUTL.

The PTUTL program is called for execution with a monitor XEQ control record:

```
// XEQ PTUTL
```

The PTUTL program is also available as an IBM-supplied stand-alone program on tape BP17. The operating procedure for both PTUTL programs is in Figure 9-12, Chapter 9. An example of using this program is also included under "Stand-alone Paper Tape Utility Program (PTUTL)" in Chapter 9.

Chapter 5. Control Records

You use control records to specify operations performed by the Disk Monitor 2 System. The use of these control records provides for stacked jobs with a minimum of operator intervention. The order of control records, source statements, and data in stacked jobs is described under "Stacked Input Arrangement" in Chapter 6.

The control records in this chapter are grouped according to the monitor program that they are associated with. These groups are:

- Monitor control records
- Supervisor control records
- DUP control records
- Assembler control records
- FORTRAN control records
- RPG control records

Each section of this chapter consists of a general function description, the order in which the control records are placed in the input stream, general coding considerations, and a description of each control record.

Other less frequently used control records are included in Chapter 4, "Monitor System Library." The control records described in Chapter 4 apply to specific, infrequently performed procedures.

MONITOR CONTROL RECORDS

functions

The monitor control records described in this section define control and load functions that are performed by the monitor system. These functions are:

- Initializing jobs
- Loading the assembler, the language compilers, or the Disk Utility Program into core for execution
- Starting the execution of your programs
- Printing comments during monitor system operations
- Changing print devices during monitor system operations

The JOB monitor control record defines and initializes the beginning of jobs. Other monitor control records are placed behind the JOB control record to specify the operations to be performed during a job. A detailed description of the order of control records, program statements, and data files in the input stream is in Chapter 6 under "Stacked Input Arrangement."

coding

Information must be coded in the indicated card columns in monitor control record formats. Columns 1 and 2 always contain slashes (//). The character ƀ and *reserved* card columns indicate that the columns must be blank. You can replace card columns shown as *not used* with comments.

// JOB

general function

A JOB monitor control record defines the start of a new job. This control record causes the supervisor to initialize a job, which includes:

- The initialization of parameters in the core communications area (COMMA) and in sector @DCOM
- The setting of the temporary mode indicator if the job is executed in temporary mode
- The definition of the cartridges to be used during the current job
- The definition of the cartridge that contains the core image buffer used for the current job
- The definition of the cartridge that contains working storage used during the current job
- The definition of the cartridge that contains the unformatted I/O disk buffer area for use during the current FORTRAN job
- The definition of a new heading printed on each page printed by the principal print device
- The reading of EQUAT supervisor control records into the supervisor control record area (SCRA)

format

Card column	Contents	Explanation
1 through 6	//ƀJOB	
7	Reserved	
8	Temporary mode indicator	*T* or blank. A *T* indicates that temporary mode is desired for this job.
9 through 10	Reserved	
11 through 14	First ID	This is the ID of the master cartridge (logical drive 0).
15	Reserved	
16 through 19	Second ID	This is the ID of the cartridge on logical drive 1.
20	Reserved	
21 through 24	Third ID	This is the ID of the cartridge on logical drive 2.
25	Reserved	
26 through 29	Fourth ID	This is the ID of the cartridge on logical drive 3.
30	Reserved	
31 through 34	Fifth ID	This is the ID of the cartridge on logical drive 4.
35	Reserved	
36 through 39	CIB ID	This is the ID of the cartridge containing the CIB to be used during this job.

Card column	Contents	Explanation
40	Reserved	
41 through 44	Working storage ID	This is the ID of the cartridge containing the working storage to be used by the monitor during this job. See *FILES, for details on working storage for your programs.
45	Reserved	
46 through 49	Unformatted disk I/O ID	This is the ID of the cartridge containing the unformatted disk I/O area to be used during this job.
50	Reserved	
51 through 58	Date, name, etc.	This information is printed at the top of every page of the listing on the principal print device during this job.
59	Not used	
60 and 61	EQUAT record count	This number specifies how many EQUAT records follow this JOB record.
62 through 80	Not used	

additional field information

Temporary Mode Indicator. A T in column 8 causes all programs and/or data files stored by DUP in the user area during the current job to be deleted from the user area when the next // JOB control record is read. Temporary mode places restrictions on some of the DUP operations as shown in the followng chart:

DUP operations	Restrictions
DUMP	None
DUMPDATA, DUMPDATAƀE	None
STORE	None
STORECI	To UA only
STOREDATA, STOREDATAE	To UA and WS only
STOREDATACI	To UA only
STOREMOD	Not allowed
DUMPLET	None
DUMPFLET	None
DWADR	Not allowed
DELETE	Not allowed
DEFINE FIXED AREA	Not allowed
DEFINE VOID ASSEMBLER	Not allowed
DEFINE VOID FORTRAN	Not allowed
DEFINE VOID RPG	Not allowed
DEFINE VOID COBOL	Not allowed
DFILE	To UA only
MACRO UPDATE	Not allowed

First ID through Fifth ID. These IDs define the cartridges that are used during the current job. These cartridges can be mounted on the physical disk drives in any order; the order of the IDs on the JOB control record specifies the logical assignments for the cartridges. The first through the fifth IDs correspond to logical drives 0 through 4, and must be specified consecutively. When 3 drives are being used, only the first through the third IDs are specified.

The cartridge-related entries of the core communications area (COMMA) and sector @DCOM are filled according to the logical order specified by the JOB control record. The first ID can be left blank, in which case the master cartridge for the last JOB will also be the master cartridge for the current JOB. A cartridge ID is not required when only one cartridge is used during the current JOB. In this case, the master cartridge from the last JOB or that was specified during a cold start is used.

The first cartridge ID can be used to define a system cartridge that is different from the one currently being used as logical drive 0. The specified cartridge must be the same monitor modification level as the one it replaces.

CIB ID. This is the ID of the cartridge that contains the core image buffer to be used during the current job. The CIB ID is optional. If this ID is omitted, the CIB on the master cartridge is assumed by the system. If the CIB on the specified cartridge has been deleted, the CIB on the master cartridge is assumed for the current job. Core image programs are built faster when the specified CIB is on a cartridge other than the master cartridge.

Working Storage ID. This field specifies the cartridge that contains the working storage that is used during the current job. The working storage ID is optional. If this ID is omitted, working storage on the master cartridge is used except when otherwise specified on DUP control records (see "DUP Control Records" in this chapter).

Core image programs are built faster when the specified working storage is on a cartridge other than the master cartridge. They can be built even faster when the IBM system area, the CIB, and working storage are all on separate cartridges.

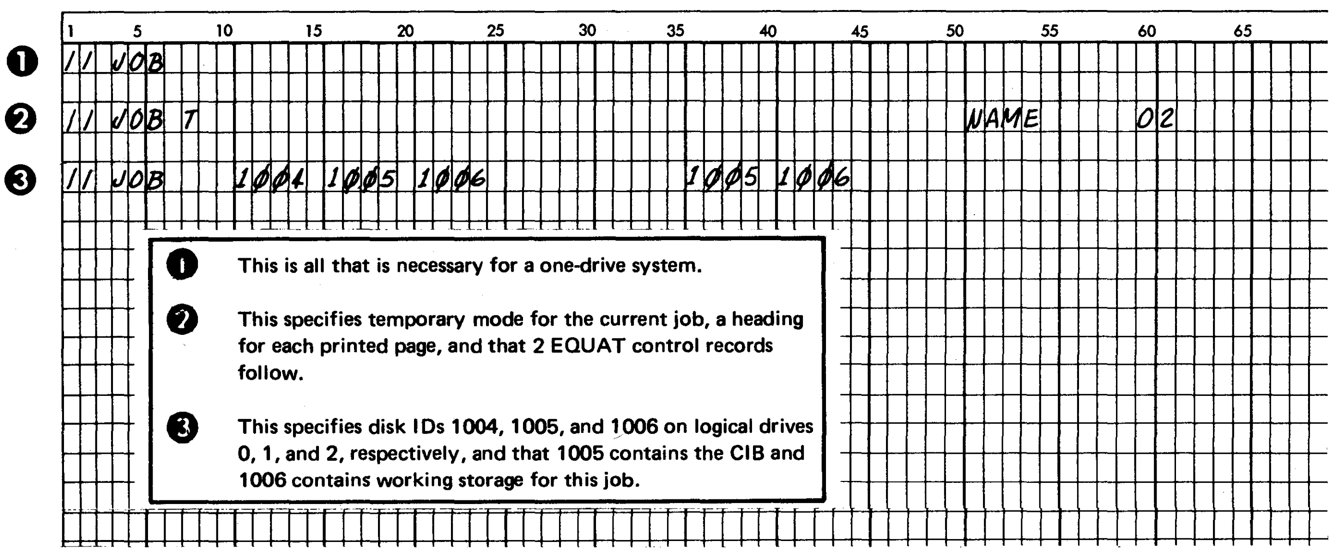
Programs are assembled or compiled faster when system working storage is on another cartridge. (See "*FILES" under "Supervisor Control Records" in this chapter for specifying working storage for use by your programs.)

Unformatted Disk I/O ID. This field specifies the cartridge that contains the unformatted I/O disk buffer area to be used during the current job. The unformatted disk I/O ID is specified when only unformatted I/O (data file named $$$$$) is used during execution of a FORTRAN program. (See "Initializing $$$$$ Data Files for Use With FORTRAN Unformatted I/O" in Chapter 6 for more information.)

Date, Name, Etc. This information is printed on the top of each page printed by monitor system programs, except RPG. This causes a skip to channel 1 on the 1132 or 1403 printer or 5 consecutive carriage returns on the console printer. The page count is reset to one, and the current page heading is replaced with whatever appears in columns 51 through 58 of the JOB control record. HDNG statements (assembler language) and ** records (FORTRAN header control record) cause additional information to be printed.

EQUAT Record Count. This parameter specifies the number of EQUAT supervisor control records (if any) that follow the JOB control record. These records are read and written in the supervisor control record area (SCRA).

// JOB Examples



// ASM

general function

This control record causes the supervisor to read into core storage and transfer control to the assembler. Any assembler control records used and the source program statements to be assembled must follow an ASM control record. Monitor comments control records (// *) cannot follow an ASM control record.

format

Card column	Contents	Explanation
1 through 6	//ƀASM	
7 through 80	Not used	

// FOR

general function

This control record causes the supervisor to read into core storage and transfer control to the FORTRAN compiler. Any FORTRAN control records used and the source statements being compiled must follow a FOR control record. Monitor comments control records (// *) cannot follow this control record.

format

Card column	Contents	Explanation
1 through 6	//bFOR	
7 through 80	Not used	

// RPG

general function

This control record causes the supervisor to read into core storage and transfer control to the RPG compiler. RPG control cards and specification statements must follow an RPG control record. Monitor comments control records (// *) cannot follow an RPG control record.

format

Card column	Contents	Explanation
1 through 6	//bRPG	
7 through 80	Not used	

// COBOL

general function

This control record causes the supervisor to read into core storage and transfer control to the COBOL compiler (a program product). Monitor comments (// *) control records cannot follow a COBOL control record.

format

Card column	Contents	Explanation
1 through 8	//bCOBOL	
9 through 80	Not used	

// DUP

general function

This control record causes the supervisor to read into core storage and transfer control to the control portion of the Disk Utility Program (DUP). A DUP control record (see "DUP Control Records" in this chapter) must follow this control record. Only one // DUP monitor control record is required to process any number of DUP control records. Monitor comments control records (// *) can follow the DUP monitor control record.

format

Card column	Contents	Explanation
1 through 6	//bDUP	
7 through 80	Not used	

// XEQ

general function

This control record causes the supervisor to initialize for execution of a core load.

Comments control records (// *) can follow an XEQ control record if supervisor control records do not follow and if data is not entered through the principal input device during execution. The comments control records are printed after execution is complete.

format

Card column	Contents	Explanation
1 through 6	//bXEQ	
7	Reserved	
8 through 12	Name	This is the name (left-justified) of the DSF program or DCI program to be executed.
13	Reserved	
14	Core map indicator	*L* or blank. An *L* indicates that a core map is to be printed for this and all DSF programs linked to during this execution.
15	Reserved	
16 and 17	Count	A decimal number (right-justified) that indicates the number of supervisor control records that follow.
18	Reserved	
19	Disk I/O subroutine indicator	This specifies the disk I/O subroutine to be loaded into core by the core image loader for use by the core load during execution.
20	Reserved	
21 through 24	Cartridge ID	The ID of the cartridge that contains the mainline program in its working storage (valid only if a name is not specified in columns 8 through 12; blanks in this field indicate that the program is in system working storage when a name is not specified in columns 8 through 12).
25	Not used	
26	LOCAL-call-LOCAL indicator	A punch in this column enables a LOCAL subroutine to call another LOCAL.
27	Not used	
28	Special ILS indicator	A punch in this column indicates that ILSs for this core load should be chosen from the special ILSs.
29 through 80	Not used	

Note: When column 14 is blank, no warning is given if a file is truncated while a FORTRAN core load is being built.

additional field information

Name. This is the name of the program, stored in the user area or fixed area, that is executed.

When this field is omitted, the program to be executed is assumed to be stored in system working storage, or in working storage on the cartridge specified in columns 21 through 24 of this control record.

Core Map Indicator. An L punched in column 14 of this control record causes the printing of a core map for the program being executed and for all programs linked to during execution (see "Reading a Core Map and a File Map" in Chapter 6 for examples of core maps).

Count. A right-justified decimal number in columns 16 and 17 indicates the number of supervisor control records (LOCAL, NOCAL, FILES, and G2250) that follow this control record.

Disk I/O Subroutine Indicator. A decimal number in column 19 identifies the disk I/O subroutine used by the core load during execution.

Column 19	*Disk I/O subroutine*
blank or Z	DISKZ
0 or 1	DISK1
N	DISKN

Any other character is invalid and causes execution to be bypassed. All DSF programs that are linked to during execution must use the same disk I/O subroutine as the program that calls them.

LOCAL-Call-LOCAL Indicator. A punch (any character) in column 26 provides for a LOCAL subroutine to call another LOCAL subroutine during execution, provided the restrictions listed under "LOCAL-Calls-a-LOCAL" in Chapter 6 are met.

Special ILS Indicator. A punch (any character) in column 28 indicates that special interrupt level subroutines (ILSs named with an X before the number, as ILSX4) are used for this core load. If column 28 is blank, the standard set of ILSs is used.

In addition to the functions of the standard ILSs, special ILSs at the beginning of their execution save the contents of index register 3 and set this register to point to the transfer vector. Special ILSs restore the original contents of index register 3 at the end of their execution. Because the special ILSs save and restore the contents of index register 3, you can use this register in your programs.

Special ILSs require 5 more words of core storage per ILS than standard ILSs. The special ILSs for interrupt levels 2 and 4 are loaded, together with other subroutines, as part of the core load. You can write ILSs to replace any of the IBM-supplied ILSs, standard or special.

// XEQ Examples

```
1   5    10   15   20   25   30   35   40   45   50
// XEQ
// XEQ NAME      ø2       X X
// XEQ       L      1ø04
```

1. This specifies execution of the program stored in working storage on the master cartridge.
2. This specifies that the named program (in the UA or WS) is to be executed, that two supervisor control records follow, that a LOCAL calls another LOCAL, and that the special ILSs are to be used for this core load.
3. This specifies the printing of a core map, and that the program stored in working storage on disk 1004 is to be exectued.

// * (Comments)

general function

This control record causes the alphameric comments contained on the // * control record to be printed on the principal print device. The information is read and printed, and the next control record is read from the input stream. Comments control records can be used preceding a PAUS monitor control record to instruct the operator as to what he is to do during the pause in monitor system operations.

When the console printer is used to print monitor and supervisor control records as a result of a CPRNT monitor control record, comments control records are printed on the principal printer.

Comments control records cannot immediately follow an ASM, RPG, FOR, or COBOL monitor control record. Comments control records can follow an XEQ control record if supervisor control records do not follow and if data is not entered from the principal input device during execution.

format

Card column	Contents	Explanation
1 through 4	//ƀ*	
5 through 80	Comments	Any alphameric characters can be used.

// PAUS

general function

This control record causes the supervisor to pause at a WAIT instruction. Supervisor operation continues when you press PROGRAM START on the console. This pause allows you to perform operator actions, such as add cards to the card reader, change satellite disk cartridges, or change paper tapes within a JOB stream. The status of the monitor system is not changed during a pause.

Monitor comments control records (// *) preceding a PAUS control record can describe the operator actions performed during the pause.

format

Card column	Contents	Explanation
1 through 7	//ƀPAUS	
8 through 80	Not used	

// TYP

general function

This control record temporarily assigns the console keyboard as the principal input device. The keyboard replaces the card or paper tape reader as the principal input device until a TEND monitor control record is entered through the keyboard.

The use of the keyboard as the principal input device for entering control records, program statements, and data is described under "Entering Jobs from the Console Keyboard" in Chapter 7.

format

Card column	Contents	Explanation
1 through 6	//ƀTYP	
7 through 80	Not used	

// TEND

general function

This control record reassigns the card or paper tape reader as the principal input device. The reassignment is to the device that was the principal device before the TYP monitor control record was read.

A TEND control record can be entered only from the keyboard.

format

Card column	Contents	Explanation
1 through 7	//ƀTEND	
8 through 80	Not used	

// EJECT

general function

This control record causes the 1403 Printer or 1132 Printer, whichever is the principal print device, to skip to a new page and print the page header. When the console printer is assigned as the principal printer, or when a CPRNT monitor control record has been processed, 5 lines are skipped and the page header is printed.

format

Card column	Contents	Explanation
1 through 8	//ƀEJECT	
9 through 80	Not used	

// CPRNT

general function

This control record causes monitor and supervisor control records that follow CPRNT to be printed on the console printer. All other control records and monitor comments control records are printed on the principal print device.

An EJECT monitor control record read after a CPRNT affects the console printer rather than the principal print device.

A CEND monitor control record is used to return the printing of monitor and supervisor control records to the principal print device. A system reload and/or the DEFINE VOID function of the Disk Utility Program (DUP) also restores the original principal print device.

format

Card column	Contents	Explanation
1 through 8	//ƀCPRNT	
9 through 80	Not used	

// CEND

general function

This control record restores the printing device that was the principal printer before a CPRNT monitor control record was processed.

format

Card column	Contents	Explanation
1 through 7	//ƀCEND	
8 through 80	Not used	

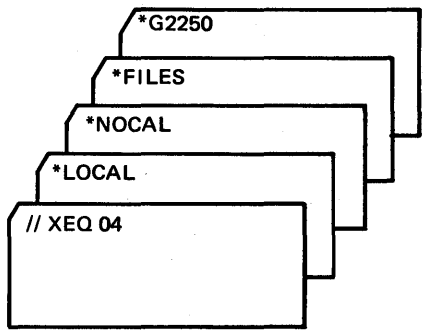
SUPERVISOR CONTROL RECORDS

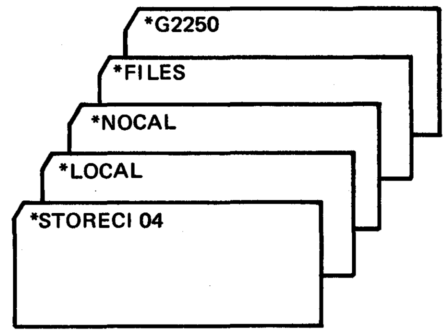
functions

Supervisor control records are used by the core load builder to:

- Provide for subroutine overlays during execution, *LOCAL
- Include in the core load subroutines that are not called, *NOCAL
- Equate disk storage data files defined in a mainline program during compilation or assembly to specific files that are stored on disk, *FILES
- Provide graphic display capabilities, *G2250
- Substitute a subroutine with another subroutine, *EQUAT

LOCAL, NOCAL, FILES, and G2250 supervisor control records are placed in the input stream following an XEQ monitor control record, which names a mainline program stored in disk system format, or following a STORECI DUP control record.

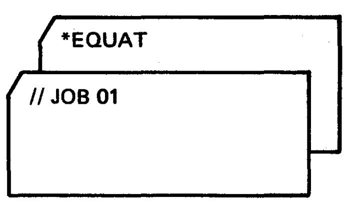




In either case, the control records are written on disk in the supervisor control record area (SCRA), from which the core load builder reads them for processing during construction of a core load.

Up to 99 supervisor control records can follow an XEQ or STORECI control record. Supervisor control records do not have to be placed in any special order by type; however, all the control records of one type must be kept together.

EQUAT control records are placed after a JOB monitor control record and maintain their function until the next JOB control record is read from the input stream.



The supervisor reads EQUAT control records and writes them into the SCRA, from which the core load builder reads them for processing during construction of a core load.

coding

An asterisk (*) is coded in column one of all supervisor control records. The rest of the information specified in supervisor control records, except the G2250 control record, is coded continuously; that is, blanks (referred to as embedded blanks) cannot be coded within the characters in a record. Information specified in the G2250 control record must be coded in the fields indicated in the G2250 format description in this section.

The program name that is coded in all types of supervisor control records can be either the primary entry point name or any secondary entry point name in the program.

*LOCAL

general function

This control record specifies the names of LOCAL (load-on-call) subroutines that are to be read, when called during execution, into the LOCAL overlay area of a core load. (See "Rules for LOCAL and NOCAL Usage" and "LOCAL-Calls-a-LOCAL" in Chapter 6.)

format

```
1   5    10   15   20   25   30   35   40   45   50
*LOCALMAIN1,SUB1,SUB2,...,SUBn
```

Note: **Embedded blanks are not allowed in a LOCAL control record.**

additional field information

MAIN1. You replace MAIN1 with the name of the DSF mainline program that is already stored in the user area on disk.

,SUB1,SUB2,... SUBn. You replace SUB1 through SUBn with the names of the subroutines that are used as LOCALs with the specified mainline program.

continuation records

The specification of LOCAL subroutines can be continued from one LOCAL control record to another by placing a comma after the last subroutine specified on each LOCAL control record, except the last. The name of the mainline program is not included on the continuation control records.

continuation example

```
1   5    10   15   20   25   30   35   40   45   50
*LOCALMAIN1,SUB1,SUB2,
*LOCALSUB3,
.
.
.
*LOCALSUBn
```

The results would be the same if the control records were:

```
1   5    10   15   20   25   30   35   40   45   50
*LOCALMAIN1,SUB1
*LOCALMAIN1,SUB2
.
.
.
*LOCALMAIN1,SUBn
```

All LOCAL subroutines that are used by each mainline program during execution must be specified on LOCAL control records following the XEQ monitor control record that starts execution.

coding for linked programs

Separate LOCAL control records must be used for each mainline program that calls LOCAL subroutines during execution.

example

```
1   5    10   15   20   25   30   35   40   45   50
*LOCALMAIN1,SUB1,SUB2,SUB5,...,SUBn
*LOCALMAIN2,SUB3,SUB4,...,SUBn
```

MAIN2. You replace MAIN2 with the name of a mainline program that is called by the program represented by MAIN1.

mainline program in working storage

When the mainline program is to be executed from working storage, the name of the mainline program is omitted from LOCAL control records. This same format is used when LOCAL control records are specified with the Disk Utility Program (DUP) STORECI operation.

example

```
1   5    10   15   20   25   30   35   40   45   50
*LOCAL,SUB1,SUB2,...,SUBn
```

*NOCAL

general function

This control record specifies the names of NOCAL (load-although-not-called) subroutines that are to be associated with a specified mainline program. NOCAL subroutines are included in the core load even though they are not called. (See "The Use of NOCALs" and "Rules for LOCAL and NOCAL Usage" in Chapter 6.)

NOCAL control records are coded in the same format as LOCAL supervisor control records, except that *NOCAL is coded in place of *LOCAL.

format examples

```
1   5    10   15   20   25   30   35   40   45   50
*NOCALMAIN1,SUB1,SUB2,...,SUBn

*NOCAL,SUB1,SUB2,...,SUBn
```

In the first format example, the specified NOCAL subroutines are included in the core load built for the stored mainline program, MAIN1. In the second format example, the specified NOCAL subroutines are included in the core load built for a mainline program in working storage. See "*LOCAL" for information about continuing a control record to another, and coding for linking between programs.

*FILES

general function

This control record equates the file numbers specified in FORTRAN DEFINE FILE statements or in assembler FILE statements to the names of data files that are stored in the user area and fixed area, or in working storage other than system working storage.

All the data files in the user area or fixed area that are used by core loads during execution must be defined on FILES control records following the XEQ monitor control record that starts execution. All files thus defined are available for use by each core load in the execution.

Data files that are equated for a program that is stored in disk core image (DCI) format must be stored in fixed areas for successful execution of the program. (See "Disadvantages of Storing a Program in Disk Core Image Format" in Chapter 6.) When data files are equated for a DCI program and are stored on other cartridges, the data files must be stored in the same location on the other cartridges as they were when the DCI program was stored for successful program execution. Also, the other cartridges must be on the same logical drives as they were when the DCI program was stored. These restrictions are necessary because the core load builder places in the define file table in the DCI program header an absolute sector address, including the drive code, for each equated data file.

No more than 159 data files can be equated for one execution.

format

```
1   5    10   15   20   25   30   35   40   45   50
*FILES(FILE1,NAME1),...,(FILEn,NAMEn)
*FILES(FILE1,NAME1,CAR1),...,(FILEn,NAMEn,CARn)
*FILES(FILE1,,CAR1),...,(FILEn,,CARn)
```

Note: Embedded blanks are not allowed in a FILES control record.

additional field information

FILE1 Through FILEn. You replace these with the file numbers that are specified in the FORTRAN DEFINE FILE statements or assembler FILE statements in your program.

NAME1 Through NAMEn. You replace these with the names of the data files that are stored on disk. Names can be omitted as in the third *FILES record in the format. When omitted, 2 commas are required in the control record format, and the file is placed in working storage on the specified disk.

CAR1 Through CARn. These are the IDs of the cartridges on which the respective data files are stored. The cartridge ID can be omitted. When omitted, the corresponding data file is assumed to be on the cartridge on the lowest logical drive.

continuation records

The specification of data files can be continued from one *FILES control record to another by placing a comma after the last right parenthesis on each *FILES control record, except the last.

continuation example

```
1   5    10   15   20   25   30   35   40   45   50
*FILES(FILE1,NAME1),
*FILES(FILE2,NAME2,CAR2),
.
.
.
*FILES(FILEn,NAMEn,CARn)
```

*G2250

general function

This control record causes the graphic subroutine package (GSP) communication module (GCOM) to be included in a core load immediately following the mainline program. Other supporting subroutines are also loaded into this area depending on the parameters specified in the *G2250 control record. (See the publication *IBM 1130/2250 Graphic Subroutine Package for Basic FORTRAN IV*, GC27-6934, for instructions on properly loading the mainline program, and for information concerning the use of GSP subroutines as LOCALs and core storage layout requirements.

format

Card column	Contents	Explanation
1 through 11	*G2250mlmne	Specifies that graphic support is required for the named mainline program. You replace *mlmne* with the name of the program. If the program being executed is in working storage, the program name is omitted.
12	Reserved	
13	U, blank, or N	*U* indicates the character stroke subroutine containing upper case, numeric, and special characters is loaded. Blank indicates the character stroke subroutine containing upper case, lower case, numeric, and special characters is loaded. *N* indicates that a character stroke subroutine is not loaded.
14	Reserved	
15	Blank or N	Blank indicates the scissoring subroutine is loaded. *N* indicates the scissoring subroutine is not loaded.
16	Reserved	
17	Blank or N	Blank indicates the ICA area expansion subroutine is loaded. *N* indicates the ICA area expansion subroutine is not loaded.
18	Reserved	
19	Blank or N	Blank indicates the index controlled entity subroutine is loaded. *N* indicates the index controlled entity subroutine is not loaded.
20	Reserved	
21	Blank or N	Blank indicates the level controlled direct entry subroutine is loaded. *N* indicates the level controlled direct entry subroutine is not loaded.
22 through 80	Not used	

examples

```
*G2250MLMNE N N N N N
*G2250MLMNE U
*G2250MLMNE
```

*EQUAT

general function

With this control record, you specify the substitution of subroutines during the building of a core load. This control record can also substitute symbolic names in assembler language DSA statements (limited to assembler programs). The EQUAT control record cannot be used to substitute subroutines for RPG programs.

More than one EQUAT control record can be used if the exact number of records used is punched in columns 60 and 61 of the preceding // JOB monitor control record. (Information about using EQUAT control records is under "Use of the EQUAT Record" in Chapter 6.)

format

```
*EQUAT(SUB1,SUB2),...(SUBm,SUBn)
```

additional field information

SUB1 Through SUBm. You replace these with the names of the subroutines that you want the core load builder to substitute for the subroutines represented by SUB2 through SUBn during the building of a core load. This same order of substitution is used when substituting symbolic names for DSA statements.

Note. The maximum number of pairs of subroutines that can be specified is 25.

During the following functions, the substitution of SUB2 for SUB1 is accomplished in the execution of the mainline program from working storage and the storing of MAIN.

example

```
// JOB                                                        1
*EQUAT(SUB1,SUB2)
 .
 .
 .
// XEQ
 .
 .
 .
// DUP
*STORECI    WS  UA  MAIN
 .
 .
// JOB
```

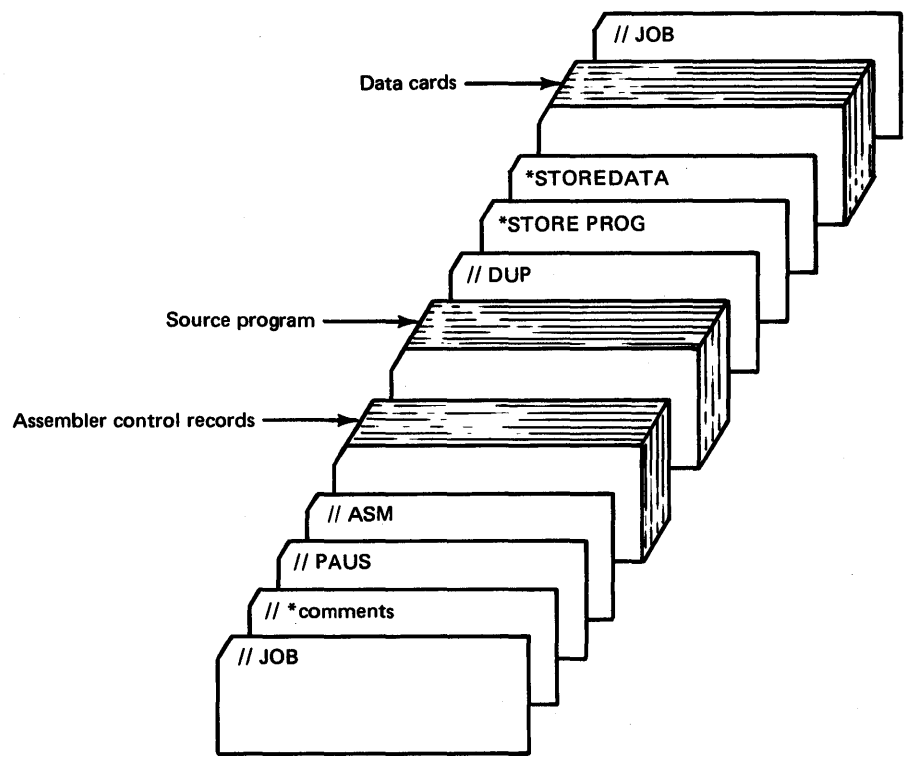
DUP CONTROL RECORDS

functions

DUP control records are used to specify operations to be performed by the Disk Utility Program. The types of operations that DUP control records specify are:

- Dumping and deleting programs and data files from disk
- Storing programs and data files on disk
- Printing the contents of the fixed location equivalence table (FLET) and the location equivalence table (LET)
- Rewriting sector addresses in working storage
- Defining a fixed area on disk
- Deleting monitor system programs from disk
- Allocating disk space for data files and macro libraries
- Calling the Macro Update Program (MUP) into operation

DUP control records are placed in the input stream after a DUP monitor control record (// DUP) as follows:



coding

DUP control records *generally* follow the format described in the following text. All fields in the control record, except the *count* field, are left-justified and, unless otherwise stated, are required. Additional field information is included, when necessary, in the description of the specific control record.

Column 1. Column 1 always contains an asterisk (*).

Operation Field. Code the name of the desired DUP operation in columns 2 through 12 (2 through 21 for the DEFINE operation, and 2 through 13 for the MACRO UPDATE operation). Columns 2 through 6 identify the basic operation (*STORE*DATACI); columns 7 through 12 (or 21) identify the extended operation (STORE*DATACI*). Where shown in the control record format, a blank character (b̸) is required within or following the operation name.

From and To Fields. Code the *from* symbol in columns 13 and 14; that is, the symbol specifying the disk area or I/O device from which information is to be obtained (the source). Code the *to* symbol in columns 17 and 18; that is, the symbol specifying the disk area or I/O device to which information is to be transferred (the destination). The valid *from* and *to* symbols are:

Symbol	Disk area or I/O device
UA	User area on disk
FX	Fixed area on disk
WS	Working storage on disk
CD	Card I/O device. If the 1134 Paper Tape Reader is defined as the principal input device, CD is equivalent to PT.
PT	Paper tape
PR	Principal print device

Note. The symbols UA, FX, and WS, when used, each specify an area on disk but do not identify the cartridge on which the area is found.

Name Field. Code the name of the program, data file, or macro library involved in the specified operation in columns 21 through 25. The name that you specify in this field for a store operation is the name assigned to the program, data file, or macro library, and is used to generate or search for a LET or FLET entry. The name can consist of up to 5 alphameric characters, and must be left-justified in the field. The first character must be alphabetic (A-Z, $, #, @), and blanks (embedded blanks) are not allowed between characters of the name.

When referencing a program or data file stored on disk, the specified name must be an *exact* duplicate of the LET or FLET entry.

Count Field. The count coded in columns 27 through 30 is a right-justified decimal integer. The function of the *count* field is defined in the individual control record formats for those operations that require it.

From and To Cartridge ID Fields. Code the *from cartridge ID* in columns 31 through 34; that is, the ID of the cartridge that contains the disk area from which information is to be obtained. Code the *to cartridge ID* in columns 37 through 40; that is, the ID of the cartridge that contains the disk area to which information is to be transferred.

Either or both of these cartridge IDs can be omitted. When a cartridge ID is omitted, and the corresponding *from* or *to* field (columns 13 and 14 or 17 and 18) is the user area or fixed area, a search is made of the LET (and FLET) on each cartridge specified in the current JOB monitor control record. The search starts with the cartridge on logical drive zero (the master cartridge) and continues through logical drive 4. If the *from* or *to* field (columns 13 and 14 or 17 and 18) is working storage, a default to system working storage is made when cartridge IDs are omitted. When a cartridge ID is specified, the LET (and FLET) only on the specified cartridge is searched, or working storage on the specified cartridge is used.

The use of the *from* and *to cartridge IDs* makes it possible for DUP (1) to transfer programs and data files from one cartridge to another without deleting them from the source cartridge, and (2) to process a program or data file even though the same name appears in the LET or FLET on more than one cartridge.

Unused Columns. All columns indicated as reserved between column 2 and the last format field on each control record must be left blank. The columns between the last format field and column 80 are not used by DUP and are available for your remarks.

Altering LET and FLET

The 2 tables, location equivalence table (LET) and fixed location equivalence table (FLET), are directories to the contents of the user area and fixed area, respectively, on disk. You can alter the contents of these 2 tables through the use of DUP store and delete operations only.

Before storing a program or data file, DUP searches LET and FLET for the name specified in the control record. When a cartridge is specified in the *to cartridge ID* field on the control record, LET (and FLET) on only that disk is searched for the specified name. When a *to cartridge ID* is not specified, LET (and FLET) on all cartridges defined in the current JOB monitor control record is searched. If the specified name is not found in any LET or FLET, disk storage is allocated for the program or data file. The specified name is assigned to the program or data file and is used to generate a new entry in LET or FLET.

When dumping or deleting a program or data file from the user area or fixed area, the name specified in the control record is searched for in LET and FLET in the same order as the search before a store operation. If the specified name is found, the program or data file is dumped or deleted as specified in the control record.

Information Transfer and Format Conversion

Figure 5-1 summarizes the DUP operations that transfer infromation from one device or disk area to another device or disk area. In addition, the format covnersions that are made during the transfer of information are shown. The different formats are described in Appendix I. The acronyms used in Figure 5-1 for the various formats are:

Acronym	Format
DSF	Disk system format
DDF	Disk data format
DCI	Disk core image format
CDS	Card system format
CDD	Card data format
CDC	Card core image format
PTS	Paper tape system format
PTD	Paper tape data format
PTC	Paper tape core image format
PRD	Printer data format
NCF	Name code format

You should pay particular attention to Figure 5-1 when performing dump, store, and delete operations, such as, dumping to cards and later using the cards to store the information back on the disk. Note that more than one way to dump and store data and porgrams is allowed, such as dumping a program to cards and later storing it back to disk.

From Area Symbols, with Formats		*To* Area Symbols, with Formats														
		UA			FX		WS			CD			PT			PR
		DSF	DDF	DCI	DDF	DCI	DSF	DDF	DCI	CDS	CDD	CDC	PTS	PTD	PTC	PRD
UA	DSF						DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP DUMPDATA
	DDF							DUMP DUMPDATA			DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI							DUMPDATA	DUMP		DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
FX	DDF							DUMP DUMPDATA			DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI							DUMPDATA	DUMP		DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
WS	DSF	STORE STOREMOD	STOREDATA	STORECI	STOREDATA	STORECI				DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP DUMPDATA
	DDF		STOREMOD STOREDATA		STOREMOD STOREDATA						DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI		STOREDATA	STOREMOD STOREDATACI	STOREDATA	STOREMOD STOREDATACI					DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
CD	CDS	STORE	STOREDATA	STORECI	STOREDATA	STORECI	STORE	STOREDATA								
	CDD		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							
	CDC		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							
PT	PTS	STORE	STOREDATA	STORECI	STOREDATA	STORECI	STORE	STOREDATA								
	PTD		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							
	PTC		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							

Note: DUMPDATA E and STOREDATAE are the same as DUMPDATA and STOREDATA, respectively, except that information on disk for DUMPDATA E is assumed to be in packed EBCDIC format, and input for STOREDATAE is converted to packed EBCDIC format.

Figure 5-1. Summary of DUP transfer and conversion operations

Restrictions Caused by Temporary Mode

When temporary mode is indicated in the current JOB monitor control record, some DUP operations are restricted or not allowed. The following chart shows the restriction, if any, on DUP operations when temporary mode is indicated.

DUP operations	Restrictions
DUMP	None
DUMPDATA, DUMPDATAbE	None
STORE	None
STORECI	To UA only
STOREDATA, STOREDATAE	To UA and WS only
STOREDATACI	To UA only
STOREMOD	Not allowed
DUMPLET	None
DUMPFLET	None
DWADR	Not allowed
DELETE	Not allowed
DEFINE FIXED AREA	Not allowed
DEFINE VOID ASSEMBLER	Not allowed
DEFINE VOID FORTRAN	Not allowed
DEFINE VOID RPG	Not allowed
DEFINE VOID COBOL	Not allowed
DFILE	To UA only
MACRO UPDATE	Not allowed

*DUMP

general function

This control record (1) transfers information from the user area or fixed area to working storage, or (2) makes information from the user area, fixed area, or working storage available as card, paper tape, or printed output. Card, paper tape, and print formats are illustrated in Appendix I.

DSF programs are transferred from the user area or fixed area to output devices in 2 phases. The programs are first moved to system working storage, then to the output device. As a result, information residing in working storage before the DUMP operation is destroyed.

DCI programs and data files are transferred directly from the user area or fixed area to the output device. The contents of working storage remain unchanged.

DUP obtains the number of disk blocks to be dumped from the LET or FLET entry for a DSF program or a data file, or from the appropriate working storage indicator in sector @DCOM if the dump is from working storage. The actual core load length in words of a DCI program is dumped. The word count is obtained from the core image header. Dumps of a DSF program and a DCI program are contained in Appendix I.

format

Card column	Contents	Explanation
1 through 6	*DUMPb̸	
7 through 12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is required except when the dump is from working storage to the printer.
26 through 30	Reserved	
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers and format conversions performed by the DUMP operation.

*DUMP summary chart

From symbols, including formats	*To* symbols, including formats
UA(DSF)	WS(DSF)
UA or WS(DSF)	CD(CDS) PT(PTS) PR(PRD)
UA or FX(DDF)	WS(DDF)
UA, FX, or WS(DDF)	CD(CDD) PT(PTD) PR(PRD)
UA or FX(DCI)	WS(DCI)
UA, FX, or WS(DCI)	CD(CDC) PT(PTC) PR(PRD)

additional field information

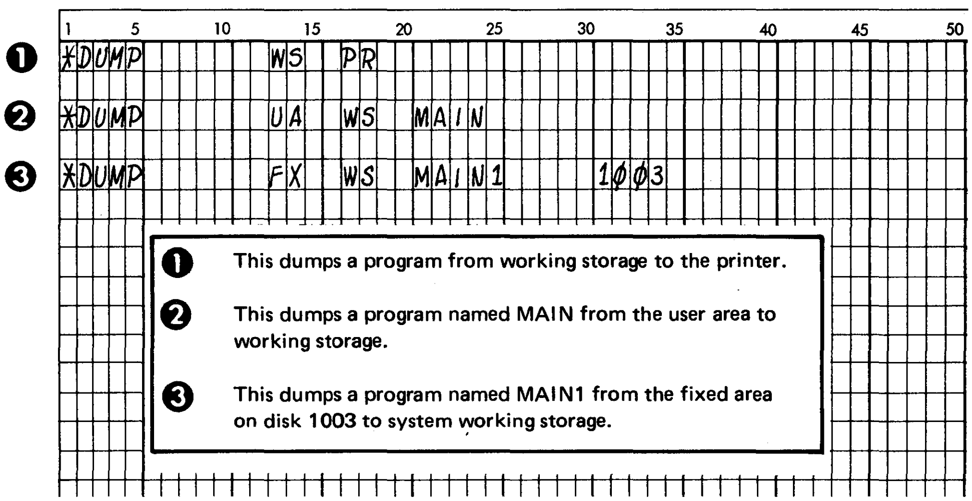
From Symbol. When a dump is from working storage and the corresponding working storage indicator is zero, an error message is printed.

To Symbol. When a dump is to cards and a 1442, Model 6 or 7, is used, each card is checked to see that it is blank before it is punched. If a nonblank card is read, the monitor system prints an error message and waits at $PRET with /100F displayed in the ACCUMULATOR.

Note 1. The program name in a DSF mainline program header is cleared to zeros when the program is transferred from the user area to working storage.

Note 2. The subtype in a subroutine header is set to zero when the subroutine is dumped from the user area to cards

**DUMP Examples*



*DUMPDATA

general function

This control record (1) transfers information from the user area or fixed area on disk to working storage, or (2) makes information from the user area, fixed area, or working storage available as card, paper tape, or printed output. Card, paper tape, and print formats are illustrated in Appendix I.

The contents of working storage are not changed when dumping to output devices, because information is transferred from the user area, fixed area, or working storage directly to the output devices.

The DUMPDATA operation differs from the DUMP operation in that the information is always in data format after transfer. Also, the amount of information transferred depends on the *count* field of the DUMPDATA control record rather than the block count of the program or data file.

format

Card column	Contents	Explanation
1 through 10	*DUMPDATAb̸	
11 and 12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is required except when the dump is from working storage to the printer.
26	Reserved	
27 through 30	Count	The count (a right-adjusted decimal number) specifies the number of sectors to be dumped.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers and format conversions performed by DUMPDATA.

*DUMPDATA summary chart

From symbols, including formats	*To* symbols, including formats
UA(DSF)	WS(DDF)
UA or WS(DSF)	CD(CDD) PT(PTD) PR(PRD)
UA or FX(DDF)	WS(DDF)
UA, FX, or WS(DDF)	CD(CDD) PT(PTD) PR(PRD)
UA(DCI) or FX(DDF)	WS(DDF)
UA, FX, or WS(DCI)	CD(CDD) PT(PTD) PR(PRD)

additional field information

To Symbol. When a dump is to cards and a 1442, Model 6 or 7, is used, each card is checked to see that it is blank before it is punched. If a nonblank card is read, the monitor system prints a message and waits at $PRET with /100F displayed in the ACCUMULATOR.

Count. This field specifies the number of sectors to be dumped. The *count* overrides the contents of the working storage indicator or the disk block count in the LET or FLET entry; this number of sectors is dumped regardless of the length of the program or data file.

**DUMPDATA Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*DUMPDATA   UA  CD  DATA
*DUMPDATA   FX  WS  DATA1     1Ø03  1Ø07
*DUMPDATA   WS  PT  DATA2     1ØØ2
```

❶ This dumps a data file named DATA from the user area to cards.

❷ This dumps a data file named DATA1 from the fixed area on cartridge 1003 to working storage on cartridge 1007.

❸ This dumps a data file named DATA2 from working storage on cartridge 1002 to paper tape.

***DUMPDATA E**

general function

This control record (1) transfers information from the user area or fixed area to working storage, or (2) makes information from the user area, fixed area, or working storage available as card or printed output.

The DUMPDATA E operation to output devices differs from the DUMPDATA operation in that the information on disk, which is assumed to be in packed EBCDIC form, 40 words per 80 card columns, is converted to card image format. Thus, the information printed on a printer is one line per source card (80 print positions), and card output is an exact, full 80 column duplicate of the input cards in the corresponding STOREDATAE operation. When the destination is working storage, format conversion does not occur.

The contents of working storage are not changed when dumping to output devices, because information is transferred from the user area, fixed area or working storage directly to the output devices.

format

Card column	Contents	Explanation
1 through 11	*DUMPDATAbE	
12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is required except when the dump is from working storage to the printer.
26	Reserved	
27 through 30	Count	The count (a right-adjusted decimal number) specifies the number of sectors to be dumped.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers performed by DUMPDATA E.

*DUMPDATA E summary chart

From symbols	*To* symbols
UA or FX	WS
UA, FX, or WS	CD PR

additional field information

To Symbol. When a dump is to cards and a 1442, Model 6 or 7, is used, each card is checked to see that it is blank before it is punched. If a nonblank card is read, the system prints a message and waits at $PRET with /100F displayed in the ACCUMULATOR.

Count. This field specifies the number of sectors to be dumped. The *count* overrides the contents of the working storage indicator and the disk block count in the LET or FLET entry; this number of sectors is dumped regardless of the length of the program or data file.

**DUMPDATA E Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*DUMPDATA E UA  WS  DATA
*DUMPDATA E FX  CD  DATA1     1003
*DUMPDATA E WS  PR
```

1. This dumps a data file named DATA from the user area to working storage.
2. This dumps a data file named DATA1 from the fixed area to cards.
3. This dumps a data file from working storage to the printer.

*DUMPLET

general function

This operation prints the contents of the location equivalence table (LET) on the principal print device. Also, the contents of the fixed location equivalence table (FLET) are printed if a fixed area has been defined on the disk. A program name or data file name can be specified in this control record to dump only the LET or FLET entry for that program or data file. A printout of a DUMPLET operation is in Appendix D.

format

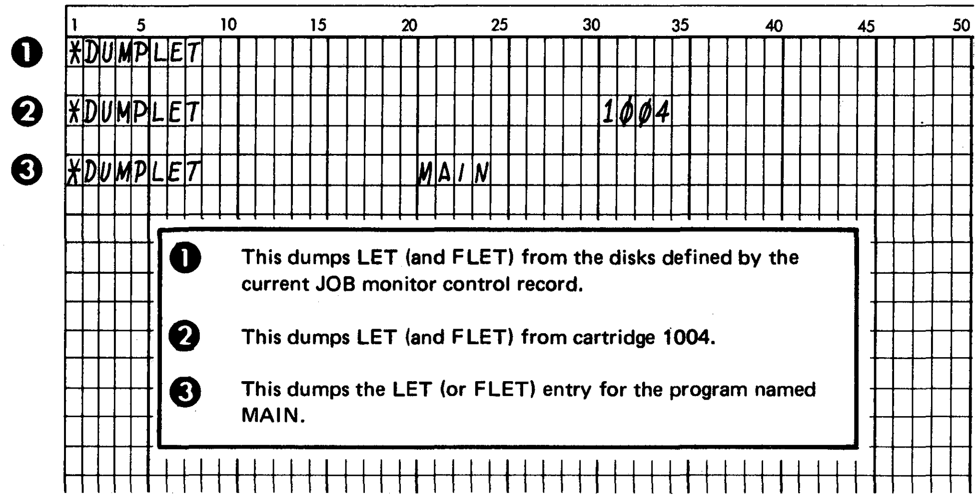
Card column	Contents	Explanation
1 through 8	*DUMPLET	
9 through 20	Reserved	
21 through 25	Name	Name specifies that only the LET or FLET entry for that program or data file is printed.
26 through 30	Reserved	
31 through 34	*From* cartridge ID	The cartridge ID specifies that only the LET (and FLET) on that cartridge is dumped.
35 through 80	Not used	

additional field information

Name. This optional field specifies the name of a program or data file whose LET or FLET entry is to be printed. LET and FLET on all cartridges defined in the current JOB monitor control record are searched unless a cartridge ID is specified in columns 31 through 34. When the name field is omitted, the entire contents of LET (and FLET) are printed.

From Cartridge ID. The *from* cartridge ID specifies that only the LET (and FLET) on that cartridge is printed or searched when a name is specified in columns 21 through 25. When the *from* cartridge ID field is omitted, LET (and FLET) on all cartridges defined by the current JOB monitor control record are printed or searched.

**DUMPLET Examples*



*DUMPFLET

general function

This operation prints the contents of the fixed location equivalence table (FLET) on the principal print device. A program name or data file name can be specified in this control record to dump the FLET entry only for that program or data file.

format

Card column	Contents	Explanation
1 through 10	*DUMPFLETb̸	
11 through 20	Reserved	
21 through 25	Name	Name specifies that only the FLET entry for that program or data file is printed.
26 through 30	Reserved	
31 through 34	*From* cartridge ID	The cartridge ID specifies that only the FLET on that cartridge is printed.
35 through 80	Not used	

additional field information

Name. This optional field specifies the name of a program or data file whose FLET entry is to be printed. FLET on all cartridges defined in the current JOB monitor control record is searched for the name unless a cartridge ID is specified in columns 31 through 34. When the *name* field is omitted, the entire contents of FLET are printed.

From Cartridge ID. The *from* cartridge ID specifies that only the FLET on that cartridge is printed or searched when a name is specified in columns 21 through 25. When the cartridge ID field is omitted, the FLET on all cartridges defined by the current JOB monitor control record is printed or searched.

**DUMPFLET Examples*

```
         1   5    10   15   20   25   30   35   40   45   50
❶        *DUMPFLET
❷        *DUMPFLET           MAIN1
❸        *DUMPFLET           MAIN2     1Ø02
```

❶ This dumps FLET from the disks defined by the current JOB monitor control record.

❷ This dumps the FLET entry for the program named MAIN1.

❸ This dumps the FLET entry for the program named MAIN2 from cartridge 1002.

*STORE

general function

This operation (1) transfers information from working storage to the user area, or (2) accepts information from the input devices and transfers it to working storage or the user area.

All transfer of information from the input devices to the user area is accomplished in 2 phases. The information is first moved to system working storage, then to the user area. Because of this, information residing in working storage before the STORE operation is destroyed, and the appropriate working storage indicator in sector @DCOM is set to zero.

The Disk Utility Program (DUP) makes the required LET entry for the program being stored. The name you specify in columns 21 through 25 is assigned to the program and is used to generate the LET entry. The LET entry includes the program name, the format of the program, the number of disk blocks the program occupies, and the disk block address. An entry is also made in LET for each entry point in the program being stored.

format

Card column	Contents	Explanation
1 through 6	*STORE	
7 through 10	Reserved	
11	Subtype (0, 1, 2, 3, or 8)	For type 3, 4, 5, and 7 subroutines only.
12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is required except when the STORE operation is to working storage.
26 through 30	Reserved	
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers and format conversions performed by the STORE operation.

*STORE summary chart

From symbols, including formats	*To* symbols, including formats
WS(DSF)	UA(DSF)
CD(CDS)	UA or WS(DSF)
PT(PTS)	UA or WS(DSF)

additional field information

Subtype. This optional field places a subtype number in the header of a subroutine, type 3, 4, 5, or 7. The subtype number that can be specified for each type of subroutine is:

Subroutine description	Type	Code in subtype field
In-core subroutines	3, 4	0
Disk FORTRAN I/O subroutines	3	1
Arithmetic subroutines	3	2
Nondisk FORTRAN I/O and "Z"	3	3
"Z" device subroutines	5	3
Function subroutines	4	8
Dummy ILS02, ILS04 stored in monitor system library	7	1
User-written ILS02, ILS04 that replace dummy ILS02, ILS04	7	0

From Symbol. If the STORE operation is from working storage and the corresponding working storage indicator is zero, an error message is printed.

**STORE Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*STORE      CD  WS
*STORE      WS  UA  MAIN
*STORE    0 CD  UA  ILS04
```

1. This reads a program from cards and stores it in working storage.
2. This names a program in working storage MAIN and stores it in the user area.
3. This reads from cards an ILS04 you have written and stores it in the user area.

*STOREDATA

general function

This control record (1) transfers information from working storage to the user area or fixed are, or (2) accepts information from input devices and moves it to working storage, the user area, or fixed area. DUP assumes that input to this operation is in data format; output from this operation is always in data format.

Information is transferred directly from the input devices to the user area or fixed area. Thus, the contents of working storage remain the same if the STORE operation is to the fixed area. Because the boundary between the user area and working storage is moved by store and delete operations, a STOREDATA operation to the user area destroys information residing in working storage before the STOREDATA operation.

DUP makes the required LET or FLET entry. The name you specify in columns 21 through 25 is assigned to the data file or macro library and is used to generate the LET or FLET entry. DUP also supplies the disk block count required in the LET or FLET entry if the source is cards or paper tape. If the source is working storage, the sector count coded in the STOREDATA control record is used.

format

Card column	Contents	Explanation
1 through 10	*STOREDATA	
11 and 12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is not required when the STOREDATA operation is from cards or paper tape to working storage.
26	Reserved	
27 through 30	Count	If the source is working storage, the count is the number (in decimal) of sectors of data to be stored. This count overrides the contents of the working storage indicator. If the source is cards, the count is the number (in decimal) of cards to be read. If the source is paper tape, the count is the number (in decimal) of paper tape records to be read.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

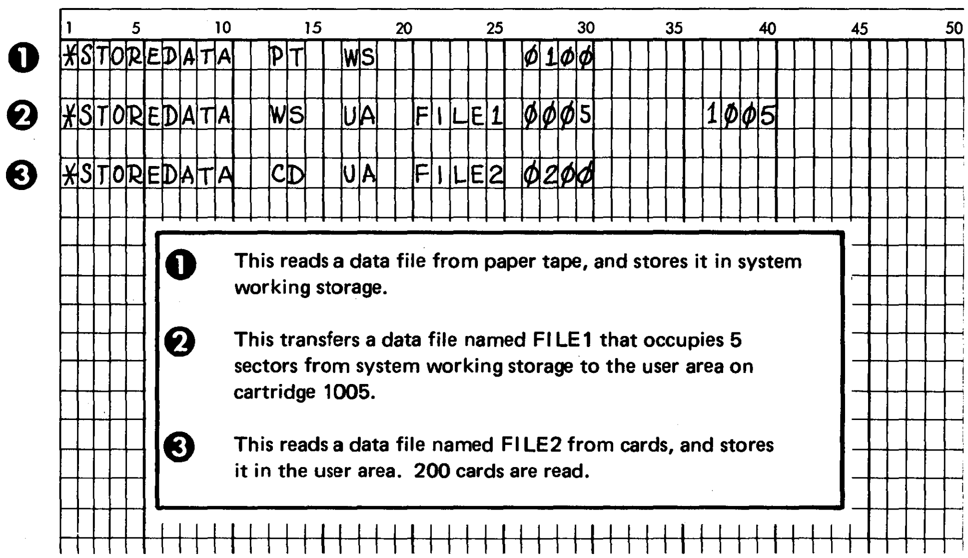
The following chart is a summary of the information transfers and format conversions performed by STOREDATA.

*STOREDATA summary chart

From symbols, including formats	*To* symbols, including formats
WS(DSF, DDF, DCI)	UA or FX(DDF)
CD(CDS, CDD, CDC)	UA, FX, or WS(DDF)
PT(PTS, PTD, PTC)	UA, FX, or WS(DDF)

Note. When temporary mode is indicated in column 8 of the current JOB monitor control record, the STOREDATA operation is restricted to storing in the UA and WS only.

**STOREDATA Examples*



*STOREDATAE

general function

This control record (1) transfers information from working storage to the user area or fixed area, or (2) accepts information from the card reader and transfers it to working storage, the user area, or fixed area.

When input is from cards, the source cards are converted to packed EBCDIC format, that is 2 columns per word, or 8 cards per sector. Thus, the input is assumed to be any of the 256 EBCDIC characters in card code. When the source is working storage, no conversion takes place.

Information is transferred directly from the input device to the user area or fixed area. Thus, when the STOREDATAE operation is to the fixed area, the contents of working storage are not changed. When the STOREDATAE operation is to the user area, the contents of working storage are destroyed because the boundary between the user area and working storage is moved back and forth by delete and store operations.

The Disk Utility Program (DUP) makes the required LET or FLET entry. The name that you specify in columns 21 through 25 is assigned to the data file and is used to generate the LET or FLET entry. Also, DUP supplies the disk block count required in the LET or FLET entry if the source is cards or paper tape. If the source is working storage, the sector count specified in the STOREDATAE control record is used.

format

Card column	Contents	Explanation
1 through 11	*STOREDATAE	
12	Reserved	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is not required when the STOREDATAE operation is from cards to working storage.
26	Reserved	
27 through 30	Count	If the source is working storage, the count is the number (in decimal) of sectors of data to be stored. This count overrides the contents of the working storage indicator. If the source is cards, the count is the number (in decimal) of cards to be read.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers performed by STOREDATAE.

*STOREDATAE summary chart

From symbols, including formats	*To* symbols, including formats
WS	UA or FX
CD	UA, FX, or WS

Note. When temporary mode is indicated in column 8 of the current JOB monitor control record, the STOREDATAE operation is restricted to storing in the UA and WS only.

additional field information

Count. The corresponding dump operation, DUMPDATA E, transfers a whole number of sectors to cards. To avoid unwanted output, the number of cards stored should consequently be a multiple of 8 (blank cards can be added for that purpose).

**STOREDATAE Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*STOREDATAE WS  FX  FILE5 0002
*STOREDATAE CD  WS        0056      1003
```

❶ This transfers a data file named FILE5 from working storage to the fixed area. The file occupies 2 sectors.

❷ This reads a data file of 56 cards into working storage on cartridge 1003.

*STOREDATACI

general function

This control record (1) transfers information from working storage to the user area or fixed area on disk, or (2) accepts information from input devices and moves it to working storage, the user area, or fixed area.

If the input is from cards or paper tape, the STOREDATACI operation assumes the input is in card or paper tape core image format. If the input is from working storage (the information has been previously dumped to working storage or stored in working storage from an input device), the appropriate working storage indicator must indicate disk core image (DCI) format; otherwise, the STOREDATACI operation is not performed. Output from the STOREDATACI operation is always in disk core image format.

All transfer of information from input devices to the user area or fixed area is done directly; that is, the transfer is not made via working storage. Thus, when the STOREDATACI operation stores information from an input device to the fixed area, the contents of working storage are not destroyed. Note, however, the contents of working storage are destroyed when storing from an input device to the user area because the boundary between the user area and working storage is moved back and forth by delete and store operations.

The Disk Utility Program (DUP) makes the required LET or FLET entry. The name that you specify in columns 21 through 25 is assigned to the data file and is used to generate the LET or FLET entry. Also, DUP computes the disk block count required in the LET or FLET entry from the count specified in the STOREDATACI control record.

format

Card column	Contents	Explanation
1 through 12	*STOREDATACI	
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	A name is not required when the STOREDATACI operation is to working storage.
26	Reserved	
27 through 30	Count	The count (a right-justified decimal number) is the number of records (sectors, cards, or paper tape records) in the core image input. The count is not required if the source is working storage; however, when used in this case, the count overrides the contents of the working storage indicator.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

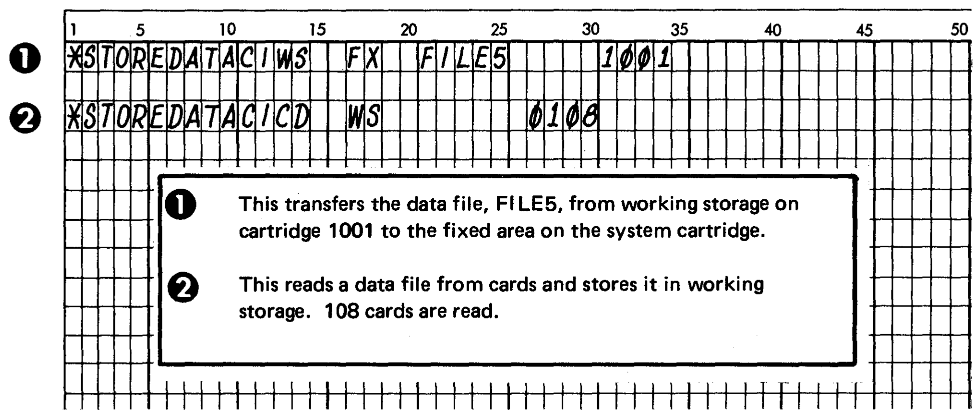
The following chart is a summary of the information transfers and format conversions performed by STOREDATACI.

***STOREDATACI summary chart**

From symbols, including formats	*To* symbols, including formats
WS(DCI)	UA or FX(DCI)
CD(CDC, CDD)	UA, FX, or WS(DCI)
PT(PTC, PTD)	UA, FX, or WS(DCI)

Note. When temporary mode is indicated in column 8 of the current JOB monitor control record, the STOREDATACI operation is restricted to storing in the UA only.

**STOREDATACI Examples*



*STORECI

general function

This control record obtains an object program from working storage or from an input device, converts it into a core image program using the core load builder, and stores the core image program in the user area or fixed area.

The core load builder (CLB) is called to build a core image program for the STORECI operation as if execution were to follow; that is, that portion of the core load residing below core location 4096 (decimal) in 4K systems, or 5056 in larger systems, is placed in the system core image buffer, and LOCALs and/or SOCALs are placed in system working storage. (See "Construction of a Core Load" in Chapter 3.) The STORECI operation stores all these portions of the core image program in the user area, fixed area, or working storage.

A DCI program stored in the user area or fixed area includes the transfer vector built by the core load builder; however, neither the disk I/O subroutine nor COMMON, if any, is included.

The Disk Utility Program (DUP) makes the required LET or FLET entry for the core image program as it is stored. The name that you specify in columns 21 through 25 is assigned to the DCI program and is used to generate the LET or FLET entry. Also, DUP obtains the disk block count required in the LET or FLET entry from the core load builder.

format

Card column	Contents	Explanation
1 through 8	*STORECI	
9	Disk I/O subroutine indicator	This column specifies the disk I/O subroutine to be used by the core load during execution.
10	Reserved	
11	LOCAL-can-call-LOCAL indicator	A punch (any character) in this column enables a LOCAL subroutine to call another LOCAL.
12	Special ILS indicator	A punch (any character) in this column indicates that ILSs for this core load should be chosen from the special ILSs.
13 and 14	*From* symbol	See the following summary chart.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	
26	Reserved	
27 through 30	Count	A decimal number (right-justified) that indicates the number of supervisor control records (FILES, LOCAL, NOCAL, and G2250) that follow.
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41	Reserved	
42	Core map indicator	*N* or blank. An *N* indicates that a core map is not to be printed for this core load. A blank causes a core map to be printed.
43 through 80	Not used	

The following chart is a summary of the information transfers and format conversions performed by STORECI.

*STORECI summary chart

From symbols, including formats	*To* symbols, including formats
WS(DSF)	UA or FX(DCI)
CD(CDS)	UA or FX(DCI)
PT(PTS)	UA or FX(DCI)

Note. When temporary mode is indicated in column 8 of the current JOB monitor control record, the STORECI operation is restricted to storing in the UA only.

additional field information

Disk I/O Subroutine Indicator. This column specifies the disk I/O subroutine that is loaded into core by the core image loader for use by the core load during execution. The character punched in this column for each disk I/O subroutine is:

Column 9	*Disk I/O subroutine*
0 or 1	DISK1
N	DISKN
blank or Z	DISKZ

Any other character is invalid and causes the printing of an error message.

LOCAL-Call-LOCAL Indicator. A punch (any character) in column 11 allows a LOCAL subroutine to call another LOCAL subroutine during execution if the restrictions listed under "LOCAL-Calls-a-LOCAL" in Chapter 6 are met.

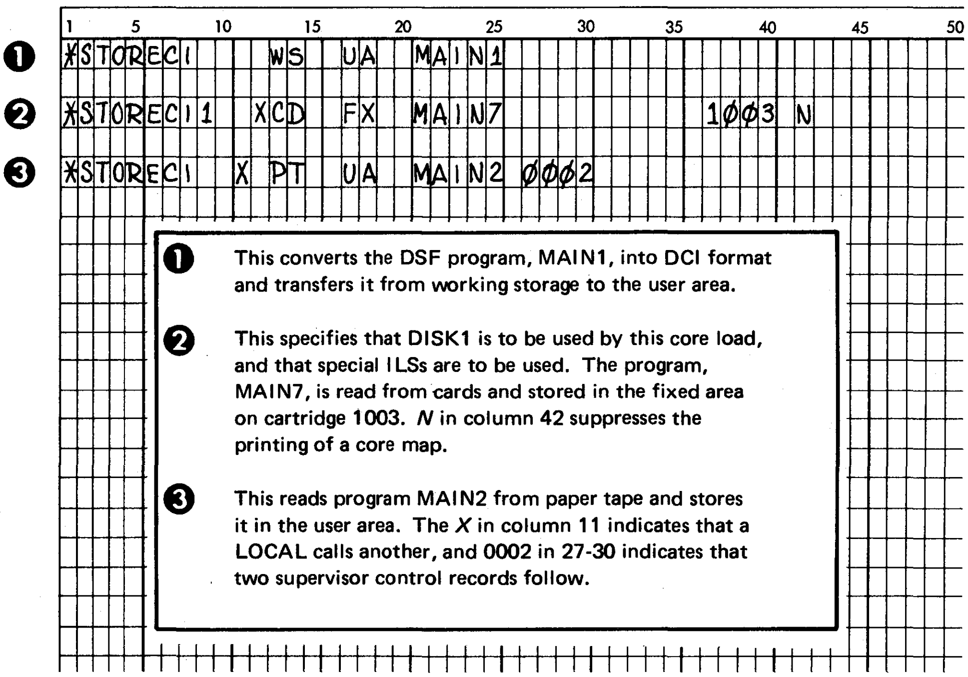
Special ILS Indicator. A punch (any character) in column 12 indicates that special interrupt level subroutines (ILSs named with an X before the number, as ILSX4) are to be used for this core load. If column 12 is blank, the standard set of ILSs is used.

In addition to the functions of the standard ILSs, special ILSs at the beginning of their execution save the contents of index register 3 and set this register to point to the transfer vector. Special ILSs restore the original contents of index register 3 at the end of their execution. Because the special ILSs save and restore the contents of index register 3, you can use this register in your programs.

Special ILSs require 5 more words of core storage per ILS than standard ILSs. The special ILSs for interrupt levels 2 and 4 are loaded, together with other subroutines, as part of the core load. You can write ILSs to replace any of the IBM-supplied ILSs, standard or special.

Count. A right-justified number in columns 27 through 30 that indicates the number of supervisor control records following this control record. DUP reads these control records for use by the core load builder before the STORECI operation is performed. The program name (columns 21 through 25 of this control record) must not be used on the LOCAL, NOCAL, and G2250 control records. Data files specified in the FILES supervisor control records that follow must be stored in the fixed area (see "Use of Defined Files" in Chapter 6).

**STORECI Examples*



STOREMOD

general function

This control record transfers information from working storage into the user area or fixed area.

If the name specified in columns 21 through 25 is identical to an entry in LET or FLET, the information in working storage overlays the DSF program, DCI program, or data file in the user area or fixed area for that entry. The format of working storage must match the format of the LET or FLET entry that is replaced.

The STOREMOD operation permits you to modify a DSF program, DCI program, or data file stored in the user area or fixed area without changing its name or relative position within the storage area. However, the length of the program or data file in working storage after being changed cannot be greater than the length of the old version of the program or data file that it replaces in the user area or fixed area. No change is made to the LET or FLET entry as a result of this operation.

If the name on the STOREMOD control record does not match an entry in LET or FLET, a simple STORE operation is performed (see "*STORE" in this section). The STOREMOD operation is not allowed when temporary mode is indicated in the current JOB monitor control record.

format

Card column	Contents	Explanation
1 through 10	*STOREMODḇ	
11 and 12	Reserved	
13 and 14	*From* symbol	The source is *always* working storage.
15 and 16	Reserved	
17 and 18	*To* symbol	See the following summary chart.
19 and 20	Reserved	
21 through 25	Name	
26 through 30	Reserved	
31 through 34	*From* cartridge ID	
35 and 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

The following chart is a summary of the information transfers and format conversions performed by STOREMOD.

***STOREMOD summary chart**

From **symbols, including formats**	*To* **symbols, including formats**
WS(DSF)	UA(DSF)
WS(DDF)	UA or FX(DDF)
WS(DCI)	UA or FX(DCI)

Note: The format and size indicators of a data file in working storage must match those of the existing LET or FLET entry. Since the execution of your program that references data files stored in working storage does not set these indicators, a subsequent STOREMOD does not work. These indicators can be set prior to execution by performing a DUMPDATA operation of the stored data file to WS.

**STOREMOD Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*STOREMOD   WS  UA  MAIN1
*STOREMOD   WS  FX  FILE1     1ØØ2  1ØØ3
```

1. This replaces the program, MAIN1, stored in the user area with an updated version from working storage.

2. This replaces the data file, FILE1, stored in the fixed area on cartridge 1002 with an updated version from working storage on cartridge 1003.

*DELETE

general function

This operation removes a specified DSF program, DCI program, or data file from the user area or fixed area. The deletion is accomplished by the removal of the program or data file LET or FLET entry, including the dummy entry for associated padding, if any. The DELETE operation is not allowed if temporary mode is indicated in the current JOB monitor control record.

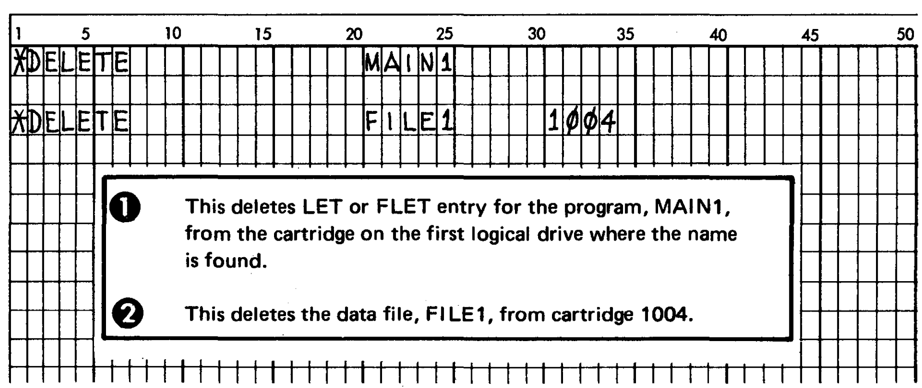
When a program or data file is deleted from the user area, that area is packed so that (1) the areas represented by the remaining LET entries are contigious, and (2) working storage is increased by the amount of disk storage formerly occupied by the deleted program or data file. The contents of working storage are not destroyed by the DELETE operation.

When a DCI program or a data file is deleted from the fixed area, that area is not packed. The FLET entry for the deleted DCI program or data file, including the dummy entry for associated padding, if any, is replaced by a single dummy entry (1DUMY). This 1DUMY entry represents the area formerly occupied by the deleted DCI program or data file, and its padding. DUP store operations can place new entries in the deleted areas of the fixed area.

format

Card column	Contents	Explanation
1 through 8	*DELETEb̸	
9 through 20	Reserved	
21 through 25	Name	
26 through 30	Reserved	
31 through 34	*From* cartridge ID	The deletion is performed on the specified cartridge only. If a cartridge ID is not specified, and the program or data file name (columns 21 through 25) is present in LET or FLET of more than one cartridge specified for this JOB, deletion is from the first logical drive on which the name is found.
35 through 80	Not used	

**DELETE Examples*

❶
❷


*DEFINE

general function

This control record performs 3 functions.

- It initially establishes the fixed area and its size on disk.
- It increases or decreases the size of the fixed area.
- It deletes the assembler, FORTRAN compiler, RPG compiler, or COBOL compiler, or any combination of these 4 programs from the IBM system area on the master cartridge.

define a FX

The definition of a fixed area on disk allows you to store in fixed locations the programs and data files, which you can subsequently refer to by their sector addresses. The fixed area is defined in cylinder increments; the minimum required storage space is one cylinder. When a fixed area is defined, the system uses one cylinder for the fixed location equivalence table (FLET). This cylinder used for FLET is included in the total size of the fixed area; therefore, the initial definition of the fixed area must be at least 2 cylinders.

increase or decrease the FX

Increases and decreases in the size of the fixed area must also be made in cylinder increments. The fixed area cannot be decreased by a number greater than the number of unused cylinders after the last program or data file stored in the fixed area. If all DCI programs and data files have been deleted from the fixed area, and a DEFINE FIXED AREA control record decreases the fixed area to less than 2 cylinders, the fixed area and FLET are deleted from the cartridge. The fixed area and FLET are also deleted if the DEFINE FIXED AREA control record specifies a decrease that exceeds the number of cylinders of the fixed area.

format of DEFINE FIXED AREA

Card column	Contents	Explanation
1 through 8	*DEFINEƀ	
9 through 18	FIXEDƀAREA	
19 through 26	Reserved	
27 through 30	Count	In initial definition of the fixed area, the count is the number (in decimal) of cylinders to be allocated as the fixed area; a minimum of 2 must be specified. After initial definition, the count is the number of cylinders by which the fixed area is to be increased or decreased.
31	Sign	Blank if the fixed area is being increased; a minus sign if the fixed area is being decreased.
32 through 36	Reserved	
37 through 40	Cartridge ID	This ID specifies the cartridge that is being altered; when omitted, the system cartridge is assumed.
41 through 80	Not used	

Note. The DEFINE FIXED AREA operation is not allowed if temporary mode is indicated in the current JOB monitor control record.

**Define Fixed Area Examples*

```
   1   5    10   15   20   25   30   35   40   45   50
❶ *DEFINE FIXED AREA         ØØØ5
❷ *DEFINE FIXED AREA         ØØØ2-     1ØØ2
```

❶ This defines a 5 cylinder fixed area on the master cartridge.

❷ This decreases the size of the fixed area on cartridge 1002 by 2 cylinders.

delete the assembler or compiler

Deletion of the assembler, FORTRAN compiler, RPG compiler, or COBOL compiler causes the specified monitor program to be removed from the IBM system area on the master cartridge. The IBM system area is then packed so that remaining programs and areas occupy the area formerly occupied by the deleted monitor program. SLET entries are updated to reflect the new disk storage allocations for the monitor programs. The reload table is used to make adjustments in the programs that use disk storage addresses from SLET.

When the assembler, FORTRAN compiler, RPG compiler, or COBOL compiler is to be deleted, you must perform this deletion before defining a fixed area on the cartridge, or after completely removing a defined fixed area (see the previous discussion of decreasing the size of the fixed area). Once one of these programs is deleted, it can be restored by performing an initial load only.

format of DEFINE VOID

Card column	Contents	Explanation
1 through 8	*DEFINEƀ	
9 through 13	VOIDƀ	
14 through 22	ASSEMBLER or FORTRANƀƀ or RPGƀƀƀƀƀƀ or COBOLƀƀƀƀ	
23 through 80	Not used	

Note. The DEFINE VOID operation is not allowed when temporary mode is indicated in the current JOB monitor control record.

The processing of a DEFINE VOID operation restores the original system principal printer if a CPRNT monitor control record has specified that monitor and supervisor control records be printed on the console printer.

*DWADR

general function

This operation causes a sector address to be written on every sector of working storage on the cartridge specified by the DWADR control record or, if a cartridge ID is not specified, on every sector of system working storage. The operation restores correct disk sector addresses in working storage if they have been modified during execution of your program. The contents of working storage prior to the DWADR operation are destroyed.

A dummy // DUP monitor control record is printed on the principal printer following the printing of the *DWADR control record and the DUP exit message.

format

Card column	Contents	Explanation
1 through 6	*DWADR	
7 through 36	Reserved	
37 through 40	Cartridge ID	This ID specifies the cartridge on which the working storage sector addresses are to be rewritten.
41 through 80	Not used	

Note. The DWADR operation is not allowed if temporary mode is indicated in the current JOB monitor control record.

*DFILE

general function

This operation reserves disk space in either the user area or fixed area as a named data file or macro library. Data is not moved as a result of the DFILE operation; this function provides disk space allocation only. The contents of working storage are not changed except when defining space in the user area; the contents of working storage on that drive are destroyed since the user area and working storage are adjacent areas. (See "Use of Defined Files" in Chapter 6 for a suggested use of this control record.)

DUP makes the required LET or FLET entry. The name specified on the DFILE control record is assigned to the area and is used to generate the LET or FLET entry. DUP uses the sector count specified on the DFILE control record to supply the disk block count in the LET or FLET entry.

format

Card column	Contents	Explanation
1 through 6	*DFILE	
7 through 16	Reserved	
17 and 18	*To* symbol	Area in which the file is to be reserved: UA for user area, FX for fixed area.
19 and 20	Reserved	
21 through 25	File name	The name assigned to the area reserved for the data file or macro library.
26	Reserved	
27 through 30	Count	The number (in decimal) of sectors to be reserved
31 through 36	Reserved	
37 through 40	*To* cartridge ID	
41 through 80	Not used	

Note. The DFILE operation is restricted to reserving space only in the UA when temporary mode is indicated in the current JOB monitor control record.

*MACRO UPDATE

general function

This operation causes execution of the Macro Update Program (MUP). The MUP performs:

- Initialization of a macro library
- Physical or logical concatenation of macro libraries
- Addition, deletion, or name redefinition of stored macros
- Statement addition or deletion within a stored macro
- Punching of stored macros into cards
- Listing of macro library contents either at statement or macro level

The functions to be performed by MUP are indicated by means of MUP control statements. The format and functions of these control statements are described in the publication *IBM 1130/1800 Assembler Language,* GC26-3778. The MUP control statements immediately follow the MACRO UPDATE DUP control record in the job stream.

The Macro Update Program requires an IBM 1131 Central Processing Unit, Model 2 or 3, with 8192 (decimal) or more words of core storage. If the MACRO UPDATE DUP control record is read by a system with 4096 words of core storage, it is considered an invalid control record. The MUP cannot be used if temporary mode is indicated in the current JOB monitor control record.

format

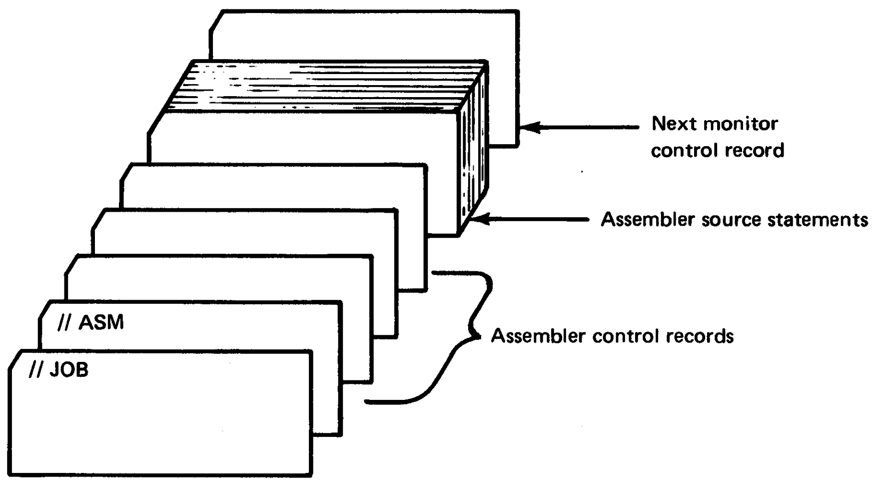
Card column	Contents	Explanation
1 through 13	*MACRObUPDATE	
14 through 36	Reserved	
37 through 80	Not used	

Note. Keyboard or paper tape input to the MUP of the Disk Utility Program assumes a one-to-one relationship with any corresponding card input record. Thus, position 1 of assembler statements that are input record for MUP corresponds to card column 1 and not to column 21.

ASSEMBLER CONTROL RECORDS

functions

Assembler control records are used to specify optional operations that affect the assembler and assembly output. These control records are placed in the input stream as follows:



Assembler control records can be entered in card or paper tape form along with the source program card deck or paper tape, or they can be entered from the console keyboard (see "Entering Jobs From the Console Keyboard" in Chapter 7).

In most cases, the source program is passed through the assembler only once. This is always true when input is from the keyboard or paper tape reader. When input is from cards, passing the source deck through the assembler a second time (2-pass mode) may be required. Further information about 2-pass mode is presented in the descriptions of the TWO PASS MODE, LIST DECK, and LIST DECK E control records in this section. These 3 control records and the PUNCH SYMBOL TABLE control record are *ignored* when entered from the keyboard or paper tape reader.

coding assembler control records

All assembler control records have the following format:

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	Option	Replace *option* with the keywords for the control record being used.
72 through 80	Not used	

Note. Assembler control records are coded in free form; that is, any number of blanks can occur between the characters of the *option.* However, only one blank can separate the last character of the *option* and the first character of any required numeric field. Remarks can be included after the option or numeric field; however, at least one blank must separate the last character of the option or numeric field and the remarks.

If an assembler control record contains an asterisk in column one, but the *option* is not identical with the format shown for the control record, the control record followed by an assembler error message is printed in the control record listing. The control record in error is ignored; an error does not result, but the specified *option* is not performed.

coding keyboard and paper tape input

Assembler control records are coded the same for card, paper tape, and keyboard input. Assembler language source statements are coded the same for keyboard and paper tape input as for cards, with the following exceptions:

- The source statements do not contain leading blanks corresponding to card columns 1 through 20.
- The source statements are limited to 60 characters

The first record processed by the assembler is checked for an asterisk as the first character. If an asterisk is the first character, the record is considered an assembler control record. This procedure continues until the first nonasterisk character is detected as the first character. For this record, and all following records (up to and including the END statement), the first character of each record is treated as if it were in card column 21; therefore, the first noncontrol record should not be an * comments statement.

Note 1. Paper tape input to the assembler is punched into paper tape in PTTC/8 code, one frame per character. Any delete codes punched in paper tape are passed over by the assembler; assembly is continuous until the end.

Note 2. Keyboard and paper tape input to the Macro Update Program (MUP) of DUP assumes a one-to-one relationship with the corresponding card input. Thus, position one of assembler statements that are input for MUP corresponds to card column 1 and not to column 21.

*TWO PASS MODE

general function

This control record causes the assembler to read the source program deck twice. TWO PASS MODE must be specified when:

- You want a list deck punched by the 1442 Card Read Punch, Model 6 or 7 (see "*LIST DECK" and "*LIST DECK E" in this chapter).
- A one-pass operation cannot be performed because the intermediate output (source records) exceeds the capacity of working storage.

This control record is *ignored* if source statements are entered through the keyboard or the paper tape reader.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	TWO PASS MODE	
72 through 80	Not used	

If a copy of the source deck, including all assembler control records, is placed behind the original, the source deck is read twice, and a stacked job is possible in 2-pass mode.

When a deck is being assembled in 2-pass mode, the assembler is ready to read another card as soon as pass one processing of the END card is completed. Therefore, the source deck or a copy of the source deck must be placed immediately behind the END card of the first-pass deck. A monitor control record after the first END card causes the assembler to execute a CALL EXIT; the assembly is not completed.

If the source deck has not been copied, the END card must be the last card in the hopper. To continue:

1. Press START on the card reader and PROGRAM START on the console to process the END card when the reader goes not ready.
2. Remove the source deck from the stacker and place it in the hopper.
3. Press START on the card reader and PROGRAM START on the console again.

The operation can be made continuous if you remove the source cards from the stacker during pass one and place them behind the END card in the hopper.

To complete the assembly at the end of pass 2, press START on the card reader and PROGRAM START on the console to process the END card for the second pass.

*LIST

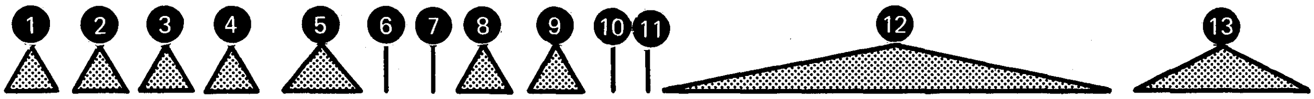
general function

This control record causes the assembler to provide a printed listing of the source program on the principal print device (1403 Printer, 1132 Printer, or console printer). If a LIST control record is not used, only those statements in which assembly errors are detected are listed. When 2-pass mode is specified, all BSS, BES, ORG, and EQU statements that contain errors are listed during pass one of the assembly.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	LIST	
72 through 80	Not used	

The format of a printed listing for an 8K or larger system is shown by:




```
100F 0  A814      00018 M       D       SAVE+2     DIVIDE BY (I+J)
1010 0  8012      00019 M       A       SAVE+1       AND ADD (A+B)/C
1011 0  3000      00020         WAIT
1012 0  6038      00021         EXIT
1014    0000      00022         BSS   E 0
1014 00 0000C000  00023    B    DEC     49152
1016 00 0000E000  00024    F    DEC     57344
```

1 Address of the instruction; address of the label, if any

2 Relocation indicators

3 One of the following:

a. First word of the assembled code

b. For EBC statements, the number of EBC characters

c. For BSS and BES statements, the number of words reserved for the block

d. For ENT, ILS, and ISS statements, 3 and 4 are the entry label in name code

e. For LIBF and CALL statements, 3 and 4 are the name of the subroutine in name code

4 One of the following:

a. Second word of assembled code

b. For ENT, ILS, and ISS statements, 3 and 4 are the entry label in name code

c. For LIBF and CALL statements, 3 and 4 are the name of the subroutine in name code

5 Statement number

6 Error flags, if any

7 Macro code indicator, if any

8 Label

9 Operation code

10 Format

11 Tag

12 Operands (and your comments)

13 ID and sequence number, if any

When LIST is specified for a 4K system, or with 2-pass mode, the format of the printed listing is:

```
100F 0  A814      M        D        SAVE+2       DIVIDE BY (I+J)
1010 0  8012      M        A        SAVE+1         AND ADD (A+B)/C
1011 0  3000               WAIT
1012 0  6038               EXIT
1014    0000               BSS   E  0
1014 00 0000C000     B     DEC      49152
1016 00 0000E000     F     DEC      57344
```

1 Address of the instruction; address assigned to the label, if any

2 Relocation indicators

3 One of the following:

a. First word of the assembled code
b. For EBC statements, the number of EBC characters
c. For BSS and BES statements, the number of words reserved for the block
d. For ENT, ILS, and ISS statements, 3 and 4 are the entry label in name code
e For LIBF and CALL statements, 3 and 4 are the name of the subroutine in name code

One of the following:

a. Second word of assembled code
b. For ENT, ILS, and ISS statements, 3 and 4 are the entry label in name code
c. For LIBF and CALL statements, 3 and 4 are the name of the subroutine in name code

5 Error flags, if any

6 Macro code indicator, if any

7 Label

8 Operation code

9 Format

10 Tag

11 Operands (and your comments)

12 ID and sequence number, if any

A complete sample program listing is in Appendix H.

*XREF

general function

This control record causes the assembler to produce a statement numbered listing and a statement numbered cross-reference symbol table on the principal print device if the core size is 8K or larger. This control record is invalid if the core size is 4K, and, if detected, is ignored. A warning message is printed.

A LIST control record is not needed when XREF is used. When neither an XREF nor a LIST control record is used, only those statements in which assembly errors or warnings are detected are listed. When 2-pass mode is specified, all BSS, BES, ORG, and EQU statements that contain errors are listed during pass one of the assembly.

The cross-reference symbol table is not printed if 2-pass mode is specified or if symbol table overflow occurs during assembly. When either of these conditions occur, the XREF control record produces only a listing.

The assembler does not assign sequence numbers to comments statements when a LIST OFF statement in your program is in effect. Because of this, the statement numbers in a cross-reference symbol table listing for the same program may be different from one assembly to another, depending on whether or not the program contains LIST OFF (and LIST ON) statements.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	XREF	
72 through 80	Not used	

The format of the statement-numbered listing is the same as the format shown under "*LIST" for a system with a core size of 8K or larger. The format of the cross-reference symbol table is:

```
 1           2      3     4           5
K1          105D    0   00071      00007,R 00013,R 00038,R 00057,R 00063,R
K16         106C    0   00083      00123,R
K20         105E    0   00072
K32         105F    0   00073
K40         1060    0   00074      00065,R
K640        1061    0   00075      00003,R 00019,R
LINE        159F    0   00131      00044,R 00116,R 00117,R 00121,R
LINES       1064    0   00078      00062,R 00064,M 00068,M
LOOP        1022    0   00026      00040,B
```

1 Symbol

2 Value of the symbol

3 Relocation indicator

4 Statement number of statement that defines the symbol

5 Statement numbers and associated reference type indicators (B for branch to, M for modification, or R for reference to) for the statements that use the symbols

Multiply defined symbols are flagged in the cross-reference symbol table with the message ***MULTIPLY-DEFINED***. Undefined symbols are listed separately under the header ***UNDEFINED SYMBOLS***. Symbols that refer to the system symbol table are flagged with SYSMB in the statement number field of the cross-reference entry.

A list of the statement numbers of all statements flagged with errors or warnings is printed at the end of the statement numbered listing under the header: ERROR STATEMENT LINE NUMBERS.

*LIST DECK

general function

This control record causes a list deck to be punched when the principal I/O device is a 1442 Model 6 or 7 Card Read Punch. This control record is *ignored* if entered from the 2501 Card Reader, the paper tape reader, or the keyboard.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	LIST DECK	
72 through 80	Not used	

The LIST DECK option requires 2 passes of the source deck (TWO PASS MODE) through the assembler. Object information is punched into columns 1 through 19 during pass two.

The card column contents of a punched list deck card are:

Card column	Contents
1 through 4	Address of the instruction; address assigned to the label, if any.
5	Blank
6 and 7	Relocation indicators
8	Blank
9 through 12	One of the following: 1. First word of the assembled code. 2. For EBC statements, the number of EBC characters. 3. For BSS and BES statements, the number of words reserved for the block. 4. For ENT, ILS, and ISS statements, columns 9 through 16 contain the entry label in name code. 5. For LIBF and CALL statements, columns 9 through 16 contain the name of the subroutine in name code.
13 through 16	One of the following: 1. Second word of the assembled code. 2. For ENT, ILS, and ISS statements, columns 9 through 16 contain the entry label in name code. 3. For LIBF and CALL statements, columns 9 through 16 contain the name of the subroutine in name code.
17	Blank
18 and 19	Error flags, if any
20	Macro code indicator, if any
21 through 25	Label
26	Blank
27 through 30	Operation code
31	Blank
32	Format
33	Tag
34	Blank
35 through 71	Operands (and your comments)
72	Blank
73 through 80	ID and sequence number, if any

*LIST DECK E

general function

This control record causes a list deck to be punched when the principal I/O device is a 1442 Model 6 or 7 Card Read Punch. This control record is *ignored* if entered from a 2501 Card Reader, paper tape reader, or the keyboard.

The LIST DECK E option requires 2 passes of the source deck (TWO PASS MODE) through the assembler. Only error flags, if any, are punched (columns 18 and 19) during the second pass. Assembler error detection codes are described in Appendix A.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	LIST DECK E	
72 through 80	Not used	

*PRINT SYMBOL TABLE

general function

This control record causes the assembler to print a listing of the symbol table on the principal print device. The printed symbols are grouped 5 per line. Multiply defined symbols are preceded by the letter M. Symbols with absolute values in a relocatable program are preceded by the letter A. These M and A flags are not counted as assembly errors.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	PRINT SYMBOL TABLE	
72 through 80	Not used	

*PUNCH SYMBOL TABLE

general function

This control record causes the symbol table to be punched as a series of EQU source cards. Each source card contains one symbol. These cards can be used as source input to the system symbol table when the SAVE SYMBOL TABLE control record is used with an assembly in which they are included.

This control record is *ignored* if entered from the paper tape reader or the keyboard.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	PUNCH SYMBOL TABLE	
72 through 80	Not used	

If the principal input device is the 1442 Model 6 or 7 Card Read Punch, sufficient blank cards must be placed between the source program END card and the next monitor control record when stacked job input is being used. In estimating the number of blank cards required, allow one card for each symbol used in the source program. Unnecessary blank cards are passed. (If a nonblank card is read when punching on the 1442 Model 6 or 7, the assembler waits at $PRET with /100F displayed in the ACCUMULATOR.)

If the system configuration is 2501/1442, place blank cards in the 1442 hopper and press START on the 1442 before beginning the assembly.

Note. Do not place nonblank cards in the 1442 Model 5. The punch may be damaged if an attempt is made to punch a hole where a hole exists. An error *is not* detected.

*SAVE SYMBOL TABLE

general function

This control record causes the symbol table generated by this assembly to be saved on disk as a system symbol table. This system symbol table is saved until another assembly with a SAVE SYMBOL TABLE control record causes a new system symbol table to replace the old one. This control record is also used with the SYSTEM SYMBOL TABLE control record to add symbols to the system symbol table.

Note. The SAVE SYMBOL TABLE requires that the assembly be absolute (an ORG statement defining the core load origin must be used in your program). Thus, all symbols in the system symbol table have absolute values.

When the symbol table punched by a PUNCH SYMBOL TABLE control record is included in the system symbol table being generated by this assembly, place the punched EQU cards after the SAVE SYMBOL TABLE control record.

If any assembly errors are detected, or if the symbol table exceeds 100 symbols, the system symbol table is not saved, and an assembler error message is printed.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	SAVE SYMBOL TABLE	
72 through 80	Not used	

*SYSTEM SYMBOL TABLE

general function

This control record causes a previously built system symbol table to be added to the symbol table for this assembly as the assembly begins. This allows you to refer to symbols in the system symbol table without redefining the symbols in your source program. Also, this control record can be used with a SAVE SYMBOL TABLE control record to add symbols from this assembly to the system symbol table.

Note. All symbols in the system symbol table have absolute values.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	SYSTEM SYMBOL TABLE	
72 through 80	Not used	

*LEVEL

general function

This control record specifies the interrupt levels serviced by an ISS and the associated ILS subroutines. This control record is required for the assembly of an ISS subroutine. The interrupt level number is a decimal number in the range 0 through 5. If the device operates on 2 interrupt levels (for example, the 1442 Card Read Punch), one LEVEL control record is required for each interrupt level on which the device operates. The assembler accepts no more than 2 interrupt levels for a device. At least one blank must separate the word LEVEL and the interrupt level number.

If a LEVEL control record is not used when assembling an ISS subroutine, an error message is printed at the end of the assembly.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	LEVELƀn	*n* is an interrupt level number (decimal)
72 through 80	Not used	

*OVERFLOW SECTORS

general function

This control record allows you to specify the number of sectors of working storage to be used by the assembler for symbol table overflow and/or macro processing. When this control record is used, the assembler allocates one more sector than the total number specified. This additional sector is used as a working sector by the assembler.

If more than one OVERFLOW SECTORS control record is used, the last record is used to allocate the overflow sectors.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	OVERFLOW SECTORSƀ n1, n2, n3	*n1* is the number of sectors for symbol table overflow; *n2* is the number of sectors for macro parameter list overflow; *n3* is the number of sectors for temporary macro definition.
72 through 80	Not used	

Note. If any of the number fields are not specified in an OVERFLOW SECTORS control record, the commas within the record cannot be eliminated.

additional field information

OVERFLOW SECTORS. The decimal numbers coded after OVERFLOW SECTORS specify the number of sectors to be allocated for (1) symbol table overflow, *n1,* (2) macro parameter list overflow, *n2,* and (3) temporary macro definition overflow, *n3.*

n1

The number of sectors (*n1*) reserved for symbol table overflow is specified as a decimal number in the range 0 through 32. When the entry is zero or not specified, symbol table overflow is not allowed. If the entry is greater than 32, only 32 sectors are assigned for symbol table overflow. If, during assembly, the symbol table overflow exceeds the number of sectors allocated by the OVERFLOW SECTORS control record, an error message is printed. The approximate maximum number of symbols that can be defined in a program is determined by the size of core storage:

Size of core storage (in decimal words)	*Approximate maximum number of symbols*
4096	3500
8192	4165
16384	6895
32768	12355

n2

The macro processor portion of the assembler uses working storage to contain macro parameter list overflow. The OVERFLOW SECTORS control record specifies the number of sectors (*n2*) to be reserved. If n2 is zero or not specified, a comma must be coded, but macro parameter list overflow is not allowed.

compute largest parameter list size

The size (in words) of the total parameter list storage required for an assembly is the size of the largest parameter list within the assembly. The size of a parameter list (in words) can be estimated by using the following formula:

$$\text{Number of words} = 3+N+\sum_{i=1}^{N} \tfrac{1}{2}(m_i+1)$$

where

N is the number of parameters, including nested macros, within a macro call.
M_i is the number of characters per parameter.

For example, the macro call:
EXPND APHA,BETA,C is computed as 3+3+½(5+1)+½(4+1)+½(1+1)=12 words.

compute n2

If the computed size of the largest parameter list within an assembly does not exceed 100 words, parameter list overflow sectors are not required. Otherwise, the number of sectors (*n2*) required can be computed with the following formula:

$$n2=1/100(x-100)$$

where

x equals the size (in words) of the largest parameter list.

n3

The macro processor portion of the assembler uses working storage to store temporary macro definitions (macros that apply only to the assembly in which they are defined). The OVERFLOW SECTORS control record specifies the number of sectors (*n3*) to be reserved for storing the temporary macros. If n3 is zero or not specified, a comma must be coded, but storage of temporary macro definitions is not allowed.

compute n3

The number of working storage sectors (n3) required for storing temporary macro definitions is calculated as: K/40

where

K is the sum of the number of statements in each temporary macro definition.

*COMMON

general function

This control record allows you to specify the length (in words) of COMMON that is shared by the program being assembled and a FORTRAN program compiled prior to this assembly. The number of words of COMMON used by the FORTRAN program can be obtained from a listing of the program. The use of this control record provides for the saving of COMMON when linking between FORTRAN mainlines and assembler mainlines.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 71	COMMONb̸ nnnnn	*nnnnn* is the number (in decimal) of words of COMMON to be saved between links.
72 through 80	Not used	

*MACLIB

general function

This control record specifies that the macro library is used during assembly. The MACLIB control record is invalid on 4K systems and with both LIST DECK options.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 8	MACLIBb̸	
9 through 13	Macro library name	
14 through 71	Reserved	
72 through 80	Not used	

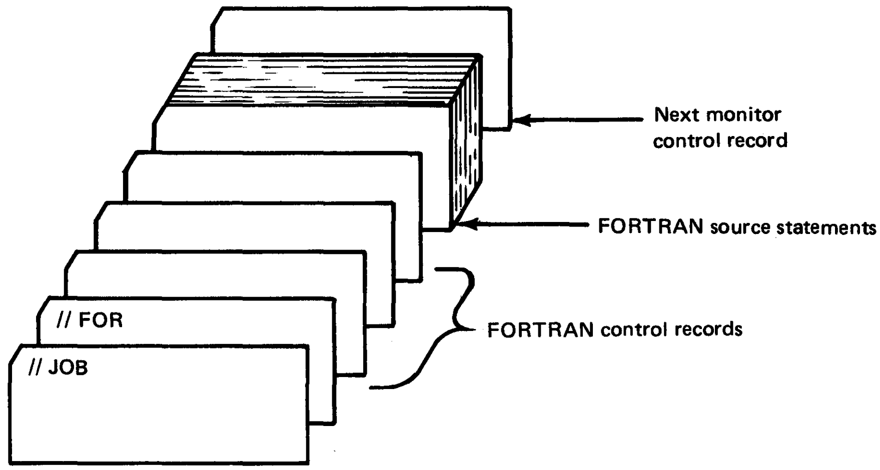
additional field information

Macro library name. This name must be an exact duplicate of the name given to the macro library when it was defined by a STOREDATA or DFILE DUP control record. A MACLIB control record is ignored if an invalid macro library name is specified.

FORTRAN CONTROL RECORDS

functions

FORTRAN control records specify optional operations that affect the FORTRAN compiler and program execution. These control records are placed in the input stream as follows:



FORTRAN control records can be entered in card or paper tape form along with the source program deck or tape, or they can be entered from the console keyboard (see "Entering Jobs from the Console Keyboard" in Chapter 7).

The IOCS, NAME, and ORIGIN control records can be used only with mainline programs; the others can be used with both mainline programs and subprograms.

coding

All FORTRAN control records have the following format:

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	Option	Replace *option* with the keywords for the control record being used.
73 through 80	Not used	

Note. FORTRAN control records are coded in free form; that is, any number of blanks can occur between the characters of the *option*. Remarks are not allowed.

If a FORTRAN control record contains an asterisk in column one, but the *option* is not identical with the format shown for the control record, the asterisk is replaced with a minus sign on the control record listing. The control record in error is ignored; an error does not result, but the specified *option* is not performed. This same action is taken if the specified address is not valid in an ORIGIN control record.

*IOCS

general function

This control record specifies the I/O devices that are used during execution of a FORTRAN core load. Only the devices required should be included. Any number of IOCS control records can be used to specify the required devices.

All I/O devices that are used by FORTRAN subprograms called in a FORTRAN core load must be included on the IOCS control records associated with the mainline FORTRAN program. Assembler language subroutines that are included in a FORTRAN core load can use any of the other I/O device subroutines in addition to those specified on the IOCS control records for the FORTRAN mainline program.

format

Card column	Contents	Explanation
1	*	
2 through 72	IOCS (d, d, . . . , d)	*d* is a valid device name selected from the following list.
73 through 80	Not used	

Names for I/O devices to be used are specified in the IOCS control record. These names are enclosed in parentheses and separated by commas. The devices, their associated IOCS names, and the I/O subroutines called for each device are:

Device	*IOCS device name	Subroutine called
1442 Card Read/Punch, Model 6 or 7	CARD	CARDZ
2501 Card Reader	2501 READER	READZ
1442 Card Punch, Model 5 (1442 Model 6 or 7 if used as a punch only)	1442 PUNCH	PNCHZ
Console printer	TYPEWRITER	TYPEZ
Keyboard	KEYBOARD	WRTYZ
1132 Printer	1132 PRINTER	PRNTZ
1403 Printer	1403 PRINTER	PRNZ
1134/1055 Paper Tape Reader/Punch	PAPER TAPE	PAPTZ
1627 Plotter	PLOTTER	PLOTX
Disk	DISK	DISKZ
Disk (unformatted disk I/O)	UDISK	DISKZ

Note. CARD is used for the 1442 Card Read/Punch, Model 6 or 7, and 1442 PUNCH is used for the 1442 Card Punch, Model 5 (1442 PUNCH can be used for a 1442, Model 6 or 7, if the function is punch only; 1442 PUNCH uses less core storage). CARD and 1442 PUNCH are mutually exclusive; therefore, the use of both of these names in IOCS control records for the same compilation is not allowed.

**IOCS Examples*

```
1   5    10   15   20   25   30   35   40   45   50
*IOCS(CARD,1403 PRINTER,DISK)

*IOCS(PAPER TAPE,1132 PRINTER,DISK)
```

*LIST SOURCE PROGRAM

general function

This control record causes the source program, as it is entered, to be listed on the principal print device.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	LIST SOURCE PROGRAM	
73 through 80	Not used	

*LIST SUBPROGRAM NAMES

general function

This control record causes the names of all subprograms (including subprograms called by EXTERNAL statements) called by the compiled program to be listed on the principal print device.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	LIST SUBPROGRAM NAMES	
73 through 80	Not used	

*LIST SYMBOL TABLE

general function

This control record causes the absolute or relative addresses for the following items to be listed on the principal print device.

- Variable names
- Numbered statements
- Statement functions
- Constants

The addresses are relative unless an ORIGIN control record specifies the core address where the first word of the core load is placed for execution.

A constant in a STOP or PAUSE statement is treated as a hexadecimal number. This hexadecimal number and its decimal equivalent appear in the list of constants. The hexadecimal number is displayed in the ACCUMULATOR when the system waits at $PRET during the execution of the PAUSE or STOP statement.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	LIST SYMBOL TABLE	
73 through 80	Not used	

*LIST ALL

general function

This control record causes the source program, associated subprogram names, and the symbol table to be listed on the principal print device. When this control record is used, the previously described LIST SOURCE PROGRAM, LIST SUBPROGRAM NAMES, and LIST SYMBOL TABLE control records are not required.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	LIST ALL	
73 through 80	Not used	

The FORTRAN sample program in Appendix H is listed by a LIST ALL control record.

*EXTENDED PRECISION

general function

This control record allocates 3 words of core storage for arithmetic values (real and integer) instead of the standard two and generates linkage to the extended precision subprograms.

The FORTRAN compiler normally operates in standard precision; that is, 2 words (a sign, 23 significant bits, and an exponent) of core storage are allocated for each arithmetic value. Through the use of the EXTENDED PRECISION control record, the compiler can be made to yield 31 significant bits by allocating 3 words of core storage for each arithmetic value.

Standard precision, extended precision, and arithmetic subprograms are discussed in the publication *IBM 1130 Subroutine Library,* GC26-5929.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	EXTENDED PRECISION	
73 through 80	Not used	

*ONE WORD INTEGERS

general function

The FORTRAN compiler normally assigns 2 words of core storage for each real and integer value (see the previous discussion of the EXTENDED PRECISION control record). The ONE WORD INTEGERS control record causes all integer values to be assigned one word of core rather than the standard 2 words, or 3 words when an EXTENDED PRECISION control record is used.

An 1130 FORTRAN integer can have any value in the range of $-2^{15}+1$ to $2^{15}-1$. Any value in this range can be contained in one word (16 bits) of core storage; therefore, integer values can contribute rather significantly to inefficient use of core storage because of the extra word allocated for standard or extended precision. Because of this, the use of the ONE WORD INTEGERS control record conserves core.

Note. If this control record is used, the program does not conform to the USASI Basic FORTRAN standard for data storage, and will require modification for use with non-1130 FORTRAN systems.

format

Card column	Contents	Explanation
1	*	Asterisks
2 through 72	ONE WORD INTEGERS	
73 through 80	Not used	

*NAME

general function

This control record causes the specified program name to be printed at the end of the program listing.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	NAMEɓxxxxx	*xxxxx* is the name of the mainline program and is five consecutive characters (including blanks) starting in the first nonblank column after NAME. At least one blank must separate NAME and the mainline program name.
73 through 80	Not used	

** (Header Information)

general function

This control record causes the information specified in columns 3 through 72 to be printed at the top of each page printed during compilation when a 1403 Printer or 1132 Printer is the principal print device. When the first statement of the program is read, the printer skips to a new page (a skip to channel 1), prints the heading, and begins listing the program statements.

format

Card column	Contents	Explanation
1 and 2	**	Asterisks
3 through 72	Any string of characters	
73 through 80	Not used	

*ARITHMETIC TRACE

general function

This control record causes the value of each variable to be printed each time it is changed during program execution. An asterisk immediately precedes each printed value.

Console entry switch 15 must be turned on, and an IOCS control record specifying the console printer, 1132 Printer, or 1403 Printer must be included in the FORTRAN control records. When more than one of these print devices is specified, the fastest device is used for printing the traced values. Tracing is stopped if console entry switch 15 is turned off. This provides for tracing only a part of a program. Tracing can be restarted by turning console entry switch 15 back on.

You can trace selected portions of your program by placing statements that start and stop tracing in the source program. These statements, CALL TSTRT and CALL TSTOP, are placed where needed in the program. In addition to these statements, console entry switch 15 must be on and an IOCS control record specifying a print device and an ARITHMETIC TRACE control record must be included in the FORTRAN control records.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	ARITHMETIC TRACE	
73 through 80	Not used	

*TRANSFER TRACE

general function

This control record causes the values of IF expressions and computed GO TO indexes to be printed during program execution. Two asterisks immediately precede each printed value of an IF statement. Three asterisks immediately precede the value printed for the index of a computed GO TO statement.

Console entry switch 15 must be turned on, and an IOCS control record specifying the console printer, 1132 Printer, or 1403 Printer must be included in the FORTRAN control records. When more than one of these print devices is specified, the fastest device is used for printing the traced values. Tracing is stopped if console entry switch 15 is turned off. This provides for tracing only a part of a program. Tracing can be restarted by turning console entry switch 15 back on.

You can trace selected portions of your program by placing statements that start and stop tracing in the source program. These statements, CALL TSTRT and CALL TSTOP, are placed where needed in the program. In addition to these statements, console entry switch 15 must be on and an IOCS control record specifying a print device and a TRANSFER TRACE control record must be included in the FORTRAN control records.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	TRANSFER TRACE	
73 through 80	Not used	

*ORIGIN

general function

This control record allows you to specify the core address where the core image loader starts loading a program into core for execution. When an ORIGIN control record is used, absolute addresses are printed in the listing that is produced by the compiler. This allows you to see exactly where the program statements and constants are during execution.

format

Card column	Contents	Explanation
1	*	Asterisk
2 through 72	ORIGINbddddd *or* ORIGIN/xxxx	This is the starting core address expressed as a decimal number (ddddd) of 3 to 5 digits or as a 4 hexadecimal number (/xxxx) of 1 to 4 digits preceded by a slash.
73 through 80	Not used	

additional field information

ORIGIN. The origin of a program cannot be specified below the disk I/O subroutine that is used by the core load. The origin is determined by adding decimal 30 to the next higher addressed word above the end of the disk I/O subroutine used by the core load. If the address you specify is an odd number, the system uses the next highest even address as the origin. The following chart lists the lowest possible origins. If an invalid address is specified, the control record is ignored.

Disk I/O subroutine in core	Core load origin	
	Decimal	Hexadecimal
DISKZ	510	/01FE
DISK1	690	/02B2
DISKN	960	/03C0

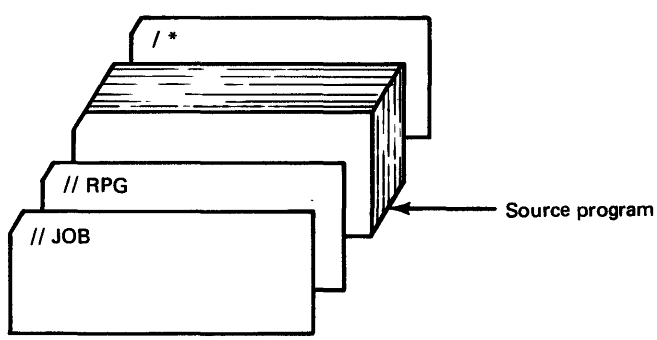
RPG CONTROL CARDS

functions

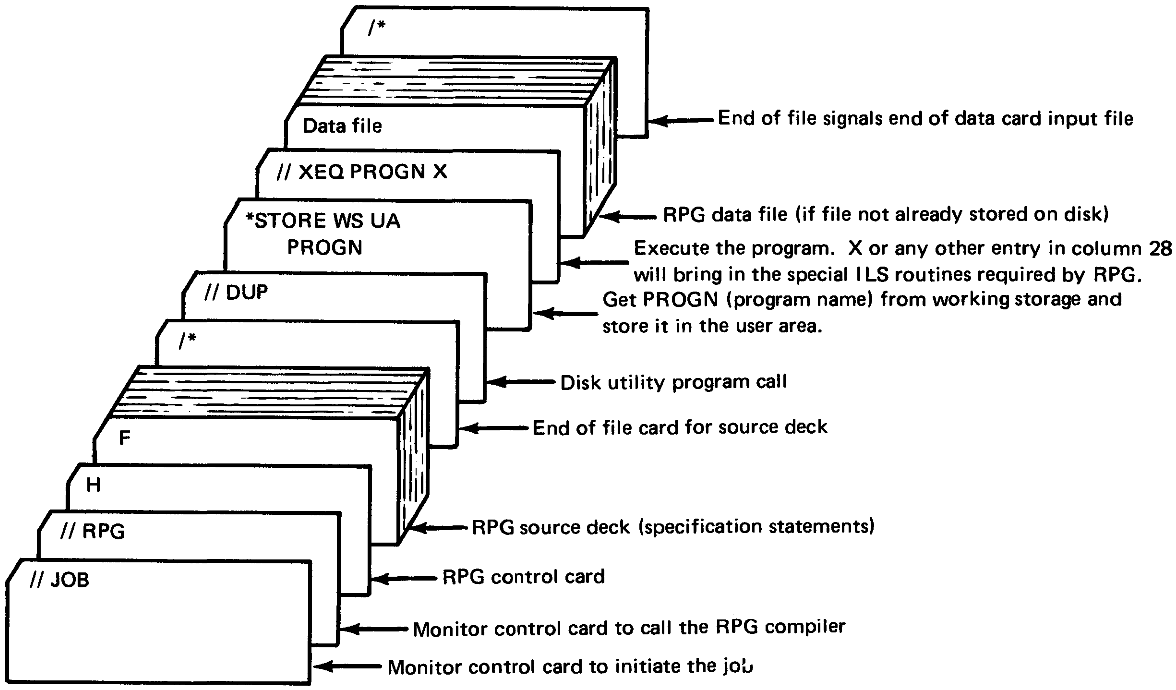
Two RPG control cards specify operations to be performed by the RPG compiler. The first, the RPG control card, acts as a header for the source deck. Information coded in this control card indicates the compiler operations to be performed.

The second control card, the RPG end-of-file control card, is required as the last card of a source program or a data file.

The RPG control cards are placed in the input stream as follows:

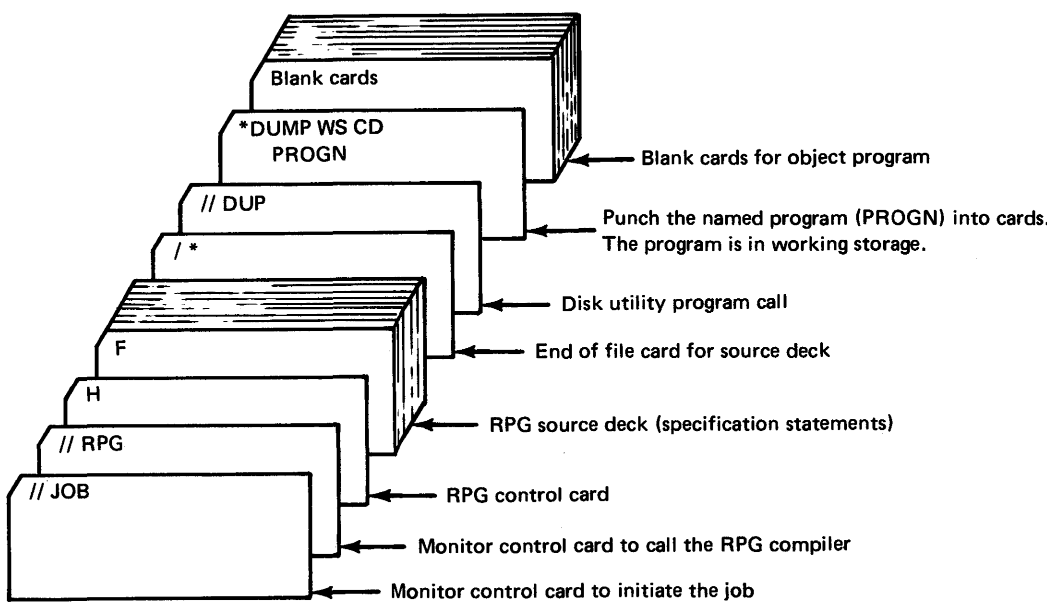


The following illustrates the stacked input required to compile an RPG source program, store the object program in the user area, and execute the object program:

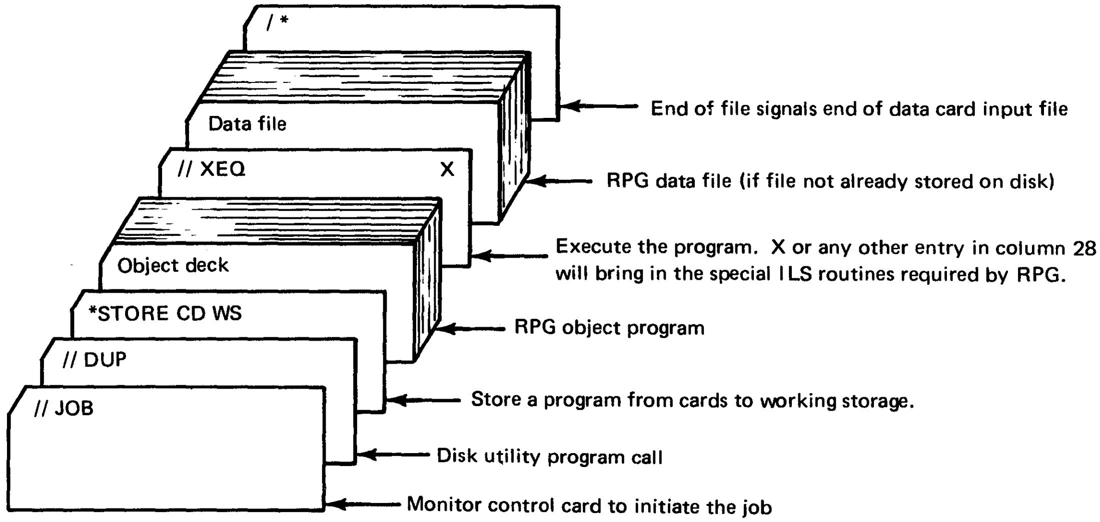


If the // DUP and *STORE records are omitted, the program is executed from working storage; however, the program is not available for future execution because it is not saved.

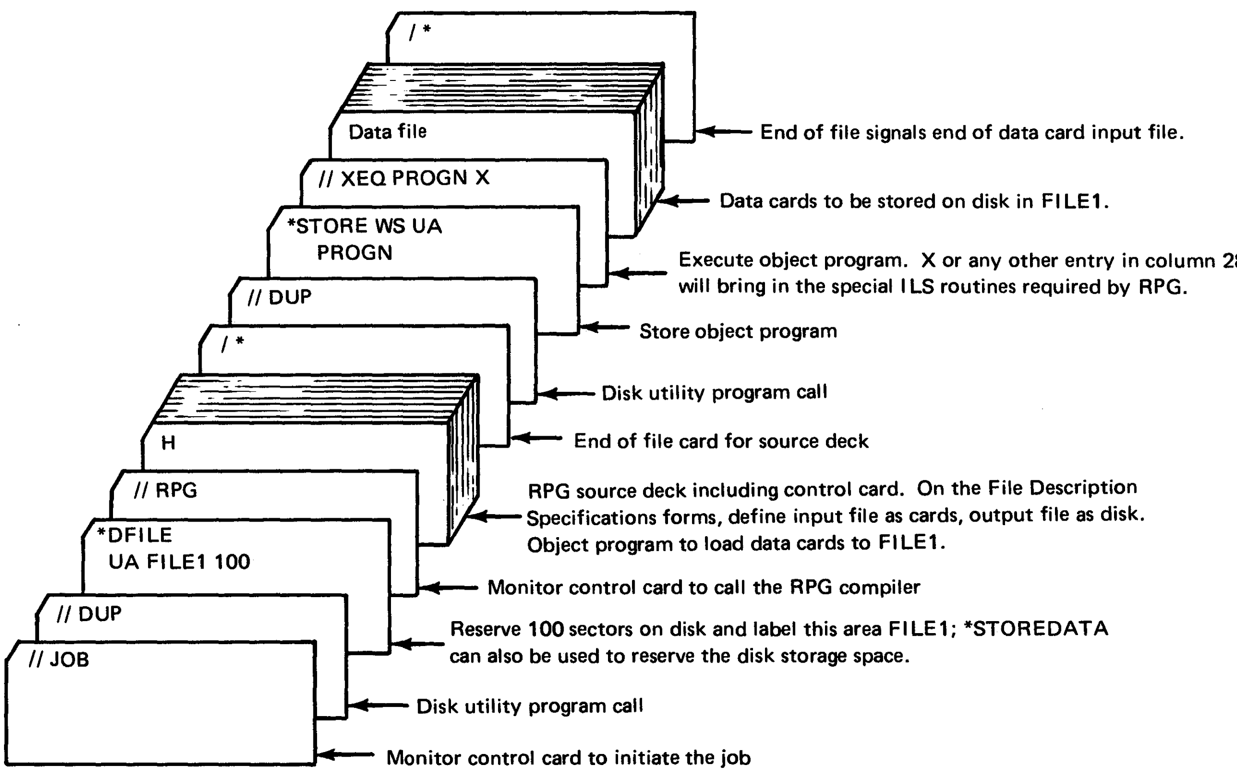
If the program being compiled is not executed often, storing it on cards rather than on disk may be advisable. The following illustrates the stacked input required to compile an RPG program and punch an object deck:



Then, the input stacked required to execute the object program from cards is illustrated by:



Most RPG programs require input data during program execution. This data can be on data cards at execution time or can be stored at any time before execution in a predefined data file on disk. The following illustrates how a data file can be built on disk by an RPG program:



The RPG compiler prints addresses of various routines in the key addresses of object program table. For example, the *close files* routine (located near the end of the mainline program) is included in this table. This routine may require from 2 to 16 additional words (hexadecimal) depending on the type and number of files to be closed. The address of this routine can be helpful when dealing with programs that exceed the available core storage. By adding the number of additional words to the address of the close files routine, the size of the generated mainline program can be determined.

RPG data files may be sequential or indexed-sequential (ISAM). On an ISAM load function, the compiler prints the following information:

- Filename
- Number of sectors required if overflow is not needed
- Number of sectors required if 10 percent overflow is needed

This information can be used to reserve file space for ISAM records. See "Assembler and RPG Disk File Organization and Processing" in Chapter 6 for detailed information about RPG disk data files.

RPG Control Card

general function

This first card of an RPG source program immediately following the RPG monitor control record must be an RPG control card. The information coded in columns 6 and 11 of this card indicate the functions that are to be performed by the RPG compiler. All other entries in the control card are described in the publication *IBM 1130 RPG Language,* GC21-5002.

format

Card column	Contents	Explanation
1 through 5	Described in *IBM 1130 RPG Language*	
6	H	Identifies this card as an RPG control card
7 through 10	Reserved	
11	Blank, B, or D	Blank indicates compilation with a listing of the program. *B* indicates compilation only. *D* indicates a listing only.
12 through 80	Described in *IBM 1130 RPG Language*	

End-of-File Control Card

general function

This control card designates the end of an RPG source program and an RPG data file; therefore, an end-of-file control card must be the last card of an RPG source program and an RPG data file.

format

Card column	Contents	Explanation
1 and 2	/*	
3 through 80	Not used	

Chapter 6. Programming Tips and Techniques

The information in this chapter is planned to help you use the 1130 Disk Monitor System, version 2, more efficiently. The information is presented in the following order:

1. General tips on monitor control and usage
2. Data file processing
3. Tips for the assembler programmer
4. Tips for the FORTRAN programmer
5. RPG object program considerations

TIPS ON MONITOR CONTROL AND USAGE

The tips in this section are of general interest to all programmers of the 1130 DM2 system. These tips include:

- Arranging stacked jobs
- Using temporary job mode
- Using the disk I/O subroutines
- Restoring destroyed cartridges
- Avoiding overprinting
- Using programs and data files more efficiently
- Using LOCALs, NOCALs, and SOCALs
- Reading core maps and file maps
- Reading the transfer vector
- Using SYSUP for changing cartridges during program execution

Stacked Job Input Arrangement

Input to the monitor system includes control records, source programs, object programs, and data that are arranged logically by job. The monitor JOB control record designates the start of a job. You should consider the following when arranging the input for any job:

- Any number of comments (// *) control records can be used before ASM, RPG, FOR, COBOL, DUP, or XEQ monitor control records. Comments control records cannot immediately follow ASM, RPG, FOR, or COBOL control records.

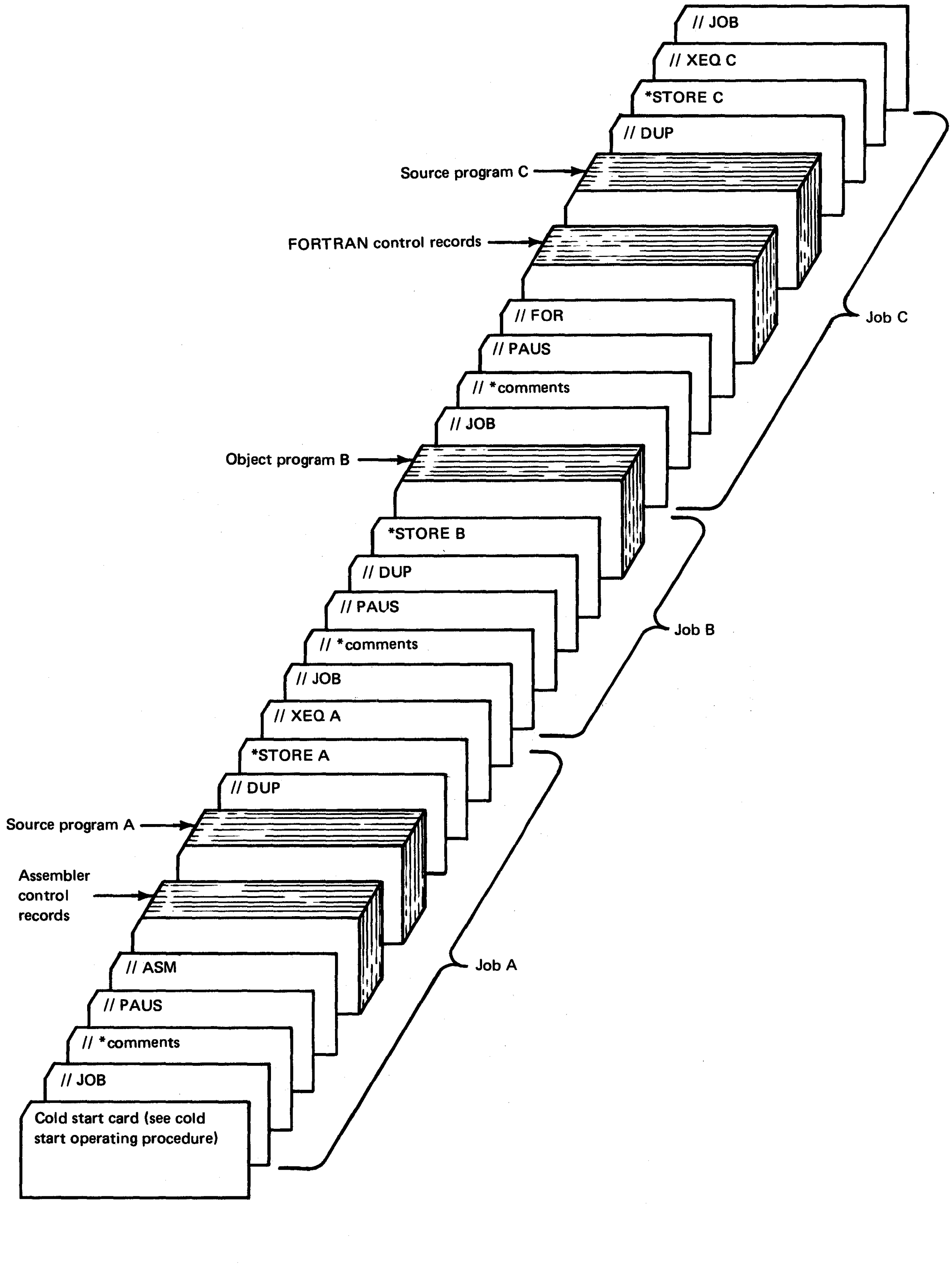
  When an *EQUAT supervisor control record is used after a JOB monitor control record, a comments control record cannot be placed between the JOB record and the EQUAT record. A comments control record cannot be placed between a // DUP control record and the following DUP control record (* . . .).

  When supervisor control records are used after an XEQ or STORECI control record, comments control records cannot be placed between the XEQ or STORECI and the following supervisor control records.

- Any records other than monitor control records that remain after completion of an assembly, compilation, or a subjob (XEQ) are passed until the next monitor control record is read. Also, after a Disk Utility Program (DUP) operation is completed, any records other than monitor control records or other DUP control records are bypassed.

- If an error is detected in an assembly or compilation or during the building of a core load for execution (XEQ), the resulting object program and any program or programs that follow within the current job are not executed. Also, all DUP functions are passed until the next valid ASM, FOR, RPG, or JOB control record is read if an error is detected in an assembly or compilation or during the building of a core load because of a DUP STORECI function.
- If a monitor control record is read by the assembler, by one of the compilers, or during Macro Update Program (MUP) operations, execution of the assembler, compiler, or MUP is ended. The function indicated by the monitor control record is performed.

The following stacked input arrangement assembles or compiles, stores, and executes programs A and C, if source program errors do not occur and if working storage is large enough.


// JOB
// XEQ C
*STORE C
// DUP
Source program C
FORTRAN control records
// FOR
// PAUS
// *comments
// JOB
Job C
Object program B
*STORE B
// DUP
// PAUS
// *comments
// JOB
Job B
// XEQ A
*STORE A
// DUP
Source program A
Assembler control records
// ASM
// PAUS
// *comments
// JOB
Job A
Cold start card (see cold start operating procedure)

If an error occurs in one of the source programs, the DUP *STORE operation is not performed for that program, and all following XEQ requests before the next JOB control records are bypassed. Thus, if the successful completion of one program depends upon the successful completion of the previous one, both programs should be considered as one job and the XEQ control records should not be separated by a JOB record.

How to Use Temporary Job Mode

Temporary job mode (indicated by a T in column 8 of a monitor JOB control record) causes all programs stored in the user area during the temporary job to be deleted automatically when the next JOB control record is processed.

In some cases, the available space in the user area may not be large enough for storage of a newly assembled or compiled program. When this happens, you must use the DUP delete function to clear the user area of old programs, and then store the new program. The necessity for such deletions can be avoided by using temporary mode when running jobs that included programs likely to be replaced at a later time, or that are infrequently used.

Temporary mode is particularly useful when debugging a new program.

Using the Disk I/O Subroutines

All core loads, whether they use disk I/O or not, require one of the 3 disk I/O subroutines. As a minimum, a disk subroutine reads the core load into core and executes CALL EXIT, CALL LINK, CALL DUMP, and/or CALL PDUMP.

uses and how to call

Source programs written in assembler, FORTRAN, RPG, or COBOL can call any of the 3 I/O subroutines; however, only one disk I/O subroutine can be referenced in a given core load. Because of this, all programs and subroutines linked to in a core load must use the same disk I/O subroutine. The subroutine used by a core load is indicated in an XEQ monitor control record or a STORECI DUP control record. (Control records are described in Chapter 5.) Generally, DISKZ is used by FORTRAN, RPG, and COBOL core loads and DISK1 or DISKN by assembler language core loads.

functions

DISKZ is intended for use in an error-free environment, because it does no preoperative error checking. DISKZ is the shortest of the disk subroutines.

DISK1 and DISKN provide more functions than DISKZ. These additional functions include:

- Validity checking of word count and sector addresses
- File protection
- LIBF-type calling sequence
- Validity checking of the function indicator
- Write without readback check option
- Write immediate
- Word count can be on an odd boundary

DISKN provides 2 more functions than those just listed:

- Simultaneous operation of as many as 5 disks
- Faster operation when transferring more than 320 words

More detailed information about the disk I/O subroutines is in the publication *IBM 1130 Subroutine Library*, GC26-5929.

Restoring Destroyed Cartridges

Cartridges containing data and/or programs in the user or fixed area that are difficult to replace can sometimes be restored for use after access to information on the cartridge is destroyed.

use DCIP disk analysis

Use the disk analysis function of the stand-alone utility program DCIP to restore sector addresses if only sector addresses are affected. (DCIP is described in Chapter 9.)

use a system reload

A system reload can be performed if part of the monitor system (except LET, FLET, user and fixed area) is destroyed. Include in the reload the entire monitor system, except the system library.

use DCIP patch

Use the patch function of the stand-alone utility program DCIP to restore individual words that are destroyed on a cartridge.

How to Avoid Overprinting When Using // CPRNT

In order to avoid overprinting when using the monitor CPRNT control record, the FORTRAN programmer should provide for spacing an extra line after the last output statement in a program.

The assembler programmer should provide for spacing after printing following the last output statement in the program.

How to Avoid Overprinting When Linking Between Programs

Overprinting when linking between programs can be avoided by coding your program to space one line before linking to another program. This should be done because the core load builder assumes that a space before printing is not necessary; all monitor programs have a space after print. Overprinting should be avoided because an important core load builder message may not be readable.

Usage of the EJECT Monitor Control Record

An EJECT monitor control record is used during a job to start printing of a new page on the principal printer. For example, comments control records can be placed in a more readable position for the operator if followed by an EJECT control record.

```
1   5    10   15   20   25   30   35   40   45   50
// JOB
    .
    .
    .
// * (MESSAGE TO OPERATOR)
// EJECT
// PAUSE
```

Duplicate Program and Data File Names

Names that are duplicates of IBM-supplied programs should be avoided in DUP store and delete operations. (The names of IBM-supplied programs are in Appendix C.) If a program being stored or deleted has the same name as an IBM program, the results of subsequent operations are not predictable.

Because the DUP store functions check for duplicate names, 2 programs or data files with the same name cannot be stored on one disk. Two programs or data files can, however, have the same name if stored on separate disks. If your system has more than one disk drive, having programs with the same name on more than one disk on the system can cause problems when an attempt is made to execute or delete the named program.

```
1   5    10   15   20   25   30   35   40   45   50
// JOB    1111 2222
   .
   .
// DUP
*STORE              PROG1           1111
   .
   .
// DUP
*STORE              PROG1           2222
   .
   .
   .
// XEQ PROG1
   .
   .
// DUP
*DELETE             PROG1
```

This sequence of control records cause PROG1 on the cartridge labeled 1111 to be executed when you may have wanted PROG1 on 2222 executed. A similar problem can occur in the delete operation. In this example, PROG1 on 1111 is deleted; you may have wanted to delete the program on 2222.

To avoid this problem:

- Assign a unique name to each program and data file.
- If you do not know the contents of a cartridge that is on the system, and the cartridge is not needed for your job, make the drive not ready.

Disadvantages of Storing a Program in DCI Format

Before you decide to convert to and store a program in disk core image (DCI) format, consider the advantages gained in loading time of a DCI program against the following disadvantages.

system maintenance

An important consideration is the effect that system maintenance can have on a DCI program. Subroutines from the IBM-supplied system library that are called by a program are stored with a program in DCI format. If system maintenance changes a subroutine after a DCI program is stored, the subroutine in the system library is changed; however, the copy stored with the DCI program is not. In this case, the DCI program must be deleted and rebuilt (STORECI) after the maintenance modification is made.

size of working storage

If the user or fixed area is expanded after a DCI program is stored, working storage files that are referenced by the DCI program may extend beyond the available working storage during execution. This problem is not recognized until an attempt is made to perform disk I/O operations past the end of the cartridge.

data files not in working storage

Another important consideration concerns DCI programs that reference files that are not placed in working storage during execution. An error occurs if an attempt is made to store in DCI format a program that references a file in the user area, because the location (sector address) of the referenced file may change as a result of program deletions. The DCI program subsequently references such a file by the old sector address. The results are unpredictable.

A similar problem can occur if the DCI program references a file stored in the fixed area, even though the operation is allowed. The file might be deleted and another stored in its place after the DCI program is stored. This problem can be complicated by the fact that not only are sector addresses built into a DCI program, but the logical drive codes are also. In this case, you must make certain that every time the program is executed that all the required disk cartridges are mounted on the same logical drives as when the program was originally stored.

difference in core size

A DCI program can be executed on a system with a configured core size different from the system on which the core load was built, if the size of the core load does not exceed the different core size.

Size Discrepancies in Stored Programs

The disk block count of a program is printed and becomes a part of the LET or FLET entry when the program is stored. When a program is stored from cards to the user or fixed area, the disk block count can be greater than when the same program is stored from working storage. The reason for this discrepancy is that a DSF header is created for each card when a program is stored from cards to disk. Therefore, any 2 headers in the stored file are a maximum of 51 words apart. When the program is stored from working storage, the distance between headers is limited by the disk buffer size, 320 words.

The increased disk block count noted when the program is stored from cards accommodates the expanded size of the file caused by the additional headers.

Dumping and Restoring Data Files

Dumping of important data files to cards is often advisable so that the files can be restored later if the cartridge containing them is destroyed. The DUP DUMPDATA function punches sequence numbers in columns 73 through 80 of the data cards as the file is dumped. The numbers start with one and are incremented by one on each card. Thus, the last sequence number is the actual number of data cards; that is, the *count* field you specify on the STOREDATA control record when restoring the file to the user area or fixed area.

DUMPDATA dumps by *sector* count. Therefore, the control record:

```
1   5    10   15   20   25   30   35   40   45   50
*DUMPDATA   WS  CD  NAMEF  001
```

causes one sector to be dumped to 6 cards; 5 cards of 54 words and one card of 50 words. The last 4 words of card 6 are not used.

STOREDATA stores by *card* count. The record:

```
1   5    10   15   20   25   30   35   40   45   50
*STOREDATA  CD  UA  NAMEF  006
```

causes the contents of the 6 cards, excluding the 4 unused words on card 6, to be stored back in one sector. The 4 unused words exceeding the 320 that can be contained in a sector are truncated. When a STOREDATA follows a DUMPDATA for the same file, truncated words do not cause a problem, because these words do not contain data. However, if the card file being stored by a STOREDATA is produced by a function other than a DUMPDATA, words of the source file may be lost if every word of every card contains data. To prevent the possible loss of data, calculate the card count to be specified in the STOREDATA control record as follows:

1. Use the formula: $\frac{C \times 54}{320} = S$

   where

   C is the actual number of cards; 54 is the number of data words that can be contained in a card; 320 is the number of words that can be contained in a sector, and *S* is the number of sectors required for the file.
2. If this formula produces a remainder that is less than 54 and not zero, add one to the card count to be specified in the STOREDATA control record, and place a blank card at the end of the data deck.

Use of Defined Files

When an *FILES supervisor control record follows a // XEQ monitor control record, the core load builder searches LET and/or FLET for a specified file name. If the name is found, the sector address of the file is inserted in the file table identified by the associated file number specified on the *FILES control record. (A file table is created during program assembly or compilation by the assembler FILE statement or the FORTRAN DEFINE FILE statement, respectively.) If the file name is not found in LET or FLET, the file is defined in working storage.

An *FILES control record after an *STORECI DUP control record is processed in the same way, except that files found in the user area are flagged as invalid.

A suggested way of initially allocating a disk area for a data file in the user area or fixed area is to use the DUP *DFILE function. The number of sectors to be reserved is determined on the basis of the number of records the file is to contain, and the size of each record. Use the following to calculate the number of required sectors for a file:

1. Compute the number (N) of records that can be contained in one sector:

$$N = \frac{320}{L}$$

   where

   L is the length in words of each record in the file. Disregard the remainder, if any.

2. Compute the number of required sectors (S):

$$S = \frac{M}{N}$$

   where

   M is the total number of records in the file.
   N is the number of records computed in Step 1.
   Round the answer to the next higher number if the answer has a remainder. This answer is the sector count that you specify in an *DFILE control record to reserve file space in the user area or fixed area.

Mainline Programs that Use All of Core

Before you write a program that occupies all of core storage, consider that extensive re-writing may be required if IBM-supplied subroutines called by the core load are expanded due to modifications.

The Use of LOCALs

A core load that is too large to fit into core for execution can be executed by specifying as LOCALs some of the subroutines called by the core load. Since a core load that utilizes LOCALs does not execute as fast as it does without LOCALs, keep the following in mind when specifying LOCALs:

- Specify infrequently called subroutines as LOCALs.
- Plan your program so as to minimize the number of times that LOCALs are called into core.
- Keep the number of specified LOCALs to a minimum.

LOCAL-Calls-a-LOCAL

The assembler language programmer can execute core loads in which a LOCAL calls another LOCAL. Any character punched in column 26 of the XEQ control record causes all DSF core loads for that execution to allow LOCALs to call LOCALs. In a series of LOCAL-call-LOCAL subroutines, you must pass the link word (mainline program return address) in all LOCALs (type 4 or 6 subroutines) that are referenced by CALL statements. The return address must be passed in order to return from the last LOCAL to the place from which the first LOCAL was called. Assembler is the only language that allows the return address to be passed. Therefore, LOCAL-calls-a-LOCAL is restricted to assembler language use.

LOCAL and NOCAL Control Record Usage

When using LOCAL and NOCAL control records, keep the following in mind:

- A subroutine cannot be specified as a LOCAL if it calls another subroutine also specified as a LOCAL. For example, if A is a LOCAL subroutine and A calls B and B calls C, neither B nor C can be specified as LOCAL subroutines for the same program. The assembler programmer can avoid this restriction by using the LOCAL-calls-a-LOCAL option discussed in the previous section of this chapter.
- If a subroutine is specified as a LOCAL and SOCALs are employed, the subroutine is m made a LOCAL even though it otherwise would have been included in one of the SOCAL overlays.
- If a subroutine is specified as a LOCAL, it is included in the core image program even if it is not called.
- When using LOCAL control records, the total number of mainlines and subroutines specified cannot exceed:

  $$3M + 2S \leqslant 640$$

  where

  M is the total number of mainlines specified in the LOCAL control records.
  S is the total number of subrouitnes specified in the LOCAL control records.

  If execution is from working storage, the mainline program in working storage is counted as one, although it is not specified on a LOCAL record. This restriction also applies to NOCAL control records.
- Only subroutine types 3, 4, 5, and 6 can be named on LOCAL and NOCAL control records. (A description of subprogram types is included in Appendix I.) Subprogram types 3 and 5 are referenced by LIBF statements, and types 4 and 6 with CALL statements. Types 5 and 6 are ISSs; types 3 and 4 are subprograms.
- Conversion tables, such as EBPA and HOLTB, cannot be used as LOCALs. The conversion tables are listed in Appendix C.
- SCAT1, SCAT2, and SCAT3 cannot be used as LOCALs.

The Use of NOCALs

NOCALs provide a method of including a subroutine in a core load even though the subroutine is not called. The advantages of NOCALs can be illustrated by the following.

manually executed debug subroutines

You can write debugging subroutines, such as a specialized dump subroutine, and include them in a core load as NOCALs. Then during program execution, you can execute the debugging subroutine by manually branching to its entry point.

If an interrupt service subroutine (ISS) for level 5 is made a NOCAL during a core load, you can execute it by pressing PROGRAM STOP; an interrupt on level 5 is made, and PROGRAM START returns execution to the mainline program. A subroutine to monitor execution of a mainline program or to gather statistical information can be designed.

ISS trace subroutine using NOCAL

The following sample trace subroutine for interrupt level 5, ILS05, determines when the contents of a core location are destroyed by being changed to zero. Location /0500 is used in the example. This subroutine is written and stored as subtype zero in the user area. The sample ISS is assembled as level 5 and stored in the user area. The ISS trace subroutine is specified as a NOCAL when the mainline program is executed; the ISS and associated ILS05 are included as a part of the core load. During a WAIT instruction in the mainline program, the console mode switch is turned to INT RUN to cause a level 5 interrupt after execution of each mainline statement. The trace subroutine is entered and, in this example, waits when core location /0500 becomes zero. A dump of the program can be used to determine the conditions that caused the change to zero.

```
Label       Operation   F T     Operands & Remarks
21    25    27    30   32 33    35   40   45   50   55   60   65   70
***************************************************
*                                                 *
*   EXAMPLE OF ILS SUBROUTINE FOR LEVEL 5         *
*   TO ALLOW TRACING DURING INTERRUPT RUN MODE    *
*                                                 *
***************************************************
      ILS   05
ADDR  DC        /0047
*   00=ENT PT REL TO ISS BEG AND 47='ISTV+ISS NO
*   CORE LD BLDR PUTS ACTUAL ENT PT OF ISS IN ADDR
INT   DC        *-*
      STD       TEMP      SAVE ACC-EXT
      STS       RETUR     SAVE STATUS
      STX    1  XR1+1     SAVE XR1
      STX    2  XR2+1     SAVE XR2
      BSI   I   ADDR
RETUR LDS       0         RESTORE STATUS
      LDD       TEMP      RESTORE ACC-EXT
XR1   LDX   L1  *-*       RESTORE XR1
XR2   LDX   L2  *-*       RESTORE XR2
      BOSC  I   INT       EXIT-TURN OFF INT LEVEL
TEMP  BSS   E   2         (ACC-EXT)
      END
***************************************************
*                                                 *
*   EXAMPLE OF ISS TRACE SUBROUTINE               *
*   ENTERED AFTER EACH MAINLINE INSTRUCTION       *
*                                                 *
***************************************************
      ISS   20  TRACY     ISS NO. IS 20
TRACY DC        *-*
      LD    L   LOC       SUSPECTED LOCATION
      BSC       +-        TEST AND WAIT FOR OPERATOR
      WAIT                ACTION WHEN FIRST ZERO
      BSC   I   TRACY     RETURN IF STILL NONZERO
LOC   EQU       /0500
      END
```

Note. Provision must be made to test the device status word for the keyboard/console printer if you want to distinguish between level 5 interrupts initiated by the PROGRAM STOP key and interrupts from INT RUN (see *IBM 1130 Functional Characteristics,* GA26-5881).

The Use of SOCALs

restrictions

A subroutine that is included in one SOCAL overlay must not call a subroutine included in another SOCAL overlay or cause another SOCAL overlay to be loaded into core before execution of the current SOCAL is complete. This restriction is required because the IBM-supplied 1130 subroutines that are used in SOCALs are not re-enterable.

Note that disk I/O is used every time a SOCAL is read into core, thus disk I/O is sometimes entered without your direct knowledge.

When the 1627 Plotter is used by a program, the following subroutines must not be in a SOCAL for that program: EADD, FADD, FMPY, EMPY, XMD, XMDS, and FARC. These must instead be incore subroutines. You can accomplish this by:

1. Dumping these programs to cards or WS
2. Deleting the programs
3. Storing the programs with subtype zero

decreasing execution time

The use of SOCALs increases the length of time for execution of a program. Some of the extra time can be avoided by planning your program so as to minimize the number of times that SOCALs are called into core. Ideally, your program should be written in sections, each employing a single SOCAL; input, computation, and output. Plan input and output carefully so as to separate disk and nondisk operations whenever possible.

Reading a Core Map and a File Map

The core maps described in this section are taken from the sample programs supplied with the monitor system. Sample program listings are in Appendix H. These maps include:

- The execution address of the mainline program
- The names and execution addresses of all subroutines in the core load
- The file allocations

The following is the core map from the assembler sample program (program 2):

assembler core map

```
// XEQ        L
R 41   7908 (HEX) WDS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
 FSQR    0248
LIBF TRANSFER VECTOR
 FARC    069A
 XMDS    067E
 HOLL    062E
 PRTY    05DE
 EBPA    058E
 FADD    04DD
 FDIV    053C
 FLD     0488
 FADDX   04E3
 FMPYX   049E
 FSTO    046C
 FGETP   0452
 NORM    0428
 TYPE0   0312
 EBPRT   02AC
 IFIX    0280
 FLOAT   0230
SYSTEM SUBROUTINES
 ILS04   00C4
 ILS02   00B3
      01FE (HEX) IS THE EXECUTION ADDR
```

Message R41 (not an error message) indicates that /7908 words of core storage are not occupied by the core load. Only one subroutine (FSQR) is called with a CALL statement, but several subroutines are called with LIBF statements. The ILS02 and ILS04 subroutines are required; however, their addresses indicate that they are a part of the resident monitor and not in the core load. The entry point address to the mainline program is /01FE.

The following is the core map from the FORTRAN sample program run on a 4K system (program 1):

FORTRAN core map on 4K system

```
// XEQ         L  2

*LOCAL,FLOAT,FARC,IFIX,PAUSE,HOLEZ

*FILES(103,FILEA)
FILES ALLOCATION
  103 02EA  0001  0ED0 FILEA
  101 C000  0001  0ED0  02EC
  102 C001  0001  0ED0  02EC
STORAGE ALLOCATION
R 40  03BF (HEX) ADDITIONAL CORE REQUIRD
R 43  0124 (HEX) ARITH/FUNC SOCAL WD CNT
R 44  06B2 (HEX) FI/O, I/O SOCAL WD CNT
R 45  02B6 (HEX) DISK FI/O SOCAL WD CNT
R 41  0004 (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 XMDS   09AA SOCAL 1
 EBCTB  0F51 SOCAL 2
 HOLTB  0F15 SOCAL 2
 GETAD  0ED2 SOCAL 2
 NORM   07C0
 FADDX  0955 SOCAL 1
 FSBRX  092C SOCAL 1
 FMPYX  08F8 SOCAL 1
 FDIV   08A6 SOCAL 1
 FSTOX  076C
 FLDX   0788
 SDCOM  0978 SOCAL 3
 SDFX   08E3 SOCAL 3
 SDWRT  0901 SOCAL 3
 SIOFX  09A6 SOCAL 2
 SUBSC  07A2
 SIOI   09AA SOCAL 2
 SCOMP  0983 SOCAL 2
 SWRT   08A2 SOCAL 2
 SRED   08A7 SOCAL 2
 FSTO   0770
 FLD    078C
 PRNTZ  0DF8 SOCAL 2
 CARDZ  0D48 SOCAL 2
 SFIO   09BF SOCAL 2
 SDFIO  0960 SOCAL 3
 HOLEZ  086A LOCAL
 PAUSE  086A LOCAL
 IFIX   086A LOCAL
 FARC   086A LOCAL
 FLOAT  086A LOCAL
SYSTEM SUBROUTINES
 ILS04  00C4
 ILS02  00B3
 ILS01  0F56
 ILS00  0F6F
 FLIPR  0804
      04C1 (HEX) IS THE EXECUTION ADDR
```

The principal difference between the assembler core map and this FORTRAN core map is that the FORTRAN core map includes a file map.

File 103 is equated to a disk data file named FILEA by the *FILES control record. Under FILES ALLOCATION, file 103 is listed with a beginning sector address of /02EA, is one sector in length, and is stored on a cartridge labeled 0ED0. If file 103 had required more than the 2 sectors available in FILEA, the record count would have been reduced to make the file fit in FILEA, and the file map entry would be:

103 /2EA 0002 0ED0 FILEA TRUNCATED

Files 101 and 102 are in working storage and are not defined in the *FILES control record. The last entry for each file indicates whether the file is int he user or fixed area, or in working storage. If the file is in the user or fixed area, this entry is the name of the file (FILEA in this case). If the file is in working storage, the last entry for each file is the sector address of working storage.

The second entry for each file in the user or fixed area is the absolute sector address of the first sector of the file. For files in working storage, the second entry is the address relative to the first sector of working storage. Thus, the absolute sector address of file 101 is /0000 + /02EC; for file 102, /0001 + /02EC.

Note that this program when run on a 4K system requires both LOCALs and SOCALs. The programmer defines the LOCALs in the *LOCAL control record. These subroutines are identified by the term LOCAL in the core map. The core load builder selects the SOCAL subroutines, and these subroutines are identified by the term SOCAL followed by a SOCAL overlay number in the core map. SOCAL option 2 is used for this program because all 3 SOCAL overlay numbers are used. SOCAL option 1 uses SOCAL overlay 1 and 2 only.

Under STORAGE ALLOCATION, message R40 indicates that the core load exceeds the capacity of core storage before SOCALs are employed by /03BF words. Messages R43, R44, and R45 indicate that SOCALs 1, 2, and 3 require /0124, /06B2, and /02B6 words of core, respectively. This information indicates that since SOCAL 2 is much larger than SOCAL 1, more arithmetic and function subprograms can be called at little extra cost in core. Message R41 indicates that after SOCALs are employed, /0004 words of core are not used by this core load.

The following is the core map from the same FORTRAN sample program (program 1), but run on an 8K system:

FORTRAN core map on 8K system

```
// XEQ        L  2

*LCCAL,FLCAT,FARC,IFIX

*FILES(103,FILEA)
FILES ALLCCATICN
  103 02EA  0001  0EDO FILEA
  101 000C  0001  0EDO  02EC
  102 0001  0001  0EDO  02EC
STCRAGE ALLCCATICN
R 41  0C08 (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 EBCTB   12BF
 HCLTB   1283
 GETAD   1240
 XMDS    1224
 HCLEZ   11EE
 PAUSE   11D8
 NORM    11AE
 FADDX   1159
 FSBRX   1130
 FMPYX   10FC
 FDIV    10AA
 FSTOX   1052
 FLDX    106E
 SDCOM   0842
 SDFX    07AD
 SDWRT   07CB
 SICFX   0B26
 SUBSC   1088
 SICI    0B2A
 SCCMP   0B03
 SWRT    0A22
 SRED    0A27
 FSTO    1056
 FLD     1072
 PRNTZ   0F78
 CARDZ   0EC8
 SFIO    0B3F
 SDFIC   082A
 IFIX    1338 LOCAL
 FARC    1338 LCCAL
 FLCAT   1338 LCCAL
SYSTEM SUBRCUTINES
 ILS04   00C4
 ILS02   00B3
 ILS01   1366
 ILS00   137F
 FLIPR   12D2
      04C1 (HEX) IS THE EXECUTION ADDR
```

Note that fewer LOCALs are specified, and that SOCALs are not necessary; the entire program can be contained in 8K core.

The following is the core map from the RPG sample program (problem 3):

RPG core map

```
// XEQ       L            R
R 41   6D16 (HEX) WDS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
 RGERR  0C24
 HLEBC  0A1A
LIBF TRANSFER VECTOR
 RGSI5  11E4
 RGBLK  11AA
 RGEDT  105A
 RGMV2  0FA6
 RGADD  0DDD
 RGSI1  0D80
 RGMV5  0C72
 RGMV3  0D50
 RGCMP  0CFE
 RGMV1  0C6A
 PRNT1  0A9A
 ZIPCO  097A
 CARDO  087C
SYSTEM SUBROUTINES
 ILSX4  1249
 ILSX2  126D
 ILSX1  1286
 ILSX0  12A3
      020F (HEX) IS THE EXECUTION ADDR
```

The information in the RPG core map that is different from the assembler or FORTRAN core maps is that the special ILS subroutines (named with an X, as ILSX4) are used. The special ILS subroutines are required by RPG and are called when any character is punched in column 28 of the // XEQ control record.

Locating FORTRAN Allocation Addresses

Variable, constant, and statement allocation addresses are relative to the loading address of a FORTRAN program if an *ORIGIN control record is not used. The loading address (origin) is determined by adding decimal 30 to the next higher addressed word above the end of the disk I/O subroutine used by the core load. The following chart lists the lowest possible origins, depending on the disk I/O subroutine in core:

Disk I/O subroutine in core	Core load origin	
	Decimal	Hexadecimal
DISKZ	510	/01FE
DISK1	690	/02B2
DISKN	960	/03C0

The absolute addresses of variables, constants, and statements are found by adding their allocation addresses (obtained from a listing) to the loading address.

If an *ORIGIN control record is used, you designate the loading address (not lower than the addresses in the previous chart). In this case, the allocation addresses printed in a listing are absolute addresses.

The variable allocations that follow are taken from the FORTRAN sample program (program 1) in Appendix H.

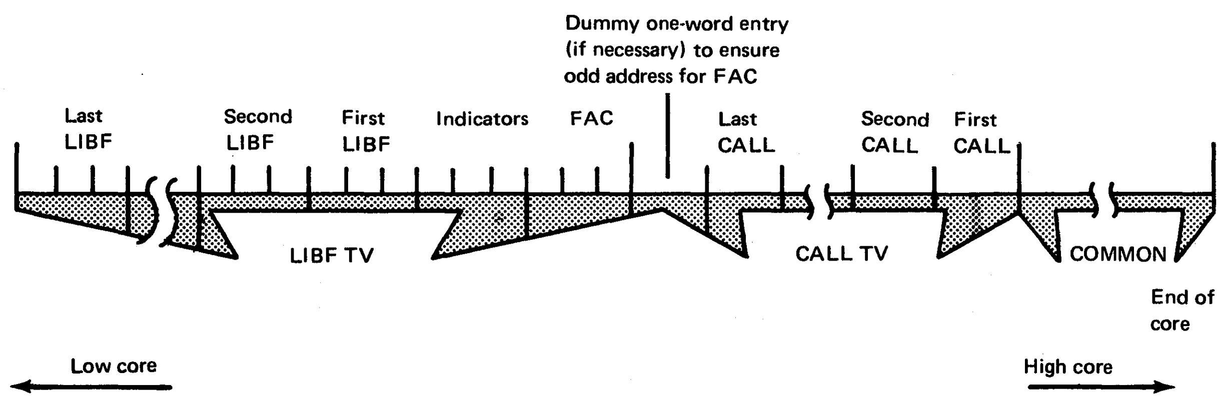
```
VARIABLE ALLOCATIONS
     A(R  )=00DC-0016        X(R  )=00F0-00DE        B(R  )=01EC-00F2
    V3(I  )=01F2             M(I  )=01F3             L(I  )=01F4
    L2(I  )=01F8            N1(I  )=01F9            N (I  )=01FA
     K(I  )=01FE            IK(I  )=01FF            I1(I  )=0200
     D(R  )=01EE            V1(I  )=01F0            V2(I  )=01F1
    M1(I  )=01F5            M2(I  )=01F6            L1(I  )=01F7
     N(I  )=01FB             I(I  )=01FC             J(I  )=01FD
```

The real variable array A is allocated between the loading address + /00DC and the loading address + /0016. Constant and statement allocations are calculated in a similar manner. Notice that the 100-element array A requires 200 core locations (2 words per element). Because all FORTRAN arrays are allocated in reverse order, A (1) is assigned the two relative addresses /00DC and /00DD, A (2) begins at /00DA, and A (3) begins at /00D8.

The relocation factor (the actual core address of the first word) of a FORTRAN subprogram is obtained by subtracting the relative entry point address (from the subprogram compilation listing) from the actual entry point address (in the core map).

Reading the Transfer Vector

The contents of the transfer vector are determined from a core dump by starting at the high end of core and marking off words backwards as illustrated by the following:



Use the following steps to locate contents of the transfer vector:

1. Mark off the number of words in COMMON, if any. For a FORTRAN program, the word count of COMMON is obtained from a program listing. For an assembler program, the word count of COMMON is as you specified in an *COMMON assembler control record.
2. Mark off one word for each CALL-type subroutine, if any. Each word is filled, during building of a core load, with the entry point address of the called subroutine. The subroutines called by a program are listed in a core map and file map.
3. If the last CALL entry is an odd address, mark off an extra word to ensure an odd address beginning for the FAC (real number pseudo accumulator).
4. Mark off the next 3 words for the FAC, which is always present in the transfer vector.
5. Mark off the next 3 words for the indicators. These indicators are always present and are used by various subroutines to indicate overflow, underflow, and divide check.
6. Mark off 3 words for each LIBF-type subroutine. Word one (with the lowest address) contains the return address address. Word 2 always contains /4C00, and word 3 contains the entry point address of the called subroutine. The subroutines called by LIBFs in a program are listed in a core map.

Note. Transfer vector entries contain entry point addresses to special LOCAL/SOCAL linkage if the called subroutines are designated as LOCALs or SOCALs (see "Construction of a Core Load" in Chapter 3).

SYSUP

The system update (SYSUP) mainline program in the system library allows you to change disk cartridges during the processing of a job. SYSUP *must be* called when cartridges are changed. Code your program to call SYSUP immediately after mounting the new cartridges.

This program updates DCOM on the master cartridge (logical drive 0) with the IDs and DCOM information from all satellite cartridges mounted on the system and that are specified in the special SYSUP calling sequence.

The following is an example of the assembler language SYSUP calling sequence:

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–70)
	LD			CHNG
	WAIT			(Change Cartridges)
				(Press PROGRAM START)
	CALL			SYSUP CALL DCOM UPDATE
	DC			LIST
	•			
	•			
	•			
CHNG	DC			/nnnn IN ACCUMULATOR AT WAIT
LIST	DC			/aaaa CART ID OF LOGICAL Ø
	DC			/bbbb CART ID OF LOGICAL 1
	DC			/cccc CART ID OF LOGICAL 2
	DC			/dddd CART ID OF LOGICAL 3
	DC			/eeee CART ID OF LOGICAL 4

Continuation of the job must be delayed until any newly mounted cartridges are ready. The assembler WAIT statement and the FORTRAN PAUSE statement provide the necessary delay.

The IDs of the cartridges being used must be specified. If zero is specified for the master cartridge (logical drive 0), the master cartridge for the current job is assumed. When less than 5 cartridges are used, specify the IDs for the cartridges to be used and an ID of zero to indicate to SYSUP that all cartridges have been specified. If, for example, 3 cartridges are used for a SYSUP operation, the cartridge ID list is coded as follows:

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–70)
LIST	DC			/ØØØØ (Assume master cartridge)
	DC			/1111 (Cart ID of logical 1)
	DC			/2222 (Cart ID of logical 2)
	DC			/ØØØØ (Indicates end of list)

The FORTRAN calling sequence for SYSUP is:

```
      PAUSE 1234
                      (Change cartridges)
                      (Press PROGRAM START)
      CALL SYSUP(a)
```

where

a indicates the last item in an array that contains the IDs of the cartridges being used for the SYSUP operation. For example:

CALL SYSUP (K(5))

K is a one-word integer array. Because FORTRAN arrays are stored in reverse order, the first item read by SYSUP is the last item K(5) stored in the array. Thus, K(5) is the entry for logical drive 0, the master cartridge. This item in the array can contain zero, in which case, the master cartridge defined for the current job is assumed.

The array cannot be longer than 5 words, but it can be shorter. If less than 5 words are used, the first item K(1) placed in the array must be zero to indicate to SYSUP that all cartridges have been specified. For example, a 3-cartridge FORTRAN array is specified as (K(4)) with K(1) containing zero.

After execution of SYSUP is completed, a list of the cartridges is printed. Error messages printed during SYSUP operation are included in Appendix A.

Reeling

Reeling is the process of continuing a long data file from one cartridge to other cartridges and is done with SYSUP and program linking. This operation might be performed as follows.

on a single drive system

Suppose your system has only one disk drive, the internal disk in an 1131 CPU, and you want to sequentially process a long data file that does not fit on one cartridge. The first part of the file can be defined on one cartridge and the second part on another. The program that accesses this file can be written as 2 parts and linked together. The first part processes the first part of the data file, and the second part of the program processes the rest of the data file.

Assume the program is written in FORTRAN, and the termination of the first link consists of a PAUSE (to allow for mounting the second cartridge in place of the first), followed by CALL SYSUP and CALL LINK to the second part of the program. When SYSUP is called, DCOM and COMMA are updated on the second cartridge.

```
1   5    10   15   20   25   30   35   40   45   50   55   60   65      72
      WRITE (3,4Ø)
 4Ø   FORMAT (4ØHØLINK NO. 1 EXECUTED. CHANGE CARTRIDGES.////)
      PAUSE 1111
      CALL SYSUP (L(2))
      CALL LINK (LINK2)
```

The only constraint is that the second cartridge must be a system cartridge. If the FORTRAN compiler is not on the second cartridge, the second part of the program can be compiled on the first cartridge, dumped to cards, and stored on the second cartridge. Sample program 5 in Appendix H illustrates how this is accomplished. For this sample program, both cartridges are system cartridges, both contain a fixed area, but only cartridge 0ED0 includes the FORTRAN compiler. The second part of the program (LINK2) is compiled on the first cartridge, dumped to cards, and stored on cartridge 0ED4 that contains the second part of the data file.

One-word integers are specified for both parts of the program. Thus, the 2-word array referenced in LINK1 contains a zero in L(1), and the second cartridge ID in L(2). Because FORTRAN arrays are stored in reverse order, SYSUP first reads L(2) that identifies the new cartridge on the system and L(1) that indicates no more cartridges.

Another method of using SYSUP that is suitable to any FORTRAN precision is to call an assembler language subroutine, with undefined precision, that calls SYSUP.

on a multidrive system

Sample program 6 in Appendix H illustrates sequential file processing with 2 cartridges and 2 disk drives. If your system has more than one disk drive, you can avoid the SYSUP *and* CALL LINK sequence of sample program 5 by naming both cartridges on the // JOB control record. As in the description of program 5, you must write your program to process the 2 portions of the data file separately, even though they may have the same name. In the case of duplicate names, the *FILES control record can name the 2 files, both with the same name but with different cartridge IDs.

All files referenced in a given core load must be stored in the user or fixed area when the core load is built. This applies to *FILES references and assembler DSA statements alike. If you desire to, you can divide your program into links, each with its own associated file.

reeling in general

If sufficient drives are not simultaneously available for all cartridges involved to be specified, a reeling method must be used. Any cartridge that contains a data file that is named in an *FILES control record must be on the system at the time the *FILES control record is processed after either a // XEQ or *STORECI control record. Similarly, a DCI program that accesses files in a fixed area must be executed with the same cartridges on the same drives as when the program was built.

For example, if sample program 5 in Appendix H is stored in DCI format with cartridge 0ED0 on logical drive 0 and cartridge 0ED4 on logical drive 1, these cartridges must be on the same logical drives each time the program is executed.

These requirements are due to the fact that the core load builder assigns absolute sector addresses, including logical drive codes, for files in the user or fixed area as a core load is built.

DATA FILE PROCESSING

This section describes disk data file organization and processing as follows:

- FORTRAN formatted and unformatted I/O
- Assembler and RPG sequential and indexed sequential access method (ISAM) files

File organization includes defining the required disk space for a new file, and how data is placed in the file. File processing includes how information in files is used and modified.

FORTRAN Disk File Organization and Processing

The FORTRAN READ and WRITE statements call disk I/O subroutines to access disk data files. The disk files are organized sequentially like magnetic tape files, except that random access is possible. This analogy to magnetic tape files is helpful in understanding the processing of the file records. Data conversion is not possible with FORTRAN I/O. The terms formatted and unformatted refer only to the organization of records within files.

The logical unit numbers and maximum record sizes that are used in FORTRAN READ and WRITE statements are listed in Figure 6-1. Avoid the use of the actual logical unit numbers in READ and WRITE statements; the use of integer variables provides for easier program modification.

Logical unit number	Device	Kind of transmission	Record size allowed
1	Console Printer	Output only	120
2	1442 Card Read/ Punch	Input/output	80
3	1132 Printer	Output only	1 carriage control + 120
4	1134/1055 Paper Tape Reader Punch	Input/output	120, plus max. of 80 case shifts for PTTC/8 code, plus NL code
5	1403 Printer	Output only	1 carriage control + 120
6	Keyboard	Input only	80
7	1627 Plotter	Output only	120
8	2501 Card Reader	Input only	80
9	1442 Card Punch	Output only	80
10	UDISK	Unformatted input/output without data conversion	320*

*Unformatted disk I/O comprises 320 word records (including a 2-word header). The first word of the header must contain the count of the physical record within the logical record (see example following). The second word of the header must contain the number of effective words in the individual physical record. The second word of the header of the last physical record within a logical record must have the sign bit (-) on. Unformatted disk characters are stored in as they appear in core storage.

Example:

```
DIMENSION A (400)      800 words
WRITE (10) A
```

Physical records (maximum record length 320 words due to disk sector size)

Logical record (total number of words to be written)

1	318	DATA WORDS
2	318	DATA WORDS
3	-164	DATA WORDS

164 and sign bit (/80A4). *Not* /FF5C.

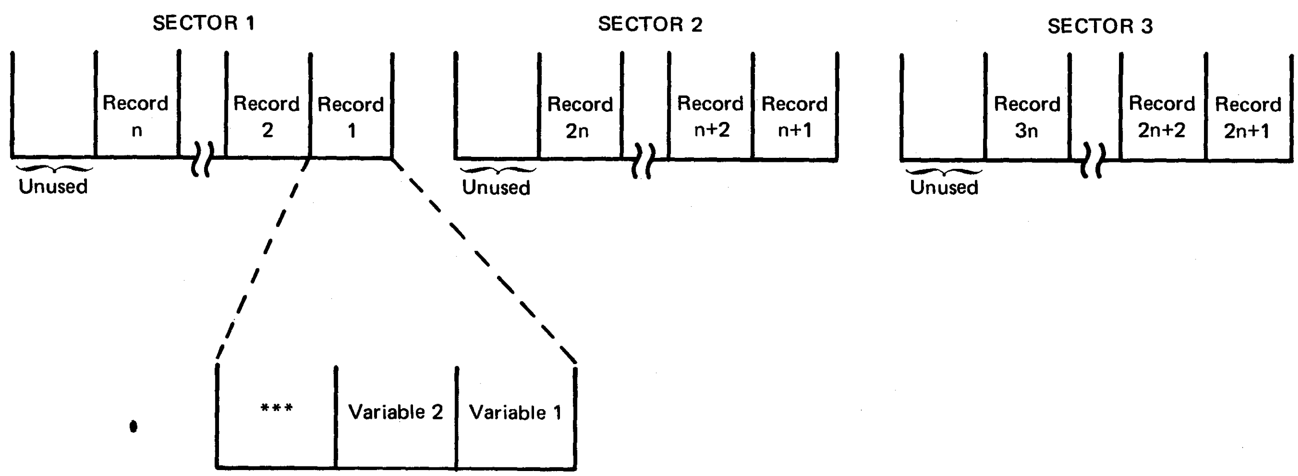
An end-of-file record occupies one sector. Word one of the header must be 1 and word two must be a negative zero (/8000).

Figure 6-1. FORTRAN I/O logical units and record sizes

Formatted FORTRAN I/O Statements

A formatted disk file is created by a FORTRAN DEFINE FILE statement. The file is assigned to working storage unless the file number is equated to an existing file in the user area or fixed area by an *FILES supervisor control record (see "Use of Defined Files" in this chapter). The DEFINE FILE statement specifies the number of records in the file and the record length. In analogous magnetic tape terminology, a formatted file contains fixed length records with a maximum record length of 320 words.

File records are written backwards in the physical sectors; the first record begins at the end of the first sector. Records are filled backwards, with an exact core image of each variable written adjacent to the previously written record. The following illustrates how sectors and records are filled.



If writing of variables specified in a WRITE statement exceeds the record size, writing continues into the next record until the variable list is exhausted. However, if the total size of the file is exceeded because of data exceeding the defined record size, the I/O operation halts with /F101 displayed in the ACCUMULATOR.

formatted data file example

This example assumes a FORTRAN program with the following specification statements:

```
      DEFINE FILE 1(100,4,U,KK)
      DIMENSION R(5),I(5)
      DATA R/1.0,2.0,3.0,4.0,5.0/,I/1,2,3,4,5/
```

For this example, file 1 is equated to a 2-sector file named DATA1 (in the user area or fixed are) by the following *FILES control record:

```
*FILES(1,DATA1)
```

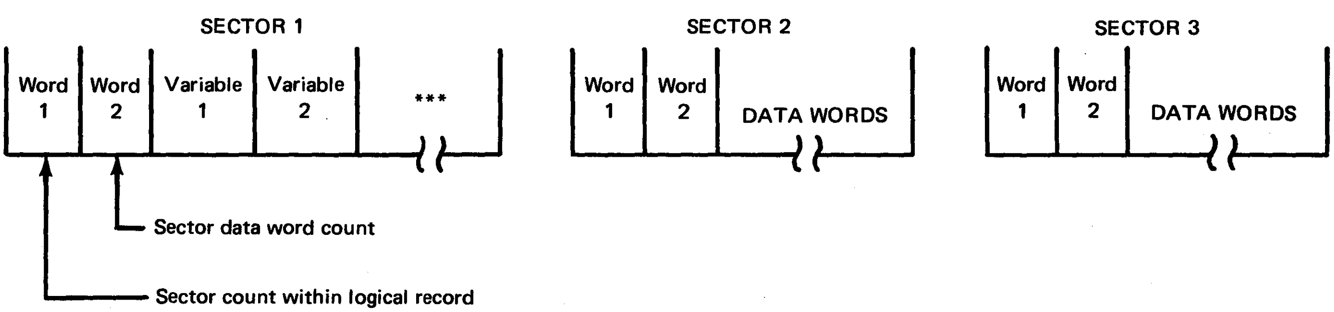
The following shows the contents of the first 2 records of DATA1 after each of the WRITE statements under "I/O executed" is executed. (Assume that the words of DATA1 contained FFFF before execution. XXXX entries indicate unreferenced FORTRAN fill words.)

Precision specified	I/O statements executed	Record 2 of DATA1	Record 1 of DATA1
*ONE WORD INTEGERS	DO 5 J = 1,2 5 WRITE (1'J)I(J)	FFFF FFFF FFFF 0002	FFFF FFFF FFFF 0001
*ONE WORD INTEGERS	DO 5 J = 1,2 5 WRITE (1'J)I(J),R(J),I(J)	0002 4000 0082 0002	0001 4000 0081 0001
*ONE WORD INTEGERS	WRITE (1'1)(I(J),J=1,5)	FFFF FFFF FFFF 0005	0004 0003 0002 0001
None	DO 5 J = 1,2 5 WRITE (1'J)I(J)	FFFF FFFF 0002 XXXX	FFFF FFFF 0001 XXXX
*EXTENDED PRECISION	DO 5 J = 1,2 5 WRITE (1'J)I(J)	FFFF 0002 XXXX XXXX	FFFF 0001 XXXX XXXX
*EXTENDED PRECISION	DO 5 J = 1,2 5 WRITE (1'J)R(J)	FFFF 0082 4000 0000	FFFF 0081 4000 0000
*EXTENDED PRECISION *ONE WORD INTEGERS	WRITE (1'1)I(1),R(1),I(2)	FFFF FFFF FFFF 0002	0081 4000 0000 0001

Unformatted FORTRAN I/O Statements

FORTRAN I/O subroutines can be used for unformatted disk I/O; an analogy to magnetic tape files is that unformatted files contain variable length records. A data file for unformatted I/O must be named $$$$$ and can reside in either the user area or fixed area (see "Initializing $$$$$ Data Files for Use With FORTRAN Unformatted I/O" in this chapter).

The logical record length is determined by the size or the object code of the I/O-statement variable list and is limited only by the total file size. If the length of a record exceeds 318 words, it is segmented to fit into consecutive sectors. Every sector begins with a 2-word header. Word 1 contains the relative sector number within that logical record, and word 2 is the count of the data words following the header. The following illustrates how unformatted sectors are filled:



The last sector of a logical record has a sign bit set in the second word of the header. The remaining words of the last sector are not used. Therefore, an unformatted WRITE statement containing a single one-word integer variable uses only three words of each sector; the 2-word header and the data word.

The FORTRAN I/O statements BACKSPACE, REWIND, and END FILE statements are used only with unformatted disk files. These statements provide a further simulation of magnetic tape file processing, and position the I/O pointer to the correct logical record within a file.

Initializing $$$$$ Data Files for Use With FORTRAN Unformatted I/O

You must define in the user area or fixed area a data file with the name $$$$$ prior to executing a FORTRAN mainline program or subroutine that uses unformatted I/O. One file can be defined on each cartridge; however, only one $$$$$ file can be referenced in any one job.

The following example shows the control records for defining a $$$$$ file on a satellite cartridge and executing the program ML1 that uses unformatted I/O, where:

- The satellite cartridge ID is 1004
- The system cartridge ID is 1001
- A data file of 100 sectors is defined

```
1   5    10   15   20   25   30   35   40   45   50
// JOB   1ØØ1      1ØØ4
// DUP
*DEFINE FIXED AREA        14        1ØØ4
*STOREDATA  WS  FX  $$$$$ 1ØØ11ØØ1  1ØØ4
// JOB   1ØØ1      1ØØ4
// XEQ ML1
```

Note that an *FILES control record defining the $$$$$ file is not required after the XEQ control record.

Sample program 4 in Appendix H uses unformatted I/O and END FILE, BACKSPACE, and REWIND statements. The program writes 3 logical records of different lengths to a $$$$$ data file. Each logical record begins on a sector boundary and extends into additional sectors as required.

After the completion of each WRITE statement (of records A, B, and C), a pointer is moved to the beginning of the next logical record. In the case of the END FILE statement, the pointer is similarly positioned beyond the record generated by END FILE. The second BACKSPACE statement moves the pointer to the beginning of record C, which is subsequently read into area F.

The REWIND statement sets the pointer to logical record A, then a READ statement with no area specified advances the pointer to record B. Only the first half of B is read into area E, since the record lengths are in the ratio 2:1.

Assembler and RPG Disk File Organization and Processing

The disk I/O subroutines supplied with Disk Monitor 2, direct access, sequential access, and indexed sequential access method (ISAM), are used by both assembler and RPG language programmers. The key to the use of the disk I/O subroutines is an understanding of the basic principles of disk file organization and processing.

File Organization

File organization is the method of arranging data records on a direct access storage device; that is, building the file. Two types of file organization are available with DM2; sequential and indexed sequential (ISAM).

sequential file organization

A sequentially organized file is one in which records are placed on the disk in the same order they are read in, one after another. That is, record 6 cannot be written until record 5 is written, record 5 until record 4. Sequential files can be processed sequentially or randomly.

indexed sequential (ISAM) file organization

An indexed sequential file is one in which records are placed on the disk in ascending sequence by a record key. The record key can be a part number, man number, or any other identifying information that is present in the records in the file. In addition, an indexed sequential file uses an index table to indicate to the processing program the general location of desired records. Each index entry contains a cylinder address and the highest record key on that cylinder. For cylinders that have overflowed, the index also contains the overflow sector address and the key of the first sector overflowed from that cylinder (see the descriptions of overflow sectors and areas under "Indexed Sequential Access Method Files" and "Contents of an ISAM File" later in this chapter).

Index tables are analogous to the index card file in a library. If you know the title of a book (the record key), you can look in the card file (index table) until you find the card (index entry) for that book. On the card is a number (cylinder address) where the book (record) is located. You go to the shelf and find (seek) the number (cylinder address) you are looking for. Now you can search for the particular book (record) by title (record key).

Records in an indexed sequentially organized file can be processed sequentially or randomly.

File Processing

File processing is the method of retrieving data records from a file; that is, using the file. Four methods of file processing are available with DM2.

- Sequential processing of sequentially organized files
- Random processing of sequentially organized files
- Sequential processing of indexed sequential (ISAM) files
- Random processing of indexed sequential (ISAM) files

sequential processing of sequential files

When sequentially processing sequential files, all records in the file are processed in the order of the file starting with the first physical record in the file.

random processing of sequential files

When sequential files are randomly processed, the sequence of record processing is not related to the physical sequence of the records in the file. To find a record in a sequentially organized file, your program must specify the record number. The record number indicates the relative position (sequential location) of the record in the file. The disk I/O subroutine calculates the sector address from the record number and reads the proper record.

sequential processing of ISAM files

When sequentially processing ISAM files, all records in the file are available in a sequence determined by the record key. Processing can start at the beginning of the file or at any point within the file.

random processing of ISAM files

To find a random record in an ISAM file, code your program to search the index table using the record's key. The matching index entry points to the cylinder that contains the record. The indicated cylinder is then searched for the desired record; the match is made by record key. This kind of processing can be called processing in a random sequence with record keys.

Calculating Sequentially Organized and ISAM File Sizes

You initially define a file on a disk with the DUP *DFILE or *STOREDATA function. These functions set aside a specified number of sectors for the file, and enter the file name in LET or FLET. This file name that you assign to the file must be used in all future references to the file.

Sequentially Organized Files

The number of sectors required for a file depends on the size of records and the number of records. The records are fixed in length and can be defined as any size between one word (2 characters) and 320 words (640 characters). Records cannot be extended across sector boundaries; thus, a 320 word record (one sector) and a 161 word record each require one sector of disk space. Careful planning is required in calculating optimum record size for your file.

1. Compute the number of words (L) in a record:

   $$L = \frac{C}{2}$$

   where

   C is the record size in characters. Round the answer to the next higher number if the answer has a remainder.

2. Compute the number of records (*N*) that can be contained in one sector:

   $$N = \frac{320}{L}$$

   where

   L is the length in words of each record computed in Step 1. Disregard the remainder, if any. 320 is the number of words in a sector.

3. Compute the number of required sectors (S):

   $$S = \frac{R+1}{N}$$

   where

   R is the number of records in the file, and N is the number of records per sector computed in Step 2. Round the answer to the next higher number if the answer has a remainder. This answer is the sector count that you specify in an *DFILE or *STOREDATA control record to reserve file space in the user area or fixed area.

To change record sizes or add records to a sequential file, the file must be rebuilt. If a revised file requires additional sectors, it must be redefined and rebuilt. A sequentially organized file is built using the sequential access routine. A sequential file can be processed by either the sequential access subroutine or the direct access subroutine. These subroutines are described in the publication *IBM 1130 Subroutine Library*, GC26-5929.

Indexed Sequential Access Method Files

The number of sectors (*S*) required for an ISAM file is computed by the following formula:

$$S = P + I + O + F$$

where

P is the number of prime data sectors, *I* is the number of index sectors, *O* is the number of overflow sectors, and *F* is always one sector for the file label.

compute prime data sectors

The number of prime data sectors (*P*) is computed as follows:

$$P = \frac{R+N-1}{N}$$

where

R is the approximate number of records in the file, and *N* is the number of records per sector. Disregard the remainder, if any. The number of records (*N*) is computed by:

$$N = \frac{320}{L+2}$$

where

L is the length in words of each record. The maximum record length in words is 318; records cannot cross sector boundaries.

compute index sectors

The number of index sectors (*I*) is computed as follows:

$$I = \frac{C+E-1}{E}$$

where

C is the number of prime data cylinders, and *E* is the number of index entries per sector. Disregard the remainder, if any. The number of prime data cylinders is computed as follows:

$$C = \frac{P+7}{8}$$

where

P is the number of prime data sectors. Disregard the remainder, if any. The number of index entries (*E*) per sector is computed by:

$$E = \frac{320}{X} \quad \text{(disregard any remainder)}$$

where

X is the index entry size computed by:

$$X = 2K+3$$

where

K is the key length in words; maximum 25 words (50 characters). If the length of the key in characters is an odd number, add one when calculating the number or words; that is, 49 characters require 25 words.

overflow sectors

You decide on the number of sectors to be provided for overflow before the file must be rebuilt. This overflow area is automatically assigned to start at the sector following the last sector of prime data. This assignment is done by the ISAM load (close) subroutine.

file label

When computing file size, always add one sector for the file label.

If you wish, an assembler language program can be used to perform the preceding calculations. You need know only the index entry size (*X*) as previously discussed, the length of a record in words, the approximate number of records in the file, and an estimate of the number of sectors of overflow area needed. A program to calculate all values previously discussed is included as sample program 7 in Appendix H. The values calculated by the program or by you are required as entries in the disk file information (DFI) tables for the ISAM subroutines. An indexed sequential file is built using the ISAM load subroutine, expanded using the ISAM add subroutine and processed by either the ISAM sequential or ISAM random subroutine. These subroutines are described in the publication, *IBM 1130 Subroutine Library,* GC26-5929.

Contents of an ISAM File

An indexed sequential access method (ISAM) file is composed of:

- File label
- Index
- Prime data area
- Overflow area

The relative position of these components within the ISAM file is:

File label	Index	Prime data area	Overflow area

ISAM file label

The first sector of any ISAM file is the file label. This label contains information required by the ISAM subroutines for processing the file. The file label is built by the ISAM load function, updated by ISAM add, and used by ISAM random and sequential subroutines. All label operations are performed automatically by the ISAM subroutines. The only file label operation that you perform is to reserve one sector for the label when the file is initially defined.

The format of an ISAM label is:

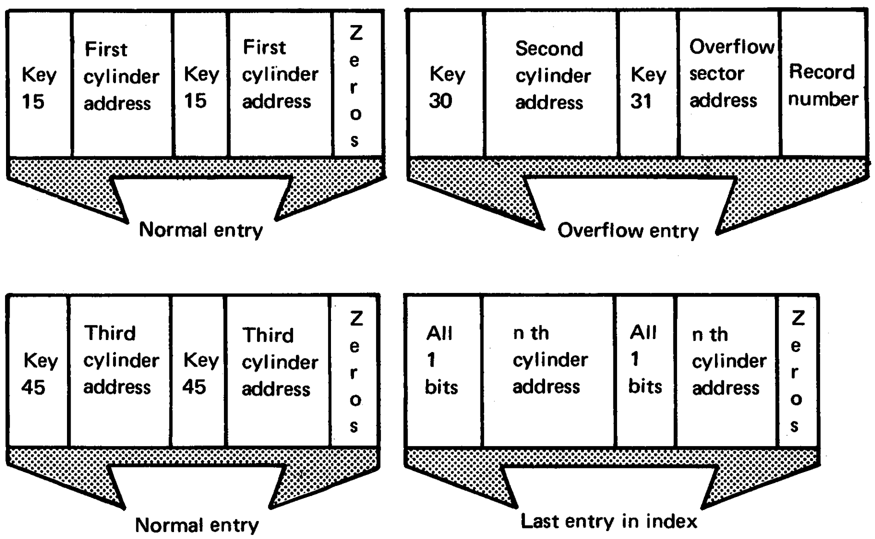
Word number	Label entry description
1	Key length
2	Record length
3	Number of index entries per sector
4	Index entry length
5	Number of records per sector
6	Record number of last prime data record
7	Index entry number of last entry in file
8	Sector address of last prime data record
9	Sector address of last index entry
10	Sector address of next overflow record
11	Record number of next overflow record

ISAM file index

The ability to read or write records anywhere in an ISAM file is provided by the file index. An entry in this index contains a cylinder address and the highest record key that is associated with that cylinder. The ISAM subroutines locate a given record by searching the index for the key and then searching the specified cylinder for the desired record, again searching by key. To increase the efficiency of the ISAM subroutines, one sector of the index is retained in core storage for each file.

The key can be a part number or an employee name or any other identifying information that is contained in any record in the file. The key entries in the index are the numbers in ascending collating sequence of the highest key on each cylinder. The end-of-file record key is the key with the highest possible value; all bits are ones.

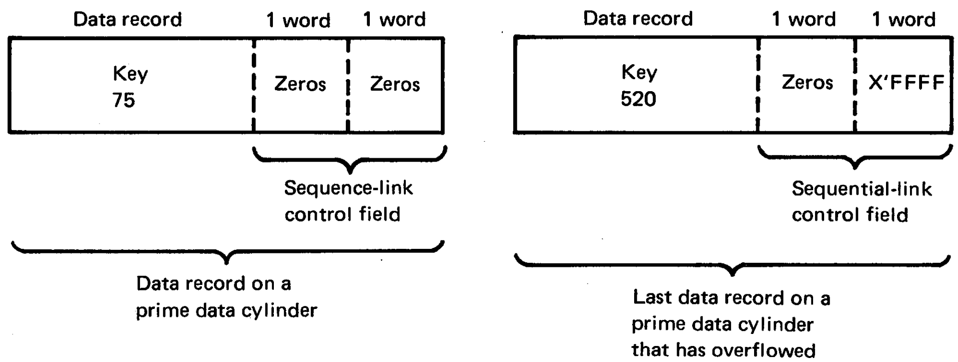
The following is a portion of an index table. Note that each entry contains 2 sets of the same information. The second set is overlaid to show overflow data when the affected cylinder overflows.



prime data area

The prime data area contains the data records that are placed in the file by the ISAM load subroutine. The records must all be the same length (maximum 318, decimal, words). The ISAM subroutine adds a 2-word control field to each record. This control field, called the sequence-link control field, is used in the overflow area as a chaining indicator. The control field indicates whether or not a cylinder has overflowed.

Prime data area records appear as follows:



overflow area

When a new record is added to an indexed sequential file, it is placed according to key sequence. If records were to remain in precise physical order, the insertion of each new record would require all records with higher keys to be shifted up. However, because ISAM files have an overflow area, a new record can be entered into its proper position and only cause records with higher keys to be shifted on that cylinder. The record that is forced off the end of the cylinder by the addition of the new record is written in the overflow area.

The index entry of any cylinder that has overflowed points to the overflow sector address and record number of the record placed in the overflow area. When 2 or more records are added in key order, the overflowed records are chained together through the entries in their sequence-link control field. The entry in the first record points to the second, the second to the third, and the third to the fourth. The last overflow record in the chain has a sequence-link control field of all zeros.

You specify the number of cylinders for the overflow area when you initially define the file. Then the ISAM subroutines place the records in the overflow area in the order that they overflow, not in key sequence.

To illustrate the overflow area, assume that on cylinder 6 of a defined file, the last 3 entries have keys 150, 152, and 154. Key 154 identifies cylinder 6 in the index. When you add a record with key 153, a record on another cylinder, and a record with key 151, the overflow area appears as follows:

Overflow area

Key 154	Zeros	Zeros		Zeros	Zeros	Key 153	Overflow sector address	Reg 0001

First record overflowed. The sequence-link control field is zeros indicating the end of a chain.

Record overflowed from another cylinder

Last record overflowed. The sequence-link control field points to the next key in sequence. In this case it's key 154 in the overflow area.

Key 152 now identifies cylinder 6 in the index; the overflow entry in the index for cylinder 6 points to the overflow area.

Deleting Duplicate Records Caused by a Disk Error During an ISAM Add Operation

If a disk error (/5004 displayed in the console ACCUMULATOR) occurs during an ISAM add operation, a record may be duplicated in the file. To check for a duplicate record, list the file or part of the file using the ISAM sequential retrieve. If a duplicate record is found, one copy must be deleted.

To determine which record to delete, dump the file using a DUP *DUMP function, and check the index entry for the affected cylinder. If the key of the duplicate record is less than or equal to the first key in the index entry, delete the second of the 2 records. If the key of the duplicate record is greater than the first key in the index entry, delete the first of the 2 records. In both cases, the remaining record is the one that is processed by the ISAM random retrieve function.

Note that the duplicate record is not physically deleted; it is deleted by performing a sequential read and flagging the copy that is no longer to be used.

TIPS FOR ASSEMBLER LANGUAGE PROGRAMMERS

The tips in this section are provided to help you with:

- Grouping assembler mnemonics to shorten assembly time
- Using index register 3
- Double buffering for faster I/O operations
- Using the 1403 conversion subroutines
- Writing ISSs and ILSs

Grouping of Assembler Mnemonics

The Monitor System Assembler Program is divided into overlay phases, each phase processing a certain group of mnemonics. Each time a mnemonic is processed during assembly, the overlay phase required to process it is read into core, unless the overlay is already residing in core.

Assembly time can be shortened by grouping mnemonics of a common type in your source program; thus fewer disk reads of overlay phases are required by the assembler. The following is a list of the mnemonics as they are grouped within the assembler program:

1. ABS, FILE, ENT, ISS, ILS, SPR, EPR
2. DCs and imperative instructions, such as A, LD, EOR, BSC
3. DEC and XFLC
4. DMES
5. HDNG, ORG, EQU, BSS, BES, LIST, SPACE, EJCT, DUMP, PDMP
6. LIBF, CALL, DSA, LINK, EXIT, EBC, DN

Assembler Program Use of Index Register 3

In general, index register 3 (XR3) is reserved to point to the transfer vector. Normally, you can use this register in your program; however, if you use LIBF statements, you must code your program to do the following:

1. At the beginning of your program, save the contents of XR3
2. Before each LIBF, save your program's contents of XR3 and restore the original contents (the pointer to the transfer vector) to XR3
3. After each LIBF, restore your program's contents to XR3

Under certain conditions, you cannot use index register 3 even if you code your program to save and restore its contents. These conditions include core loads that overlap I/O operations and core loads that use the synchronous communications adapter. When these conditions exist, you can use index register 3 if you specify that a special set of interrupt level subroutines (named with an X as ILSX4) be included in a core load. You specify the use of the special ILSs in a monitor XEQ control record.

Double Buffering in Assembler Programs

The IBM 2501 Card Reader, Model A2, rated at 1000 cards per minute, presents a special problem when you want maximum performance from card I/O operations. If any conversion of the card data is required, the reading speed can drop to 500 cards per minute. The use of double buffering can prevent the loss of speed.

The principle of double buffering is to read into one buffer while converting and processing the data from another buffer. This scheme uses additional core for the extra buffer and additional programming involved, but in most cases, card throughput should remain at 1000 cards per minute. The following coding example illustrates the double buffering technique used for reading cards from the 2501, and converting them to EBCDIC.

```
Label  Operation  F T  Operands & Remarks
21  25 27    30  32 33 35   40        45   50   55   60   65   70

      LIBF         READØ     PRIME DOUBLE-BUFFERED
      DC           /1ØØØ     *CARD READING. THIS READ
      DC           BUF1      *PERFORMED ONCE ONLY.
*                                                             *
READ  LIBF         READØ     THIS READ WILL NOT START
      DC           /1ØØØ     *UNTIL PREVIOUS READ
SET1  DC           BUF2      *IS COMPLETED
*                                                             *
      LIBF         ZIPCO     BR TO EXECUTE FAST CONVERT
      DC           /11Ø1     IBM CARD CODE TO EBCDIC
SET2  DC           BUF1+1    INPUT AREA ADDRESS
SET3  DC           BUF1+1    OUTPUT AREA ADDRESS
      DC           8Ø        NO. OF COLUMNS TO CONVERT
*                                                             *
      CALL         HLEBC     CONVERSION TABLE FOR ZIPCO
*                                                             *
      LDD          BFADR
      STO          SET1      CHANGE READ BUFFER ADDRESS
      RTE          16        EXCHANGE BUFFER ADDRESSES
      STD          BFADR     *FOR NEXT TIME THRU LOOP.
      A            ONE
      STO          SET2      CHANGE INPUT AND OUTPUT BFR
      STO          SET3      *ADDRESSES FOR CONVERSION.
*                                                             *
*                                                             *
*     CODING FOR REQUIRED PROCESSING SHOULD FOLLOW            *
*                                                             *
*                                                             *

      .
      .
      .
      B            READ
*                                                             *
*                                                             *
*     CONSTANTS AND WORK AREAS                                *
*                                                             *
*                                                             *
ONE   DC           1         CONSTANT VALUE OF 1
BUF1  DC           8Ø        WORD COUNT FOR CARD BFR 1
      BSS          8Ø        CARD BUFFER 1
BUF2  DC           8Ø        WORD COUNT FOR CARD BFR 2
      BSS          8Ø        CARD BUFFER 2
*                                                             *
*                                                             *
*     THE FOLLOWING PAIR OF ADDRESSES ARE EXCHANGED           *
*    EACH TIME THROUGH THE CARD READING LOOP                  *
*                                                             *
*                                                             *
      BSS        E Ø         MAKE NEXT LOCATION EVEN
BFADR DC           BUF1      ADDRESS OF CARD BUFFER 1
      DC           BUF2      ADDRESS OF CARD BUFFER 2
```

Assembler Program Use of 1403 Conversion Subroutines

Two monitor system subroutines can be used by assembler object programs to convert from EBCDIC to 1403 Printer code. These subroutines are EBPRT and ZIPCO.

By using the execution times listed in the publication *IBM 1130 Subroutine Library,* GC26-5929 EBPRT requires an average of 156 ms (milliseconds) to convert a 120 character line compared to an estimate of 72 ms per line for ZIPCO.

The speeds at which the 1403 Printer can print a line are:

Model 6 (340 LPM) – 176 ms/line; Model 7 (600 LPM) – 100 ms/line

Considering these speeds, running the printer at rated speed is difficult or impossible, depending on the model when EBPRT is used. If overlapped I/O is attempted, running either model at rated speed is impossible. Because of this, the assembler language programmer is advised to use ZIPCO for all EBCDIC-to-1403 Printer code conversions.

Writing ISSs and ILSs

Interrupt service subroutines (ISSs) for all 1130 devices and interrupt level subroutines (ILSs) for all 1130 interrupts are provided with the monitor system; however, if you want to, you may write your own.

ISS subroutines

These rules must be followed when writing ISSs:

1. Precede the ISS statement (see rule 3) with a LIBR statement if the subroutine is to be called by a LIBF rather than a CALL.
2. Precede the subroutine with an EPT (extended) or an SPR (standard) statement if precision specification is necessary.
3. Precede the subroutine with an ISS statement (only one) that defines the entry point and the ISS number. The ISS numbers used in the IBM-supplied ISS and ILS subroutines are listed in Figure 6-2. The assembler ISS statement is described in the publication *IBM 1130/1800 Assembler Language,* GC26-3778. Note that the ISS numbers assigned by the IBM-supplied subroutines range from 1 through 11. You can assign ISS numbers from 12 through 20; assign these numbers starting with 20.

ISS number	Device	Device interrupt level assignments	n
1	1442 Card Reader Punch	0, 4	+4, +7
2	Input keyboard/console printer	4	+4
3	1134/1055 Paper Tape Reader/Punch	4	+4
4	2501 Card Reader	4	+4
5	Disk storage	2	+5
6	1132 Printer	1	+4
7	1627 Plotter	3	+4
8	Synchronous Communications Adapter	1	+4
9	1403 Printer	4	+4
10	1231 Optical Mark Page Reader	4	+4
11	2250 Display Unit	3	+4

Figure 6-2. I/O device ISS numbers and ILS interrupt levels

4. When assembling an ISS, include an assembler *LEVEL control record for each interrupt level associated with the device.
5. The entry points of an ISS are defined by the related ILS. Consider this when you write an ISS that is to be used with an IBM-supplied ILS. The IBM ILS executes a BSI statement to the ISS entry point plus *n* (see the +n column in Figure 6-2). Your ISS subroutine must return to the ILS via a BSC statement (not a BOSC).

The following listing is an example of an ISS subroutine.

```
// ASM
*XREF
*LEVEL 4

*****************************************************   ISSC0020
*TITLE- READC                                       *   ISSC0030
*FUNCTICN/CPERATICN-                                *   ISSC0040
*      THIS 1130 SUBRCUTINE CPERATES THE PRIMARY    *   ISS00050
*      2501 CARD READER.  IT INITIATES REQUESTED    *   ISS00060
*      OPERATICNS, PRCCESSES CPERATICN CCMPLET      *   ISSC0070
*      INTERRUPTS, ANC ALTOMATICALLY INITIATES      *   ISSC0080
*      ERRCR RECCVERY PRCCEDURES.                   *   ISS00090
*                                                   *   ISSC0100
*ENTRY PCINTS-                                      *   ISSC0110
*      1. READC CALL ENTRANCE FOR TEST CR READ      *   ISSC0120
*               CPERATICNS.  E.G.  LIBF    READO    *   ISSC0130
*                                  DC      /1CCO    *   ISS00140
*                                  DC      ICBUF    *   ISSC0150
*      I                           DC      ICBUF    *   ISSC0150
*      2. RE048 CPERATION COMPLETE INTERRUPT ENTRY  *   ISSC0160
*               PCINT.                              *   ISSC0170
*INPUT- NCNE OTHER THAN FROM THE PARAMETERS IN      *   ISSC0180
*      LIBF CALLINC SEQUENCE.                       *   ISS00190
*CUTPUT- RCUTINE WILL TRANSFER 0 TO 80 COLS FROM    *   ISSC0200
*      CARD TC I/O BUFFER AS SPECIFIED BY CALLING   *   ISSC0210
*      SECUENCE.  FORMAT IS 12 BITS PER BUFFER WORD*    ISS00220
*      LEFT JUSTIFIEC.                              *   ISSC0230
*EXTERNAL SUBROUTINES- NONE.                        *   ISS00240
*EXITS-                                             *   ISS00250
*   NCRMAT-                                         *   ISSC0260
*      1. RE180 IF NC PRE-CP ERROR HAS BEEN DE-     *   ISSC0270
*               TECTED, THE EXIT FROM RE180 IS      *   ISS00280
*               TO THE CALLER AFTER THE REQUESTED   *   ISS00290
*               2501 CPERATICN HAS BEEN INITIATED   *   ISSC0300
*      2. RE348 THE EXIT FROM RE348 IS BACK TO THE  *   ISSC0310
*               CALLER VIA ILS04 AFTER OP-COMPLETE  *   ISS00320
*               PROCESSING HAS BEEN FINISHED.       *   ISSC0330
*   ERRCR-                                          *   ISS00340
*      1. RE180 IF A PRE-OP ERROR OR NCT READY      *   ISSC0350
*               CONDITION IS CETECTEC, SUBROUTINE   *   ISS00360
*               WILL BRANCH TC HEX 0029 VIA RE180   *   ISSC0370
*               AND CISPLAY CNE OF TWC CCDES IN     *   ISSC0380
*               ACCUMULATOR.                        *   ISS00390
*               4000 IS CISPLAYED IF 2501 IS NOT    *   ISSC0400
*               READY.  4C01 IS DISPLAYED IF AN     *   ISSC0410
*               ERROR IS DISCOVERED IN CALLING      *   ISSC0420
*               PARAMETERS OR AREAS REFERENCED BY   *   ISSC0430
*               THEM.                               *   ISS00440
*TABLES/WCRK AREAS- NCNE.                           *   ISS00450
*ATTRIBUTES- REUSABLE, CAN READ UP TC 80 COLUMNS    *   ISSC0460
*               CF BINARY DATA.  IF A WORD CCUNT    *   ISSC0470
*               CF ZERC IS SPECIFIED, THE READ      *   ISSC0480
*               CPERATICN ACTS AS A FEED.           *   ISSC0490
```

```
                          00050         LIBR                                                  ISS00510
0000       19141130  00051    1130  ISS   04 READ0      4                                     ISS00520
                          00052    ************************************************************  ISS00530
                          00053    *             LOADER DEFINED LOCATIONS                   *  ISS00540
                          00054    ************************************************************  ISS00550
0000 0    692E         00055    READ0 STX    1 RE144+1      LIBF ENTRANCE                      ISS00560
0001 00 658000000 00056          LDX   I1 *-*           LOADER STORES TV ADDR(+2)          ISS00570
0003 0    7C03         00057          MDX       RE060         BR TO PROCESS CALL                 ISS00580
0004 0    0000         00058    RE048 DC        *-*           OP-CMPLTE INTERRUPT   (+4)        ISS00590
0005 01 4C00004B   00059          BSC   L  RE336         BR TO PROCESS INT                  ISS00600
                          00060    ************************************************************  ISS00610
                          00061    *             LIBF PROCESSING                            *  ISS00620
                          00062    ************************************************************  ISS00630
                          00063    *     THIS PORTION STORES CALLING SEQUENCE INFO         *  ISS00640
                          00064    *     AND CHECKS THE DEVICE STATUS BEFORE ANY I/O       *  ISS00650
                          00065    *     OPERATION IS INITIATED. A CALLING ERROR OR         *  ISS00660
                          00066    *     NOT READY 2501 CAUSES AN ERROR EXIT TO             *  ISS00670
                          00067    *     LOCATION 41. IF THE OPERATION WILL CAUSE AN       *  ISS00680
                          00068    *     INTERRUPT, THE ROUTINE IS SET BUSY AN THEAN       *  ISS00690
                          00069    *     IOCS COUNTER IS INCREMENTED TO INDICATE           *  ISS00700
                          00070    *     AN INTERRUPT IS PENDING.                           *  ISS00710
                          00071    ************************************************************  ISS00720
0007 0    D042         00072    RE060 STD       RE324         SAVE ACC                           ISS00730
0008 0    2829         00073          STS       RE168         SAVE STATUS                        ISS00740
0009 0    6A27         00074          STX    2 RE156+1       SAVE XR2                           ISS00750
000A 0    C100         00075          LD     1 0             XR1 = ADDR OF CALL+1               ISS00760
000B 0    1800         00076          SRA       12            IS FUNCTION TEST                   ISS00770
000C 01 4C200012   00077          BSC   L  RE072,Z       BR IF NOT                          ISS00780
000E 0    C031         00078          LD        RE228         IS SUBR BUSY                       ISS00790
000F 0    4818         00079          BSC       +-            SKIP IF YES                        ISS00800
0010 0    7101         00080          MDX    1 +1            NO, EXIT TO CALL+3                 ISS00810
0011 0    7019         00081          MDX       RE120         EXIT TO CALL+2                     ISS00820
0012 0    902F         00082    RE072 S         RE240         IS FUNCTION LEGAL                  ISS00830
0013 01 4C200035   00083          BSC   L  RE192,Z       BR IF NOT                          ISS00840
0015 0    C02A         00084    RE084 LD        RE228         IS SUBR BUSY                       ISS00850
0016 01 4C200015   00085          BSC   L  RE084,Z       YES, LOOP                          ISS00860
0018 0    082D         00086    RE096 XIO       RE288-1       IS DEVICE READY                    ISS00870
0019 01 4C040037   00087          BSC   L  RE204,E       BR IF NOT                          ISS00880
001B 00 C5800001   00088          LD    I1 1             OBTAIN WORD COUNT                  ISS00890
001D 0    4818         00089          BSC       +-                                               ISS00900
001E 0    7005         00090          MDX       RE108         BR ON Z WD CNT                     ISS00910
001F 01 4C280035   00091          BSC   L  RE192,Z+      BR IF WD CNT NEG                   ISS00920
0021 0    9024         00092          S         RE276         0 THRU 80 IS LEGAL                 ISS00930
0022 01 4C300035   00093          BSC   L  RE192,Z-      BR IF OVER 80                      ISS00940
0024 0    7101         00094    RE108 MDX    1 +1            XR1 POINTS TO 2ND PARAM            ISS00950
0025 0    C100         00095          LD     1 0             SAVE DATA ADDR                     ISS00960
0026 0    D01D         00096          STO       RE264                                            ISS00970
0027 00 74010032   00097          MDX   L  $IOCT,1       INCREMENT IOCS COUNTER             ISS00980
0029 0    6816         00098          STX    0 RE228         SET SUBR BUSY INDR                 ISS00990
002A 0    0819         00099          XIO       RE264         INITIATE READ                      ISS01000
002B 0    7101         00100    RE120 MDX    1 +1            XR1 POINTS TO RTN ADDR             ISS01010
002C 0    C01D         00101          LD        RE324         RESTORE ACC                        ISS01020
002D 0    6906         00102    RE132 STX    1 RE180+1       SET RETURN ADDRESS                 ISS01030
002E 00 65000000   00103    RE144 LDX   L1 *-*           RESTORE STATUS                     ISS01040
0030 00 66000000   00104    RE156 LDX   L2 *-*           AND INDEX REGISTERS                ISS01050
0032 0    2000         00105    RE168 LDS       *-*                                              ISS01060
0033 00 4C000000   00106    RE180 BSC   L  *-*           EXIT                               ISS01070
0035 0    C013         00107    RE192 LD        RE312         ERROR CODE - ILLEGAL CALL          ISS01080
0036 0    7004         00108          MDX       RE216         BR TO SET RETURN ADDRS             ISS01090
0037 0    1801         00109    RE204 SRA       1             IS DEVICE BUSY                     ISS01100
0038 01 4C040018   00110          BSC   L  RE096,E       BR IF YES                          ISS01110
003A 0    C00D         00111          LD        RE300         ERROR CODE - DVCE NOT RDY          ISS01120
003B 0    71FF         00112    RE216 MDX    1 -1            XR1 = CALLING ADDRESS              ISS01130
003C 00 6D000028   00113          STX   L1 $PRET         STORE CALL ADDR IN 40              ISS01140
003E 0    6129         00114          LDX    1 $PRET+1       XR1 = EXIT ADDRESS                 ISS01150
003F 0    70ED         00115          MDX       RE132         BR TO EXIT                         ISS01160
```

```
                     00116   ************************************************** ISS01170
                     00117   *              CONSTANTS                         * ISS01180
                     00118   ************************************************** ISS01190
0040 0   0000        00119   RE228 DC      0           SUBR BUSY INDR           ISS01200
0042     0000        00120         BSS  E  0                                    ISS01210
0042 0   0001        00121   RE240 DC      +1          CONSTANT                 ISS01220
0043 0   4F01        00122   RE252 DC      /4F01       SENSE WITH RESET         ISS01230
0044 0   0000        00123   RE264 DC      *-*         I/O BUFFER ADDRESS       ISS01240
0045 0   4E00        00124         DC      /4E00       IOCC TO INITIATE READ    ISS01250
0046 0   0050        00125   RE276 DC      +80         CONSTANT                 ISS01260
0047 0   4F00        00126   RE288 DC      /4F00       SENSE DSW WITHOUT RESET  ISS01270
0048 0   4C00        00127   RE300 DC      /4C00       CONSTANT FOR DVC NOT RDY ISS01280
0049 0   4C01        00128   RE312 DC      /4C01       CST FOR BAD CALL         ISS01290
004A 0   0000        00129   RE324 DC      *-*         SAVED ADD                ISS01300
0028                 00130   $PRET EQU     /28         PRE-OPERATIVE ERROR TRAP ISS01310
0032                 00131   $IOCT EQU     /32         I/O COUNTER              ISS01320
008D                 00132   $PST4 EQU     /8D         POST-OPERATIVE ERROR TRAP ISS01330
                     00133   ************************************************** ISS01340
                     00134   *              OP-COMPLETE INTERRUPT PROCESSING  * ISS01350
                     00135   ************************************************** ISS01360
                     00136   *   THIS PORTION IS ENTERED FROM AN INTERRUPT    * ISS01370
                     00137   *   LEVEL SUBRT. IF NO ERROR HAS BEEN DETECTED   * ISS01380
                     00138   *   THE ROUTINE IS SET NOT BUSY AND THE IOCS     * ISS01390
                     00139   *   COUNTER IS DECREMENTED TO INDICATE           * ISS01400
                     00140   *   INTERRUPT PROCESSING COMPLETED. OTHERWISE    * ISS01410
                     00141   *   THE SUBR. GOES TO THE POST-OPERATIVE ERROR-  * ISS01420
                     00142   *   TRAP AND WAITS UNTIL THE OPERATOR HAS        * ISS01430
                     00143   *   INTERVENED AND THE 2501 BECOMES READY, AT    * ISS01440
                     00144   *   WHICH TIME THE CARDS ARE POSITIONED AND THE  * ISS01450
                     00145   *   I/O OPERATION IS RE-INITIATED.               * ISS01460
                     00146   ************************************************** ISS01470
004B 0   08F6        00147   RE336 XIO     RE252-1     SENSE DSW WITH RESET     ISS01480
004C 0   1003        00148         SLA     3           IS OPERATION OK          ISS01490
004D 01  4C020056    00149         BSC  L  RE360,C     BR IF ERROR              ISS01500
004F 00  74FF0032    00150         MDX  L  $IOCT,-1    DECREMENT IOCS           ISS01510
0051 0   1000        00151         NOP                 IN CASE OF SKIP          ISS01520
0052 0   1810        00152         SRA     16                                   ISS01530
0053 0   D0EC        00153         STO     RE228       CLEAR ROUT BUSY INDIC    ISS01540
0054 01  4C800004    00154   RE348 BSC  I  RE048       EXIT                     ISS01550
0056 0   08EB        00155   RE360 XIO     RE252-1     SENSE DSW FOR READY      ISS01560
0057 01  4C04005B    00156         BSC  L  RE365,E     TO ERROR EXIT IF NOT RDY ISS01570
0059 0   08EA        00157         XIO     RE264       RE-INITIATE FUNCTION     ISS01580
005A 0   70F9        00158         MDX     RE348       BR TO EXIT               ISS01590
005B 0   C0EC        00159   RE365 LD      RE300       LD NOT READY ERROR CODE  ISS01600
005C 00  4400008D    00160         BSI  L  $PST4       POST-OPERATIVE ERROR TRAP ISS01610
005E 0   70F7        00161         MDX     RE360       TRY AGAIN                ISS01620
0060                 00162         END                                          ISS01630
```

```
                         CRCSS-REFERENCE

SYMBCL     VALUE   REL  CEFN      REFERENCES

REACO      COCC    1    C0055     CCC51,R
REC48      C0C4    1    CC058     C0154,B
REC60      COC7    1    CC072     CCC57,B
REC72      C012    1    C0082     0CC77,B
REC84      C015    1    C0084     CC085,B
REC96      C018    1    C0086     CC11C,B
RE1C8      C024    1    C0094     CC090,B
RE120      C02B    1    C01CC     C0081,B
RE132      C02D    1    C01C2     C0115,B
RE144      C02E    1    001C3     C0055,M
RE156      0030    1    001C4     0C074,M
RE168      C032    1    C0105     CCC73,M
RE180      C033    1    C0106     CC1C2,M
RE192      0035    1    C0107     C0083,B C0091,B CC093,B
RE204      CC37    1    C0109     CC087,B
RE216      CC3B    1    C0112     C0108,B
RE228      004C    1    C0119     CCC78,R CC084,R CC098,M C0153,M
RE240      C042    1    C0121     C0082,R
RE252      C043    1    C0122     C0147,R C0155,R
RE264      C044    1    00123     00096,M OCC99,R C0157,R
RE276      C046    1    C0125     00092,R
RE288      C047    1    C0126     C0086,R
RE3C0      C048    1    C0127     C0111,R C0159,R
RE312      0049    1    C0128     001C7,R
RE324      C04A    1    C0129     C0072,M C01C1,R
RE336      C04B    1    C0147     C0059,B
RE348      0054    1    C0154     C0158,B
RE360      C056    1    C0155     C0149,B C0161,B
RE365      C05B    1    C0159     CC156,B
$ICCT      C032    C    C0131     C0097,M C015C,M
$PRET      C028    0    00130     C0113,M CC114,R
$PST4      C08C    C    C0132     0016C,B

   COC CVERFLCW SECTCRS SPECIFIED
   CCC CVERFLCW SECTCRS REQUIRED
   C32 SYMBCLS DEFINED
    NC ERRCR(S) ANC   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY
```

ILS subroutines

An ILS is included in a core load only if requested by an ISS that is a part of the same core load. The IBM-supplied ILS02 and ILS04 subroutines are a part of the resident monitor unless you delete them from the system library and replace them with ILSs that you write for interrupt levels 2 and 4. These rules must be followed when writing an ILS:

1. Precede the subroutine with an ILS statement that identifies the interrupt level involved.
2. Precede all statements with an ISS branch table. If the associated interrupt level status word (ILSW) is not scanned (that is, a single ISS handles all interrupts on the level involved) in the ILS, a one-word table is sufficient; the minimum table size is one word. A zero must follow the branch table. If the ILSW is scanned, the ISS branch table must include one word for each used bit of the ILSW:

   ISS branch table

ILSW bit X (highest bit used)	Define one word for each bit used.
.	
.	
.	
ILSW bit 1	
ILSW bit 0	

   Each entry in the ISS branch table identifies the entry point within an ISS for the associated ILSW bit. The actual linkage is generated by the core load builder. Before processing by the CLB, each word in the ISS branch table has the following format:

   - Bits 0 through 7 contain an increment that is added to the entry point address of the corresponding ISS subroutine to obtain the interrupt entry point address within the ISS for the ILSW bit. (In IBM-written ISSs, this increment is +4 for the primary interrupt level and +7 for the secondary interrupt level. See column +n in Figure 6-2.)
   - Bits 8 through 15 contain the value of @ISTV plus the ISS number of the ISS associated an ILSW bit. The value of @ISTV can be obtained from the cross-reference symbol table at the end of the resident monitor listing in Appendix G.

   @ISTV is the address of the interrupt transfer vector (ITV) in low core. Any ISS branch table entries that represent unused bits in an ILSW must have the value @ISTV.

   During the building of a core load, the CLB places the entry point address of an ISS in the location of the ITV that corresponds to the ISS number specified in the ISS statement. The CLB generates an ISS entry point address by adding the increment in bits 0 through 7 to the address in the location of the ITV pointed to by bits 8 through 15. Then the CLB replaces the ISS branch table word with this generated interrupt entry point address. (See Step 4 for the use of these addresses.)
3. The ILS entry point must immediately follow the ISS branch table and must be loaded as a zero. The core load builder assumes that the first zero word in the program is the end of the branch table and is also the entry point of the ILS. An interrupt causes a BSI to this entry point.
4. The ILSW bit that is on is determined with a SLCA statement. At the completion of this statement, the specified index register contains a relative value equivalent to that bit position in the ISS branch table. The address in the ISS branch table can then be used by a BSI instruction to reach the ISS that corresponds to an ILSW bit position.
5. To clear the interrupt level when an ILS that you write is used with an IBM-supplied ISS, code your ILS to exit via the return linkage with a BOSC statement.

6. When you write an ILS, it must replace the equivalent IBM-supplied ILS. Delete the IBM ILS, and store your ILS as ILS0x, where x = 0, 1, 2, 3, 4, or 5.
7. The IBM-supplied ILS02 and ILS04 subroutines are stored as subtype one. An ILS that you write to replace either of these must be stored as subtype zero.
8. The ISS branch table for the IBM-supplied version of ILS04 can have no more than 9 entries. An ILS that you write to replace ILS04 can support all 16 possible ISS branch table entries.

The following listing is an example of an ILS subroutine.

```
// ASM
*XREF

                 ************************************************** U1J00020
                 *                                                * U1J00030
                 * NAME - ILSX4                                   * U1J00040
                 * FUNCTION/OPERATION - INTERRUPT LEVEL SUBROUTINE * U1J00050
                 *    FOR LEVEL 4.                                * U1J00060
                 * ENTRY POINT - ENTERED AT IX420 BY A HARDWARE   * U1J00070
                 *    BSI VIA LOCATION 12 DECIMAL.                * U1J00080
                 * INPUT - NONE                                   * U1J00090
                 * OUTPUT - NONE                                  * U1J00100
                 *EXTERNAL SUBROUTINES - NONE                     * U1J00110
                 * EXITS -                                        * U1J00120
                 *    NORMAL - BOSC INDIRECT THROUGH IX420.       * U1J00130
                 *    ERROR - NONE                                * U1J00140
                 * TABLES/WORK AREAS - NONE                       * U1J00150
                 * ATTRIBUTES - REUSABLE                          * U1J00160
                 * NOTES - INDEX REGISTERS 1, 2, AND 3, STATUS,   * U1J00170
                 *    ACCUMULATOR AND EXTENSION ARE SAVED UPON    * U1J00180
                 *    ENTRY AND RESTORED AFTER INTERRUPT SERVICED. * U1J00190
                 *                                                * U1J00200
                 ************************************************** U1J00210
                       ILS  04                                      U1J00230
0000 0  0033     IX410 DC       /0033     DEVICD *-* AND ISS NO. *-* U1J00240
0001 0  0033           DC       /0033     DEVICD *-* AND ISS NO. *-* U1J00250
0002 0  0033           DC       /0033     DEVICD *-* AND ISS NO. *-* U1J00260
0003 0  043D           DC       /043D     1231 +4 AND ISS NO. 10   U1J00270
0004 0  043C           DC       /043C     1403 +4 AND ISS NO.  9   U1J00280
0005 0  0437           DC       /0437     2501 +4 AND ISS NO.  4   U1J00290
0006 0  0734           DC       /0734     1442 +4 AND ISS NO.  1   U1J00300
0007 0  0435           DC       /0435     CONSOLE +4 AND ISS NO. 2 U1J00310
0008 0  0436           DC       /0436     1134/1055 +4 AND ISS NO. 3 U1J00320
                 *                                                  U1J00330
0009 0  0000     IX420 DC       0         INTERRUPT ENTRY          U1J00340
000A 0  D81B           STD      IX480     SAVE ACC AND EXTENSION,  U1J00350
000B 0  280F           STS      IX430     *STATUS,                 U1J00360
000C 0  6910           STX   1  IX441+1   *XR1,                    U1J00370
000D 0  6A11           STX   2  IX442+1   *XR2,                    U1J00380
000E 0  6B12           STX   3  IX443+1   *XR3                     U1J00390
000F 00 678000E4       LDX  I3  $XR3X     POINT TO TRANSFER VECTOR U1J00400
0011 0  0818           XIO      IX495-1   SENSE KEYBOARD           U1J00410
0012 0  1002           SLA      2         IS IT INTERRUPT REQUEST  U1J00420
0013 00 44A8002C       BSI   I  $IREQ,+Z  *KEY, BR IF YES          U1J00430
0015 0  1000           NOP                                         U1J00435
0016 0  6109           LDX   1  9         NUMBER OF DEVICES ON LEVEL U1J00440
0017 0  0810           XIO      IX490-1   SENSE ILSW               U1J00450
0018 0  1140           SLCA  1  0         SHIFT AND DECREMENT XR1  U1J00460
0019 01 4580FFFF       BSI  I1  IX410-1   BR TO DEVICE ISS         U1J00470
                 *                                                  U1J00480
001B 0  2C00     IX430 LDS      0         RESTORE STATUS,          U1J00490
001C 00 65000000 IX441 LDX  L1  *-*       *XR1,                    U1J00500
001E 00 66000000 IX442 LDX  L2  *-*       *XR2,                    U1J00510
0020 00 67000000 IX443 LDX  L3  *-*       *XR3,                    U1J00520
0022 0  C803           LDD      IX480     *ACC AND EXTENSION       U1J00530
0023 01 4C000009       BOSC  I  IX420     TURN OFF INTERRUPT, RETURN U1J00540
                 *                                                  U1J00550
0026    0002     IX480 BSS   E  2         ACCUMULATOR AND EXTENSION U1J00560
0028 0  0000           DC       0                                  U1J00570
0029 0  0300     IX490 DC       /0300     IOCC TO SENSE ILSW       U1J00580
002A 0  0000           DC       0                                  U1J00590
002B 0  0F00     IX495 DC       /0F00     SENSE IOCC FOR DEYBOARD  U1J00600
002C             $IREQ EQU      /002C     ADDR OF ISS FOR INT REQ  U1J00610
00E4             $XR3X EQU      /00E4     ADDR OF TRANSFER VECTOR  U1J00620
002C                   END                                         U1J00630
```

```
SYMBCL      VALUE   REL  DEFN      REFERENCES

IX410       COCO    1    00022     C0046,B
IX420       CCC9    1    C0032     CC053,B
IX430       C01B    1    CCC48     CCC34,M
IX441       C01C    1    CC049     CCC35,M
IX442       C01E    1    C0050     C0036,M
IX443       C02C    1    CC051     C0C37,M
IX480       C026    1    C0055     0C033,M  CC052,R
IX490       0029    1    C0057     CC044,R
IX495       C02B    1    CC059     C0039,R
$IREQ       C02C    C    CC060     0C041,B
$XR3X       C0E4    C    CC061     00038,R

   CCC OVERFLCW SECTCRS SPECIFIEC
   C00 OVERFLCW SECTCRS REQUIREC
   011 SYMBCLS CEFINEC
    NC ERRCR(S) ANC   NC WARNING(S)   FLAGGED IN ABOVE ASSEMBLY
```

Assembler INT REQ Service Subroutine

Pressing the interrupt request key (INT REQ) on the console keyboard causes the ILS in use for interrupt level 4 (ILS04 or ILSX4) to execute a BSI I $IREQ. Thus, the function of the INT REQ key depends on the contents of location $IREQ. The system initializes $IREQ with the address $I420 in the resident monitor. This setting terminates the current job, and all control records are bypassed until the next JOB monitor control record is read. You can alter the function of the INT REQ key by coding your program to place, in $IREQ, the address of an INT REQ service subroutine that you have written.

An INT REQ service subroutine that you write can read the console entry switches and set program indicators. You should remember that your subroutine is executed with interrupt level 4 on, preventing recognition of other interrupts on level 4 or 5. Because of this, the following should be kept in mind when you code an INT REQ service subroutine:

- A LIBF or CALL to a subroutine from your service subroutine can cause a recurrent-entry problem. If the called subroutine is already in use when you press INT REQ, the new LIBF or CALL in your subroutine destroys the original return address and disrupts the operation of the called subroutine.
- A LIBF or CALL to an ISS can cause an endless loop if the called ISS operates on level 4 and a test for operation completed is performed by your service subroutine. This loop occurs because the interrupt indicating the operation is complete is delayed until the INT REQ key interrupt is turned off.
- Your subroutine must perform an XIO sense keyboard/console with reset before returning.
- Your subroutine must increment the return address by 6 when returning to the ILS subroutine. A BSC instruction must be used to go back to the ILS where the interrupt is turned off.

Note. When the core load of your program contains the TYPEZ, WRTYZ, TYPE0, or WRTY0 subroutine, the XIO sense keyboard/console with reset can be omitted. In this case, code your subroutine to return to the return address plus one.

Two sample subroutines are included in this section to illustrate how the function of the INT REQ key can be altered temporarily. These subroutines can be called by either FORTRAN or assembler programs. Both subroutines perform the same function; when INT REQ is pressed, the console entry switches are read. If console entry switch zero is off, program execution continues from where it was interrupted. If console entry switch zero is on, the system exits to the next job. The first of the sample INT REQ service subroutines (Figure 6-3) illustrates the coding that can be used by any core load. The second of the sample INT REQ service subroutines (Figure 6-4) illustrates the coding that can be used by a core load that contains TYPEZ, WRTYZ, TYPE0, or WRTY0.

Label 21–25		Operation 27–30	F 32	T 33		Operands & Remarks 35–70

```
****************************************************
*
*   A CALL TO THIS SUBROUTINE WILL CHANGE THE
*  CONTENTS OF $IREQ IN THE RESIDENT MONITOR. IF
*   THE INTERRUPT REQUEST KEY IS PRESSED AFTER A
*   CALL TO THIS SUBROUTINE HAS BEEN EXECUTED, A
*   BRANCH TO THE SECOND PART OF THE SUBROUTINE
*   WILL TAKE PLACE. THIS SUBROUTINE CAN BE USED IN
*   ANY CORE LOAD AND WILL PREVENT FLUSHING TO THE
*   NEXT JOB IF THE INT REQ KEY IS PRESSED.
*
****************************************************
*
      ENT        IREQ
*
*   THIS PORTION WILL BE ENTERED WHEN A CALL IREQ
*   IS EXECUTED
*
IREQ  DC         *-*          ENTRY POINT
      STX     1  IR010+1      SAVE XR1
      LDX    L1  INTR         SET XR1=ADDR OF INTR PORTN
      STX    L1  $IREQ        SET INTERRUPT BRANCH ADDR
IR010 LDX    L1  *-*          RESET XR1
      BSC    I   IREQ         RETURN TO CALLING PGM
*
$IREQ EQU        /002C
*
*   THIS PORTION WILL BE ENTERED WHEN THE INTERRUPT
*   REQUEST KEY IS PRESSED
*
INTR  DC         *-*          ENTRY POINT FROM ILS
      XIO        IN910        READ THE CONSOLE SWITCHES
      LD         IN930        LD WHAT HAS BEEN READ
      BSI    L   $1420,Z+     FLUSH TO NEXT JOB IF NEG
      XIO        IN920        SENSE KEYBOARD WITH RESET
      MDX    L   INTR,6       INCREMENT RETURN ADDR
      BSC    I   INTR         RETURN TO ILS
*
      BSS    E   0            CREATE EVEN ADDR
IN910 DC         IN930        IOCC TO READ CONSOLE
      DC         /3A00        *SWITCHES
IN920 DC         *-*          IOCC TO SENSE KEYBOARD WITH
      DC         /0F01        *RESET
IN930 EQU        IN920        VALUE READ FROM CONSOLE SW
$1420 EQU        /00E6
      END
```

Figure 6-3. INT REQ service subroutine for any core load

Label		Operation		F	T		Operands & Remarks
21 25		27 30		32	33		35 40 45 50 55 60 65 70

```
**************************************************
*
*   A CALL TO THE SUBROUTINE WILL CHANGE THE
*  CONTENTS OF $IREQ IN THE RESIDENT MONITOR. IF
*   THE INTERRUPT REQUEST KEY IS PRESSED AFTER A
*   CALL TO THIS SUBROUTINE HAS BEEN EXECUTED, A
*   BRANCH TO THE SECOND PART OF THE SUBROUTINE
*   WILL TAKE PLACE. THIS SUBROUTINE CAN ONLY BE
*   USED IF TYPE0, WRTY0, TYPEZ, OR WRTYZ IS IN
*   THE CORE LOAD AND WILL PREVENT FLUSHING TO THE
*   NEXT JOB IF THE INT REQ KEY IS PRESSED.
*
**************************************************
*
      ENT     IREQ
*
*    THIS PORTION WILL BE ENTERED WHEN A CALL IREQ
*    IS EXECUTED.
*
IREQ  DC      *-*       ENTRY POINT
      STX   1 IR010+1   SAVE XR1
      LDX  L1 INTR      SET XR1=ADDR OF INTR PORTN
      STX  L1 $IREQ     SET INTERRUPT BRANCH ADDR
IR010 LDX  L1 *-*       RESET XR1
      BSC  I  IREQ      RETURN TO CALLING PGM
*
$IREQ EQU     /002C
*
*    THIS PORTION WILL BE ENTERED WHEN THE INTERRUPT
*    REQUEST KEY IS PRESSED.
*
INTR  EQU     /002C
*
*    THIS PORTION WILL BE ENTERED WHEN THE INTERRUPT
*    REQUEST KEY IS PRESSED
*
INTR  DC      *-*       ENTRY POINT FROM ILS
      XIO     IN910     READ THE CONSOLE SWITCHES
      LD      IN920     LD WHAT HAS BEEN READ
      BSI  L  $I420,Z+  FLUSH TO NEXT JOB IF NEG
      MDX  L  INTR,1    INCREMENT THE RETURN ADDR
      BSC  I  INTR      RETURN TO ILS
*
      BSS  E  0         CREATE EVEN ADDR
IN910 DC      IN920     IOCC TO READ CONSOLE
      DC      /3A00     *SWITCHES
IN920 DC      *-*       VALUE READ FROM CONSOLE SW
$I420 EQU     /00E6
      END
```

Figure 6-4. INT REQ service subroutine for core load using TYPEZ, WRTYZ, TYPE0, or WRTY0

TIPS FOR FORTRAN PROGRAMMERS

The tips in this section will help you when:

- Referencing different data files by using the supervisor *EQUAT control record
- Using valid input data during program execution
- Controlling the console printer during program execution
- Entering data for arrays so as to provide efficient dumping of a DSF program

Tips for Use of the EQUAT Control Record

The supervisor *EQUAT function is used to substitute a subroutine for another called subroutine in core loads that are being built. Thus, a program does not have to be recompiled or reassembled to reference different subroutines.

For example, suppose that your FORTRAN mainline program prints on the 1132 Printer, and you want to have it print on the 1403 instead. Without an EQUAT control record, you would have to change the *IOCS control record and recompile the program. With EQUAT, you have only to specify on the EQUAT control record that PRNZ (the 1403 subroutine) is to be substituted for PRNTZ (the 1132 subroutine) when the core load is built. When EQUAT is used, the core load builder compares each call in the program with the left-hand name of each specified subroutine pair on the EQUAT control record. Each time a match is found, the core load builder substitutes the right-hand name of the EQUAT subroutine pair for the name in the calling statement of the program. Note that the EQUAT control record is associated with the monitor JOB control record, which implies that all core loads that are built for the job be built from the same substitution list.

The use of EQUAT is not restricted to I/O substitutions. You might, for example, have several versions of a subroutine, each stored under a different name. With EQUAT, any of these subroutines can be used without recompiling or reassembling the calling programs.

You must remember that the calling sequence of any substitute pair must be identical since the core load builder does no more than substitute one name for the other. Thus, CARDZ cannot be substituted for PRNZ because the 80-column count associated with CARDZ is incompatible with the 120-word count associated with PRNZ. The equatable FORTRAN I/O subroutines are:

1132 Printer	1403 Printer	2501 Card Reader	1442 Card Reader Punch	Console printer keyboard	1055 Punch 1134 Reader	1627 Plotter	Notes
PRNTZ	PRNZ	————	————	————	————	————	————
————	————	READZ	CARDZ	TYPEZ	————	————	Input only
————	————	————	————	TYPEZ	PAPTZ	*VCHRI,WCHRI	Output only
————	————	————	————	WRTYZ	PAPTX	*VCHRI,WCHRI	Output only

*VCHRI — extended precision
WCHRI — standard precision

The following lists the possible entries in a FORTRAN *IOCS control record and the subroutine each entry implies:

*IOCS entry	Subroutine called
CARD	CARDZ
2501 READER	READZ
1442 PUNCH	PNCHZ
TYPEWRITER	WRTYZ
KEYBOARD	TYPEZ
1132 PRINTER	PRNTZ
1403 PRINTER	PRNZ
PAPER TAPE	PAPTZ
PLOTTER	PLOTX
DISK	DISKZ
UDISK	DISKZ

The FORTRAN programmer should also remember that the *name of a function subroutine* as stored in the system library must be used in an EQUAT control record; not the function name that is coded in FORTRAN statements.

EQUAT can also be used to allow a FORTRAN program to overlap the operations of the 1132 Printer with the synchronous communication adapter (SCA). The operations of these I/O devices cannot be overlapped unless the 1132 is serviced by PRNT2. EQUAT can change PRNTZ (the subroutine used by FORTRAN I/O for 1132 printing) to the name PRTZ2 (a special subroutine to interface between PRNTZ and PRNT2). 1132 printing is then performed by PRNT2 and can be overlapped with the SCA.

Invalid Characters in FORTRAN Source Cards

Any invalid FORTRAN character in a FORTRAN source card is converted to an ampersand, causing the compiler to print an error message. The error message that is printed depends on the kind of statement in which the invalid character is found. The FORTRAN character set is listed in Appendix C of the publication *IBM 1130/1800 Basic FORTRAN IV Language,* GC26-3715.

FORTRAN Object Program Paper Tape Data Record Format

Data records of up to 80 EBCDIC characters in paper tape PTTC/8 code can be read or written by FORTRAN object programs. Delete and newline codes are recognized. Delete codes and case-shifts are not included in the 80 characters. When a newline code is read before the 80th character, the record is terminated. If the 80th character is not a newline code, the 81st character read is assumed to be a newline code.

Keyboard Input of Data Records During FORTRAN Program Execution

Data records of up to 80 characters can be read from the keyboard by a FORTRAN READ statement. Data values must be right justified in their respective fields.

keyboard operation

If you want to key in less than 80 characters, press EOF to stop transmittal. Also, pressing ERASE FIELD or the backspace key (←) allows you to reenter a record when you make a mistake during data entry. If the keyboard appears to be locked, press REST KB. Select the correct case shift before entering data.

buffer status after keyboard entry

The input buffer is filled with blanks before you enter a data record. Therefore, when you press EOF before you have entered 80 characters, the rest of the buffer remains blank. If more data is necessary to satisfy the list items in the DATA statement, the remaining numeric fields (I, E, or F) are stored in core as zeros, and alphameric fields (A or H) are stored as blanks. Processing is continuous; errors do not result from the previous condition.

Note. Information about buffer status after pressing ERASE FIELD or the backspace key (←) is under "Functions of Console Operator Keys During Monitor System Control" and "Entering Jobs from the Console Keyboard," respectively, in Chapter 7.

FORTRAN Program Control of the Console Printer

You can code your program to control spacing, tabulating, and shifting on the console printer by assigning unique values for desired operations to variables. These variables must be assigned as integers, and A-conversion must be used in the FORMAT statement for these variables.

The operations that can be performed and the values that are assigned to them include:

Operation	*Value*
Backspace	5696
Carrier return	5440
Line feed	9536
Shift to print black	5184
Shift to print red	13632
Space	16448
Tabulate	1344

As an example of console printer control, assume that a variable, X, is printed in the existing black ribbon shift and that another variable, Y, is printed in red after a tabulation. Following the printing of Y, the ribbon is shifted back to black. The following statements perform these functions:

```
      I=1344
      J=13632
      K=5184
      L=1
      WRITE (L,3)X,I,J,Y,K
  3   FORMAT (F12.6,2A1,F12.6,A1)
```

FORTRAN logical unit 1, as specified in the WRITE statement, is the console printer. The sequence of operations to be performed are:

- Print X
- Tabulate
- Shift to print red
- Print Y
- Shift to print black

Each control variable counts as one character and must be included in the count of the maximum line length.

Length of FORTRAN DATA Statement

An error (DATA statement too long to compile, due to internal buffering) occurs if:

$$(G_1 + G_2 + \ldots + G_n) > 355$$

where

N is the number of constants in this DATA statement.

Each *G* is a constant with the factor:

$$G = 1 + C + (K_1 + K_2 + \ldots + K_V)$$

where

C is the length in words of this constant and V is the number of variables loaded with this constant.

Each such variable has a factor of:

K = 1 for a nonsubscripted variable or K = 2 for a subscripted variable

// Records Read During FORTRAN Program Execution

Any //b̸ record read by CARDZ, READZ, or PAPTZ during a FORTRAN program execution causes an immediate CALL EXIT. Only the //b̸ characters are recognized by CARDZ, READZ, or PAPTZ. Any other data punched in this record is not available to programs in the monitor system, and the record is not printed. After the //b̸ record is read, the supervisor searches for the next valid monitor control record entered from the reader.

For offline listing purposes, however, this record can contain comments, such as // END OF DATA.

FORTRAN I/O Errors

If input/output errors are detected during execution, the program stops; do not continue execution. The error is indicated by a code displayed in the console ACCUMULATOR (see Appendix B for a list of the codes and their causes).

When an output field is too small to contain a number, the field is filled with asterisks and execution continues.

The I/O subroutines used by FORTRAN (PAPTZ, CARDZ, PRNTZ, WRTYZ, TYPEZ, PNCHZ, READZ, PRNZ) wait on any I/O device error or device not in a ready condition. Ready the device, and press PROGRAM START to continue.

Error detection in functional and arithmetic subroutines is possible by the use of source program statements. Refer to "Machine and Program Indicator Tests" in the publication *IBM 1130/1800 Basic FORTRAN IV Language,* GC26-3715.

Dumping FORTRAN DSF Programs to Cards

Arrays are always allocated backwards in core storage by the FORTRAN compiler. Because of this basic principal of the compiler, DSF output may be somewhat inefficient when dumped to cards if arrays are included in DATA statements. Such statements can cause cards to be punched with only one data word each.

To circumvent this inconvenience, write every element of an array explicitly in a DATA statement, starting with the element of the highest order.

RPG OBJECT PROGRAM CONSIDERATIONS

An RPG object program requires the special interrupt level subroutines (ILSs named with an X, as ILSX4). You code any character in column 28 of an XEQ monitor control record and in column 12 of a STORECI DUP control record to cause the special ILSs to be included in a core load. If the program is stored in core image (STORECI), the special ILSs are stored with the program on disk.

The storing of programs in disk core image format on disk is not recommended (see "Disadvantages of Storing a Program in DCI Format" in this chapter).

Chapter 7. Operating the 1130 Disk Monitor System

This chapter contains procedures that are used frequently during the operations of the 1130 Disk Monitor System. These procedures include:

- General procedures for readying the components of the 1130 for operation
- Procedures for performing a cold start of the monitor system
- General operating procedures that are used while the monitor system is in operation

The procedures for readying the 1130 components are performed when a device is to be used and is not ready. The central processing unit must be the first device readied as the console POWER switch, when turned on, supplies power to the entire 1130 computing system. The procedures for the I/O devices need not be performed in the order presented; however, if the disk drives are readied first, other devices can be readied while the disk drives are reaching operating speed. Detailed procedures for changing forms, tapes, and cartridges are not included here; they are in the publication *IBM 1130 Operating Procedures,* GA26-5717.

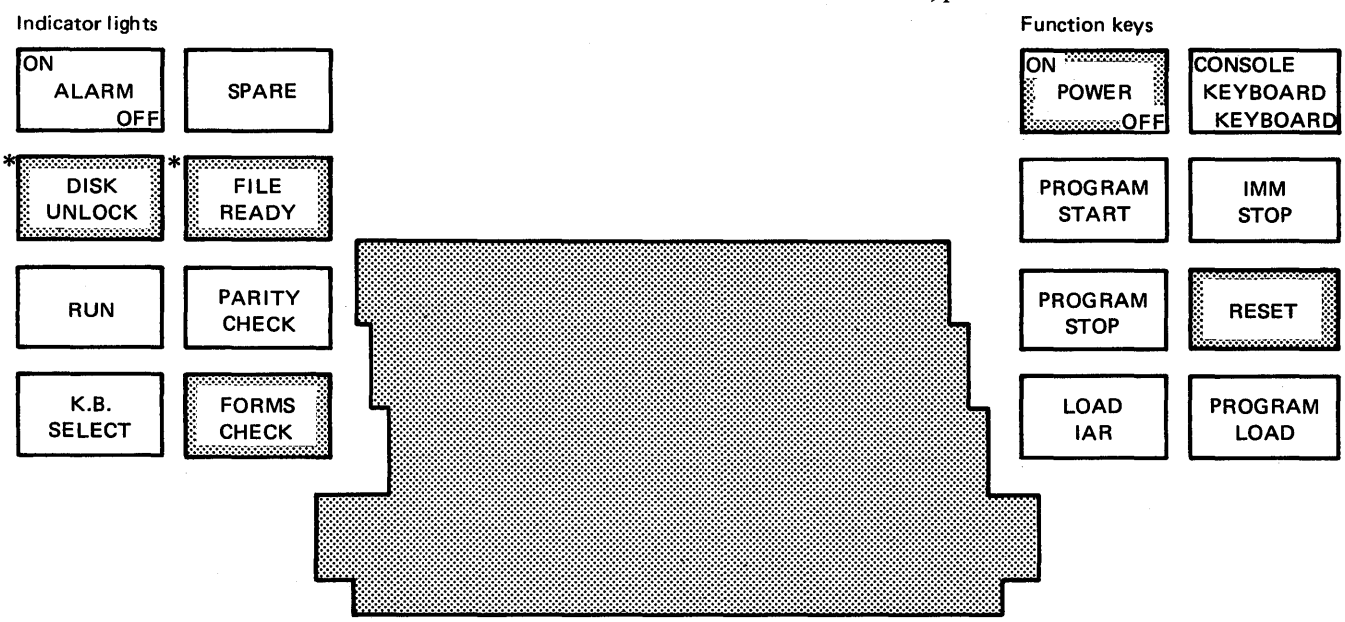
The functions of the cold start program and operating procedures for performing a cold start from cards or from paper tape are described in detail.

The procedures used while the monitor system is in operation are:

- Loading control records, program statements, and data records
- Controlling the system with the PROGRAM STOP, PROGRAM START, INT REQ, and IMM STOP function keys on the console
- Displaying and altering selected core storage locations
- Manually dumping core storage

READYING THE 1131 CENTRAL PROCESSING UNIT (with an internal disk)

Operator action	System response *or* Error indicator and corrective action
1. Move the console POWER switch to ON. This switch supplies power to the entire system, and must be on before any of the I/O devices are readied.	If the FORMS CHECK light comes on, insert or adjust the paper in the console printer.
	If the DISK UNLOCK light comes on, insert a cartridge in the single disk drive.
2. Move the DISK switch on the disk drive to ON. The disk drive requires approximately 90 seconds to reach operating speed.	The FILE READY light comes on when the disk drive reaches operating speed.
	If any other indicator lights on the console are on, press RESET.



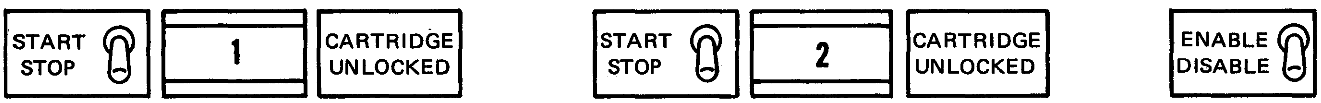
*These indicators are blank on an 1131 CPU that does not contain an internal single disk drive.

READYING THE 1131 CENTRAL PROCESSING UNIT (without an internal disk)

Operator action	System response *or* Error indicator and corrective action
1. Move the console POWER switch to ON. This switch supplies power to the entire system, and must be on before any of the I/O devices are readied.	If the FORMS CHECK light comes on, insert or adjust the paper in the console printer. If any other indicator lights on the console are on, press RESET.
2. Ready the 2311 Disk Storage Drives as described under "Readying the 2311 Disk Storage Drive" in this chapter.	

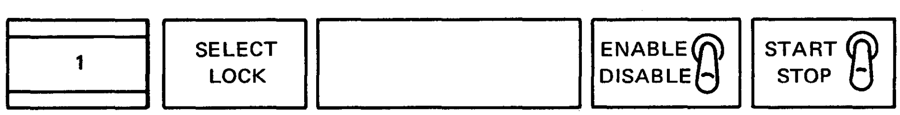
READYING THE 2310 DISK STORAGE DRIVE

Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	If the CARTRIDGE UNLOCKED lights on the disk drive operator's panel are on, insert disk cartridges.
2. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.	The READY light on the 1133 is on.
3. Move the START/STOP switch to the START position for the cartridges being used. The drives require approximately 90 seconds to reach operating speed.	The indicators showing the drive numbers come on when the disks reach operating speed.
4. Move the ENABLE/DISABLE switch on the disk storage drive to the ENABLE position.	



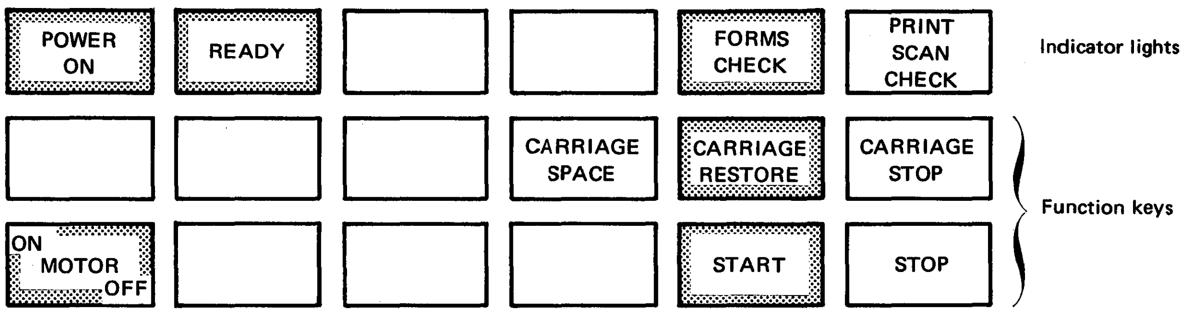
READYING THE 2311 DISK STORAGE DRIVE

Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	
2. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.	The READY light on the 1133 is on.
3. Insert a disk pack in the 2311, if necessary.	
4. Move the START/STOP switch to the START position. The disks require approximately 60 seconds to reach operating speed.	The green indicator showing the drive number comes on when the disks reach operating speed.
5. Move the ENABLE/DISABLE switch on the disk storage drive to the ENABLE position.	



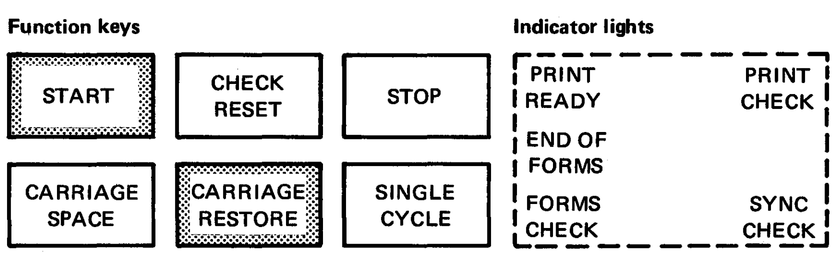
READYING THE 1132 PRINTER

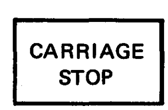
Operator action	System response *or* Error indicator and corrective action
1. Move the printer MOTOR switch to ON.	The printer POWER ON light comes on. If the printer FORMS CHECK light comes on, insert or adjust the paper in the printer.
2. Press CARRIAGE RESTORE.	
3. Press START.	The READY light comes on.



READYING THE 1403 PRINTER

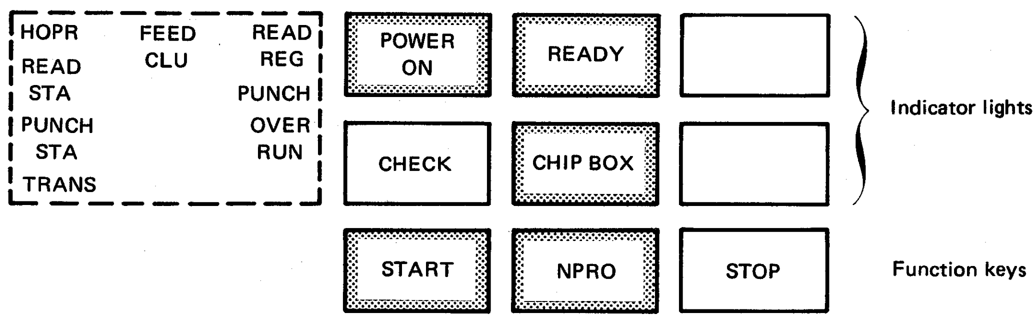
Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	If any indicator lights on the printer other than PRINT READY are on, correct the condition (see the publication *IBM 1130 Operating Procedures,* GA26-5717).
2. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.	The READY light on the 1133 is on.
3. Press the CARRIAGE RESTORE key on the printer.	
4. Press START.	The PRINT READY light comes on.





READYING THE 1442 MODEL 6 AND 7 CARD READ PUNCH

Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	The 1442 POWER ON and HOPR indicator lights are on. If the CHIP BOX light is on, empty the chip box. If any indicator lights other than HOPR are on, correct the condition (see Appendix B).
2. Press the NPRO key.	The HOPR light goes off.
3. Place the cards to be processed in the hopper, face down, 9-edge first.	
4. Press the START key.	The READY light comes on.



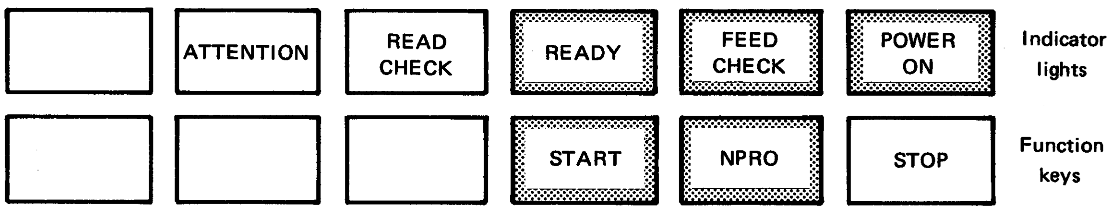
READYING THE 1442 MODEL 5 CARD PUNCH

Operator action

Follow the procedure for readying Models 6 and 7 with one exception; use blank cards in Step 3 rather than cards ready for processing.

READYING THE 2501 CARD READER

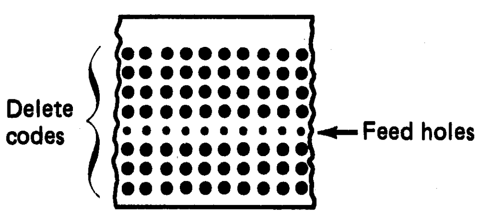
Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	The card reader POWER ON and FEED CHECK lights are on. If any other indicators are on, correct the condition (see Appendix B).
2. Press NPRO.	The FEED CHECK light goes off.
3. Place cards to be processed in the hopper, face down, 9-edge first.	
4. Press START.	The READY light comes on.

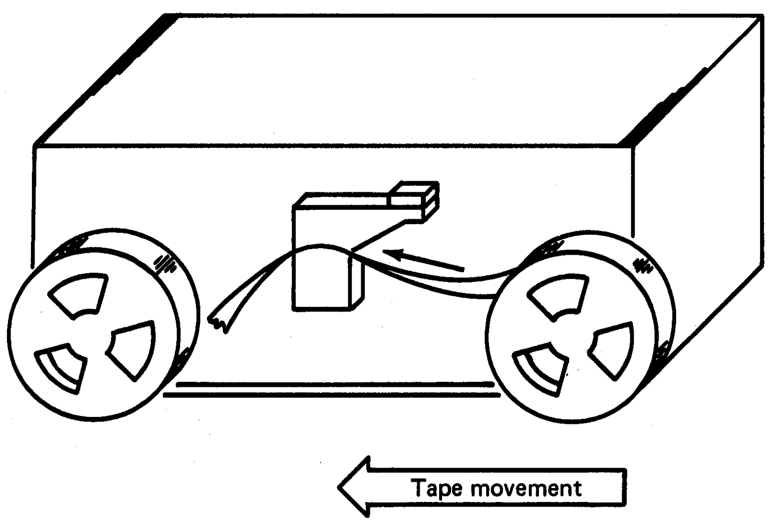


READYING THE 1134 PAPER TAPE READER

Operator action

1. Be sure system power is turned on.
2. Insert a tape to be processed in the paper tape reader; position under the read starwheels any of the delete codes that follow the program ID in the tape leader.





READYING THE 1055 PAPER TAPE PUNCH

Operator action

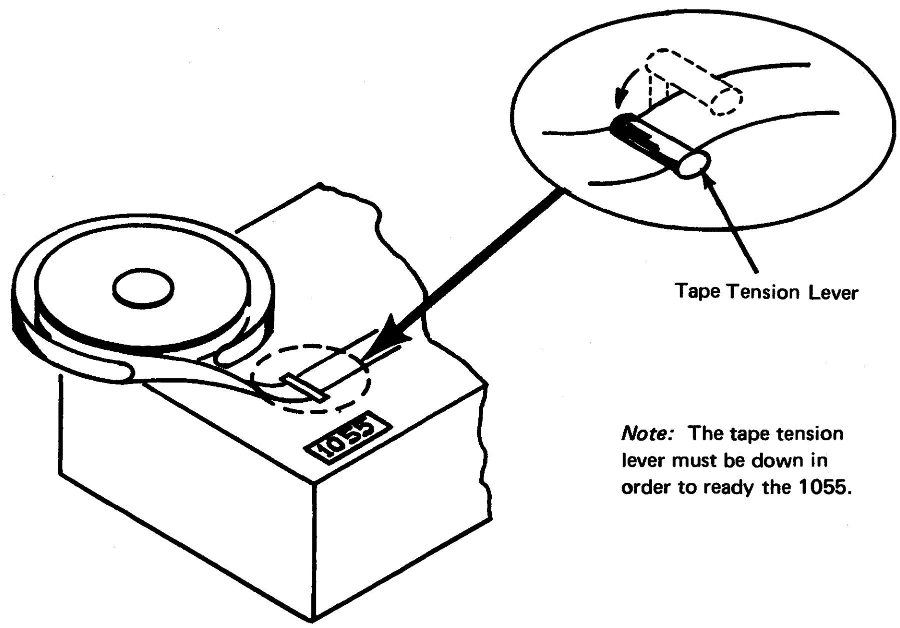
1. Be sure system power is turned on.

2. Insert a blank tape in the paper tape punch.

3. Press the DELETE key on the punch and hold down while performing Step 4. Do not release the DELETE key.

4. With the DELETE key held down, press the FEED key and hold down to punch several inches of delete codes.

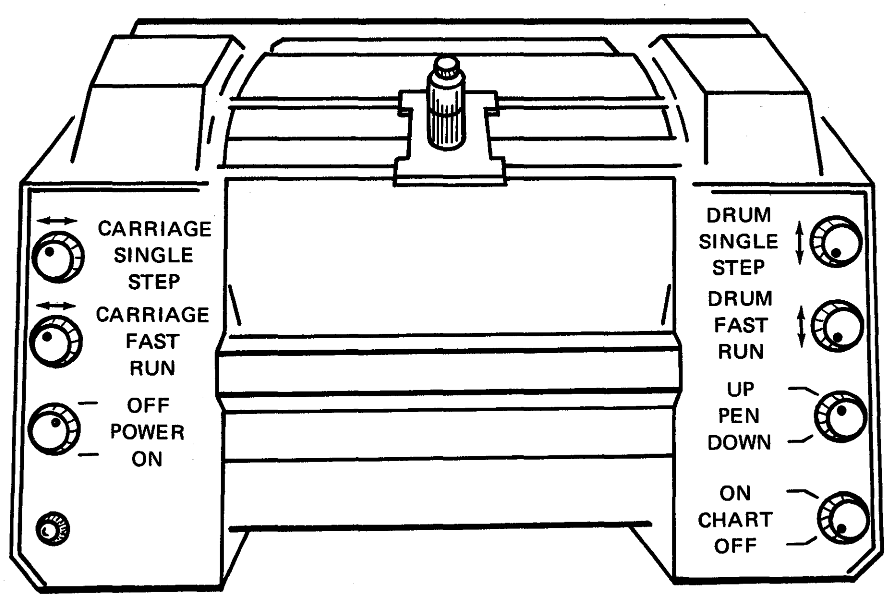
5. Release the FEED key *before* the DELETE key.



Note: The tape tension lever must be down in order to ready the 1055.

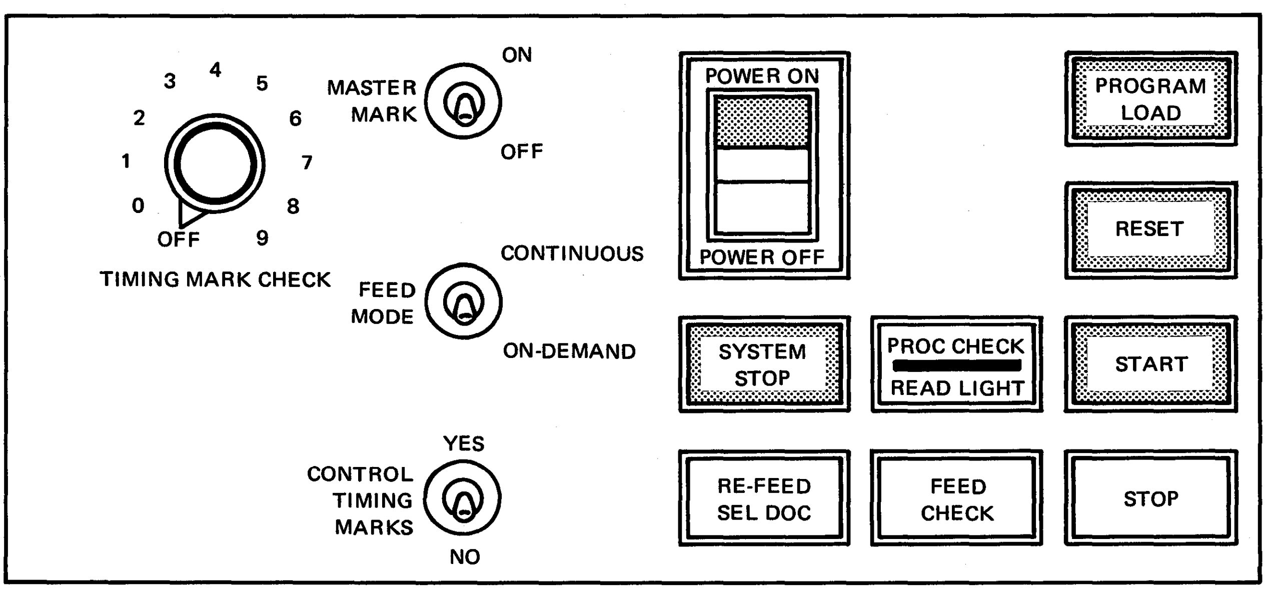
READYING THE 1627 PLOTTER

Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	
2. Turn the 1627 POWER switch to the ON position.	The POWER ON light comes on.
	If the pen is not in the up position, move the PEN switch first to DOWN, then to UP. If a single sheet of chart paper is used, be sure the CHART switch is in the OFF position.
3. With the pen in the UP position, use the 2 DRUM (X axis) and the 2 CARRIAGE (Y axis) controls to position the pen for the first plot.	



READYING THE 1231 OPTICAL MARK PAGE READER

Operator action	System response *or* Error indicator and corrective action
1. Be sure system power is turned on.	The 1231 POWER ON light is on.
2. Place the data sheets in the hopper with the side to be read facing up and the top edge positioned to feed first.	
3. Move the FEED MODE switch to ON-DEMAND.	
4. Press PROGRAM LOAD.	The PROGRAM LOAD light comes on.
5. Press RESET.	The hopper is raised to the ready position. The RESET light goes off and the START light comes on.
6. Press START.	The PROGRAM LOAD light goes off.
7. Press START again.	The START light goes off. All indicator lights should be off, with one exception: the SYSTEM STOP light can be on.



COLD START PROCEDURE

The cold start procedure is initiated when the cold start record is read by the card reader or the paper tape reader. This record causes the cold start program stored in cylinder 0 of the system cartridge to be read into core storage. The cold start program gains control and reads the resident image and the DISKZ subroutine from cylinder 0 into the resident monitor portion of low core storage. Program control is then assumed by the skeleton supervisor portion of the resident monitor.

During the cold start program, a dummy // JOB control record is printed on the principal printer, and the following cartridge status information is printed:

```
LOG DRIVE    CART SPEC    CART AVAIL    PHY DRIVE
  XXXX         XXXX         XXXX          XXXX

VX MXX     ACTUAL XXK     CONFIG XXK
```

where

LOG DRIVE is always a single entry of zero.

CART SPEC is the cartridge ID written on the system cartridge when initialized.

CART AVAIL is the same as CART SPEC. When more than one disk drive is on the computer, the IDs of any other disk cartridges that are ready are also listed.

PHY DRIVE is the physical drive number you enter in the console entry switches. This drive is also logical drive zero. When more than one disk drive is on the computer, the physical drive numbers of any other disk cartridges that are ready are also listed.

VX MXX is the version and modification of the monitor system on the current system cartridge.

ACTUAL XXK is the physical core size of the 1130.

CONFIG XXK is the configured core size on the system cartridge.

Note. The monitor system is not supported unless the physical core size at least equals the configured core size.

The monitor system is now operational and is ready to receive the first JOB monitor control record.

Note. If your system has only one disk drive (the internal disk in the 1131 CPU or one 2311), you should cold start after changing cartridges, or packs, to avoid possible errors in the location of disk areas on system cartridges.

If an attempt is made to cold start a nonsystem cartridge, an error message (THIS IS A NONSYSTEM CARTRIDGE or NONSYS. CART. ERROR) is printed on the console printer. Error stops can occur during the cold start procedure. They are listed and explained under "Cold Start Program Error Waits" and "ISS Subroutine Preoperative Error Waits" in Appendix B.

Note. Do not perform a cold start with an uninitialized cartridge online.

The cold start procedure is started from the card reader or the paper tape reader as described in the following procedures.

Card System Cold Start Procedure

1. Ready the devices to be used.
2. If your 1130 has only one disk drive, be sure all console entry switches are off. For systems with more than one disk drive, be sure switches 0 through 11 are off; set switches 12 through 15 to the drive number (in binary) of the physical drive that contains the system cartridge:

   Drive 0–Switches 12 through 15 off
   Drive 1–Switch 15 on
   Drive 2–Switch 14 on
   *Drive 3–Switches 14 and 15 on
   *Drive 4–Switch 13 on
   Drive 5–Switches 13 and 15 on
   Drive 6–Switches 13 and 14 on
   Drive 7–Switches 13, 14, and 15 on
   *Drive 8–Switch 12 on
   *Drive 9–Switches 12 and 15 on
   Drive 10–Switches 12 and 14 on
   *Not used on a 2311 Disk Storage Drive, Model 12

3. Place the cold start card in the card reader wired for cold start. Then place cards to be processed in the card reader.
4. Press START on the card reader. (If both a 2501 and a 1442, Model 6 or 7, are present, make the reader wired for cold start ready and make sure the other reader is not ready by pressing STOP.)
5. Press IMM STOP on the console.
6. Press RESET on the console.
7. Press PROGRAM LOAD on the console.

Paper Tape System Cold Start Procedure

1. Ready the devices to be used, except the paper tape reader.
2. If your 1130 has only one disk drive, be sure all console entry switches are off. For systems with more than one disk drive, be sure switches 0 through 11 are off; set switches 12 through 15 to the drive number (in binary) of the physical drive that contains the system cartridge as follows:

   Drive 0–Switches 12 through 15 off
   Drive 1–Switch 15 on
   Drive 2–Switch 14 on
   Drive 3–Switches 14 and 15 on
   Drive 4–Switch 13 on

3. Insert tape BP15, cold start paper tape record, in the paper tape reader. Position under the read starwheels one of the delete codes after the program ID.
4. Press IMM STOP on the console.
5. Press RESET on the console.
6. Press PROGRAM LOAD on the console.

USING THE 1130 WITH THE MONITOR SYSTEM

When the I/O devices required for a job are online and ready, and the monitor system is running, jobs can be entered from the card reader, the paper tape reader, or the console keyboard. The following procedures describe how jobs are entered.

Entering Jobs from the Card Reader

1. Place the cards to be processed in the card hopper, face down, 9-edge first, and press START on the card reader.
2. Check that the console mode switch is set to RUN.
3. Press PROGRAM START on the console.
4. When the last card is indicated (hexadecimal /1000 for the 1442 Card Reader or /4000 for the 2501 Card Reader) in the ACCUMULATOR on the console display panel, press START on the card reader and PROGRAM START on the console so that the last card is released. This step need not be done if blank cards follow the last card processed.

Entering Jobs from the Paper Tape Reader

1. Insert the tape to be processed in the paper tape reader. Position under the read starwheels one of the delete codes after the program ID.
2. Check that the console mode switch is set to RUN.
3. Press PROGRAM START on the console.

Entering Jobs from the Console Keyboard

A single monitor control record or an entire program including all required control records and data records can be entered from the console keyboard. Monitor control is transferred to the keyboard when a // TYP monitor control record is read from the principal input device.

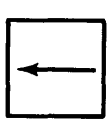
Control is returned to the principal input device when a // TEND monitor control record is entered from the keyboard. The formats of these 2 control records are described in Chapter 5 under "Monitor Control Records."

starting keyboard operation

When the // TYP control record is read, the console printer performs a carrier return and the KB SELECT light on the keyboard operator's panel comes on. The system is ready to accept input from the keyboard.

Enter all control records, program statements, and/or data records in their correct format. Use the space bar for blanks. As each character is entered, it is printed on the console printer. Press EOF to indicate the end of each line. When this key is pressed, an NL (new line) character is placed in the next character position of the input buffer, and the typing element is returned to the left margin of the next line.

Up to 80 characters can be entered in one line through the console keyboard. If an error is made during entry of a line, you can either backspace to correct the error or erase the entire line and reenter it.



When the TYPEO I/O subroutine is being used, a line is corrected during entry by pressing the backspace (←) key as many times as required until you reach the first character that has to be corrected. The first time that you press the backspace key, the last character printed on the console printer is slashed. The location address of the next character to be entered in the input buffer is decremented by one each time the backspace key is pressed.

For example, assume that you have entered *DELET and want to change it to *DEFINE.

1. Press the backspace key 3 times. (The T is slashed: *DELE~~T~~.)
2. Enter the correct characters. (The corrected line appears as *DELE~~T~~FINE on the console printer. The input buffer now contains *DEFINE; the characters FIN replace LET in the buffer.)

Note. When the TYPEZ I/O subroutine is being used, the backspace key functions the same as the ERASE FIELD key.

ERASE FIELD

A line can be erased when you press ERASE FIELD. This key signals an interrupt response subroutine that the previously entered characters are incorrect and are being reentered. Two slashes are printed on the console printer (when the TYPEO I/O subroutine is being used), and the typing element is returned to the left margin of the next line. The correct characters that you enter replace the previously entered characters in the input buffer. The previous message is not deleted from the input buffer; if the previous message is longer than the new one, the characters from the previous message remain (following the NL character that terminates the new message).

Note. When the TYPEZ I/O subroutine is being used, the two slashes are not printed when ERASE FIELD is pressed.

REST KB

If the keyboard appears to be locked (keys cannot be pressed), press REST KB (the restore keyboard key). The correct case shift must be selected before data is entered.

stopping keyboard operation

Continue entering control records, program statements, and/or data records as just described until all are entered. Then enter a // TEND control record, and press EOF. Control is returned to the principal input device.

Functions of Console Operator Keys During Monitor System Control

PROGRAM STOP

Pressing PROGRAM STOP causes an interrupt of the monitor system programs. This is a level 5 interrupt and causes an entry to the PROGRAM STOP key trap in the skeleton supervisor, if no user-written subroutines are associated with level 5.

If a higher interrupt level is being serviced when you press PROGRAM STOP, the PROGRAM STOP interrupt is masked until the current operation is complete.

The PROGRAM STOP key trap consists of a wait and a branch. Execution of the monitor system programs is continued when you press PROGRAM START. The status of the monitor system and of core storage is not changed when the system is stopped with the PROGRAM STOP key.

PROGRAM START

Pressing PROGRAM START also continues execution of the monitor system programs from ISS subroutine waits. A code in the ACCUMULATOR on the console display panel indicates the reason for the wait. ISS subroutine waits and their causes are listed in Appendix B.

INT REQ

Pressing the interrupt request (INT REQ) key immediately terminates the current job. System control returns to the supervisor, which searches through the input stream for the next JOB monitor control record. You have the option of programming this key for a different use (see Chapter 6. "Programming Tips and Techniques"). Portions of the monitor system that cannot be interrupted before completion, such as SYSUP, delay the interrupt until the operation is complete when INT REQ is pressed.

IMM STOP

Pressing the immediate stop (IMM STOP) key immediately stops processing.

Note. Do not press IMM STOP when the monitor system is running. The contents of a system cartridge can be destroyed, necessitating a reload of the system.

Displaying or Altering the Contents of a Selected Core Location

select a core location

To select a specific core location:

1. Press PROGRAM STOP on the console.
2. Turn the console mode switch to LOAD.
3. Set the console entry switches to the desired 4-character hexadecimal core address. Switches 0 through 3 represent the first hexadecimal character, 4 through 7 the second, 8 through 11 the third, and 12 through 15 the fourth.
4. Press LOAD IAR on the console. The selected address is loaded into the IAR and is displayed in the INSTRUCTION ADDRESS indicator on the console display panel.

display contents of the location

To display the contents of the selected core location:

1. Turn the console mode switch to DISPLAY.
2. Press PROGRAM START. The contents are displayed in the STORAGE BUFFER indicator on the console display panel. Repeatedly pressing PROGRAM START displays the contents of consecutive core locations.

alter contents of location

To alter the contents of the selected core location:

1. Set the new contents (in hexadecimal) in the console entry switches.
2. Turn the console mode switch to LOAD.
3. Press PROGRAM START.

return to system control

After the contents of the selected core location have been displayed and/or altered, return to system control:

1. Turn the console mode switch to RUN.
2. Press PROGRAM START. Execution begins at the location specified in the IAR.

Manual Dump of Core Storage

When a problem occurs during the execution of a core load and a dump of core storage is needed, you can execute a manual dump of core storage:

1. Press PROGRAM STOP.
2. Turn the console mode switch to LOAD.
3. Set the address plus one of the dump entry point ($DUMP+1) to the skeleton supervisor in the console entry switches.
4. Press LOAD IAR on the console.
5. Turn the console mode switch to RUN.
6. Press PROGRAM START.

A dump of the contents of core storage is printed in hexadecimal, then the dump program (see "Disk-Resident Supervisor Programs" in Chapter 3) executes a CALL EXIT to terminate execution of the core load in progress.

If the $IOCT, $DBSY, or $SCAT indicators in the resident monitor are nonzero when the branch to $DUMP+1 is made, the skeleton supervisor begins a loop testing these indicators. When this occurs:

1. Press PROGRAM STOP.
2. Display, and change to zero if necessary, the contents of each of these locations.
3. Restart the manual dump of core storage.

Chapter 8. Monitor System Initial Load and System Reload

initial load

An initial load is the process of loading the complete disk monitor system onto an initialized disk cartridge. An initial load is performed when:

- An 1130 computing system is installed
- Data contained on a system cartridge has been destroyed making the disk unuseable
- The assembler and/or any of the compilers are to be loaded onto a system cartridge

reload

A system reload is the process of loading modifications to the disk monitor system onto a system cartridge. A system reload is performed when:

- Existing phases of system programs are being added or expanded
- New system programs are being added
- The I/O device configuration is being changed

Any combinations of the previous functions can be performed during a reload. The following should be kept in mind when preparing to perform a reload:

- The cushion area must be large enough to absorb the increased length of system programs when they are added or expanded.
- Program additions must follow the last system program currently on the cartridge. Working storage must be equal to or larger than the length of the program being added, plus 31 sectors.
- System configuration is performed each time a system reload is performed. Reconfiguration is necessary when a system cartridge is copied from a system with a different configuration.

Initial load and reload procedures are performed with IBM-supplied system loaders, control records, system programs, and with control records that you punch. The information supplied by IBM is contained on paper tapes for paper tape systems and on disk cartridges for card systems. The contents of the disk cartridge must be dumped to cards before the system can be loaded. A preload operating procedure for dumping the monitor system to cards is contained in this chapter.

This chapter:

1. Describes the general functions and contents of IBM-supplied control records
2. Discusses the general functions, formats, ana uses of the control records that you must punch
3. Presents sample operating procedures for punching paper tape control records, performing a card system preload, initial load, and reload, and performing a paper tape system initial load and reload

You may use these operating procedures as they are presented, or you may modify them to meet the needs of your computing system. For those who are already familiar with similar procedures, the headings in each block can be used as reminders as you perform the procedure. For those who need more information, detailed steps for performing these procedures are provided. Not all steps of each procedure need to be done every time it is used; do only those steps that are necessary.

Appendixes A and B contain descriptions of error messages and halt codes that can occur during the operations of any of the initial load and reload procedures.

IBM-SUPPLIED SYSTEM LOADER CONTROL RECORDS

The IBM-supplied control records for initial load and reload operations are:

- SCON and TERM (for card systems only)
- Phase identification (PHID)
- Type 81

These control records must be used in all initial load and reload operations. The placement of these control records in the card decks and paper tapes is illustrated at the beginning of each of the procedures for load and reload at the end of this chapter.

The general functions and formats of these control records are discussed in the following text.

SCON and TERM Control Records

general function

These control records, together with the REQ control records that you punch, comprise the system configuration control record. They define the beginning and ending of the system configuration control record. A system configuration control record must be included in an initial load, a reload, and a configure operation.

SCON and TERM cards are included with the information supplied from IBM for card systems. For a paper tape system, you punch the SCON and TERM control records in the system configuration tape as described in "Preparation of Load Mode and System Configuration Control Tapes" in this chapter.

SCON and TERM control record formats

Card column	Contents
1 through 4	SCON *or* TERM
5 through 80	Blanks

Phase Identification (PHID) Control Records

general function

Each monitor system program, except the resident monitor and the cold start program, is divided into several parts called phases. PHID control records contain the beginning and ending phase ID numbers of the programs in the monitor system. All numbers in the ID fields of the PHID control records are in ascending sequence and in the order in which the system programs are loaded onto a disk. The ID entries in the PHID control record are loaded into the system location equivalence table (SLET), a directory to the disk locations of the monitor system programs.

When system programs are added or modified during a reload, the PHID control record must be changed to reflect any new phase ID limits of the programs and/or phases.

format of first PHID card

Card column	Contents
1 through 4	PHID
6 through 8 and 10 through 12	IDs of the first and last phases of DUP
14 through 16 and 18 through 20	IDs of the first and last phases of the FORTRAN compiler
22 through 24 and 26 through 28	IDs of the first and last phases of the COBOL compiler program product
30 through 32 and 34 through 36	IDs of the first and last phases of the supervisor
38 through 40 and 42 through 44	IDs of the first and last phases of the core load builder
46 through 48 and 50 through 52	IDs of the first and last phases of the system I/O device subroutines
54 through 56 and 58 through 60	IDs of the first and last phases of the core image loader
64	*1* (indicates continuation to the second PHID card)
66 through 68	*Vxx* (where *xx* is the disk monitor system version number)
70 through 72	*Mxx* (where *xx* is the version modification number)
73 through 80	Card identification and sequence number

Note: All card columns omitted in this format contain blanks.

format of second PHID card

Card column	Contents
1 through 4	PHID
6 through 8 and 10 through 12	IDs of the first and last phases of the RPG compiler
14 through 16 and 18 through 20	IDs of the first and last phases of DUP, part 2
22 through 24 and 26 through 28	IDs of the first and last phases of the macro assembler
29 through 65	Blanks
66 through 68	*Vxx* (where *xx* is the disk monitor system version number)
70 through 72	*Mxx* (where *xx* is the version modification number)
73 through 80	Card identification and sequence number

Note: All card columns omitted in this format contain blanks.

If you have a paper tape system, the IBM-supplied PHID control record is on tape BP03.

System Program Sector Break Cards (Card Systems)

In order to allow you to load only a portion of a monitor program during a card system reload, each program phase is preceded with a sector break card that identifies the phase. These cards have a 1 punch in column 4, and the monitor system version and modification level are punched in the cards starting in column 67 (VxMxx). A description of the function of sector break cards is in Appendix I.

The following is a list of the monitor system sector break cards.

Phase number	Program or program phase name		ID starting in column 73
XX	RES SKELETON SUPY, COMMA, DISKZ, COLD START PROGRAM	Part of system loader	EMN
XX	SYS LDR–PHASE 2–OVERLAY 0		FP2
XX	SYS LDR–PHASE 2–OVERLAY 1		FP2
XX	SYS LDR–PHASE 2–OVERLAY 2		FP2
XX	SYS LDR–PHASE 2–OVERLAY 3		FP2
	DUP		
01	DUP COMMON SUBROUTINES, CCAT		J01
02	DUP CTRL RECORD PROCESSOR		J02
03	DUP STORE PHASE		J03
04	DUP *FILES, *LOCAL, *NOCAL PHASE		J04
05	DUP DUMP PHASE		J05
06	DUP DUMP LET/FLET PHASE		J06
07	DUP DELETE PHASE		J07
08	DUP DEFINE PHASE		J08
09	DUP EXIT PHASE		J09
0A	DUP CARD I/O INTERFACE		J10
0B	DUP KEYBOARD INPUT INTERFACE		J11
0C	DUP PAPER TAPE I/O INTERFACE		J12
0D	DUP UPCOR PHASE SAVED BY DEXIT DURING STORECI		J17
0E	DUP PRINCIPAL INPUT WITH KEYBOARD		J17
0F	DUP PRINCIPAL W/O KEYBOARD		J17
10	DUP PAPER TAPE I/O		J17
11	DUP STORE CI		J17
12	DUP MODIF DUMMY PHASE		J17
	FORTRAN compiler		
1F	FOR INPUT PHASE		K01
20	FOR CLASSIFIER PHASE		K02
21	FOR CHECK ORDER/STMNT NO. PHASE		K03
22	FOR COMMON SUBR OR FUNCTION PHASE		K04
23	FOR DIMENSION, REAL, INTEGER		K05
24	FOR REAL CONSTANT PHASE		K06
25	FOR DEFINE FILE, CALL LINK EXIT		K07
26	FOR VARIABLE, STMNT FUNC PHASE		K08
27	FOR DATA STATEMENT PHASE		K09
28	FOR FORMAT STATEMENT PHASE		K10
29	FOR SUBTRACT DECOMPOSITION PHASE		K11
2A	FOR ASCAN I PHASE		K12
2B	FOR ASCAN II PHASE		K13
2C	FOR DO, CONTINUE, ETC. PHASE		K14
2D	FOR SUBSCRIPT OPTIMIZE PHASE		K15
2E	FOR SCAN PHASE		K16
2F	FOR EXPANDER I PHASE		K17

Phase number	Program or program phase name	ID starting in column 73
30	FOR EXPANDER II PHASE	K18
31	FOR DATA ALLOCATION PHASE	K19
32	FOR COMPILATION ERROR PHASE	K20
33	FOR STATEMENT ALLOCATION PHASE	K21
34	FOR LIST STATEMENT ALLOCATION	K22
35	FOR LIST SYMBOL TABLE PHASE	K23
36	FOR LIST CONSTANTS PHASE	K24
37	FOR OUTPUT I PHASE	K25
38	FOR OUTPUT II PHASE	K26
39	FOR RECOVERY (EXIT) PHASE	K27
	COBOL compiler (program product)	
51 . . 5C	PHASE NUMBERS USED BY THE COBOL COMPILER	
	Supervisor	
6E	SUP PHASE 1–MONITOR CONTROL RECORD ANALYZER	N01
6F	SUP PHASE 2–JOB CONTROL RECORD PROCESSOR	N01
70	SUP PHASE 3–DELETE TEMPORARILY STORED PROGRAM LET	N01
71	SUP PHASE 4–XEQ CONTROL RECORD PROCESSOR	N01
72	SUP PHASE 5–SUPERVISOR CONTROL RECORDS PROCESSOR	N01
73	SYSTEM DUMP-CORE-TO-PRINTER	N02
74	AUXILIARY SUPERVISOR	N03
	Core load builder	
78	CORE LOAD BUILDER, PHASE 0/1	OCB
79	CORE LOAD BUILDER, PHASE 2	OCB
7A	CORE LOAD BUILDER, PHASE 3	OCB
7B	CORE LOAD BUILDER, PHASE 4	OCB
7C	CORE LOAD BUILDER, PHASE 5	OCB
7D	CORE LOAD BUILDER, PHASE 6	OCB
7E	CORE LOAD BUILDER, PHASE 7	OCB
7F	CORE LOAD BUILDER, PHASE 8	OCB
80	CORE LOAD BUILDER, PHASE 9	OCB
81	CORE LOAD BUILDER, PHASE 10	OCB
82	CORE LOAD BUILDER, PHASE 11	OCB
83	CORE LOAD BUILDER, PHASE 12	OCB
84	CORE LOAD BUILDER, PHASE 13	OCB

Phase number	Program or program phase name	ID starting in column 73
	System device subroutines, disk I/O	
8C	SYS 1403	PMN
8D	SYS 1132	PMN
8E	SYS CONSOLE PRINTER	PMN
8F	SYS 2501	PMN
90	SYS 1442	PMN
91	SYS 1134	PMN
92	SYS KEYBOARD	PMN
93	SYS 2501/1442 CONVERSION	PMN
94	SYS 1134 CONVERSION	PMN
95	SYS KEYBOARD CONVERSION	PMN
96	DISKZ	PMN
97	DISK1	PMN
98	DISKN	PMN
	Core image loader	
A0	CORE IMAGE LOADER, PHASE 1	PMN
A1	CORE IMAGE LOADER, PHASE 2	PMN
	RPG compiler	
B0	RESIDENT	PR1
B1	ENTER FILES	PR2
B2	ENTER INPUT	PR3
B3	ENTER CALCULATION	PR4
B4	ENTER OUTPUT	PR5
B5	ASSIGN INDICATORS	PR6
B6	ASSIGN FIELD NAMES	PR7
B7	ASSIGN LITERALS	PR8
B8	EXTENDED FILE AND INPUT DIAGNOSTIC	PR9
B9	EXTENDED CALCULATION AND OUTPUT DIAGNOSTIC	PRA
BA	DIAGNOSTIC MESSAGE 1	PRB
BB	DIAGNOSTIC MESSAGE 2	PRC
BC	DIAGNOSTIC MESSAGE 3	PRD
BD	ASSEMBLE 1 I/O	PRE
BE	ASSEMBLE 2 I/O	PRF
BF	ASSEMBLE 3 I/O	PRG
C0	ASSEMBLE 4 I/O	PRH
C1	ASSEMBLE TABLES	PRJ
C2	ASSEMBLE CHAIN AND RAF	PRK
C3	ASSEMBLE INPUT FIELDS	PRL
C4	ASSEMBLE CONTROL LEVELS	PRM
C5	ASSEMBLE MULTI FILE LOGIC	PRN
C6	ASSEMBLE GET ROUTINES	PRO
C7	ASSEMBLE CALCULATIONS 1	PRP
C8	ASSEMBLE CALCULATIONS 2	PRQ
C9	ASSEMBLE OUTPUT FIELDS	PRR
CA	ASSEMBLE PUT ROUTINES	PRS
CB	ASSEMBLE FIXED DRIVER	PRT
CC	TERMINATE COMPILATION	PRU
	DUP part 2	
CD	DUP CTRL—PART 2	PS0
CE	MACRO UPDATE PROGRAM	PS1

Phase number	Program or program phase name	ID starting in column 73
	Assembler	
CF	ASM INITIALIZATION PHASE	PTM
D0	ASM CARD CONVERSION PHASE	PTM
D1	ASM DSF OUTPUT PHASE	PTM
D2	ASM INTERMEDIATE INPUT PHASE	PTM
D3	ASM END STATEMENT PHASE	PTM
D4	ASM ASSEMBLY ERROR PHASE	PTM
D5	ASM CONTROL CARDS 1	PTM
D6	ASM CONTROL CARDS 2	PTM
D7	ASM DUMMY PHASE (SYST SYMBOL TBL)	PTM
D8	ASM SYMBOL TABLE OPTIONS PHASE	PTM
D9	ASM EXIT PHASE	PTM
DA	ASM PROG HEADER MNEMONICS PHASE	PTM
DB	ASM FILE STATEMENT PHASE	PTM
DC	ASM COMMON SUBROUTINES, ASCOM	PTM
DD	ASM PROG CONTROL MNEMONICS PHASE	PTM
DE	ASM IMPERATIVE STATEMENTS PHASE	PTM
DF	ASM DECML EFLC PROCESSING PHASE	PTM
E0	ASM DECIMAL CONVERSION PHASE	PTM
E1	ASM PROG LINKING PHASE	PTM
E2	ASM DMES PROCESSING PHASE	PTM
E3	ASM PUNCH CONVERSION PHASE	PTM
E4	ASM INTERMEDIATE DISK OUTPUT	PTM
E5	ASM SYMBOL TABLE OVERFLOW	PTM
E6	ASM G2250 PH1	PTM
E7	ASM DIVISION OPERATOR PHASE	PTM
E8	ASM CONTROL CARDS 3	PTM
E9	ASM MACRO PHASE 1—SPECIAL OP AND PREPROCESSING	PTM
EA	ASM MACRO PHASE 1A—SPECIAL PSEUDO OPS	PTM
EB	ASM MACRO PHASE 1B—CONDITIONAL ASSEMBLY	PTM
EC	ASM MACRO PHASE 2—MACRO DEFINITION	PTM
ED	ASM MACRO PHASE 2A—MACRO DEFINITION	PTM
EE	ASM MACRO PHASE 2B—MACRO DEFINITION	PTM
EF	ASM MACRO PHASE 3—EXPANSION	PTM
F0	ASM MACRO PHASE 3A—EXPANSION	PTM
F1	ASM MACRO PHASE 3B—EXPANSION	PTM
F2	ASM CROSS REFERENCE—PART 1	PTM
F3	ASM CROSS REFERENCE—PART 2A	PTM
F4	ASM CROSS REFERENCE—PART 2B	PTM
F5	ASM CROSS REFERENCE—PART 2C	PTM
F6	ASM CROSS REFERENCE—PART 3	PTM

Type 81 Control Record

general function

The type 81 control record defines the end of the loading of the monitor system programs and/or phases. After the type 81 control record is read, a record of the principal print device and the principal I/O devices is placed in the system location equivalence table (SLET). (Principal I/O devices are discussed under "System Configuration Control Records" in this chapter.) Also during an initial load, the disk communications area (DCOM) and location equivalence table (LET) are initialized, and the reload table is established.

format of type 81 control record

Card column	Contents
1 and 2	Blanks
3	A 6 punch
4	A 1 punch
5 through 80	Blanks

Note. These punches are /8100 in card data format (CDD) in word 3, thus, the name type 81.

If reconfiguration is all that is being done by a reload operation, place the type 81 control records immediately after the PHID control record.

SYSTEM LOADER CONTROL RECORDS THAT YOU PUNCH

The control records that you punch for initial load and reload operations are:

- Load mode that defines whether the operation is an initial load or a reload
- System configuration that defines the I/O devices of your system
- CORE (optional) that allows you to define a core size other than the actual core size of the computer

The general functions, formats, and uses in initial load and reload operations for these control records are described in the following text.

Note. When the 1627 Plotter is used by a program, the following subroutines must not be in a SOCAL for that program: EADD, FADD, FMPY, EMPY, XMD, XMDS, and FARC. These must instead be incore subroutines. You can accomplish this during a system load by storing the programs with subtype zero.

Load Mode Control Record

general function

The load mode control record informs the system loader whether the operation is an initial load or a reload. This control record can also be used to bypass the assembler, FORTRAN compiler, COBOL compiler, or RPG compiler during an initial load or reload.

format

Card column	Contents	Explanation
1 through 4	MODE	
5 through 7	Blanks	
8	I or R	*I* indicates initial load. *R* indicates reload.
9 through 11	Blanks	
12	A or blank	*A* indicates the assembler is not being loaded. Blank indicates the assembler is being loaded.
13	F or blank	*F* indicates the FORTRAN compiler is not being loaded. Blank indicates the FORTRAN compiler is being loaded.
14	R or blank	*R* indicates the RPG compiler is not being loaded. Blank indicates the RPG compiler is being loaded.
15	C or blank	*C* indicates the COBOL compiler (a program product) is being loaded. Blank indicates the COBOL compiler is not being loaded.
16 through 80	Blanks	

Note. If the assembler or the FORTRAN, RPG, or COBOL compiler is not loaded in an initial load or was deleted by a DUP DEFINE VOID operation, they can be loaded by an initial load operation only. Columns 12, 13, and 14 must contain A, F, or R, respectively, and column 15 must be blank for a reload operation to reflect the status of the cartridge.

card system use

For a card system, a load mode control card is placed in an initial load or reload card deck immediately behind the first part of the system loader. The order of cards for an initial load and reload is illustrated in Figures 8-2 and 8-4 under "Card System Initial Load Operating Procedure" and "Card System Reload Operating Procedure," respectively, in this chapter.

paper tape system use

For a paper tape system, this control record is entered between the IBM-supplied tapes, BP01 and BP03, as illustrated in Figures 8-7 and 8-9 under "Paper Tape System Initial Load Operating Procedure" and "Paper Tape System Reload Operating Procedure" in this chapter. A procedure for punching a load mode control tape is included under "Preparation of Load Mode and System Configuration Control Tapes" in this chapter.

System Configuration Control Records

general function

System configuration control records (REQ) allow you to define the system I/O devices that are a part of your computer system. Punch one control record for each device. Missing or extra REQ records may cause initial load operations to fail.

format

Device	Card columns: 1 through 3	9 and 10[1]	15 through 20	
1442 Card Read/Punch Card Punch	REQ	1	1442-5 1442-6 1442-7	whichever is applicable
Paper Tape Reader and/or Punch	REQ	3	1134	Unit ID is optional
2501 Card Reader	REQ	4	2501	
1132 Printer	REQ	6	1132	
1403 Printer	REQ	9	1403	

Note. I/O devices not listed are initialized as part of the system; REQ control records are not required. If an REQ control record is punched for a 1442, columns 15 through 20 must be coded to indicate the model.

[1] ISS numbers, right justified. Maximum entry number ISS 20.

card system use

For a card system, REQ cards are placed in an initial load or reload card deck between the IBM-supplied SCON and TERM cards. If the optional CORE card is used, it must be placed before or after the REQ cards, not between any of them. The order of cards for an initial load and reload is illustrated in Figures 8-2 and 8-4 under "Card System Initial Load Operating Procedure" and "Card System Reload Operating Procedure," respectively, in this chapter.

paper tape system use

For a paper tape system, these control records are punched in the system configuration tape. The procedure for punching this tape is included in "Preparation of Load Mode and System Configuration Control Tapes" in this chapter. The system configuration tape is entered between the IBM-supplied tapes, BP02 and BP03, as illustrated in Figures 8-7 and 8-9 under "Paper Tape System Initial Load Operating Procedure" and "Paper Tape System Reload Operating Procedure" in this chapter.

principal I/O devices

When more than one input device or output device of a type is configured for a system, the fastest device defined in the REQ control records is used by the system. The following chart lists the principal I/O devices selected by the system.

Device specified on REQ control records	Principal I/O device
2501, 1442, paper tape	2501 input, 1442 output
1442, paper tape	1442 input/output
Paper tape	Paper tape input/output
1403, 1132	1403 output

When both a 1403 Printer and an 1132 Printer are configured, the 1403 is used by the system as the principal printer. You can specify the use of the console printer as the principal print device with // TYP and // CPRNT monitor control records. (These control records are described in Chapter 5.)

CORE Control Record

general function

This control record is an optional record that allows you to define a core size that is different than the actual size of core.

format

Card column	Contents	Explanation
1 through 4	CORE	
5	Blank	
6 through 8	04K, 08K, 16K, or 32K	The entry chosen specifies the core size you are defining.
9 through 80	Blanks	

card system use

For a card system, a CORE control card is placed in an initial load or reload card deck before or after the REQ card and between the IBM-supplied SCON and TERM cards. The order of cards for an initial load and reload is illustrated in Figures 8-2 and 8-4 under "Card System Initial Load Operating Procedure" and "Card System Reload Operating Procedure," respectively, in this chapter.

paper tape system use

For a paper tape system, this control record (when used) is punched in the system configuration tape. The procedure for punching this tape is included in "Preparation of Load Mode and System Configuration Control Tapes" in this chapter. The system configuration tape is entered between the IBM-supplied tapes, BP02 and BP03, as illustrated in Figures 8-7 and 8-9 under "Paper Tape System Initial Load Operating Procedure" and "Paper Tape System Reload Operating Procedure" in this chapter.

Preparation of Load Mode and System Configuration Control Tapes

Paper tape control records must be punched in PTTC/8 (perforated tape transmission code). The load mode and system configuration control tapes are punched by using the Paper Tape Utility Program (PTUTL). Initially, these control records are punched by using the stand-alone PTUTL tape, BP17, that is supplied by IBM.

The materials that you need to prepare the load mode and system configuration control tapes are:

- The Paper Tape Utility Program (PTUTL) tape, BP17
- A blank tape

The preparation of the load mode and system configuration control tapes do not have to be punched consecutively as in the procedure in Figure 8-1. These control records can be prepared separately by using the portions of the procedure that are applicable to the record being punched.

Turn on system power

Move the console POWER switch to ON.

Load the PTUTL Program tape, BP17

1. Insert the PTUTL tape, BP17, in the paper tape reader.
2. Position under the read starwheels one of the delete codes after the program ID.
3. Move the console mode switch to RUN.
4. Press IMM STOP on the console.
5. Press RESET on the console.
6. Press PROGRAM LOAD on the console.

   The core image loader is read into core storage, and the system waits with /006C displayed in the ACCUMULATOR.

7. Press PROGRAM START to finish the reading of PTUTL.

   When the reading of BP17 is complete, the system waits with /00C9 in the ACCUMULATOR.

8. Press PROGRAM START again.

   The system waits again with /1111 in the ACCUMULATOR.

9. Turn console entry switches 2 and 3 on.

   2 indicates keyboard input.
   3 indicates that records are to be punched by the paper tape punch.
   Complete operating procedures for PTUTL are in Chapter 8.

Ready the paper tape punch

1. Insert a blank tape in the paper tape punch.
2. Punch a leader of delete codes with the DELETE key.

B2

A2

Figure 8-1 (Part 1 of 4). Preparation of paper tape load and reload control tapes

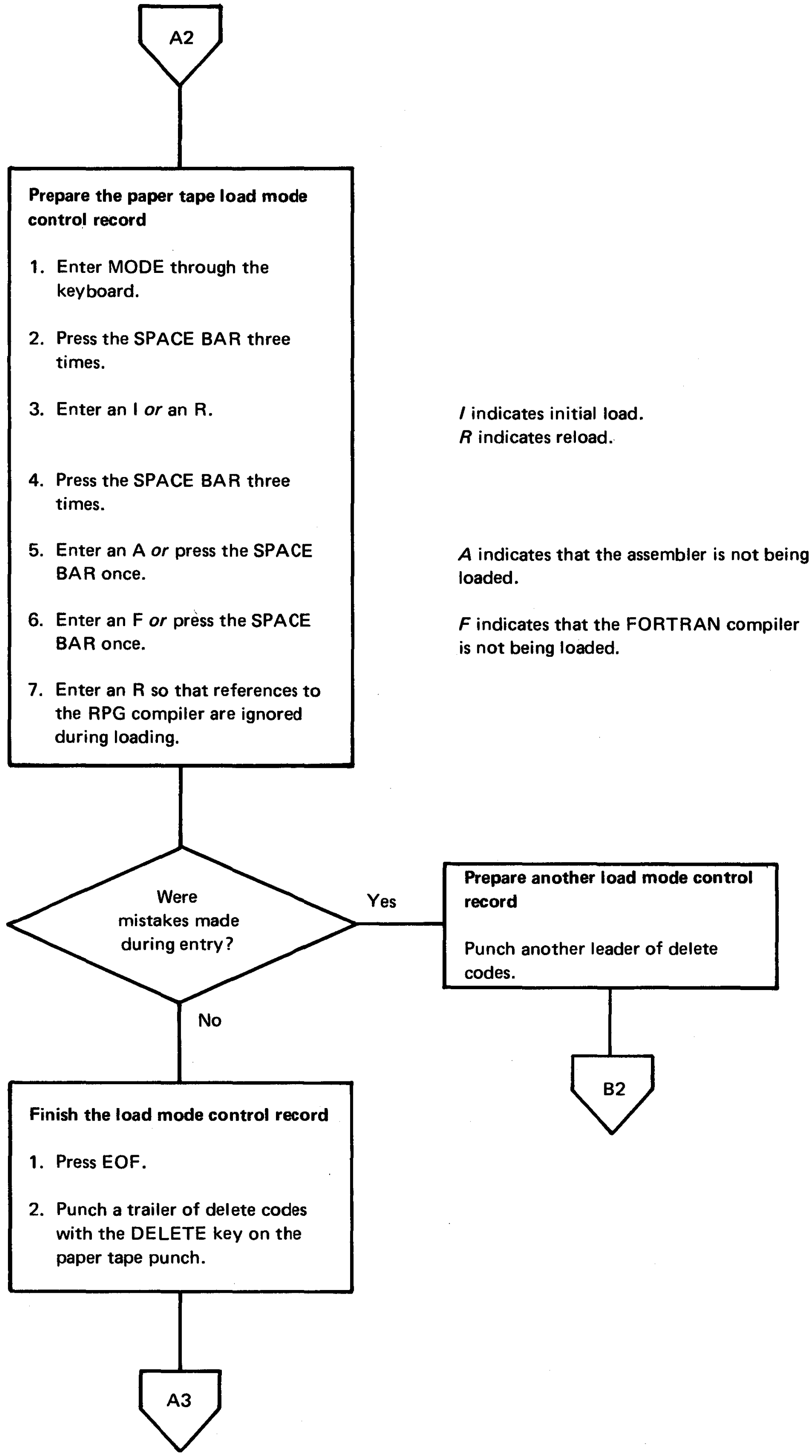
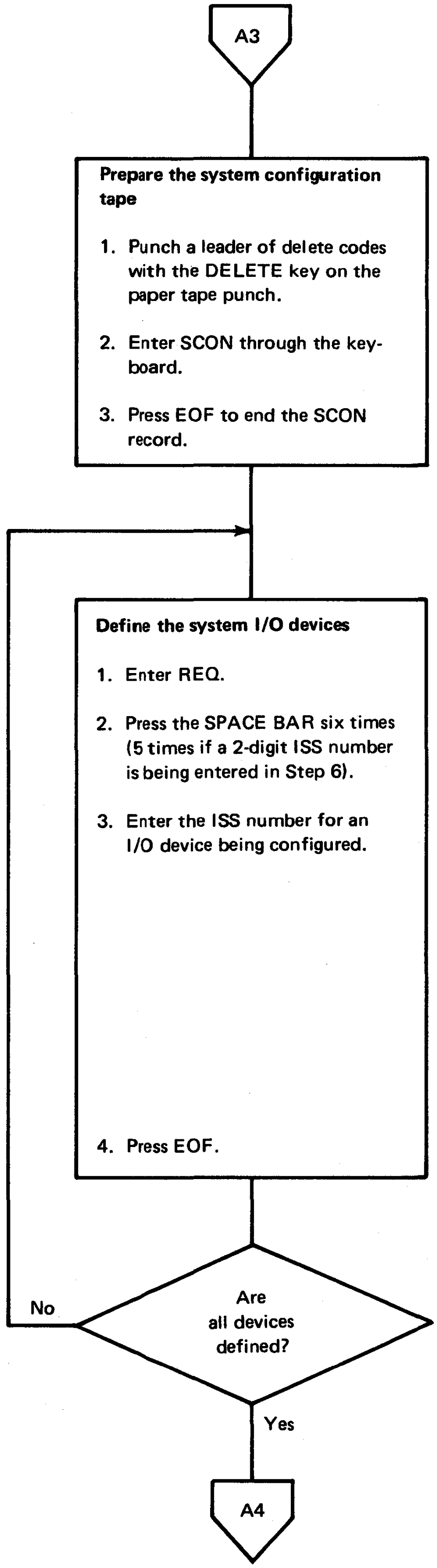


Figure 8-1 (Part 2 of 4). Preparation of paper tape load and reload control tapes

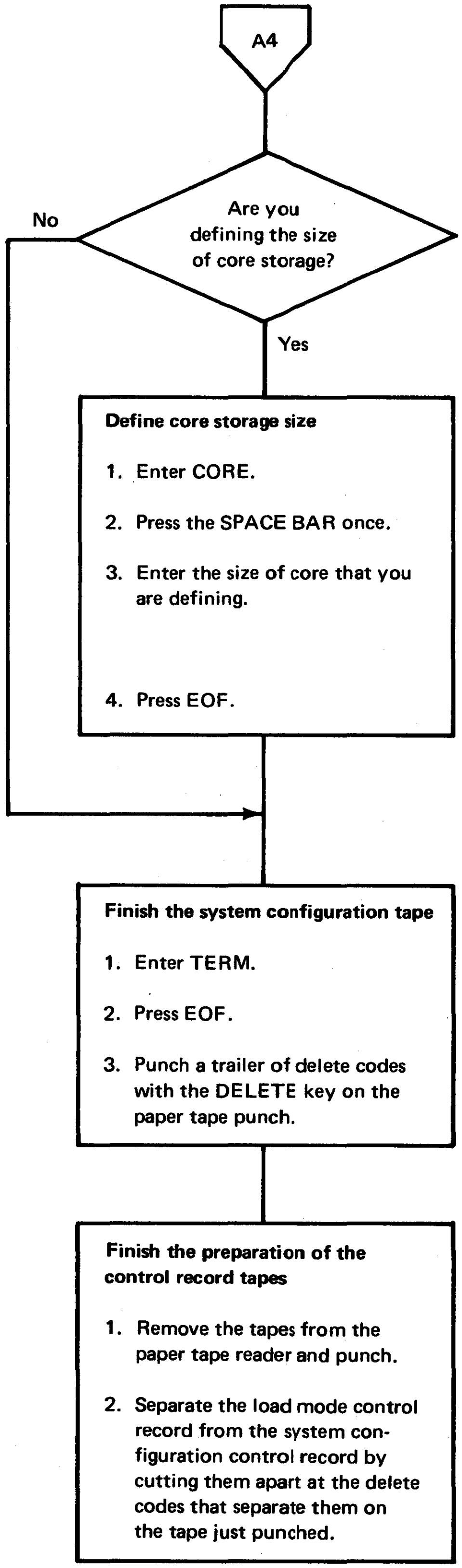


If errors are made during the preparation of this tape, repeat from here.

I/O device	*ISS number*
1442 Card Read Punch/Card Reader	1
Paper tape reader and/or punch	3
2501 Card Reader	4
1132 Printer	6
1403 Printer	9

Note: Maximum ISS entry is 20.

Figure 8-1 (Part 3 of 4). Preparation of paper tape load and reload control tapes



04K defines core size of 4K.
08K defines core size of 8K.
16K defines core size of 16K.
32K defines core size of 32K.

Figure 8-1 (Part 4 of 4). Preparation of paper tape load and reload control tapes

CARD SYSTEM INITIAL LOAD OPERATING PROCEDURE

The materials that you need to perform a card system initial load procedure are:

- An uninitialized disk
- IBM-supplied system cards
- Load mode and REQ (and CORE, if used) cards that you punched. An I must be punched in column 8 of the load mode card

The initial load cards and card decks that are being used in the initial load procedure must be arranged in the order shown in Figure 8-2.

Note. If your computing system has 2311 Disk Storage Drives, replace the DISKN subroutine included in the system device subroutines with the DISKN subroutine included with the stand-alone utilities. The DISKN included in the system device subroutines is identified by the letters PMN beginning in card column 73. The sequence numbers are included in the materials supplied with the modification level of your system. The DISKN included with the stand-alone utilities is identified by the letters PMNDN beginning in card column 73.

You perform a card system initial load procedure as shown in Figure 8-3.

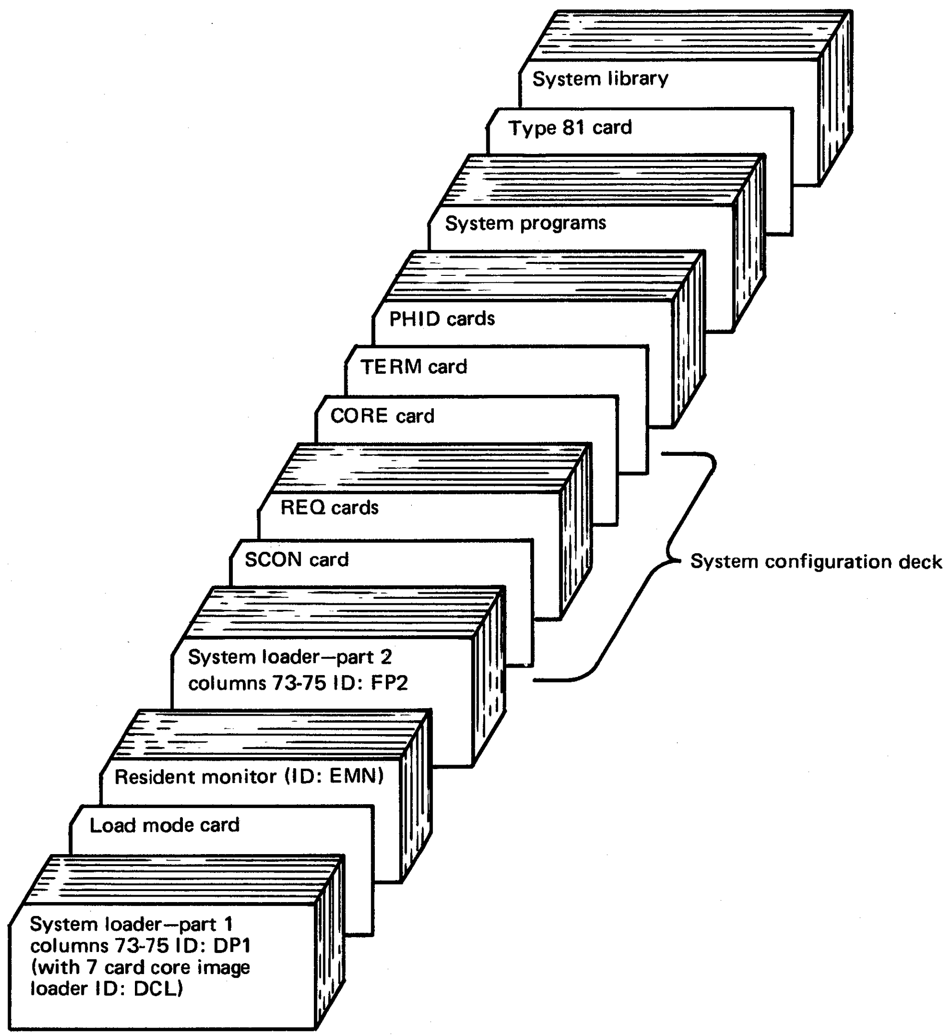


Figure 8-2. Card system initial load cards

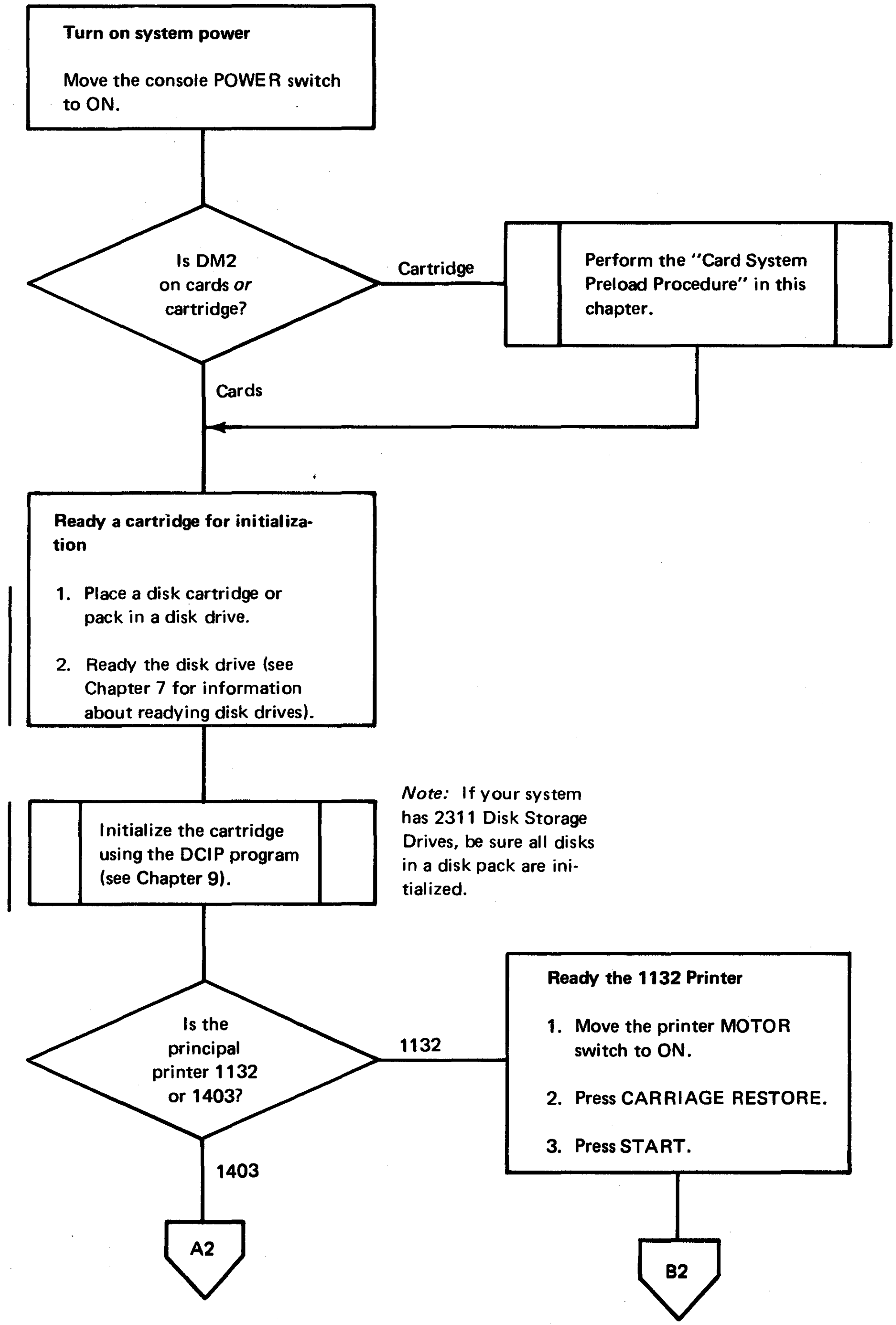


Figure 8-3 (Part 1 of 3). Card system initial load procedure

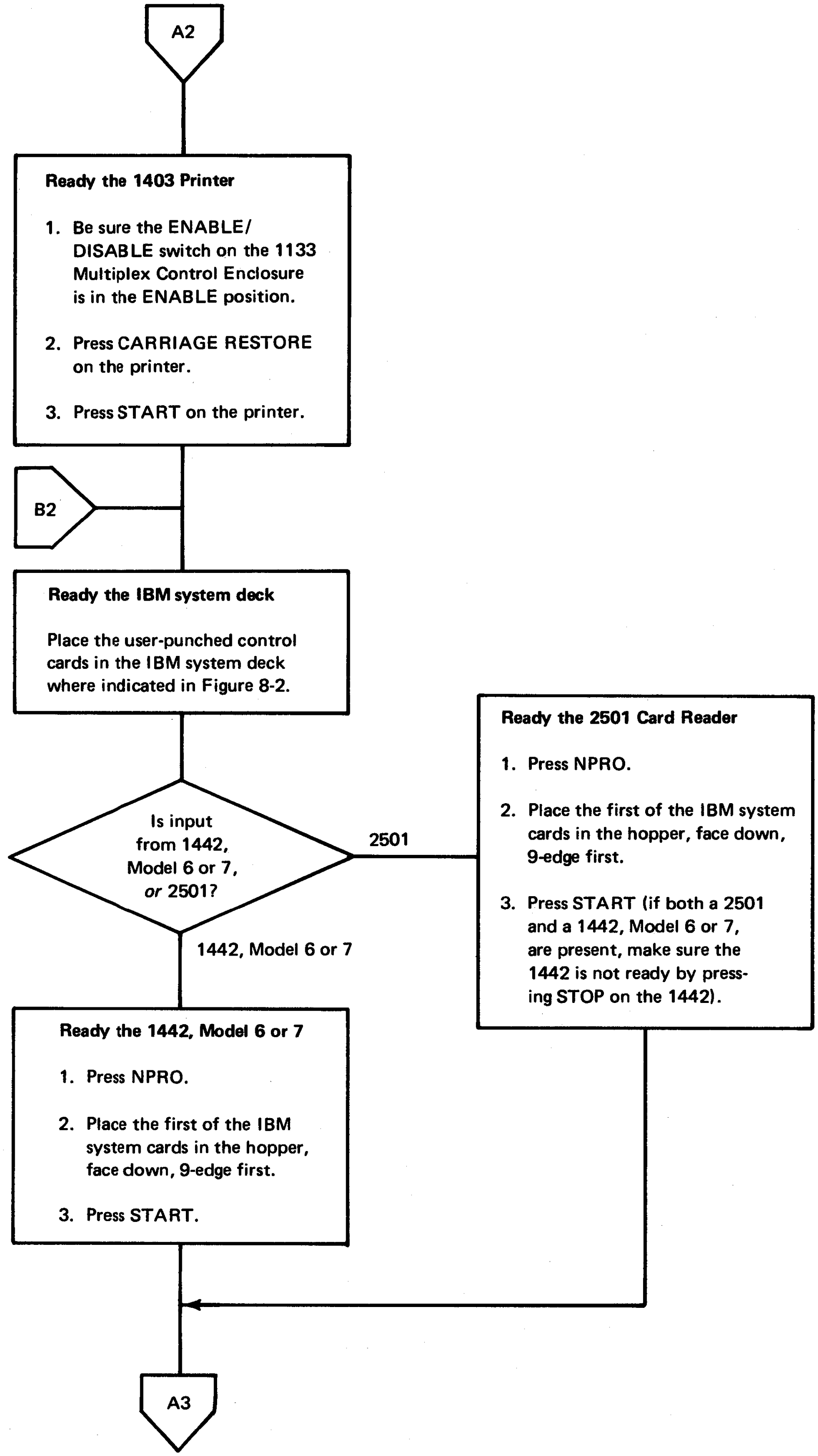


Figure 8-3 (Part 2 of 3). Card system initial load procedure

A3

Start the reading of the IBM system deck

1. Set the console entry switches 12 through 15 to indicate the physical drive number of the drive that contains the initialized cartridge (switches 0 through 11 must be off).

Drive 0 – all off
Drive 1 – switch 15 on
Drive 2 – switch 14 on
*Drive 3 – switches 14 and 15 on
*Drive 4 – switch 13 on
Drive 5 – switches 13 and 15 on
Drive 6 – switches 13 and 14 on
Drive 7 – switches 13, 14, and 15 on
*Drive 8 – switch 12 on
*Drive 9 – switches 12 and 15 on
Drive 10 – switches 12 and 14 on
*Not used on a 2311 Disk Storage Drive, Model 12

2. Turn the console mode switch to RUN.
3. Press IMM STOP on the console.
4. Press RESET on the console.
5. Press PROGRAM LOAD on the console.

Reading of the IBM system deck begins.

Finish procedure

Continue placing IBM system cards in the reader hopper until all of the cards have been placed in the hopper.

If the system halts (halt codes displayed in the ACCUMULATOR on the console display panel), refer to Appendix B. If the system prints a message on the console printer, refer to Appendix A.

The system prints a message on the principal printer when loading of the monitor system is complete.

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational.

Figure 8-3 (Part 3 of 3). Card system initial load procedure

CARD SYSTEM RELOAD OPERATING PROCEDURE

The materials that you need to perform a card system reload procedure are:

- A system cartridge
- An IBM-supplied cold start card and blank cards (2 are enough)
- IBM-supplied system cards
- Load mode and REQ (and CORE, if used) cards that you punched. An R must be punched in column 8 of the load mode card

The reload cards that are being used in the system reload must be arranged in the order shown in Figure 8-4.

system reconfiguration

Reconfiguration is done each time a reload procedure is performed and is necessary when a system cartridge is copied from a system with a different configuration. If reconfiguration is all that is being done by a reload operation, place the type 81 control record immediately after the PHID control records.

phase and program revision or addition

Be sure the phase identification (PHID) control records reflect the phase ID limits of the system programs being added or in which phases are being revised or added. The programs or phases being revised or added by the reload procedure must be placed in ascending phase ID sequence immediately behind the IBM-supplied PHID control records.

The record immediately following the last phase being loaded must be an end-of-program card (see "End-of-Program (EOP) Card" in Appendix I). In this case, the EOP card can have words 1, 2, and 4 through 54 blank. The message END OF RELOAD is printed on the console printer when a system reload is complete.

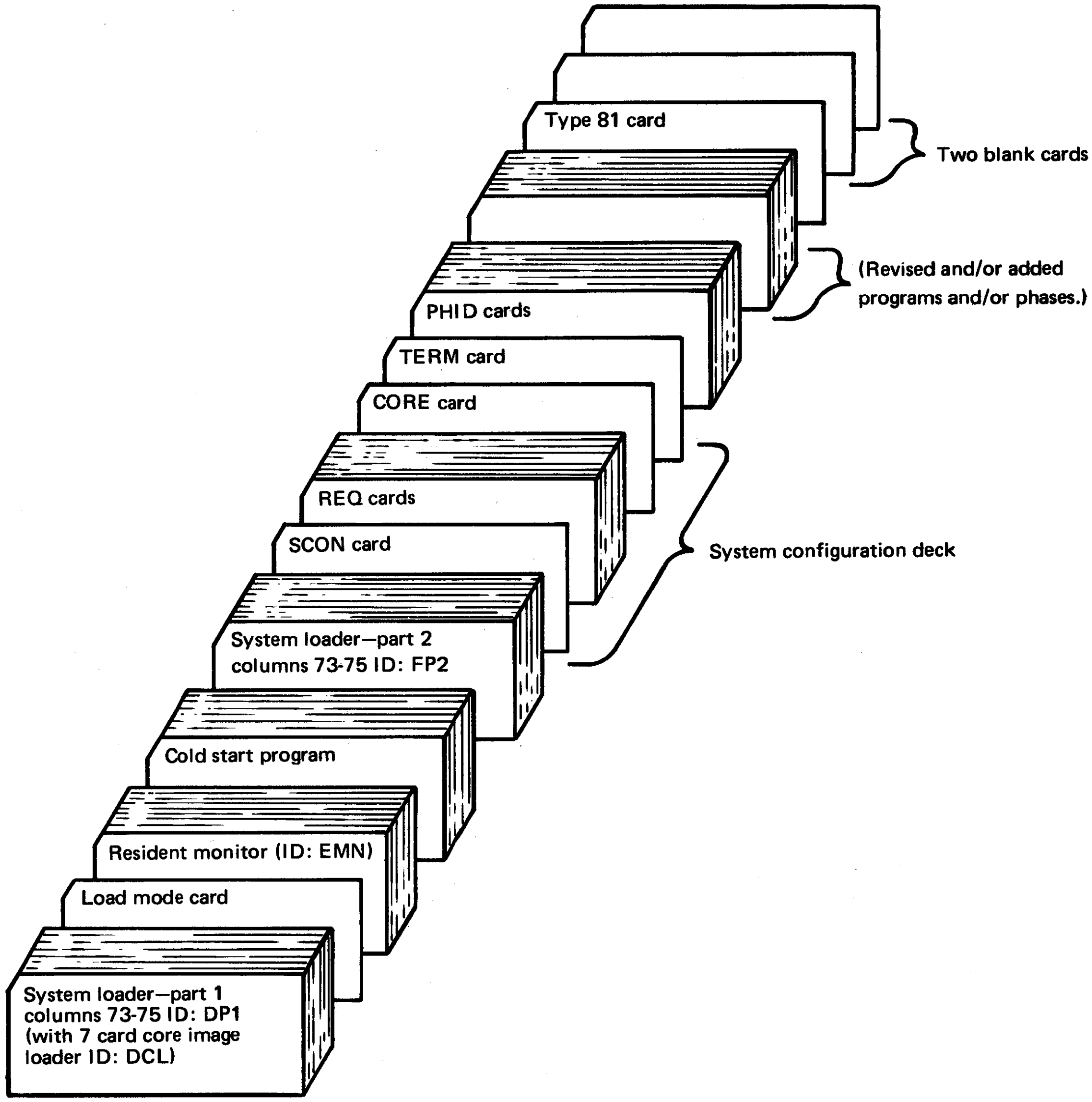


Figure 8-4. Card system reload cards

The reload function can link to MODIF if a // XEQ MODIF control record follows directly after the type 81 control card. This function can be performed together with any combination of the reload functions. The END OF RELOAD message is not printed, but the // XEQ MODIF control record is printed on the principal printer. You perform a card system reload procedure as shown in Figure 8-5.

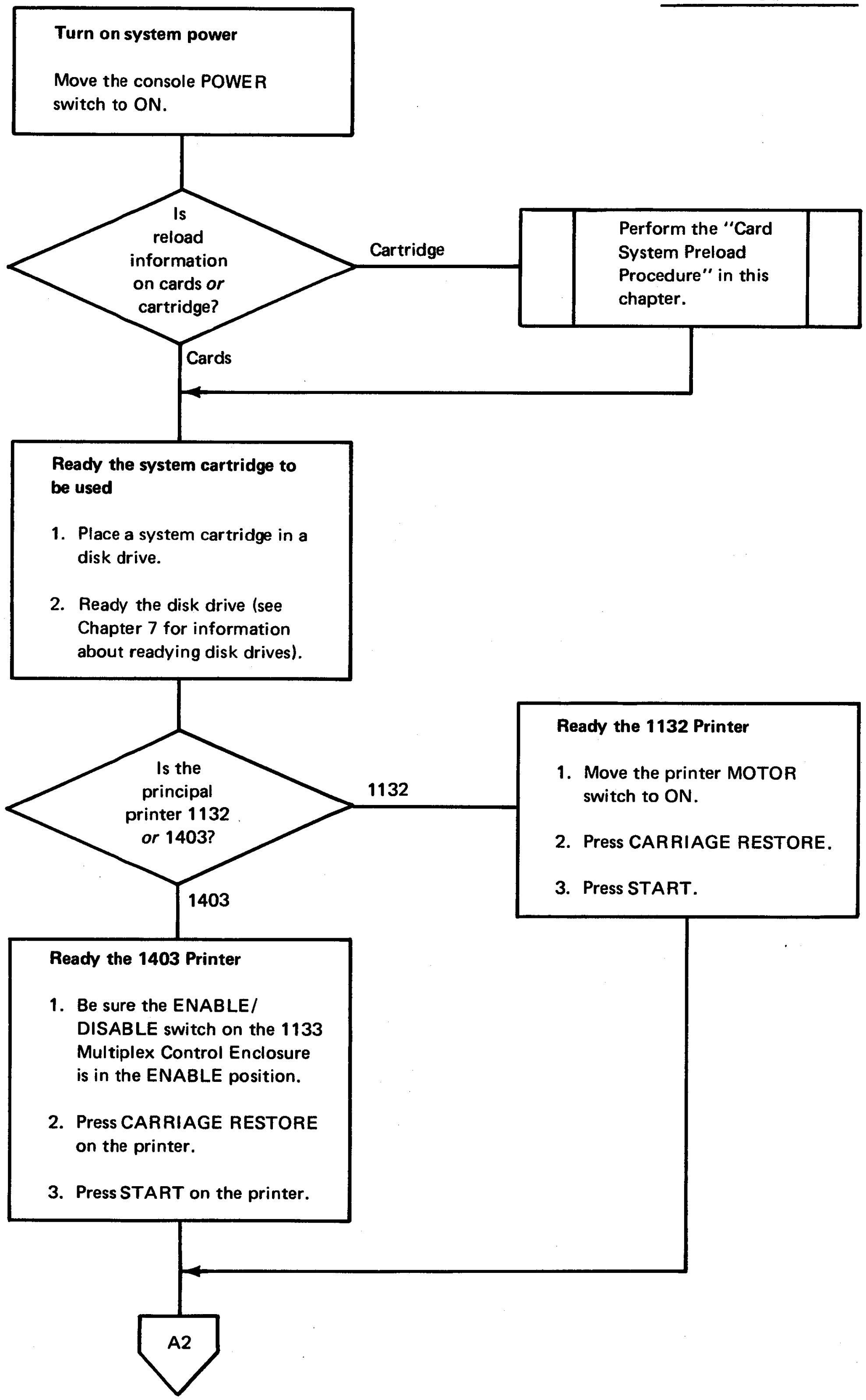


Figure 8-5 (Part 1 of 4). Card system reload procedure

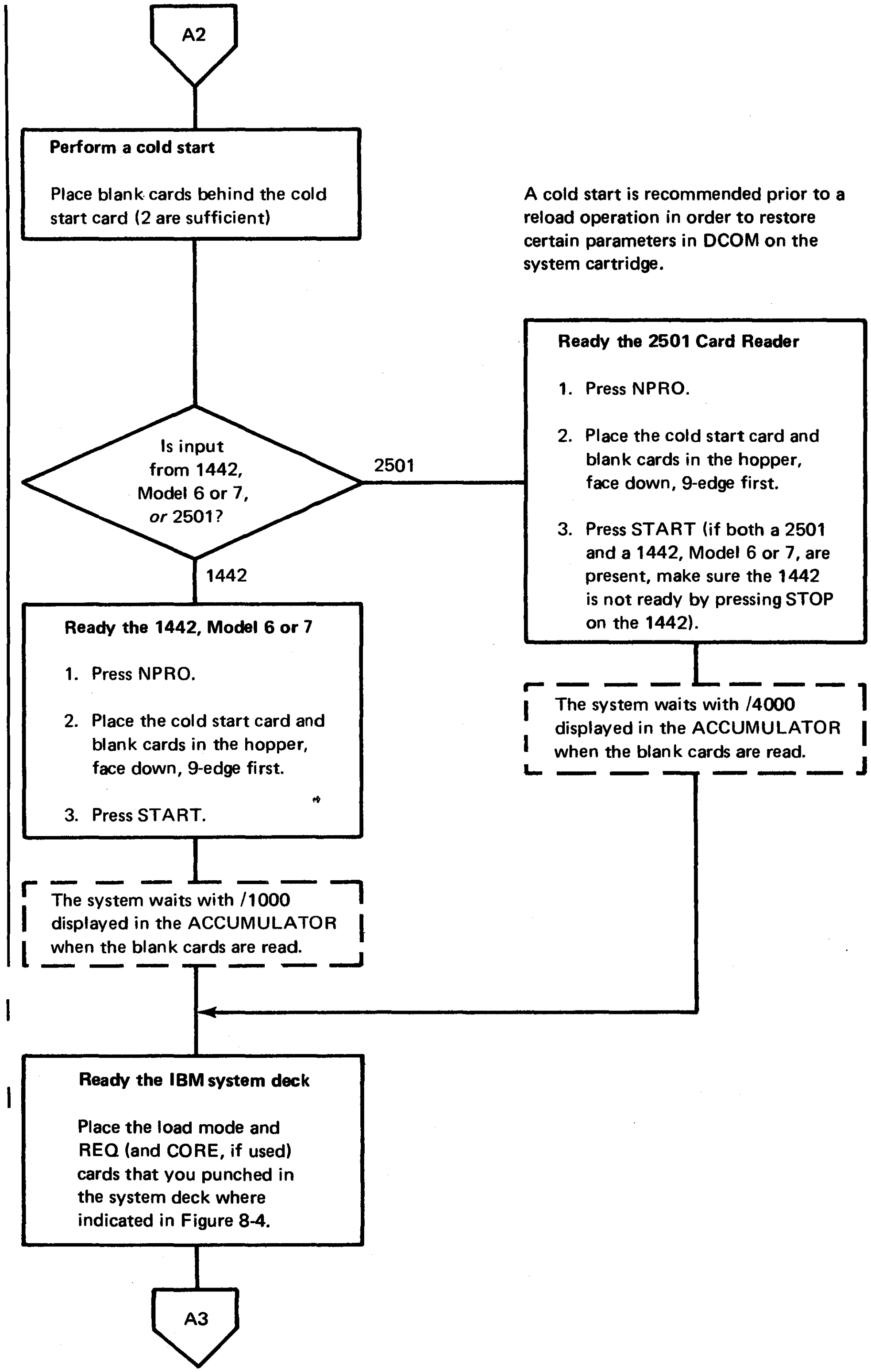


Figure 8-5 (Part 2 of 4). Card system reload procedure

A3

Is input from 1442, Model 6 or 7, *or* 2501?

2501

Ready the 2501 Card Reader

1. Press NPRO.
2. Place the first of the reload deck in the hopper, face down, 9-edge first.
3. Press START (if both a 2501 and a 1442, Model 6 or 7, are present, make sure the 1442 is not ready by pressing STOP on the 1442).

1442

Ready the 1442, Model 6 or 7

1. Press NPRO.
2. Place the first of the reload deck in the hopper, face down, 9-edge first.
3. Press START.

Start the reading of the IBM system deck

1. Set the console entry switches 12 through 15 to indicate the physical drive number of the drive that contains the system cartridge (switches 0 through 11 must be off).

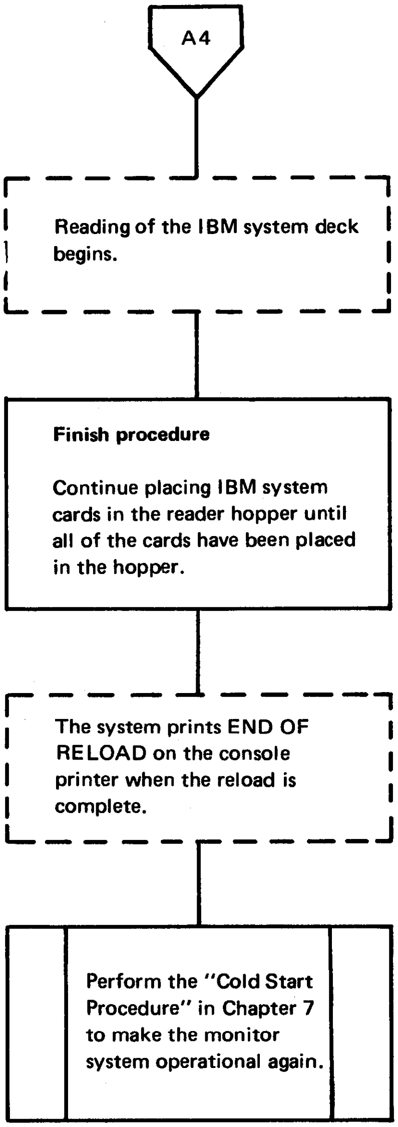
Drive 0 – all off
Drive 1 – switch 15 on
Drive 2 – switch 14 on
*Drive 3 – switches 14 and 15 on
*Drive 4 – switch 13 on
Drive 5 – switches 13 and 15 on
Drive 6 – switches 13 and 14 on
Drive 7 – switches 13, 14 and 15 on
*Drive 8 – switch 12 on
*Drive 9 – switches 12 and 15 on
Drive 10 – switches 12 and 14 on

*Not used on a 2311 Disk Storage Drive, Model 12

2. Turn the console mode switch to RUN.
3. Press IMM STOP on the console.
4. Press RESET on the console.
5. Press PROGRAM LOAD on the console.

A4

Figure 8-5 (Part 3 of 4). Card system reload procedure



If the system halts (halt codes displayed in the ACCUMULATOR on the console display panel), refer to Appendix B. If the system prints a message on the console printer other than END OF RELOAD, see Appendix A.

Figure 8-5 (Part 4 of 4). Card system reload procedure

CARD SYSTEM PRELOAD OPERATING PROCEDURE

The materials that you need to perform a card system preload procedure are:

- A preload (UCART) cartridge
- An IBM-supplied cold start card
- Blank cards; the dump of the monitor system requires approximately 5400 cards

The dump is accomplished by loading the Monitor II cold start card supplied with the cartridge from IBM. The format of the preload cartridge is such that the same cold start card that is used to make the monitor system operational is used to call the disk-to-card dump program (UCART).

You perform a card system preload procedure as shown in Figure 8-6.

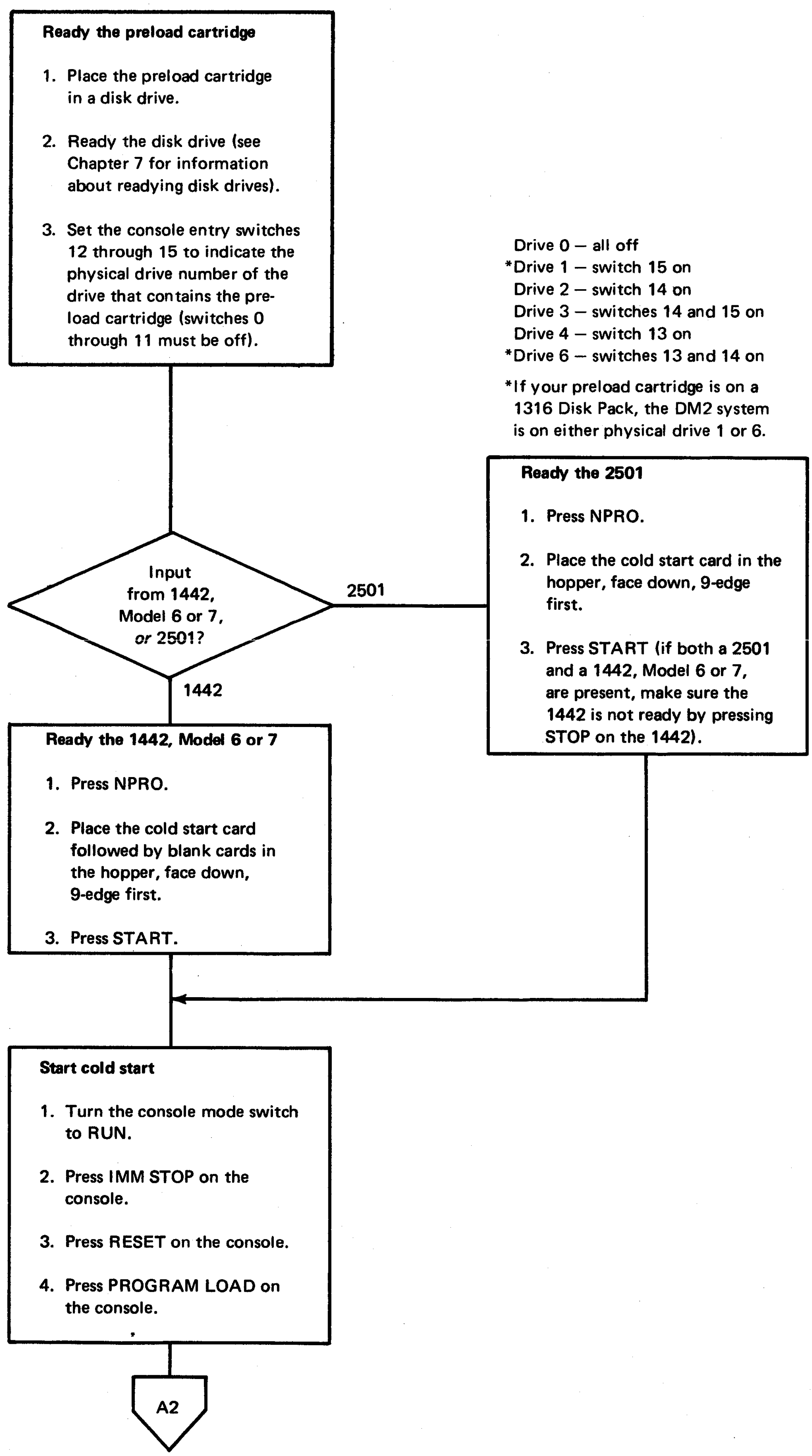


Figure 8-6 (Part 1 of 2). Card system preload procedure

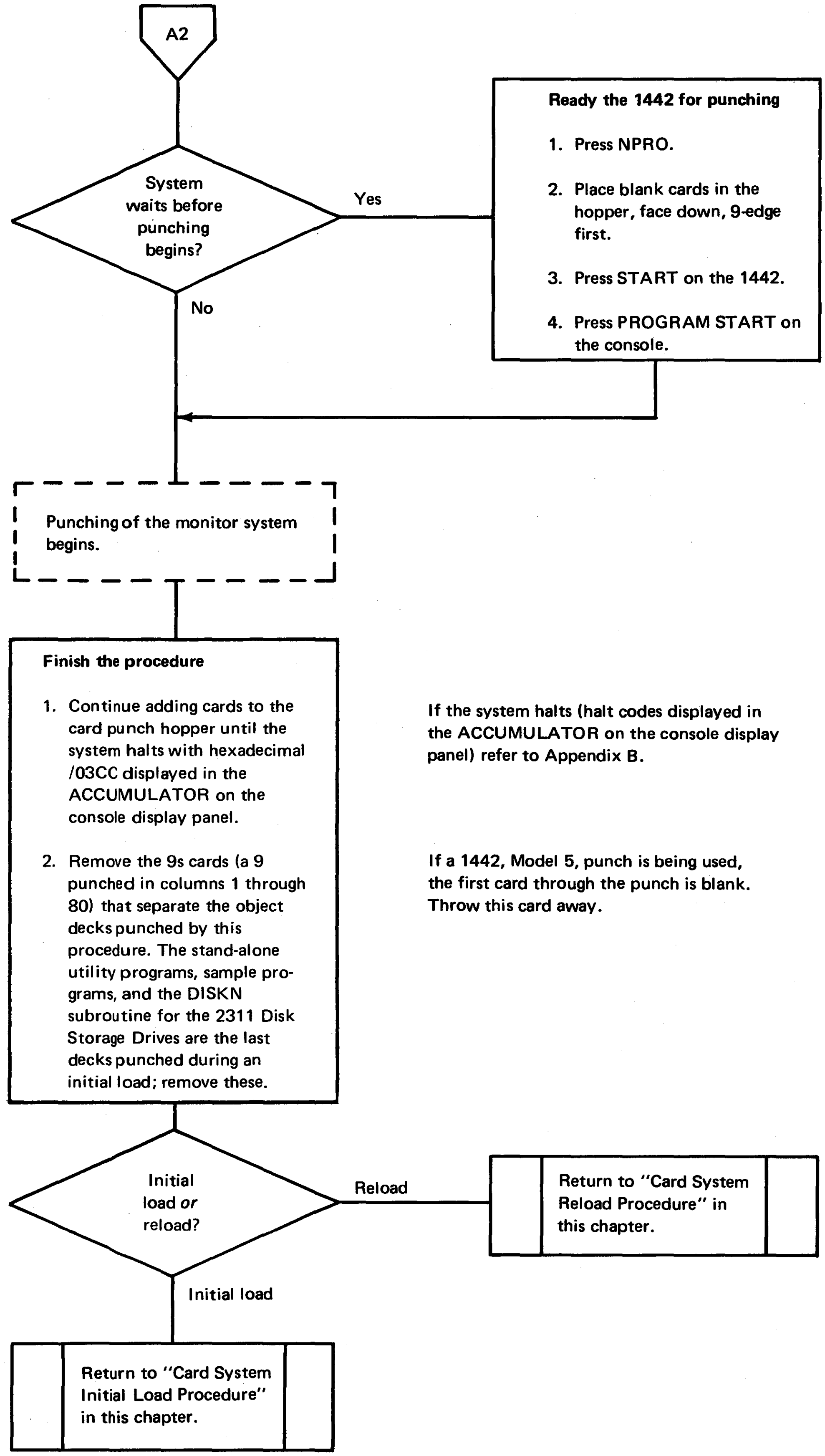


Figure 8-6 (Part 2 of 2). Card system preload procedure

PAPER TAPE SYSTEM INITIAL LOAD OPERATING PROCEDURE

The materials that you need to perform a paper tape system initial load procedure are:

- An uninitialized disk cartridge
- DCIP (Disk Cartridge Initialization Program) tape, BP16
- IBM-supplied system tapes, BP01-BP14
- Load mode control record tape and system configuration record tape that you punched

If the assembler or the FORTRAN compiler is not being loaded, the corresponding tapes (BP05 or BP07) can be omitted; however, if they are not loaded, they cannot be loaded during a system reload procedure. The assembler and the FORTRAN compiler can be loaded during an initial load procedure only.

Load only those system library tapes (BP09 through BP14) that are required for your system. Tapes BP01-BP14 that are being used in the initial load must be arranged in the order shown in Figure 8-7.

Tape BP15 is the cold start record that is used to make the monitor system operational after the initial load is complete. Tapes BP16-BP20 are stand-alone utilities and are not loaded as part of the monitor system. However, you use BP17 (PTUTL) to punch the load mode and system configuration tapes that are used during initial load and BP16 (DCIP) to initialize the disk cartridge during initial load. Tapes BP21 and BP22 are sample programs that you can execute under monitor system control after the initial load is complete (see "Entering Jobs From the Paper Tape Reader" in Chapter 7).

You perform a paper tape system initial load procedure a shown in Figure 8-8.

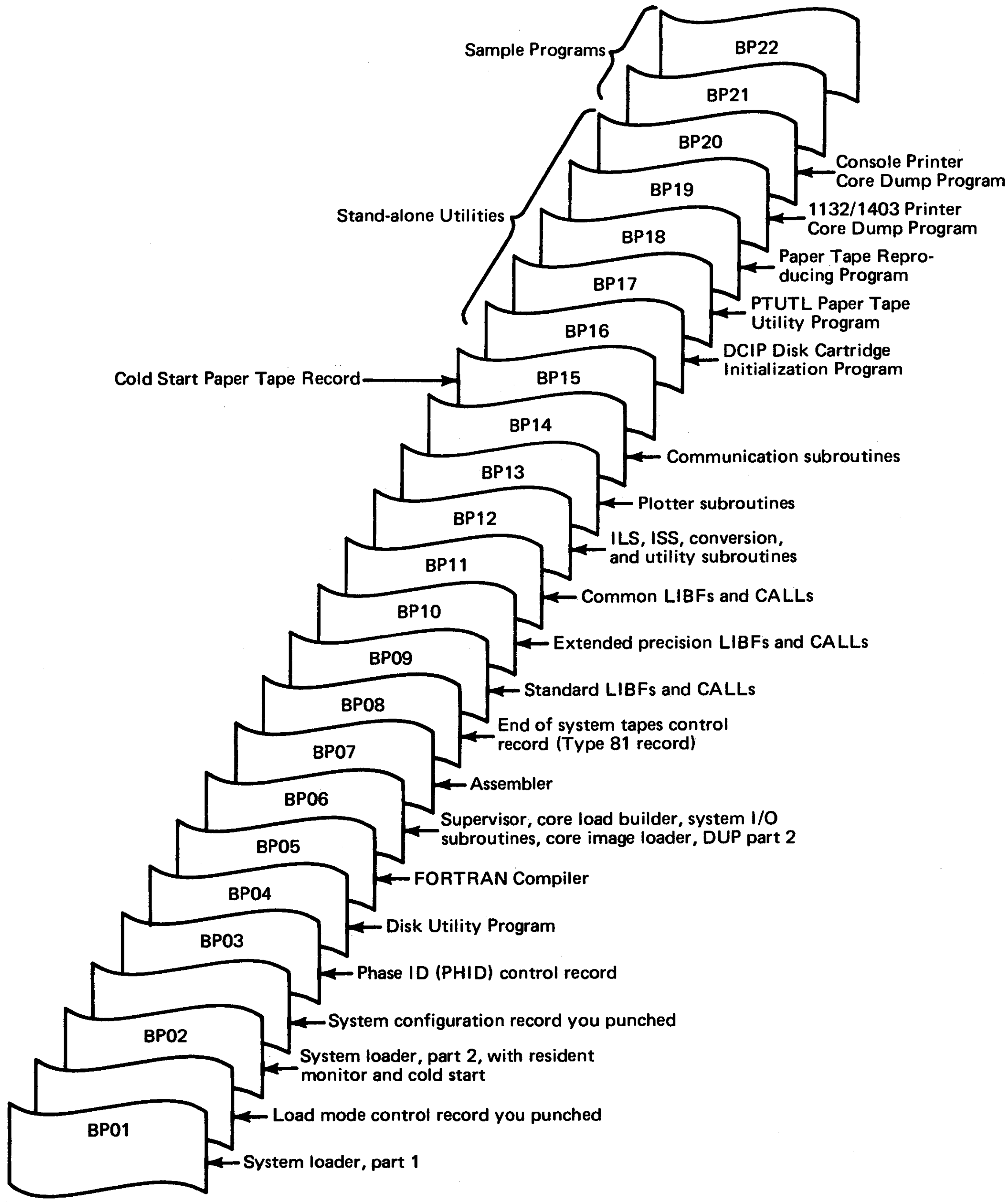


Figure 8-7. Paper tape system load tapes

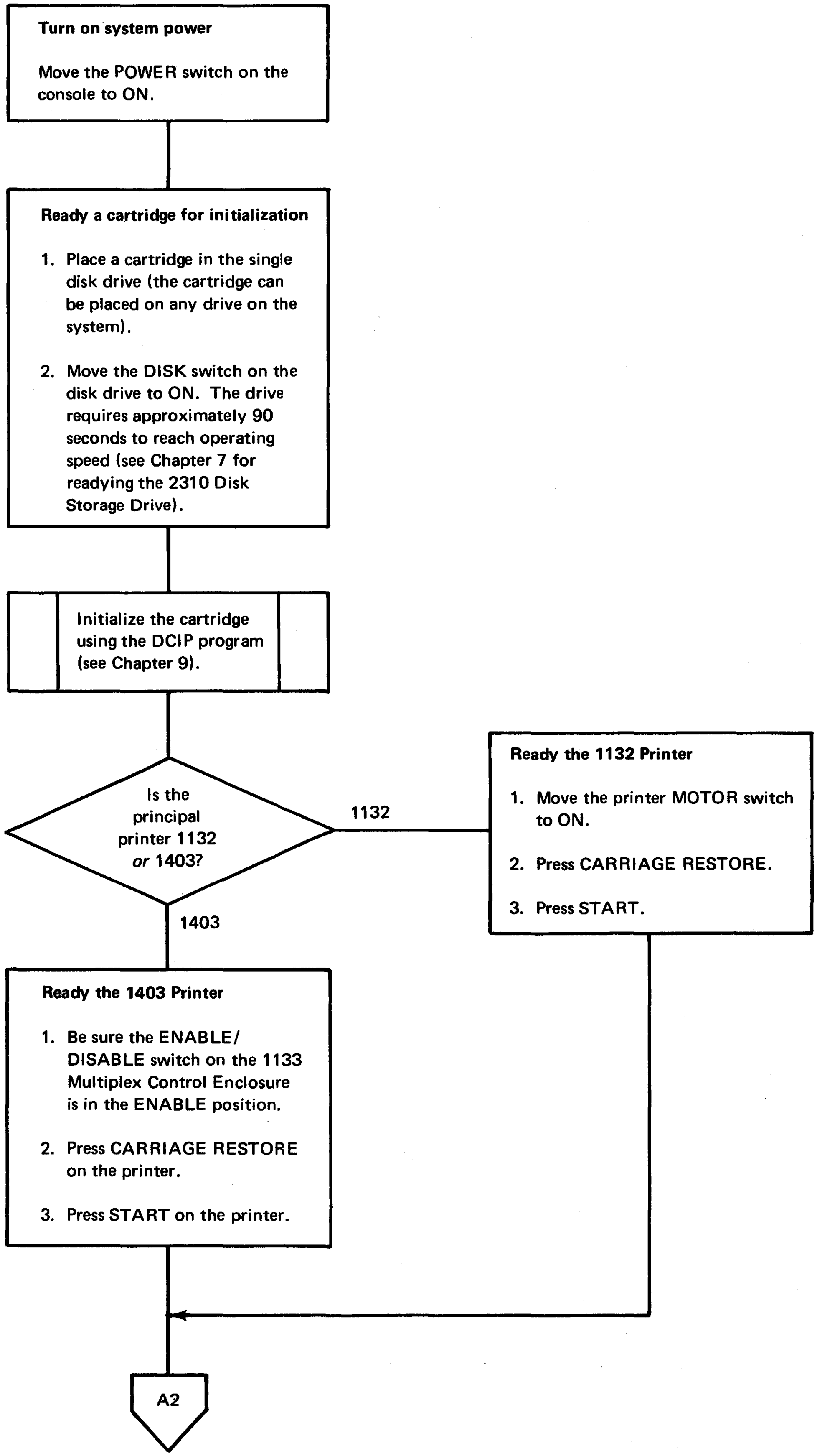


Figure 8-8 (Part 1 of 3). Paper tape system initial load procedure

A2

Ready the IBM system tapes

Place the user-punched control tapes in the IBM system tapes where indicated in Figure 8-7.

Ready the paper tape reader

1. Insert BP01, system loader, Part 1, in the paper tape reader.

When loading tapes, position under the read starwheels any of the delete codes that follow the program ID in the tape leader.

Start the reading of the Monitor System

1. Set the console entry switches 12 through 15 to indicate the physical drive number of the drive that contains the initialized cartridge (switches 0 through 11 must be off).

Drive 0 – all off
Drive 1 – switch 15 on
Drive 2 – switch 14 on
Drive 3 – switches 14 and 15 on
Drive 4 – switch 13 on

2. Turn the console mode switch to RUN.
3. Press IMM STOP on the console.
4. Press RESET on the console.
5. Press PROGRAM LOAD on the console.

The core image loader is read into core storage from BP01, and the system waits with /006C displayed in the ACCUMULATOR.

6. Press PROGRAM START to finish the reading of Part 1 of the system loader.

When reading of BP01 is complete, the system waits with /00C9 displayed in the ACCUMULATOR.

7. Press PROGRAM START again.

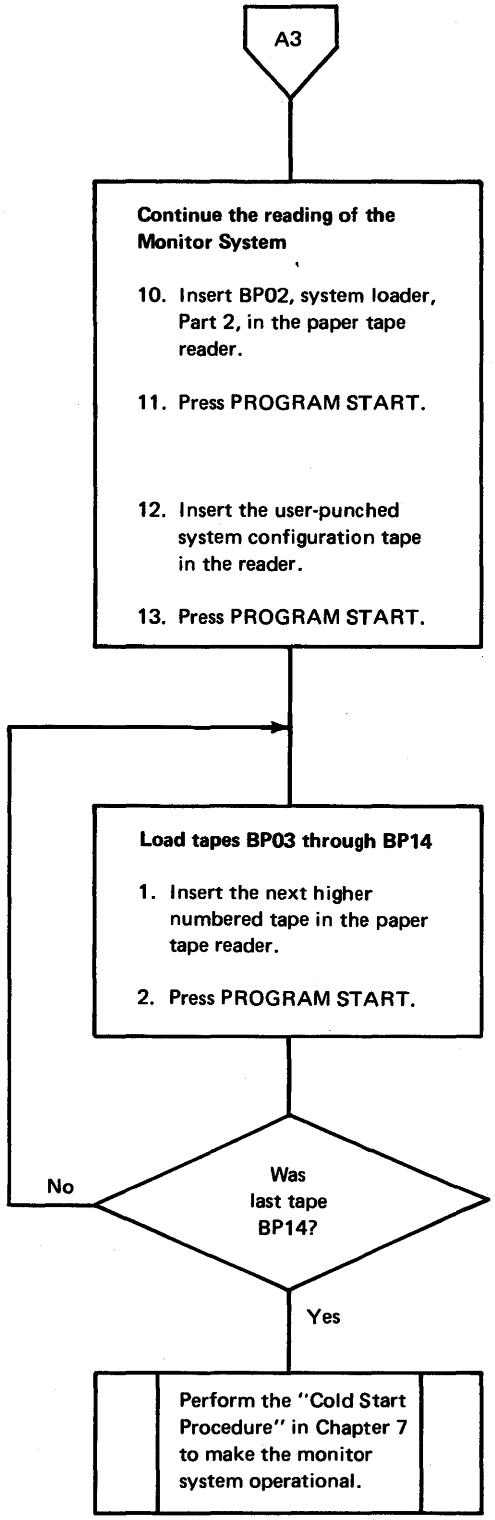
The system waits again with /3000 displayed in the ACCUMULATOR.

8. Place the user-punched load mode control record tape in the reader.
9. Press PROGRAM START.

The system waits with /3000 in the ACCUMULATOR when reading of the tape is complete.

A3

Figure 8-8 (Part 2 of 3). Paper tape system initial load procedure



The system waits with /3000 displayed in the ACCUMULATOR when reading of BP02 is complete.

The system waits with /3000 displayed in the ACCUMULATOR when reading of the user-punched tape is complete.

The system waits with /3000 displayed in the ACCUMULATOR when reading of each tape is complete.

Figure 8-8 (Part 3 of 3). Paper tape system initial load procedure

PAPER TAPE SYSTEM RELOAD OPERATING PROCEDURE

The materials that you need to perform a paper tape system reload procedure are:

- A system cartridge
- Cold start paper tape record, BP15
- System tapes
- Load mode control record tape and system configuration record tape that you punched

The paper tapes to be used in the reload must be arranged in the order shown in Figure 8-9. The tapes for the system programs and/or phases that are being added or expanded must be arranged in ascending tape number order. Also, all programs being loaded must have phase ID numbers within the limits of the IDs punched in the PHID tape, BP03.

Note. If the assembler and/or FORTRAN compiler have been deleted or were not loaded during an initial load, they cannot be loaded during a system reload procedure. An initial load must be performed to load these 2 programs onto a cartridge.

You perform a paper tape system reload procedure as shown in Figure 8-10.

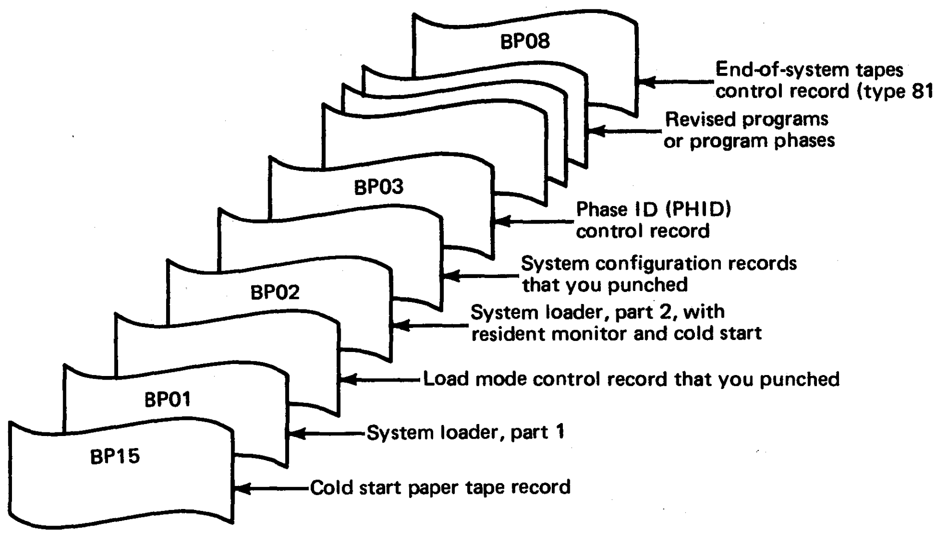


Figure 8-9. Paper tape system reload tapes

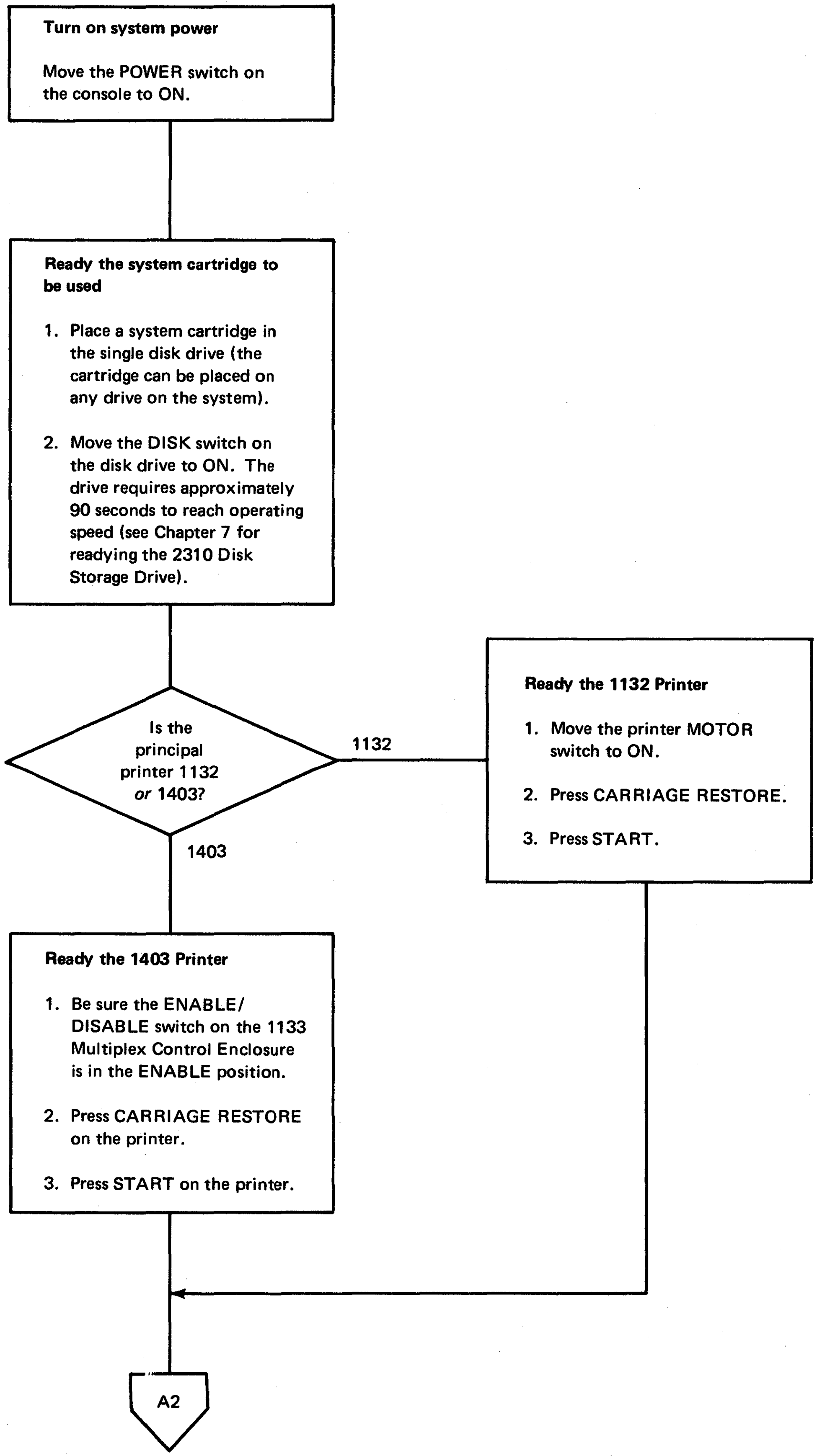


Figure 8-10 (Part 1 of 4). Paper tape system reload procedure

A2

Perform a cold start

1. Insert tape BP15, cold start paper tape record, in the paper tape reader.

   A cold start is recommended prior to a reload operation in order to restore certain parameters in DCOM on the system cartridge.

2. Set the console entry switches 12 through 15 to indicate the physical drive number of the drive that contains the system cartridge (switches 0 through 11 must be off).

   Drive 0 – all off
   Drive 1 – switch 15 on
   Drive 2 – switch 14 on
   Drive 3 – switches 14 and 15 on
   Drive 4 – switch 13 on

3. Turn the console mode switch to RUN.
4. Press IMM STOP on the console.
5. Press RESET on the console.
6. Press PROGRAM LOAD on the console.

   The system waits with /3000 in the ACCUMULATOR when reading of the cold start record is complete.

Ready the IBM system tapes

Place the load mode and system configuration control record tapes that you punched between the IBM reload tapes where indicated in Figure 8-9.

A3

Figure 8-10 (Part 2 of 4). Paper tape system reload procedure

A3

Start the reading of the reload tapes

1. Insert tape BP01, system loader, Part 1, in the paper tape reader.
2. Press PROGRAM START on the console.

   The core image loader is read into core storage from BP01, and the system waits with /006C displayed in the ACCUMULATOR.

3. Press PROGRAM START again to finish the reading of Part 1 of the system loader.

   When reading of BP01 is complete, the system waits with /00C9 displayed in the ACCUMULATOR.

4. Press PROGRAM START again.

   The system waits again with /3000 displayed in the ACCUMULATOR.

5. Place the user-punched load mode control record tape in the reader.
6. Press PROGRAM START.

   The system waits with /3000 in the ACCUMULATOR when reading of the tape is complete.

7. Insert tape BP02, system loader, Part 2, in the paper tape reader.
8. Press PROGRAM START.

   The system waits with /3000 in the ACCUMULATOR when reading of BP02 is complete.

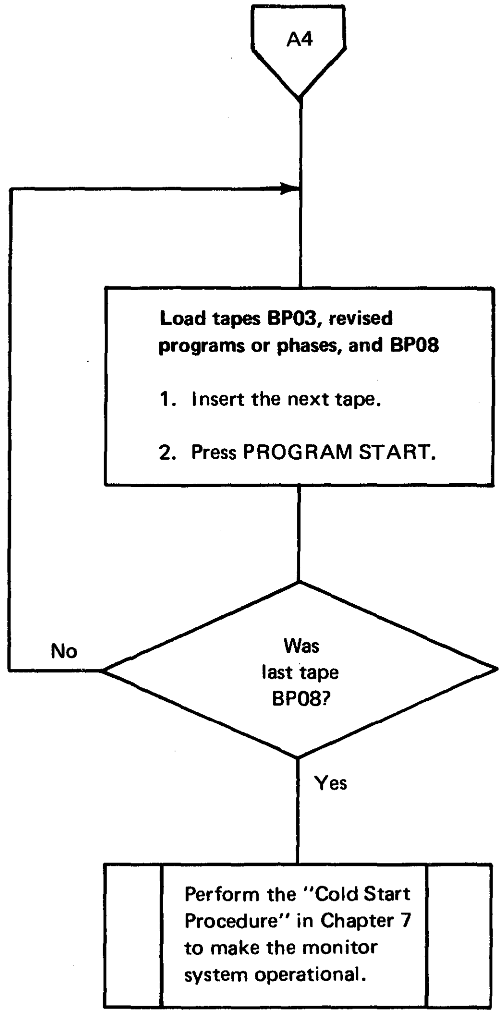
Configure system

1. Insert the user-punched system configuration tape in the reader.
2. Press PROGRAM START.

   The system waits with /3000 in the ACCUMULATOR when reading of the system configuration tape is complete.

A4

Figure 8-10 (Part 3 of 4). Paper tape system reload procedure



The system waits with /3000 in the ACCUMULATOR when reading of each tape is complete.

Figure 8-10 (Part 4 of 4). Paper tape system reload procedure

Chapter 9. Stand-alone Utility Programs

The stand-alone utility programs are each self-loading and complete with subroutines. These programs are separate from the monitor system library and enable you to perform operations without monitor system control. The stand-alone utility programs are:

- Console Printer Core Dump
- Printer Core Dump
- Disk Cartridge Initialization Program (DCIP)
- Paper Tape Reproducing
- Paper Tape Utility (PTUTL)

The first 3 of these are available in cards and paper tapes; the last 2 on paper tape only.

This chapter:

1. Describes the general functions of each of the stand-alone utility programs.
2. Presents sample operating procedures for using these programs.

You may use these operating procedures as they are presented, or you may modify them to meet the needs of your computing system. For those who are already familiar with similar procedures, the headings in each block can be used as reminders as you perform the procedure. For those who need more information, detailed steps for performing these procedures are provided. Not all steps of each procedure need to be done every time the procedure is used; do only those steps that are necessary.

Appendix B lists the halt codes that are displayed in the ACCUMULATOR on the console display panel if errors occur during these procedures.

CONSOLE PRINTER CORE DUMP

Selected portions of core storage are printed on the console printer when you use the Console Printer Core Dump Program.

dump format

Each core location is dumped as a 4-digit hexadecimal word with a space separating each word. The first word dumped is from the starting address that you specify through the console entry switches.

The materials that you need to use the Stand-alone Console Printer Core Dump Program are:

- Console Printer Core Dump Program card

-or-

- Console Printer Core Dump Program paper tape, BP20

Figure 9-1 is the operating procedure for the stand-alone Console Printer Core Dump Program.

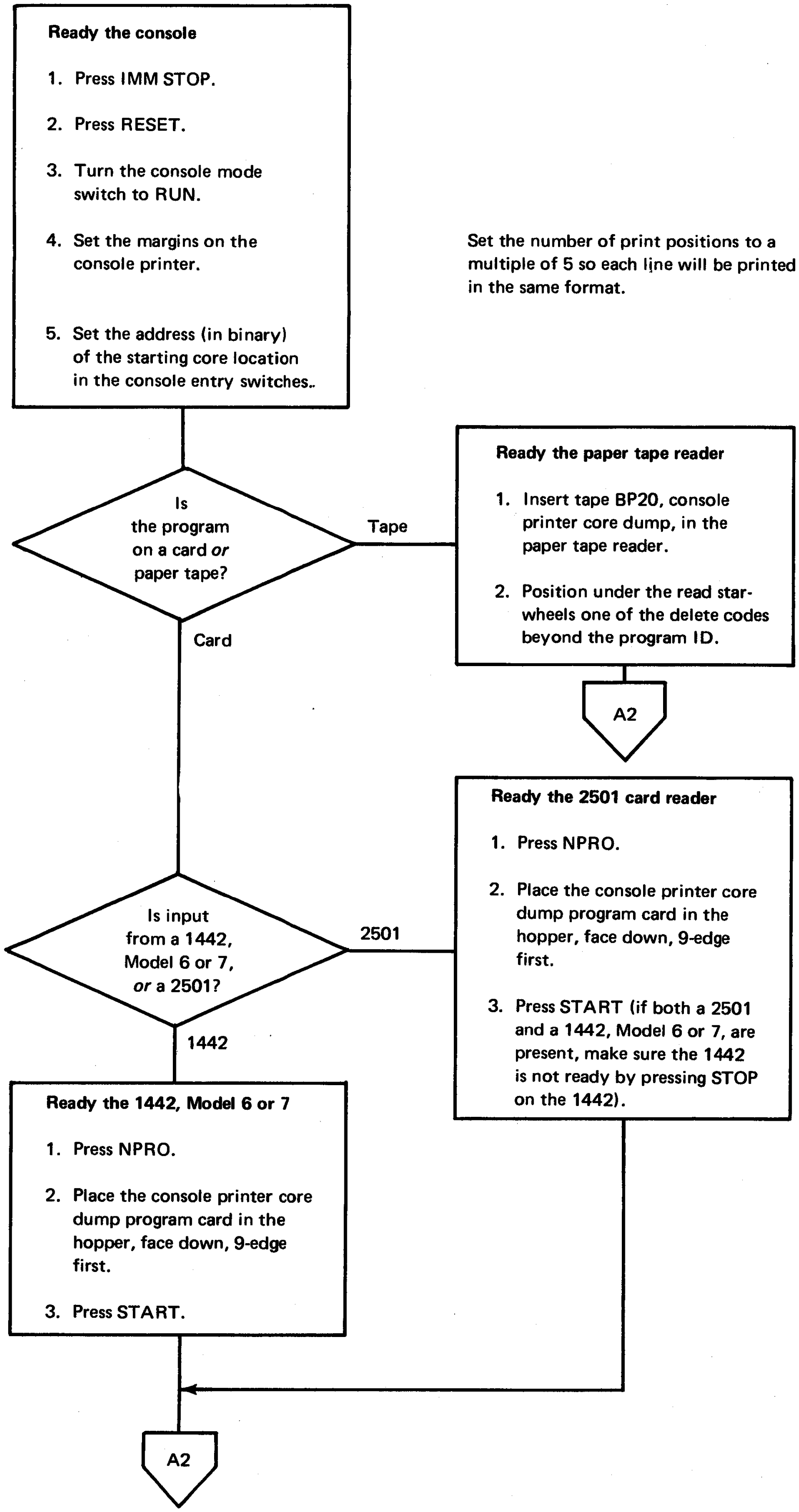


Figure 9-1 (Part 1 of 2). Console printer core dump operating procedure

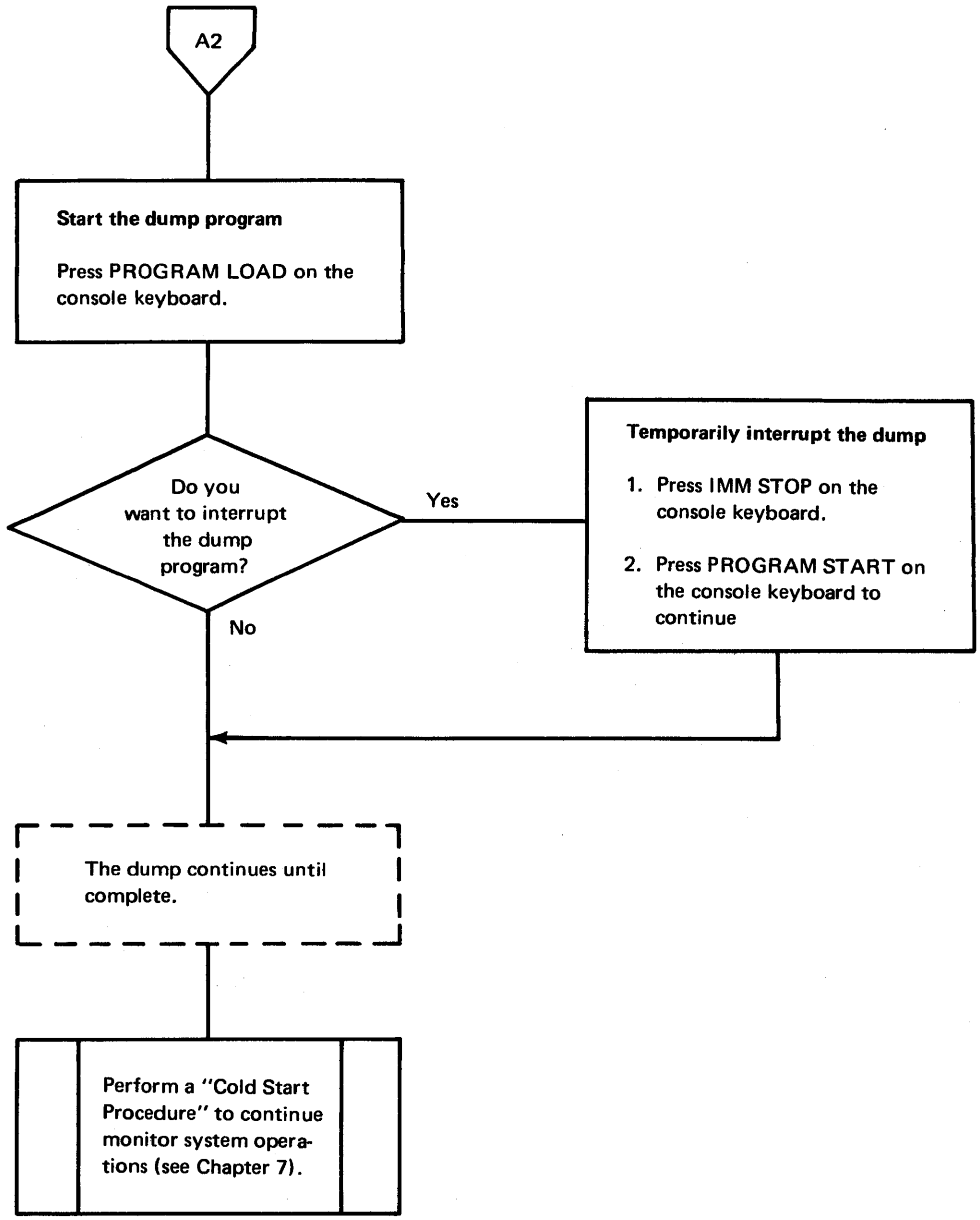


Figure 9-1 (Part 2 of 2). Console printer core dump operating procedure

PRINTER CORE DUMP PROGRAM

This program dumps core storage (in hexadecimal) beginning at location $ZEND on either the 1403 Printer or the 1132 Printer. The printer selected is the one that is ready; when both are ready, the 1403 is selected.

dump format

Each line begins with a 4-digit hexadecimal address that is followed by sixteen 4-digit hexadecimal words. A space separates the address and each word in the printed line. An additional space is inserted between each group of 4 words.

To decrease dump time, the program does not print consecutive duplicate lines. Before printing a line, the program compares the next 16 words of core with those just printed. If they are identical, the program goes on to the next 16 words of core. The program continues comparing lines until the first line not identical to the last line printed is found. The printer then spaces a line and the 16 words of the unidentical line are printed. The address printed at the beginning of this line is that of the first word of the unidentical line.

The materials that you need to use the Stand-alone Printer Core Dump Program are:

- Printer Core Dump Program card deck, SDMP punched in column 73 through 76

-or-

- Printer Core Dump Program paper tape, BP19

Figure 9-2 is the operating procedure for the stand-alone Printer Core Dump Program.

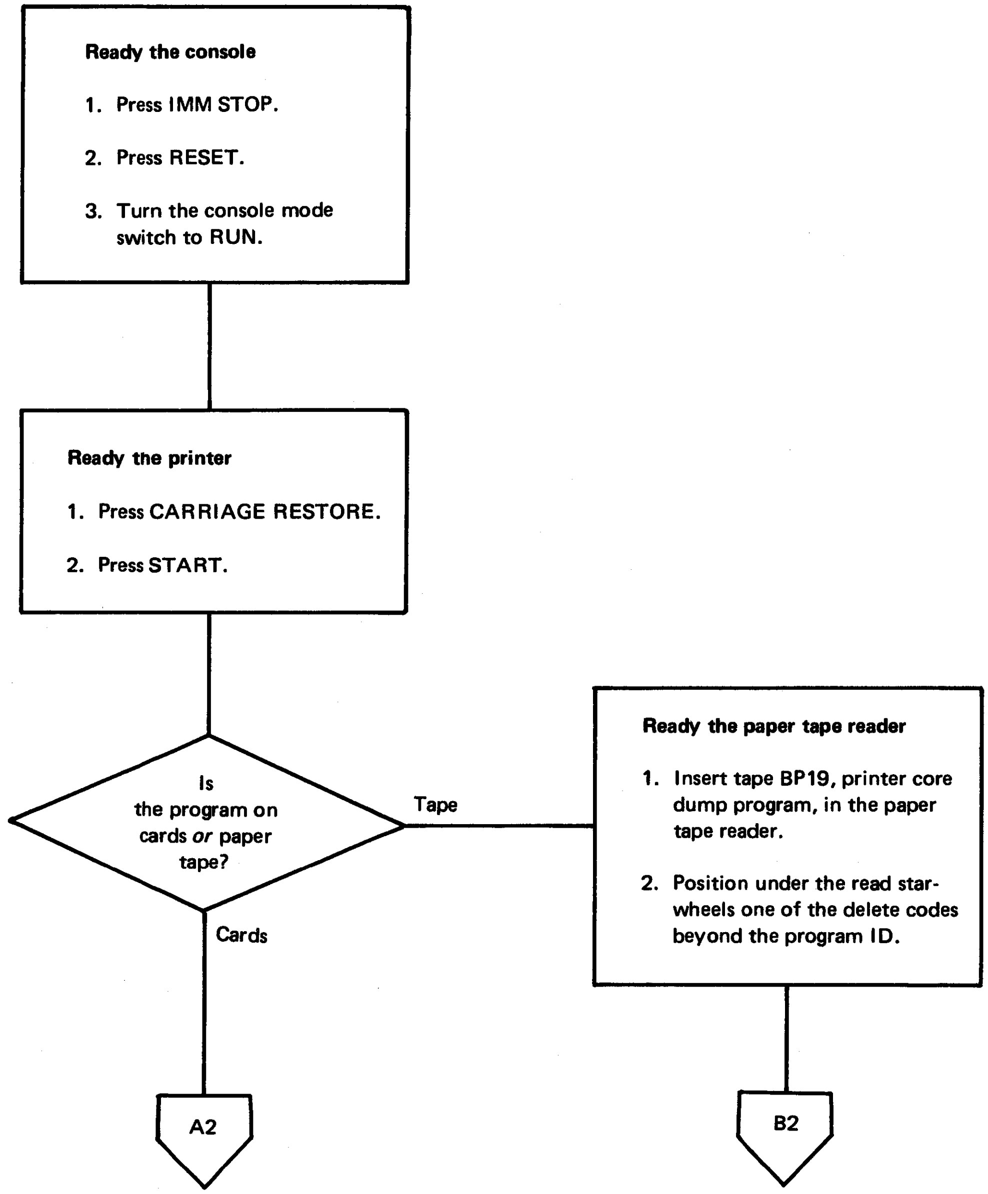


Figure 9-2 (Part 1 of 3). Printer Core Dump Program operating procedure

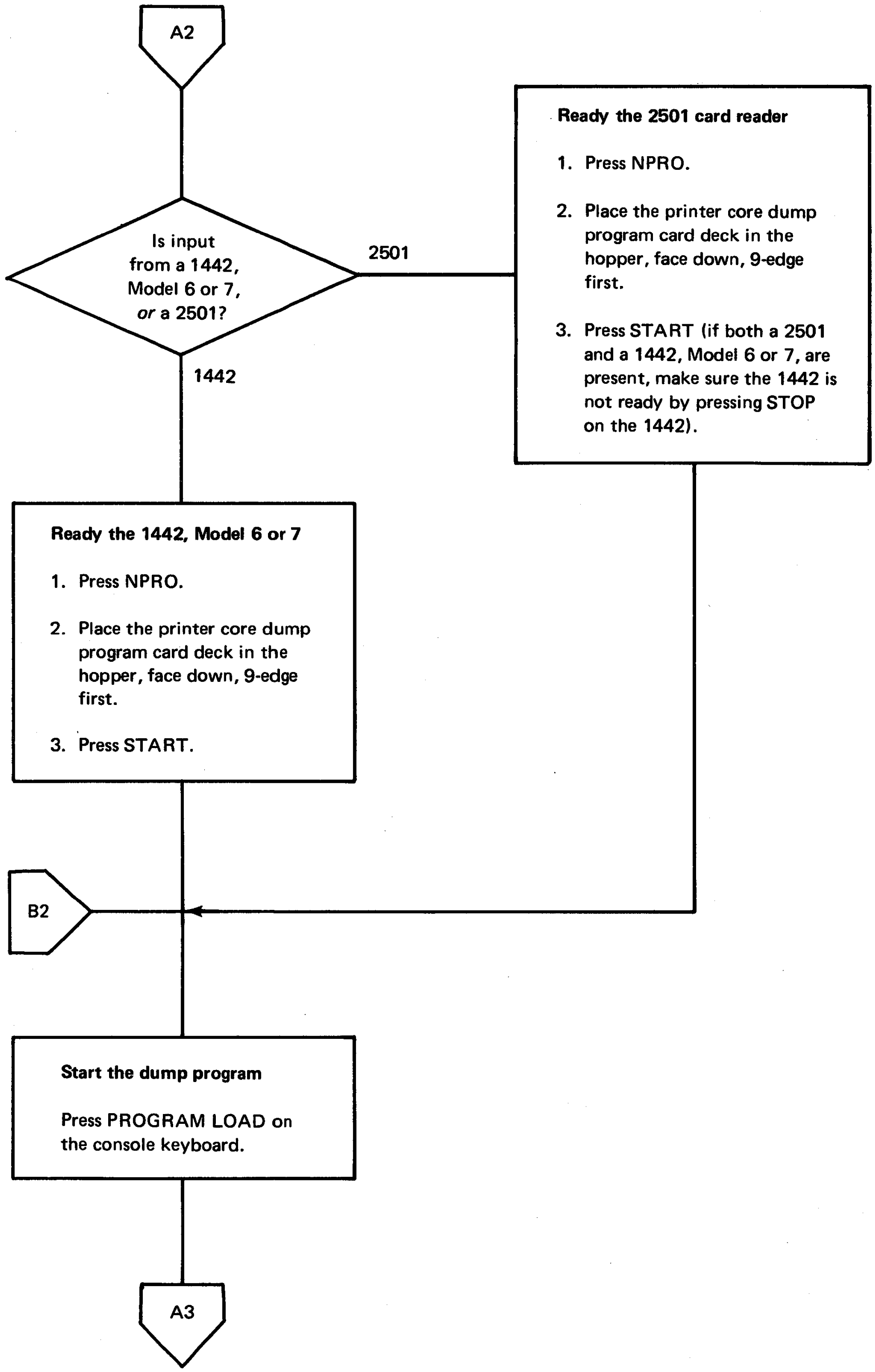


Figure 9-2 (Part 2 of 3). Printer Core Dump Program operating procedure

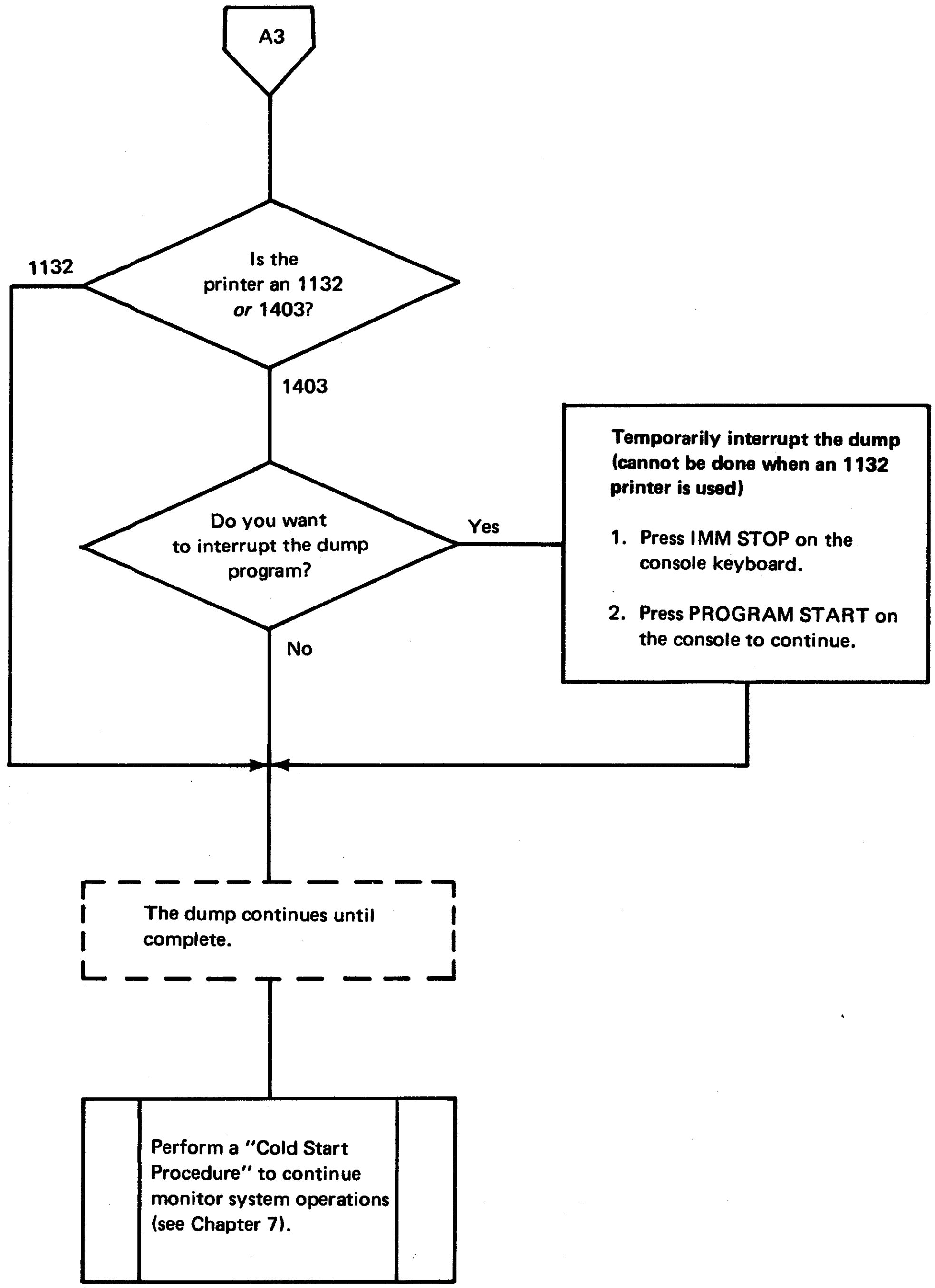


Figure 9-2 (Part 3 of 3). Printer Core Dump Program operating procedure

DISK CARTRIDGE INITIALIZATION PROGRAM (DCIP)

The Disk Cartridge Initialization Program (DCIP) is composed of:

- A disk initialization subroutine
- A disk copy subroutine
- A disk dump subroutine
- A disk patch subroutine
- A disk analysis subroutine
- A disk compare subroutine

Initialization of a cartridge is required before the monitor system can be loaded onto the cartridge. If sector @IDAD and/or sector @DCOM are destroyed on a disk, disk initialization is the only DCIP subroutine that can be performed on the disk.

The following text describes the functions of DCIP and provides sample operating procedures for using all of the functions of DCIP.

Disk Initialization Subroutine

This subroutine prepares a new disk cartridge for use and makes an old cartridge available to be used for other purposes. The initialization subroutine:

- Tests sectors to determine which, if any, are defective and fills in the defective cylinder table accordingly.
- Writes a sector address on every sector, including defective sectors.
- Establishes a file-protected area for the disk cartridge.
- Places an ID on the disk cartridge.
- Establishes a disk communications area (sector @DCOM), a location equivalence table (LET), and a core image buffer (CIB).

The monitor system disk I/O subroutines operate with up to 3 defective cylinders on a cartridge. That is, 3 cylinders that contain one or more defective sectors. A cartridge cannot be initialized if cylinder 0 is defective, or if a sector address cannot be written on every sector.

The contents of sectors @IDAD, @DCOM, and @RIAD in cylinder 0 are established during initialization (see Chapter 2 for a general description of the contents of these sectors). A message and the program that prints it are written in sector @IDAD. The message is:

THIS IS A NONSYSTEM CARTRIDGE

This message is printed when an attempt is made to cold start a nonsystem cartridge that is initialized with DCIP.

Disk Copy Subroutine

This subroutine copies the contents from one cartridge (the source cartridge) onto another cartridge (the object cartridge). Before the copy is performed, the subroutine checks to ensure that the cartridge being copied and the object cartridge have been initialized. The cartridge ID, copy code, and defective cylinder data are not copied from the source cartridge.

Disk Dump Subroutine

This subroutine dumps sectors of a cartridge that you select on the principal printer.

Each sector is preceded by a 3-word header and is printed in 20 lines; sixteen 4-digit hexadecimal words per line. Two sectors are printed on each page.

The first digit of the first header word is the drive number; the remaining 3 digits are the physical sector address of the sector being dumped. The second header word is the actual address of the sector being dumped. The third word is the logical sector address, taking into account any defective cylinders. If you dump a sector that is in a defective cylinder, the third word of the header contains DEFC.

Disk Patch Subroutine

This subroutine allows you to change the contents, word-by-word, of selected disk sectors. The contents of the sector being modified are printed, on the principal printer, both before and after the changes are made.

A one-word buffer is used to store the contents of a specified word as you are modifying it. Six special characters are used to control the use of this buffer. These characters and their functions are listed in the disk patch operating procedure in Figure 9-7 under "DCIP Operating Procedures" in this chapter.

Disk Analysis Subroutine

This subroutine reads each sector of a selected cartridge 16 times.

If a read error occurs, the address of the sector being read is printed. You can then dump the contents of the sector in error if you wish.

If a sector address is incorrect, the incorrect address is printed, and the correct address is then written on the sector.

Disk Compare Subroutine

This subroutine of DCIP reads the corresponding sectors of 2 cartridges and compares the contents word by word. The addresses from both cartridges of any sectors that do not compare are printed.

DCIP Operating Procedures

The operating procedures in this section include a program load procedure (Figure 9-3) for DCIP and procedures (Figures 9-4 through 9-9) for performing the 6 functions of DCIP.

The following general comments should be kept in mind while using any of the DCIP functions.

1. If a disk drive is not ready, the system halts with /50X0 displayed in the ACCUMULATOR on the console display panel; X is the number of the physical drive that is not ready.
2. If your system has 2 card readers, ready only the reader that you use for cold start.
3. The messages printed during DCIP functions refer to the console entry switches as *bit* switches.
4. All console entry switch settings that you enter are printed on the console printer as 4-digit hexadecimal numbers.
5. If you turn on an invalid console entry switch during any of the DCIP functions, ENTRY ERR . . .RETRY is printed. To continue, turn off the incorrect switch, turn on the correct one, and press PROGRAM START.

6. A DCIP function can be stopped at any time by pressing INT REQ on the console keyboard. The system prints the DCIP option message. This gives you the choice of repeating the current function or selecting a new one. Following the option message, you can change disk cartridges or packs, if necessary, before continuing. If you wish to discontinue using DCIP at this point, perform a cold start procedure (see Chapter 7) to make the monitor system operational.

   Note. If you press INT REQ while a disk is being copied or initialized, the results of the use of the object cartridge (in the copy operation) or the partially initialized cartridge are unpredictable.

The materials that you need to perform the function of DCIP are the IBM-supplied DCIP card deck (DCIP punched in columns 73 through 76) or paper tape (BP16) and any of the following depending on the function you are using:

- An uninitialized disk for disk initialization
- A system or nonsystem cartridge and an initialized disk for the copy function. The copy function is usable only if your system can contain more than one disk at a time.
- A system or nonsystem cartridge for the dump function
- A system or nonsystem cartridge for the disk patch function
- A system or nonsystem cartridge for disk analysis
- Two system or nonsystem cartridges whose contents are supposed to be the same for the disk compare function. The compare function is usable only if your system can contain more than one disk at a time.

Have all of the cartridges you are going to use ready before you load the DCIP program as follows.

Ready the console

1. Press IMM STOP.
2. Press RESET.
3. Turn the load mode switch to RUN.

Ready the printer

1. Press CARRIAGE RESTORE.
2. Press START.

Note. If the 1403 or 1132 Printer is not ready when you load DCIP, or if your system does not have a 1403 or 1132, the console printer is the principal print device.

Ready the disk drives to be used during the DCIP functions

1. Place the disk cartridges or packs to be used in the disk drives.
2. Turn the disk drives on (see Chapter 7 for more information about readying disk drives).

A2

Figure 9-3 (Part 1 of 4). Load DCIP operating procedure

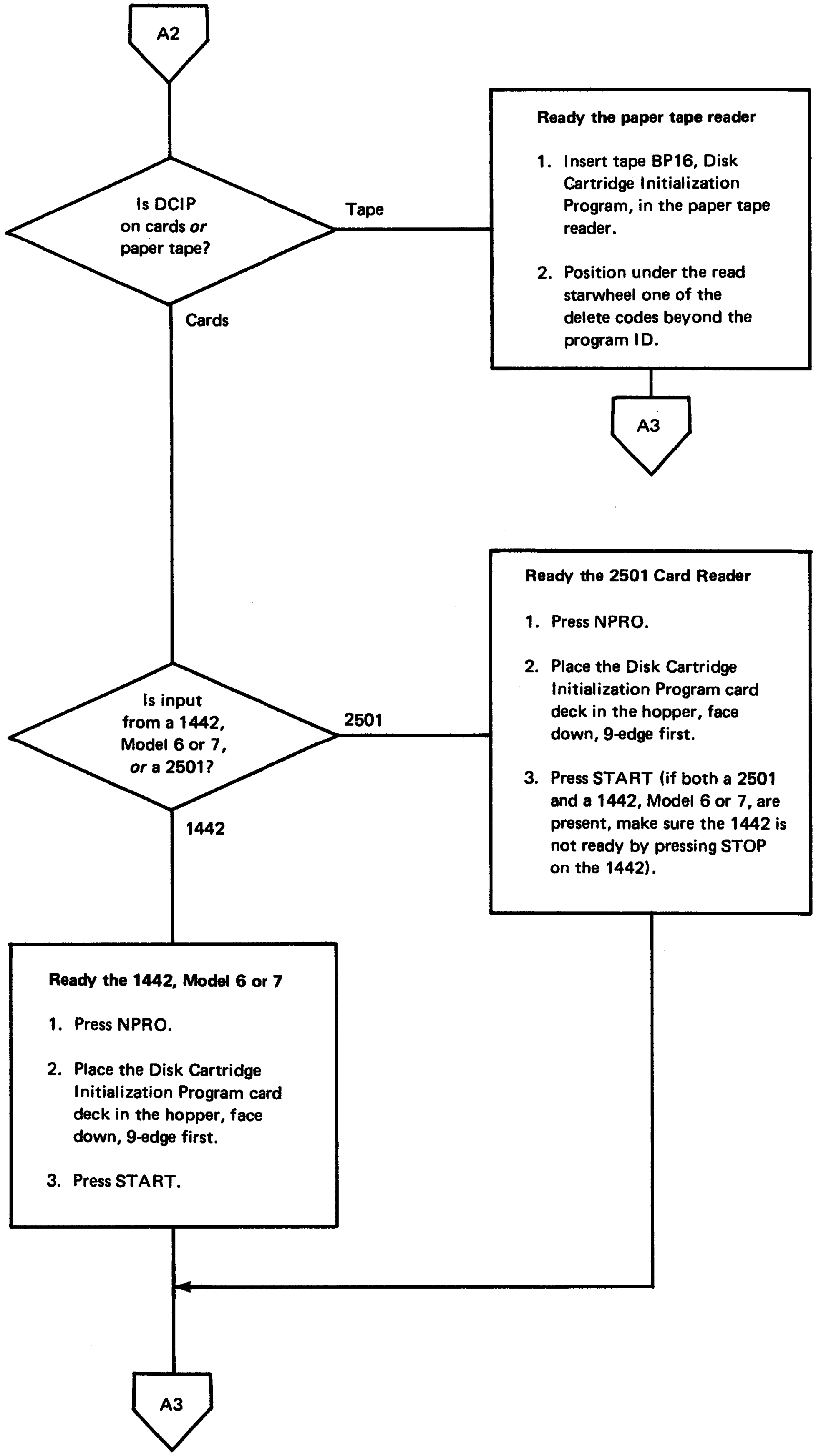


Figure 9-3 (Part 2 of 4). Load DCIP operating procedure

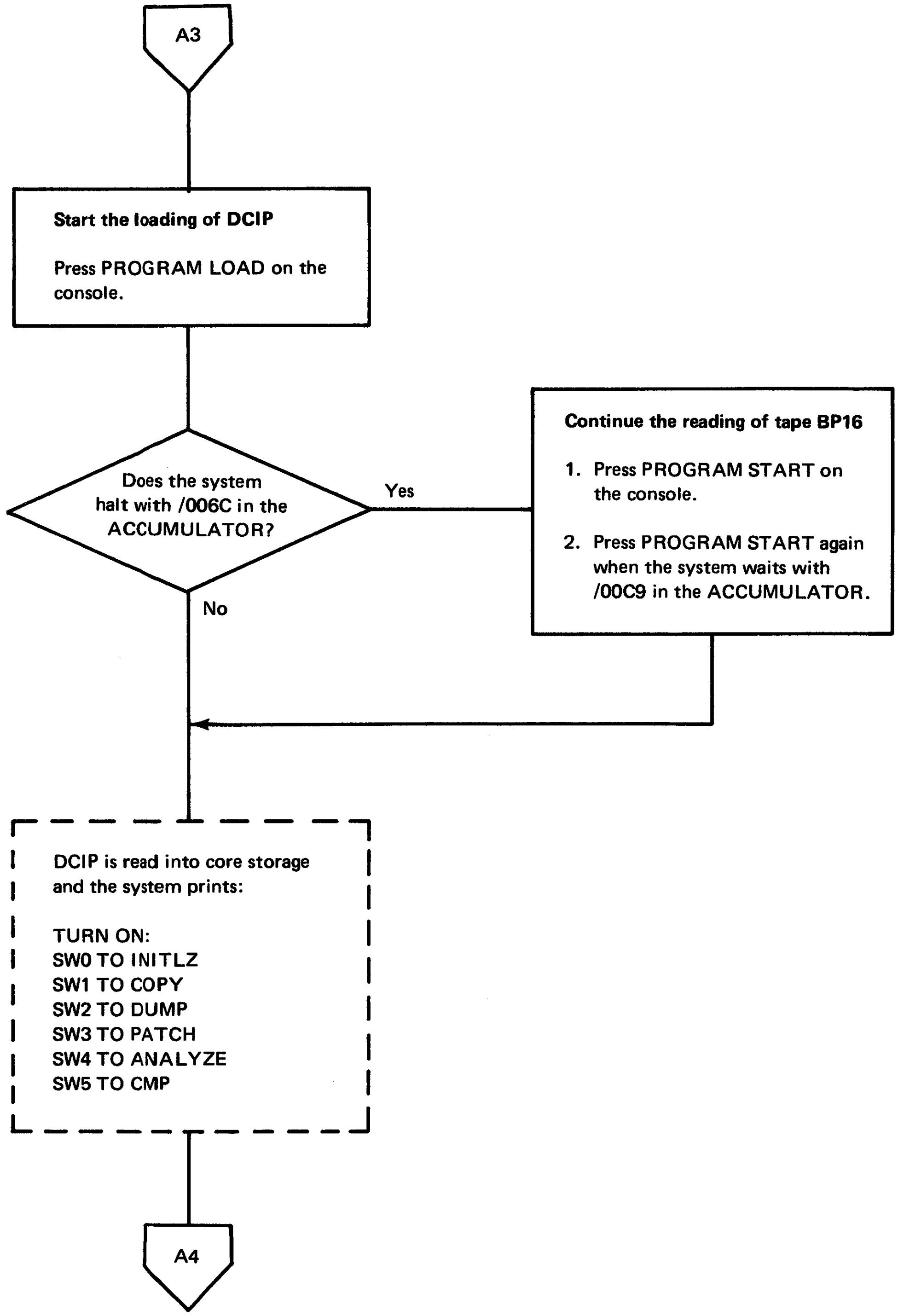


Figure 9-3 (Part 3 of 4). Load DCIP operating procedure

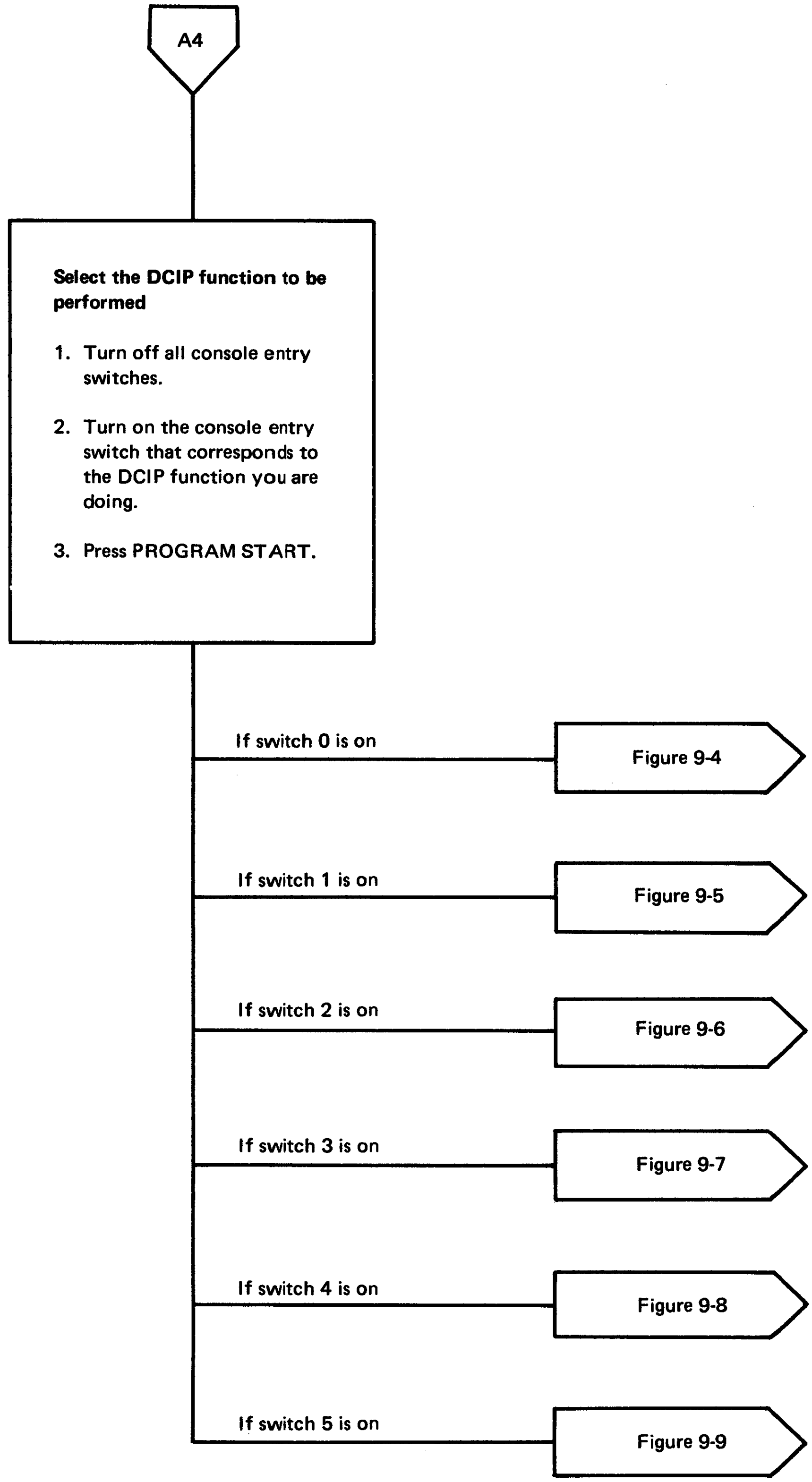


Figure 9-3 (Part 4 of 4). Load DCIP operating procedure

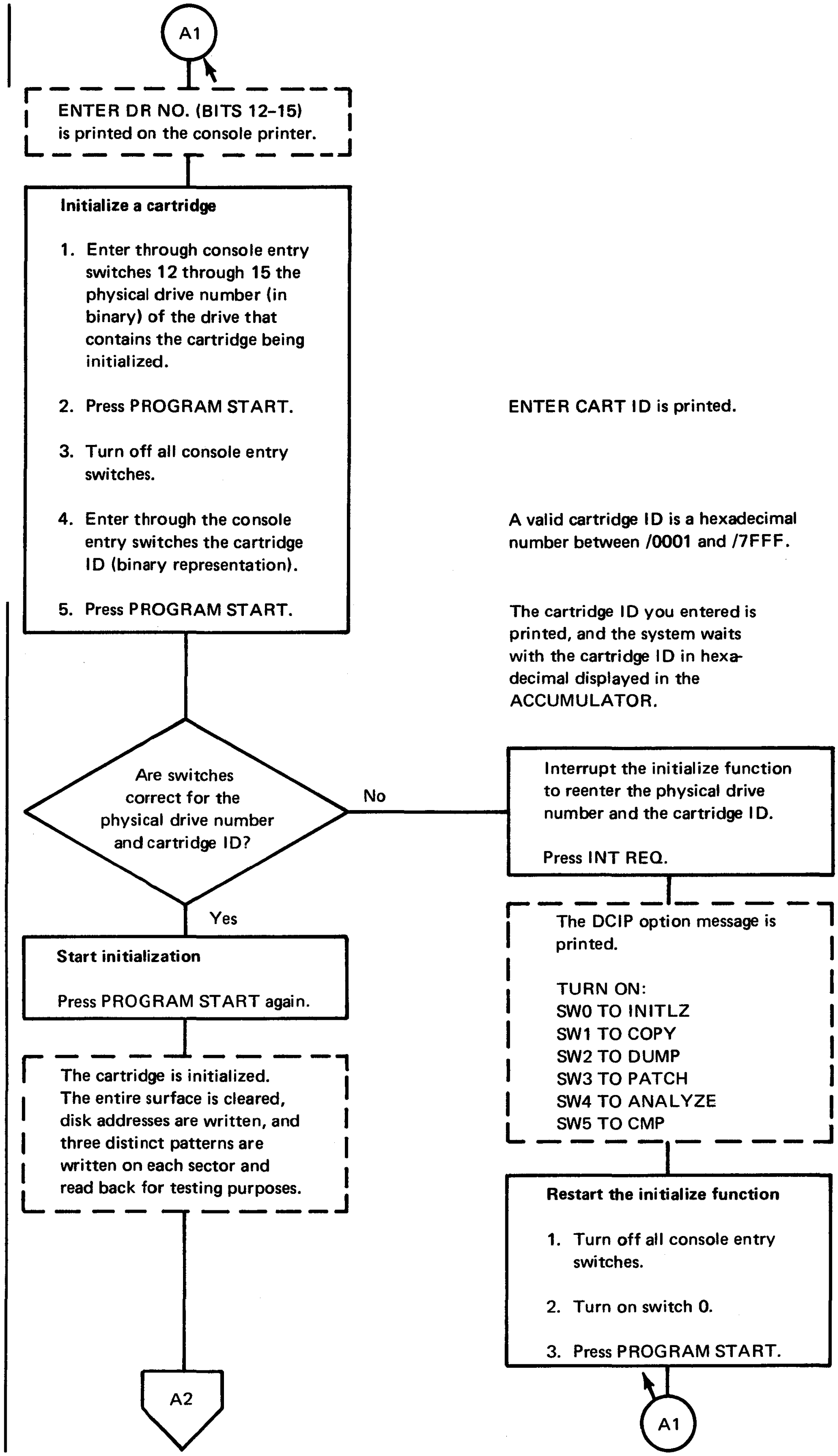


Figure 9-4 (Part 1 of 5). Operating procedure for DCIP initialize function

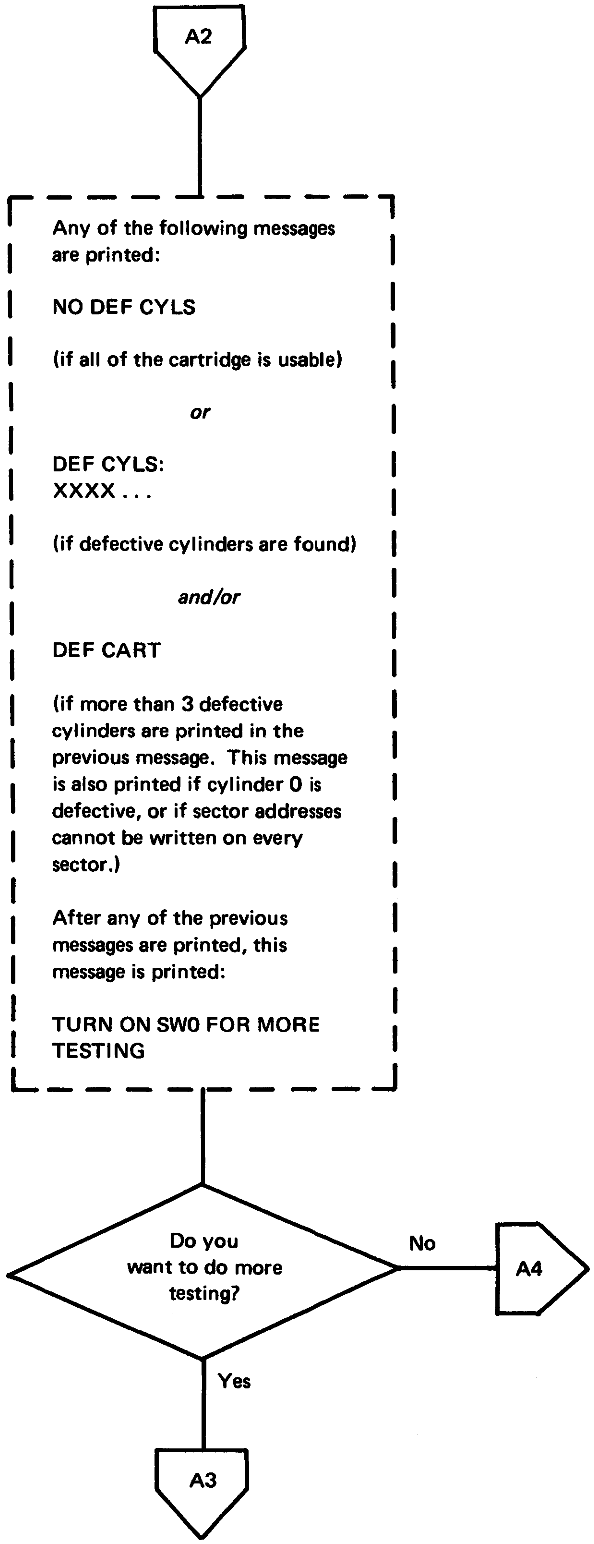


Figure 9-4 (Part 2 of 5). Operating procedure for DCIP initialize function

A3

Continue testing the cartridge

1. Turn off all console entry switches.
2. Turn on switch 0.
3. Press PROGRAM START.

ENTER REPEAT CNT: (BITS 11-15) is printed.

4. Enter through console entry switches 11 through 15 the number (in binary) of times you want the cartridge tested. (This provides an additional opportunity to find marginal cylinders and reduce chances of disk errors later on.)

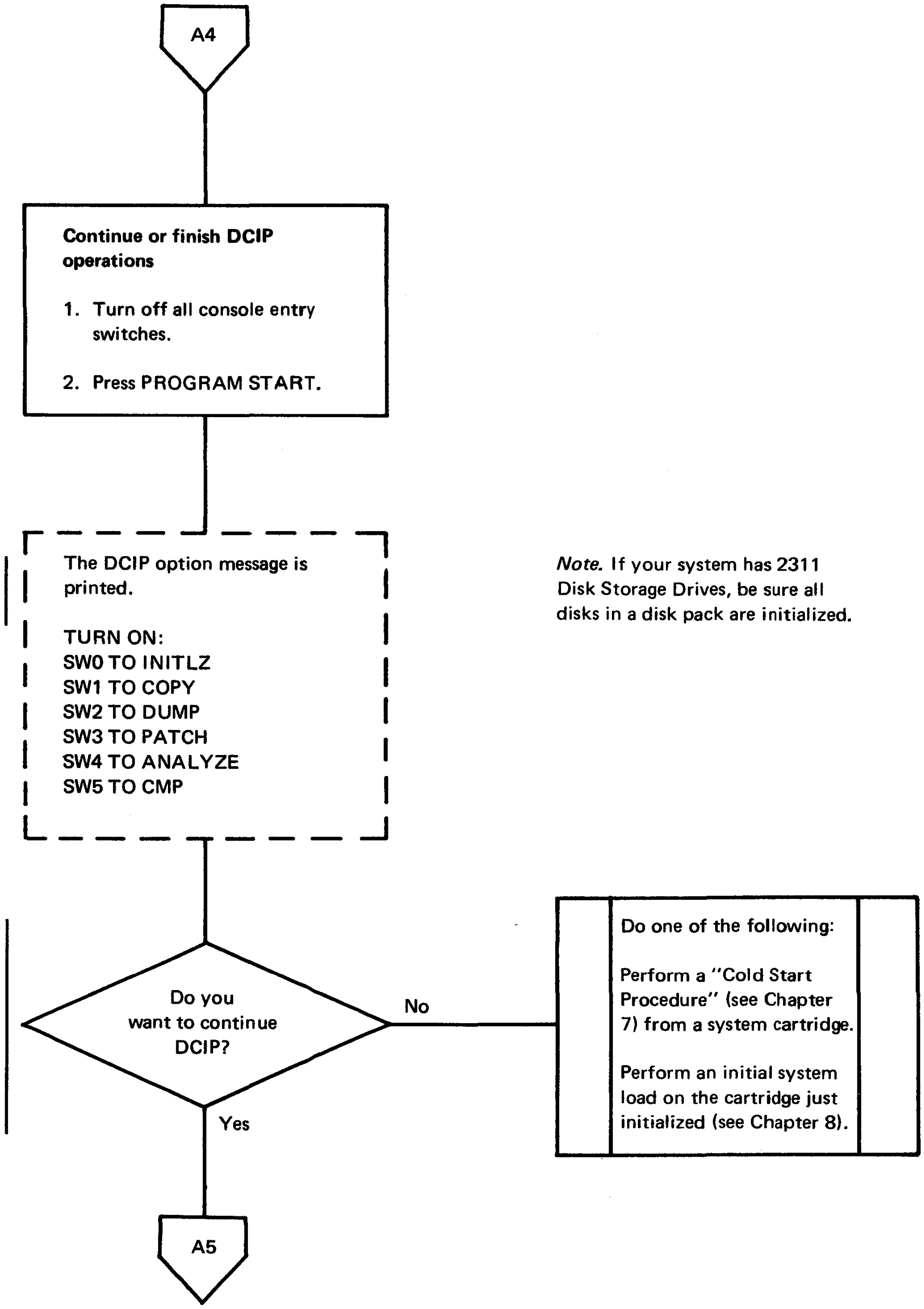
A maximum of 31 (decimal) can be entered.

5. Press PROGRAM START.

Initialization is repeated. Each cylinder is tested the number of times you specified.

A2

Figure 9-4 (Part 3 of 5). Operating procedure for DCIP initialize function



Note. If your system has 2311 Disk Storage Drives, be sure all disks in a disk pack are initialized.

Figure 9-4 (Part 4 of 5). Operating procedure for DCIP initialize function

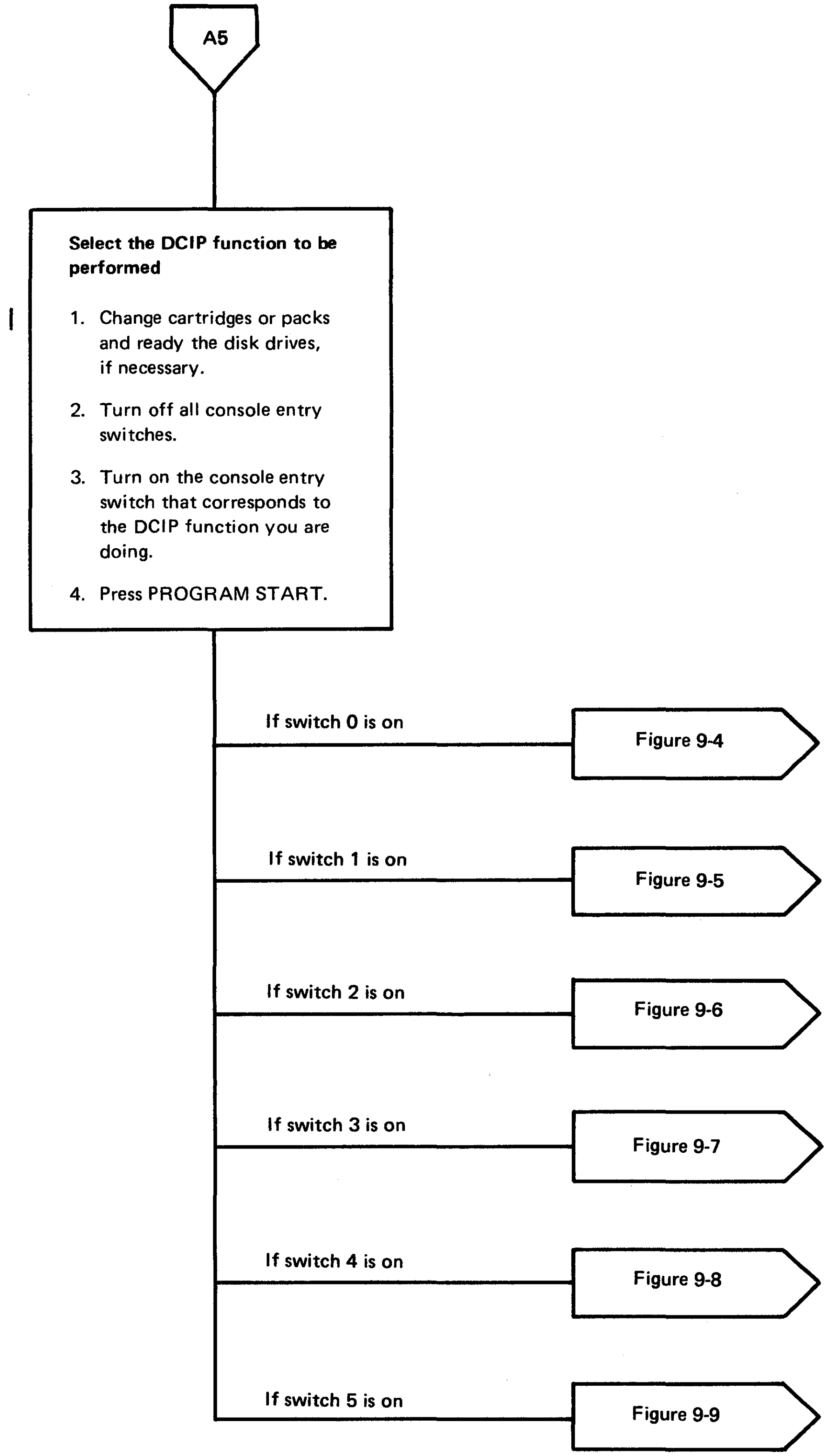


Figure 9-4 (Part 5 of 5). Operating procedure for DCIP initialize function

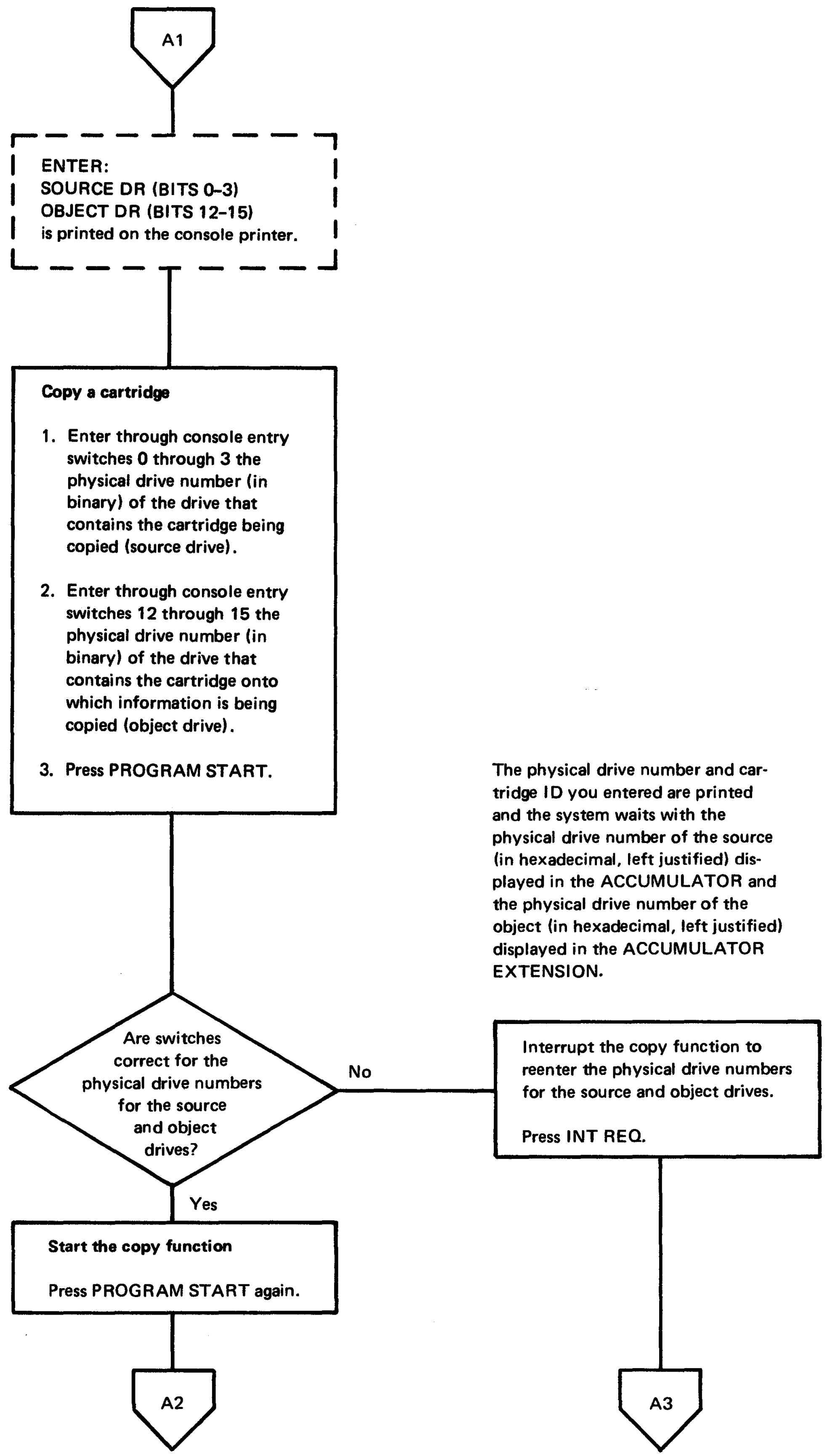


Figure 9-5 (Part 1 of 8). Operating procedure for DCIP copy function

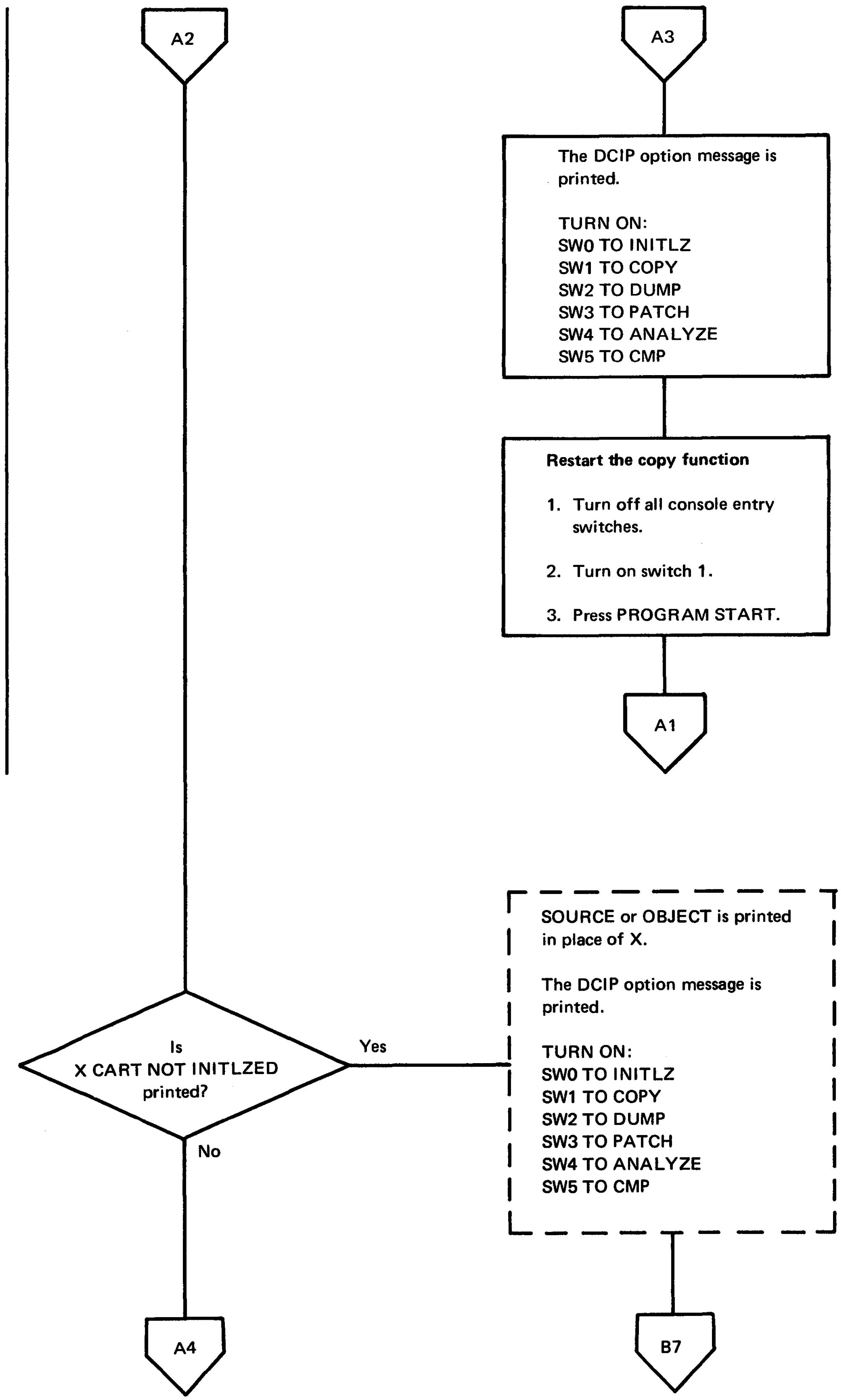


Figure 9-5 (Part 2 of 8). Operating procedure for DCIP copy function

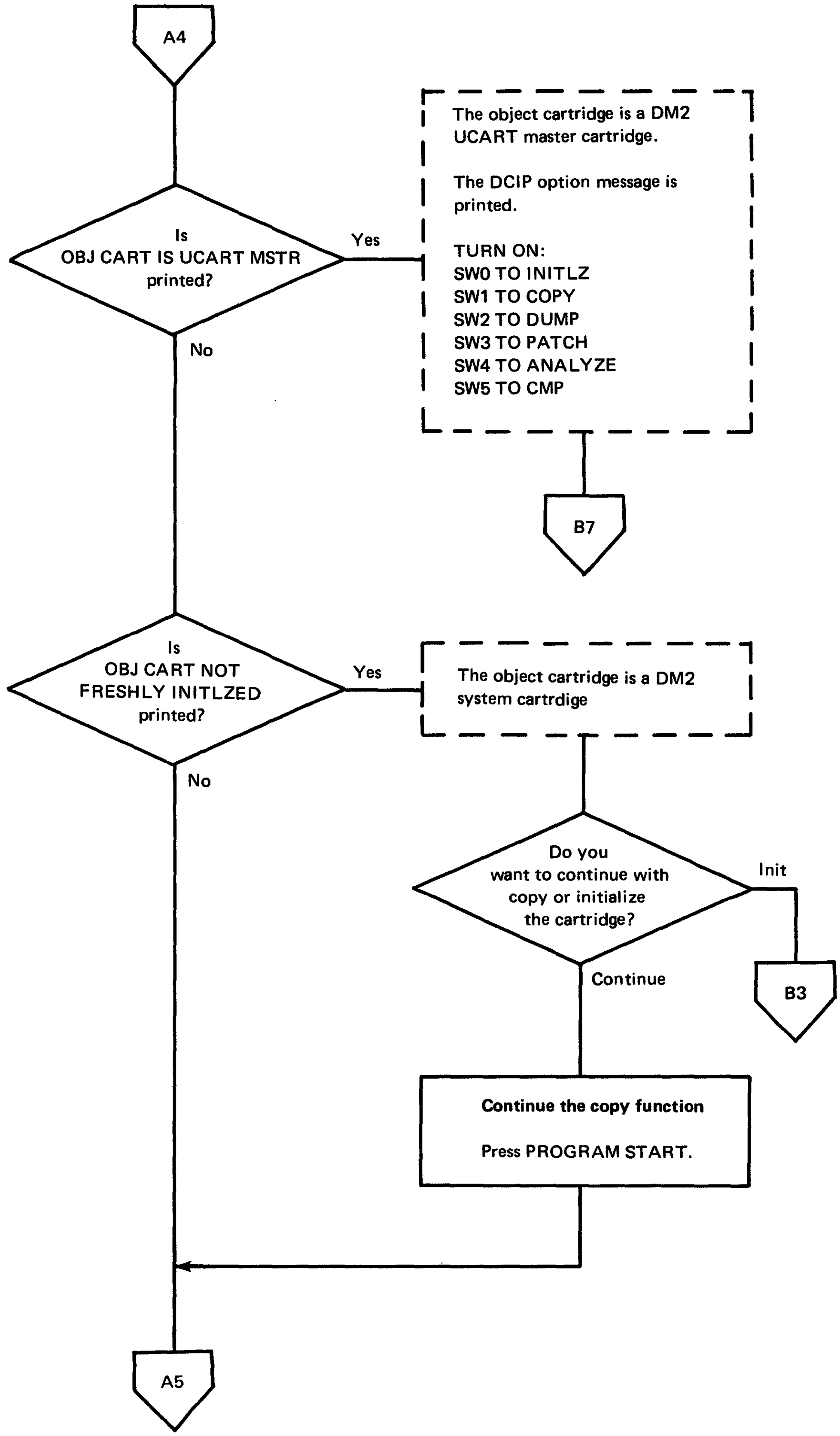


Figure 9-5 (Part 3 of 8). Operating procedure for DCIP copy function

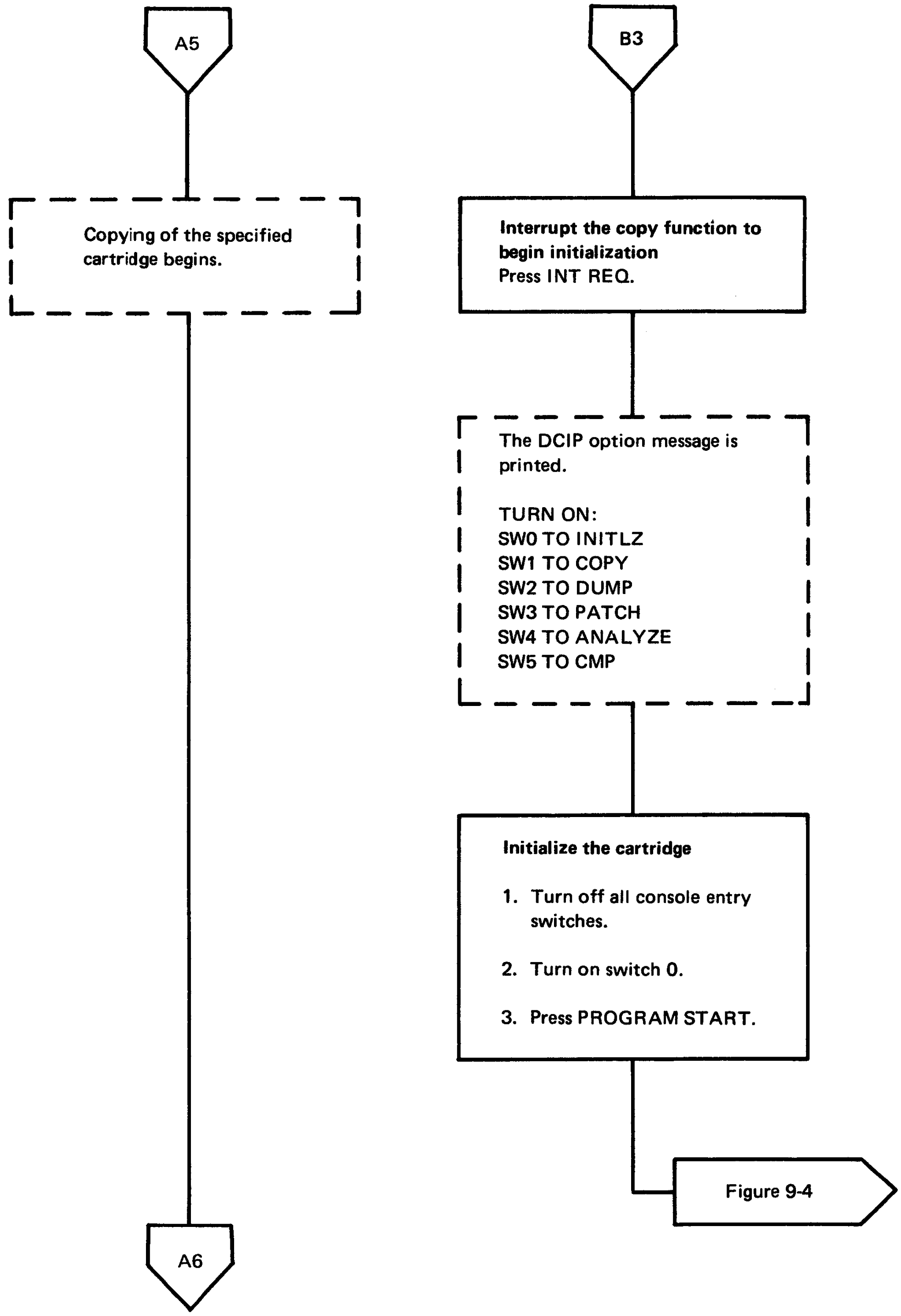


Figure 9-5 (Part 4 of 8). Operating procedure for DCIP copy function

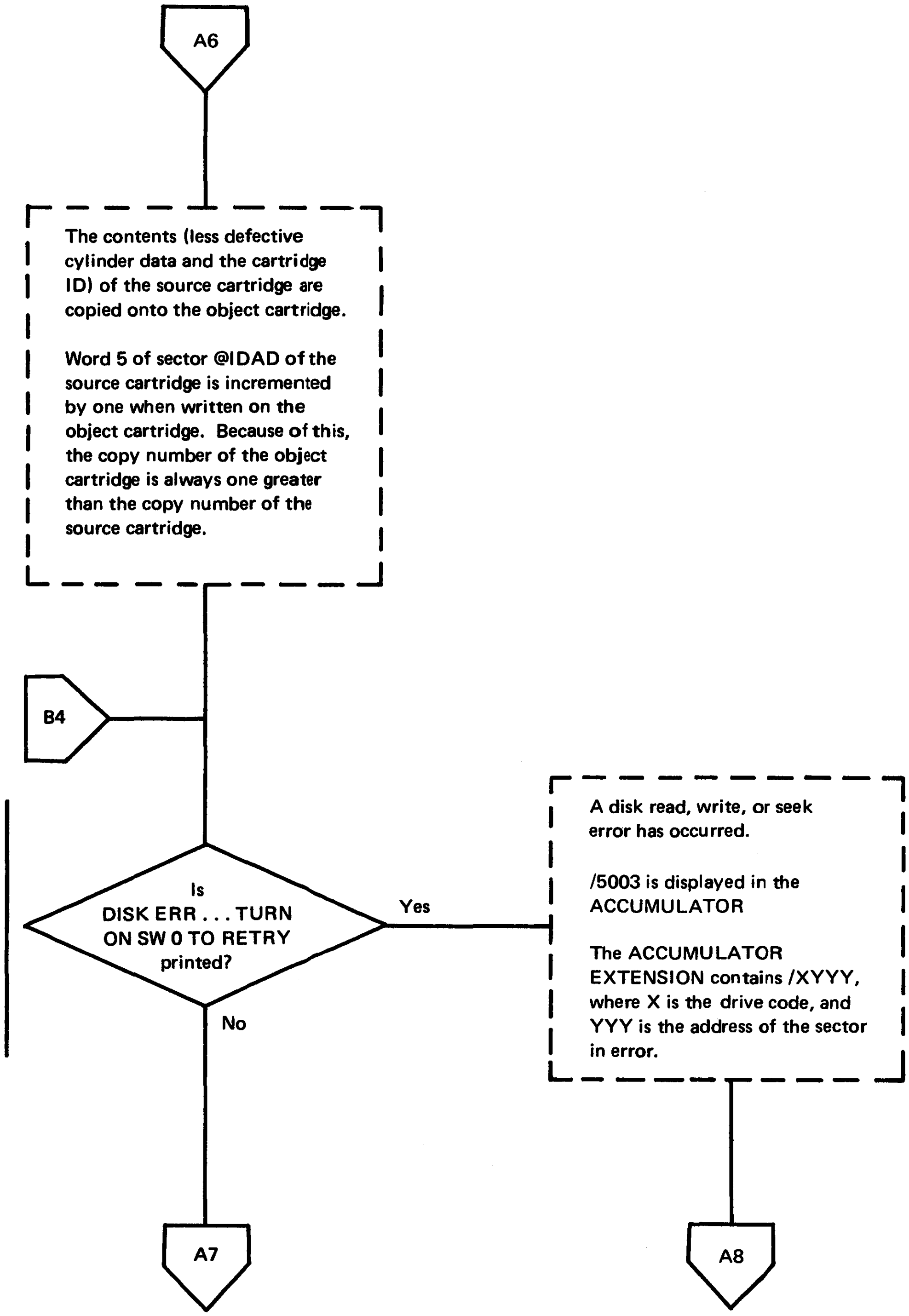


Figure 9-5 (Part 5 of 8). Operating procedure for DCIP copy function

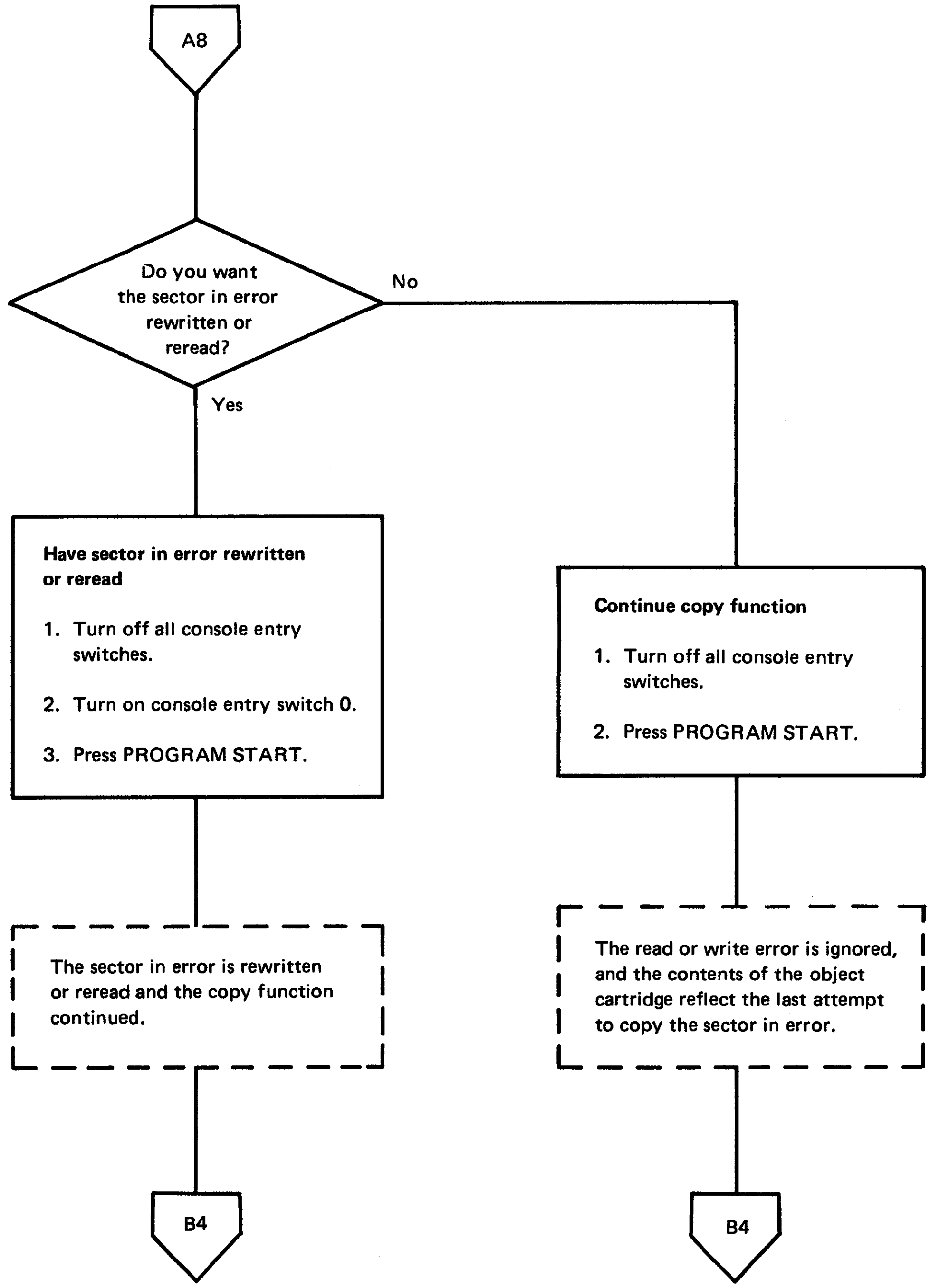


Figure 9-5 (Part 6 of 8). Operating procedure for DCIP copy function

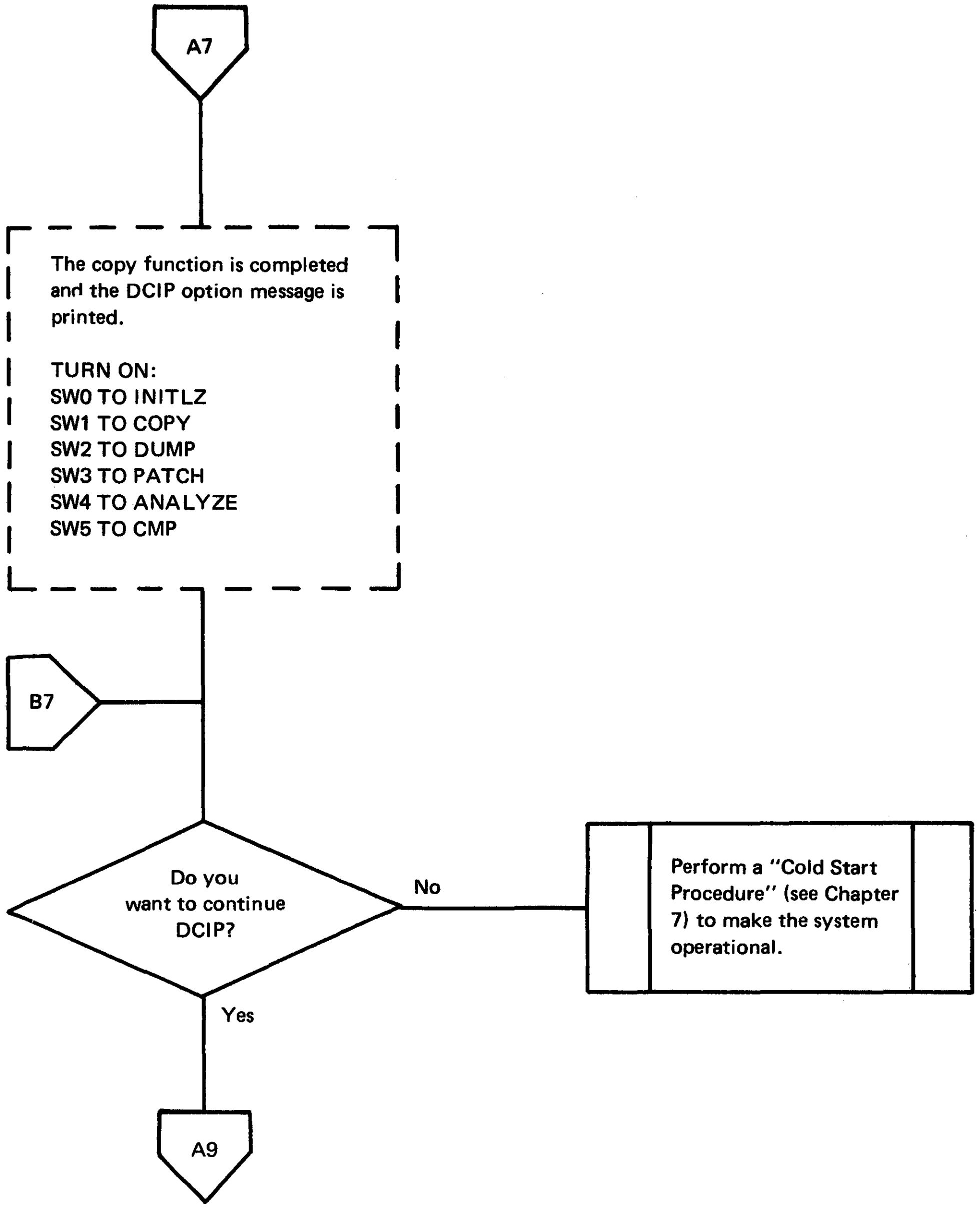


Figure 9-5 (Part 7 of 8). Operating procedure for DCIP copy function

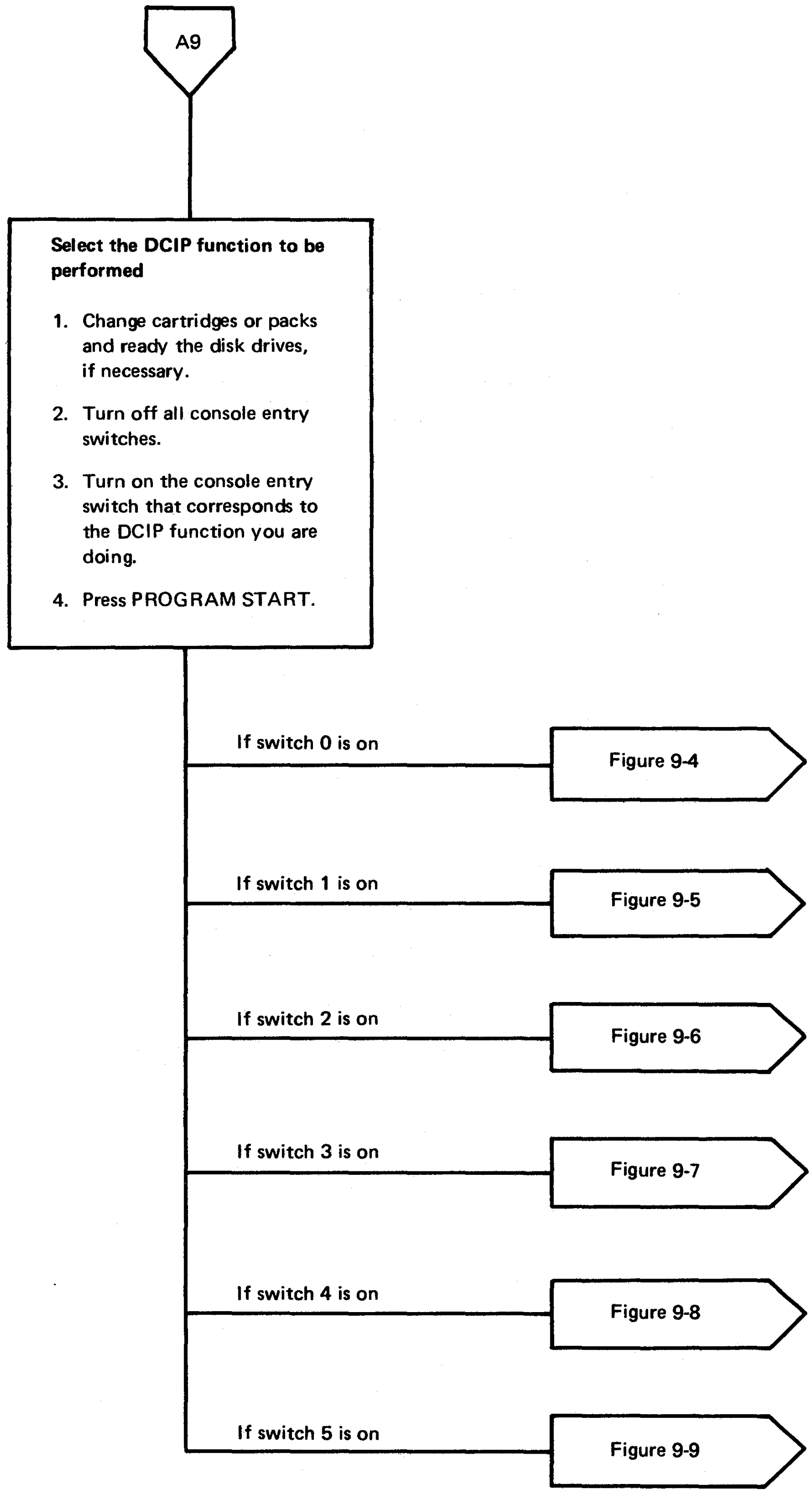


Figure 9-5 (Part 8 of 8). Operating procedure for DCIP copy function

ENTER. . . PHYS DR NO. (BITS 0–3)
SECTR ADDR (BITS 4–15)
is printed on the console printer

Dump specified sectors of a disk cartridge

1. Enter through console entry switches 0 through 3 the physical drive number (in binary) of the drive that contains the cartridge from which data is being dumped.

2. Enter through console entry switches 4 through 15 the physical address of the first sector being dumped.

   The sector address is a right-adjusted hexadecimal number, maximum /0657. (A logical sector address, obtained from LET or FLET, must be adjusted for defective cylinders.)

3. Press PROGRAM START.

   ENTER NO. OF SCTRS TO DUMP is printed.

4. Enter through the console entry switches the number of consecutive sectors to be dumped.

   The number is a right-adjusted hexadecimal value; the maximum value depends on the starting address entered in Step 2.

5. Press PROGRAM START.

Dumping of the cartridge begins.

A2

Figure 9-6 (Part 1 of 4). Operating procedure for DCIP dump function

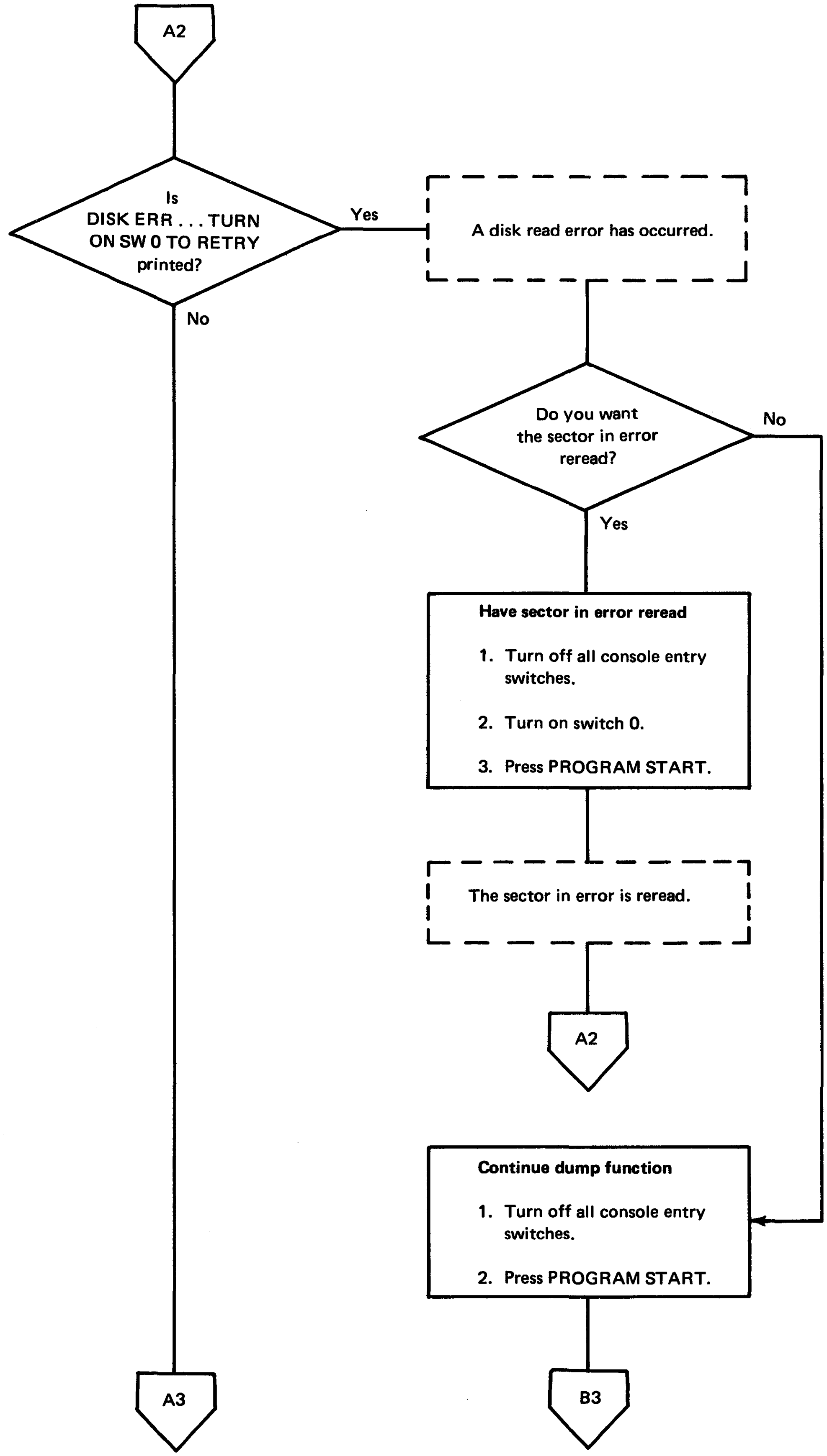


Figure 9-6 (Part 2 of 4). Operating procedure for DCIP dump function

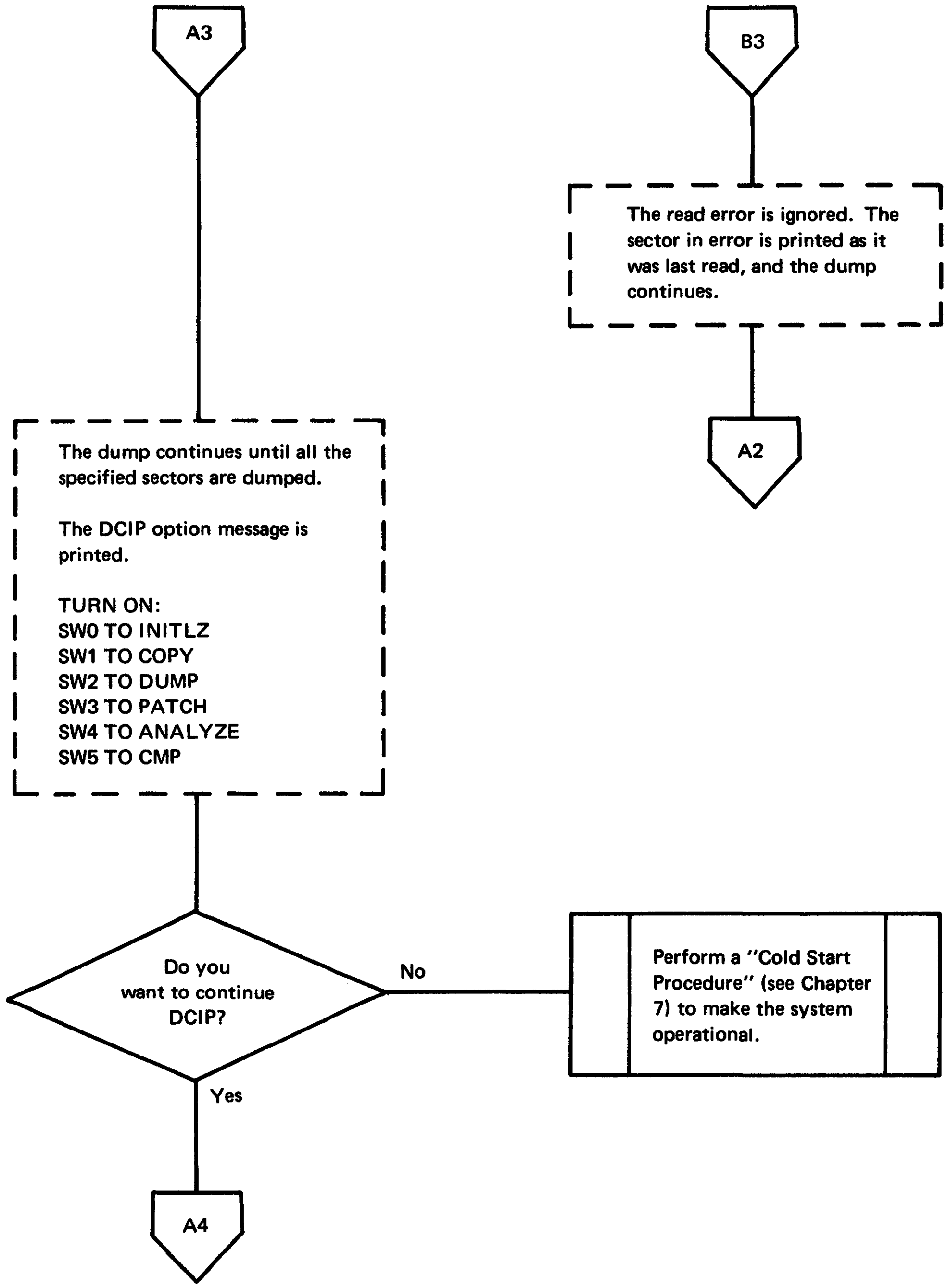


Figure 9-6 (Part 3 of 4). Operating procedure for DCIP dump function

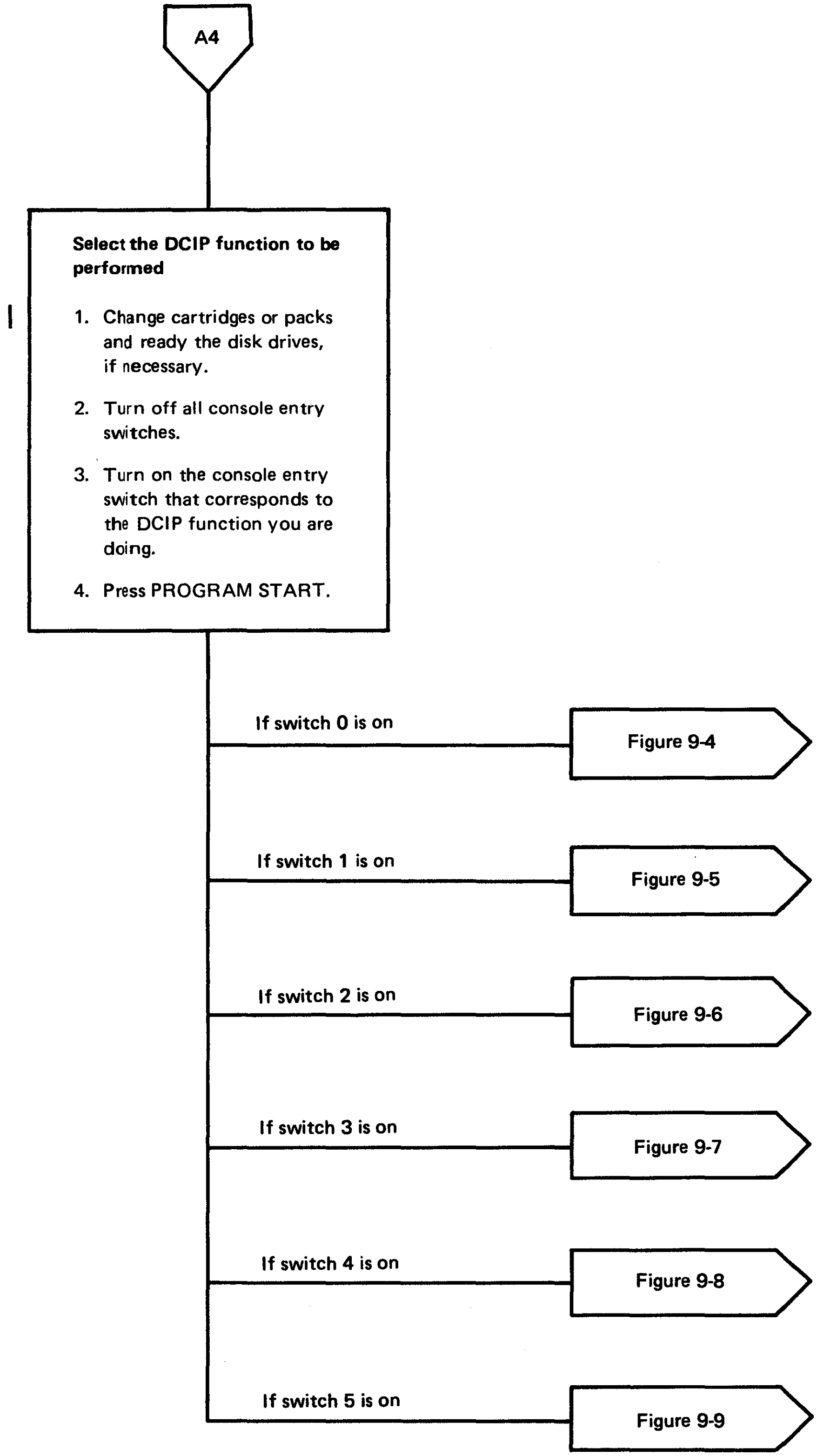


Figure 9-6 (Part 4 of 4). Operating procedure for DCIP dump function

ENTER:
PHYS DR NO. (BITS 0-3)
SCTR ADDR (BITS 4-15)
is printed on the console printer.

Start the patch function

1. Enter through console entry switches 0 through 3 the physical drive number (in binary) of the drive that contains the cartridge being patched.

2. Enter through console entry switches 4 through 15 the address of the sector being patched.

   The sector address is a right-adjusted hexadecimal number, maximum /0657.

3. Press PROGRAM START.

   The specified sector is dumped, and the following message is printed:
   ENTER RLTV ADDR OF SCTR WD TO CHANGE.

4. Enter through the console entry switches the relative address of the sector word being changed.

   The relative address of the sector word is a right-adjusted hexadecimal number in the range /0000 through /013F.

   Note: If the sector address is being changed, enter /FFFF (-1).

5. Press PROGRAM START.

   The KEYBOARD SELECT indicator on the console keyboard is turned on.

A2

Figure 9-7 (Part 1 of 4). Operating procedure for DCIP patch function

A2

Six special character keys of the console keyboard are used to control patch functions. The 6 keys and their functions are:

EOF — causes the last 4 hexadecimal characters entered through the keyboard to be stored at the relative address displayed in the ACCUMULATOR EXTENSION.

> — causes the relative address in the ACCUMULATOR EXTENSION to be incremented by one word.

< — causes the relative address in the ACCUMULATOR EXTENSION to be decremented by one word. The address cannot be decremented past the first data word (relative address /0000) by this character. /FFFF must be entered through the keyboard.

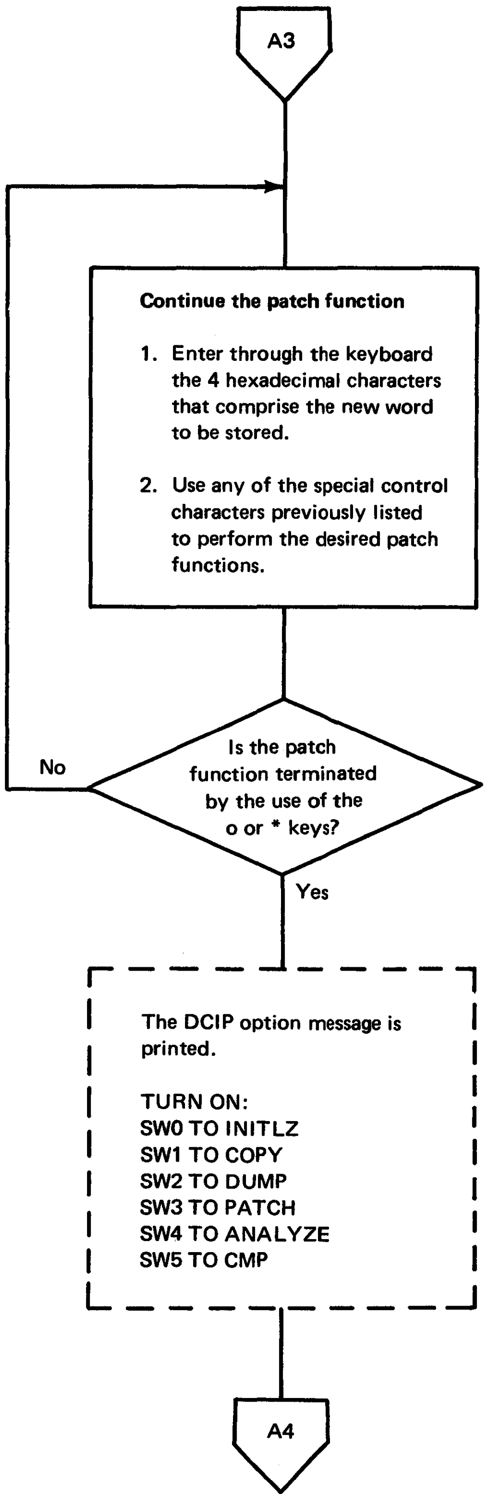
R — causes printing of the message that requests the relative address of the sector word to be changed. Thus, the relative address can be changed by more than one word.

. — causes all remaining words of the sector from the address in the ACCUMULATOR EXTENSION to the end of the sector to be filled with the last 4 hexadecimal characters entered through the keyboard. Then patching is terminated.

* — terminates the patch function. The modified sector is stored on the disk, and is dumped to the principal printer.

A3

Figure 9-7 (Part 2 of 4). Operating procedure for DCIP patch function



The characters are printed on the console printer as you enter them.

ENTRY ERR . . . RETRY is printed if you press an invalid key; use the correct key to continue.

Figure 9-7 (Part 3 of 4). Operating procedure for DCIP patch function

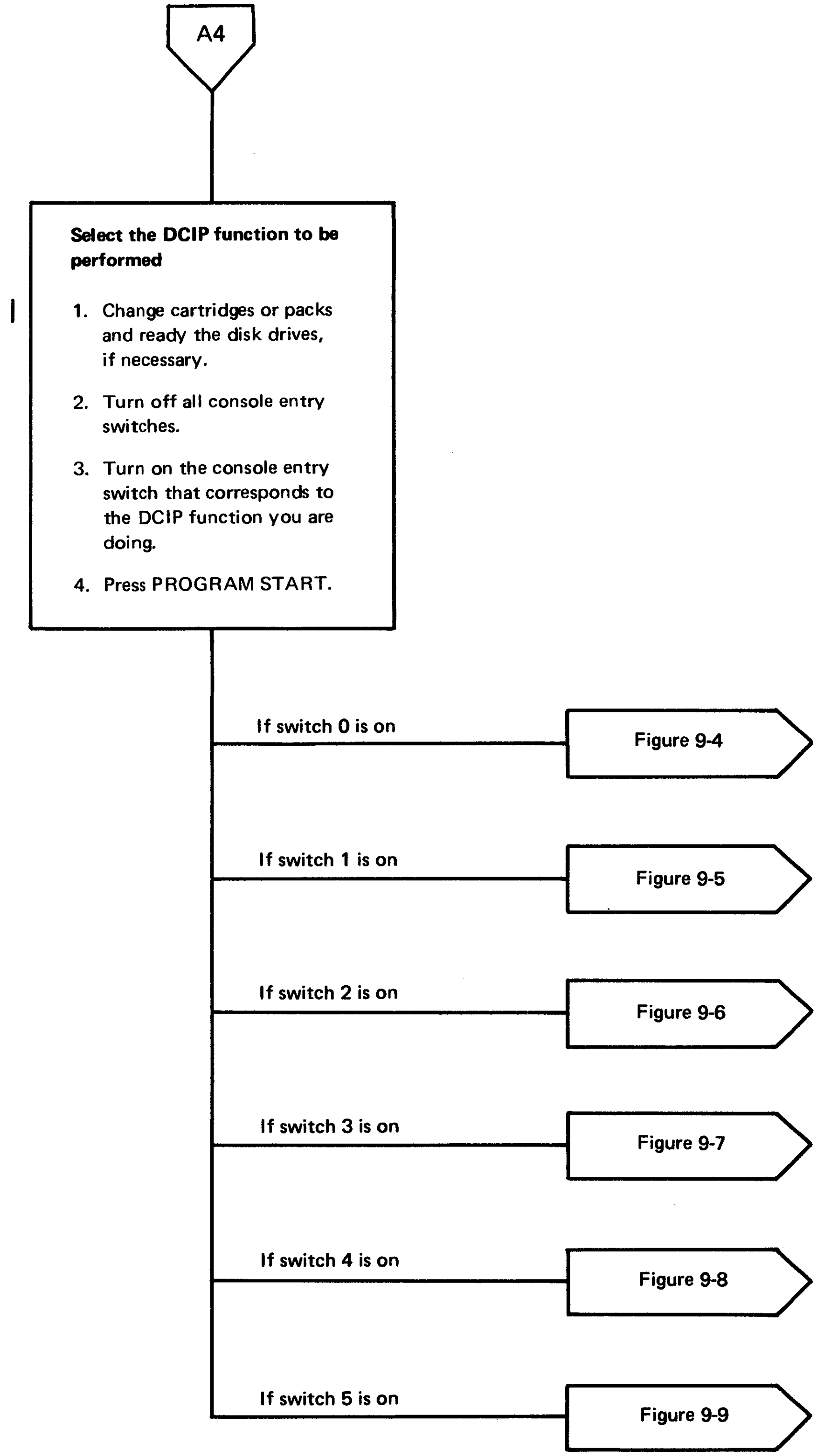


Figure 9-7 (Part 4 of 4). Operating procedure for DCIP patch function

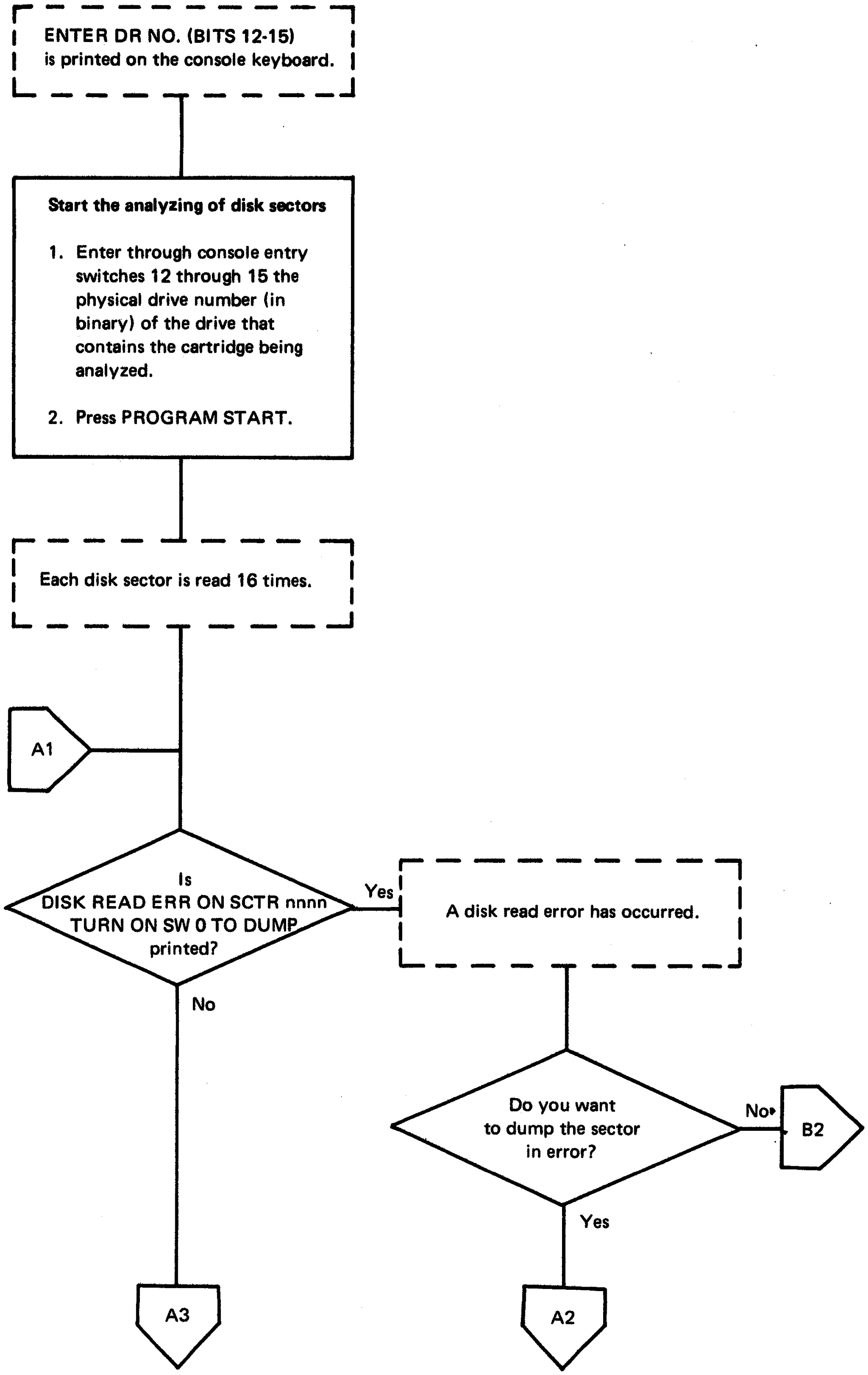


Figure 9-8 (Part 1 of 4). Operating procedure for DCIP analysis function

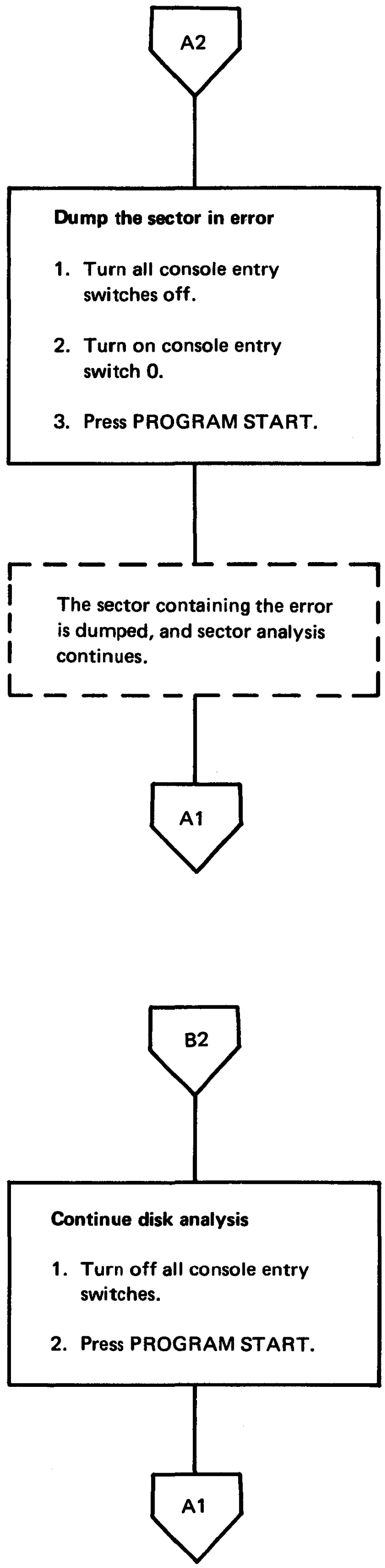


Figure 9-8 (Part 2 of 4). Operating procedure for DCIP analysis function

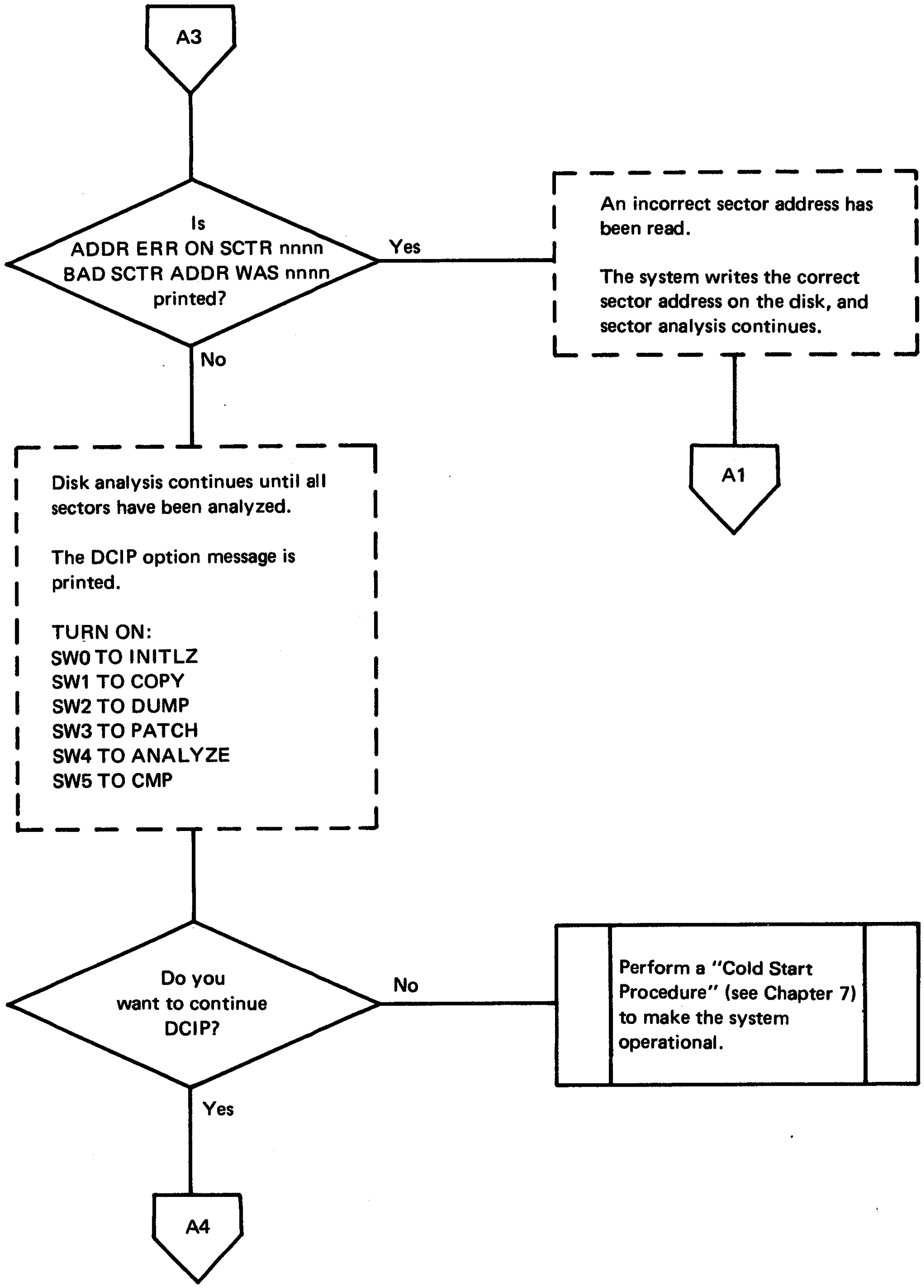


Figure 9-8 (Part 3 of 4). Operating procedure for DCIP analysis function

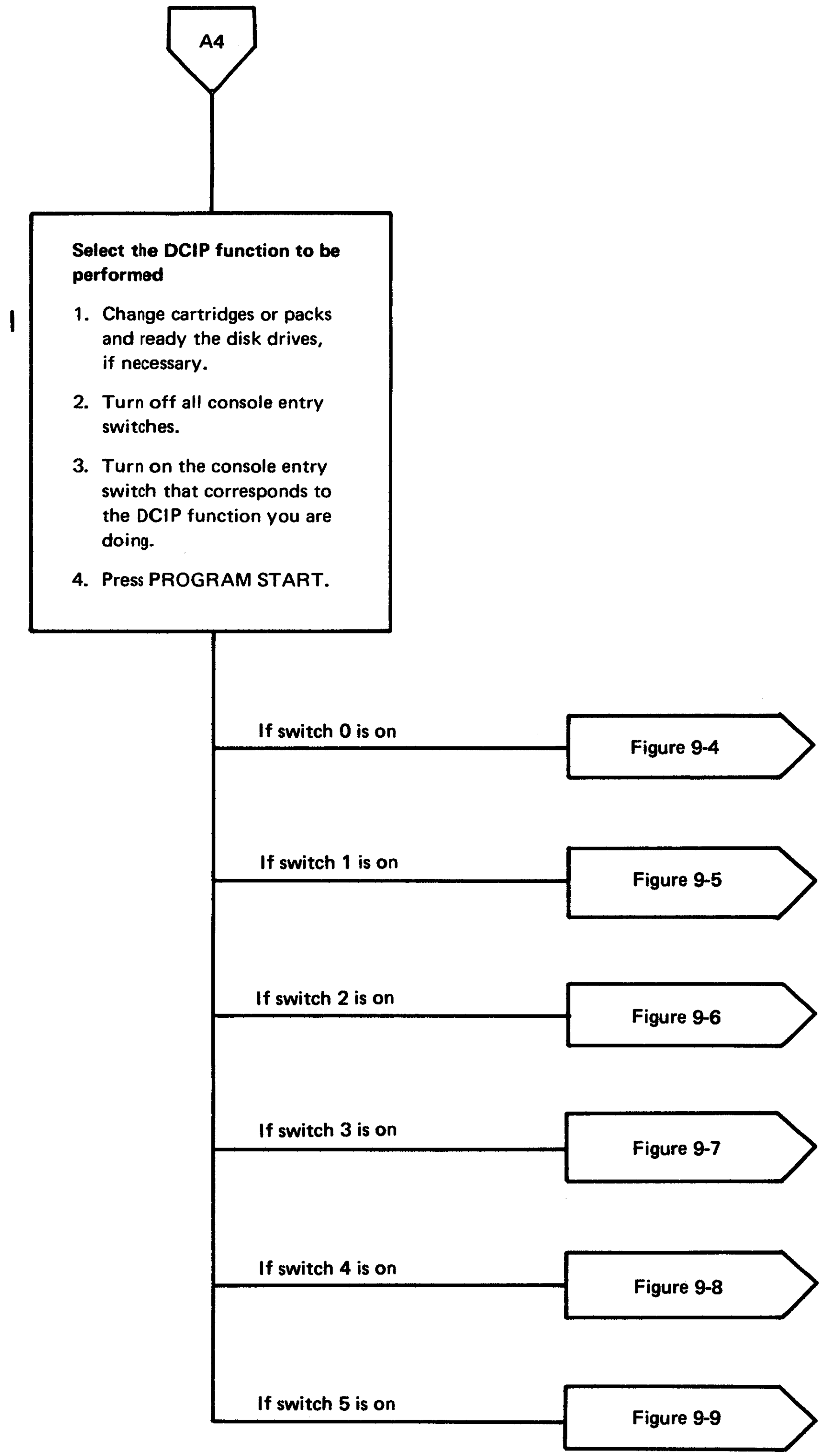


Figure 9-8 (Part 4 of 4). Operating procedure for DCIP analysis function

A1

ENTER:
SOURCE DR (BITS 0-3)
OBJECT DR (BITS 12-15)
is printed on the console printer.

Start the compare function

1. Enter through console entry switches 0 through 3 the physical drive number (in binary) of the drive that contains one of the cartridges being compared (source drive).

2. Enter through console entry switches 12 through 15 the physical drive number (in binary) of the drive that contains the other cartridge that is being compared (object drive).

3. Press PROGRAM START.

The program compares each logical sector of the source cartridge with its counterpart on the object cartridge. If the contents of the 2 sectors do not compare, this message is printed:

CMP ERR ON SCTRS xxxx yyyy

The compare function continues until it is complete.

The DCIP option message is printed.

TURN ON:
SW0 TO INITLZ
SW1 TO COPY
SW2 TO DUMP
SW3 TO PATCH
SW4 TO ANALYZE
SW5 TO CMP

A2

Figure 9-9 (Part 1 of 2). Operating procedure for DCIP compare function

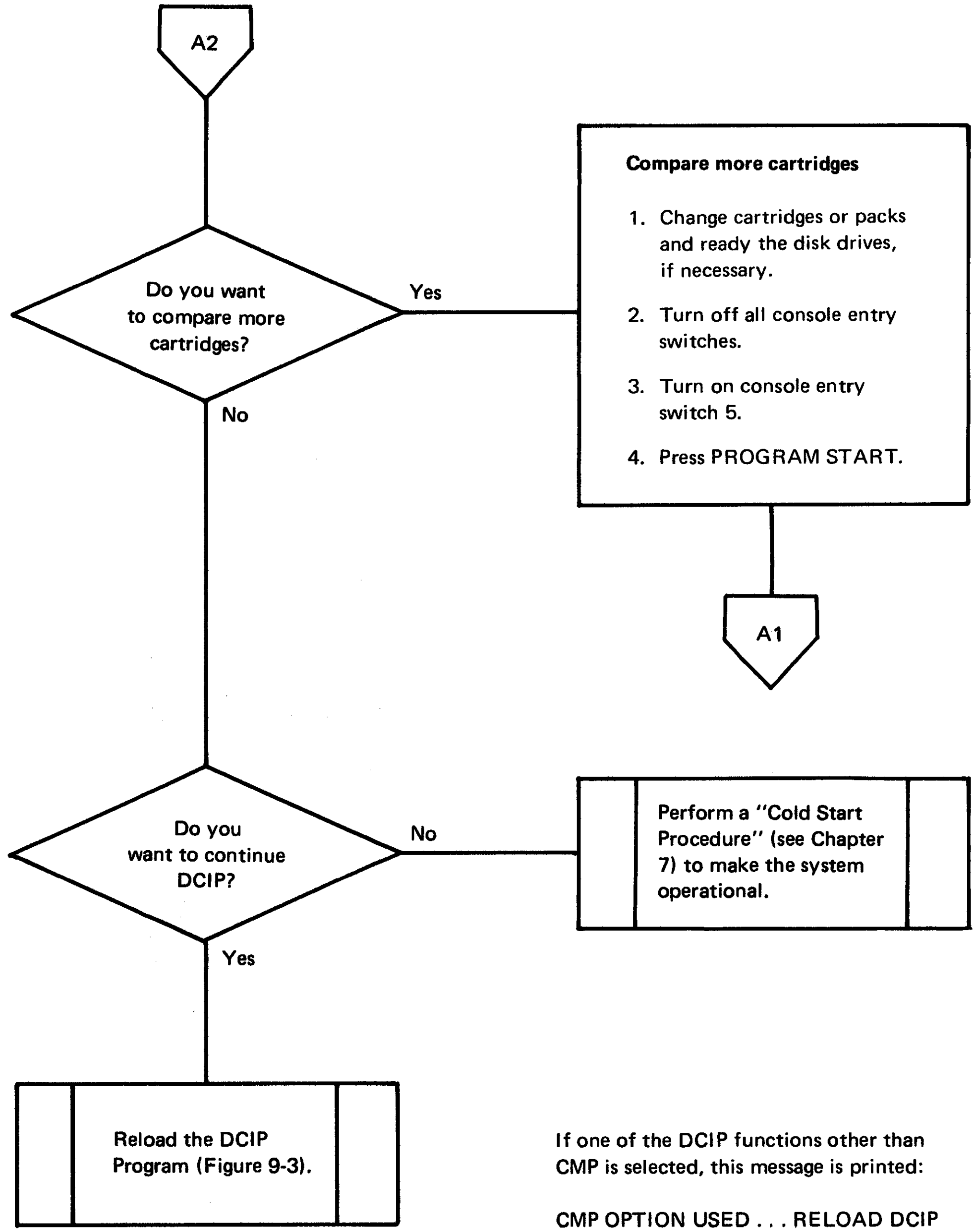


Figure 9-9 (Part 2 of 2). Operating procedure for DCIP compare function

PAPER TAPE REPRODUCING PROGRAM

This program, available only with the paper tape system, copies information from one paper tape onto another. The program reads and punches characters with no intermediate conversion.

The materials that you need to reproduce paper tapes are:

- The Paper Tape Reproducing Program tape, BP18
- The tape being reproduced
- Blank tape

Figure 9-10 is the operating procedure for the stand-alone paper tape reproducing program.

Load the paper tape reproducing program, BP18

1. Insert tape BP18 in the paper tape reader.
2. Position under the read star-wheels one of the delete codes beyond the program ID.
3. Move the console mode switch to RUN.
4. Press IMM STOP on the console.
5. Press RESET on the console.
6. Press PROGRAM LOAD on the console.

   The program is read into core storage, and the system waits with /1111 displayed in the ACCUMULATOR.
7. Remove BP18 from the paper tape reader.

Ready a tape to be reproduced

1. Insert the paper tape that is to be reproduced.
2. Position under the read star-wheels one of the delete codes.

A2

Figure 9-10 (Part 1 of 4). Paper tape reproducing operating procedure

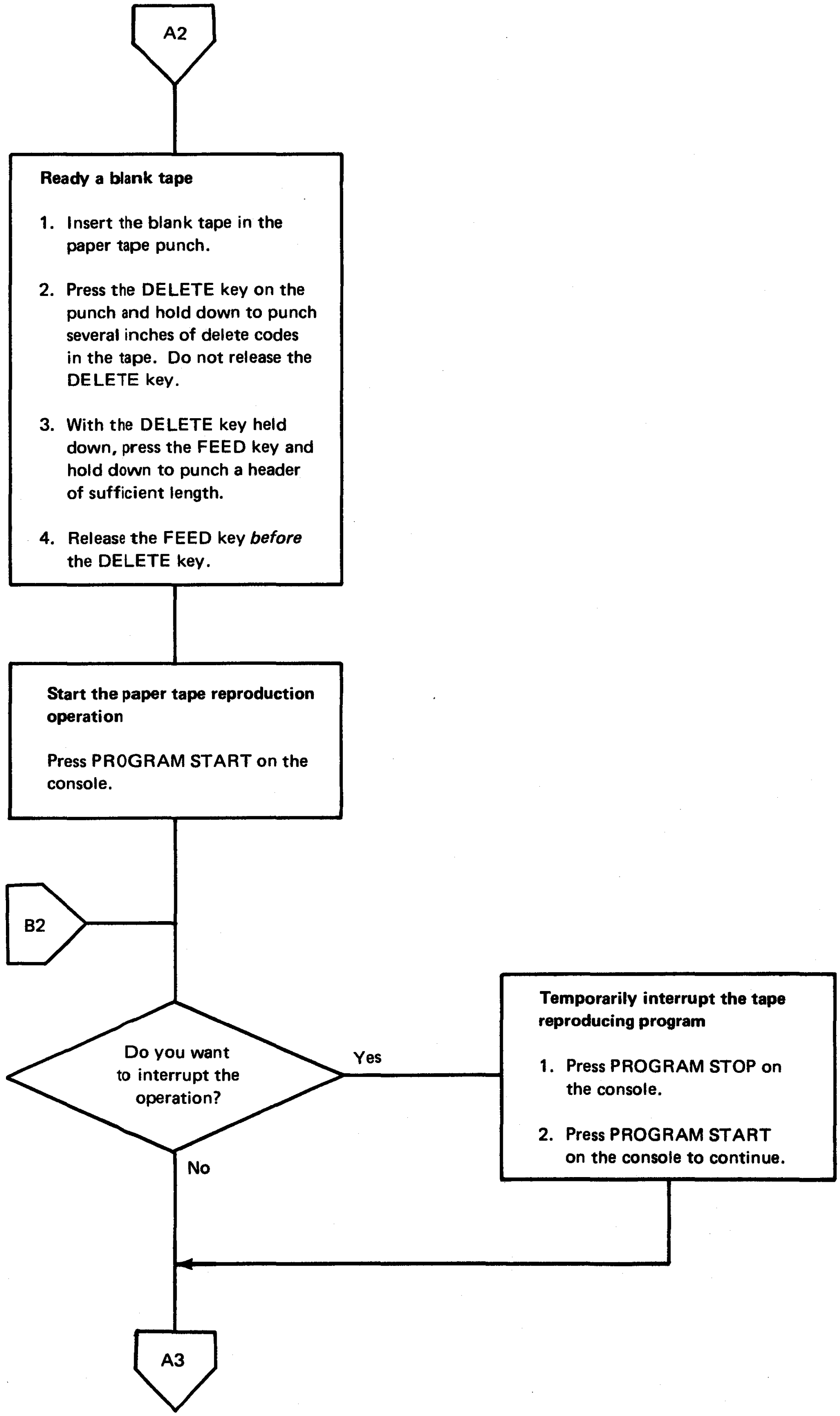


Figure 9-10 (Part 2 of 4). Paper tape reproducing operating procedure

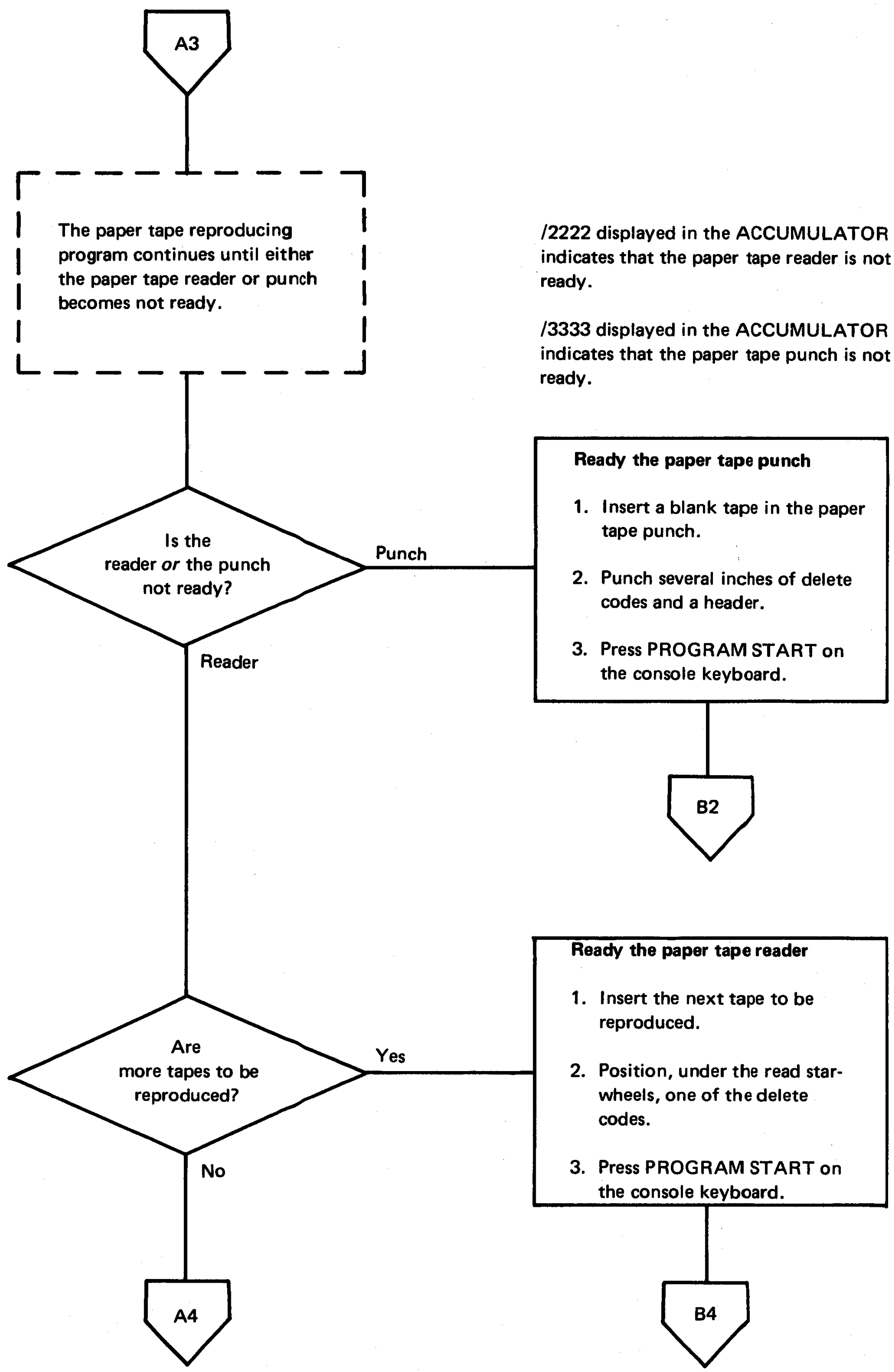


Figure 9-10 (Part 3 of 4). Paper tape reproducing operating procedure

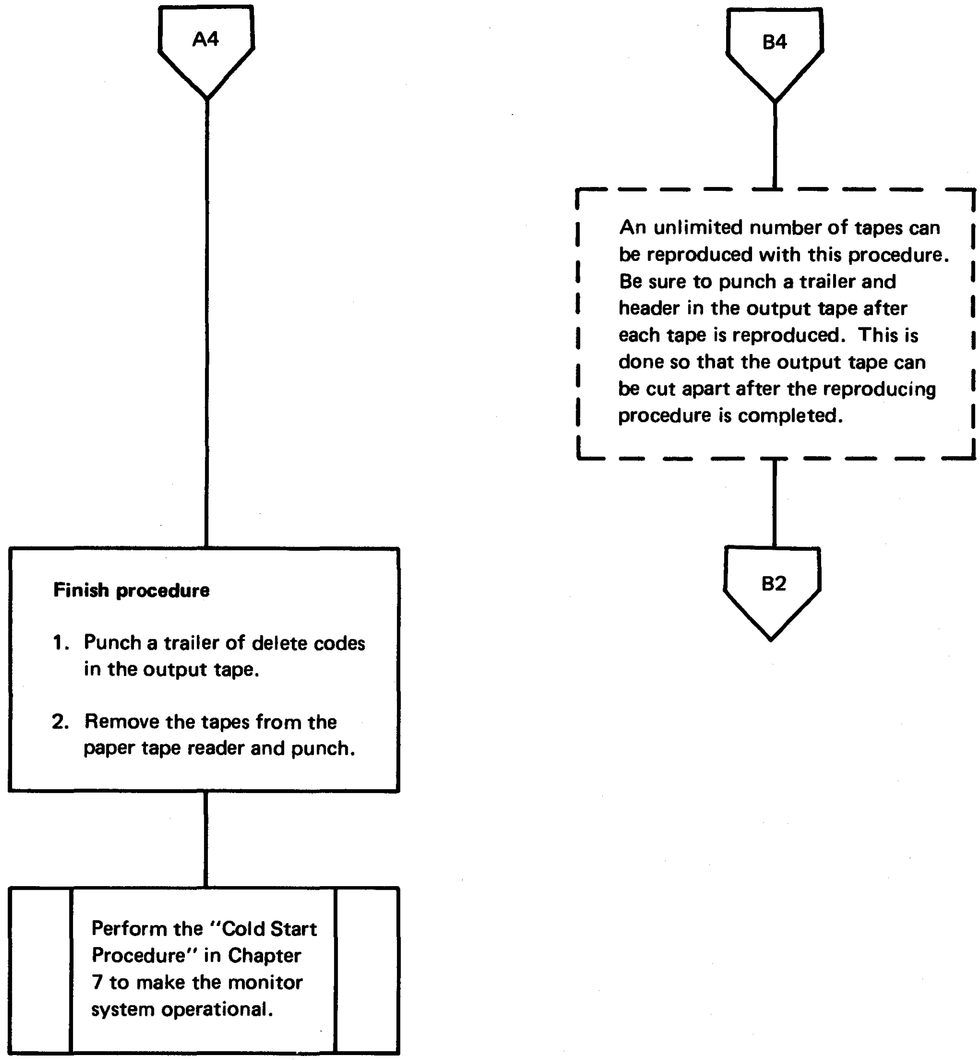


Figure 9-10 (Part 4 of 4). Paper tape reproducing operating procedure

STAND-ALONE PAPER TAPE UTILITY PROGRAM (PTUTL)

This program, available only with the paper tape system allows you to enter records from the the 1134 Paper Tape Reader or the console keyboard. Program output is to the 1055 Paper Tape Punch and/or the console printer. This program is also included as an executable program in the Monitor System Library (see Chapter 4).

The materials that you need to use the PTUTL program are:

- The PTUTL (Paper Tape Utility Program) tape, BP17
- Blank tape if output from the PTUTL program is to be punched into tape
- Previously punched tape if they are being changed

Figure 9-11 is the operating procedure for loading the stand-alone PTUTL program, and Figure 9-12 is the operating procedure for using both the stand-alone PTUTL and the PTUTL mainline program from the system library.

Load the PTUTL Program, BP17	
1. Insert the PTUTL tape, BP17, in the paper tape reader.	
2. Position one of the delete codes beyond the program ID under the read starwheels.	
3. Move the console mode switch to RUN.	
4. Press IMM STOP on the console.	
5. Press RESET on the console.	
6. Press PROGRAM LOAD on the console.	The core image program is read into core storage, and the system waits with /006C displayed in the ACCUMULATOR.
7. Press PROGRAM START to finish the reading of PTUTL.	When the reading of BP17 is complete, the system waits with /00C9 in the ACCUMULATOR.
8. Press PROGRAM START again.	The system waits with /1111 displayed in the ACCUMULATOR.

Figure 9-12

Figure 9-11. Loading the stand-alone PTUTL tape

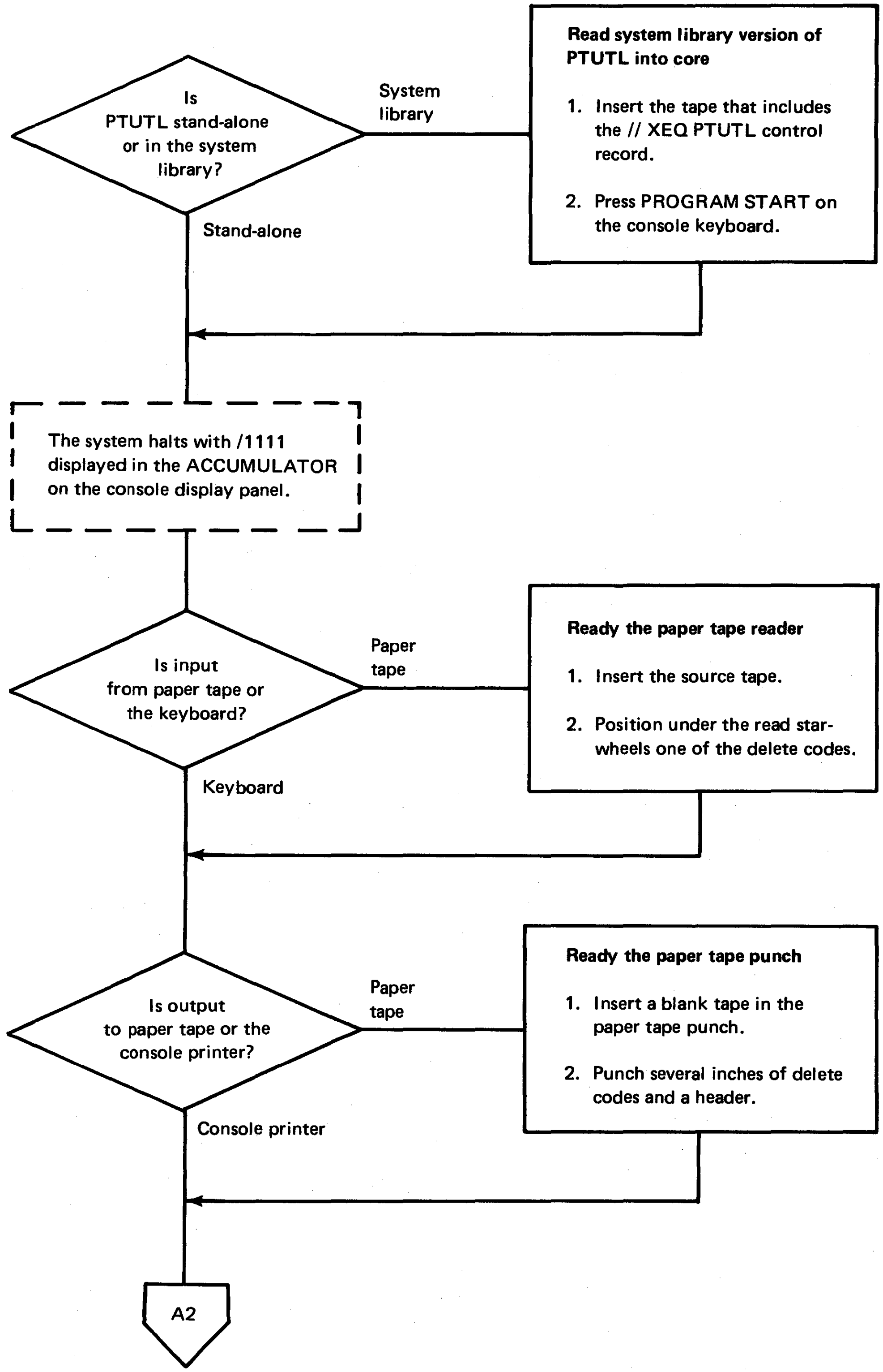


Figure 9-12 (Part 1 of 4). PTUTL operating procedure

A2

Make changes and/or additions

1. Turn on the appropriate console entry switches to perform the PTUTL functions you want.

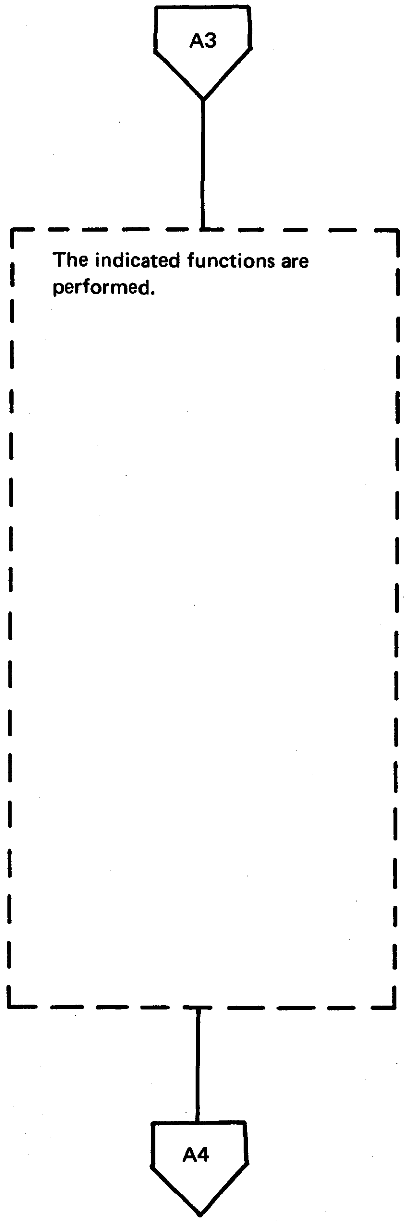
Console entry switch on	PTUTL function performed
0	Print record after reading
1	Read records from the paper tape reader
2	Accept keyboard input
3	Punch records on the paper tape punch
14	Wait after punching with /3333 in the ACCUMULATOR
15	Wait after printing with /2222 in the ACCUMULATOR

All other console entry switches must be off.

2. Press PROGRAM START.

A3

Figure 9-12 (Part 2 of 4). PTUTL operating procedure



If you want to omit a record just read and printed (switches 0, 1, and 15 on) from an output tape, do not change the switches and press PROGRAM START again.

A record just read and printed (switches 0, 1, and 15 on) is replaced by keyboard input if you turn on console entry switch 2 just before pressing PROGRAM START.

The system subroutine TYPE0 is used by PTUTL during keyboard input. These operating features of that subroutine apply:

1. An input record cannot exceed 80 characters.
2. Pressing the backspace key (←) cancels the last character entered.
3. Pressing ERASE FIELD cancels the entire record so you can reenter the record.
4. Pressing EOF indicates that input of a record is complete.

Figure 9-12 (Part 3 of 4). PTUTL operating procedure

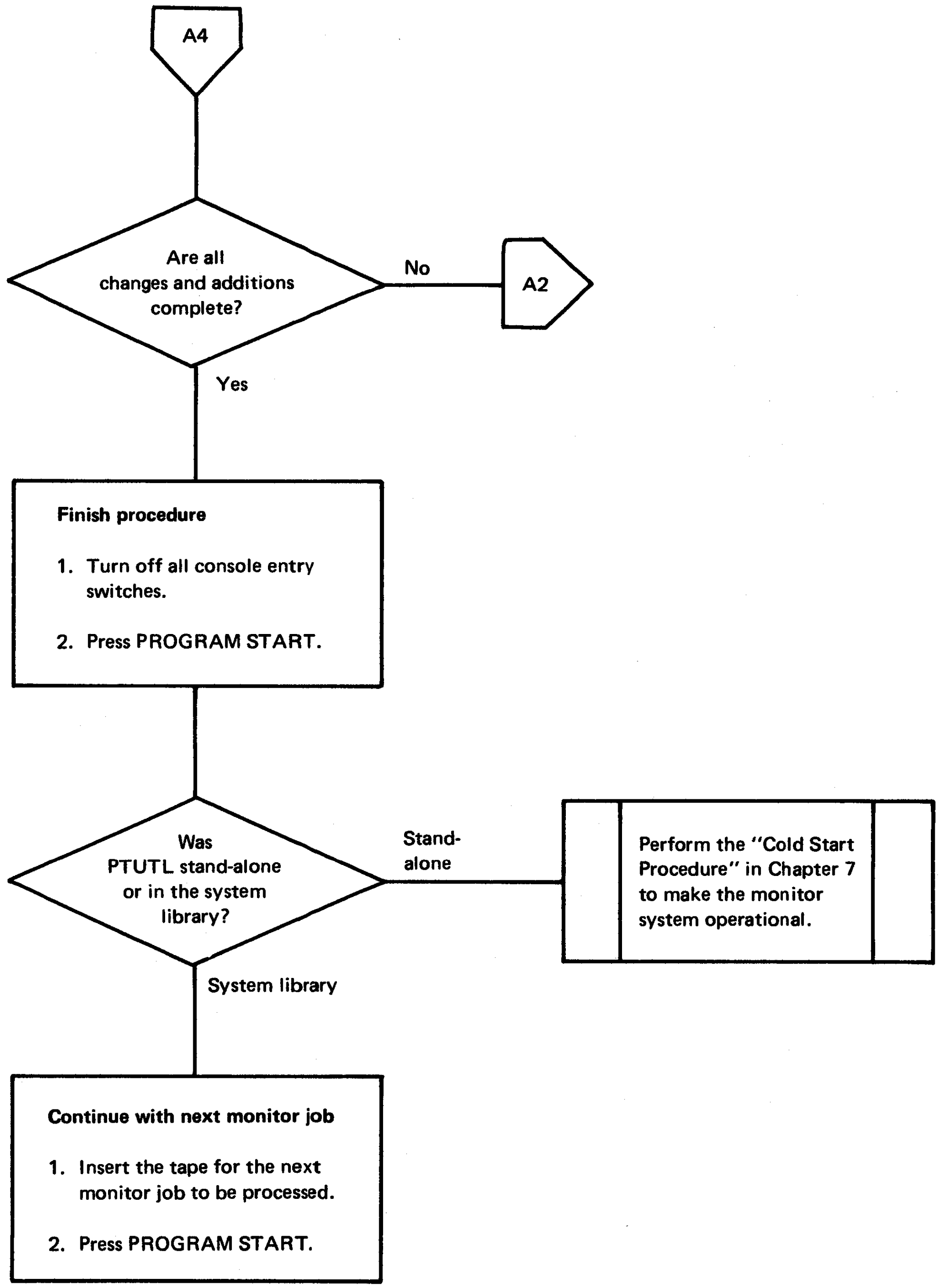


Figure 9-12 (Part 4 of 4). PTUTL operating procedure

PTUTL Example

This example shows you how to change previously punched records. Assume that the following records are punched in a tape:

// JOB
// * (comments record)
// ASM
// DUP
ASM control records
Source program

You have decided to alter the comments record, insert a // PAUSE control record after the comments record, and delete the // DUP control record. The procedure you follow is:

Your action		System response
1.	Load into core storage and start execution of PTUTL.	The system waits with /1111displayed in the ACCUMULATOR on the console display panel.
2.	Insert the source tape and ready the paper tape punch and the console printer. Punch a leader of delete codes in the output tape.	
3.	Turn on console entry switches 1, 3, and 14.	
4.	Press PROGRAM START.	The // JOB control record is read, punched in the output tape, and the system waits with /3333 in the ACCUMULATOR.
5.	In addition to the console entry switches already turned on, turn on 0, 2, and 15.	
6.	Press PROGRAM START.	The comments record is read and printed on the console printer. The system waits with /2222 in the ACCUMULATOR.
7.	Press PROGRAM START again.	The K.B. SELECT indicator on the console keyboard turns on and /3333 is displayed in the ACCUMULATOR.
8.	Enter the new comments record in the proper format.	
9.	Press EOF.	The new comments record is punched in the output tape; the system waits with /2222 in the ACCUMULATOR.
10.	Turn off console entry switch 1.	
11.	Press PROGRAM START.	The K.B. SELECT indicator turns on, and /3333 is displayed in the ACCUMULATOR.
12.	Enter the // PAUS control record.	
13.	Press EOF.	The // PAUS control record is punched in the output tape; the system waits with /2222 in the ACCUMULATOR.
14.	Turn off console entry switches 0, 2, and 15.	

Your action	System response
15. Turn on console entry switch 1. (Switches 3 and 14 should still be on.)	
16. Press PROGRAM START.	The // ASM control record is read and punched in the output tape; the system waits with /3333 in the ACCUMULATOR.
17. Turn off all console entry switches except 1.	
18. Turn on console entry switches 0 and 15.	
19. Press PROGRAM START.	The // DUP record is read and printed on the printer but is not punched in the output tape. The system waits with /2222 in the ACCUMULATOR.
20. Press PROGRAM START again.	The next input record is read into the I/O buffer, overlaying the // DUP control record. (The // DUP control record is deleted.)
21. Turn off console entry switches 0 and 15.	
22. Turn on console entry switch 3. (Switches 1 and 3 should be on.)	
23. Press PROGRAM START.	The remainder of the source tape is read in and reproduced in the output tape, record for record. The paper tape reader not-ready wait (/3005 in the ACCUMULATOR) occurs when all of the source tape has been reproduced.
24. Turn off all console entry switches.	
25. Press PROGRAM START.	A CALL EXIT is executed.

The remote job entry (RJE) feature of the IBM System/360 Operating System allows you to enter jobs into the operating system job stream via communication lines from terminals (work stations) at distant locations. RJE includes a unique job entry control language (JECL) that controls operations of the work station. For a general description of RJE, RJE terminology, and JECL, see the publication *IBM System/360 Operation System Remote Job Entry,* GC30-2006.

This chapter provides information for operators and programmers using an 1130 as a remote work station in an RJE environment, and describes machine and device requirements, input and output at the work station, communication considerations, operating procedures, user-exit subroutine, and generation and loading of the work station program.

Messages printed by the RJE program are included in Appendix A.

MACHINE AND DEVICE REQUIREMENTS

The RJE program for an 1130 work station requires at least an 1131 Central Processing Unit, Model 2B, a card reader, and a line printer (with a 120 character print line). The 1130 computing system must be connected to a 600–2400 bit-per-second line via a synchronous communications adapter in binary mode.

An optional compress-expand feature requires 16K words of core storage if the 1132 Printer is used, or 8K words if the 1403 Printer is used. The compress-expand feature eliminates blanks from data transmitted across the communication line.

An IBM-supplied RJE exit subroutine stores data from your IBM System/360 Operating System job on an 1130 disk. The data thus stored can be processed by other programs that you write. You can write an exit subroutine to replace the one supplied by IBM and direct the output from your System/360 job to any available 1130 I/O device. When you write an exit subroutine, an 1130 system with 16K words of core storage is required. Information about writing an exit subroutine is included under "User-Exit Subroutine" in this chapter.

COMMUNICATION CONSIDERATIONS

The 1130 RJE Work Station Program provides the standard RJE communications interface to the System/360 Operating System (the operating system) RJE communications network by using the SCAT2 and SCAT3 binary synchronous communications subroutines. These subroutines are stored in the monitor system library and provide the following capabilities:

- Point-to-point contention operation on leased lines
- Point-to-point operation on switched lines
- Multipoint operation with the 1130 system as slave station

All data transmissions between the operating system and an 1130 work station are in EBCDIC transparent mode, except headings, which are transmitted in normal mode. The 1130 RJE Work Station Program communicates with the operating system in 3 modes: monitor, receive, and transmit.

monitor mode

The work station program enters monitor mode from either transmit or receive mode. In this mode, the work station waits for output from the communication line or input from the card reader or console keyboard.

receive mode

The work station program enters receive mode when output is available for the work station. In this mode, the work station program reads output from the line until it receives an end-of-data indication from the operating system or until the operator discontinues the output (presses PROGRAM STOP on the console keyboard). The work station program then enters monitor mode.

transmit mode

This mode is entered at work station startup and when input is available at the work station. The work station program writes to the communication line in transmit mode. Transmission to the line continues until a logical end of file (the . . *null* command) or an RJEND command is encountered in the input stream. (RJE work station commands are described in the publication *IBM System/360 Operating System Remote Job Entry,* GC30-2006.)

If monitor mode is entered from transmit mode with a logical end-of-file indication caused by a . . *null* command, transmit mode is not entered again until operator intervention indicates that more input is available.

Communication Considerations for Switched Lines

The operating system disconnects the line if a switched communication line is inactive for a period of approximately 21 seconds. This occurs when:

- A work station output device error is not corrected within the specified time.
- A user-written exit subroutine fails to return control within the specified time (see "User-Exit Subroutine" in this chapter).
- An operator response to an RJE message is not entered within the specified time.

Note. Some RJE messages allow approximately 3 minutes for an operator response. The RJE Work Station Program operator messages are included in Appendix A.

INPUT AT THE WORK STATION

Input to the RJE program is accepted from the card reader, the keyboard, and from one or more disk storage units.

card input

System/360 jobs (with or without JED statements) and job entry control language (JECL) statements are accepted as input from the card reader. The first JECL statement at work station startup *must be* an RJSTART command submitted from the card reader. After that, JECL statements are not sequence checked.

keyboard input

The only valid input from the keyboard is work station commands and responses to RJE operator messages. Input is accepted from the keyboard between jobs being entered from the card reader when the operator indicates that he has input to submit (only in a point-to-point line configuration). The 1130 RJE Work Station Program checks this input only for the JECL identifier (. . . followed by at least one blank).

disk input

A special 1130 RJE control card is used to specify that input is from one or more disk storage units. This control card, . . DATA, is described under "JECL for the 1130 Work Station" in this chapter. A . . DATA control card can be placed in the card input stream or on disk. 1130 work station commands are placed on disk with the STOREDATAE operation of the Disk Utility Program (see "DUP Control Records" in Chapter 5).

The . . DATA control card contains information that allows the RJE program to read input alternately from the card reader and from the disk. Data to be read from disk must be stored there prior to RJE processing by you. This data must be stored in 80-character records in 8-bit packed code (EBCDIC) format (eight records per disk sector) in consecutive sectors. Data can be stored on disk by:

- Using the STOREDATAE function of the Disk Utility Program prior to executing the RJE Work Station Program
- Specifying that output from a job be placed on a disk

After the information on disk has been read to the end of file (see "JECL for the 1130 Work Station" in this chapter for a description of the end-of-file indications), the RJE program resumes reading from the card reader.

Note. Although work station commands can be submitted from disk, only System/360 jobs and input data sets are recommended to be placed on disk in order to simplify work station operation.

changed LOGON affect on input

If you are logged on because of a LOGON command entered from the card reader or disk, and you enter a new LOGON command from the keyboard, all pending input meant for the previous LOGON from the card reader and/or disk is submitted under the new LOGON ID entered from the keyboard. To prevent this, the LOGON that was entered from the card reader or disk must be resubmitted as the last command entered from the keyboard before card or disk input is continued.

Generation of the 1130 RJE Work Station Program

The 1130 RJE Work Station Program is supervised by the 1130 Disk Monitor System Version 2. You store the IBM-supplied RJE program in the user area by using the *STORE function of the Disk Utility Program (DUP). You then define your work station configuration by executing a program that is part of the RJE program and that is named RJE00. This program reads a data card that you code with the following optional parameters:

$$\begin{bmatrix}\text{LINE=P}\\ \text{LINE=S}\\ \text{LINE=M (x,y)}\end{bmatrix}\begin{bmatrix}\text{,UEXIT=(address 1, address 2)}\\ \text{,UEXIT=USER}\end{bmatrix}\begin{bmatrix}\text{,COMPRESS=NO}\\ \text{,COMPRESS=YES}\end{bmatrix}$$

LINE=P specifies that the work station is connected over a point-to-point leased line.

LINE=S specifies that the work station is connected over a point-to-point switched line.

LINE=M (x,y) specifies that the work station is connected over a multipoint line, where
x is the polling character
y is the selection character.

UEXIT=(address 1, address 2) specifies the starting and ending addresses of the area on disk that has been reserved for storing data directed to the user exit, where
address 1 is the starting address
address 2 is the ending address.

The addresses must be in the form *xaaa*, where
x is the logical disk drive number from 0 to 4
aaa is the sector address.

This area must be reserved prior to executing the RJE Work Station Program.

UEXIT=USER specifies that the IBM-supplied user-exit subroutine is replaced by one that you have written.

COMPRESS=NO specifies that blanks are not to be eliminated from data transmitted across the communication line.

COMPRESS=YES specifies that blanks are to be eliminated from data transmitted across the communication line.

These optional parameters can be used in any order, and if more than one of them is specified, they must be separated by commas. The default options assumed when the RJE Work Station Program is first generated, are a leased point-to-point contention line, no reserved disk space for user-exit output, and no elimination of blanks. When this data card is used to redefine the RJE configuration and the LINE and/or COMPRESS parameters are omitted, the program assumes the last parameters specified as the current line configuration; however, if the UEXIT parameter is omitted, space is not reserved on disk for user-exit data.

The RJE00 program saves the information specified by these parameters in a disk data file reserved for common constants used by the RJE program.

work station RJE generation

The following example shows the coding for generating the 1130 Work Station Program:

```
1   5    10   15   20   25   30   35   40   45   50
// JOB
// XEQ RJEØØ
LINE=M(A,B),UEXIT=(21BØ,22BØ),COMPRESS=YES
```

The first 2 cards are the monitor control records needed to load the program that processes the information in the third card. The third card specifies that the RJE work station is on a multipoint line, that its polling character is A, and its selection character is B, and that it will compress input to the operating system program and expand output from the operating system program. For storing data that is directed to the user exit, an area is reserved on disk drive 2 starting at sector 1B0 and ending with sector 2B0.

JECL FOR THE 1130 WORK STATION

The job entry control language (JECL) used with the 1130 work station is described under "Job Entry Control Language" in the publication *IBM System/360 Operating System Remote Job Entry,* GC30-2006, with one addition. The additional command allows you to alternate the source of input between disk and cards. The format of this command is:

ID	Operation	Operand
..	DATA	DMS $\left\{\begin{array}{l}\text{, C} \\ \text{, D, xaaa [, bbbb]}\end{array}\right\}$

..	is the JECL identifier and must be in columns one and two.
DATA	must be preceded and followed by at least one blank.
DMS	identifies the card as an 1130 JECL command.
C	indicates that input follows from cards.
D	indicates that input follows from disk, where *x* is the logical disk drive number, *aaa* is the disk sector address (hexadecimal), and *bbbb* is a hexadecimal number specifying the length of the disk data file in blocks, two blocks per 80-character record (16 blocks per sector).

If *D* is specified, the logical disk drive number and the sector address are required, but the block count is optional. When the block count is not specified, you must indicate the end of data on disk by using a . . DATA command to transfer reading data either to the card reader or to another disk area. The optional block count for disk data causes the RJE program to read data from disk until the specified number of blocks has been read, unless an end-of-file indicator (. . DATA command, . . *null* command, or . . RJEND command) is read first. When the specified number of disk blocks is read or an end-of-file indicator is read, reading from disk stops, and input continues from the card reader.

Data on disk must start at the beginning of a sector and continue on to consecutive sectors if necessary. Each sector must contain eight 80-character records in 8-bit code (EBCDIC), except the last sector, which can be less than 320 words.

The . . DATA command is not recognized between a // DD DATA statement and the corresponding /* in an IBM System/360 Operating System job.

Note 1. Restart problems may occur if jobs are chained on disk (that is, referenced by only one . . DATA command from the card reader), and a line error occurs that requires the work station to resubmit the RJSTART command and all unacknowledged input. To avoid these problems, reference each job with a . . DATA command from the card reader.

Note 2. You must specify the cartridges that are used during RJE on a monitor JOB control record. A logical drive number as specified on the JOB control record must be used in the . . DATA command.

End-of-File Indicators

The end-of-file indicator on disk is the . . DATA command. This command passes reading to another disk file or to the card reader. The end-of-file indicators for the card reader are the . . *null* command and the . . RJEND command.

Note. The . . *null* command and the . . RJEND command can be read from disk and have the same effect as if they were read from the card reader; that is, reading is stopped both from the card reader and from the disk.

OUTPUT TO THE WORK STATION

Output to the work station consists of job output and messages. Job output, consisting of SYSOUT data sets created by the job, is directed to the printer, the card punch, or a user-exit subroutine. Each job output data set is directed to the device associated with the SYSOUT class specified in the DD statement for that output data set. RJE system messages are directed to the console printer or the line printer.

You can specify carriage control for printer output with a special control character as the first byte of each data record; either System/360 machine code or ASA control characters are allowed. Output is single spaced with a skip to channel one when channel 12 is sensed in the carriage tape and control characters are not specified or are not recognized by the equipment.

You can specify stacker-select for punched output, if available, by specifying a special control character as the first byte of each data record; either System/360 machine code or ASA control characters are allowed. Stacker one is selected if control characters are not specified or are not recognized by the equipment.

The 1130 RJE Work Station Program includes a user-exit subroutine that accepts data sets directed to it and writes them on disk in an area that you reserve prior to executing the RJE program.

The IBM-supplied user-exit subroutine can be replaced by an exit subroutine that you write. Your subroutine can process data directed to the user-exit and write output to any available device (see "User-Exit Subroutine" in this chapter for more detailed information).

If you do not write a user-exit subroutine, the IBM RJE program user-exit subroutine writes data sets consecutively on disk, each data set beginning at a disk sector boundary. However, when the RJE program is reloaded at a later time, data sets previously written on disk are unprotected and may be destroyed since any user-exit data sets written after RJE is reloaded begin at the first sector of the reserved area. For each data set written, information is printed on the principal printer.

The primary output device for messages is the console printer. The secondary device is the line printer. You select the line printer as the message device by turning on console entry switch 0.

Note. Data directed to disk can be referenced later by a . . DATA command. To do this, you must define your data set as fixed blocked or unblocked with a logical record length of 80 bytes and no control characters.

Discontinuing and Continuing Output

Job output is discontinued by operator intervention. The operator presses the console keyboard PROGRAM STOP key, then the PROGRAM START key, and the system prints the J90 OCR=message. The operator then responds by typing D to discontinue output.

Output is also discontinued by the 1130 RJE Work Station Program when a user-exit subroutine is not present for output directed to the user-exit and one of the following errors occurs:

- An area is not reserved for user-exit output.
- The reserved output area is exhausted.
- An unrecoverable disk write error occurs.

These errors are indicated to the operator in error messages. To correct the first 2 problems, terminate the RJE program by submitting an RJEND command (after all pending input has been transmitted), and then specify a reserved area on disk by executing the RJE00 program (see "Generation of the 1130 RJE Work Station Program" in this chapter). Reload the RJE program (see "Work Station Startup" in this chapter), and discontinue output immediately by operator intervention. Then, enter a CONTINUE command with the BEGIN operand; otherwise, data is lost.

To correct the third error, enter a CONTINUE command with the BEGIN operand. The data set is then written again, starting at a new sector.

In general, once output is discontinued, no other output is transmitted to the work station until the disposition of the discontinued output is specified by a CONTINUE command.

Other conditions that cause output to be discontinued are:

- A change in form number is found at the operating system
- The work station program requests discontinuation
- An irrecoverable error occurs during an output operation

If either of the first 2 conditions occurs, you specify the disposition of the output with the CONTINUE command. The third condition requires error recovery procedures.

User-Exit Subroutine

The operating system RJE program passes physical records to the user-exit subroutine, either the one that is supplied with the RJE program or the one that you write to replace it. This section describes the programming requirements that must be included in your subroutine.

The subroutine entry point must be named UEXIT, and the subroutine must be stored in the user area (after deleting the resident module with the same name). You should save and restore the contents of registers 1 and 3 at the beginning and end of your subroutine. To specify that your subroutine be executed, use the UEXIT=USER parameter in the configuration data card used to generate the RJE program.

The user-exit subroutine gains control when output becomes available for it. Upon entry, the return address is stored in the first word of the subroutine, and index register 1 contains the address of a parameter list that describes the output being passed to the subroutine. This parameter list with the following format is aligned on an even word boundary.

+0	Starting address
+1	Ending address
+2	Logical record length
+3	Control character type
+4	Record format
+5	End of data

Data characters are packed 2 characters per 1130 word. The blocks start on a word boundary, but they end in the middle of a word if they contain an odd number of characters.

starting address — The starting address is the 1130 core storage address of the block of data being received from the operating system. This address has the following format: the 15 leftmost bits are the core storage address, and the rightmost bit indicates whether the data starts in the first 8 bits or the second 8 bits of the first word at that location. Zero indicates that data begins in bit zero at the starting address; one indicates that data begins in bit 7 at the starting address.

ending address — This is the ending address plus one of the data block being received from the operating system. The format of the ending address is the same as the starting address.

logical record length — When fixed length records are being passed, this word contains the length of logical records. If variable or undefined records are being passed, this word is zero.

control character type — This is a code that indicates the type of control characters being used.

0—No control characters
1—IBM System/360 machine code
2—ASA code

record format This word contains a code that indicates the type of data records being transmitted.

1–Fixed unblocked
2–Fixed blocked
3–Variable unblocked
4–Variable blocked
5–Undefined

end of data When this word is zero, the end of data is indicated.

The user-exit subroutine that you write must use the same I/O subroutines that the 1130 RJE program uses.

Device	*I/O Subroutine*
1132 Printer	PRNT2
1403 Printer	PRNT3
1442, Model 6 or 7, Card Read/Punch	CARD1
2501 Card Reader	READ1
1442, Model 5, Card Punch	PNCH1
Console Keyboard	TYPE0
Disk	DISKZ

Note. Your user-exit subroutine must return control to the RJE program within approximately 21 seconds in order to maintain communication with the operating system.

OPERATING PROCEDURES

This section includes information about beginning and ending RJE jobs, as well as information about console keyboard operation during execution of the RJE program.

Work Station Startup

To start RJE operation, the 1130 RJE Work Station Program must be loaded into core storage. This program is loaded by specifying the program name RJE in a monitor XEQ control record. The work station program then loads into core the programs and subroutines from the system library that correspond to the configuration of your system. To load these programs and subroutines, the work station program uses information stored on disk by the RJE generation program and information in the disk monitor system that specifies the principal I/O devices.

Note. The console printer cannot be the principal print device.

The following example shows the coding to start and end the execution of the RJE program:

```
1  5    10   15   20   25   30   35   40   45   50
// JOB
// XEQ RJE
.. RJSTART

   JECL statements and operating system job

.. RJEND
```

The RJSTART command must be the first RJE command entered. An error message is printed when the RJSTART command is not the first entered. To continue, place an RJSTART command in the card reader, and press START on the card reader and PROGRAM START on the console keyboard. If the work station is connected to the operating system over a switched line, a message to call the central system is printed.

The RJSTART command is followed either by input to be sent to the operating system or by an end-of-file indicator (see the following section "The Null Command"). When contact is made with the operating system, the RJSTART command and all other commands, if any, before the first job entry (the System/360 job with or without the JED card) or before the end-of-file indicator, are transmitted.

The work station is logically attached to the RJE system when the RJSTART command is acknowledged. All pending messages and immediate job output is received at the work station. All pending input, if any, is transmitted, or the work station program waits for output from the operating system. The sequence of events is system dependent.

The Null Command

The *null* command is provided for the 1130 work station to indicate the end of file on the card reader. This command is coded with the identifying characters (. .) in columns 1 and 2. All other columns remain blank. The null command must be the last card in the input stream. When this command is read, the card reader is effectively closed even though communication is maintained with the operating system.

Operator intervention is required to resume input from the card reader after the null command has been read (see the following section "Console Keyboard Procedures" in this chapter).

Console Keyboard Procedures

Four RJE functions that you can start from the 1130 console keyboard are:

- Indicating card reader input
- Indicating keyboard input
- Discontinuing output
- Initiating an abnormal closedown of the RJE program

You start any of these by:

1. Pressing PROGRAM STOP on the console keyboard
2. Pressing PROGRAM START

The message J90 OCR= is printed on the console printer. Your response to this message indicates the function to be performed. The replies to this message are listed with other RJE messages in Appendix A.

If you type B when message J90 is printed, keyboard input is indicated. The system prints the message J93 PROCEED and the K.B. SELECT light on the console turns on when the RJE program can service keyboard input. You can then enter commands, each ended by pressing EOF. After entering the last command, press EOF an extra time to indicate the end of keyboard input; the last EOF must not be entered until the keyboard select (K.B. SELECT) light turns on.

You indicate abnormal closedown of the RJE program by typing T in response to the J90 message. This reply causes the work station program to be terminated and the contents of core storage to be printed.

The operating system notes an error condition and logically detaches and disconnects the work station if it is connected over a switched line. The work station is logically detached if connected with the central system over a leased or multipoint line and a line operation is in progress when you request termination through the keyboard. Also, if the RJE program is not reloaded, the work station is logically detached if the central system tries to contact the work station while the communication line is idle.

Note 1. If the console keyboard procedure is used when the console printer is already in use, the message is not printed. However, the PROGRAM START key must be pressed to continue processing.

Note 2. The INT REQ key *cannot* be used when the RJE program is being used. Pressing INT REQ prevents information in the skeleton supervisor that is modified by the RJE program from being restored. As a result, the disk monitor system may function improperly.

Error Recovery Procedures

Facilities are provided to recover from both communication errors and local device errors at the 1130 work station. Operator intervention may be necessary to correct the condition causing the error. Error messages are printed when errors occur, except for a forms check error on the console printer. In the latter case, when the FORMS CHECK light on the console keyboard turns on, you must turn on console entry switch 1 to retry the operation. Communications on the line are maintained only if the error is corrected within approximately 21 seconds. If errors cannot be corrected within the time allowed, the operating system logically detaches the work station from the RJE system. In addition, if the work station is connected over a switched line, the operating system breaks the connection.

RJE messages and error messages are described in Appendix A.

Unrecoverable communication errors result when communication is lost with the operating system because of either line errors or a failure at the central system. In either case, the work station is logically detached by the operating system and restart procedures are necessary. The response received when restart procedures are executed indicates whether the error is due to a line error or a failure at the central system.

Restart Procedures

Restart procedures must be used when the message J51 LINE ERROR OCR= is printed. These procedures involve regaining communication with the operating system and submitting an RJSTART command and are indicated when you type A in response to the J51 message. A complete description of this message is included in Appendix A.

The restart procedures cause output to automatically resume either where it was interrupted (after a line error) or at the beginning of the job (after a failure at the central system). If output is being written to disk at the time of a line error you should immediately discontinue the output and enter a CONTINUE command with the BEGIN operand.

If output is being punched in cards or printed at the time of a line error, a duplication of the last transmission block may occur when the program is restarted. The printer skips to a new page when RJE is restarted if the data set being printed is without control characters.

If a line error occurs during an input operation, all unacknowledged input must be resubmitted. Furthermore, a line error in the middle of a job implies that the whole job must be resubmitted from the beginning. Before the job can be transmitted again with the same job name, the old job that was partially sent to the central system must be deleted. Deletion is sometimes automatic, but if not, you must delete the job.

Note. The work station restart procedure after a central system failure is similar to the restart procedure after an unrecoverable line error. The primary difference is that after a system failure, an inprocess data set is rewritten from the *beginning* rather than from the last valid block.

Messages Sent to Work Stations

Detailed descriptions of all messages sent to an 1130 work station from the operating system RJE program are in "Messages Sent to Work Stations" in the publication *IBM System/360 Operating System Remote Job Entry*, GC30-2006.

RJE Program Console Entry Switches

Three console entry switches are used by the RJE Work Station Program

Console Entry Switch	*Console Entry Switch Function*
0 (off)	Indicates that RJE messages from the central system are printed on the console printer
0 (on)	Indicates that RJE messages from the central system are printed on the line printer
1	If on when the console printer becomes not ready, the operation is retried.
2	If on, the error statistics accumulated by the subroutines SCAT2 or SCAT3 are printed on the console printer at the end of the RJE run.

Error Statistics

Error statistics are accumulated during an RJE run by the subroutines SCAT2 and SCAT3. If you want these error statistics printed, turn on console entry switch 2 prior to the end of the RJE run.

The error statistics accumulated during the last RJE run can be printed if you execute a program called RJSTA that is a part of the RJE program package.

Appendix A. Monitor System Operational and Error Messages

This appendix includes all monitor system operational and error messages and codes, except for the messages for the stand-alone utility programs. The messages for these programs are included in Chapter 9 with the descriptions of the programs.

The messages in the appendix are ordered alphabetically by an error prefix letter. Unless otherwise noted, the messages are printed on the principal printer. All monitor system control records are also printed on the principal printer.

The messages, in sequential order, are:

Error code prefix	Figure number	Figure title including program name
—	A-1	Assembler error detection codes
A	A-2	Assembler error messages
C	A-3	FORTRAN error codes
C	A-4	FORTRAN error messages
D	A-5	DUP/MUP error messages
E	A-6	System loader error messages
G	A-7	SGJP error messages
J	A-8	RJE work station error messages
J	A-9	RJE work station messages
M	A-10	Phase 1. System control record program error messages
M	A-11	Phase 2. System control record program error messages
—	A-12	SYSUP - DCOM update error messages
Note	A-13	RPG compiler error notes
R	A-14	Core load builder error messages
S	A-15	Auxiliary supervisor error messages
—	—	Monitor system mainline programs messages

ASSEMBLER ERROR CODES AND MESSAGES

At the completion of an assembly, the following messages are printed on the principal printer:

XXX OVERFLOW SECTORS SPECIFIED
XXX OVERFLOW SECTORS REQUIRED
XXX SYMBOLS DEFINED
XXX ERROR(S) AND XXX WARNING(S) FLAGGED IN ABOVE ASSEMBLY

If LIST DECK or LIST DECK E control records are used, the error detection codes listed in Figure A-1 are punched in columns 18 and 19. These error detection codes are also printed if the program is listed. Figure A-1 includes the error flag (code), your coding violation that caused the error, and the assembler action.

For the first error detected in each statement, the assembler stores and then punches (or prints) the appropriate code; the code for a second error is stored, overlaid by any subsequent errors, and the code for the last error detected is punched (or printed). Thus, if more than 2 errors are detected in the same statement, only the first and last are indicated in columns 18 and 19 when LIST DECK or LIST DECK E is used, or are printed when the program is listed.

At the end of an assembly, a message is printed indicating the number of assembly errors detected in the source program (see the last of the assembly messages previously listed). Since no more than 2 errors are flagged per statement, the error count in the message may exceed the actual number of error flags.

Assembler error messages are listed in Figure A-2. These messages include the message number and message, the cause of the error, and the action you must take to correct the error.

Flag	Coding error	Assembler action
A	Address error	
	An attempt has been made to specify a displacement field, directly or indirectly, outside the range of -128 to +127.	The displacement is set to zero.
C	Condition code error	
	A character other than +, -, Z, E, C, or O is detected in the first operand of a short branch statement or the second operand of a long BSC, BOSC, or BSI statement.	The displacement is set to zero.
F	Format code error	
	A character other than L, I, X, or blank is detected in column 32; L or I format is specified for a statement that is valid only in short form, or I format is specified when not allowed.	The statement is processed as if L format were specified, unless the statement is valid only in short form. The statement is then processed as if X format were specified.
L	Label error	
	An invalid symbol is detected in the label field.	The label is ignored.
M	Multiply defined label error	
	A duplicate symbol is encountered in the label field.	The first occurrence of a symbol in the label field is used to define its value; subsequent occurrences of the symbol in the label field cause a multiply defined indicator to be inserted in the symbol table entry (bit 0 of the first word).
O	Operation code error	
	An operation code is not valid.	The statement is ignored and the address counter is incremented by 2.
	An ISS, ILS, ENT, LIBR, SPR, EPR, or ABS is incorrectly placed.	The statement is ignored.
Q	Warning flag	A possible problem code is detected; that is, a modify memory statement with a displacement of zero.
R	Relocation error	
	An expression does not have a valid relocation.	The expression is set to zero.
	An absolute displacement is not specified.	The displacement is set to zero.
	An absolute origin is specified in a relocatable program.	The specified origin is ignored.
	An absolute operand is not specified in a BSS or BES statement.	The operand is assumed to be zero.
	A relocatable operand is not in an END statement of a relocatable mainline program.	Columns 9 through 12 are left blank; the entry is assumed to be relative zero.
	The operand of an ENT statement is not relocatable.	The statement is ignored.

Figure A-1 (Part 1 of 2). Assembler error detection codes

Flag	Coding error	Assembler action
S	Syntax error	
	An invalid expression (that is, an invalid symbol, adjacent operators, invalid constant) is used.	The expression is set to zero.
	An invalid character is used in a record.	If an invalid character is used in an expression, label, operation code, format, or tag field, additional errors may occur.
	The main program entry point is not specified as the operand in an END statement.	Columns 9 through 12 are left blank; the entry is assumed to be relative zero.
	The syntax of an EBC statement is incorrect (that is, a delimiter is not in column 35, a zero character count).	Columns 9 through 12 are left blank; the address counter is incremented by 17.
	An invalid label is used as an operand in an ENT or ISS statement.	The statement is ignored.
	An operand label occurs in more than one ENT statement.	All entries are built as usual.
T	Tag error	
	Column 33 contains a character other than blank, 0, 1, 2, or 3 instruction statement.	A tag of zero is assumed.
U	Undefined symbol	
	A symbol used in an expression is not defined.	The value of the expression is set to absolute zero.
W	An x- or y-coordinate, or both, is not within the specified range; or an operand is invalid.	The operand is set to zero.
X	A character other than R or I is in column 32; or a character other than D or N is in column 33.	The field is set to zero.
Z	An invalid condition is in a conditional branch or interrupt order.	The condition bits in the first word are set to zero.

Figure A-1 (Part 2 of 2). Assembler error detection codes

Error number and message	Cause of error	Your response
A01 MINIMUM W.S. NOT AVAILABLE . . . ASSEMBLY TERMINATED	The available working storage is less than the specified number of overflow sectors plus one.	Do one of the following: 1. Reduce the specified number of overflow sectors (the number specified is zero if an *OVERFLOW SECTORS control record is not used). 2. If your system has more than one disk drive, use a monitor JOB control record to specify system working storage on the cartridge that has the most working storage available.
A02 SYMBOL TABLE OVERFLOW . . . ASSEMBLY TERMINATED	The number of sectors of symbol table overflow is greater than the number of overflow sectors available.	Use an *OVERFLOW SECTORS control record to increase the number of overflow sectors for this assembly (maximum 32 sectors).
A03 DISK OUTPUT EXCEEDS W.S.	Intermediate output (pass 1) or final DSF output (pass 2) exceeds the capacity of working storage less the specified number of overflow sectors.	If this error occurs during pass 1, restart the assembly using an *TWO PASS MODE control record. If this error occurs during pass 2, see the corrective actions for message A01.
A04 SAVE SYMBOL TABLE INHIBITED	One of the following occurs when an *SAVE SYMBOL TABLE control record is used:	
	1. The program is relocatable.	Add an ABS statement to your program and reassemble.
	2. The program contains assembly errors.	Correct the program errors and reassemble.
	3. The source program contains more than 100 symbols.	Reduce the number of symbols and reassemble.
A05 XXX ERRONEOUS ORG, BSS, OR EQU STATEMENTS IN ABOVE ASSEMBLY	XXX is the number of ORG, BSS, BES, and/or EQU statements undefined in the first pass. At the end of pass 1, these statements are printed on the principal printer.	
	If the error is due to forward referencing, the error is not detected during pass 2.	When forward references are attempted, correct them and reassemble the program.
A06 LOAD BLANK CARDS	A card containing a punched column between 1 through 71 is read while a symbol table is being punched (*PUNCH SYMBOL TABLE specified for this assembly).	The system waits with /100F displayed in the console ACCUMULATOR. 1. Press NPRO on the card reader. 2. Place blank cards in front of the card just read 3. Press reader START. 4. Press console PROGRAM START. *Note:* If output is being punched on a 1442, Model 5, a punched card cannot be detected. In addition, the card punch may be damaged if an attempt is made to punch a hole where a hole already exists.

Figure A-2 (Part 1 of 2). Assembler error messages

Error number and message		Cause of error	Your response
A07	ABOVE CONTROL STATEMENT INVALID	The control record option does not agree, character for character, with its valid format. An invalid library name is detected on an *MACLIB control record, or multiple *MACLIB control records are detected.	The control record is ignored.
A08	MACLIB UNDEFINED	An attempt is made to define a stored macro when a macro library is not associated with this assembly.	Reassemble specifying a valid macro library.
A09	PARAMETER LIST OVERFLOW . . . ASSEMBLY TERMINATED	The disk parameter-list spill area is undefined or exceeded.	Reassemble specifying a larger parameter-list disk area (see "*OVERFLOW SECTORS" in Chapter 5).
A10	MACRO AREA OVERFLOW . . . ASSEMBLY TERMINATED	The disk area for macro definitions is undefined or exceeded.	Reassemble specifying a larger macro-definition disk area (see "*OVERFLOW SECTORS" in Chapter 5).
A12	NEST LEVEL EXCEEDS 20 . . . ASSEMBLY TERMINATED	An attempt is made to nest more than 20 macro calls.	Redefine the macro nest and reassemble.
A21	*LEVEL CONTROL STATEMENT MISSING	A program is assembled as an ISS subroutine without the required *LEVEL control record.	Reassemble using an *LEVEL control record.
A22	INVALID LIST DECK OPTION . . . ASSEMBLY TERMINATED	LIST DECK or LIST DECK E is specified when macros are called.	Reassemble and do not specify either LIST DECK or LIST DECK E options.

Figure A-2 (Part 2 of 2). Assembler error messages

FORTRAN MESSAGES AND ERROR CODES

compilation messages

Near the end of compilation, the FORTRAN compiler prints core usage information and the features supported as follows:

```
FEATURES SUPPORTED
    EXTENDED PRECISION
    ONE WORD INTEGERS
    TRANSFER TRACE
    ARITHMETIC TRACE
    ORIGIN
    IOCS
CORE REQUIREMENTS FOR XXXXX
COMMON YYYYY VARIABLES YYYYY PROGRAM YYYYY
```

where

XXXXX is the program name specified in the *NAME control record or in the SUBROUTINE or FUNCTION statement.
YYYYY is the number of words allocated for the specified parts of the program.

During a subprogram compilation, the compiler prints the following message:

```
RELATIVE ENTRY POINT ADDRESS IS XXXX (HEX)
```

where

XXXX is the address of the entry point relative to the address of the first word of the subprogram being compiled.

The compiler prints the following messages for successful and unsuccessful compilations, respectively:

```
END OF COMPILATION
COMPILATION DISCONTINUED
```

compilation error messages

During compilation, the compiler checks to determine if certain errors occur. If one or more of these errors are detected, the compiler prints the error messages at the conclusion of compilation, and the object program is not stored on disk. Only one error is detected for each statement. In addition, due to the interaction of error conditions, the occurrence of some errors may prevent the detection of others until the errors detected first are corrected. With the exception of the messages listed in Figure A-4, the error messages printed by the FORTRAN compiler have the following format:

```
C nn ERROR IN STATEMENT NUMBER xxxxx+yyy
```

where

C nn is the error code number in Figure A-3. xxxxx is all zeros until the first numbered statement is encountered in your program. When a valid statement number is encountered, xxxxx is replaced by that statement number. Statement numbers on specification statements and statement functions are ignored. When xxxxx is all zeros, yyy is the statement line in error (excluding comments and continuation lines). When xxxxx is a valid statement number, yyy is a count of statements from that numbered statement (counted as 0) to the statement in error. If the erroneous statement has a statement number, yyy is not printed.

For example:

```
   DIMENSION E(I,6,6)              (error C 08)
   DIMENSION F(4,4),G(2,7),
  1H(34,21),I(5,8)                 (recall that the 1 in column 6
                                   indicates a continuation line)
   DIMENSION J(3,2,6))             (error C 16)
   FORMAT (I50,F5.2))              (error C 27)
10 WRITE (1'C) ARRAY
   WRITE (1'C) ARRAYS              (error C 07)
```

This example causes the following error messages to be printed:

```
C 08 ERROR AT STATEMENT 00000+001
C 16 ERROR AT STATEMENT 00000+003
C 27 ERROR AT STATEMENT 00000+004
C 07 ERROR AT STATEMENT 10    +001
```

Look up the error numbers in Figure A-3 to determine the causes of the errors.

Note that a FORTRAN compiler error message can be caused by an invalid character in the source statement. In that case, the character in question is replaced with an ampersand in the listing. Errors in specification statements and any other obvious errors should be examined first. Since variables are not defined when a statement contains a compiler error, valid statements that reference the variables may also be flagged.

Error code	Cause of error
C01	Nonnumeric character in statement number
C02	More than 5 continuation cards, or continuation card out of sequence
C03	Syntax error in CALL LINK or CALL EXIT statement
C04	Unrecognizable, misspelled, or incorrectly formed statement
C05	Statement out of sequence
C06	A statement follows a STOP, RETURN, CALL LINK, CALL EXIT, or GO TO statement, or an IF statement does not have a statement number
C07	Name longer than 5 characters, or name not starting with an alphabetic character
C08	Incorrect or missing subscript within dimension information (DIMENSION, COMMON, REAL, or INTEGER)
C09	Duplicate statement number
C10	Syntax error in COMMON statement
C11	Duplicate name in COMMON statement
C12	Syntax error in FUNCTION or SUBROUTINE statement
C13	Parameter (dummy argument) appears in COMMON statement
C14	Name appears twice as a parameter in SUBROUTINE or FUNCTION statement
C15	*IOCS control record in a subprogram
C16	Syntax error in DIMENSION statement
C17	Subprogram name in DIMENSION statement
C18	Name dimensioned more than once, or not dimensioned on first appearance of name.
C19	Syntax error in REAL, INTEGER, or EXTERNAL statement
C20	Subprogram name in REAL or INTEGER statement, or a FUNCTION subprogram containing its own name in an EXTERNAL statement
C21	Name in EXTERNAL that is also in a COMMON or DIMENSION statement
C22	IFIX or FLOAT in EXTERNAL statement
C23	Invalid real constant
C24	Invalid integer constant
C25	More than 15 dummy arguments, or duplicate dummy argument in statement function argument list
C26	Right parenthesis missing from a subscript expression
C27	Syntax error in FORMAT statement
C28	FORMAT statement without statement number
C29	Field width specification greater than 145
C30	In a FORMAT statement specifying E or F conversion, w greater than 127, d greater than 31, or d greater than w, where w is an unsigned integer constant specifying the total field length of the data, and d is an unsigned integer constant specifying the number of decimal places to the right of the decimal point
C31	Subscript error in EQUIVALENCE statement
C32	Subscripted variable in a statement function
C33	Incorrectly formed subscript expression
C34	Undefined variable in subscript expression
C35	Number of subscripts in a subscript expression, and/or the range of the subscripts does not agree with the dimension information
C36	Invalid arithmetic statement or variable; or, in a FUNCTION subprogram the left side of an arithmetic statement is a dummy argument or in COMMON
C37	Syntax error in IF statement
C38	Invalid expression in IF statement

Figure A-3 (Part 1 of 3). FORTRAN error codes

Error code	Cause of error
C39	Syntax error or invalid simple argument in CALL statement
C40	Invalid expression in CALL statement
C41	Invalid expression to the left of an equal sign in a statement function
C42	Invalid expression to the right of an equal sign in a statement function
C43	In an IF, GO TO, or DO statement, a statement number is missing, invalid, incorrectly placed, or is the number of a FORMAT statement
C44	Syntax error in READ, WRITE or FIND statement
C45	*IOCS record missing with a READ or WRITE statement (mainline program only)
C46	FORMAT statement number missing or incorrect in a READ or WRITE statement
C47	Syntax error in input/output list; or an invalid list element; or, in a FUNCTION subprogram, the input list element is a dummy argument or in COMMON
C48	Syntax error in GO TO statement
C49	Index of a computed GO TO is missing, invalid, or not preceded by a comma
C50	*TRANSFER TRACE or *ARITHMETIC TRACE control record or CALL PDUMP statement present, with no *IOCS control record in a mainline program
C51	Incorrect nesting of DO statements; or the terminal statement of the associated DO statement is a GO TO, IF, RETURN, FORMAT, STOP, PAUSE, or DO statement
C52	More than 25 nested DO statements
C53	Syntax error in DO statement
C54	Initial value in DO statement is zero
C55	In a FUNCTION subprogram the index of DO is a dummy argument or in COMMON
C56	Syntax error in BACKSPACE statement
C57	Syntax error in REWIND statement
C58	Syntax error in END FILE statement
C59	Syntax error in STOP statement
C60	Syntax error in PAUSE statement
C61	Integer constant in STOP or PAUSE statement greater than 9999
C62	Last executable statement before END statement is not a STOP, GO TO, IF, CALL LINK, CALL EXIT, or RETURN statement
C63	Statement contains more than 15 different subscript expressions
C64	Statement too long to be scanned, because of compiler expansion of subscript expressions or compiler addition of generated temporary storage locations
C65*	All variables undefined in an EQUIVALENCE list
C66*	Variable made equivalent to an element of an array in such a manner as to cause the array to extend beyond the original of the COMMON area
C67*	Two variables of array elements in COMMON are equated, or the relative locations of two variables or array elements are assigned more than once (directly or indirectly). This error is also given if an attempt is made to allocate a standard precision real variable at an odd address by means of an EQUIVALENCE statement
C68	Syntax error in an EQUIVALENCE statement; or an illegal variable name in an EQUIVALENCE list
C69	Subprogram does not contain a RETURN statement, or a mainline program contains a RETURN statement
C70	No DEFINE FILE statement in a mainline program that has disk READ, WRITE, or FIND statements
C71	Syntax error in DEFINE FILE statement
C72	Duplicate DEFINE FILE statement, more than 75 DEFINE FILES, or DEFINE FILE statement in subprogram

Figure A-3 (Part 2 of 3). FORTRAN error codes

Error code	Cause of error
C73	Syntax error in record number of disk READ, WRITE, or FIND statement
C74	Defined file exceeds disk storage size
C75	Syntax error in DATA statement
C76	Names and constants in a DATA statement not in a one-to-one correspondence
C77	Mixed mode in DATA statement
C78	Invalid hollerith constant in a DATA statement (see "Length of FORTRAN DATA Statement" in Chapter 6)
C79	Invalid hexadecimal specification in a DATA statement
C80	Variable in a DATA statement not used elsewhere in the program or dummy variable in DATA statement
C81	COMMON variable loaded with a DATA specification
C82	DATA statement too long to compile, due to internal buffering. Refer to the section TIPS FOR FORTRAN PROGRAMMERS

* The detection of a code 65, 66, or 67 error prevents any subsequent detection of any of these three errors.

Figure A-3 (Part 3 of 3). FORTRAN error codes

Error number and message	Cause of error
C85 ORIGIN IN SUBPROGRAM	An ORIGIN control record was detected in a subprogram compilation.
C86 INVALID ORIGIN	An attempt has been made to relocate a word at an address exceeding 7FFF (hexadecimal).
C96 WORKING STORAGE EXCEEDED	The working storage area on disk is too small to accommodate the compiled program in disk system format.
C97 PROGRAM LENGTH EXCEEDS CAPACITY	The error occurs when the program in internal compiler format is too large to be contained in core working storage, and the program must be reduced in size in order to compile.
C98 SUBROUTINE INITIALIZE TOO LARGE	During compilation of subprograms a subroutine initialize statement (CALL SUBIN) is generated. The CALL SUBIN statement initializes all references to dummy variables contained within the subprogram to the appropriate core location in the calling program. The nature of the FORTRAN compiler limits the size of any statement in internal compiler format to 511 words. In the case of CALL SUBIN, the size is calculated by the following formula: $S = 5 + ARG + N$ where ARG is the number of arguments in the subroutine parameter list and N is the total number of times the dummy arguments are used within the subprogram. S is the total size of the CALL SUBIN statement; if S ever exceeds 511, an error occurs and the above error message is printed.
C99 CORE REQUIREMENTS EXCESSIVE	The error occurs when the total core requirements exceed 32767 words.

Figure A-4. FORTRAN error messages

DUP AND MUP MESSAGES AND ERROR MESSAGES

DUP messages

When a Disk Utility Program (DUP) function is performed without errors, an informational message is printed on the principal printer. Information messages are described in the following text.

At the end of a DEFINE VOID, one of the following messages is printed:

ASSEMBLER VOIDED
FORTRAN VOIDED
RPG VOIDED
COBOL VOIDED

At the end of a DEFINE FIXED AREA function, the following message is printed:

CART ID XXXX CYLS FXA XXXX DBS AVAIL XXXX FLET SECTOR ADDR XXXX

where

CYLS FXA XXXX is the decimal number of cylinders minus one in the fixed area (the additional cylinder is used for FLET).
DBS AVAIL XXXX is the hexadecimal number of disk blocks remaining in the fixed area after the last program or data file stored there.
FLET SECTOR ADDR XXXX is the hexadecimal sector address of the first cylinder in the fixed area (the sector address of FLET).

At the end of a dump of LET or FLET, the following sign-off message is printed:

END OF DUMPLET/FLET

All other DUP operations, except MUP are followed by this message:

CART ID XXXX DB ADDR XXXX DB CNT XXXX

where

DB ADDR XXXX is the hexadecimal starting address of the program or data file.
DB CNT XXXX is the hexadecimal number of disk blocks being deleted, stored, or dumped.

The error messages printed by DUP are listed in Figure A-5. These messages include the message number and message, the causes of the error messages, and your corrective actions where appropriate.

MUP messages

The sign-off message of the Macro Update Program (MUP) is:

UPDATE COMPLETED

Informational messages that can be printed during a MUP run are:

ABOVE MACRO PURGED

that follows a PURGE control record, and

ABOVE MACRO RENAMED AS
SSSS DDDD MNAME

where

SSSS is the sector address in hexadecimal.
DDDD is the displacement in hexadecimal.
MNAME is the new macro name.

The error messages printed by MUP are listed in Figure A-5. These messages include the message number and message, the causes of the error messages, and your corrective actions where appropriate.

Error number and message		Cause of error	Your response
D01	NAME IS NOT PRIME ENTRY	The primary entry point name of the program in working storage does not match the name on the DUP control record.	
D02	INVALID HEADER RECORD TYPE	One of the following is detected: 1. A non-DSF program 2. A mispositioned header 3. Foreign data 4. An erroneous subtype	
D03	INVALID HEADER LENGTH	Word 6 of the DSF header is outside the range of 3 through 45. Other causes are similar to those of message D02, except for subtype.	
D05	SECONDARY ENTRY POINT OR NAME ALREADY IN LET	The specified secondary entry point name is already in LET.	Delete the specified entry point name before storing this subroutine.
D06	ENTRY POINT NAME ALREADY IN LET/FLET	The specified name is already in LET or or FLET.	Delete the specified name from LET or FLET before storing this program or data file.
D12	INVALID DISK I/O SPECIFIED	The disk I/O subroutine coded (column 9) on the STORECI control record is other than 0, 1, N, Z, or blank.	
D13	INVALID FUNCTION FIELD	An invalid DUP function is specified on the DUP control record.	
D14	INVALID FROM (CC 13-14)	One of the following: 1. Unacceptable characters are in columns 13 and 14 of the DUP control record. 2. The FROM field specified is not valid with this DUP function.	
D15	INVALID TO FIELD (CC 17-18)	One of the following: 1. Unacceptable characters are in columns 17 and 18 of the DUP control record. 2. The TO field specified is not valid with this DUP function.	
D16	INVALID NAME FIELD (CC 21-25)	One of the following: 1. A required name is not specified. 2. The specified name contains a syntax error.	

Figure A-5 (Part 1 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D17	INVALID COUNT FIELD (CC 27-30)	Columns 27 through 30 are blank or include alphabetic characters. The count field requires a decimal number.	
D18	INVALID FUNCTION DURING TEMPORARY	This function is not allowed during the JOB T mode.	
D19	CARTRIDGE NOT ON SYSTEM	The cartridge specified as the TO or FROM cartridge is not specified on the JOB control record as being used for this job.	
D20	CARTRIDGE ID OUTSIDE VALID RANGE (0001-7FFF)		Correct the cartridge ID and retry.
D21	INVALID STOREMOD. SIZE OF REPLACEMENT EXCEEDS SIZE OF ORIGINAL	The replacement version of the program or data file is larger than the current stored version.	Delete the old version of the program or data file and retry.
D22	PROGRAM NOT IN WORKING STORAGE	One of the following: 1. The disk block count for the requested program in working storage is zero. 2. The program is not in working storage.	
D23	INVALID SYSTEM OVERLAY SUBTYPE SPECIFIED	The system overlay subtype indicator (column 11) on a STORE control record is not in the range 0 through 9.	
D24	COUNT FIELD TOO LARGE	One of the following: 1. The count field extends beyond column 30 of a DEFINE FIXED AREA control record. 2. Column 31 is not a minus sign.	
D25	REQUIRED FORMAT NOT IN W.S.	During a STOREMOD, the format of the LET or FLET entry does not agree with the format in working storage.	
D26	NAME NOT FOUND IN LET/FLET	The name specified on a DELETE or DUMP control record is not in LET or FLET.	
D27	SOURCE NOT IN DSF	The format indicator of the FROM cartridge indicates that working storage on this cartridge does not contain a DSF program.	
D30	INVALID RECORD TYPE	An invalid type binary record has been read when storing from cards or paper tape.	

Figure A-5 (Part 2 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D31	PROGRAM OR DATA EXCEEDS DESTINATION DISK AREA	The number of disk blocks required to store a program or data file exceeds the amount of space available in the specified TO field.	
D32	INVALID CORE IMAGE CONVERSION	The core load builder has inhibited the continuation of STORECI. The specific reason has been printed by the core load builder (see "Core Load Builder Error Messages" in this appendix).	
D33	LET/FLET OVERFLOW. A CORE DUMP FOLLOWS	A ninth sector of LET/FLET is required (or a seventh sector of LET on a non-system cartridge) for the LET/FLET entry. A core dump follows this message since the affected cartridge may have to be reloaded. The dump allows you to locate the condition that caused the error. Use of the affected cartridge is not recommended until the problem is investigated.	You must delete a program with a LET or FLET entry of similar size before this program can be stored.
D41	INVALID STORECI CONTROL RECORD	A control record read after a STORECI is not a LOCAL, NOCAL, FILES, or G2250 record, or a mainline name is specified on a LOCAL, NOCAL, or G2250 record.	
D42	STORECI CONTROL RECORDS INCORRECTLY ORDERED	LOCAL, NOCAL, FILES, and G2250 control records are intermixed.	All records of a given type must be loaded together.
D43	INCORRECT CONTINUATION	A comma at the end of a record indicates continuation to the next record; however, it is not continued.	
D44	ILLEGAL CHARACTER IN RECORD	An illegal character, probably a blank, is in the record.	
D45	ILLEGAL FILE NUMBER	One of the following: 1. A nonnumeric character is in a file number. 2. A file number is more than 5 characters long.	
D46	ILLEGAL NAME	One of the following: 1. A name is more than 5 characters long. 2. A name contains characters other than A through z, 0 through 9, or $. 3. A name contains embedded blanks.	

Figure A-5 (Part 3 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D47	ILLEGAL CARTRIDGE ID	One of the following: 1. The specified cartridge ID is not in the range /0001 through /7FFF. 2. The specified cartridge ID contains an invalid character.	
D48	SCRA BUFFER OVERFLOW	The supervisor control record area (SCRA) cannot contain all the LOCAL, NOCAL, FILES, or G2250 information.	
D70	LAST ENTRY IN LET/FLET NOT 1DUMY	A DELETE operation cannot find the end of LET or FLET. The header for this LET/FLET sector contains the count of unused words in this sector. This count should point to the last 1DUMY entry; however, the entry to which it now points is not a 1DUMY.	
D71	1DUMY ENTRY IN LET/FLET IS FOLLOWED BY A SECONDARY ENTRY POINT	The name on the DELETE control record points to a secondary entry point that follows a 1DUMY entry point. The primary entry is not in LET/FLET.	
D72	FIRST ENTRY IN LET/FLET SECTOR IS A SECONDARY ENTRY POINT	The LET/FLET table is improperly constructed; the first entry is not a primary entry.	
D80	FIXED AREA PRESENT	The FORTRAN compiler, RPG compiler, or assembler cannot be eliminated if a fixed area is defined on the disk.	
D81	ASSEMBLER NOT IN SYSTEM	The assembler has been previously deleted from the system.	
D82	FORTRAN NOT IN SYSTEM	The FORTRAN compiler has been previously deleted from the system.	
D83	INCREASE VALUE IN COUNT FIELD (CC 27-30)	The count field read is a value of zero or one; the first DEFINE FIXED AREA requires one cylinder for FLET plus one cylinder of fixed area. Thereafter, as little as one cylinder of additional fixed area can be defined.	
D84	DEFECTIVE SLET		The cartridge must be reloaded.
D85	FIXED AREA NOT PRESENT	The control record specifies a decrease in the fixed area, or specifies the fixed area as the TO field, and a fixed area is not on the cartridge.	

Figure A-5 (Part 4 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D86	DECREASE VALUE IN COUNT FIELD	One of the following: 1. Enough working storage is not available to allow the fixed area to be defined or expanded by the amount specified in the count field (cc 27 through 30). 2. The number of unused cylinders in the fixed area is insufficient to decrease the fixed area the amount specified in the count field. This message is preceded by a count of the number of cylinders available: XXXX CYLS AVAILABLE. The count is a decimal number.	
D87	RPG NOT IN SYSTEM	The RPG compiler has been previously deleted from the system.	
D88	COBOL NOT IN SYSTEM	The COBOL compiler (a program product) has been previously deleted from the system.	
D90	CHECK SUM ERROR	One of the following: 1. A check sum error is detected in a binary card or paper tape record. 2. Binary cards are out of order.	
D92	INVALID DISKZ CALL. A CORE DUMP FOLLOWS	While performing a DUP function, an attempt has been made to read or write sector 0, or to read or write with a negative word count. This is a system error. A core dump follows this message since the affected cartridge may have to be reloaded. The dump allows you to locate the condition that caused the error. Use of the affected cartridge is not recommended until the problem is investigated.	
D93	CARTRIDGE OVERFLOW	While performing a DUP function, an attempt has been made to read or write a sector beyond 1599 decimal.	
D100	LIBRARY NOT FOUND	The library named on a LIB, BUILD, JOIN, or CONCAT statement cannot be found on drives currently in use. If the statement is a LIB, BUILD, or JOIN, all statements are ignored until the next LIB, BUILD, or ENDUP statement is encountered. If the statement is a CONCAT, processing continues with the next control statement.	Correct the name field in the statement in error, or change the // JOB control record to include the drive on which the named library resides, or define the macro library using a *DFILE or *STOREDATA control record.

Figure A-5 (Part 5 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D101	INVALID SUBFIELD COL XX	One of the following: 1. If on an INSERT or DELETE statement, the sequence number is incorrectly specified; that is, it is negative, nonnumeric, or the sequence numbers are reversed. 2. If on a SELECT statement, an incorrect parameter is specified. 3. If on a NAME statement, an invalid parameter was detected, and processing continues with the next LIB, BUILD, or ENDUP statement. 4. If on an INSERT or DELETE statement, processing continues with the next control statement. 5. If on a SELECT statement, processing continues with the remainder of the statement.	XX indicates the column in which the error was found. Correct the error and rerun the portion of the job that is affected.
D102	ILLEGAL REQUEST	One of the following: 1. An invalid statement was detected. 2. An INSERT or DELETE statement is not preceded by an UPDATE or RENAME statement. 3. An OUTPUT operation was requested using a cartridge configured for paper tape. Processing continues with the next control statement.	Correct the error and rerun the portion of the job that is affected.
D103	LIBRARY OVERFLOW	One of the following: 1. The library last specified by a LIB or BUILD statement does not have enough room to perform the operation. 2. If on a JOIN or an ADD statement, the operation is suppressed and the library is restored to its previous state. 3. If on an INSERT statement, the statements listed prior to the message are the only ones that can be included. Processing continues with the next LIB, BUILD, or ENDUP statement.	Do one of the following: 1. Purge unneeded macros or delete unneeded statements to obtain additional space in the current library. If this is not possible, define a larger library using an *DFILE or *STOREDATA control record, join the old library to a new one, and delete the old library. Once the additional space is obtained, rerun the portion of the job that is affected. 2. If on an INSERT statement, you may have to alter the INSERT statement as the statements in the macro library may have been resequenced.

Figure A-5 (Part 6 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D104	MACRO NOT FOUND	The macro name specified on an OUTPUT, PURGE, RENAME, or UPDATE statement cannot be found in the library being processed. Processing continues with the next control statement.	Do one of the following: 1. Correct the macro name on the statement in error. 2. Specify the correct macro library. Then, rerun the portion of the job that is affected.
D105	SEQUENCE NUMBER NOT FOUND	The sequence number on an INSERT or DELETE statement is out of the range of the macro and cannot be found, or the sequence numbers on multiple INSERT and/or DELETE statements for the same macro are out of order. Processing continues with the next control statement.	Place a correct sequence number on the statement in error, and rerun the portion of the job that is affected.
D106	LIBRARY NOT SPECIFIED	An attempt was made to operate on a macro without specifying a macro library. Processing continues with the next LIB, BUILD, or ENDUP statement.	Place a LIB or BUILD statement before the statement before the statement in error, and rerun the portion of the job that is affected.
D107	SPILL OVERFLOW	Macro text insertions have caused the capacity of working storage spill to be exceeded. Processing continues with the next LIB, BUILD, or ENDUP statement.	Correct the sequence numbers in the unprocessed INSERT statements, if necessary, and rerun these statements. Additional disk drives may have to be defined to provide adequate working storage.
D108	CONTROL STATEMENT READ	An * or // statement has been read, and the MUP run is terminated. Control is returned to the supervisor for a // statement or to DUP for an * statement.	
D109	NAME STATEMENT NOT FOUND	The operation attempted requires a NAME statement, and one has not been processed after the last LIB or BUILD statement. Processing continues with the next LIB, BUILD, or ENDUP statement.	Insert a NAME statement, and rerun the portion of the job that is affected.
D110	INVALID NAME	One of the following: 1. The name field on a LIB, BUILD, JOIN, CONCAT, UPDATE, ADD, PURGE, RENAME, or OUTPUT statement was left blank. 2. The name specified is invalid. 3. Apostrophes are improperly placed. If on a LIB, BUILD, or JOIN statement, processing continues with the next LIB, BUILD, or ENDUP statement. If on a CONCAT, UPDATE, ADD, PURGE, RENAME, or OUTPUT statement, processing continues with the next control statement.	

Figure A-5 (Part 7 of 8). DUP/MUP error messages

Error number and message		Cause of error	Your response
D112	NONBLANK CARD READ ENTER BLANK CARDS		1. Remove the stacked input from the card hopper. 2. Press NPRO to clear out nonblank cards. 3. Place blank cards followed by the NPRO nonblank cards and the stacked input in the hopper. 4. Press reader START and console keyboard PROGRAM START.
D116	LIBRARY NOT INITIALIZED	One of the following: 1. The library named on a LIB, JOIN, or CONCAT statement is not properly initialized. 2. The library specified on a BUILD statement is not a data file. If on a LIB, or JOIN statement, processing continues with the next LIB, BUILD, or ENDUP statement. If on a CONCAT statement, processing continues with the next control statement.	Do one of the following: 1. Initialize the library with a BUILD statement, and rerun the portion of the job that is affected. 2. Correct the BUILD statement and rerun the portion of the job that is affected.
D117	INVALID PARAMETER	One of the following: 1. A parameter has been detected that was not defined in the NAME statement. 2. More than 20 parameters are specified in a NAME statement. 3. A parameter greater than one character was used in the format or tag field. If the error occurs during an OUTPUT operation, the operation is terminated and processing continues with the next control statement. If the error occurs during a listing operation, this is a warning message, and the invalid parameter is printed as //N where N is 1 through 20. *Note:* N may be truncated if the field size is exceeded.	

Note. In addition to the DUP error messages just listed, the following message:

NO SUCH ERROR MESSAGE NUMBER

can be printed immediately followed by a 2-digit hexadecimal number. This message is an indication of a system error. The message is likely to be printed if DUP operations are performed while the physical core size and the configured core size do not agree. This situation is not supported by most system programs.

Figure A-5 (Part 8 of 8). DUP/MUP error messages

SYSTEM LOADER MESSAGES AND ERROR MESSAGES

Informational messages are not printed during an initial load.

At the completion of a reload, the following message is printed:

END OF RELOAD

The error messages and the corrective action that you perform are listed in Figure A-6. Procedures A and B that are referenced under the column "Your response" are included at the end of the figure.

Error number and message		Your response
	From phases 1 and 2	
E01	CHECKSUM ERROR	Follow procedure A or restart initial load. If the input is paper tape, this message can be caused by a paper tape read error. In such a case, follow procedure B.
E02	INVALID RECORD OR BLANK	Follow procedure A or restart initial load.
E03	SEQ ERROR OR MISSING RECORDS	Follow procedure A or restart initial load. The missing record may be end-of-program record.
E04	ORG BACKWARD	Inspect the deck for records missing or out of sequence. Correct the deck and restart from the record in error.
E05	INITIALIZE THE CARTRIDGE	The cartridge ID cannot be found in DCOM because DCOM is defective or an attempt is being made to initial load a cartridge that has not just been initialized or has been improperly initialized. Initialize and initial load the cartridge.
	From phase 1 only	
E11	INVALID DRIVE NO.	Set all bit switches off. Set bit switches to select physical drive number and press PROGRAM START. Drive 0—All switches off Drive 1—Switch 15 on Drive 2—Switch 14 on Drive 3—Switches 14 and 15 on Drive 4—Switch 13 on Drive 5—Switches 13 and 15 on Drive 6—Switches 13 and 14 on Drive 7—Switches 13, 14, and 15 on Drive 8—Switch 12 on Drive 9—Switches 12 and 15 on Drive 10—Switches 12 and 14 on
E12	ID SECTOR DATA INVALID	Initialize using DCIP or DISC and follow with an initial load.
E13	CONFIG DECK ERROR	System configuration deck may be missing, out of place, or may contain an error in one or more records. Correct the deck and restart load.
E14	FILE PROTECT ADDR TOO HIGH	This error occurs on a reload only. The last program in the user area extends into the last two cylinders on the cartridge. These cylinders are required by the system loader during a reload operation. The file protect address must be lowered before a reload can be accomplished.
E15	PHID RECORD ERROR	Follow procedure A or reload and restart.
E16	INITIAL LOAD THE CARTRIDGE	The ID sector indicates that this cartridge has not been loaded since initialization by DCIP or DISC. Only an initial load may be performed.
E17	ERROR IN LOAD MODE RECORD	Follow procedure A or restart load.
E18	PAPER TAPE ERROR	The paper tape system loader has found a word count greater than 54. This is probably due to incorrect sequencing of tapes, a faulty tape, or a paper tape reader malfunction. Correct error and restart load.
E19	INVALID SLET/RELOAD TABLE CHECKSUM	System loader will ignore the checksum and continue if PROGRAM START is pressed. However, the cartridge should be initialized and an initial load performed.
	From phase 2 only	
E20	FIXED AREA PRESENT	Programs may not be added to a cartridge with a fixed area defined. Press PROGRAM START to restore the resident image and DCOM.
E21	SYSTEM DECK ERROR	A defective record follows the sector break record. Correct the deck and restart the initial load or continue the reload from the preceding sector break record.

Figure A-6 (Part 1 of 3). System loader error messages

Error number and message	Your response
E22 SCRA OVERLAY – STOP	The cushion area used for allowing expanded or added phases has been used up. An initial load must be performed to store these phases on the cartridge. Press PROGRAM START to restore the resident image and DCOM.
E23 PHASE ID OUT OF SEQUENCE	The ACCUMULATOR displays the phase ID that is out of sequence (from last card read). Place the decks in proper order and continue from the sector break record of the correct phase.
E24 PHASE MISSING	Error occurred when phase ID (word 11) of last record read was processed. Inspect load mode record, PHID record and phase ID of previously loaded phase to determine which phase is now required. Locate missing phase, place deck in reader starting with sector break record of missing phase and continue.
E25 PHASE ID NOT IN PHID RECORD	The ACCUMULATOR displays the extraneous phase ID. To ignore the phase press PROGRAM START. To load the phase correct the PHID record and restart the load.
E26 PHASE ID NOT IN SLET	If the error occurred during reload table processing, the ACCUMULATOR displays the phase ID sought, and the extension displays the ID of the phase requesting the SLET search. Press PROGRAM START to place zeros in the entry and process the next. If the extension displays zeros, a phase is being added, and the phase which should precede it cannot be found. The ACCUMULATOR displays the phase ID searched for. Press PROGRAM START to restore the resident image and DCOM.
E27 DEFECTIVE SLET	SLET is defective. Initialize the cartridge and perform an initial load.
E28 SLET FULL	The ACCUMULATOR displays the ID of a phase that may not be added because the SLET table is full. Press PROGRAM START to ignore the phase and continue. An initial load should be performed as SLET is probably defective.
E29 PROGRAM NOT PRESENT	A program or phases of a program defined in the primary PHID record cannot be reloaded unless the program is currently on the cartridge. Press PROGRAM START to ignore the phases of this program.
E30 RELOAD TABLE FULL	If this error occurs before the '81' record is read the ACCUMULATOR displays the ID of a phase which may not be loaded because the reload table is full. Press PROGRAM START to ignore the phase and continue.
E31 MISSING PHASE ID DUE TO DEFECTIVE SLET OR RELOAD TABLE	The ACCUMULATOR displays the ID of a phase listed in the reload table as a phase requiring SLET information but the phase itself does not appear in SLET. Initialize the cartridge and perform an initial load.
E32 MISSING SYSTEM I/O PHASE	All system I/O subroutines must be on the cartridge and in SLET. Initialize the cartridge and perform an initial load.

Procedure A

If cards are being read from a 1442 Card Read Punch:

1. Lift the remaining cards from the hopper and press nonprocess run out (NPRO).
2. Correct the card in error (first card nonprocessed out) and place the two nonprocessed cards ahead of the cards removed from the hopper.
3. Place the deck back in the hopper.
4. Press reader START.
5. Press console PROGRAM START.

Figure A-6 (Part 2 of 3). System loader error messages

If cards are being read from a 2501 Card Reader:

1. Lift the remaining cards from the hopper and press NPRO.
2. a. Correct the card in error (last card in stacker prior to NPRO) and place this card followed by the single nonprocessed card ahead of the cards removed from the hopper or,
   b. If the error occurred after the PHID card was read and before the type 81 card was read the system loader is in double buffer mode. Correct the card in error (in this case the second from last card in the stacker when the error occurred) and place the last two cards from the stacker and the nonprocessed card ahead of the cards removed from the hopper. Note, however, that the last card in the stacker will be the next card processed since it is already in the double-buffer.
3. Place the deck back in the hopper.
4. Press reader START.
5. Press console PROGRAM START.

If the input is paper tape, procedure A is applicable only to errors E15 and E17.

Procedure B

1. Place a mark on the tape adjacent to the highest sprocket tooth under the read starwheels as a point of reference.
2. Count back (from that mark) the number of frames displayed in the ACCUMULATOR and mark the tape.
3. Reposition the tape in reader so that the last mark is at the point of reference.
4. Press console PROGRAM START.

Note: Corrective actions for error messages E04, E21, E23, and E24 are not applicable to paper tape since a faulty tape must normally be replaced in full.

Figure A-6 (Part 3 of 3). System loader error messages

SATELLITE GRAPHIC JOB PROCESSOR ERROR MESSAGES

Figure A-7 lists the error messages that are printed by the satellite graphic job processor (SGJP). The numbered messages are printed on the console printer; the messages preceded by IKyxxxz are displayed on the 2250 screen.

SGJP is described in detail in the publication *IBM System/360 Operating System and 1130 Disk Monitor System User's Guide for Job Control from an IBM 2250 Display Unit Attached to an IBM 1130 System*, GC27-6938.

Error number (if any) and message	Cause of error	Your response
G01 INITIALIZATION FAILURE	Contact has not been made with SGJP in the System/360 during an attempt to initialize the telecommunications line via the GTNIT data transmission subroutine.	Ensure that the System/360 operator has issued a VARY ON command for the 1130/2250 subsystem on which this error message is printed. Then, using the console keyboard, type either an R to retry the operation or a C to cancel SGJP.
G02 LINE ERROR	An attempt to transmit data to the System/360 is unsuccessful because of an I/O error; standard retries are unsuccessful.	Using the console keyboard, type either an R to retry the operation or a C to cancel SGJP.
G03 SYNCHRONIZATION ERROR	The operation is not completed, either because both the System/360 and the 1130/2250 subsystem are in read mode, or because the System/360 terminated communication.	Using the console keyboard, type either an R to retry the operation or a C to cancel SGJP.
IKyxxxz message text THE SATELLITE GRAPHIC JOB PROCESSOR MUST RESTART	SGJP is terminated because an internal error occurred. If the error recurs, refer to the publication, *IBM System/360 Operating System Messages and Codes*, GC28-6631, under the message code (IKyxxxz) for further explanation of the error condition.	Perform the END function, which causes the LOG ON frame to reappear. Perform the LOG ON operation again.
IKyxxxz message text THE SATELLITE GRAPHIC JOB PROCESSOR MUST TERMINATE	SGJP must be terminated because an internal error occurred. If the error recurs, refer to the publication, *IBM System/360 Operating System Messages and Codes*, GC28-6631, under the message code (IKyxxxz) for further explanation of the error condition.	Perform the END function. This returns SGJP to the state it was in before the initial (CANCEL key) attention.

Figure A-7. SGJP error messages

RJE MESSAGES AND ERROR MESSAGES

The error messages that are printed by the RJE program are listed in Figure A-8. The first digit of the messages has the following meaning:

0–Error in RJE00
1–Error in the initializing part of RJE
2–Error during the processing of the RJE program; does not require an operator reply through the console keyboard
5–Error during the processing of the RJE program; requires a reply through the console keyboard from the operator

Messages that are not caused by errors but are printed by the RJE program are listed in Figure A-9.

Error number and message	Cause of error	System action	Your response
J01 INVALID CARD	The control card that contains the work station generation information is invalid or contains invalid information (see "Generation of the 1130 RJE Work Station Program" in Chapter 10).	The work station prepares to read a new data card.	Enter a valid data card.
J10 INVALID PRINTER	Information from the disk monitor system indicates that the principal print device is not an 1132 Printer or a 1403 Printer.	The work station program exits to the disk monitor supervisor.	Reload the RJE Work Station Program after performing a system reload that specifies the 1132 or the 1403 as the principal print device (see Chapter 8 for information about system reload).
J11 INVALID READER	Information from the disk monitor system indicates that the principal I/O device for system is not a 1442 Card Reader or a 2501 Card Reader.	The work station program exits to the disk monitor supervisor.	Reload the RJE Work Station Program after performing a system reload that specifies the 1442 Card Reader or the 2501 Card Reader as the principal I/O device (see Chapter 8 for information about system reload).
J12 LOGICAL DRIVE X NOT IN SYSTEM	The area on disk reserved for your exit data is on a logical disk drive that is not present during this RJE run. The logical drive number replaces X in the message.	The work station program exits to the disk monitor supervisor.	Change your exit parameters or ready the requested logical drive, and reload the RJE Work Station Program.
J13 TOO MANY EQUATS	The number of subroutines equated by you and the RJE program in the current job is more than 25.	The work station program exits to the disk monitor supervisor.	Reload the RJE Work Station Program with a smaller number of subroutines specified in the *EQUAT control record. *Note:* The RJE program internally requires the following number of EQUATS. Compress/expand feature—2 pairs 2501 Card Reader—2 pairs 1132 Printer—1 pair
J14 DISK ERROR OCR=	A permanent error is encountered while attempting to read data from disk during the initialization part of the RJE program.	The program continues according to your response.	Enter one of the following codes: T – Indicates exit to the disk monitor supervisor requesting a terminating dump of the contents of core storage on the printer. X – Indicates exit to the disk monitor supervisor without printing the contents of core storage on the printer.

Figure A-8 (Part 1 of 5). RJE Work Station Program error messages

Error number and message		Cause of error	System action	Your response
J20	RJSTART MISSING	The requirement for an RJSTART command is not satisfied.	The program waits for your response.	Enter an RJSTART command through the card reader, and press PROGRAM START on the console to resume processing.
J21	.. DATA INVALID	A .. DATA command contains invalid parameter. *Note:* This message is also printed if the requested logical disk drive is not present.	The program waits for your response.	Use the operator communication request facility (see message J90 in Figure A-9).
J22	INVALID INPUT	The input entered from the console keyboard does not start with the JECL identifier (. .) followed by at least one blank.	The program waits for more input from the keyboard.	Enter a work station command or press EOF.
J23	INPUT ABORTED BY CENTRAL	The central system has terminated input from the work station and sends a message that explains why input was terminated (see "Messages Sent to Work Stations" in *IBM System/360 Operating System Remote Job Entry,* GC30-2006, for a list of the messages).	The program waits for input from the line.	When the message from the central system is printed, take the indicated action. To resume input, follow the procedures described under "Console Keyboard Procedures" in Chapter 10.
J51	LINE ERROR OCR=	An unrecoverable error is encountered while reading or writing on the communication line, or the line cannot be opened.	The RJE program closes the communication line, if it is open, and waits for your response.	Enter one of the following codes through the console keyboard: A — Indicates that input is available at the card reader. If you select this option, the first card in the card reader must be an RJSTART command. On a switched line, the line must be disconnected before the restart is tried. If this is not done automatically by the work station program, you must do it. Dial again when J91 ESTABLISH LINE CONNECTION is printed. T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of core storage to the printer. X — Indicates exit to the disk monitor supervisor, without printing the contents of core storage on the printer.

Figure A-8 (Part 2 of 5). RJE Work Station Program error messages

Error number and message	Cause of error	System action	Your response
J52 DISK ERROR INPUT OCR=	A permanent error is encountered while attempting to read input from disk. This message is printed only if your disk input is being read at the time the error occurs.	Reading of input data files and card reader input is discontinued. Any available output from the central system is accepted after you make your response. The system continues according to your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): A — Indicates that input is available at the card reader. B — Indicates that commands are to be read from the console key-board. C — Indicates that available output is accepted. (Any pending keyboard input is processed first.) T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer. *Note:* You may have to resubmit a job that has been partially entered, but must precede this by either obtaining the output of, or deleting, the job in question.
J53 DISK ERROR OUTPUT OCR=	An unrecoverable error is encountered while attempting to write data on disk. This message is printed only if data is being written on disk by the IBM-supplied user-exit routine.	Output from the central system is discontinued. The disposition of the output is specified by the use of the CONTINUE command. The system continues as directed by your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): A — Indicates input is available at the card reader. (Any pending keyboard and disk input is processed first.) B — Indicates that commands are to be read from the console keyboard. C — Indicates that any pending input (keyboard, disk or card) is processed. If input is not available, the system maintains the line operations. T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer.
J54 DISK ERROR OCR=	An unrecoverable error is encountered while attempting to read RJE constants or error messages from disk. If this message is printed, an RJE error message that indicates the original error may not be printed.	The program continues according to your response.	Enter one of the following codes: T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer. X — Indicates exit to the disk monitor supervisor without printing the contents of core storage on the printer.

Figure A-8 (Part 3 of 5). RJE Work Station Program error messages

Error number and message	Cause of error	System action	Your response
J55 END OF DISK AREA OCR=	You did not reserve space or reserved too little space on disk for user-exit output data sets.	Output from the central system is discontinued. The system continues as directed by your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): A — Indicates that input is available at the card reader. (Any pending keyboard and disk input is processed first.) B — Indicates that commands are to be read from the console keyboard. C — Indicates that any pending input (keyboard, disk, or card) is processed. If pending input does not exist, the system maintains the line operations. T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer.
J56 CARD READER ERROR OCR=	An error has occurred on the card reader that requires your intervention.	The system waits for your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): A — Indicates you have corrected the problem, and the program resumes card reader input. E — Indicates that you could not correct the problem. The program assumes an end-of-file (. . null card) indication closes the card reader.
J57 CARD PUNCH ERROR OCR=	An error has occurred on the card punch that requires your intervention.	The system waits for your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): D — Indicates you could not correct the problem. Output from the central system is discontinued and a . . CONTINUE command has to be transmitted to resume output. P — Indicates that you have corrected the problem, and the program resumes card punch output.

Figure A-8 (Part 4 of 5). RJE Work Station Program error messages

Error number and message	Cause of error	System action	Your response
J58 PRINTER ERROR OCR=	An error has occurred on the printer that requires your intervention. This message is also printed if the length of the records received from the central system exceeds the size of a print line.	The system waits for your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): D – Indicates you could not correct the problem. Output from the central system is discontinued, and a . . CONTINUE command must be transmitted to resume output. P – Indicates that you have corrected the problem, and the program resumes printer output.
J59 PREOPERATIVE ERROR CODE XXXX OCR=	A preoperative error has occurred in the user-exit subroutine, or a logical disk drive has been referenced that was present during the job processing preceding the loading of the work station program, but that has later become not ready. The pre-operative error code that replaces XXXX is explained in Appendix B.	The system waits for your response.	Enter one of the following codes (within approximately 3 minutes on a switched line): C – Indicates that you have corrected the problem, and the program retries the operation. T – Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer. X – Indicates exit to the disk monitor supervisor without printing the contents of core storage on the printer.

Figure A-8 (Part 5 of 5). RJE Work Station Program error messages

Message number and message	Reason for message	System action	Your response
J90 OCR=	You have indicated that you want to communicate with the system by pressing PROGRAM STOP and PROGRAM START on the console keyboard.	The system waits for your response.	Enter one of the following codes (within approximately 21 seconds for switched lines and also within the same time limit on a leased or multipoint line, if a line operation is in progress): A — Indicates that input is available at the card reader. B — Indicates that commands are to be submitted from the console keyboard. D — Indicates that receiving output is to be discontinued. N — Indicates that the system ignore the request. T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer.
J91 ESTABLISH LINE CONNECTION	This message is printed only on a switched line 1130 work station. You must establish a connection with the central system.	The system waits for you to complete the connection.	Perform the dial-up procedure to establish the connection with the central system (see "Operating Procedures" in the *IBM 1130 Synchronous Communications Adapter Subroutines,* GC26-3706).
J92 DATA rrrr0c0f TO DISK AT xaaa,bbbb	This message is printed only when the IBM-supplied user-exit subroutine is used to write a data set to disk. The message codes have the following meanings. rrrr — The logical record length in hexadecimal for fixed blocked or unblocked records. c — The type of control characters used, where c may have the following values: 0 — No control characters used 1 — IBM System/360 machine code 2 — ASA control characters are used	The user-exit data set is written on disk. The disk block information part of the message is written when the data set is completed; therefore, if a line error or a disk error occurs before the whole data set is received, this portion of the message remains blank.	

Figure A-9 (Part 1 of 3). RJE Work Station Program messages

Message number and message	Reason for message	System action	Your response
J92 (Continued)	f – The IBM System/360 Operating System record format, where f may have the following values: 1 – Fixed unblocked records 2 – Fixed blocked records 3 – Variable unblocked records 4 – Variable blocked records 5 – Undefined records x – The logical disk drive number. aaa – The starting sector address of the data set in hexadecimal. bbbb – The length of the data set in disk blocks where there are 40 packed EBCDIC characters per block (16 disk blocks per sector). The last block may not be filled.		
J93 PROCEED	This message is printed as a result of a B reply to a J90 OCR= message. The work station is ready to receive commands from the keyboard.	The K.B. SELECT light on the console keyboard is turned on, and the program waits for input from the keyboard.	Enter the desired commands with an EOF after each command. After entering the last command, press EOF again to indicate the end of input. (On a switched line, you have approximately 3 minutes to enter each command.)
J94 PUNCHED OUTPUT	A SYSOUT data set is to be punched on a Model 6 or 7 card read punch that is also used to read card input, and a coded card is in the punch station.	The system waits for your action.	You may load blank cards in the punch and then press any character key or the space bar to resume processing. If you want the output to be punched in the prepunched cards, you press any character key or the space bar as just described. You must take action within approximately 3 minutes to maintain line communication. If this time limit is exceeded, a line error occurs. The RJE program is then restarted as described under message J51. You receive punched output if you place an RJSTART command, a null command, and the blank cards in the card reader, then reply A to the line error message.

Figure A-9 (Part 2 of 3). RJE Work Station Program messages

Message number and message	Reason for message	System action	Your response
J94 (Continued)			*Note:* If punched output is to be sent to a 1442 Card Read Punch that is also used for reading, all punched output should be specified as deferred.

Figure A-9 (Part 3 of 3). RJE Work Station Program messages

SUPERVISOR MESSAGES AND ERROR MESSAGES

The monitor supervisor causes all monitor system control records to be printed on the principal printer.

During a DCOM update operation (after each JOB control record or when your program calls SYSUP), the following information is printed:

```
LOG DRIVE CART SPEC CART AVAIL PHY DRIVE
   XXXX        XXXX        XXXX        XXXX
```

where

LOG DRIVE is the drive number specified on the JOB control record (or in the calling sequence of the SYSUP subroutine).
CART SPEC is the specified cartridge ID.
CART AVAIL is the available cartridge ID.
PHY DRIVE is the physical drive number starting with zero.

One line is printed for each physical drive that is ready on the system. The logical drive may be different from the physical drive; that is, physical drive zero may be defined as logical drive 2.

After the cartridge information is printed, the following is printed:

```
V2MXX  ACTUAL XXK  CONFIG XXK
```

where

V2MXX is the current version and modification level of the 1130 Disk Monitor System
ACTUAL XXK indicates the physical core size
CONFIG XXK indicates the configured core size specified by a system load or reload

Figures A-10 and A-11 list the error messages, and their causes, that are printed by Phases 1 and 2, respectively, of the System Control Record Program. Figure A-12 lists the error messages that are printed by the SYSUP DCOM update program.

SYSUP waits with zero displayed in the ACCUMULATOR if it fails to find the SLET entry for the principal printer subroutine. This error can be caused by your replacing the master cartridge with a nonsystem cartridge. Press INT REQ on the console keyboard to flush to the next job. An error printout during SYSUP results in termination of execution.

Error number and message		Cause of error
M11	INVALID MONITOR CONTROL RECORD	A // record was not recognized as a valid monitor control record.
M12	EXECUTION SUPPRESSED	$NXEQ was set upon detection of an error that would prevent successful execution by the system. Execution is bypassed.
M13	DUP SUPPRESSED	$NDUP was set upon detection of an error that would prevent successful DUP operation. DUP is bypassed.
M14	SYSTEM PROGRAM DETECTED MONITOR CONTROL RECORD	A system program has detected a monitor control record when none was expected. The control record is passed to the MCRA for processing. This situation often occurs as a result of a missing END statement in an assembler language program.
M15	ILLEGAL CARTRIDGE ID	A cartridge ID contains an illegal character or is a negative number. The job is terminated.
M16	PROGRAM VOIDED	ASM, FOR, or RPG required but the FORTRAN compiler and/or assembler and/or RPG compiler was either not loaded by the system loader or was voided by a DUP DEFINE.

Figure A-10. Phase 1. System Control Record Program error messages

Error number and message	Cause of error
M21 ABOVE RECORD NOT A SUPERVISOR CONTROL RECORD	The last record read is not a LOCAL, NOCAL, G2250, or FILES, record.
M22 SUPERVISOR CONTROL RECORDS INCORRECTLY ORDERED	LOCAL, NOCAL, FILES and G2250 records cannot be intermixed. All records of each type must be kept together.
M23 INCORRECT CONTINUATION	A comma at the end of the record indicated that the record would be continued; however, it was not.
M24 ILLEGAL CHARACTER IN RECORD	An illegal character, probably a blank, appeared in the record.
M25 ILLEGAL FILE NUMBER	A non-numeric character appears in a file number or the number is more than 5 characters long.
M26 ILLEGAL NAME	A name is more than 5 characters long, or contains characters other than A through Z, 0 through 9, or $, or a name contains embedded blanks.
M27 ILLEGAL CARTRIDGE ID	The cartridge ID specified is not in the range /0001 through /7FFF or contains an illegal character.
M28 SCRA BUFFER OVERFLOW	The supervisor control record area (SCRA) cannot contain all the LOCAL, NOCAL, FILES, EQUAT, or G2250 record information.
M29 ILLEGAL DISK SUBROUTINE REQUESTED	A character other than 0, 1, N, Z, or blank appeared in column 19 of the XEQ card.
M30 INVALID CHAR. IN G2250 OPTION COLUMN	A character other than U, N, or blank appeared in column 13, 15, 17, 19, or 21 of the *G2250 control record.
M31 REQUESTED W.S. DR NOT AVIL.	The requested cartridge has not been specified in the job record.

Figure A-11. Phase 2. System Control Record Program error messages (Phase 2 errors cause execution to be bypassed)

Cartridge ID and message	Cause of error
XXXX IS NOT AN AVAILABLE CARTRIDGE ID	A requested cartridge ID is not on any cartridge on the system, or the ID is not listed #CIDN of the DCOM on the cartridge.
XXXX IS A DUPLICATED SPECIFIED CARTRIDGE ID	The cartridge ID was listed as appearing on more than one drive on the JOB card.
XXXX IS A DUPLICATED AVAILABLE CARTRIDGE ID	A specified ID appears on more than one cartridge on the system.
XXXX IS NOT A SYSTEM CARTRIDGE	An attempt has been made to specify a non-system cartridge as the master cartridge (logical 0).

Figure A-12. SYSUP – DCOM update error messages

RPG COMPILER MESSAGES AND ERROR NOTES

compiler messages

Near the end of compilation, core usage information and literal parameters are printed in the following format:

INDICATORS
IND DISP . . Indicators through H9 are printed for all programs (relative address)

FIELD NAMES

FIELD DISP	L	T	D . . .
(field name)	(field length)	(field type)	(number of decimal positions)

LITERALS
LITERAL LENGTH TYPE DISP . . .

KEY ADDRESS OF OBJECT PROGRAM
Name of routine Hex DISP . . .

END OF COMPILATION

See "Sample Program 3" in Appendix H for an actual program listing.

The relative address that is printed can be used to compute the actual address of the indicator, field, literal, or routine the program is loading. The actual address is computed as follows: add the relative address to the execution address (as printed in the core map) and subtract hexadecimal 11 from the sum. The answer is the actual address.

compilation errors

If working storage is exceeded, compilation is terminated and the following message is printed:

WORKING STORAGE EXCEEDED

If terminating errors are detected during compilation, the following messages are printed:

ERROR(S) IN COMPILATION
END OF COMPILATION

The program is executed if any of the detected errors are in the correctable class; that is, an asterisk (*) preceeds the error note number (see Figure A-13 for an explanation of the asterisk).

Compiler error notes are printed as follows:

1. As each statement is processed, it is checked for invalid conditions. When an error is detected, the error note:

   NOTE xxx

   is printed on the line following the line in error in the columns reserved for program ID. (xxx is a 3-digit error note number.)
2. The source program is checked for invalid file references (modified, unreferenced, multidefined) and error notes are printed as required. These notes are printed within or below the source listing in the following format:

   NAME NOTE xxx

   NAME is replaced with the name of the invalid file reference.

3. After the printout of indicators, field names, and literals at the end of compilation, any errors on extended diagnostics are printed in the following format:

	Seq. No.	*Error*
EXTENDED FILE DEF. EXT. AND/OR INPUT DIAGNOSTICS	xxxx	NOTE xxx
EXTENDED CALCULATION SPECIFICATION DIAGNOSTICS	xxxx	NOTE xxx
EXTENDED OUTPUT SPECIFICATION DIAGNOSTICS	xxxx	NOTE xxx

   The sequence number (xxxx) is a 4-digit number that is assigned to program statements. Comments cards are not assigned sequence numbers. Some error messages (such as, 227 and 228) are printed together with the number of the statement following the error because the error cannot be determined until then.

4. After the extended diagnostics, a summary of all error messages is printed as follows:

   DIAGNOSTIC MESSAGE EXPLANATIONS
   NOTE xxx y error message (y is the specification type)
   or
   NOTE *xxx y error message
   ***UNCORR ERR JOB TERM
   A message is printed for each error.

All RPG Compiler error notes are listed and explained in Figure A-13. The term *specification is dropped* means that a statement is no further processed by the compiler; the term *no immediate action taken* means that the compiler continues processing a statement by looking for additional errors. An * preceding an error note number indicates that the error cannot be corrected. The program is not executed, and the key addresses of the program are not printed.

Note	Spec type	Error message	Cause of error	System action
* 1	F	FILE TYPE COL 15 INVALID	File Type entry is not I, O, U, or C, or is blank.	I is assumed.
* 2	F	PROC MODE COL 28 INVALID	Mode of Processing entry is not L, R, or blank.	Blank is assumed.
* 3	F	REC ADDR COL 29-30 INVALID	Length of Record Address Field (or key length) entry is invalid or is blank.	08 is assumed.
4	F	REC ADDR TYPE COL 31 INVALID. CORRECT ENTRY ASSUM	Warning only. The correct value for the file type (column 32) is assumed.	Blank is assumed for sequential files. K is assumed for ISAM files.
5	F	TABLE FILE COL 16 REQ E COL 39. E ASSUM	Extension Code entry must be E if File Designation entry is T (table file).	E is assumed.
* 6	F	FILE DESIGN INVALID WITH INPUT FILE	File Designation entry column 16 is not P, S, R, C, or T with an input file (I in column 15).	P is assumed.
7	F	OF IND COL 33-34 INVALID BLK ASSUM	Overflow Indicator entry is invalid for the device type specified.	Blanks are assumed.
8	F	FILE TYPE COL 15 INVALID 0 ASSUM	File Type entry is invalid with a printer device in columns 40 through 46.	0 is assumed.
9	F	MULT PRI FILES DEF. SEC ASSUM	Only one primary file (P in column 16) is allowed. Other input files are designated as secondary (S in column 16).	Secondary is assumed for all but first input file.
* 11	F	FILE ORG COL 32 INVALID	File Organization entry is not I, numeric (1 through 9), or blank; or, two I/O areas are specified for a table file.	Blank is assumed.
12	F	EXT CODE COL 39 NOT BLK BLK ASSUM	Extension Code must be blank for output files.	Blank is assumed.
13	F	EOF COL 17 INVALID E ASSUM	End of File entry is not E or blank.	E is assumed.
14	F	SEQ COL 18 INVALID A ASSUM	Sequence entry not A, D, or blank.	A is assumed.
15	F	FILE DESIG COL 16 NOT BLK. BLK ASSUM	File Designation entry is not blank for an output file.	Blank is assumed.
* 16	F	C IN FILE TYPE COL 15 INVALID WITH DEVICE	File Type entry C requires card read punch in device columns 40 through 46.	READ 42 is assumed.
17	F	REC ADDR FILE REQ E COL 39. E ASSUM	File Designation entry R (record address file) requires an E in Extension Code column.	E is assumed.
18	F	FILE FMT INVALID. F ASSUM	File Format (column 19) is not F. 1130 RPG uses fixed length records only.	F is assumed.

Figure A-13 (Part 1 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
19	F	BLOCK LNG COL 20-23 NOT BLK. BLK ASSUM	Block Length must be blank for 1130 RPG.	Blanks are assumed.
20	F	REC LNG COL 24-27 INVALID. 120 ASSUM PRINTER. ALL ELSE 80	Record Length is improperly specified or is blank.	120 is assumed for printer. 80 is assumed otherwise.
* 21	F	U IN FILE TYPE COL 15 INVALID WITH DEVICE	File Type entry U requires disk I/O in device columns 40 through 46.	DISK is assumed.
22	F	COL 17-18 INVALID WITH PRINTER. BLK ASSUM	End-of-File and Sequence entries are invalid with a printer.	Blanks are assumed.
23	F	COL 28 INVALID WITH CHAIN FILE, R ASSUME	Mode of processing must be random for chain file.	R is assumed.
* 24	F	MORE THAN 8 SEC FILES DEF	The number of secondary files (S1 in column 16) exceeds the maximum allowable 8.	8 is assumed.
25	F	OF IND COL 33-34 INVALID BLK ASSUM	Overflow indicator not OF on OV.	Blanks are assumed.
27	F	EOF COL 17 NOT BLK WITH OUTPUT. BLK ASSUM	End-of-File entry must be blank with output files.	Blank is assumed.
29	F	EXT CODE 39 INVALID. E ASSUM	Extension Code entry is not E or blank with input file.	E is assumed.
* 30	E	FROM FILENAME COL 11-18 INVALID	From Filename entry is missing or not left-justified.	Specification is dropped.
* 31	E	FROM FILENAME COL 11-18 INVALID	From Filename entry was not defined on a File Description Specification form.	Specification is dropped.
* 32	E	FROM FILENAME COL 11-18 INVALID	From Filename entry requires an E in Extension Code column on the File Description Specifications form.	Specification is dropped.
* 33	E	CHAINING FLD COL 9-10 INVALID	Chaining Field entry is not C1, C2, or C3 for chaining file (same entry as columns 61 and 62 of Input Specifications form).	Specification is dropped.
* 34	E	SEQ COL 7-8 INVALID	Record Sequence entry must be 2 alphabetic or 2 numeric characters for chaining file (same entry as columns 15 and 16 of Input Specifications form).	Specification is dropped.
* 35	E	TO FILENAME COL 19-26 INVALID	To Filename entry is missing or not left-justified on RAF or chaining type specifications.	Specification is dropped.
* 36	E	TO FILENAME COL 19-26 INVALID	To Filename entry was not defined on RAF or chaining type specifications on a File Description Specifications form.	Specification is dropped.

Figure A-13 (Part 2 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
* 37	E	TO FILENAME COL 19-26 INVALID	To Filename entry is not the same as the filename defined as a RAF or chaining type specification on a File Description Specifications form.	Specification is dropped.
38	E	COL 33-57 NOT BLK. BLK ASSUM	Columns 33 through 57 of the Extension Specifications form must be blank for all chaining type specifications.	Blanks are assumed.
39	E	COL 7-10 NOT BLK. BLK ASSUM	Columns 7 through 10 of the Extension Specifications form must be blank for all RAF type specifications.	Blanks are assumed.
40	E	COL 33-57 NOT BLK. BLK ASSUM	Columns 33 through 57 of the Extension Specifications form must be blank for all RAF type specifications.	Blanks are assumed.
41	E	COL 7-10 NOT BLK. BLK ASSUM	Columns 7 through 10 of the Extension Specifications form must be blank for all table type specifications.	Blanks are assumed.
* 42	E	TO FILENAME COL 19-26 INVALID	To Filename entry is missing or not left-justified.	Specification is dropped.
* 43	E	TO FILENAME COL 19-26 INVALID	To Filename entry was not defined on a File Description Specifications form.	Specification is dropped.
* 44	E	TO FILENAME COL 19-26 INVALID	To Filename entry is not defined as an output file on a File Description Specifications form.	Blanks are assumed.
* 45	E	TBL NAME COL 27-32 OR 46-51 INVALID	Table Name entries missing or not left-justified. Columns 46-51 are required for alternating input formats only.	Specification is dropped.
* 46	E	COL 27-29 OR 46-48 NOT TAB	First 3 characters of table names must be TAB. Columns 46 through 48 are required for alternating input formats only.	TAB is assumed.
* 47	E	NO OF TBL ENTRIES COL 33-35 NOT NUMERIC	Number of table entries per record. These columns must contain a right-justified decimal number.	10 is assumed.
* 48	E	NO OF TBL ENTRIES COL 36-39 NOT NUMERIC	Number of table entries per table. These columns must contain a right-justified decimal number.	100 is assumed.
* 49	E	TBL ENTRY LNG COL 40-42 OR 52-54 NOT NUMERIC	Length of table entry. These columns must contain a right-justified decimal number. Columns 52 through 54 are required for alternating input formats only.	8 is assumed.
50	E	PACKED ENTRY COL 43 OR 55 INVALID. BLK ASSUM	Packed entry is not P or blank, or invalid for specified device.	Blank is assumed.

Figure A-13 (Part 3 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
* 51	E	NUM DEC POS COL 44 OR 56 INVALID	Decimal positions is not blank or a number.	Zero is assumed.
52	E	TBL SEQ COL 45 OR 57 INVALID. BLK ASSUM	Sequence entry is not A, D, or blank.	Blank is assumed.
* 53	E	FORM TYPE COL 6 NOT VALID	The next specification should have been an E or I specification.	Specification is dropped.
56	F	COL 47-65, 67-70 MUST BE BLK FOR 1130 RPG	Specified columns are not used with 1130 RPG except for ISAM load files.	Blanks are assumed.
* 57	F	ISAM NUMBER OF RECORDS INVALID	The number of records specified for an ISAM load (columns 47 through 52) is not numeric or left-justified.	One is assumed.
60	H	NO RPG CONTROL CARD. BLK ASSUM	Warning only. A compilation and listing will be performed for this run.	Blanks are assumed for all entries.
61	H	COL 11 INVALID. BLK ASSUM	Type of run. This entry should be B, D, or blank.	Blank is assumed.
63	H	COL 17-20 INVALID. BLK ASSUM	Sterling entries are not blank, 0, 1, or 2, as required.	Blanks are assumed.
64	H	COL 21 INVALID. BLK ASSUM	Inverted print option entry is not I or blank.	Blank is assumed.
65	H	COL 26 INVALID. BLK ASSUM	Alternating collating sequence entry is not A or blank.	Blank is assumed.
67	H	PROG NAME COL 75-80 INV. RPGOBJ ASSUM	Program Name entry on RPG Control Card is invalid.	RPGOBJ is assumed.
* 71	C	RSLT FLD COL 43-48 REQUIRED	Result Field name is required but is missing.	Specification is dropped.
72	C	RSLT FLD COL 43-48 MUST BE BLK. BLK ASSUM	Result Field must be blank for COMP, GOTO, EXIT, TAG, SETOF, SETON, CHAIN, BEGSR, ENDSR, EXSR, and EXCPT.	Blanks are assumed.
* 73	C	FACT1, COL 18-27 INVALID	Factor 1 requires a fieldname, label, or literal with the specified operation.	Numeric literal 1 is assumed.
* 74	C	FACT2 COL 33-42 INVALID	Factor 2 requires a fieldname, label, or literal with the specified operation.	Numeric literal 1 is assumed.
75	C	RSLT IND COL 54-59 INVALID. 00 ASSUM	Resulting Indicator is not 01 through 99, H1 through H9, L1 through L9, OF, or OV.	00 is assumed for indicator in error.
76	C	FACT1 COL 18-27 MUST BE BLK. BLK ASSUM	Factor 1 entry must be blank for the operation being performed.	Blanks are assumed.
77	C	FACT2 COL 33-42 MUST BE BLK. BLK ASSUM	Factor 2 entry must be blank for the operation being performed.	Blanks are assumed.

Figure A-13 (Part 4 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
* 78	C	CTRL LEVEL COL 7-8 INVALID	Control Level column 7 is not L or blank.	Blank is assumed.
* 79	C	DETAIL CALC DOES NOT PRECEDE TOTAL CALC	A detail calculation, columns 7 and 8 blank, follows a total calculation, columns 7 and 8 L0 through L9 or LR.	L0 is assumed.
* 80	C	FACT1 COL 18-27 INVALID	Factor 1 entry is not left-justified.	Numeric literal 1 is assumed.
* 81	C	FACT2 COL 33-42 INVALID	Factor 2 entry is not left-justified.	Numeric literal 1 is assumed.
* 82	C	FACT1 COL 18-27 INVALID	Factor 1 entry is an improperly stated literal or field name.	Numeric literal 1 is assumed.
* 83	C	FACT2 COL 33-42 INVALID	Factor 2 entry is an improperly stated literal or field name.	Numeric literal 1 is assumed.
* 84	C	FACT1 COL 18-27 INVALID	Factor 1 entry is a field name of more than 6 characters.	First six characters are assumed.
* 85	C	FACT2 COL 33-42 INVALID	Factor 2 entry is a field name of more than 6 characters.	First six characters are assumed.
* 86	C	OPER CODE COL 28-32 INVALID	Operation code is missing or unrecognizable.	MOVE operation code is assumed.
87	C	CTRL LEV COL 7-8 INVALID. L0 ASSUM	Column 7 is L but column 8 is not 0 through 9 or R.	L0 is assumed.
* 89	C	RSLT FLD COL 43-48 REQUIRED	Result Field entry is improperly defined.	Specification is dropped.
* 94	C	RSLT FLD LNG COL 49-51 INVALID	Field Length entry is blank, not numeric, or not right-justified; or, Field Length entry contains an embedded blank.	014 is assumed. 0 is assumed for blank.
* 95	C	DEC POS COL 52 INVALID	Decimal Position entry is not blank or numeric.	0 is assumed.
96	C	HLF ADJ COL 53 INVALID. H ASSUM	Half adjust entry is not H or blank.	H is assumed.
* 97	C	RSLT IND COL 54-59 REQUIRED	A resulting indicator is required for this operation.	Internal indicator is assigned.
* 98	C	IND COL 9-17 INVALID	Indicator entry improperly defined.	Indicator is dropped.
*100	I	STERL COL 71-74 INVALID	Sterling entry not numeric or sterling not defined on RPG Control Card. This note can be printed by input or output specifications.	Blanks are assumed.

Figure A-13 (Part 5 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
*101	I	FLD REC RELATION IND COL 63-64 INVALID	Field Record Relation Indicator unrecognizable.	Blanks are assumed.
*102	I	PLUS, MINUS, ZERO/BLK IND COL 65-70 INVALID	Indicator columns 65 through 70 unrecognizable.	Blanks are assumed.
*103	I	OVER 60 REC TYPE SPEC'S	Input has more than 60 record identification columns 6 through 42.	Specification is dropped.
*109		INPUT OR OUTPUT SPECS MISSING OR INVALID	Input or output specifications are required.	Job is terminated.
110	I	FORM TYPE COL 6 INVALID	Form Type is not I, C, or O and column 7 does not contain an *.	Specification is dropped.
*111	I	FILENAME COL 7-14 INVALID	Filename entry is not defined.	Specification is dropped.
*112	I	FILENAME COL 7-14 INVALID	Filename entry is not correctly defined on the File Description form.	Specification is dropped.
*113	I	'AND' CD OUT OF SEQ	'AND' card is first card in deck, first specification after field name, or invalid file type.	Specification is dropped.
*114	I	NO RECORD ID IN CARD BEFORE 'AND' CARD	Record ID entry columns 21 through 41 of Input Specifications form required in card before 'AND' card.	Specification is dropped.
*115	I	'OR' CD OUT OF SEQ	'OR' card is first card in deck, first specification after field name, or invalid file type.	Specification is dropped.
*116	I	FILENAME COL 7-14 INVALID	Filename entry not left-justified.	Specification is dropped.
*117	I	FILENAME COL 7-14 INVALID	Filename entry begins with a numeric character.	Specification is dropped.
*118	I	FILE AND FLD NAME ARE BOTH ON SAME SPEC	File and field names cannot both appear on same specification.	Filename entry is assumed.
119	I	SEQ COL 15-16 BLK. AA ASSUM	Sequence entry must be 2 alpha or 2 numeric characters.	AA is assumed.
*120	I	SEQ COL 15-16 ALPHA SEQ AFTER NUM SEQ	Alpha sequence entries must appear before numeric sequence entries.	Numeric sequence last used is assumed.
*121	I	SEQ COL 15-16 IS INVALID	Ascending numeric sequence is required, or the first entries must begin with 01.	Numeric sequence last used is assumed.
122	I	NUMBER ENTRY COL 17 INVALID. N ASSUM	Sequence is numeric and the number entry column is not N or 1.	N is assumed.

Figure A-13 (Part 6 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
123	I	OPTION ENTRY COL 18 INVALID. 0 ASSUM	Sequence is numeric, and the option entry column is not 0 or blank.	0 is assumed.
124	I	REC IDENTIFYING IND COL 19-20 INVALID. BLK ASSUM	Record Identifying Indicator entry is not 01 through 99.	Blanks are assumed.
125	I	STKR SEL COL 42 INVALID. BLK ASSUM	Stacker Select entry is one of the following: 1. Not 1, 2, or blank. 2. Specified with 2 I/O areas. 3. Invalid with the reader specified.	Blank is assumed.
*126	I	INVALID INPUT FILE	Input file has been specified as I, C, or U in column 15 of File Description Specifications form and no input specifications are found for that file. The file was not defined on an Extension Specifications form.	No immediate action taken.
*127	I	POSITION ENTRY COL 21-24, 28-31, 35-38 INVALID	Position entry contains a non-numeric character.	0 is assumed.
128	I	'NOT' ENTRY COL 25, 32 OR 39 INVALID. N ASSUM	'NOT' entry not N or blank.	N is assumed.
129	I	C/Z/D ENTRY COL 26, 33 OR 40 INVALID. C ASSUM	Combined/Zone/Digit entry is not C, Z, or D.	C is assumed.
130	I	FIELD NAME SPEC OUT OF SEQ	Field Name Type specification is first in deck, after invalid filename or invalid AND or OR specification.	Specification is dropped.
*131	I	FLD NAME COL 53-58 INVALID	Field Name entry is not left-justified.	Specification is dropped.
*132	I	FLD NAME COL 53-58 INVALID	Field Name entry does not begin with an alphabetic character.	Specification is dropped.
*133	I	FROM OR TO COL 44-51 INVALID	From or To columns are blank.	0001 is assumed.
*134	I	FROM OR TO COL 44-51 INVALID	From or To columns contain a non-numeric character.	0 is assumed.
*135	I	TO COL 48-51 LESS THAN FROM COL 44-47	Defined field length less than 1.	1 is assumed.
*136	I	PACKED INPUT FLD INVALID	Packed input field length defined by From and To fields is greater than 8, or packed field is invalid for input device.	8 is assumed.
137	I	PACKED ENTRY COL 43 INVALID. P ASSUM	Packed entry is not P or blank.	P is assumed.
*138	I	DEC POS COL 52 INVALID	Decimal Positions are not numeric.	0 is assumed.
*139	I	NUMERIC FLD GT 14	Numeric field length is greater than 14 characters.	Field length of 14 is assumed.

Figure A-13 (Part 7 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
*140	I	CTRL LEV COL 59-60 INVALID	Column 59 not L.	L in column 59 is assumed.
*141	I	CTRL LEV COL 59-60 INVALID	Column 60 is not numeric.	1 in column 60 is assumed.
*142	I	MATCH OR CHAIN ENTRY COL 61-62 INVALID	Column 61 not M or C.	M in column 61 is assumed.
*143	I	MATCH OR CHAIN ENTRY COL 61-62 INVALID	Column 62 is not numeric.	1 in column 62 is assumed.
*144	I	MATCH ENTRY COL 61-62 NOT M1-M9	Match entry is invalid.	M9 is assumed.
145	I	RSLT IND COL 65-68 SPECIFIED FOR NON-NUM FLD. INDIGN	Plus and minus indicators cannot be used with an alphameric field.	Indicator is ignored.
*146		ALPHA FLD GT 256	Alphameric field length is more than 256 characters.	Field length of 256 is assumed.
*147	I	STERL FLD INVALID	Sterling field has more than 3 decimal positions specified.	3 is assumed.
*148	I	STERL FLD INVALID	Sterling field has no decimal positions specified.	0 is assumed.
149	I	REC ID SPEC OUT OF SEQ OR NO FIELDS FOR GIVEN REC	Warning only. Record ID specification is out of order, or no fields are indicated for a given record.	No immediate action is taken.
*150	I	PACKED FLD MUST BE NUMERIC	Decimal Position entry column 52 is blank.	0 is assumed.
*151	I	FROM TO OR RECORD ID ZERO	From, To, or Position entries are zero.	0001 is assumed.
*152	I	FLD REC POS BLK, BUT TEST CHAR PRESENT	Position entry 27, 34, or 41 contains a valid test character.	No immediate action is taken.
*155	F	KEY SIZE EXCEEDS REC LNG	Key length columns 29 and 30 (ISAM file) is greater than record length.	No immediate action is taken.
*158	F	KEY LNG EXCEEDS 50	Key length columns 29 and 30 (ISAM file) is more than 50 characters.	50 is assumed.
*159		FLD NAME BEGINS WITH 'TAB' BUT IS NOT TBL NAME	Field name beginning with TAB is not a table name. Tables are defined on Extension Specifications form columns 27 through 32.	Specification is dropped.
*160		FORM TYPE COL 6 INVALID	Next Form Type entry should have been 0.	Specification is dropped.

Figure A-13 (Part 8 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
*161	O	INVALID OUTPUT SPEC	Column 6 of specification contains an O, but column 7 does not have * or start of filename. There is no H/D/T/E specified in column 15. The specification is not an AND or OR.	Specification is dropped.
*162	O	FILENAME COL 7-14 INVALID	Filename entry is missing, improperly defined, or undefined.	Specification is dropped.
*163	O	H/D/T/E ENTRY COL 15 OUT OF SEQ	Output lines must be sequenced as follows: H/D/T/E.	Specification is dropped.
*164	O	LINE TYPE COL 15 INVALID	Line Type entry must be H, D, T, or E.	H is assumed.
*165	O	IND COL 23-31 MISSING ON 'OR' SPEC. 00 ASSUM	'OR' specification requires conditioning indicators in columns 23 through 31.	Indicator 00 is assumed.
166	O	IND COL 23-31 MISSING ON 'AND' SPEC. SPEC DROPPED	'AND' specification requires conditioning indicators in columns 23 through 31.	Specification is dropped.
167	O	COL 32-70 MUST BE BLK ON LINE SPEC. BLK ASSUME	File ID and CONTROL specification requires columns 32 through 70 blank.	Blanks are assumed.
168	O	FIELD NAME COL 32-37 INVALID. SPEC DROPPED	Field Name entry is not left-justified.	Specification is dropped.
*169	O	IND COL 23-25, 26-28, OR 29-31, INVALID OR OF OR OV NOT IN 33-34 OF FDS. SPEC DROPPED	Output Indicator entry is incorrect.	Blanks are assumed.
*170	O	CARD OUT OF ORDER	'OR' or 'AND' card is out of sequence.	Specification is dropped.
*171	O	CARD OUT OF ORDER	Field type specification with column 15 blank is not preceded by a valid line type specification.	Specification is dropped
*172	O	OUTPUT FLD SPEC WITH ENTRIES IN COL 7-22	Output field specification requires columns 7 through 22 blank.	Entries in columns 7 through 22 are ignored.
173	O	LEAD OR CLOSE QUOTE COL 45-70 MISSING. NO EDIT	Edit word must be enclosed by apostrophes.	No editing is performed.
174	O	EDIT CODE COL 38 INVALID OR USED WITH ALPHA FLD. BLK ASSUM	Edit code used is invalid or an edit code has been specified with an alpha field.	Blank is assumed.
175	O	BLANK AFTER COL 39 INVALID. BLK ASSUM	Blank After entry not B or blank.	Blank is assumed.
176	O	PACKED ENTRY COL 44 INVALID. BLK ASSUM	Packed entry not P or blank, field is not numeric, or packed field is invalid.	Blank is assumed.
177	O	COL 17-22 NON-BLK ON 'AND' SPEC. BLK ASSUM	Columns 17 through 22 are not blank on 'AND' specification.	Blanks are assumed.

Figure A-13 (Part 9 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
178	O	END POS COL 40-43 INVALID. SPEC DROPPED	End position in Output Record entry is blank, alphabetic, or is incompatible with constant or edit word.	Specification is dropped.
179	O	LEAD OR CLOSE QUOTE COL 45-70 MISSING. SPEC DROPPED	Constant must be enclosed by apostrophes.	Specification is dropped.
*180	C	FLD NAMED COL 43-48 GT 14	On an arithmetic operation, the field named in columns 43 through 48 is longer than 14 characters.	Specification is dropped.
*181	C	MOVE ZONE OPER INVALID	Incorrect alphameric or numeric fields have been specified for this Move Zone operation. Only the low zone of a numeric field can be referred to.	Specification is dropped.
*183	C	FIELD NAME UNDEF	The field name in Factor 1, Factor 2, or Result Field is undefined.	Specification is dropped.
184		FLD NAME UNREF	Warning only. Field Name entry is unreferenced field or table name.	No immediate action is taken.
*185		FLD NAME MULT-DEF	Field Name entry columns 53 through 58 Input Specification, columns 43 through 48 Calculation Specification, or columns 32 through 37 Output Specification contain a multidefined field name. The field name has been defined as alpha and numeric or as same field type with different lengths or as numeric field with different decimal positions.	No immediate action is taken.
*186	C	ARITH OPER SPECIFIED WITH ALPHA FLD	Arithmetic operation specified in operation columns 28 through 32 with an alphameric field specified in Factor 1, Factor 2, or Result field.	Specification is dropped.
*187	C	COMP OPER SPECIFIED WITH ALPHA AND NUM FLD	Alphameric and numeric field being compared. Compare operations are valid only between like fields.	Specification is dropped.
188	C	RSLT FLD LNG COL 49-51 MAY NOT BE LARGE ENOUGH	Warning only. The Result Field may not be long enough to contain the true result.	No immediate action is taken.
*189	C	FACT2 OR RSLT FLD NOT TBL NAME	LOKUP requires table names in Factor 2 columns 33 through 42, and Result Field columns 43 through 48 (if specified).	Specification is dropped.
*190	C	EXSR OPER CALLS ITSELF	Name in Factor 2 is the name of the subroutine of which the EXSR operation is a part (a subroutine may not call itself).	Specification is dropped.
*191	C	TESTZ OPER INVALID	Result Field entry columns 43 through 48 is numeric. TESTZ tests for a high-order zone punch of an alpha field.	Specification is dropped.

Figure A-13 (Part 10 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
192	C	GOTO AND TAG OPERS ARE NOT IN SAME CALC SECTION	Label of the TAG operation and the corresponding GOTO are not in Detail or Total calculations.	Specification is dropped.
193	C	HLF ADJ COL 53 IS INCOMPATIBLE. BLK ASSUM	The number of positions of the arithmetic result is less than or equal to the specified decimal position of the Result Field; therefore, half-adjust cannot be performed.	Blank is assumed.
*194	C	LOKUP OPER INVALID DUE TO UNEQUAL LNGS	Length of Factor 1 columns 18 through 27 and Factor 2 columns 33 through 42 are not equal.	Specification is dropped.
*196	C	MVR OPER NOT PRECEDED BY DIV	There is no remainder to move.	MVR operation is ignored.
*197	C	MVR OPER PRECEDED BY DIV WITH HLF ADJ	Half-adjust effectively removes any remainder.	MVR operation is ignored.
*198	C	LOKUP OPER SPECIFIED WITH ALPHA AND NUM FLD	Factor 1 columns 18 through 27 and Factor 2 columns 33 through 42 must both be alpha or numeric.	Specification is dropped.
*199	C	HIGH AND LOW RSLT IND SPEC FOR LOKUP OPER	High and Low Resulting indicators are both specified for LOKUP operation.	Low indicator is ignored.
*200	F	NO PRIMARY FILE SPECIFIED	No P in column 16 of File Description Specifications form. One file must be defined as primary.	Job is terminated.
*201		FORM TYPE COL 6 INVALID	Next Form Type entry should have been F, E or I.	Specification is dropped.
*202	F	FILENAME COL 7-14 INVALID	Filename incorrectly specified.	Specification is dropped.
*203	F	MORE THAN 10 FILENAMES SPEC	More than the maximum of 10 files are specified.	Only the first 10 are processed.
204	F	UNREF FILENAME	Warning only. A file defined on the File Description Specifications form has not been used in the program.	No immediate action is taken.
205	F	FILE TYPE COL 15 INVALID WITH READ01	Device entry READ01 requires an I in File Type column 15.	Specification is dropped.
*206	F	DEVICE COL 40-46 INVALID	Device name is unrecognizable.	Job is terminated.
207	F	FILENAME COL 7-14 MULT-REF	The filename is specified on the Input or Output Format Specifications form more than once.	No immediate action is taken.
*208	F	FILENAME COL 7-14 MULT-DEF	The same filename is defined on two File Description Specifications forms.	Second specification is dropped.

Figure A-13 (Part 11 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
*210		NO IND OR ONLY PREDEF IND SPEC FOR INPUT REC	At least one indicator is required on input specifications.	Job is terminated.
*212		UNDEFINED RESULT IND	Result indicator used but not defined.	No immediate action is taken.
213		UNREFERENCED IND	Warning only. Indicator specified but not used.	No immediate action is taken.
215	F	FILE DESCR SPEC WITH E COL 39 NOT REF ON EXT SPEC	File description specification with E in column 39 is not used on an extension specification.	No immediate action is taken.
219	O	FLD NAME COL 32-37 UNDEFINED. SPEC DROPPED	Name must be defined on Input or Calculation Specifications form.	Specification is dropped.
*221	I	MATCH FLD LNGS INCOMPATIBLE	Sum of Matching Field lengths must be equal for all record types having matching records specified, or matching fields separated by fields conditioned on Field Record Relation indicators.	No immediate action is taken.
*222	E	TBL NAME MULT-DEF	Same name used for two tables, or the table has been defined as alpha and numeric or as same type with 2 lengths or decimal positions.	No immediate action is taken.
*223	I	FLD IS OUTSIDE THE REC	The input field specified in columns 44-51 is outside the physical record specified in columns 24-27 of the file description specification.	No immediate action is taken.
*224	I	SPLIT CHAIN FLDS IMPROPER	Split chain fields are improperly specified.	No immediate action is taken.
*225	I	SPLIT CTRL FLDS IMPROPER	Split control fields are improperly specified.	No immediate action is taken.
*226	I	SPLIT MATCH FLDS INVALID	Split matching fields are not allowed.	No immediate action is taken.
*227	I	MATCH FLD LNGS INCOMPATIBLE	All match fields of the same level must be the same length on all record types.	No immediate action is taken.
*228	I	CTRL FLD LNG INCOMPATIBLE	The control field on a given control level must be the same length for all record types.	No immediate action is taken.
*229	I	CHAIN FLD LNG INCOMPATIBLE	All fields using the same chaining indicator must be the same length on all record types.	No immediate action is taken.
*230	I	CTRL FLD LNG GT 247	The sum of the control fields on all levels used on a record type cannot exceed 247 characters.	No immediate action is taken.
*231	I	FLD AREA GT REC SIZE	Input field area size exceeds input record length.	No immediate action is taken.
232	O	PRINTER FILE BLK COL 17-22. SPACE 1 AFTER ASSUM	Entry required in columns 17 through 22 for printer carriage control.	Single space after is assumed.

Figure A-13 (Part 12 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
233	O	STKR SEL COL 16 INVALID. BLK ASSUM	Stacker select invalid for output device, or entry is incorrect (not 1 or 2).	Blank is assumed.
234	O	SPACE BEFORE, COL 17, INVALID. 1 ASSUM	There is an entry in column 17, but it is not 0, 1, 2, or 3.	Single space before is assumed.
235	O	SPACE AFTER, COL 18, INVALID. 1 ASSUM	There is an entry in column 18, but it is not 0, 1, 2, or 3.	Single space after is assumed.
236	O	SKIP BEFORE, COL 19-20 INVALID. BLK ASSUM	There is an entry columns 19 and 20, but it is not 01 through 12 or with an 1132 Printer the skip is to channel 7, 8, 10, or 11.	Blanks are assumed.
237	O	SKIP AFTER, COL 21-22, INVALID. BLK ASSUM	There is an entry in columns 21 through 22 but it is not 01 through 12 or with an 1132 Printer the skip is to channel 7, 8, 10, or 11.	Blanks are assumed.
238	O	PACKED FLD COL 44 NOT NUM. BLK ASSUM	Output field is alpha.	Blank is assumed.
239	O	EDIT CODE COL 38 SPECIFIED ON ALPHA FLD. BLK ASSUM	Alpha fields cannot be edited with an edit code.	Blank is assumed.
240	O	STERL SPECIFIED ON NON-NUM FLD, NO STERL ASSUM	Sterling option columns 71 through 74 requested for alpha field.	No sterling is assumed.
*241	O	EDIT WD TOO SMALL	Edit word is too small for field.	No immediate action is taken.
*242	O	EDIT FLD NOT NUM	Alpha fields not edited.	No immediate action is taken.
*243	O	DOLLAR SIGN INVALID	Both fixed and floating dollar sign have been specified.	No immediate action is taken.
*244	O	BOTH CR AND – USED	Both CR and minus are used for credit.	No immediate action is taken.
*245	O	OUTPUT SPEC INVALID	Output specifications are missing or are invalid for this program.	Job is terminated.
*246	O	PAGE FLD IS DEF AS ALPHA	PAGE defined on Input Specifications form with no decimal position in column 52.	No immediate action is taken.
*247	O	FLD LNG GT END POS COL 40-43	Output field length is greater than the indicated End Position in Output Record columns 40 through 43.	No immediate action is taken.
*248	F	INDEX SEQ FILE ADDITION COL 66 INVALID	Column 66 must contain an A for ISAM ADD functions.	No immediate action is taken.
*250	F	INDEX SEQ KEY LNG COL 29-30 INVALID	Key Length entry columns 29 and 30 is not numeric.	8 is assumed.

Figure A-13 (Part 13 of 14). RPG compiler error notes

Note	Spec type	Error message	Cause of error	System action
251	F	INDEX SEQ KEY START POS COL 35-38 INVALID. 1 ASSUM	Key field must start in position one of record.	0001 is assumed.
*252	O	END POS GT RCD LNG	Output field length is greater than Record length (columns 40 through 43).	No immediate action is taken.
*254	O	'ADD' COL 16-18 MUST BE SPEC	'ADD' must be specified if records are added to an ISAM file.	Specification is dropped.
*255	C	FACT2 COL 33-42 INVALID	Entry in Factor 2 must be filename described as a chained file on the File Description Specifications forms.	No immediate action is taken.
256	C	CTRL LEV COL 7-8 INVALID. SR ASSUM	Closed subroutine must follow total calculations.	SR is assumed.
*257	C	ERROR IN SEQ OF ENDSR-BEGSR	BEGSR operation must come first.	No immediate action is taken.
*258	C	BEGSR OR EXSR FACTORS INVALID	BEGSR—Subroutine name must appear in Factor 1 columns 18 through 27. EXSR—Subroutine name must appear in Factor 2 columns 33 through 42.	No immediate action is taken.
259	C	COL 49-59 MUST BE BLK WITH EXSR OR EXCPT. BLK ASSUM	EXSR or EXCPT operation codes require columns 49 through 59 blank.	Blanks are assumed.
260	C	COL 9-17 MUST BE BLK WITH BEGSR. BLK ASSUM	BEGSR operation code requires columns 9 through 17 blank.	Blanks are assumed.
262	C	COL 49-53 MUST BE BLK WITH CHAIN. BLK ASSUM	CHAIN operation code requires columns 49 through 53 blank.	Blanks are assumed.
263	C	IND COL 56-57 MUST BE THE SAME AS IND COL 54-55 HIGH ASSUM	The same indicator must be specified as high and low indicator.	High indicator is assumed for high and low.
264	O	CHAIN SPECIFIED WITH IND IN COL 58-59. BLK ASSUM	Equal indicator cannot be specified on chaining operation.	Blanks are assumed.
*265	C	PAGE FLD INVALID	Page field must be numeric. Field length must be 4 with zero decimal positions.	Field length of 4 and zero decimal positions are assumed.
270	O	SKIP INVALID FOR CONSOLE PRINTER. BLK ASSUM	Console printer has no provisions for forms skipping. Columns 19 through 22 must be blank.	Blanks are assumed.

Figure A-13 (Part 14 of 14). RPG compiler error notes

CORE LOAD BUILDER MESSAGES

Except for the core load map described in Chapter 6, "Programming Tips and Techniques," and messages R41–R45 listed in Figure A-14, the core load builder does not print informational messages. All core load builder messages are listed in Figure A-14. These messages include the message number and message, the causes of the error messages, and your corrective actions where appropriate.

Error number and message		Cause of error	Your response
R00	LOCALS/SOCALS OVERFLOW WORK STORAGE	Enough working storage is not available to accommodate the LOCAL and/or SOCAL overlays required by the core load.	Do one of the following: 1. Change the working storage ID on the JOB control record to the ID of the cartridge on the system that contains the most available working storage. 2. Create more working storage on the present cartridge by deleting subroutines, subprograms, and/or data that is no longer required.
R01	ORIGIN BELOW 1ST WORD OF MAINLINE	The core load builder has been instructed to load a word into an address lower than the first word of the mainline program.	Do one of the following: 1. Remove the ORG statement that is causing the problem. 2. Assign the mainline program origin at a lower address.
R02	DEFINE-FILE(S) OVERFLOW WORK STORAGE	Enough working storage is not available to accommodate any records of the defined file(s).	See the options for error message R00.
R03	NO DSF PROGRAM IN WORKING STORAGE	Working storage does not contain a program when the core load builder is called.	Load the desired program into working storage.
R05	INVALID LOADING ADDR FOR ILS02	ILS02 has been loaded into low COMMON. If error message R48 is also printed, see R48.	Make the mainline program longer so that ILS02 can be loaded in a higher address.
R06	FILE(S) TRUNCATED (SEE FILE MAP)	At least one defined file has been truncated, either because the previously defined storage area in the user area or fixed area is inadequate, or because enough working storage is not available to store the file.	Do one of the following: 1. Redefine the user area or fixed area file. 2. Change the record count specification in the DEFINE FILE statement.
R07	TOO MANY ENTRIES IN LOAD	More than approximately 375 different entry points are referenced in the core load by CALL and/or LIBF statements. If your system has a 4K core size, the number is approximately 125.	Divide the core load into 2 or more links.
R08	CORE LOAD EXCEEDS 32K	The core load builder has been instructed to load a word into a core address that exceeds 32767 (a negative number). The loading process is immediately terminated, since the core load builder cannot process negative addresses. This error is probably caused by bad data being read from the disk.	
R09	LIBF TV REQUIRES 82 OR MORE ENTRIES	At least 82 different entry points are referenced in the core load by LIBF statements.	Divide the core load into 2 or more links.
R16	XXXXX IS NOT IN LET OR FLET	The program name or data file name printed cannot be found in LET or FLET.	Store the program or data file. If the name cannot be explained, the program being loaded has probably been destroyed (bad data was read from the disk).

Figure A-14 (Part 1 of 4). Core load builder error messages

Error number and message	Cause of error	Your response
R17 XXXXX CANNOT BE A LOCAL/ NOCAL	The program named in this message is either a type that cannot appear on a *LOCAL control record, or is a LOCAL that has been referenced, directly or indirectly, by another LOCAL.	
R18 XXXXX LOADING HAS BEEN TERMINATED	The loading of the mainline program named in this message has been terminated as a result of the errors listed in the messages preceding this one.	
R19 XXXXX IS NOT A DATA FILE	The area named in this message does not begin at a sector boundary, which implies that it is not a data file but a DSF program, and thus a possible error.	Choose another area for the storage of this file.
R20 XXXXX COMMON EXCEEDS THAT OF ML	The length of COMMON for the subroutine named in this message is longer than that of the mainline program.	Define more COMMON for the mainline program.
R21 XXXXX PRECISION DIFFERENT FROM ML	The precision, both real and integer, for the subroutine named in this message is incompatible with that of the mainline program.	Make *EXTENDED PRECISION or *ONE WORD INTEGERS the same in the named subroutine and the mainline program.
R22 XXXXX AND ANOTHER VERSION REF'ENCED	At least 2 different versions of the same ISS have been referenced; that is, CARDZ and CARD0 (FORTRAN uses CARDZ). If a disk subroutine is named in the message, it is possible that the XEQ control record specifies one version (DISKZ) whereas the program references another (DISKN). (A blank in column 19 of the XEQ control record causes DISKZ to be used.)	Change the references so that the core load uses only one version of any given I/O subroutine.
R23 XXXXX SHOULD BE IN THE FIXED AREA	The area named in this message is in the user area.	References in DEFINE FILE and DSA statements for *STORECI functions must be to the fixed area.
R39 XXXX is not CURRENTLY MOUNTED	XXXX is a cartridge ID specified on an *FILES card, but not the ID of a cartridge currently mounted.	Change *FILES card to reference an available cartridge or mount the requested cartridge and restart the job.
R40 XXXX (HEX) = ADDITIONAL CORE REQUIRED	One of the following: 1. If the core load was executed, /XXXX is the number of words by which it exceeded core before the core load builder made it fit by creating special overlays (SOCALs). 2. If the core load was not executed, /XXXX is the number of words still required after the core load builder has attempted to make it fit by using SOCALs.	For the second case, create more links or LOCALs.
R41 XXXX (HEX) WORDS UNUSED BY CORE LOAD	*Not an error.* /XXXX is the number of words of core storage not used by this core load.	
R42 XXXX (HEX) IS THE EXECUTION ADDR	*Not an error.* This message follows every successful conversion from DSF to DCI when a core map is requested.	

Figure A-14 (Part 2 of 4). Core load builder error messages

Error number and message		Cause of error	Your response
R43	XXXX (HEX) = ARITH/FUNC SOCAL WD CNT	*Not an error.* Special overlays (SOCALs) are required. /XXXX is the length of the arithmetic/function overlay (see "Incorporating Subroutines" in Chapter 3).	
R44	XXXX (HEX) = FI/O, I/O SOCAL WD CNT	*Not an error.* Special overlays (SOCALs) are required. /XXXX is the length of the FORTRAN I/O, I/O, and conversion subroutine overlay (see "Incorporating Subroutines" in Chapter 3).	
R45	XXXX (HEX) = DISK FI/O SOCAL WD CNT	*Not an error.* Special overlays (SOCALs) are required. /XXXX is the length of the disk FORTRAN I/O overlay, including the 320 word buffer.	
R46	XXXX (HEX) = AN ILLEGAL ML ADDR	One of the following: 1. /XXXX is the address where the core load builder has been requested to start loading the mainline program. However, this address is lower than the highest address occupied by the version of disk I/O requested for this core load. 2. This error may also be caused by starting an absolute mainline program at an odd location. An ORG to an even location, followed by a BSS of an odd number of words, has the same effect as an ORG to an odd location.	Do one of the following: 1. Assign the mainline program origin at a higher address. 2. Request a shorter version of disk I/O. 3. Assign the mainline program origin at an even boundary.
R47	XXXX (HEX) TOO MANY WDS IN COMMON	The length of COMMON specified in the mainline program plus the length of the core load exceeds core storage by /XXXX words.	
R48	XXXX (HEX)	This message is printed with message R05.	The hex value is the number of words that must be added to your mainline program. The reason for this addition is that ILSX2 or a user-written ILS02 would have been loaded into an area where word count and sector address are temporarily placed by the disk routine as a result of an entry to the $DUMP entry point in the skeleton supervisor.
R64	XXXXX IS BOTH A LIBF AND A CALL	The subroutine named in this message is either improperly referenced; that is, a CALL instead of a LIBF or vice versa, or ha or has been referenced in both CALL and LIBF statements.	
R65	XXXXX HAS MORE THAN 14 ENTRY POINTS	This message usually means that the subroutine has been destroyed since a subroutine is not stored if it contains more than 14 entry points.	

Figure A-14 (Part 3 of 4). Core load builder error messages

Error number and message		Cause of error	Your response
R66	XXXXX HAS AN INVALID TYPE	One of the following: 1. The subroutine named in this message: ● Has designated on an XEQ control record and is not a mainline program, or ● Contains a type code other than 3 (LIBF subprogram, not an ISS), 4 (CALL subprogram, not an ISS), 5 (ISS referenced by LIBF), 6 (ISS referenced by CALL), or ● Has been stored with an appropriate subtype. 2. This error can also be caused by a DSA statement referencing a DSF program, or a CALL or LIBF referencing a program in DCI or DDF.	
R67	XXXX HAS AN INVALID GSB ADDRESS	The subroutine named has a Graphic Short Branch order address that is larger than 8191 after relocation.	
R68	XXXXX FILE NUMBER PREVIOUSLY USED	The data file named in this message appears on an *FILES control record equated to a file number that has been previously assigned to another data file.	Change the file numbers on the *FILES control record to point to unique data files.

Figure A-14 (Part 4 of 4). Core load builder error messages

AUXILIARY SUPERVISOR ERROR MESSAGES

The auxiliary supervisor does not print informational messages. Figure A-15 lists the auxiliary supervisor error messages.

Error number and message		Cause of error
S00	INVALID FUNCTION CODE	The auxiliary supervisor received an illegal parameter.
S01	XXXXX IS NOT IN LET/FLET	The core image loader is unable to find the name specified in this message in LET or FLET.
S02	XXXXX IS A DATA FILE	The specified name cannot be executed since it is a data file, not a program.

Figure A-15. Auxiliary supervisor error messages

MONITOR SYSTEM LIBRARY MAINLINE PROGRAMS MESSAGES AND ERROR MESSAGES

The following text describes the informational messages and error messages printed by the mainline programs that are a part of the monitor system library. These programs are described in Chapter 4.

IDENT Messages

At the end of execution of the IDENT program, the following message is printed:

```
PHYSICAL DRIVE        CART. ID
     YYYY               XXXX
```

YYYY is replaced with the physical drive number, beginning with 0000, and XXXX is replaced with actual cartridge IDs. One line is printed for each ready drive.

DISC Messages and Error Messages

When DISC is executed, the contents of the *ID control record are printed on the principal print device. Then, if errors occur, any of the following messages may be printed, depending on the errors:

Error message	Cause of error
CARTRIDGE XXXX INVALID ... LOGICAL 0 ID	The ID of the master cartridge (logical drive 0) is specified as a current ID on the *ID control record. XXXX is the ID of the master cartridge.
CARTRIDGE XXXX NEW LABEL IS INVALID	The new label XXXX is outside the range /0001 through /7FFF.
CARTRIDGE XXXX IS NOT AVAILABLE	A selected cartridge with the ID XXXX is not on the system or the selection of XXXX results in the definition of more than 5 LOGICAL drives.
CARTRIDGE XXXX IS DEFECTIVE	Sector @IDAD, or more than 3 cylinders, on the identified cartridge are defective (to identify the defective cylinders, initialize the cartridge with the stand-alone program DCIP).

At the end of reinitialization, the following is printed:

XXXXYYYY NOT DONE
or
XXXXYYYY COMPLETE

where
XXXX is the old (FID1) cartridge ID.
YYYY is the new (TID1) cartridge ID.

One of these messages is printed for each satellite cartridge that is reinitialized. A NOT DONE message is printed only if an error message has been printed.

ID Messages and Error Messages

At completion of the execution of the ID program, the following is printed:

FFFF TTTT NOT DONE
or
FFFF TTTT COMPLETE

where
FFFF is the FROM cartridge ID.
TTTT is the TO cartridge ID.

One of these messages is printed for each cartridge ID that is changed (maximum of 4). The NOT DONE message is printed when a selected cartridge is not found on the system.

COPY Messages and Error Messages

At completion of the copy program, one of the following messages is printed for each copy requested on the *ID control record:

FFFF TTTT NOT DONE
FFFF TTTT NOT PRES
FFFF TTTT NO. ERROR
FFFF TTTT COMPLETE

where
FFFF is the source cartridge ID.
TTTT is the object cartridge ID.
NOT PRES indicates that the cartridge with the requested ID is not on the system.
NO. ERROR indicates that the requested ID is not within the range /0001–/7FFF.

When at least one COMPLETE message is printed, all of the cartridges on the system are listed.

DLCIB Messages and Error Messages

When the CIB is deleted from a cartridge, the following message is printed at the completion of the DLCIB program:

```
CART  UA/FX  FPAD
XXXX  YYYY   NNNN
```

where

XXXX is the cartridge ID.
YYYY is the sector address of the user area.
NNNN is the file protect address.

If the CIB cannot be deleted,

```
XXXX ERROR
```

is printed. XXXX is the cartridge ID.

This error message is printed if:

- The cartridge ID specified in the *ID control record is not on the system.
- The cartridge ID specified in the *ID control record is not specified on the current JOB monitor control record.
- The specified cartridge is a system cartridge.
- The CIB is already deleted from the specified cartridge.
- The CIB on the specified cartridge is specified as system CIB by the current JOB monitor control record.

MODIF Messages and Error Messages

When execution of MODIF is completed successfully, the following messages are printed on the principal printer:

```
MODIF EXECUTION 0WXX
MODIF COMPLETED 0YZZ
```

where

WXX is the old version and modification number.
YZZ is the new version and modification number.

If an error is detected during execution of MODIF, an error message is printed in the following format:

```
ERROR# XXXX XXXX
```

where

XXXX represents hexadecimal numbers.

The system waits for an operator response. All MODIF errors and operator recovery procedures are listed in Figure A-16.

Figure A-16 (Part 1 of 3). MODIF error numbers

Error number	Description	Operator's switch option	Operator recovery procedure (Note that the instruction PRESS START, if not stated, is implied in each of the following procedures.)	Remarks	First hexadecimal number printed	Second hexadecimal number printed
1	Invalid patch control record (*MON or *SUB)	No switches on	Correct error and reread from corrected patch control record. (If the error has occurred on the first patch control record, restart the modification.)			
		Switch 0 on	Press START to call EXIT	This terminates modification.		
2	Checksum error on binary patch data record	No switches on	Rechecksum and reread from preceding patch control record. (If the error has occurred on the first patch control record, restart the modification.)		Amount of checksum difference	Number of binary records read after patch header (including record in error)
		Switch 0 on	Press START to call EXIT	This terminates modification. If word 2 is blank, the test for a valid checksum is not made.		
		Switch 15 on	Reread card in error (cards may be out of order).			
3	Invalid hex data record	No switches on	Correct error and reread from preceding patch control record.			
		Switch 0 on	Press START to call EXIT	This terminates modification.		
		Switch 15 on	Reread card in error.			
4	Modification level error in system modification update	No switches on	Correct error and reread from corrected patch control record.		Present version and modification level (from DCOM on disk)	Level of version and modification (from patch control record)
		Switch 0 on	Press START to call EXIT	This terminates modification.		

Error number	Description	Operator's switch option	Operator recovery procedure	Remarks	First hexadecimal number printed	Second hexadecimal number printed
5	New modification level lower than current level in system modification update	No switches on	Correct error and reread from corrected patch control record.		Present version and modification level (from DCOM on disk)	Level of version and modification (from patch control record)
		Switch 0 on	Press START to call EXIT	This terminates modification.		
		Switch 15 on	Press START to continue	Level is reduced and program continues.		
6	Monitor control record or // DEND card read before required number of patches read	No switches on	Press START to continue	New patch control record is read.	Number of patches not installed	
		Switch 0 on	Press START to call EXIT	This terminates modification.		
7	DCOM configuration indicators do not agree with SLET or required system I/O routine missing	Switch 0 on	Press START to call EXIT	This terminates modification.	Contents of ACCUMULATOR when error was detected	Address +2 from which error branch was executed
8	DUP control record errors (DELETES or STORES)	No switches on	Press START to continue		XXYY where XX is the number of DUP errors detected (see DUP error printout) and YY is the number of DUP control records not processed.	Number of DUP control records specified on *SUB patch control record
9	SLET ID not found	No switches on	Press START to continue		SLET ID in question	
A	Patch exceeds space allotted on disk for this phase	No switches on	Press START to continue		High core patch address	High core SLET address
B	// DEND card not found (patches completed but version and modification level in DCOM not updated)	No switches on	Press START to call EXIT	This terminates modification.		

Figure A-16 (Part 2 of 3). MODIF error numbers

Error number	Description	Operator's switch option	Operator recovery procedure	Remarks	First hexadecimal number printed	Second hexadecimal number printed
C	Modification level error in general temporary fix	No switches on	Press START to call EXIT	This terminates modification. Preceding patches in this MODIF JOB have been installed.	Present version and modification level (from DCOM on disk)	Version and modification level (from patch control record)
D	Modification level error in restricted temporary fix	No switches on	Press START to call EXIT	This terminates modification. Preceding patches in this MODIF JOB have been installed.	Present version and modification level (from DCOM on disk)	Version and modification level (from patch control record)
E	System modification update mixed with temporary fixes	No switches on	Press START to call EXIT	This terminates modification. Preceding patches in this MODIF JOB have been installed.		

Figure A-16 (Part 3 of 3). MODIF error numbers

MODSF Messages and Error Messages

All update requests read by MODSF are listed on the principal printer, along with an indication of the results of the requests. Upon successful completion of an update that does not expand a program:

MODIFICATIONS MADE

is printed after the list of requests. When an *END control record is read and the program is not expanded:

SUCCESSFUL COMPLETION

is printed after the *END control record.

When an update that expands a program is successfully completed:

MODIFICATIONS MADE IN WORKING STORAGE

is printed after the list of requests. When an *END control record is read after a successful update that expands a program:

(*DELETE/*STORE RECORDS MUST FOLLOW)

is printed after the *END control record.

When an error is detected by MODSF:

ERROR nn
PROGRAM WAS NOT MODIFIED

is printed after the list of requests (nn represents the error number). Any previous program for which the message:

MODIFICATIONS MADE

has been printed, have been successfully updated; the current program is not updated, and any succeeding programs are bypassed. A program is never partially updated by MODSF. The MODSF error codes that are printed in the error message are listed in Figure A-17.

Error number	Cause of error
01	MODSF cannot be run in a temporary job mode.
02	MODSF cannot be run with DUP suppressed.
03	First card is not *PRO.
04	Last card was encountered before *END card.
05	Monitor control record was encountered.
06	*Card neither *PRO nor *END.
07	Column which must be blank was not blank in patch control record.
08	Version/modification (columns 6 through 8) invalidly specified or omitted.
09	Version/modification (columns 6 through 8) does not match system cartridge.
10	Program name (columns 10 through 14) is invalid or omitted.
11	Number of patch data records (columns 16 through 19) is not a valid positive hexadecimal value.
12	Cartridge ID (columns 23 through 36) is not validly specified.
13	Cartridge specified (columns 23 through 26) is not online.
14	Program specified (columns 10 through 14) cannot be found on requested cartridge.
15	Name specified in columns 10 through 14 is a secondary entry point.
16	Name specified in columns 10 through 14 is a core-image program.
17	Name specified in columns 10 through 14 is a data file.
18	Addressing mode (column 21) is neither D nor P.
19	Invalid address is specified for verification (columns 28 through 31, 38 through 41, 48 through 51, 58 through 61).
20	Invalid value is specified for verification (columns 33 through 36, 43 through 46, 53 through 56, 63 through 66).
21	During verification, a nonmatch was detected.
22	Number of patch data records does not match number specified.
23	Patch address is a valid hexadecimal value.
24	Column in patch data record which must be blank was not blank.
25	In addressing mode P, relocation mode indicator is not A, R, L, or C.
26	Patch address is an invalid hexadecimal value.
27	Patch address is within BSS or area skipped by ORG.

Figure A-17 (Part 1 of 2). MODSF error codes

Error number	Cause of error
28	Attempt was made to change relocation mode of an LIBF.
29	Relocation mode of second word of LIBF is not A.
30	Attempt to patch in an LIBF where non-LIBF appears in program.
31	Program requiring expansion is not followed by *END patch control record.
32	More than 31 words are to be updated.
33	Insufficient working storage for expansion.
34	Address specified for verification beyond end of program or in area skipped by BSS or ORG.

Figure A-17 (Part 2 of 2). MODSF error codes

DFCNV Messages and Error Messages

Each DFCNV control record is printed on the principal printer as it is read. At the end of successful processing of the DFCNV control records, the following message is printed:

DISK DATA FILE CONVERSION COMPLETED

As errors are detected in DFCNV control records, diagnostic messages are printed. All diagnostic errors, except the warning messages, cause program termination. If an error is detected on the file description card, program termination is immediate; all other errors are diagnosed before program termination. All messages, except F10, are printed before data conversion begins. All DFCNV diagnostic error messages are listed in Figure A-18.

Error number and message		Cause of error
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 Y = O, output file error
F03	FILE SIZE INVALID–Y	1 File size calculated from File Description data exceeds actual file size
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 (column 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 Specification card precedes / * card
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 zeros or nines in record YYYYY.

1 Program termination immediate
2 Warning only
3 No columns indication

Figure A-18. DFCNV error messages

Appendix B. Monitor System Error Wait Codes

System loader, FORTRAN I/O and RPG object program errors cause the system to wait at $PRET. At the wait, bits 2 and 3 of the OPERATION REGISTER are on. FORTRAN I/O errors are identified by the Fxxx code in the ACCUMULATOR. RPG object program errors are identified by the Cxxx code in the ACCUMULATOR. A $PRET wait also occurs when a system I/O device is required but is not ready. The codes for all of these errors and the errors detected during the cold start program are described in this appendix.

COLD START PROGRAM ERROR WAITS

The following are the absolute addresses that are displayed in INSTRUCTION ADDRESS on the console when errors are detected during the cold start program:

INSTRUCTION ADDRESS register display	Explanation
/001F	—Invalid disk drive number in console entry switches
	—Indicated disk drive not ready
/0046	—Power is unsafe in the disk drive; turn drive off and on for a retry
	—Disk read error
	—Waiting for interrupt from seek operation
/0048	—Waiting for interrupt from reading sector @IDAD

Note. When any of these errors occur, perform another cold start.

ISS SUBROUTINE PREOPERATIVE ERROR WAITS

A preoperative error is an error condition that is detected before an I/O operation is started. The following preoperative error conditions cause the monitor system to wait at $PRET, $PST1, $PST2, $PST3, or $PST4:

- Device not ready
- Error check in device
- Illegal parameter or illegal specification in an I/O area

When a preoperative error condition is detected:

- The address of $PRET+2 is displayed in the INSTRUCTION ADDRESS on the console.
- An error code represented by 4 hexadecimal digits is displayed in the console ACCUMULATOR, where digit 1 identifies the ISS called:

  1–CARDx or PNCHx
  2–TYPEx or WRTYx
  3–PAPTx
  4–READx
  5–DISKx
  6–PRNT1, PRNT2 or PRNTZ
  7–PLOT1, PLOTx
  8–SCATx
  9–PRNT3 or PRNZ
  A–OMPR1

  Digits 2 and 3 are not used (zero).

  Digit 4 identifies the error, where

  0–Device not ready
  1–Illegal parameter or illegal specification in I/O area

- $PRET contains the address of the call in question. The ISS is set up to attempt initiation of the operation a second time if the call is reexecuted. Pressing console PROGRAM START returns control to the ISS for a reexecution of the call.

When a preoperative error wait occurs, you can do one of the following:

- Correct the error condition if possible and press PROGRAM START
- Note the contents of the ACCUMULATOR and location $PRET, dump core storage, and proceed with the next job

All ISS subroutine error waits are listed and described in Figure B-1.

ACCUMULATOR display	Device causing wait	Cause of wait
/1000	1442 Card Read/Punch or 1442 Card Punch	Device is not ready, or last card indicator is on or read.
/1001		Illegal device, device is not in system, illegal function, word count is over +80, or word count is zero or negative.
/100F		This wait occurs in a DUP operation after a D112 error message has been printed.
/2000	Keyboard/Console Printer	Device is not ready.
/2001		Device is not in system, illegal function, or word count is zero or negative.
/2002		Keyboard input is expected (TYPEZ only).
/3000	1134/1055 Paper Tape Reader/Punch	Device is not ready.
/3001		Illegal device, illegal function, word count is zero or negative, or illegal check digit.
/4000	2501 Card Reader	Device is not ready.
/4001		Illegal function, word count is over +80, or word count is zero or negative.
/5000	Disk	Device is not ready. Make device ready and press PROGRAM START.
/5001		Illegal device, device is not in system, invalid function, attempt to write in file protected area, word count is zero or negative or starting address is over +1599. Operation is retried if PROGRAM START is pressed (DISK1 and DISKN only).
/5002		Write select/power unsafe. Turn the cartridge off, then on again, to reset the error condition. DISKZ: If PROGRAM START is pressed, the operation is retried. DISKN or DISK1: If the program is waiting at $PRET and PROGRAM START is pressed, the operation is retried. If the program is waiting at $PST2 and PROGRAM START is pressed, the program goes to EXIT. *Note.* If an interrupt on level 0 or 1 occurs when the program is waiting at $PST2. the program will go to EXIT.
/5003		Read/write/seek failure remaining after 16 attempts, or disk overflow. Error occurred during the processing of a monitor control record (DISKZ only). If a code is also displayed in the ACCUMULATOR EXTENSION, bits 0 through 3 indicate the logical drive number, and bits 4 through 15 indicate the working-storage address, except for disk overflow. Press PROGRAM START; the program is retried 16 times.
/5004		Same as /5003 (DISK1 and DISKN only), or an attempt was made to cold start from a system cartridge when an uninitialized cartridge is on a ready drive. A cold start cannot be performed until the disk is initialized or is turned off. If a code is also displayed in the ACCUMULATOR EXTENSION, bits 0 through 3 indicate the logical drive number, and bits 4 through 15 indicate the working-storage sector address plus one.
/6000	1132 Printer	Device is not ready or end of forms.
/6001		Illegal function, word count is over +60, or word count is zero or negative.

Figure B-1 (Part 1 of 2). ISS subroutine WAITs

ACCUMULATOR display	Device causing wait	Cause of wait
/7000	1627 Plotter	Device is not ready. Ready the device and press PROGRAM START.
/7001		Illegal function, or word count is zero or negative. If PROGRAM START is pressed, the operation is retried (PLOT1 only).
/8001	SCA (STR mode) (SCAT1)	Invalid function code or invalid word count.
/8002		Receive or transmit operation is not completed.
/8003		Failure to establish synchronization before attempting to perform some transmit or receive operation, or attempting to receive before receiving INQ sequence.
/8001	SCA (BSC mode) (SCAT2 or SCAT3)	Invalid function code, word count, or subfunction code.
/8002		Invalid start characters in the I/O area for a transmit operation.
/8003		Invalid number of identification characters for an identification specification operation (SCAT2 only).
/9000	1403 Printer	Device is not ready or end of forms. Make device ready and press PROGRAM START.
/9001		Illegal function, word count is over +60, zero or negative. To retry operation, press PROGRAM START (PRNT3 only).
/9002		Parity check, scan check, or ring check. Reset check and press PROGRAM START. The operation is not retried (PRNZ only).
/A000	1231 Optical Mark Page Reader	Device is not ready.
/A001		Illegal function.
/A002		Feed check, last document is processed. Clear jam, make ready, do not refeed.
/A003		Feed check, last document is not processed. Clear jam, make ready, refeed last document. If error was caused by double feed, refeed both documents.

Figure B-1 (Part 2 of 2). ISS subroutine WAITs

I/O DEVICE SUBROUTINE ERRORS

The error parameters of the card read and punch, console printer, and paper tape I/O subroutines are discussed in the following text. (The special function keys of the console keyboard are discussed in Chapter 7.)

1442 Card Subroutine Errors

CARDZ, CARD0, PNCHZ, and PNCH0 do not have an error parameter. If an error is detected during processing of an operation-complete interrupt, the subroutine traps to $PST4 with interrupt level 4 on. You can reinitiate the operation by readying the 1442, and pressing PROGRAM START on the console keyboard.

CARD1 and PNCH1 do have an error parameter. If an error is detected during processing of an operation-complete interrupt, your program can elect to terminate (clear the subroutine busy indicator, and turn off the interrupt level) or to retry the operation. A retry consists of waiting at $PST4 with interrupt level 4 on, and then reinitiating the function.

A read or feed function that is requested after the last card has been detected causes the last card to be ejected, and a trap to $PRET occurs. A punch function punches and then ejects the last card with a normal exit.

If a 1442 device error occurs, the 1442 becomes not ready until you intervene. Unless the wait is caused by a stacker full (none of the 1442 error indicators are on) or chip box indication, the 1442 card path must be cleared before proceeding. The 1442 error indicators and the position of the cards in the feed path are used to determine which cards must be placed back in the hopper.

For the card subroutines, a retry consists of positioning the cards (skipping the first card in the hopper, if necessary, on a read or feed operation) and reinitiating the function whenever the card reader is readied.

Card read error conditions are described in Figure B-2. Read errors do not apply to the 1442, Model 5.

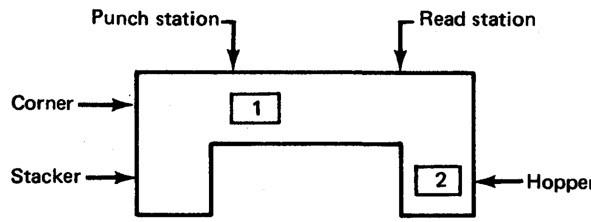
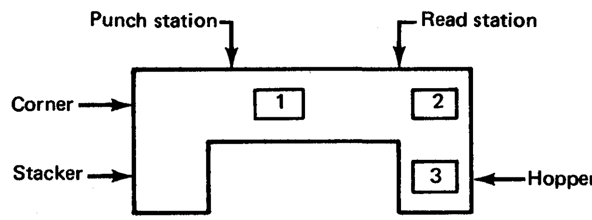
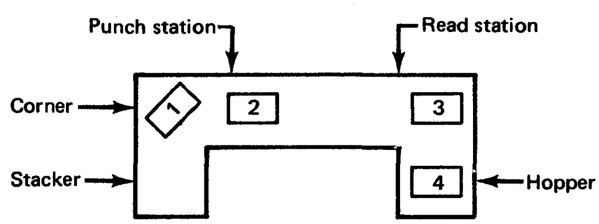
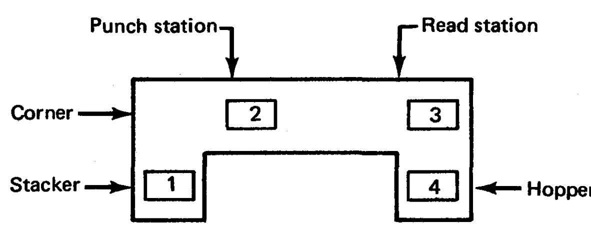
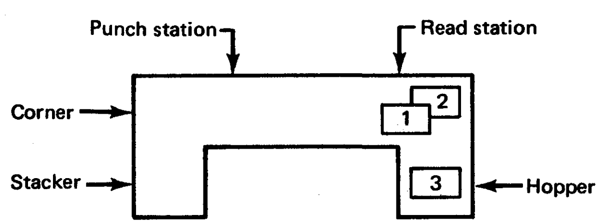
Error indicator on	Error condition	Card positions after error	Your response
HOPR	*Hopper misfeed* indicates that card 2 failed to pass properly from the hopper to the read station during the card 1 feed cycle.		When the program halts, press reader NPRO to eject card 1, place card 1 in hopper before card 2, and ready the 1442.
PUNCH STA	*Feed check (punch station)* indicates that card 1 is improperly positioned in the punch station at the completion of its feed cycle.		When the program halts, empty the hopper, clear the 1442 card path, place cards 1 and 2 in the hopper before card 3, and ready the 1442.
TRANS	*Transport* indicates that card 1 has jammed in the stacker during the feed cycle for card 2.		When the program halts, empty the hopper, clear the 1442 card path, place cards 2 and 3 in the hopper before card 4, and ready the 1442.
FEED CLU	*Feed cycle* indicates that the 1442 took an unrequested feed cycle and, therefore, cards 1, 2, and 3 are each one station ahead in the 1442 card path than they should be.		When the program halts, empty the hopper, press NPRO to eject cards 2 and 3, place cards 1, 2, and 3 in the hopper before card 4, and ready the 1442.
READ STA	*Feed check (read station)* indicates that card 1 failed to eject from the read station during its feed cycle.		When the program halts, empty the hopper, clear the 1442 card path, place cards 1 and 2 in the hopper before card 3, and ready the 1442.

Figure B-2 (Part 1 of 2). 1442 Card Read errors

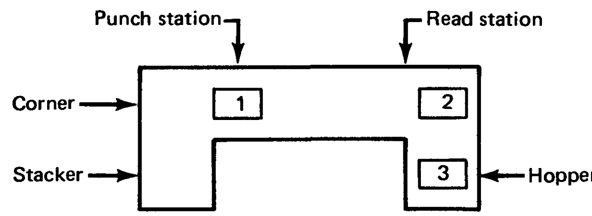
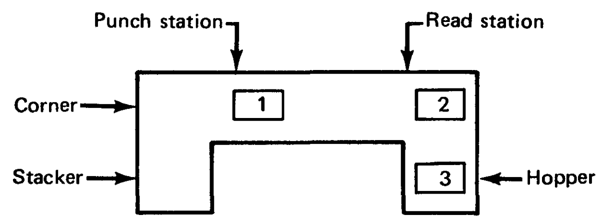
Error indicator on	Error condition	Card positions after error	Your response
READ REG	*Read registration* indicates incorrect card registration or a difference between the first and second reading of a column.		See error condition "Feed check (punch station)." Repeated failures of this type might indicate a machine malfunction.
PUNCH	*Punch check* indicates an error in output punching.		When the program halts, empty the hopper, check the card position, and press NPRO to clear the 1442 card path. If necessary, correct card 1 to pre-punched state. Place card 1 and card 2 in the hopper before card 3 and ready the 1442.

Figure B-2 (Part 2 of 2). 1442 Card Read errors

2501 Card Subroutine Errors

READZ and READ0 do not have an error parameter. If an error is detected during processing of an operation-complete interrupt, the subroutine traps to $PST4, with interrupt level 4 on. You reinitiate the operation by making the 2501 ready and pressing PROGRAM START on the console keyboard.

READ1 does have an error parameter. If an error is detected during processing of an operation-complete interrupt, your program can elect to terminate (clear the subroutine busy indicator and turn off the interrupt level), or to retry the operation. A retry consists of waiting at $PST4 with interrupt level 4 on until the 2501 is readied, and then reinitiating the function.

A read function requested after the last card has been detected causes a trap to $PRET.

If a 2501 device error occurs, the 2501 becomes not ready until the operator intervenes. Unless the stop is caused by a stacker full or cover open (ATTENTION), the 2501 card path must be cleared before proceeding. The 2501 error indicators and the position of the cards in the feed path should be used to determine the cards to be placed back in the hopper.

For the card subroutines, a retry consists of positioning the cards (skipping the first card in the hopper, if necessary) and reinitiating the read function whenever the card reader is readied.

2501 feed check error

A 2501 feed check indicates that a card has failed to feed from the hopper or that a card is mispositioned in the feed path.

To correct this error, empty the hopper and press NPRO when the program waits at $PST4. If a card has failed to feed from the hopper, place the last card in the stacker ahead of the deck remaining to be read. Place this deck in the hopper, and ready the reader.

If a card has been mispositioned in the feed path, place the last 2 cards in the stacker ahead of the deck remaining to be read. Place this deck in the hopper, and ready the reader.

2501 read check error

A read check indicates incorrect card registration or a difference between the first and second reading of a column. To correct this error when the program traps to $PST4, empty the hopper, press NPRO, place the last 2 cards in the stacker ahead of the deck remaining to be read, place this deck back in the hopper, and ready the reader.

Console Printer Subroutine Errors

If the carrier attempts to print beyond the manually positioned margins, a carrier restore (independent of the program) occurs.

When TYPE0 and WRTY0 are being used, printing begins wherever the carrier is positioned as a result of a previous print operation. TYPEZ and WRTZ provide automatic carriage return before each operation.

If the console printer indicates a not-ready condition after printing begins, the subroutines trap to $PST4 with interrupt level 4 on. After you make the console printer ready, pressing PROGRAM START causes the operation to be reinitiated.

The special function keys of the console keyboard are discussed in Chapter 7.

Paper Tape Subroutine Errors

If the reader or punch becomes not ready during an I/O operation, the subroutines exit to your program via the error parameter. You can request the subroutine to terminate (clear device busy on the interrupt level) or to wait at $PST4 for operator intervention (interrupt level 4 on).

If the 1134/1055 indicates a not-ready condition after an operation has been initiated, the subroutines trap to $PST4 with interrupt level 4 on. The operation is reinitiated by making the device ready, and pressing PROGRAM START on the console.

Card Core Image Loader Wait Code

If any kind of card reader or checksum error occurs during the loading of a card image format program into core storage, the core image loader waits at location /0020 with the number of the card to be loaded displayed in the ACCUMULATOR on the console display panel.

To continue processing:

1. Press NPRO on the card reader.
2. Place all the cards, beginning with the one whose number is displayed in the ACCUMULATOR, in the card hopper, and press START on the card reader.
3. Press PROGRAM START on the console keyboard

PAPER TAPE UTILITY PROGRAM (PTUTL) ERROR WAIT CODES

When the paper tape reader or punch becomes not ready during processing, the system waits with an error code displayed in the console ACCUMULATOR. The PTUTL error wait codes are described in Figure B-3.

ACCUMULATOR display	Error condition	Your response
/3005	Paper tape reader not ready	Ready the reader if additional tape is to be read; set the console entry switches as desired, and press PROGRAM START on the console keyboard.
/3004	Paper tape punch not ready	Ready the paper tape punch and press console PROGRAM START. To repunch the record that was being processed when the not-ready condition occurred, set console entry switches 1 and 2 off (to prevent another record from being read), set switches 3 and 14 on (punch record and wait with /3333 in the ACCUMULATOR), and press PROGRAM START. After the record is punched, return the console entry switches to the original configuration, and press PROGRAM START.

Figure B-3. PTUTL error wait codes

FORTRAN I/O WAIT CODES

When a FORTRAN I/O error occurs, the system waits at $PRET with Fxxx displayed in the console ACCUMULATOR. The program should be corrected, and the execution restarted.

Figure B-4 describes the FORTRAN I/O error waits.

ACCUMULATOR display	Cause of error	Type of FORTRAN I/O	System action if you press PROGRAM START
F000	No *IOCS card appeared with the mainline program and I/O was attempted in a subroutine.	SFIO [1]	CALL EXIT
F001	Logical unit defined incorrectly, or no *IOCS control record for specified I/O device.	SFIO [1]	Execution continues with next FORTRAN statement.
F002	Requested record exceeds allocated buffer size.	SFIO [1]	All the variables in the I/O list, following the one which has the erroneous format specification, will also be treated as errors.
F003	Illegal character encountered in input record.	SFIO [1]	The variables connected with the erroneous data fields will contain zeros. Other variables in the I/O list connected to fields in the same data record will be handled as usual.
F004	Exponent too large or too small in input field.	SFIO [1]	The variables connected with the erroneous data fields will contain zeros. Other variables in the I/O list connected to fields in the same data record will be handled as usual.
F005	More than one exponent field encountered in input field.	SFIO [1]	The variables connected with the erroneous data fields will contain zeros. Other variables in the I/O list connected to fields in the same data record will be handled as usual.
F006	More than one sign encountered in input field.	SFIO [1]	The variables connected with the erroneous data fields will contain zeros. Other variables in the I/O list connected to fields in the same data record will be handled as usual.
F007	More than one decimal point encountered in input field.	SFIO [1]	The variables connected with the erroneous data fields will contain zeros. Other variables in the I/O list connected to fields in the same data record will be handled as usual.
F008	Read of output-only device, or write of input-only device.	SFIO [1]	Execution continues with next FORTRAN statement.
F009	Read variable transmitted with an I format specification or integer variable transmitted with an E or F format specification.	SFIO [1]	The actual format specifications will be effectuated.

Figure B-4 (Part 1 of 2). FORTRAN I/O errors

ACCUMULATOR display	Cause of error	Type of FORTRAN I/O	System action if you press PROGRAM START
F020	Illegal unit reference.	UFIO [2]	UFIO not updated.
F021	Read list exceeds length of write list.	UFIO [2]	UFIO updated.
F022	Record not existing for read list element.	UFIO [2]	UFIO updated.
F023	Maximum length of $$$$$ area on the disk has been exceeded. This error is unrecoverable and results in a call exit.	UFIO [2]	CALL EXIT
F024	UFIO has not been initialized: there is no *IOCS (UDISK) record in the mainline program.	UFIO [2]	CALL EXIT
F100	File not defined by DEFINE FILE statement.	SDFIO [3]	CALL EXIT
F101	File record number too large, equal to zero, or negative. This error may be caused by attempting to access the end of a working storage file that has been truncated by the core load builder.	SDFIO [3]	CALL EXIT
F103	Disk FIO has not been initialized; there is no *IOCS (DISK) record in the mainline program.	SDFIO [3]	CALL EXIT
F105	The length of a list element (2 or 3 words, depending on the precision) exceeds the record length (1 or 2 words) defined in a DEFINE FILE statement.	SDFIO [3]	CALL EXIT
F107	An attempt has been made to read or write at an invalid sector address. This error occurs if a core image program with working storage files is executed on a system with too small working storage.	SDFIO [3]	CALL EXIT
F10A	Subscripting has destroyed the define file table and/or core image header. This occurs when a subscript exceeds the specification in a DIMENSION.	SDFIO [3]	CALL EXIT

[1] Standard FORTRAN I/O

[2] Unformatted FORTRAN I/O

[3] Standard disk FORTRAN I/O

Figure B-4 (Part 2 of 2). FORTRAN I/O errors

RPG OBJECT PROGRAM WAIT CODES

RPG object program errors cause the system to wait with Cxxx displayed in the console ACCUMULATOR. All RPG object program wait codes are described in Figure B-5.

The object program errors can be divided into 2 categories, disk I/O and general. The wait codes for disk I/O errors are in the range C000 to C05F. All others are between C100 and CFFF. Some of the disk I/O errors should not occur during normal processing. However, if incorrect object code is generated or if the object program is erroneously modified at object time, these disk I/O errors may occur. These error codes are identified with an asterisk to the right of the Cxxx number in Figure B-5.

When an RPG object program error occurs, the operator must take specific action. Generally, this means terminating the job by turning all console entry switches off and pressing PROGRAM START on the console keyboard. Certain errors, however, allow the operator to ignore the error or retry the operation by setting console entry switch 15 on, all others off, and pressing console PROGRAM START. In the case of a retry, the card in error must be placed back in the hopper before continuing. An incorrect operator action causes the error wait to reoccur.

ACCUMULATOR display	Type of processing	Meaning	Your response	Console entry switch settings
C000	Sequential file: random processing	Record number is not within the assigned limits of the file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
			If chaining, correct the card and reinsert it in the input stream, or bypass the chaining record and read the next card.	15 on, all others off. Press console START.
C001*	Sequential file: random processing	Record size is not within limits (maximum 640 characters).	Terminate the job.	All off. Press console START.
C002*	Sequential file: random processing	Records per sector is not maximum.	Terminate the job.	All off. Press console START.
C003	Sequential file: random processing	No record was found. The record number is not a positive number.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
			If chaining, correct the card and reinsert it in the input stream, or bypass the chaining record and read the next card.	15 on, all others off. Press console START.
C004	Sequential file: random processing	Write before read on an update file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C005*	Sequential file: random processing	File was accessed when not open.	Terminate the job.	All off. Press console START.
C006*	Sequential file: random processing	I/O buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C010	Sequential file: sequential processing	Disk file is full.	Terminate the job.	All off. Press console START.
C011*	Sequential file: sequential processing	A write is requested on an input file.	Terminate the job.	All off. Press console START.
C012*	Sequential file: sequential processing	A read is requested on an output file.	Terminate the job.	All off. Press console START.

Figure B-5 (Part 1 of 5). RPG Object Program error messages

ACCUMULATOR display	Type of processing	Meaning	Your response	Console entry switch settings
C013*	Sequential file: sequential processing	Record size is not within limits (maximum 640 characters).	Terminate the job.	All off. Press console START.
C014*	Sequential file: sequential processing	Number of records per sector is not maximum.	Terminate the job.	All off. Press console START.
C015*	Sequential file: sequential processing	File was accessed when not open.	Terminate the job.	All off. Press console START.
C016*	Sequential file: sequential processing	I/O buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C017	Sequential file: sequential processing	Write before read requested on an update file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C020	ISAM load processing	Invalid type of processing on load function.	Terminate the job.	All off. Press console START.
C021*	ISAM load processing	One of the following:		
		Record size not within limits (maximum 636 characters).	Terminate the job.	All off. Press console START.
		Number of records per sector is not maximum.		
C022*	ISAM load processing	Key length is greater than maximum.	Terminate the job.	All off. Press console START.
C023*	ISAM load processing	Index entry length is not same as length computed from key length.	Terminate the job.	All off. Press console START.
C024*	ISAM load processing	Number of index entries per sector does not permit maximum number of records per sector.	Terminate the job.	All off. Press console START.
C025	ISAM load processing	Prime data area is full.	Terminate the job.	All off. Press console START.
C026	ISAM load processing	Index area is full.	Terminate the job.	All off. Press console START.
C027*	ISAM load processing	File was accessed when not open.	Terminate the job.	All off. Press console START.
C028*	ISAM load processing	Index buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.

Figure B-5 (Part 2 of 5). RPG Object Program error messages

ACCUMULATOR display	Type of processing	Meaning	Your response	Console entry switch settings
C029*	ISAM load processing	Prime data buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C02A	ISAM load processing	Input record is out of sequence.	One of the following:	
			Terminate the job.	All off. Press console START.
			Correct the card and reinsert it in the input stream, or bypass the record by reading another card.	15 on, all others off. Press console START.
C030*	ISAM add processing	Invalid type of processing on add function.	Terminate the job.	All off. Press console START.
C031*	ISAM add processing	File was accessed when not open.	Terminate the job.	All off. Press console START.
C032	ISAM add processing	Key length for this job is not same as key length in file.	Terminate the job.	All off. Press console START.
C033	ISAM add processing	Record length for this job is not same as record length in file.	Terminate the job.	All off. Press console START.
C034	ISAM add processing	Attempt was made to add record already on file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C035	ISAM add processing	Overflow area is full. The file must be resequenced, or the data area must be made larger before another add run can be made.	Terminate the job.	All off. Press console START.
C036*	ISAM add processing	Index buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C040*	ISAM file: sequential processing	Invalid type of processing on retrieve or update function.	Terminate the job.	All off. Press console START.
C041*	ISAM file: sequential processing	Index buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C042*	ISAM file: sequential processing	Prime data buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C043	ISAM file: sequential processing	Key length for this job is not same as key length in file.	Terminate the job.	All off. Press console START.

Figure B-5 (Part 3 of 5). RPG Object Program error messages

ACCUMULATOR display	Type of processing	Meaning	Your response	Console entry switch settings
C044	ISAM file: sequential processing	Record length for this job is not same as record length in file.	Terminate the job.	All off. Press console START.
C045*	ISAM file: sequential processing	File accessed when not open.	Terminate the job.	All off. Press console START.
C046	ISAM file: sequential processing	Write before read requested on update file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C050*	ISAM file: random processing	Invalid type of processing on retrieve or update function.	Terminate the job.	All off. Press console START.
C051*	ISAM file: random processing	Index buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C052*	ISAM file: random processing	Prime data buffer is not on even-word boundary.	Terminate the job.	All off. Press console START.
C053	ISAM file: random processing	Key length for this job is not same as key length in file.	Terminate the job.	All off. Press console START.
C054	ISAM file: random processing	Record length for this job is not same as record length in file.	Terminate the job.	All off. Press console START.
C055*	ISAM file: random processing	File accessed when not open.	Terminate the job.	All off. Press console START.
C056	ISAM file: random processing	Write before read requested on update.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C057	ISAM file: random processing	Record not on file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C111		Numeric records or matching fields out of sequence, or record is an undefined type.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.

Figure B-5 (Part 4 of 5). RPG Object Program error messages

ACCUMULATOR display	Type of processing	Meaning	Your response	Console entry switch settings
C12n		Halt switch set by object program (n = 1-9)	One of the following:	
			Terminate the job.	All off. Press console START.
			Set the halt switches off and continue processing.	15 on, all others off. Press console START.
C400		Write before read requested on combined file.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C430		Attempt to divide by zero.	One of the following:	
			Terminate the job.	All off. Press console START.
			Continue processing. The quotient will be set to zero.	15 on, all others off. Press console START.
C450		Results of multiply over 14 positions.	One of the following:	
			Terminate the job.	All off. Press console START.
			Continue processing. The result is set to zero.	15 on, all others off. Press console START.
C500		Monitor control card is read while punching on the 1442 Reader/ Punch.	One of the following:	
			Terminate the job.	All off. Press console START.
			Bypass the record and continue processing.	15 on, all others off. Press console START.
C998		Table fields are out of sequence.	Terminate the job.	All off. Press console START.

Figure B-5 (Part 5 of 5). RPG Object Program error messages

Appendix C. Monitor System Library Listing

System library programs	Names	Type and subtype	Subroutines required	ID field (73–75)
MAINLINES				
Disk Maintenance Programs				
Disk initialization	DISC	2, None	SYSUP, RDREC, DISKZ	U6C
Print cartridge ID	IDENT	2, None	CALPR, DISKZ	U6F
Change cartridge ID	ID	2, None	RDREC, CALPR, DISKZ	U6G
Disk copy	COPY	2, None	RDREC, DISKZ	U6B
Write sector addresses in WS	ADRWS (cannot be called)	2, None	Linked from DUP DWADR	U6A
Delete CIB	DLCIB	2, None	RDREC, DISKZ	U6D
Dump system location Equivalence table	DSLET	2, None	FSLEN, DISKZ	U6E
Library maintenance	MODSF	2, None	DISKZ	U6I
System maintenance	MODIF	2, None	DISKZ	U6H
Disk data file conversion [1]	DFCNV	2, None	DISK1, ELD, FLD, NORM	W1L
Paper Tape Utility				
Keyboard or 1134 input and/or console printer or 1055 output	PTUTL	2, None	PAPHL, PAPPR, PAPT1, TYPE0	U6J
SUBROUTINES				
Utility Calls				
Selective dump on console printer	DMTD0, DMTX0	4, 0	WRTY0	U5B
Selective dump on 1132 printer	DMPD1, DMPX1	4, 0	PRNT1	U5C
Dump 80	DMP80	4, 0	None	U5A
Update DCOM	SYSUP	4, 0	FSLEN, FSYSU	U5E
Call system print	CALPR	4, 0	FSLEN	U7A
Read *ID record	RDREC	4, 0	FSLEN	U7C
Fetch phase IDs or fetch system subroutine	FSLEN, FSYSU	4, 0	DISKZ	U7B
Dummy log subroutine for SCA subroutines	IOLOG/CPLOG	4, 0	None	
Common FORTRAN Calls				
Test data entry switches	DATSW	4, 8	None	T3A
Divide check test	DVCHK	4, 8	None	T3B
Functional error test	FCTST	4, 8	None	T3C
Overflow test	OVERF	4, 8	None	T3E
Selective dump	PDUMP	4, 0	SFIO, SIOAI, SIOAF, SWRT, SCOMP	T3F
Sense light control and test	SLITE, SLITT	4, 8	None	T3G
FORTRAN trace stop	TSTOP	4, 8	TSET	T3H
FORTRAN trace start	TSTRT	4, 8	TSET	T3I
Integer transfer of sign	ISIGN	4, 8	None	T3D

[1] Not distributed to papertape users.

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
Extended Arithmetic/Function Calls				
Extended precision hyperbolic tangent	ETANH, ETNH	4, 8	EEXP, EADD, EDIV, EGETP, ELD/ESTO	S2I
Extended precision A**B function	EAXB, EAXBX	4, 8	EEXP, ELN, EMPY	S2C
Extended precision natural logarithm	ELN, EALOG	4, 8	XMD, EADD, EMPY, EDIV, NORM, EGETP	S2E
Extended precision exponential	EEXP, EXPN	4, 8	XMD, FARC, EGETP	S2D
Extended precision square root	ESQR, ESQRT	4, 8	EADD, EMPY, EDIV, EGETP, ELD/ESTO	S2H
Extended precision sine-cosine	ESIN, ESINE, ECOS, ECOSN	4, 8	EADD, EMPY, NORM, XMD, EGETP	S2G
Extended precision arctangent	EATN, EATAN	4, 8	EADD, EMPY, EDIV, XMD, EGETP, NORM	S2B
Extended precision absolute value function	EABS, EAVL	4, 8	EGETP	S2A
FORTRAN Sign Transfer Calls				
Extended precision transfer of sign	ESIGN	4, 8	ESUB, ELD	S2F
Standard precision transfer of sign	FSIGN	4, 8	FSUB, FLD	R2F
Standard Arithmetic/Function Calls				
Standard precision hyperbolic tangent	FTANH, FTNH	4, 8	FEXP, FADD, FDIV, FGETP, FLD/FSTO	R2I
Standard precision A**B function	FAXB, FAXBX	4, 8	FEXP, FLN, EMPY	R2C
Standard precision natural logarithm	FLN, FALOG	4, 8	FSTO, XMDS, FADD, EMPY, FDIV, NORM, FGETP	R2E
Standard precision exponential	FEXP, FXPN	4, 8	XMDS, FARC, FGETP	R2D
Standard precision square root	FSQR, FSQRT	4, 8	FADD, FMPY, FDIV, FGETP, FLD/FSTO	R2H
Standard precision sine-cosine	FSIN, FSINE, FCOS, FCOSN	4, 8	FADD, EMPY, NORM, XMDS, FSTO, FGETP	R2G
Standard precision arctangent	FATN, FATAN	4, 8	FADD, FMPY, FDIV, XMDS, FSTO, FGETP	R2B
Standard precision absolute value function	FABS, FAVL	4, 8	FGETP	R2A
Common Arithmetic/Function Calls				
Fixed point (fractional) square root	XSQR	4, 8	None	T1C
Integer absolute function	IABS	4, 8	None	T1B
Floating binary/EBC decimal conversions	FBTD (BIN. TO DEC.), FDTB (DEC. TO BIN.)	4, 0	None	T1A

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
Flipper for LOCAL/SOCAL Subprograms				
	FLIPR	4, 0	DISKZ, DISK1, or DISKN	U5D
FORTRAN Trace Subroutines				
Extended floating variable trace	SEAR, SEARX	3, 0	ESTO, TTEST, SWRT, SIOF, SCOMP	S2J
Fixed variable trace	SIAR, SIARX	3, 0	TTEST, SWRT, SIOI, SCOMP	T6B
Standard floating IF trace	SFIF	3, 0	FSTO, TTEST, SWRT, SIOF, SCOMP	R2K
Extended floating IF trace	SEIF	3, 0	FSTO, TTEST, SWRT, SIOF, SCOMP	S2K
Fixed IF trace	SIIF	3, 0	TTEST, SWRT, SIOI, SCOMP	T6C
Standard floating variable trace	SFAR, SFARX	3, 0	FSTO, TTEST, SWRT, SIOF, SCOMP	R2J
GO TO trace	SGOTO	3, 0	TTEST, SWRT, SIOI, SCOMP	T6A
Nondisk FORTRAN Format I/O				
FORTRAN format subroutine	SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX	3, 3	FLOAT, IFIX, ELD/ESTO or FLD/FSTO, PAUSE	T4C
FORTRAN Find Subroutines				
	SDFND	3, 1	DISKZ, DISK1, or DISKN	T4B
Disk FORTRAN I/O				
	SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDF, SDI, SDIX, SDFX, SDAI	3, 1	DISKZ, DISK1, or DISKN, PAUSE	T4A
Unformatted FORTRAN Disk I/O				
	UFIO, URED, UWRT, UIOI, UIOF, UIOAI, UIOAF, UIOFX, UIOIX, UCOMP, BCKSP, EOF, REWND	3, 1	DISKZ, DISK1, or DISKN, PAUSE	T4D
FORTRAN Common LIBFs				
FORTRAN pause	PAUSE	3, 0	None	T2A
FORTRAN stop	STOP	3, 2	None	T2B
FORTRAN subscript displacement calculation	SUBSC	3, 0	None	T2D
FORTRAN subroutine initialization	SUBIN	3, 0	None	T2C
FORTRAN trace test and set	TTEST, TSET	3, 0	None	T2E

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
FORTRAN I/O and Conversion Subroutines				
FORTRAN 1442 input/output subroutine	CARDZ	5, 3	HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04	T5A
FORTRAN 1442 output subroutine	PNCHZ	5, 3	HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04	T5G
FORTRAN 2501 input subroutine	READZ	5, 3	HOLEZ, GETAD, EBCTB, HOLTB, ILS04	T5J
Disk I/O routine (part of supervisor)	DISKZ	–	ILS02	–––
FORTRAN paper tape subroutine	PAPTZ	5, 3	ILS04	T5F
FORTRAN 1132 printer subroutine	PRNTZ	5, 3	ILS01	T5H
Call to PRNTZ to call to PRNT2 conversion	PRTZ2	5, 3	PRNT2, ILS01	WIK
FORTRAN 1403 printer subroutine	PRNZ	5, 3	ILS04	T5I
FORTRAN keyboard-typewriter subroutine	TYPEZ	5, 3	GETAD, EBCTB, HOLEZ, ILS04	T5K
FORTRAN typewriter subroutine	WRTYZ	5, 3	GETAD, EBCTB, ILS04	T5L
FORTRAN 1627 plotter subroutine	PLOTX	5, 0	ILS03	V1L
FORTRAN hollerith to EBCDIC conversion	HOLEZ	3, 3	GETAD, EBCTB, HOLTB, PAUSE	T5D
FORTRAN get address routine	GETAD	3, 3	None	T5C
FORTRAN EBCDIC table	EBCTB	3, 3	None	T5B
FORTRAN hollerith table	HOLTB	3, 3	None	T5E
FORTRAN multiple terminal communications adapter (MTCA) call interface	MTCAZ	4, 0	MTCA0	W5C
Extended Arithmetic/Function LIBFs				
Extended precision get parameter subroutine	EGETP	3, 2	ELD	S1E
Extended precision A**I function	EAXI, EAXIX	3, 2	ELD/ESTO, EMPY, EDVR	S1B
Extended precision divide reverse	EDVR, EDVRX	3, 2	ELD/ESTO, EDIV	S1D
Extended precision float divide	EDIV, EDIVX	3, 2	XDD, FARC	S1C
Extended precision float multiply	EMPY, EMPYX	3, 2	XMD, FARC	S1G
Extended precision subtract reverse	ESBR, EXBRX	3, 2	EADD	S1H
Extended add-subtract	EADD, ESUB, EADDX, ESUBX	3, 2	FARC, NORM	S1A
Extended load-store	ELD, ELDX, ESTO, ESTOX	3, 0	None	S1F
Standard Arithmetic/Function LIBFs				
Standard precision get parameter subroutine	FGETP	3, 2	FLD	R1E
Standard precision A**I function	FAXI, FAXIX	3, 2	FLD/FSTO, FMPY, FLVR	R1B
Standard precision divide reverse	FDVR, FDVRX	3, 2	FLD/FSTO, FDIV	R1D
Standard precision float divide	FDIV, FDIVX	3, 2	FARC	R1C
Standard precision float multiply	FMPY, FMPYX	3, 2	XMDS, FARC	R1G
Standard precision subtract reverse	FSBR, FSBRX	3, 2	FADD	R1H
Standard add-subtract	FADD, FSUB, FADDX, FSUBX	3, 2	NORM, FARC	R1A
Standard load-store	FLD, FLDX, FSTO, FSTOX	3, 0	None	R1F
Standard precision fractional multiply	XMDS	3, 2	None	S3I

System library programs	Names	Type and subtype	Subroutines required	ID field (73–75)
Common Arithmetic/Function LIBFs				
Fixed point (fractional) double divide	XDD	3, 2	XMD	S3G
Fixed point (fractional) double multiply	XMD	3, 2	None	S3H
Sign reversal function	SNR	3, 2	None	S3F
Integer to floating point function	FLOAT	3, 0	NORM	S3C
Floating point to integer function	IFIX	3, 0	None	S3D
I**J integer function	FIXI, FIXIX	3, 2	None	S3B
Normalize subroutine	NORM	3, 0	None	S3E
Floating accumulator range check subroutine	FARC	3, 2	None	S3A
Interrupt Service Subroutines				
1442 card read punch input/output (no error parameter)	CARD0	5, 0	ILS00, ILS04	U2A
1442 card read punch input/output (error parameter)	CARD1	5, 0	ILS00, ILS04	U2B
2501 card read input (no error parameter)	READ0	5, 0	ILS04	U2L
2501 card read input (error parameter)	READ1	5, 0	ILS04	U2M
1442 card punch output (no error parameter)	PNCH0	5, 0	ILS00, ILS04	U2H
1442 card punch output (error parameter)	PNCH1	5, 0	ILS00, ILS04	U2I
Multiple sector disk input/output (part of supervisor)	DISK1	None	ILS02	---
High speed multiple sector disk input/output (part of supervisor)	DISKN	None	ILS02	---
Synchronous communications adapter (SCA) STR mode	SCAT1	5, 0	IOLOG/CPLOG, ILS01	W1F
SCA (BSC, point-to-point mode)	SCAT2	5, 0	IOLOG/CPLOG, ILS01	W1H
SCA (BSC, multipoint mode)	SCAT3	5, 0	IOLOG/CPLOG, ILS01	W1I
Paper tape input/output	PAPT1	5, 0	ILS04	U2D
Simultaneous paper tape input/output	PAPTN	5, 0	ILS04	U2E
Character/word count paper tape input/output	PAPTX	5, 0	ILS04	U2F
Plotter output subroutine	PLOT1	5, 0	ILS03	U2G
Plotter output subroutine	PLOTX	5, 0	ILS03	V1L
1132 printer output subroutine	PRNT1	5, 0	ILS01	U2J
1132-SCA print with overlap	PRNT2	5, 0	ILS01	W1E
1403 printer output subroutine	PRNT3	5, 0	ILS04	U2K
Keyboard/console printer input/output	TYPE0	5, 0	HOLL, PRTY, ILS04	U2N
Console printer output subroutine	WRTY0	5, 0	ILS04	U2O
1231 optical mark page reader input subroutine	OMPR1	5, 0	ILS04	U2C
MTCA base section	MTCA0	5, 0	ILS03, TSM41, TSTTY	W5B
MTCA 2741 terminal select	TSM41	4, 0	None	W5D
MTCA teletype select	TSTTY	4, 0	None	W5E

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
Conversion Subroutines				
Binary word to 6 decimal characters (card code)	BINDC	3, 0	None	U4B
Binary word to 4 hexadecimal characters (card code)	BINHX	3, 0	None	U4C
6 decimal characters (card code) to binary word	DCBIN	3, 0	None	U4G
EBCDIC to console printer output code	EBPRT	3, 0	EBPA, PRTY	U3A
Card code to EBCDIC-EBCDIC to card code	HOLEB	3, 0	EBPA, HOLL	U3B
Card code to console printer output code	HOLPR	3, 0	HOLL, PRTY	U3C
4 hexadecimal characters (card code) to binary word	HXBIN	3, 0	None	U3D
PTTC/8 to EBCDIC—EBCDIC to PTTC/8	PAPEB	3, 0	EBPA	U3E
PTTC/8 to card code—card code to PTTC/8	PAPHL	3, 0	EBPA, HOLL	U3F
PTTC/8 to console printer output code	PAPPR	3, 0	EBPA, PRTY	U3G
Card code to EBCDIC—EBCDIC to card code	SPEED	3, 0	None	U3H
4 of 8 code to EBCDIC—EBCDIC to 4 of 8 code	EBC48	3, 0	HXCV, STRTB	W1A
4 of 8 code to IBM card code—IBM card code to 4 of 8 code	HOL48	3, 0	HXCV, HOLCA, STRTB	W1B
4 of 8 code to table of displacements	HXCV	3, 0	None	W1D
32-bit binary value to IBM card code decimal value	BIDEC	3, 0	None	U4A
IBM card code decimal value to 32-bit binary value	DECBI	3, 0	None	U4H
Supplement to all standard conversions except those involving PTTC/8	ZIPCO	3, 0	Any ZIPCO Conversion Table	U3I
MTCA code conversion	FEB41, BEB41, F41EB, B41EB, QEB41, Q41EB	4, 0	None	W5A
Conversion Tables				
EBCDIC and PTTC/8	EBPA	3, 0	None	U4K
Card code table	HOLL	3, 0	None	U4P
Console printer output code table	PRTY	3, 0	None	U4Q
Table of IBM card codes	HOLCA	3, 0	None	W1C
Table of 4 of 8 and EBCDIC codes	STRTB	3, 0	None	W1G
ZIPCO Conversion Tables				
EBCDIC to console printer code	EBCCP	4, 0	None	U4I
EBCDIC to IBM card code	EBHOL	4, 0	None	U4J
EBCDIC to 1403 printer code	EBPT3	4, 0	None	U4L
Console printer code to EBCDIC	CPEBC	4, 0	None	U4D
Console printer code to IBM card code	CPHOL	4, 0	None	U4E
Console printer code to 1403 printer code	CPPT3	4, 0	None	U4F
IBM card code to EBCDIC	HLEBC	4, 0	None	U4M
IBM card code to console printer code	HOLCP	4, 0	None	U4O
IBM card code to 1403 printer code	HLPT3	4, 0	None	U4N
1403 printer code to EBCDIC	PT3EB	4, 0	None	U4S
1403 printer code to console printer code	PT3CP	4, 0	None	U4R
1403 printer code to IBM card code	PTHOL	4, 0	None	U4T
Log Subroutine				
Dummy log subroutine called by SCAT1, SCAT2, SCAT3	IOLOG, CPLOG	4, 0	None	W1J

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
Interrupt Level Subroutines				
Interrupt level zero subroutine	ILS00	7, 0	None	U1A
Interrupt level one subroutine	ILS01	7, 0	None	U1B
Interrupt level two subroutine (part of supervisor)	ILS02	7, 1	None	U1C
Interrupt level three subroutine	ILS03	7, 0	None	U1D
Interrupt level four subroutine (part of supervisor)	ILS04	7, 1	None	U1E
Special Interrupt Level Subroutines (restores index register 3)				
Interrupt level zero subroutine	ILSX0	7, 0	None	U1F
Interrupt level one subroutine	ILSX1	7, 0	None	U1G
Interrupt level two subroutine	ILSX2	7, 0	None	U1H
Interrupt level three subroutine	ILSX3	7, 0	None	U1I
Interrupt level four subroutine	ILSX4	7, 0	None	U1J
Standard Plot Calls				
Standard precision character	FCHAR	4, 0	FSIN, FCOS, FPLOT, FCHRX, FLD, FSTOX, FSTO	V1F
Standard precision scale	SCALF	4, 0	FRULE	V1O
Standard precision grid	FGRID	4, 0	FPLOT, POINT, FADD, FLD, FSTO, SNR	V1H
Standard precision plot	FPLOT	4, 0	FMOVE, XYPLT, PLOTI	V1I
Extended Plot Calls				
Extended precision character	ECHAR	4, 0	ESIN, ECOS, EPLOT, ECHRX, ELD, ESTO, ESTOX	V1A
Extended precision scale	SCALE	4, 0	ERULE	V1N
Extended precision grid	EGRID	4, 0	EPLOT, POINT, EADD, ELD, ESTO, SNR	V1C
Extended precision plot	EPLOT	4, 0	EMOVE, XYPLT, PLOTI	V1D
Common Plot Call				
Point characters	POINT	4, 0	PLOTI	V1M
Standard Plot LIBFs				
Standard precision annotation	FCHRX, FCHRI, WCHRI	3, 0	FLOAT, FMPY IFIX, FADD, FLDX, FINC, XYPLT, PLOTI, FSTOX, FLD	V1G
Standard precision plot scaler	FRULE, FMOVE, FINC	3, 0	FLDX, FSUEX, FMPYX, FLD, FSTOX, FMPY, IFIX, FADD	V1J

System library programs	Names	Type and subtype	Subroutines required	ID field (73-75)
Extended Plot LIBFs				
Extended precision annotation	ECHRX, ECHRI, VCHRI	3, 0	FLOAT, FMPY, IFIX, EADD, ELDX, EINC, XYPLT, PLOTI, ESTOX, ELD	V1B
Extended precision plot scaler	ERULE, EMOVE, EINC	3, 0	ELDX, ESUEX, EMPYX, ELD, ESTOX, EMPY, IFIX, EADD, ESTO	V1E
Common Plot LIBFs				
Pen mover	XYPLT	3, 2	PLOTI	V1P
Interface	PLOTI	3, 2	PLOTX	V1K
Interrupt service	PLOTX	5, 0	ILS03	V1L
Disk I/O				
Sequential access	SEQOP, SEQIO, SEQCL	3, 0	DISKZ	W3F
Direct access	DAOPN, DAIO, DACLS	3, 0	DISKZ	W3E
ISAM load	ISLDO, ISLD, ISLDC	3, 0	DISKZ	W3D
ISAM add	ISADO, ISAD, ISADC	3, 0	DISKZ	W3C
ISAM sequential	ISEQO, ISETL, ISEQ, ISEQC	3, 0	DISKZ	W3B
ISAM random	ISRDO, ISRD, ISRDC	3, 0	DISKZ	W3A
RPG Decimal Arithmetic				
Add, subtract, and numeric compare [1]	RGADD, RGSUB, RGNCP	3, 0	None	W2T
Multiply [1]	RGMLT	3, 0	RGBTD, RGDTB, RGERR	W2S
Divide [1]	RGDIV	3, 0	RGERR	W2R
Move remainder [1]	RGMVR	3, 0	RGBTD	W2Q
Binary conversion [1]	RGBTD, RGDTB	3, 0	None	W2P
RPG Sterling and Edit				
Sterling input conversion [1]	RGSTI	3, 0	RGBTD, RGDTB, RGMV1	W4B
Sterling output conversion [1]	RGSTO	3, 0	RGBTD, RGDTB, RGMV2	W4A
Edit [1]	RGEDT	3, 0	RGMV2, RGSI5	W2O
RPG Move				
From I/O buffer to core [1]	RGMV1, RGMV5	3, 0	None	W2N
From core to I/O buffer [1]	RGMV2	3, 0	None	W2M
MOVE operation [1]	RGMV3	3, 0	None	W2L
MOVEL operation [1]	RGMV4	3, 0	None	W2K
RPG Compare				
Alphameric [1]	RGCMP	3, 0	None	W2J

[1] Not distributed to paper tape users.

System library programs	**Names**	**Type and subtype**	**Subroutines required**	**ID field (73-75)**
RPG Indicators				
Test [1]	RGSI1	3, 0	None	W2I
Set resulting on [1]	RGSI2	3, 0	None	W2H
Set on, set off [1]	RGSI3, RGSI4	3, 0	None	W2G
Test for 0 or blank [1]	RGSI5	3, 0	None	W2E
RPG Miscellaneous				
Test zone [1]	RGTSZ	4, 0	None	W2D
Convert to binary [1]	RGCVB	3, 0	None	W2C
Object time error [1]	RGERR	4, 0	None	W2B
Blank after [1]	RGBLK	3, 0	None	W2A
Alternating sequence [1]	ALTSE	None		

[1] Not distributed to paper tape users.

Appendix D. LET/FLET

The location equivalence table (LET) contains the name and disk block count of all programs, including those in the System Library, and data files stored in the user area (UA). The fixed location equivalence table (FLET) contains the names of all programs and data files stored in the fixed area (FA).

Each cartridge has a LET. FLET is optional and is defined when you use the DEFINE FIXED AREA function of the Disk Utility Program (DUP).

LET/FLET DISK FORMAT

Each sector of LET or FLET contains a 5 work sector header. All entries in LET or FLET are 3 words long and consist of a name and disk block count.

sector header format

Word	*Entry*
1	Relative sector number for this cartridge only
2	Sector address of the UA (or sector address of FX if FLET)
3	Reserved
4	Number of words available in this LET/FLET sector
5	Sector address of the next LET/FLET sector on this cartridge. Word 5 is zero if this is the last LET/FLET sector on this cartridge.

LET/FLET entry format

1, bits 0–1	00–if DSF format (LET only) 10–if DCI format 11–if data format
1, bits 2–15 and 2	Program or data file name in name code
3	Disk block (DB) count of program or data file

Sometimes unused disk space occurs because data files and programs in core image format are stored on sector boundaries. Such spaces are represented by a 1DUMY entry in LET/FLET.

A 1DUMY entry is always inserted to precede a DDF or DCI entry when the last entry is in DSF format, even when the preceding program ends on a sector boundary. In the latter case, a 1DUMY entry with a DB count of zero (blank) is generated. This 1DUMY entry is made because a DELETE operation may call for a 1DUMY padding in the future and because under certain circumstances, room for a 1DUMY entry may not be available.

1DUMY entry format

Word	*Entry*
1, bits 0–1	Reserved
1, bits 2–15 and word 2	Name code for 1DUMY
3	DB count of entry

The last entry of LET is a 1DUMY entry that reflects the current size of available working storage.

LET/FLET DUMP FORMAT

The DUP control records DUMPLET or DUMPFLET are used to dump LET and FLET, or FLET, respectively, to the principal printer. One sector of LET/FLET is printed per page. Each page is headed with the word LET or FLET, whichever is applicable. Each sector of LET/FLET dumped is preceded by 2 lines of header information. The first header line contains the contents of the following locations from COMMA/DCOM:

#CIDN—Cartridge ID, logical drive 0, 1, 2, 3, or 4
$FPAD—COMMA file protect address, logical drive 0, 1, 2, 3, or 4
#FPAD—DCOM file protect address, logical drive 0, 1, 2, 3, or 4
#CIBA—CIB address, logical drive 0, 1, 2, 3, or 4
#ULET—LET address, logical drive 0, 1, 2, 3, or 4
#FLET—FLET address, logical drive 0, 1, 2, 3, or 4

A second header line is printed that reflects information about the LET or FLET sector that is being dumped:

SCTR NO.—The relative sector number
UA/FXA—The actual sector address of the user area or fixed area
WORDS AVAIL—Available words in the sector
CHAIN ADR—Chain address to the next sector of LET or FLET

The LET/FLET entries for the sector are printed after the 2 header lines. Twenty-one lines of entries are printed, 5 entries per line, and sequenced by column. Each entry is formatted as follows:

PROG NAME—5 print positions plus a blank
FORMAT—DSF, DCI, or DDF: 3 print positions plus a blank, 4 blanks if 1DUMY or secondary entry point
DB CNT—Disk block count, 4 print positions plus a blank
DB ADDR—Logical disk block address, 4 print positions plus 5 blanks

Only the name is printed for each secondary entry. Examples of DUMPLET and DUMP-FLET follow:

```
// JOB

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         4444          4444         0003
                             1124         0004

V2 M10    ACTUAL 32K   CONFIG 32K

// DUP

*DEFINE FIXED AREA               5
CART ID 4444   CYLS FXA   0004   DBS AVAIL 0200   FLET SECTOR ADDR   01E8

*STOREDATA   CD   FX   DATA     10
CART ID 4444    DB ADDR   1F00    DB CNT     0020

*STOREDATACICD   FX   CIMGE    18
CART ID 4444    DB ADDR   1F20    DB CNT     0030

*STOREDATA   CD   UA   DATA2    10
CART ID 4444    DB ADDR   2E30    DB CNT     0020

*DUMPLET
                                          LET



=CIDN    $FPAD    =FPAD    =CIBA    =ULET    =FLET
4444     02E5     02E5     0210     0220     01E8

SCTR NO.   UA/FXA.    WORDS AVAIL.   CHAIN ADDR.
0000       0228          0000           0221
```

PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR
FADD	DSF	0008	2280	FATN	DSF	000A	22AA	ESUB				EATAN				FIXIX			
FSUB				FATAN				EADDX				EAXB	DSF	0005	231D	FLOAT	DSF	0002	235E
FADDX				FAXB	DSF	0005	22B4	ESUBX				EAXBX				IFIX	DSF	0004	2360
FSUBX				FAXBX				EAXI	DSF	0007	22F2	EEXP	DSF	000A	2322	NORM	DSF	0004	2364
FAXI	DSF	0006	2288	FEXP	DSF	0009	22B9	EAXIX				EXPN				SNR	DSF	0002	2368
FAXIX				FXPN				EDIV	DSF	0006	22F9	ELN	DSF	000B	232C	XDD	DSF	0006	236A
FDIV	DSF	0007	228E	FLN	DSF	000A	22C2	EDIVX				EALOG				XMD	DSF	0005	2370
FDIVX				FALOG				EDVR	DSF	0003	22FF	ESIGN	DSF	0003	2337	XMDS	DSF	0003	2375
FDVR	DSF	0003	2295	FSIGN	DSF	0003	22CC	EDVRX				ESINE	DSF	000B	233A	FBTD	DSF	001B	2378
FDVRX				FSIN	DSF	0009	22CF	EGETP	DSF	0003	2302	ESIN				FDTB			
FGETP	DSF	0003	2298	FSINE				ELD	DSF	0004	2305	ECOSN				IABS	DSF	0002	2393
FLD	DSF	0005	229B	FCOS				ELDX				ECOS				XSQR	DSF	0004	2395
FLDX				FCOSN				ESTO				ESQR	DSF	0006	2345	PAUSE	DSF	0003	2399
FSTO				FSQR	DSF	0006	22D8	ESTOX				ESQRT				STOP	DSF	0002	239C
FSTOX				FSQRT				EMPY	DSF	0004	2309	ETANH	DSF	0004	234B	SUBIN	DSF	0003	239E
FMPY	DSF	0005	22A0	FTANH	DSF	0005	22DE	EMPYX				ETNH				SUBSC	DSF	0003	23A1
FMPYX				FTNH				ESBR	DSF	0003	230D	SEAR	DSF	0004	234F	TTEST	DSF	0002	23A4
FSBR	DSF	0003	22A5	SFAR	DSF	0004	22E3	ESBRX				SEARX				TSET			
FSBRX				SFARX				EABS	DSF	0002	2310	SEIF	DSF	0003	2353	DATSW	DSF	0003	23A6
FABS	DSF	0002	22A8	SFIF	DSF	0003	22E7	EAVL				FARC	DSF	0003	2356	DVCHK	DSF	0002	23A9
FAVL				EADD	DSF	0008	22EA	EATN	DSF	000B	2312	FIXI	DSF	0005	2359	FCTST	DSF	0003	23AB

```
*CIDN   $FPAD   *FPAD   *CIBA   *ULET   *FLET
4444    02E5    02E5    0210    0220    01E8

SCTR NO.  UA/FXA.   WORDS AVAIL.  CHAIN ADDR.
0001      0228          0000          0222


PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB
NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR

ISIGN DSF 0003 23AE    SIOF                   EBCTB DSF 0005 2450    ILSX1 DSF 0003 24BA    HOLPR DSF 0007 25C1
OVERF DSF 0002 23B1    SIOAF                  GETAD DSF 0002 2455    ILSX2 DSF 0003 24BD    HXBIN DSF 0005 25C8
PDUMP DSF 0009 23B3    SIOFX                  HOLEZ DSF 0005 2457    ILSX3 DSF 0004 24C0    PAPEB DSF 0010 25CD
SLITE DSF 0006 23BC    SCOMP                  HOLTB DSF 0004 245C    ILSX4 DSF 0004 24C4    PAPHL DSF 0010 25DD
SLITT                  SWRT                   PAPTZ DSF 000F 2460    CARD0 DSF 0010 24C8    PAPPR DSF 000D 25ED
TSTOP DSF 0002 23C2    SRED                   PNCHZ DSF 0006 246F    CARD1 DSF 0011 24D8    SPEED DSF 0015 25FA
TSTRT DSF 0002 23C4    SIOIX                  PRNTZ DSF 000E 2475    OMPR1 DSF 0015 24E9    ZIPCO DSF 000B 260F
SDFIO DSF 0019 23C6    UFIO  DSF 001D 2427    PRNZ  DSF 000C 2483    PAPT1 DSF 000F 24FE    BIDEC DSF 0006 261A
SDAF                   URED                   READZ DSF 0005 248F    PAPTN DSF 0013 250D    BINDC DSF 0005 2620
SDAI                   UWRT                   TYPEZ DSF 0008 2494    PAPTX DSF 0015 2520    BINHX DSF 0004 2625
SDCOM                  UIOI                   WRTYZ DSF 0005 249C    PLOT1 DSF 000C 2535    CPEBC DSF 0009 2629
SDF                    UIOF                   SGOTO DSF 0003 24A1    PNCH0 DSF 000E 2541    CPHOL DSF 0009 2632
SDFX                   UIOAI                  SIAR  DSF 0004 24A4    PNCH1 DSF 000E 254F    CPPT3 DSF 0009 263B
SDI                    UIOAF                  SIARX                  PRNT1 DSF 001A 255D    DCBIN DSF 0006 2644
SDIX                   UIOFX                  SIIF  DSF 0003 24A8    PRNT3 DSF 0010 2577    DECBI DSF 0009 264A
SDRED                  UIOIX                  ILS00 DSF 0003 24AB    READ0 DSF 0007 2587    EBCCP DSF 0009 2653
SDWRT                  UCOMP                  ILS01 DSF 0003 24AE    READ1 DSF 0008 258E    EBHOL DSF 0009 265C
SDFND DSF 0006 23DF    BCKSP                  ILS02 DSF 0001 24B1    TYPE0 DSF 0012 2596    EBPA  DSF 0006 2665
SFIO  DSF 0042 23E5    EOF                    ILS03 DSF 0003 24B2    WRTY0 DSF 0009 25A8    EBPT3 DSF 0009 266B
SIOI                   REWND                  ILS04 DSF 0002 24B5    EBPRT DSF 0007 25B1    HLEBC DSF 0009 2674
SIOAI                  CARDZ DSF 000C 2444    ILSX0 DSF 0003 24B7    HOLEB DSF 0009 25B8    HLPT3 DSF 0009 267D

*CIDN   $FPAD   *FPAD   *CIBA   *ULET   *FLET
4444    02E5    02E5    0210    0220    01E8

SCTR NO.  UA/FXA.   WORDS AVAIL.  CHAIN ADDR.
0002      0228          0000          0223


PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB      PROG  FOR DB   DB
NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR    NAME  MAT CNT  ADDR

HOLCP DSF 0009 2686    MODSF DSF 006C 288B    FRULE DSF 0009 29A5    PRTZ2 DSF 0006 2B25    RGDIV DSF 002E 2C3C
HOLL  DSF 0006 268F    PTUTL DSF 000A 28F7    FMOVE                  DFCNV DSF 009F 2B2B    RGMLT DSF 0010 2C6A
PRTY  DSF 0006 2695    CALPR DSF 0007 2901    FINC                   RGBLK DSF 0005 2BCA    RGADD DSF 001A 2C7A
PT3CP DSF 0009 269B    FSLEN DSF 000B 2908    PLOTI DSF 0003 29AE    RGERR DSF 0005 2BCF    RGSUB
PT3EB DSF 0009 26A4    FSYSU                  PLOTS                  RGCVB DSF 0006 2BD4    RGNCP
PTHOL DSF 0009 26AD    RDREC DSF 0015 2913    PLOTX DSF 0009 29B1    RGTSZ DSF 0005 2BDA    ISRD0 DSF 001C 2C94
DMP80 DSF 0007 26B6    ECHAR DSF 0005 2928    POINT DSF 0008 29BA    RGSI5 DSF 0006 2BDF    ISRD
DMTD0 DSF 001A 26BD    ECHRX DSF 0028 292D    SCALE DSF 0002 29C2    RGSI3 DSF 0004 2BE5    ISRDC
DMTX0                  ECHRI                  SCALF DSF 0002 29C4    RGSI4                  ISEQ0 DSF 002A 2CB0
DMPD1 DSF 001E 26D7    VCHRI                  XYPLT DSF 0007 29C6    RGSI2 DSF 0006 2BE9    ISETL
DMPX1                  EGRID DSF 0008 2955    EBC48 DSF 000B 29CD    RGSI1 DSF 0005 2BEF    ISEQ
FLIPR DSF 0007 26F5    EPLOT DSF 0005 295D    HOL48 DSF 0008 29D8    RGCMP DSF 0006 2BF4    ISEQC
SYSUP DSF 003A 26FC    ERULE DSF 000A 2962    HOLCA DSF 0006 29E0    RGMV4 DSF 0006 2BFA    ISAD0 DSF 003E 2CDA
ADRWS DSF 0010 2736    EMOVE                  HXCV  DSF 0004 29E6    RGMV3 DSF 0004 2C00    ISAD
COPY  DSF 0022 2746    EINC                   PRNT2 DSF 001F 29EA    RGMV2 DSF 000B 2C04    ISADC
DISC  DSF 0036 2768    FCHAR DSF 0005 296C    SCAT1 DSF 004A 2A09    RGMV1 DSF 000A 2C0F    ISLD0 DSF 0026 2D18
DLCIB DSF 001F 279E    FCHRX DSF 0028 2971    STRTB DSF 0006 2A53    RGMV5                  ISLD
DSLET DSF 0045 27BD    FCHRI                  SCAT2 DSF 0069 2A59    RGEDT DSF 0013 2C19    ISLDC
IDENT DSF 000C 2802    WCHRI                  SCAT3 DSF 0061 2AC2    RGDTB DSF 0008 2C2C    DAOPN DSF 000F 2D3E
ID    DSF 001A 280E    FGRID DSF 0008 2999    IOLOG DSF 0002 2B23    RGBTD                  DAIO
MODIF DSF 0063 2828    FPLOT DSF 0004 29A1    CPLOG                  RGMVR DSF 0008 2C34    DACLS
```

*CIDN	$FPAD	*FPAD	*CIBA	*ULET	*FLET
4444	02E5	02E5	0210	0220	01E8

SCTR NO.	UA/FXA.	WORDS AVAIL.	CHAIN ADDR.
0003	0228	0105	01E8

PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR
SEQOP	DSF	001C	2D4D																
SEQIO																			
SEQCL																			
RGSTO	DSF	001A	2D69																
RGSTI	DSF	000F	2D83																
FEB41	DSF	0024	2D92																
BEB41																			
F41EB																			
B41EB																			
QEB41																			
Q41EB																			
MTCA0	DSF	0023	2DB6																
MTCAZ	DSF	000E	2DD9																
TSM41	DSF	003A	2DE7																
TSTTY	DSF	000B	2E21																
1DUMY		0004	2E2C																
DATA2	DDF	0020	2E30																
1DUMY		35B0	2E50																

FLET

*CIDN	$FPAD	*FPAD	*CIBA	*ULET	*FLET
4444	02E5	02E5	0210	0220	01E8

SCTR NO.	UA/FXA.	WORDS AVAIL.	CHAIN ADDR.
0010	01F0	0132	0000

PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR	PROG NAME	FOR MAT	DB CNT	DB ADDR
DATA	DDF	0020	1F00																
CIMGE	DCI	0030	1F20																
1DUMY		01B0	1F50																

END OF DUMPLET/FLET

```
*DUMPFLET

                                        FLET




*CIDN   $FPAD   *FPAD   *CIBA   *ULET   *FLET
4444    02E5    02E5    0210    0220    01E8

SCTR NO.  UA/FXA.   WORDS AVAIL.  CHAIN ADDR.
0010       01F0          0132          0000


 PROG  FOR DB   DB        PROG  FOR DB   DB        PROG  FOR DB   DB        PROG  FOR DB   DB        PROG  FOR DB   DB
 NAME  MAT CNT  ADDR      NAME  MAT CNT  ADDR      NAME  MAT CNT  ADDR      NAME  MAT CNT  ADDR      NAME  MAT CNT  ADDR

 DATA  DDF 0020 1F00
 CIMGE DCI 0030 1F20
 1DUMY     0180 1F50

 END OF DUMPFLET
```

Appendix E. System Location Equivalence Table (SLET)

The addresses listed in the following SLET printout are subject to change. Only the symbols and phase IDs remain constant.

SYSTEM LOCATION EQUIVALENCE TABLE (SLET)

SYMBOL	PH ID	CORE ADDR	WORD COUNT	SCTR ADDR	SYMBOL	PH ID	CORE ADDR	WORD COUNT	SCTR ADDR	SYMBOL	PH ID	CORE ADDR	WORD COUNT	SCTR ADDR	SYMBOL	PH ID	CORE ADDR	WORD COUNT	SCTR ADDR
*****	**	****	****	****	*****	**	****	****	****	*****	**	****	****	****	*****	**	****	****	****
'DDUP	01	7C50	032F	0808	'DCTL	02	11DE	05A2	000B	'STOR	03	21DE	05A2	0010	'FILQ	04	01DE	03C0	0015
'DUMP	05	41DE	054B	0018	'DL/F	06	01DE	03C0	001D	'DLTE	07	01DE	05A2	0020	'DFNE	08	01DE	05A2	0025
'EXIT	09	01DE	0500	002A	'CFCE	0A	7A06	00DE	002E	'DU11	0B	7A06	0035	002F	'DU12	0C	7A06	00D8	0030
'DU13	0D	7782	087C	0031	'DU14	0E	7A06	0248	0038	'DU15	0F	7A06	0248	003A	'DU16	10	7A06	0248	003C
'PRCI	11	01DE	0280	003E	'DU18	12	0E6E	0140	0040	'FR01	1F	760C	09F1	0041	'FR02	20	7A34	0500	0049
'FR03	21	7A34	0280	004D	'FR04	22	7A34	03C0	004F	'FR05	23	7A34	0500	0052	'FR06	24	7A34	03C0	0056
'FR07	25	7A34	0280	0059	'FR08	26	7A34	0500	005B	'FR09	27	7A34	03F0	005F	'FR10	28	7A34	03C0	0063
'FR11	29	7A34	03C0	0066	'FR12	2A	7A34	03C0	0069	'FR13	2B	7A34	03C0	006C	'FR14	2C	7A34	0500	006F
'FR15	2D	7A34	0500	0073	'FR16	2E	7A34	0500	0077	'FR17	2F	7A34	0500	007B	'FR18	30	7A34	0500	007F
'FR19	31	7A34	0404	0083	'FR20	32	7A34	03C0	0087	'FR21	33	7A34	03C0	008A	'FR22	34	7A34	0280	008D
'FR23	35	7A34	03C0	008F	'FR24	36	7A34	03C0	0092	'FR25	37	7A34	0500	0095	'FR26	38	7B8E	03C0	0099
'FR27	39	766E	0140	009C	'SUP1	6E	04FE	02FE	009D	'SUP2	6F	07FE	052B	00A0	'SUP3	70	07FE	0280	00A5
'SUP4	71	07FE	0280	00A7	'SUP5	72	07FE	03EA	00A9	'SUP6	73	0506	04F8	00AD	'SUP7	74	0400	0189	00B1
'CLB1	78	01E0	0782	00B3	'CLB2	79	05BC	04E2	00BA	'CLB3	7A	08B6	01E8	00BE	'CLB4	7B	08B6	01E8	00C0
'CLB5	7C	08B6	01E8	00C2	'CLB6	7D	08B6	01E8	00C4	'CLB7	7E	0AA0	0140	00C6	'CLB8	7F	0AA0	0140	00C7
'CLB9	80	0AA0	0140	00C8	'CLBA	81	0AA0	0140	00C9	'CLBB	82	0BE2	0140	00CA	'CLBC	83	08B6	01E8	00CB
'CLBD	84	0AA0	0140	00CD	'1403	8C	0000	0132	80CE	'1132	8D	0000	0113	00CF	'CPTR	8E	0000	011B	00D0
'2501	8F	0000	009C	00D1	'1442	90	0000	00AB	00D2	'1134	91	0000	016C	00D3	'KBCP	92	0000	0174	00D5
'CDCV	93	0000	00B9	00D7	'PTCV	94	0000	0003	00D8	'KBCV	95	0000	0003	00D9	'DZID	96	00F0	00EC	00DA
'D1ID	97	00F0	01A2	00DB	'DNID	98	00F0	02B0	00DD	'PPRT	99	0000	0113	00CF	'PIWK	9A	0000	009C	00D1
'PIXK	9B	0000	009C	00D1	'PCWK	9C	0000	00B9	00D7	'PCXK	9D	0000	00B9	00D7	'CIL1	A0	0000	0170	00E0
'CIL2	A1	0000	01C0	00E2	'RG00	B0	0212	094E	00E4	'RG02	B1	0906	0893	00EC	'RG04	B2	0906	07B3	00F3
'RG06	B3	0906	076B	00FA	'RG08	B4	0906	08C2	0100	'RG10	B5	04A6	0811	0108	'RG12	B6	073A	0867	010F
'RG14	B7	073A	06D4	0116	'RG16	B8	0762	048E	011C	'RG17	B9	0762	06E3	0120	'RG19	BA	073A	0932	0126
'RG20	BB	073A	05C8	012E	'RG21	BC	073A	06DF	0133	'RG22	BD	0782	02AE	0139	'RG24	BE	0782	06B9	013C
'RG26	BF	0782	022E	0142	'RG28	C0	0782	0491	0144	'RG32	C1	0782	06C7	0148	'RG34	C2	0782	046C	014E
'RG36	C3	0782	025F	0152	'RG38	C4	0782	050C	0154	'RG40	C5	0782	0576	0159	'RG42	C6	0782	03A5	015E
'RG44	C7	0782	05F3	0161	'RG46	C8	0782	04E9	0166	'RG52	C9	073A	03AC	016A	'RG54	CA	073A	0667	016D
'RG58	CB	073A	05FD	0173	'RG60	CC	073A	039D	0178	'DCL2	CD	11DE	0280	017B	'DMUP	CE	01DE	11DF	017D
'AS00	CF	01E0	026B	018C	'ACNV	D0	01E8	00BB	018E	'AS10	D1	01E8	0060	018F	'AS11	D2	01E8	0050	0190
'AS12	D3	027E	01B9	0191	'AERM	D4	0AC8	013E	0193	'AS01	D5	027E	01D8	0194	'AS1A	D6	027E	0115	0196
'ASYM	D7	0000	0130	0197	'AS03	D8	07A6	0250	0198	'AS04	D9	027E	01D7	019A	'AS02	DA	027E	01A2	019C
'AS2A	DB	0280	00A6	019E	'AS09	DC	0456	059E	019F	'AS05	DD	027E	01C4	01A4	'AS06	DE	027E	01D8	01A6
'AS07	DF	027E	017E	01A8	'AS7A	E0	0280	0127	01AA	'AS08	E1	027E	0198	01AB	'AS8A	E2	027E	01B5	01AD
'APCV	E3	027E	0099	01AF	'AINT	E4	0980	0058	01B0	'ASAA	E5	0980	0063	01B1	'ASGR	E6	0EBC	03C1	01B2
'ADIV	E7	027E	00B8	01B6	'AMCC	E8	027E	018A	01B7	'AM01	E9	027E	01D6	01B9	'AM1A	EA	027E	01D8	01BB
'AM1B	EB	027E	01D8	01BD	'AM02	EC	027E	01D6	01BF	'AM2A	ED	027E	01D6	01C1	'AM2B	EE	05DA	015A	01C3
'AM03	EF	07AA	0051	01C5	'AM3A	F0	027E	01B3	01C6	'AM3B	F1	12E6	0285	01C8	'AX01	F2	027E	01C9	01CB
'AX2A	F3	07A6	0054	01CD	'AX2B	F4	091C	005B	01CE	'AX2C	F5	0882	003D	01CF	'AX03	F6	0EBC	03BE	01D0

The following is a partial printout of a core dump:

ACCUMULATOR 4000 EXTENSION 0FAF XR1 0000 XR2 025A XR3 007F OVERFLOW OFF CARRY OFF

ADDR	***0	***1	***2	***3	***4	***5	***6	***7	***8	***9	***A	***B	***C	***D	***E	***F
0000	70FF	FFFB	0000	007F	0FFA	0378	007F	0000	0327	01E4	00B3	0000	00C4	0091	8000	0000
0010	0000	0000	FFFF	0000	70FF	0000	0001	0000	D900	70FF	4000	0FAF	4C00	0500	DC00	04FE
0020	C0FE	1890	4400	00F2	7400	00EE	70FD	70F4	038C	3000	4C80	0028	00E6	0388	0000	0000
0030	0000	0000	0001	0000	0000	0000	0000	0002	7019	0000	1810	7017	0001	0004	FFFF	0000
0040	D8D9	4023	282A	69D3	C480	003F	D0D2	C8F4	4400	00F2	C0F1	7007	0000	0000	0000	0000
0050	0000	0000	C0EB	D0C2	6580	0039	C101	18D0	C100	6500	01DC	D8B8	4008	C0FC	1890	4400
0060	00F2	4003	4102	0000	0000	005D	C0CB	E8A9	4C20	0066	0803	4C80	0065	2000	0000	CC80
0070	0000	0000	0000	0000	0000	0000	FFFF	0000	006E	0001	0000	0000	0000	0000	0000	0000
0080	0000	0000	3000	4C80	0081	0000	3000	4C80	0085	0000	3000	4C80	0089	0000	3000	4C80
0090	008D	0000	3000	4CC0	0091	0469	0000	0000	0000	0000	0378	0000	0000	0000	0000	8800
00A0	0000	0000	0000	0000	0658	0658	0658	0000	0000	0000	0000	0000	0000	0000	0000	0000
00B0	0000	0000	0000	0067	6906	6A07	2807	D80A	4400	00F7	6500	01DC	6600	025A	2000	C802
00C0	4CC0	00B3	0001	0000	06A9	D818	280E	690F	6A10	0816	1002	4C10	00D0	4480	002C	FFFE
00D0	6109	0810	1140	4580	054C	2003	6500	0000	6600	025A	C803	4CC0	00C4	4001	0FAF	4E00
00E0	0209	0F00	0000	0300	0000	0F01	0000	08FC	4C40	00EA	4400	003F	FFFE	0000	0011	0000
00F0	00EF	FF6A	004A	7400	00EE	70FD	7002	0000	7018	690B	6A0C	1008	D03C	18D0	D05A	7054
0100	4C00	01C1	690F	0822	6500	0000	6600	025A	C0EE	4C98	00F2	D003	1810	D0E9	4C00	00BA
0110	1000	6500	0004	6600	00F2	0816	D0C6	4810	70E7	C80A	D900	74FF	00EE	703A	C80F	C011
0120	4293	7034	0001	0378	0FFA	0378	0004	8D00	0004	8D00	0122	8E00	8C00	8F81	0EBA	0379
0130	5002	5004	FEC0	0001	0080	0600	0008	5000	0FF8	0100	0701	0007	000A	009F	FFFB	8E80
0140	0400	0141	0000	FFFF	0000	0000	1810	D0A6	74FF	0032	1000	70B8	C8D7	D900	C0E1	70D0
0150	C0E6	4400	0028	7038	7401	0032	6211	6A96	6500	0004	C900	D8C8	D8D1	1810	1084	D00E
0160	80DC	D01C	80DB	D037	80D7	8008	8007	D006	62FD	69BE	C101	E0CC	D101	9400	00A4	4828
0170	7007	C101	80C3	7401	016E	7201	70F5	D101	6600	00F2	C23D	E249	D250	C400	009F	EA4E
0180	D23A	EA43	D239	EA50	9247	D237	EA42	8247	D24D	EA48	D23B	CA3C	0A3A	D2EB	4828	70BC
0190	1002	4828	70BD	1002	4810	7003	1810	D480	019B	C101	9400	009A	4818	7014	1893	180F
01A0	1002	EA3A	18D0	4810	7002	F251	8230	DA34	420F	CA38	DA34	420F	C231	D480	019B	9101
01B0	4C20	0119	CA3C	4808	7091	8A40	DA3C	4830	1810	824F	D100	CA36	DA34	C101	EA50	D101
01C0	420F	C24D	D235	C247	4820	420F	CA32	D900	C23C	4808	70E9	7500	0140	C900	DA32	CA3C
01D0	D900	708B	0000	0000	0000	0000	0000	0000	0000	0000	00A0	0ED4	0170	00E0	0113	00CF
01E0	4C00	0000	7010	4C40	0000	6959	6A5A	6B5B	285B	D802	7052	4040	0000	0000	7001	0000
01F0	00F0	000A	1600	6945	6A46	D017	6236	C200	4830	70FD	C0FF	D200	1090	D001	6500	0000
0200	D0EE	087A	E0EF	4818	7006	1005	4810	70F9	C079	42F2	70F6	7401	0032	0000	7003	7037
0210	0877	7026	D249	086C	C201	7401	0037	6204	1890	1804	D0F2	1010	A8D4	18D0	100C	E8ED
0220	72FF	70F7	7105	6910	1890	62FE	2000	6102	1010	1004	1084	4818	4801	E8C2	4820	2001
0230	71FF	70F7	E8B8	D600	0000	7201	70F0	C8B6	6500	0000	6600	0000	70A3	7058	6500	0000
0240	6600	0000	6700	0000	2000	C8A6	709C	6801	6600	0000	723C	6A3A	6916	C100	D001	7500
0250	0000	6958	D00D	9026	1883	800B	D054	1010	1084	D001	6500	0000	C021	1100	D01C	6600
0260	0000	6101	C600	0000	8017	4802	7101	1008	8013	4802	7101	72FF	70F5	6933	71FF	7001
0270	709F	6231	6A0D	4019	C005	D2F6	D205	080C	70BF	70C4	0001	0000	BF00	3700	0002	3701
0280	0000	3404	6000	3402	0000	3480	0000	3200	FF00	3401	0000	3440	4C00	0000	6120	189B
0290	10A0	D900	D902	D904	D906	70F6	08E7	4810	7036	40F3	4802	7048	C0ED	4820	7049	08E6
02A0	6600	0000	C0DD	90D6	D0DB	72FF	4808	7022	7201	6700	0000	0000	C8CD	C300	F0D5	2002
02B0	4810	80D6	2806	1008	F0CF	4820	7002	72FF	C0C1	7000	72FF	E8FA	1008	1808	73FF	18C2
02C0	4810	70EB	18D0	D11F	71FF	70E7	6ADA	7401	0027	70AF	620F	C0FF	D227	6ABC	70F8	6100
02D0	1001	4810	700A	1007	4810	70A3	08AB	74FF	0032	1000	1010	D136	709C	1001	4810	7099
02E0	1802	4804	D17F	70F3	622E	6AA4	6278	70DE	9091	D0A0	4808	C116	4810	70D9	7401	0027
02F0	0899	0896	7086	D215	C109	D219	C107	D201	1002	1802	9395	4820	7019	C396	9108	4820
0300	7015	6500	0000	C101	E21D	D3F8	C2DE	4820	42DE	C201	1001	4828	701D	4802	7022	70C9
0310	C2F9	D3A0	820C	D39E	CAFA	70AB	C391	8219	D391	7103	C215	921E	D215	4820	009C	00D1
0320	4C00	06A7	7077	4C00	00D5	700C	4C40	0000	D835	2806	0835	C034	803B	D032	0833	C82E
0330	2000	70F4	0829	1002	4828	701B	1001	4828	7036	1010	D400	0013	74FF	0032	1000	C022
0340	D001	6600	0FAF	C202	18D0	C201	9825	4818	18D0	4820	70D8	C203	4820	70D5	6C00	000F
0350	70D2	C010	D00D	D012	C008	4804	7002	401E	7001	4028	080B	70C7	1401	4F01	0000	0000
0360	0000	1100	0FAF	1701	4E00	4F01	0FAF	4E00	0001	1402	1000	1702	3000	3000	4000	C0ED
0370	4804	70C7	D400	0033	70C4	4C00	0000	08F2	4804	7003	C0F0	D0E1	70F8	C0EC	4400	0028
0380	70F6	4C00	038E	08E0	4804	7003	C0DE	D0D5	70F8	C0E4	4400	0028	70F6	40F4	C0D5	D0D7
0390	18D0	D0D4	D0CF	6C00	0013	7401	0032	08CE	2003	7086	28FD	D003	7400	0013	70FD	7002
03A0	70EC	700C	70EA	40D2	C0C4	D0C1	C400	0033	4818	70E9	1010	D400	0033	70E9	40C7	C0AC
03B0	D0B6	18D0	80B5	D0AC	70DD	0000	0000	0000	0000	0000	0000	2542	00B9	00D7	4C00	06AD
03C0	282E	692A	6A2B	D003	D023	6250	C600	0FAF	D028	1886	1807	1883	C024	610A	E023	1140
03D0	1001	4818	7002	C020	7012	691B	C01A	1084	D008	D017	6808	C007	8016	D005	C012	1091
03E0	6500	0000	C500	03F7	4802	1008	1808	D600	0FAF	72FF	70DB	6500	0000	6600	025A	2003
03F0	70CD	0000	1FC0	001C	00EF	0000	0000	40F9	F838	F0E9	E828	60D9	D818	D0A9	A868	50C9
0400	C808	C089	8848	6A99	9858	70B9	B878	F737	7F3F	E727	6F2F	D717	5F1F	A767	AFEF	C707

ADDR	***0	***1	***2	***3	***4	***5	***6	***7	***8	***9	***A	***B	***C	***D	***E	***F
0410	4F0F	8747	8FCF	9757	9FDF	B777	BFFF	F636	7E3E	E626	6E2E	D616	5E1E	A666	AEEE	C606
0420	4E0E	8646	8ECE	9656	9EDE	B676	BEFE	F535	7D3D	E525	6D2D	D515	5D1D	A565	ADED	C505
0430	4D0D	8545	8DCD	9555	9DDD	B575	BDFD	F434	7C3C	E424	4D2C	D414	5C1C	A464	ACEC	C404
0440	5D0C	8444	8CCC	9454	9CDC	B474	BCFC	F333	7B3B	E323	6B2B	D313	5B1B	A363	ABEB	C303
0450	4B0B	8343	8BCB	9353	9BDB	B373	BBFB	F232	7A3A	E222	E02A	D212	5A1A	A262	AAEA	C202
0460	4A0A	8242	8ACA	9252	9ADA	B272	BAFA	F131	7939	6121	6929	D111	5919	A1E1	A020	C101
0470	4909	8141	8000	9151	9010	B171	B030	00D8	0040	00D4	00D6	00C4	00C9	00C6	0000	0000
0480	0000	0000	0000	0000	0000	0000	0000	0000	0000	720C	6A02	61F4	C500	0466	D500	0FBC
0490	7101	70FA	61BC	C014	D500	1000	7101	70FC	6580	0019	7101	7007	61FC	C500	0451	D500
04A0	0FBE	7101	70FA	C005	D400	000F	4C00	0038	0040	FFFF	0000	0000	0000	0000	0000	0000
04B0	0000	0000	0000	0000	0000	1000	C817	DC80	04D9	C01F	1890	4400	00F2	7400	00EE	70FD
04C0	C018	8400	054E	D010	D012	D400	0009	6500	04CF	6D00	00D4	700F	0099	0000	0113	00CF
04D0	0000	0000	0000	0000	0000	0000	0000	0000	0000	058A	058D	7200	7001	7038	C600	0550
04E0	D001	6500	0000	CC00	0014	D828	1002	1802	1888	4018	D103	180A	1086	4014	1008	D020
04F0	1002	1086	400F	E81C	D104	C818	1084	1810	1086	4008	1008	D014	1002	1086	02FE	009D
0500	C400	0064	4C10	0508	4400	0039	221D	15C0	4C08	050E	4400	0039	0764	9540	616E	6D00
0510	0078	1000	6500	054C	6D00	00D4	C841	6100	4022	C037	D400	0009	6500	057E	C400	007C
0520	4C20	0649	6500	0582	4C00	0649	C835	6101	4012	C029	801E	D400	0008	4074	C01C	802C
0530	8018	1801	1001	D018	8014	D400	06B6	C828	6102	4001	707E	053A	DD80	054A	C500	054A
0540	1890	4400	00F2	7400	00EE	70FD	4C80	053B	0001	0003	01DE	031E	03BC	0000	0000	0000
0550	0000	01E4	0324	0324	0324	0324	0099	0000	0113	00CF	009A	0000	009C	00D1	009C	0000
0560	00B9	00D7	0070	07FE	0280	00A5	0072	07FE	03EA	00A9	0078	01E0	0782	00B3	006F	07FE
0570	052B	00A0	0095	0000	0003	00D9	009D	0000	00B9	00D7	008E	0000	011B	00D0	0092	0000
0580	0174	00D5	009B	0000	009C	00D1	0071	07FE	0280	00A7	0001	7C50	032F	0008	00CF	01E0
0590	026B	018C	001F	760C	09F1	0041	00B0	0212	094E	00E4	0099	0000	0113	00CF	0051	0000
05A0	0000	0000	052E	C0D9	90B4	4C20	05AD	C0D9	90B4	4C20	05B4	C09C	7008	C0D3	90AE	4C20
05B0	05B4	C400	0654	7001	1810	D400	007C	4C80	05A2	C400	000F	4C28	05C6	4C08	05C4	C400
05C0	06D8	4400	06B8	7002	4400	069A	1010	D400	000F	C03C	D480	06B3	6500	0FB0	62D8	C100
05D0	1008	E901	D600	0FD8	7102	7201	70F8	C400	0FB2	1890	C400	0FB1	D827	9029	4C20	05E2
05E0	63F2	700E	6500	056E	62E5	63F3	C81D	9E00	0623	4C20	05F9	1090	4C20	05F9	721A	7007
05F0	7400	0071	70D1	C400	06B3	4400	06B8	4F80	0630	7104	7202	7301	70E9	C400	06B3	4400
0600	06B8	6301	4400	06CF	40C3	C5D5	0028	405C	40D1	D6C2	40C5	D1C5	40D7	C1E4	40C3	D7D9
0610	40E3	E8D7	40E3	C5D5	40E7	C5D8	40C4	E4D7	40C1	E2D4	40C6	D6D9	40D9	D7C7	40C3	C5D5
0620	40C3	D6C2	05C4	0664	063B	0641	0630	0649	0649	0655	065B	0661	0661	0661	0630	0661
0630	C902	DC00	0558	C81E	4400	00F2	7400	00EE	70FD	4C00	0516	4400	073B	7400	0036	70FD
0C90	6580	0CE0	6580	0CE0	6680	0CE1	4C80	0C77	0A5C	0AEA	1890	C02B	D06E	62EA	C040	D600
0CA0	0CDF	7201	70FC	1090	4C18	0CB2	7402	0D0B	C037	4400	0CE3	7404	0D0B	C500	0A32	4400
0CB0	0CE3	7002	740A	0D0B	7404	0D0B	C02B	4400	0CE3	7404	0D0B	C025	4400	0CE3	C807	4400
0CC0	01E1	7400	0036	70FD	4C80	0C99	7001	0CC8	0016	4040	F0F0	F0F0	4040	4040	4040	4040
0CD0	F0C5	C4F4	4040	4040	4040	4040	F0C5	C4F4	4040	4040	4040	4040	F0F0	F0F1	4040	4040
0CE0	0000	0001	0ED4	0CBE	18D0	1010	D020	6A1C	62FC	AE00	0D11	901C	4C08	0CF0	E81A	7001
0CF0	8019	7400	0D07	7005	1008	D016	C011	D00F	7007	E812	D480	0D0B	7401	0D0B	1810	D007
0D00	1810	7201	70E6	6600	0000	4C80	0CE3	0000	0009	00C0	00F9	0CDE	F000	1000	0100	0010
0D10	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0D20	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0140	0003	0001	7C50	032F	0008
0D30	0002	11DE	05A2	000B	0003	21DE	05A2	0010	0004	01DE	03C0	0015	0005	41DE	054B	0018
0D40	0006	01DE	03C0	001D	0007	01DE	05A2	0020	0008	01DE	05A2	0025	0009	01DE	0500	002A
0D50	000A	7A06	00DE	002E	000B	7A06	0035	002F	000C	7A06	00D8	0030	000D	7782	087C	0031
0D60	000E	7A06	0248	0038	000F	7A06	0248	003A	0010	7A06	0248	003C	0011	01DE	0280	003E
0D70	0012	0E6E	0140	0040	001F	760C	09F1	0041	0020	7A34	0500	0049	0021	7A34	0280	004D
0D80	0022	7A34	03C0	004F	0023	7A34	0500	0052	0024	7A34	03C0	0056	0025	7A34	0280	0059
0DA0	002A	7A34	03C0	0069	002B	7A34	03C0	006C	002C	7A34	0500	006F	002D	7A34	0500	0073
0DB0	002E	7A34	0500	0077	002F	7A34	0500	007B	0030	7A34	0500	007F	0031	7A34	0404	0083
0DC0	0032	7A34	03C0	0087	0033	7A34	03C0	008A	0034	7A34	0280	008D	0035	7A34	03C0	008F
0DD0	0036	7A34	03C0	0092	0037	7A34	0500	0095	0038	7B8E	03C0	0099	0039	766E	0140	009C
0DE0	006E	04FE	02FE	009D	006F	07FE	052B	00A0	0070	07FE	0280	00A5	0071	07FE	0280	00A7
0DF0	0072	07FE	03EA	00A9	0073	0506	04F8	00AD	0074	0400	0189	00B1	0078	01E0	0782	00B3
0E00	0079	05BC	04E2	00BA	007A	08B6	01E8	00BE	007B	08B6	01E8	00C0	007C	08B6	01E8	00C2
0E10	007D	08B6	01E8	00C4	007E	0AA0	0140	00C6	007F	0AA0	0140	00C7	0080	0AA0	0140	00C8
0E20	0081	0AA0	0140	00C9	0082	0BE2	0140	00CA	0083	08B6	01E8	00CB	0084	0AA0	0140	00CD
0E30	008C	0000	0132	80CE	008D	0000	0113	00CF	008E	0000	011B	00D0	008F	0000	009C	00D1
0E40	0090	0000	00AB	00D2	0091	0000	016C	00D3	0092	0000	0174	00D5	0093	0000	00B9	00D7
0E50	0094	0000	0003	00D8	0095	0000	0003	00D9	0096	00F0	00EC	00DA	0097	00F0	01A2	00DB
0E60	0098	00F0	0280	00DD	0099	0000	0113	00CF	009A	0000	009C	00D1	0140	0001	0000	0000
0E70	0000	0000	0000	0000	0000	0000	0209	0000	0000	0000	0000	0000	0000	0000	0000	0001
0E80	0000	0000	0000	0000	0000	0000	0000	0001	0000	010A	4000	0000	0000	0000	0000	0000
0E90	0000	468A	0000	0000	0000	0000	468A	0000	0000	0000	0000	0469	0000	0000	0000	0000
0EA0	0000	0ED4	0000	0000	0000	0ED4	0000	0000	0000	0000	0378	0000	0000	0000	0000	01E0
0EB0	0000	0000	0000	0000	0000	0000	0000	0000	0000	01E8	0000	0000	0000	0000	0388	0000
0EC0	0000	0000	0000	0000	0000	0000	0000	0000	000D	0000	0000	0000	0000	0000	0000	0000
0ED0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0FA0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0050
0FB0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
1FF0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	2000
2000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
3FF0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	4000
4000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
7FE0	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF
7FF0	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	70FF	8000

The contents of this appendix are not to be construed as an external specification; that is, the locations in this listing may be changed. $PRET, $IREQ, $EXIT, $LINK, and $DUMP are the only locations that are guaranteed.

Note. In the following listing of the resident monitor, = is equivalent to #, and ' (apostrophe) is equivalent to @. The items noted in this listing identify locations discussed throughout the text of this publication.

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

                  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 LAST HOLE IN UA   2-10 PMN00250
                  0026    * 23       * #USZE * BLK CNT OF LAST HOLE IN UA2-10 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       * #LCSW * LOCAL-CANNCT-CALL-LOCAL SW 2-2 PMN00340
                  0035    * 32       * #X3SW * SPECIAL ILS SWITCH       2-2 PMN00350
                  0036    * 33       * #ECNT * NO. OF *EQUAT RCDS       2-4 PMN00360
                  0037    * 33-34    *       * RESERVED FOR FUTURE USE  2-2 PMN00370
                  0038    * 35       * #ANDU * 1+BLOCK ADDR OF END OF USER PMN00380
                  0039    *          *       * AREA (ADJUSTED) LOGICAL DR 0 PMN00390
                  0040    * 36       *       * 1+BLOCK ADDR OF END OF USER PMN00400
                  0041    *          *       * AREA (ADJUSTED) LOGICAL DR 1 PMN00410
                  0042    * 37       *       * 1+BLOCK ADDR OF END OF USER PMN00420
                  0043    *          *       * AREA (ADJUSTED) LOGICAL DR 2 PMN00430
                  0044    * 38       *       * 1+BLOCK ADDR OF END OF USER PMN00440
                  0045    *          *       * AREA (ADJUSTED) LOGICAL DR 3 PMN00450
                  0046    * 39       *       * 1+BLOCK ADDR OF END OF USER PMN00460
                  0047    *          *       * AREA (ADJUSTED) LOGICAL DR 4 PMN00470
                  0048    * 40       * #BNDU * 1+BLOCK ADDR OF END OF USER PMN00480
                  0049    *          *       * AREA (BASE) LOGICAL DRIVE  0 PMN00490
                  0050    * 41       *       * 1+BLOCK ADDR OF END OF USER PMN00500
                  0051    *          *       * AREA (BASE) LOGICAL DRIVE  1 PMN00510
                  0052    * 42       *       * 1+BLOCK ADDR OF END OF USER PMN00520
                  0053    *          *       * AREA (BASE) LOGICAL DRIVE  2 PMN00530
                  0054    * 43       *       * 1+BLOCK ADDR OF END OF USER PMN00540
                  0055    *          *       * AREA (BASE) LOGICAL DRIVE  3 PMN00550
                  0056    * 44       *       * 1+BLOCK ADDR OF END OF USER PMN00560
                  0057    *          *       * AREA (BASE) LOGICAL DRIVE  4 PMN00570
                  0058    * 45       * #FPAD * FILE PROTECT ADDR, LOGICAL  PMN00580
                  0059    *          *       * DRIVE 0 (BASE)              PMN00590
                  0060    * 46       *       * FILE PROTECT ADDR, LOGICAL  PMN00600
                  0061    *          *       * DRIVE 1 (BASE)              PMN00610
                  0062    * 47       *       * FILE PROTECT ADDR, LOGICAL  PMN00620
                  0063    *          *       * DRIVE 2 (BASE)              PMN00630
```

DCOM monitor system parameters

DCOM cartridge parameters

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

                   0064     * 48        *       * FILE PROTECT ADDR, LOGICAL     PMN00640
                   0065     *           *       * DRIVE 3 (BASE)                 PMN00650
                   0066     * 49        *       * FILE PROTECT ADDR,LOGICAL      PMN00660
                   0067     *           *       * DRIVE 4 (BASE)                 PMN00670
                   0068     * 50        * #PCID * CARTRIDGE ID, PHYSICAL DRIVE 0 PMN00680
                   0069     * 51        *       * CARTRIDGE ID, PHYSICAL DRIVE 1 PMN00690
                   0070     * 52        *       * CARTRIDGE ID, PHYSICAL DRIVE 2 PMN00700
                   0071     * 53        *       * CARTRIDGE ID, PHYSICAL DRIVE 3 PMN00710
                   0072     * 54        *       * CARTRIDGE ID, PHYSICAL DRIVE 4 PMN00720
                   0073     * 55        * #CIDN * CARTRIDGE ID, LOGICAL DRIVE 0  PMN00730
                   0074     * 56        *       * CARTRIDGE ID, LOGICAL DRIVE 1  PMN00740
                   0075     * 57        *       * CARTRIDGE ID, LOGICAL DRIVE 2  PMN00750
                   0076     * 58        *       * CARTRIDGE ID, LOGICAL DRIVE 3  PMN00760
                   0077     * 59        *       * CARTRIDGE ID, LOGICAL DRIVE 4  PMN00770
                   0078     * 60        * #CIBA * SCTR ADDR OF CIB, LOGICAL DR 0 PMN00780
                   0079     * 61        *       * SCTR ADDR OF CIB, LOGICAL DR 1 PMN00790
                   0080     * 62        *       * SCTR ADDR OF CIB, LOGICAL DR 2 PMN00800
                   0081     * 63        *       * SCTR ADDR OF CIB, LOGICAL DR 3 PMN00810
                   0082     * 64        *       * SCTR ADDR OF CIB, LOGICAL DR 4 PMN00820
                   0083     * 65        * #SCRA * SCRA, LOGICAL DRIVE 0          PMN00830
                   0084     * 66        *       * SCRA, LOGICAL DRIVE 1          PMN00840
                   0085     * 67        *       * SCRA, LOGICAL DRIVE 2          PMN00850
                   0086     * 68        *       * SCRA, LOGICAL DRIVE 3          PMN00860
                   0087     * 69        *       * SCRA, LOGICAL DRIVE 4          PMN00870
                   0088     * 70        * #FMAT * FORMAT OF PROG IN WS, DRIVE 0  PMN00880
                   0089     * 71        *       * FORMAT OF PROG IN WS, DRIVE 1  PMN00890
                   0090     * 72        *       * FORMAT OF PROG IN WS, DRIVE 2  PMN00900
                   0091     * 73        *       * FORMAT OF PROG IN WS, DRIVE 3  PMN00910
                   0092     * 74        *       * FORMAT OF PROG IN WS, DRIVE 4  PMN00920
                   0093     * 75        * #FLET * FLET SCTR ADDR, LOGICAL DR 0   PMN00930
                   0094     * 76        *       * FLET SCTR ADDR, LOGICAL DR 1   PMN00940
                   0095     * 77        *       * FLET SCTR ADDR, LOGICAL DR 2   PMN00950
                   0096     * 78        *       * FLET SCTR ADDR, LOGICAL DR 3   PMN00960
                   0097     * 79        *       * FLET SCTR ADDR, LOGICAL DR 4   PMN00970
                   0098     * 80        * #ULET * LET SCTR ADDR, LOGICAL DR 0    PMN00980
                   0099     * 81        *       * LET SCTR ADDR, LOGICAL DR 1    PMN00990
                   0100     * 82        *       * LET SCTR ADDR, LOGICAL DR 2    PMN01000
                   0101     * 83        *       * LET SCTR ADDR, LOGICAL DR 3    PMN01010
                   0102     * 84        *       * LET SCTR ADDR, LOGICAL DR 4    PMN01020
                   0103     * 85        * #WSCT * BLK CNT OF PROG IN WS, DRIVE 0 PMN01030
                   0104     * 86        *       * BLK CNT OF PROG IN WS, DRIVE 1 PMN01040
                   0105     * 87        *       * BLK CNT OF PROG IN WS, DRIVE 2 PMN01050
                   0106     * 88        *       * BLK CNT OF PROG IN WS, DRIVE 3 PMN01060
                   0107     * 89        *       * BLK CNT OF PROG IN WS, DRIVE 4 PMN01070
                   0108     * 90        * #CSHN * SCTR CNT CUSHION,LOGICAL DR 0  PMN01080
                   0109     * 91        *       * SCTR CNT CUSHION,LOGICAL DR 1  PMN01090
                   0110     * 92        *       * SCTR CNT CUSHION,LOGICAL DR 2  PMN01100
                   0111     * 93        *       * SCTR CNT CUSHION,LOGICAL DR 3  PMN01110
                   0112     * 94        *       * SCTR CNT CUSHION,LOGICAL DR 4  PMN01120
                   0113     * 95-319    *       * RESERVED FOR FUTURE USE        PMN01130
```

DCOM cartridge parameters

```
RESIDENT MONITOR

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

                   0115     ***************************************************** PMN01150
                   0116     *                                                   * PMN01160
                   0117     *STATUS-VERSION 2, MODIFICATION 10                  * PMN01170
                   0118     *                                                   * PMN01180
                   0119     *FUNCTION/OPERATION-                                * PMN01190
                   0120     *   THIS SECTION ALWAYS REMAINS IN CORE. IT         * PMN01200
                   0121     *   IS COMPRISED OF THE COMMUNICATIONS              * PMN01210
                   0122     *   AREA (COMMA), THE SKELETON SUPERVISOR, AND      * PMN01220
                   0123     *   A DISK I/O SUBROUTINE, NOMINALLY DISKZ. (THE    * PMN01230
                   0124     *   FIRST TWO OF THESE SECTIONS ARE INTERMIXED.)    * PMN01240
                   0125     *   COMMA CONTAINS THE SYSTEM PARAMETERS REQUIR-    * PMN01250
                   0126     *   ED TO FETCH A CORE LOAD IN CORE IMAGE FOR-      * PMN01260
                   0127     *   MAT. THE SKELETON SUPERVISOR PROVIDES IN-       * PMN01270
                   0128     *   STRUCTIONS FOR INITIATING A CALL EXIT, A        * PMN01280
                   0129     *   CALL LINK, A DUMP-TO-PRINTER OR A CALL TO THE   * PMN01290
                   0130     *   AUXILIARY SUPERVISOR.  IN ADDITION, THE SKELE-* PMN01300
                   0131     *   TON SUPERVISOR CONTAINS SEVERAL TRAPS FOR CER-* PMN01310
                   0132     *   TAIN I/O FUNCTIONS/CONDITIONS. THE DISK I/O     * PMN01320
                   0133     *   SECTION CONSISTS OF A SUBROUTINE FOR READING    * PMN01330
                   0134     *   FROM OR WRITING ON A DISK CARTRIDGE ON A        * PMN01340
                   0135     *   GIVEN LOGICAL DISK DRIVE.                       * PMN01350
                   0136     *                                                   * PMN01360
                   0137     *ENTRY POINTS-                                      * PMN01370
                   0138     *   * $PRET-A TRAP FOR PREOPERATIVE I/O ERRORS.     * PMN01380
                   0139     *           THE CALLING SEQUENCE IS                 * PMN01390
                   0140     *              BSI   L  $PRET                       * PMN01400
                   0141     *   * $PSTX-A POSTOPERATIVE ERROR TRAP FOR I/O      * PMN01410
                   0142     *           DEVICES ON LEVEL X (X=1,2,3,OR 4).      * PMN01420
                   0143     *           THE CALLING SEQUENCE IS                 * PMN01430
                   0144     *              BSI  L  $PSTX                        * PMN01440
                   0145     *   * $STOP-THE PROGRAM STOP KEY TRAP.              * PMN01450
                   0146     *   * $EXIT-THE ENTRY POINT FOR THE EXIT/CALL       * PMN01460
                   0147     *          EXIT STATEMENT. THE CALLING SEQUENCE IS* PMN01470
                   0148     *              LDX    0 $EXIT                       * PMN01480
                   0149     *   * $LINK-THE ENTRY POINT FOR THE LINK/CALL       * PMN01490
                   0150     *          LINK STATEMENT. THE CALLING SEQUENCE IS* PMN01500
                   0151     *              BSI  L  $LINK                        * PMN01510
                   0152     *   * $DUMP-THE ENTRY POINT FOR THE DUMP/PDMP       * PMN01520
                   0153     *          STATEMENT. THE CALLING SEQUENCE IS       * PMN01530
                   0154     *              BSI  L  $DUMP                        * PMN01540
```

RESIDENT MONITOR

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

                           *              DC       FORMAT                       * PMN01550
                           *              DC       LIMIT1                       * PMN01560
                           *              DC       LIMIT2                       * PMN01570
                           *          WHERE LIMIT1 AND LIMIT2 ARE THE LIMITS * PMN01580
                           *          BETWEEN WHICH THE DUMP IS TO OCCUR, AND* PMN01590
                           *          FORMAT IS A CODE INDICATING THE FORMAT * PMN01600
                           *          OF THE DUMP. IF FORMAT IS NEGATIVE,     * PMN01610
                           *          THE AUXILIARY SUPERVISOR IS FETCHED     * PMN01620
                           *          AND CONTROL PASSED TO IT.               * PMN01630
                           *   * DZ000-ENTERED WHEN THE CALLER WISHES TO      * PMN01640
                           *           PERFORM A DISK I/O OPERATION. THE      * PMN01650
                           *           CALLING SEQUENCE VARIES WITH THE       * PMN01660
                           *           VERSION OF THE DISK I/O SUBROUTINE.    * PMN01670
                           *   * $I200/$I400-ENTERED WHEN THE OPERATION-      * PMN01680
                           *                 COMPLETE INTERRUPT OCCURS ON     * PMN01690
                           *                 LEVEL 2/4.                       * PMN01700
                           *                                                  * PMN01710
                           *INPUT-N/A                                         * PMN01720
                           *                                                  * PMN01730
                           *OUTPUT-WORDS 6-4090 SAVED ON THE CIB ON A CALL    * PMN01740
                           *        DUMP                                      * PMN01750
                           *                                                  * PMN01760
                           *EXTERNAL REFERENCES-N/A                           * PMN01770
                           *                                                  * PMN01780
                           *EXITS-                                            * PMN01790
                           *   * NORMAL                                       * PMN01800
                           *       *THE EXITS FROM THE SUBROUTINES AT $PRET   * PMN01810
                           *        $PST1, $PST2, $PST3, $PST4, AND $STOP     * PMN01820
                           *        ARE BRANCH INSTRUCTIONS FOLLOWING A       * PMN01830
                           *        WAIT INSTRUCTION. $STOP TURNS OFF IN-     * PMN01840
                           *        TERRUPT LEVEL 5 AFTER THE START KEY IS    * PMN01850
                           *        DEPRESSED.                                * PMN01860
                           *       *THE EXITS FROM $EXIT,$LINK,AND $DUMP ARE * PMN01870
                           *        TO THE CORE IMAGE LOADER, PHASE 1,        * PMN01880
                           *        AFTER THAT PHASE HAS BEEN FETCHED.        * PMN01890
                           *       *THE EXIT FROM DZ000 IS BACK TO THE        * PMN01900
                           *        CALLER AFTER THE REQUESTED DISK OPERA-    * PMN01910
                           *        TION HAS BEEN INITIATED.                  * PMN01920
                           *       *THE EXITS FROM $I200/$I400 ARE BACK TO    * PMN01930
                           *        THE ADDRESSES FROM WHICH THE DISK OP-     * PMN01940
                           *        ERATION COMPLETE INTERRUPT OCCURED        * PMN01950
                           *        AFTER THE INTERRUPT HAS BEEN SERVICED     * PMN01960
                           *        BY THE APPROPRIATE ISS.                   * PMN01970
                           *   * ERROR-N/A                                    * PMN01980
                           *                                                  * PMN01990
                           *TABLES/WORK AREAS-                                * PMN02000
                           *   * $ACDE                                        * PMN02010
                           *   * $CH12                                        * PMN02020
                           *   * $CILA                                        * PMN02030
                           *   * $CLSW                                        * PMN02040
                           *   * $COMN                                        * PMN02050
                           *   * $CORE                                        * PMN02060
                           *   * $CTSW                                        * PMN02070
                           *   * $CXR1                                        * PMN02080
                           *   * $CYLN                                        * PMN02090
                           *   * $DADR                                        * PMN02100
                           *   * $DBSY                                        * PMN02110
                           *   * $DCYL                                        * PMN02120
                           *   * $DMPF                                        * PMN02130
                           *   * $DREQ                                        * PMN02140
                           *   * $FPAD                                        * PMN02150
                           *   * $GCOM                                    2G2 * PMN02160
                           *   * $GRIN                                    2G2 * PMN02170
                           *   * $HASH                                        * PMN02180
                           *   * $IBT2                                        * PMN02190
                           *   * $IBT4                                        * PMN02200
                           *   * $IBSY                                        * PMN02210
                           *   * $IOCT                                        * PMN02220
                           *   * $KCSW                                        * PMN02230
                           *   * $LAST                                        * PMN02240
                           *   * $LNXQ                                    2-9 * PMN02250
                           *   * $NDUP                                        * PMN02260
                           *   * $NXEQ                                        * PMN02270
                           *   * $PBSY                                        * PMN02280
                           *   * $PGCT                                        * PMN02290
                           *   * $PHSE                                        * PMN02300
                           *   * $RMSW                                        * PMN02310
                           *   * $SCAT                                    2-4 * PMN02320
                           *   * $SNLT                                        * PMN02330
                           *   * $UFIO                                        * PMN02340
                           *   * $ULET                                        * PMN02350
                           *   * $WRD1                                        * PMN02360
                           *   * $WSDR                                        * PMN02370
                           *   * $XR3X                                    2-2 * PMN02380
                           *                                                  * PMN02390
                           *ATTRIBUTES-REUSABLE                               * PMN02400
                           *                                                  * PMN02410
                           *NOTES-                                            * PMN02420
                           *    THERE ARE WAIT INSTRUCTIONS AT $PRET+1,       * PMN02430
                           *    $STOP+1, AND $PSTX+1. DEPRESSING THE START    * PMN02440
                           *    KEY WILL RETURN CONTROL TO THE CALLER IN ALL  * PMN02450
                           *    CASES.                                        * PMN02460
```

```
RESIDENT MONITOR

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

                   0247    ***************************************************** PMN02470
                   0249    * PROVIDE PARAMETERS FOR SYSTEM LOADER                PMN02490
                   0250    *                                                     PMN02500
                   0251          ABS                                             PMN02510
0280               0252          ORG     4                                       PMN02520
0004 0  0FFA       0253          DC      4095-* WD CNT FOR WRITING CORE ON CIB   PMN02530
0005 0  0000       0254    $CIBA DC      *-*  SCTR ADDR OF THE CIB               PMN02540
0006 0  0000       0255    $CH12 DC      *-*  ADDR OF CHANNEL 12 INDICATOR       PMN02550
0007 0  0000       0256    $COMN DC      *-*  LENGTH OF COMMON (IN WORDS)        PMN02560
                   0257    *                                                     PMN02570
                   0258    * ULTIMATE RESIDENCE OF THE INTERRUPT TV              PMN02580
                   0259    *                                                     PMN02590
0008 0  0000       0260    $LEV0 DC      *-*   LEVEL 0 BRANCH ADDRESS            PMN02600
0009 0  0000       0261    $LEV1 DC      *-*   LEVEL 1 BRANCH ADDRESS            PMN02610
000A 0  00B3       0262    $LEV2 DC      $I200 LEVEL 2 BRANCH ADDR               PMN02620
000B 0  0000       0263    $LEV3 DC      *-*   LEVEL 3 BRANCH ADDRESS            PMN02630
000C 0  00C4       0264    $LEV4 DC      $I400 LEVEL 4 BRANCH ADDR               PMN02640
000D 0  0091       0265    $LEV5 DC      $STOP LEVEL 5 BRANCH ADDR               PMN02650
                   0266    *                                                     PMN02660
                   0267    *                                                     PMN02670
000E 0  0000       0268    $CORE DC      *-*  SIZE OF CORE, E.G., /1000=4K       PMN02680
000F 0  0000       0269    $CTSW DC      *-*  CONTROL RECORD TRAP SWITCH         PMN02690
0010 0  0000       0270    $DADR DC      *-*  SCTR ADDR OF PROG TO BE LOADED     PMN02700
0011 0  0000       0271    $SCAT DC      *-*  NON ZERO=SCA INTRPT PNDNG     2-4  PMN02710   ——$SCAT
0012 0  0000       0272    $DREQ DC      *-*  IND. FOR REQUESTED VERSION DKI/O   PMN02720
0013 0  0000       0273    $IBSY DC      *-*  NON-ZERO IF CD/PAP TP DEV. BUSY    PMN02730
0014    000C       0274    $HASH BSS  E  12   WORK AREA                          PMN02740
                   0275    *                                                     PMN02750
                   0276    *                                                     PMN02760
0020    0008       0277    $SCAN BSS     8    1132 SCAN AREA                  32 PMN02770
                   0278    *                                                     PMN02780
                   0279    *                                                     PMN02790
                   0280    *                                                     PMN02800
                   0281    * TRAP FOR PREOPERATIVE I/O ERRORS                    PMN02810
                   0282    *                                                     PMN02820
0028 0  0000       0283    $PRET DC      *-*       ENTRY POINT                   PMN02830   ——$PRET
0029 0  3000       0284          WAIT              WAIT TIL START KEY PUSHED     PMN02840
002A 00 4C800028   0285          BSC  I  $PRET     RETURN TO CALLER              PMN02850
                   0286    *                                                     PMN02860
                   0287    *                                                     PMN02870
002C 0  0000       0288    $IREQ DC      *-*  ADDR OF INT REQUEST SUBROUTINE     PMN02880
002D 0  0000       0289    $ULET DC      *-*  ADDR OF LET, LOGICAL DR 0          PMN02890   }
002E 0  0000       0290          DC      *-*  ADDR OF LET, LOGICAL DR 1          PMN02900   }
002F 0  0000       0291          DC      *-*  ADDR OF LET, LOGICAL DR 2          PMN02910   }LET addresses
0030 0  0000       0292          DC      *-*  ADDR OF LET, LOGICAL DR 3          PMN02920   }
0031 0  0000       0293          DC      *-*  ADDR OF LET, LOGICAL DR 4          PMN02930   }
0032 0  0000       0294    $IOCT DC      *-*  ZERO IF NO I/O IN PROGRESS      50 PMN02940   ——$IOCT
0033 0  0000       0295    $LAST DC      *-*  NON-ZERO WHEN LAST CARD SENSED     PMN02950
0034 0  0000       0296    $NDUP DC      *-*  DO NOT DUP IF NON-ZERO             PMN02960
0035 0  0000       0297    $NXEQ DC      *-*  DO NOT EXECUTE IF NON-ZERO         PMN02970
0036 0  0000       0298    $PBSY DC      *-*  NON-ZERO WHEN PRINTER BUSY         PMN02980
0037 0  0000       0299    $PGCT DC      *-*  PAGE NO. FOR HEADINGS              PMN02990
                   0300    *                                                     PMN03000
                   0301    * CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR        PMN03010
                   0302    *                                                     PMN03020
0038 0  7019       0303    $EXIT MDX     $S000     BR TO FETCH CIL, PHASE 1 56   PMN03030   ——$EXIT
                   0304    *                                                     PMN03040
                   0305    *** CALL LINK ENTRY POINT                             PMN03050
                   0306    *                                                     PMN03060
0039 0  0000       0307    $LINK DC      *-*       ENTRY POINT                57 PMN03070   ——$LINK
003A 0  1810       0308          SRA     16                                      PMN03080
003B 0  7017       0309          MDX     $S100     BR TO FETCH CIL, PHASE 1      PMN03090
003C    0000       0310          BSS  E  0                                       PMN03100
003C 0  0001       0311    $S900 DC      1    DISK PARAMETERS FOR SAVING CORE    PMN03110
                   0312    *$S900 ALSO USED AS CONSTANT 1 BY CIL PH2        2-10 PMN03111
003D 0  0004       0313          DC      $CIBA-1 *IN CONNECTION WITH DUMP        PMN03120
003F 0  FFFF       0314    $S910 DC      -1   CALL EXIT INDICATOR                PMN03130
                   0315    *$S910 ALSO USED AS CONSTANT-1 BY CIL PH2        2-10 PMN03131
                   0316    *                                                     PMN03140
                   0317    *** SAVE 1ST 4K OF CORE ON THE CIB                    PMN03150
                   0318    *                                                     PMN03160
003F 0  0000       0319    $DUMP DC      *-*       ENTRY POINT                63 PMN03170   ——$DUMP
0040 0  D809       0320          STD     $ACEX     SAVE ACCUMULATOR, EXTENSION   PMN03180
0041 0  4023       0321          BSI     $S250     CHK PNDNG INTRPT          2-4 PMN03190
0042 0  282A       0322          STS     $SSTS     SAVE STATUS               2-6 PMN03200
0043 0  6903       0323          STX    1 $CXR1    SAVE XR1                      PMN03210
0044 00 C480003F   0324          LD   I  $DUMP                                   PMN03220
0046 0  D002       0325          STO     $DMPF     SAVE DUMP FORMAT CODE         PMN03230
0047 0  C8F4       0326          LDD     $S900                                   PMN03240
0048 00 44000CF2   0327          BSI  L  DZ000     SAVE WDS 6-4095 ON CIB        PMN03250
004A 0  C0F1       0328          LD      $S900                                   PMN03260
004B 0  70C7       0329          MDX     $S100     BR TO FETCH CIL, PHASE 1      PMN03270
                   0330    *                                             2-6     PMN03280
004C    0006       0331          BSS     6         PATCH AREA                2-7 PMN03290
                   0333    *                                                     PMN03310
                   0334    *** FETCH CORE IMAGE LOADER, PHASE 1                  PMN03320
                   0335    *                                                     PMN03330
0052 0  C0EB       0336    $S000 LD      $S910                                   PMN03340
0053 0  D0C2       0337    $S100 STO     $RMSW     SAVE EXIT-LINK-DUMP SWITCH    PMN03350
0054 00 65800039   0338          LDX  I1 $LINK     LINK ADDR TO XR1              PMN03360
0056 0  C101       0339          LD     1 1        FETCH 2ND WD OF LINK NAME     PMN03370
```

```
RESIDENT MONITOR

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

0057 0  18D0       0340          RTE     16                                     PMN03380
0058 0  C100       0341          LD    1 0          FETCH 1ST WD OF LINK NAME   PMN03390
                   0342    * $S150+1 CONTAINS ADDR LAST WD OF DISK I/O MINUS 3  PMN03400
0059 00 65C00000   0343    $S150 LDX  L1 *-*        ADDR END OF DKI/O-1 TO XR1  PMN03410
005B 0  D888       0344          STD     $LKNM      SAVE LINK NAME              PMN03420
005C 0  40C8       0345          BSI     $S250      CHK ANY PNDNG INTRPT    2-4 PMN03430
005D 0  C0FC       0346          LD      $CILA                                  PMN03440
005E 0  1890       0347    $S200 SRT     16                                     PMN03450
005F 00 44000CF2   0348          BSI   L DZ000      FETCH CI LOADER, PHASE 1    PMN03460
0061 0  4003       0349          BSI     $S250      CHK DISK OP FINISHED    2-4 PMN03470
0062 0  4102       0350          BSI   1 2          BR TO CI LOADER, PHASE 1    PMN03480
                   0351    *                                                    PMN03490
0063 0  0000       0352    $GCOM DC      *-*  GRAPHIC SUBR PACKAGE INDR     2G2 PMN03500
0064 0  0000       0353    $GRIN DC      *-*  GRAPHIC INITLZN PROGRAM INDR  2G2 PMN03510
                   0354    *                                                    PMN03520
                   0355    *** SUBR TO CHECK IF ANY INTRPT IS PENDING           PMN03530
                   0356    *                                                    PMN03540
0065 0  0000       0357    $S250 DC      *-*        ENTRY POINT                 PMN03550
0066 0  C0C8       0358    $S300 LD      $IOCT      IS THERE INTRPT PNDNG       PMN03560
0067 0  E8A9       0359          OR      $SCAT      *OR SCA INTRPT PNDNG        PMN03570
0068 00 4C200066   0360          BSC   L $S300,Z    *THEN BR,IF ALL INTRPT      PMN03580
006A 0  0803       0361          XIO     $1499      RESET 2250              2-7 PMN03590
006B 00 4C800065   0362          BSC   I $S250      *IS SERVICED-RETURN         PMN03600
                   0363    *                                                2-6 PMN03610
006D 0  2000       0364    $SSTS LDS     *-*        STATUS SAVED FOR DUMP   2-7 PMN03620
006E 0  0C00       0365    $1499 DC      0          IOCC FOR RESET          2-7 PMN03630
006F 0  CC80       0366          DC      /CC80      *OF 2250                2-7 PMN03640
0070 0  0000       0367    $LNXC DC      *-*  LINK/XEQ SW, -1 LINK,+1 XEQ   2-9 PMN03650
                   0368    *                                                    PMN03660
0071 0  0000       0369    $FLSH DC      *-*  FLUSH-TO-NEXT-JOB SWITCH 1#FLUSH PMN03670
0072    0000       0370          BSS   E 0                                      PMN03680
0072 0  0000       0371    $CWCT DC      *-*  WORD COUNT AND SECTOR ADDRESS     PMN03690
0073 0  0000       0372          DC      *-*  *FOR SAVING/RESTORING COMMON      PMN03700
0074 0  0000       0373    $CCAD DC      *-*  ADDR FOR SAVING/RESTORING COMMON  PMN03710
0075 0  0000       0374    $LSAD DC      *-*  SCTR ADDR OF 1ST LOCAL/SOCAL      PMN03720
0076 0  0000       0375    $DZIN DC      *-*  DISKZ/1/N INDICATOR (-1,0,+1)     PMN03730
0077 0  0000       0376    $DCDE DC      *-*  LOGICAL DRIVE CODE FOR PROGRAM    PMN03740
0078 0  0000       0377    $PHSE DC      *-*  NO. OF PHASE NOW IN CORE          PMN03750
0079 0  0000       0378    $UFIC DC      *-*  UNFORMATTED I/O RECORD NO.        PMN03760
007A 0  0000       0379    $WSDR DC      *-*  WORKING STORAGE DRIVE CODE        PMN03770
007B 0  0000       0380    $WRD1 DC      *-*  LOADING ADDR OF THE CORE LOAD     PMN03780
007C 0  0000       0381    $KCSW DC      *-*  1 IF KB,CP BOTH UTILIZED          PMN03790
007D 0  0000       0382    $UFDR DC      *-*  UNFORMATTED I/O DRIVE CODE        PMN03800
007E 0  0000       0383    $CPTR DC      *-*  CHANNEL 12 INDICATOR FOR CP       PMN03810
007F 0  0000       0384    $1132 DC      *-*  CHANNEL 12 INDICATOR FOR 1132     PMN03820
0080 0  0000       0385    $1403 DC      *-*  CHANNEL 12 INDICATOR FOR 1403     PMN03830
                   0387    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 1       PMN03850
                   0388    *                                                    PMN03860
0081 0  0000       0389    $PST1 DC      *-*        ENTRY POINT                 PMN03870   —$PST1
0082 0  3000       0390          WAIT                                           PMN03880
0083 00 4C800081   0391          BSC   I $PST1      RETURN TO DEVICE SUBROUTINE PMN03890
                   0392    *                                                    PMN03900
                   0393    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 2       PMN03910
                   0394    *                                                    PMN03920
0085 0  0000       0395    $PST2 DC      *-*        ENTRY POINT                 PMN03930   —$PST2
0086 0  3000       0396          WAIT                                           PMN03940
0087 00 4C800085   0397          BSC   I $PST2      RETURN TO DEVICE SUBROUTINE PMN03950
                   0398    *                                                    PMN03960
                   0399    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 3       PMN03970
                   0400    *                                                    PMN03980
0089 0  0000       0401    $PST3 DC      *-*        ENTRY POINT                 PMN03990   —$PST3
008A 0  3000       0402          WAIT                                           PMN04000
008B 00 4C800089   0403          BSC   I $PST3      RETURN TO DEVICE SUBROUTINE PMN04010
                   0404    *                                                    PMN04020
                   0405    * TRAP FOR POSTOPERATIVE I/O ERRORS ON LEVEL 4       PMN04030
                   0406    *                                                    PMN04040
008D 0  0000       0407    $PST4 DC      *-*        ENTRY POINT                 PMN04050   —$PST4
008E 0  3000       0408          WAIT                                           PMN04060
008F 00 4C80008D   0409          BSC   I $PST4      RETURN TO DEVICE SUBROUTINE PMN04070
                   0410    *                                                    PMN04080
                   0411    *                                                    PMN04090
                   0412    * PROGRAM STOP KEY TRAP                              PMN04100
                   0413    *                                                    PMN04110
0091 0  0000       0414    $STOP DC      *-*        ENTRY POINT                 PMN04120   —$STOP
0092 0  3000       0415          WAIT               WAIT TIL START KEY PUSHED   PMN04130
0093 00 4CC00091   0416          BOSC  I $STOP      RETURN TO CALLER            PMN04140
                   0418    *                                                    PMN04160
                   0419    * PARAMETERS USED BY THE DISK I/O SUBROUTINES. THE   PMN04170
                   0420    * LOGICAL DRIVE CODE IS FOUND IN BITS 1-3 FOR ALL    PMN04180
                   0421    * BUT THE AREA CODE. BIT 0 WILL ALWAYS BE ZERO.      PMN04190
                   0422    *                                                    PMN04200
                   0423    *                                                    PMN04210
                   0424    *** DISK1 AND DISKN WILL NOT WRITE BELOW THE         PMN04220
                   0425    *** FOLLOWING SCTR ADDRESSES (EXCEPT WRITE IMMED).   PMN04230
                   0426    *                                                    PMN04240
0095 0  0000       0427    $FPAD DC      *-*  FILE PROTECT ADDR, LOGICAL DR 0   PMN04250
0096 0  0000       0428          DC      *-*  FILE PROTECT ADDR, LOGICAL DR 1   PMN04260
0097 0  0000       0429          DC      *-*  FILE PROTECT ADDR, LOGICAL DR 2   PMN04270
0098 0  0000       0430          DC      *-*  FILE PROTECT ADDR, LOGICAL DR 3   PMN04280
0099 0  0000       0431          DC      *-*  FILE PROTECT ADDR, LOGICAL DR 4   PMN04290
                   0432    *                                                    PMN04300
```

```
RESIDENT MONITOR

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

                    0433    *** THE ARM POSITION IS UPDATED WHENEVER A SEEK     PMN04310
                    0434    *** OCCURS.                                         PMN04320
                    0435    *                                                   PMN04330
009A 0  0000        0436    $CYLN DC      0    ARM POSITION FOR LOGICAL DRIVE 0 PMN04340
009B 0  0000        0437          DC      0    ARM POSITION FOR LOGICAL DRIVE 1 PMN04350
009C 0  0000        0438          DC      0    ARM POSITION FOR LOGICAL DRIVE 2 PMN04360
009D 0  0000        0439          DC      0    ARM POSITION FOR LOGICAL DRIVE 3 PMN04370
009E 0  0000        0440          DC      0    ARM POSITION FOR LOGICAL DRIVE 4 PMN04380
                    0441    *                                                   PMN04390
                    0442    *** BELOW ARE THE DISK AREA CODES. A ZERO           PMN04400
                    0443    *** INDICATES THE CORRESPONDING DRIVE IS NOT        PMN04410
                    0444    *** ON THE SYSTEM                                   PMN04420
                    0445    *                                                   PMN04430
009F 0  0000        0446    $ACDE DC      *-*  AREA  CODE FOR LOGICAL DRIVE 0   PMN04440
00A0 0  0000        0447          DC      *-*  AREA  CODE FOR LOGICAL DRIVE 1   PMN04450
00A1 0  0000        0448          DC      *-*  AREA  CODE FOR LOGICAL DRIVE 2   PMN04460
00A2 0  0000        0449          DC      *-*  AREA  CODE FOR LOGICAL DRIVE 3   PMN04470
00A3 0  0000        0450          DC      *-*  AREA  CODE FOR LOGICAL DRIVE 4   PMN04480
                    0451    *                                                   PMN04490
                    0452    *** THE ADR OF THE CYLINDER IN WHICH A DEFECT OC-   PMN04500
                    0453    *** CURS, IF ANY, IS STORED IN THE 1ST, 2ND, OR 3RD PMN04510
                    0454    *** WORD BELOW, DEPENDING ON WHETHER IT IS THE 1ST, PMN04520
                    0455    *** 2ND, OR 3RD DEFECT ON THE CARTRIDGE.            PMN04530
                    0456    *                                                   PMN04540
00A4 0  0000        0457    $DCYL DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1 PMN04550
00A5 0  0000        0458          DC      *-*  *FOR LOGICAL DRIVE 0           2 PMN04560
00A6 0  0000        0459          DC      *-*                                 3 PMN04570
00A7 0  0000        0460          DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1 PMN04580
00A8 0  0000        0461          DC      *-*  *FOR LOGICAL DRIVE 1           2 PMN04590
00A9 0  0000        0462          DC      *-*                                 3 PMN04600
00AA 0  0000        0463          DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1 PMN04610
00AB 0  0000        0464          DC      *-*  *FOR LOGICAL DRIVE 2           2 PMN04620
00AC 0  0000        0465          DC      *-*                                 3 PMN04630
00AD 0  0000        0466          DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1 PMN04640
00AE 0  0000        0467          DC      *-*  *FOR LOGICAL DRIVE 3           2 PMN04650
00AF 0  0000        0468          DC      *-*                                 3 PMN04660
00B0 0  0000        0469          DC      *-*  DEFECTIVE CYLINDER ADDRESSES   1 PMN04670
00B1 0  0000        0470          DC      *-*  *FOR LOGICAL DRIVE 4           2 PMN04680
00B2 0  0000        0471          DC      *-*                                 3 PMN04690
                    0473    *                                                   PMN04710
                    0474    * ILS02--THIS SUBROUTINE SAVES XR1, XR2, STATUS,    PMN04720
                    0475    *        AND THE ACCUMULATOR AND ITS EXTENSION.     PMN04730
                    0476    *        THE ADDRESS OF THE INTERRUPT SERVICE ROU-  PMN04740
                    0477    *        TINE IS STORED IN $I205 BY PHASE 2 OF      PMN04750
                    0478    *        THE CORE IMAGE LOADER. WORD 10 ALWAYS      PMN04760
                    0479    *        CONTAINS THE ADDRESS OF $I200.             PMN04770
                    0480    *                                                   PMN04780
                    0481    *                                                   PMN04790
                    0482    *                                                   PMN04800
00B3 0  0000        0483    $I200 DC      *-*       ENTRY PT (LEVEL 2 INTRUPT)  PMN04810
00B4 0  6906        0484          STX   1 $I210+1   SAVE XR1                    PMN04820
00B5 0  6A07        0485          STX   2 $I210+3   SAVE XR2                    PMN04830
00B6 0  2807        0486          STS     $I210+4   STORE STATUS                PMN04840
00B7 0  D8CA        0487          STD     $I290     SAVE ACCUMULATOR,EXTENSION  PMN04850
                    0488    * $I205+1 CONTAINS ADDR INTERRUPT ENTRY PT TO DKI/O PMN04860
00B8 00 44C00000    0489    $I205 BSI   L *-*       BR TO SERVICE THE INTERRUPT PMN04870
00BA 00 65000000    0490    $I210 LDX  L1 *-*       RESTORE XR1                 PMN04880
00BC 00 66000000    0491          LDX  L2 *-*       RESTORE XR2                 PMN04890
00BE 0  2000        0492          LDS     0         RESTORE STATUS              PMN04900
00BF 0  C802        0493          LDD     $I290     RESTORE ACCUMULATOR,EXT     PMN04910
00C0 00 4C0000B3    0494          BOSC  I $I200     RETURN FROM INTERRUPT       PMN04920
00C2    000C        0495    $I290 BSS   E 0                                     PMN04930
00C2 0  0000        0496          DC      *-*       CONTENTS OF ACCUMULATOR AND PMN04940
00C3 0  0000        0497          DC      *-*       *EXTENTION                  PMN04950
                    0499    *                                                   PMN04970
                    0500    * ILS04--THIS SUBROUTINE SAVES XR1, XR2, STATUS,    PMN04980
                    0501    *        AND THE ACCUMULATOR AND ITS EXTENSION.     PMN04990
                    0502    *        IF THE INTERRUPT IS FOR A KEYBOARD REQ-  * PMN05000
                    0503    *        UEST,AND IF A MONITOR PROGRAM IS IN CON- * PMN05010
                    0504    *        TROL, CONTROL IS PASSED TO DUMP. OTHER-  * PMN05020
                    0505    *        WISE, CONTROL IS PASSED TO THE KEYBOARD/ * PMN05030
                    0506    *        CONSOLE PRINTER SUBROUTINE. WORD 12 AL-  * PMN05040
                    0507    *        WAYS CONTAINS THE ADDRESS OF $I400.      * PMN05050
                    0508    *                                                   PMN05060
                    0509    * THE TABLE BELOW CONTAINS THE ADDRESSES OF THE     PMN05070
                    0510    * INTERRUPT SERVICE ROUTINES FOR ALL THE DEVICES    PMN05080
                    0511    * ON LEVEL 4.                                       PMN05090
                    0512    *                                                   PMN05100
                    0513    *                                                   PMN05110
                    0514    *                                                   PMN05120
00C4 0  0000        0515    $I400 DC      *-*       ENTRY POINT                 PMN05130
00C5 0  D818        0516          STD     $I490     SAVE ACCUMULATOR, EXTENSION PMN05140
00C6 0  280E        0517          STS     $I41C     SAVE STATUS                 PMN05150
00C7 0  690F        0518          STX   1 $I410+2   SAVE XR1                    PMN05160
00C8 0  6A10        0519          STX   2 $I410+4   SAVE XR2                    PMN05170
00C9 0  0816        0520          XIO     $I492     SENSE DSW                   PMN05180
00CA 0  1002        0521          SLA     2         IS THIS INTERRUPT REQUEST   PMN05190
00CB 00 4C10000D    0522          BSC   L $I403,-   BR IF NOT INTERRUPT REQUEST PMN05200
00CD 00 4480002C    0523          BSI   I $IREQ     BR IF INTERRUPT REQUEST     PMN05210
00CF 0  FFFE        0524          DC      -2        ERROR CODE                  PMN05220
00D0 0  6109        0525    $I403 LDX   1 9         NO. DEVICES ON LEVEL TO XR1 PMN05230
```

```
RESIDENT MONITOR

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

0001 0  0810       0526             XIO      $I494       SENSE ILSW                  PMN05240
0002 0  114C       0527             SLCA  1              FIND CAUSE OF INTERRUPT     PMN05250
                   0528       * $I405+1 CONTAINS ADDR OF LEVEL 4 IBT MINUS 1         PMN05260
0003 00 458C00C0   0529       $I405 BSI  I1 *-*          BR TO SERVICE THE INTERRUPT PMN05270
0005 0  2000       0530       $I410 LDS      0           RESTORE STATUS              PMN05280
0006 00 65000000   0531             LDX  L1 *-*          RESTORE XR1                 PMN05290
0008 00 66C00000   0532             LDX  L2 *-*          RESTORE XR2                 PMN05300
000A 0  C803       0533             LDD      $I490       RESTORE ACCUMULATOR, EXT.   PMN05310
000B 00 4CC000C4   0534             BOSC I  $I400        RETURN                      PMN05320
                   0535       *                                                      PMN05330
                   0536       * CONSTANTS AND WORK AREAS                             PMN05340
                   0537       * EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES          PMN05350
                   0538       *                                                      PMN05360
000D 0  0000       0539       $DDSW DC       *-*         DSW FOR THE DISK            PMN05370
000E    0002       0540       $I490 BSS  E  2    CONTENTS OF ACCUMULATOR, EXT.       PMN05380
00E0 0  0000       0541       $I492 DC       *-*                                     PMN05390
00E0 0             0542       $SYSC EQU      *-1  VERSION AND MOD NO.                PMN05400
00E1 0  0F00       0543             DC       /0F00 IOCC FOR SENSE IOCC FOR KB/CP     PMN05410
00E2    0001       0544       $I494 BSS      1     PATCH AREA                        PMN05420
00E3 0  0300       0545             DC       /0300 IOCC FOR SENSING ILSW04           PMN05430
                   0547                                                          2-2 PMN05450
                   0548       *                                                  2-2 PMN05460
00E4 0  0000       0549       $I496 DC       *-*          XR3 SETTING DURING XEQ 2-2 PMN05470
00E5 0  0F01       0550             DC       /0F01        SENSE KEY BOARD W RESET2-2 PMN05480
                   0551       *                                                  2-2 PMN05490
00E6 0  0000       0552       $I420 DC       *-*          ENTRY POINT FLUSH JOB  2-2 PMN05500
00E7 0  08FC       0553             XIO      $I496        SENSE KEY BOARD W RESET2-2 PMN05510
00E8 00 4C400CFA   0554             BOSC L   $I425        TURN OF INTERRUPT      2-2 PMN05520
00EA 00 440C003F   0555       $I425 BSI  L   $DUMP       BR TO $DUMP             2-7 PMN05530
00EC 0  FFFF       0556             DC       -2          CALLING AUX SUP         2-7 PMN05540
                   0557       *                                                  2-2 PMN05550
00ED    0001       0558             BSS      1           PATCH AREA              2-7 PMN05560
00FF 0  0000       0559       $CBSY DC       *-*  NON-ZERO WHEN DISK I/O BUSY        PMN05570   ——$DBSY


DISKZ

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

                   0561       ****************************************************** PMN05590
                   0562       *                                                    * PMN05600
                   0563       *STATUS - VERSION 2, MODIFICATION 11                 * PMN05610
                   0564       *                                                    * PMN05620
                   0565       *PROGRAM NAME-                                       * PMN05630
                   0566       *   *FULL NAME-FORTRAN/SYSTEM DISK I/O SUBROUTINE    * PMN05640
                   0567       *   *CALLING SEQUENCE-                               * PMN05650
                   0568       *      LDD     PARAM                                 * PMN05660
                   0569       *      BSI  L  DZ000                                 * PMN05670
                   0570       *    WHERE PARAM IS THE LABEL OF A DOUBLE-WORD       * PMN05680
                   0571       *    CELL CONTAINING THE FUNCTION CODE AND THE       * PMN05690
                   0572       *    ADDR OF THE I/O BUFFER,I.E., ADDR OF WD CNT.    * PMN05700
                   0573       *    SEE 'CAPABILITIES' FOR DISCUSSION OF PARAM-     * PMN05710
                   0574       *    ETERS.                                          * PMN05720
                   0575       *                                                      PMN05730
                   0576       *PURPOSE-                                            * PMN05740
                   0577       *   TO PROVIDE A SUBROUTINE TO PERFORM DISK OPERA-   * PMN05750
                   0578       *   TIONS. THIS SUBROUTINE IS INTENDED FOR USE BY    * PMN05760
                   0579       *   MONITOR PROGRAMS AND USER PROGRAMS WRITTEN IN    * PMN05770
                   0580       *   FORTRAN. THUS,IT IS INTENDED FOR USE IN AN       * PMN05780
                   0581       *   ERROR-FREE ENVIRONMENT.                          * PMN05790
                   0582       *                                                    * PMN05800
                   0583       *METHOD-                                             * PMN05810
                   0584       *   DISKZ REQUIRES A BUFFER,THE LENGTH OF WHICH IS   * PMN05820
                   0585       *   2 GREATER THAN THE NO. WORDS TO BE READ/WRIT-    * PMN05830
                   0586       *   TEN.                                             * PMN05840
                   0587       *                                                    * PMN05850
                   0588       *CAPABILITIES AND LIMITATIONS-                       * PMN05860
                   0589       *   THE WD CNT,AS WELL AS DZ000,MUST BE ON AN EVEN   * PMN05870
                   0590       *   BOUNDARY,MUST BE IN THE RANGE 0-32767. THE       * PMN05880
                   0591       *   DRIVE CODE MUST BE IN BITS 1-3 OF THE SECTOR     * PMN05890
                   0592       *   ADDR,WHICH FOLLOWS THE WD CNT. THE FUNCTION      * PMN05900
                   0593       *   INDICATOR MUST BE XX00 FOR A READ OR XX01 FOR    * PMN05910
                   0594       *   A WRITE,WHERE 'XX' MEANS ANY 2 HEXADECIMAL       * PMN05920
                   0595       *   CHARACTERS. A WD CNT OF ZERO INDICATES A SEEK.   * PMN05930
                   0596       *   (READ OR WRITE MAY BE INDICATED.) AUTOMATIC      * PMN05940
                   0597       *   SEEKING IS PROVIDED AS A PART OF READ/WRITE.     * PMN05950
                   0598       *   A WRITE IS ALWAYS WITH A READ-BACK-CHECK.        * PMN05960
                   0599       *   DISKZ MAKES NO PREOPERATIVE PARAMETER CHECKS.    * PMN05970
                   0600       *                                                    * PMN05980
                   0601       *SPECIAL FEATURES-                                   * PMN05990
                   0602       *   DISKZ PROVIDES ONLY THOSE FUNCTIONS MENTIONED    * PMN06000
                   0603       *   ABOVE. DISK1 AND DISKN OFFER THIS BASIC SET OF   * PMN06010
                   0604       *   FUNCTIONS PLUS OTHERS.                           * PMN06020
                   0605       *                                                    * PMN06030
                   0606       ****************************************************** PMN06040
                   0608       * PROVIDE PARAMETERS FOR SYSTEM LOADER                 PMN06060
                   0609       *                                                      PMN06070
00F0    0000       0610             BSS  E  0                                        PMN06080
00F0 0  00FF       0611             DC       $ZEND-*     DISKZ WORD COUNT            PMN06090
00F1 0  FF6A       0612             DC       -'DZID      PHASE ID                    PMN06100
00F2 0  00F8       0613             DC       $ZEND-6-*+1 ADDR OF SLET EXTRACT        PMN06110
00F3 0  0001       0614             DC       1           NO. ENTRIES IN SLET EXTRACT PMN06120
```

```
DISKZ

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

00F4               0615         ORG     *-2                                      PMN06130
0CF2 0  0000       0617   DZ000 DC      *-*       ENTRY POINT                    PMN06150
00F3 00 74CCCCFF   C618         MDX  L  $DBSY,0   LOOP UNTIL OPERATION IN        PMN06160
00F5 0  70FC       0619         MDX     *-3       *PROGRESS IS COMPLETE          PMN06170
00F6 C  70C2       C620         MDX     DZ020     BR AROUNC INT ENTRY POINT      PMN06180
                   C621   *                                                      PMN06190
                   0622   * INTERRUPT ENTRY POINT                                PMN06200
                   C623   *                                                      PMN06210
00F7 0  000C       0624   DZC10 DC      *-*       INTERRUPT ADDRESS              PMN06220
00F8 0  7C18       0625         MDX     DZ180     BR TO SERVICE INTERRUPT        PMN06230
0CF9 0  690B       C626   DZC2C STX   1 DZ100+1   SAVE XR1                       PMN06240
00FA 0  6ACC       C627         STX   2 DZ10C+3   SAVE XR2                       PMN06250
00FB 0  1008       0628         SLA     8         SHIFT INDICATOR 8 BITS         PMN06260
00FC 0  DC3C       0629         STC     DZ945     SAVE FUNCTION INDICATOR        PMN06270
00FD 0  18D0       063C         RTE     16                                       PMN06280
00FF 0  D05A       0631         STC     CZ235+1   SAVE ADDR OF THE I/O AREA      PMN06290
00FF 0  7054       0632         MDX     DZ230     BR TO CONTINUE                 PMN06300
01C0 0C 4CC00CC0   0633   DZC60 BSC  L  *-*       BR TO SERVICE THE INTERRUPT PMN06310
                   C634   *                                                      PMN06320
                   C635   * START ALL DISK CPERATIONS                            PMN06330
                   0636   *                                                      PMN06340
01C2 0  690F       0637   DZC7C STX   1 DZ180+1   SAVE ADDR OF THE I/O AREA      PMN06350
0103 0  C822       0638         XIO     DZ904     START AN OPERATION             PMN06360
                   0639   *                                                      PMN06370
                   064C   * RETURN TO USER                                       PMN06380
                   0641   *                                                      PMN06390
01C4 CC 65C00C00   0642   DZ10C LDX  L1 *-*       RESTORE XR1                    PMN06400
01C6 00 66CC00C0   0643         LDX  L2 *-*       RESTORE XR2                    PMN06410
01C8 0  CCEE       0644         LD      DZ010     INTERRUPT ENTRY            2-6 PMN06420
01C9 00 4C980CF2   0645         BSC  I  DZ000,+-  NO,MONITOR ENTRY           2-6 PMN06430
01CB 0  D0C3       0646         STC     DZ110+1   YES,INT ENTRY              2-6 PMN06440
010C 0  181C       0647         SRA     16        RESET                      2-6 PMN06450
01CD C  DCF9       0648         STC     CZ010     *INT ENTRY                 2-6 PMN06460
01CF 00 4CC00000   0649   DZ110 BSC  L  *-*                                  2-6 PMN06470
0110 0  1C00       065C         NOP               DUMMY OP                   2-6 PMN06480
                   0651   *                                                      PMN06490
                   0652   * SERVICE ALL INTERRUPTS                               PMN06500
                   0653   *                                                      PMN06510
0111 CC 6500CCC0   0654   DZ18C LDX  L1 *-*       ADDR CF I/O AREA TO XR1        PMN06520
0113 00 660000F2   0655         LDX  L2 DZC00     ADDR OF DZ000 TO XR2           PMN06530
0115 0  C816       0656         XIC     CZ910     SENSE THE DSW                  PMN06540
0116 C  DCC6       0657         STC     $DDSW     SAVE THE DSW                   PMN06550
0117 0  4810       0658         BSC     -         SKIP IF ERROR BIT SET      2-6 PMN06560
0118 C  7CE7       0659         MDX     DZ060     BRANCH IF ERROR BIT NOT SET PMN06570
0119 0  C80A       C660   DZ185 LDD     DZ902     RESTORE WORD COUNT             PMN06580
011A 0  D900       0661         STD   1 0         *AND SECTOR ADDRESS            PMN06590
011B 0C 74FFCCEF   C662         MDX  L  $DBSY,-1  SKIP IF 16 RETRIES DONE        PMN06600
011D 0  703A       0663         MDX     DZ235     BRANCH IF LESS THAN 16         PMN06610
                   0664   *                                                      PMN06620
                   0665   * TRAP CUT TO POSTOPERATIVE TRAP                       PMN06630
                   0666   *                                                      PMN06640
011E 0  C80F       0667         LDD     DZ912     1+SCTR ADCR TO EXTENSION       PMN06650
011F 0  C011       0668         LD      DZ915                                    PMN06660
0120 0  4293       0669   DZ190 BSI   2 $PST2-X2  BR TO POSTOPERATIVE ER TRAP PMN06670
0121 0  7034       0670         MDX     DZ232     RETRY OPERATION            2-6 PMN06680
                   0671   *                                                      PMN06690
                   0672   * CCNSTANTS AND WCRK AREAS                             PMN06700
                   0673   *                                                      PMN06710
0122    0000       0674         BSS  E  0                                        PMN06720
                   0675   * EVEN-NUMBERED LABELS ARE ON EVEN BOUNDARIES          PMN06730
0122 0  0001       0676   DZ90C DC      1    CONSTANT,READ-AFTER-SEEK WD CNT     PMN06740
0123 0  0000       0677   DZ901 DC      0    CURRENT ARM POSITION                PMN06750
0124 0  0000       0678   DZ902 DC      *-*  LAST TWO WORDS OF SECTOR            PMN06760
0125 C  0000       0679         DC      *-*  *PREVIOUSLY READ                    PMN06770
0126 C  0000       0680   DZ904 DC      *-*  IOCC FOR OPERATION CURRENTLY        PMN06780
0127 0  0000       0681   DZ905 DC      *-*  *BEING PERFORMED                    PMN06790
0128 0  0000       0682   DZ906 DC      *-*  SAVE AREA FOR IOCC FOR              PMN06800
0129 0  0CC0       0683   DZ907 DC      *-*  *USER-REQUESTED OPERATION           PMN06810
012A 0  0122       C684   DZ908 DC      DZ900 IOCC FOR READ                      PMN06820
012B 0  00C0       0685   DZ909 DC      *-*  *AFTER SEEK                         PMN06830
012C 0  000C       C686   DZ910 DC      *-*  2ND WORD OF SEEK IOCC               PMN06840
012D 0  000C       C687   DZ911 DC      *-*  SENSE IOCC                          PMN06850
012E 0  000C       0688   DZ912 DC      *-*  INTERMEDIATE WORD COUNT             PMN06860
012F 0  0000       C689   DZ913 DC      *-*  ADDR OF NEXT SEQUENTIAL SECTOR      PMN06870
0130 0  5002       C690   DZ914 DC      /5002 WRITE SELECT/POWER UNSAFE INDR     PMN06880
0131 0  5004       C691   DZ915 DC      /5004 READ/WRITE/SEEK ERROR INCICATOR PMN06890
0132 0  FECC       0692   DZ916 DC      -320 TO BE USED TO SIMULTANEOUSLY        PMN06900
0133 0  0001       0693         DC      1    *DECR WD CNT, INCR SCTR ADDR        PMN06910
0134 0  0C8C       0694   DZ920 DC      /0080 REAC CHECK BIT FOR IOCC            PMN06920
0135 0  0600       0695   DZ925 DC      /0600 2ND WD OF READ IOCC W/O AREA CD PMN06930
0136 0  0C08       0696   DZ930 DC      8    NO. SECTORS PER CYLINDER            PMN06940
0137 0  5CCC       C697   DZ935 DC      /5000 NCT READY DISPLAY CODE             PMN06950
0138 0  0FF8       0698   DZ940 DC      /0FF8 'AND' OUT DR CODE, SCTR ADDR       PMN06960
0139 0  000C       0699   DZ945 DC      *-*  FUNC INDICATOR (0#READ,1#WRITE)     PMN06970
013A 0  0701       C7C0   DZ95C DC      /0701 SENSE IOCC W/O AREA CODE           PMN06980
013B 0  00C7       0701   DZ955 DC      /0007 'AND' OUT ALL BUT SCTR NO.         PMN06990
013C 0  0CCA       C7C2   DZ960 DC      $DCYL-$CYLN BASE DEFECTIVE CYL ADDR      PMN07000
013D 0  0C9F       07C3   DZ965 DC      $ACDE BASE AREA CODE ADDR                PMN07010
013F 0  FFFB       07C4   DZ970 DC      $CYLN-$ACDE BASE ARM POSITION ADDR       PMN07020
013F C  000C       0705   DZ975 DC      *-*  2ND WORD OF READ CHECK IOCC         PMN07030
```

```
DISK7

ADCR REL OBJECT   ST.NO.  LABEL OPCD FT CPERANDS                               ID/SEQNO

0140 0  0400       0706   DZ980 DC      /04CC 2ND WD OF SEEK IOCC W/O AREA CD  PMN07040
0141 0  0141       0707   DZ985 DC      321  NO. WORDS PER SECTOR (W/ ADDR)    PMN07050
0142 0  0000       07C8   DZ990 DC      *-*  CURRENT SECTOR NO.                PMN07060
0143 0  FFFF       07C9   DZ995 DC      -1   MASK FOR COMPLEMENTING            PMN07070
                   0710   *                                                    PMN07080
                   C711   * RESERVED FOR SAVING CORE ON A DUMP ENTRY TO SKEL   PMN07090
                   0712   *                                                    PMN07100
0144    CC02       0713         BSS     2    THIS AREA MUST BE AT $CIBA+319    PMN07110
00F2 0             0714   X2    EQU     DZ000                                  PMN07120
                   0715   *                                                    PMN07130
                   0716   *                                                    PMN07140
                   0717   *                                                    PMN07150
0146 0  1810       0718   DZ210 SRA     16                                     PMN07160
0147 0  D0A6       0719         STO     $DBSY      CLEAR BUSY INDICATOR        PMN07170
0148 00 74FF0032   0720         MDX  L  $IOCT,-1   DECREMENT IOCS COUNTER      PMN07180
014A 0  1000       C721         NOP                                            PMN07190
014B 0  7088       0722         MDX     DZ100      TO EXIT                     PMN07200
                   0723   *                                                    PMN07210
                   0724   * PREPARE TO TRAP OUT ON 'POWER UNSAFE' CONDITION    PMN07220
                   0725   *                                                    PMN07230
014C 0  CED7       0726   DZ215 LDD     DZ902      RESTORE WORD COUNT      2-6 PMN07240
014D 0  D900       0727         STD   1 0          *AND SECTOR ADDRES      2-6 PMN07250
014E 0  CCE1       0728         LD      DZ914                                  PMN07260
014F C  70D0       0729         MDX     DZ190      BR TO TPAP OUT              PMN07270
                   0730   *                                                    PMN07280
                   0731   * PREPARE TO TRAP OUT ON 'NOT READY' CONDITION       PMN07290
                   C732   *                                                    PMN07300
0150 0  C0E6       0733   DZ220 LD      DZ935      FETCH ERROR CODE            PMN07310
0151 00 44C00028   0734         BSI  L  $PRET      BR TO PREOPERATIVE ERR TRAP PMN07320
0153 0  7038       0735         MDX     DZ340      RETRY THE OPERATION         PMN07330
                   0736   *                                                    PMN07340
                   0737   *                        STATEMENTS MOVED        2-1 PMN07350
                   0738   *                                                    PMN07360
0154 00 74010C32   0739   DZ23C MDX  L  $IOCT,1    INCREMENT IOCS COUNTER      PMN07370
0156 0  6211       0740   DZ232 LDX   2 'TCNT      TURN BUSY INDICATOR ON 2-10 PMN07380
0157 0  6A96       0741         STX   2 $DBSY      *                       2-6 PMN07390
0158 00 65000000   0742   DZ235 LDX  L1 *-*        ADDR I/O AREA TO XR1        PMN07400
015A 0  C9CC       0743         LDD   1 0                                      PMN07410
015B 0  D8C8       0744         STD     DZ902      SAVE WORD COUNT, SCTR ADDR  PMN07420
015C 0  D8D1       0745         STD     DZ912                                  PMN07430
015D 0  1810       0746   DZ240 SRA     16                                     PMN07440
015E 0  1084       0747         SLT     4          DRIVE CODE IN BITS 12-15    PMN07450
015F 0  DCCE       0748         STO     DZ280+1                                PMN07460
0160 0  80DC       0749         A       DZ965      COMPUTE AND STORE THE       PMN07470
0161 0  D01C       0750         STO     DZ330+1    *ADDR OF THE AREA CODE      PMN07480
0162 0  80DB       0751         A       DZ970      COMPUTE AND STORE THE       PMN07490
0163 0  D034       0752         STO     DZ350+1    *ADDR OF THE ARM POSITION   PMN07500
0164 0  80D7       0753         A       DZ960      ADD IN BASE DT ADDR         PMN07510
0165 C  80C8       0754         A       DZ280+1    ADD IN THE DRIVE            PMN07520
0166 0  8007       0755         A       DZ280+1    *CODE TWICE MORE            PMN07530
0167 C  D006       0756         STC     DZ280+1                                PMN07540
0168 0  62FD       0757         LDX   2 -3         INITIALIZE COUNTER FOR LOOP PMN07550
0169 0  698F       0758         STX   1 DZ906                                  PMN07560
016A 0  C101       C759         LD    1 1          FETCH DESIRED SECTOR ADDR   PMN07570
016B 0  E0CC       0760         AND     DZ940      'AND' OUT SECTOR NO.        PMN07580
016C 0  D101       0761   DZ250 STO   1 1          *AND DRIVE CODE             PMN07590
016D 00 94C0C0C0   0762   DZ280 S    L  *-*        SUB DEFECTIVE CYLINDER ADDR PMN07600
016F 0  4828       0763         BSC     Z+         SKIP IF BAD CYLINDER        PMN07610
0170 0  7007       0764         MDX     DZ300      BR TO CONTINUE PROCESSING   PMN07620
0171 0  C1C1       0765         LD    1 1                                      PMN07630
0172 0  80C3       0766         A       DZ930      INCREMENT SCTR ADDR BY 8    PMN07640
0173 00 7401016E   0767         MDX  L  DZ280+1,1 POINT TO NEXT DEFECTIVE CYL  PMN07650
0175 0  7201       0768         MDX   2 1          SKIP AFTER 3RD PASS         PMN07660
0176 C  7CF5       0769         MDX     DZ250      COMPARE W/ NEXT DEF CYL ADR PMN07670
0177 0  D101       0770         STO   1 1          SCTR ADDR WITH 3 DEF CYL2-4 PMN07680
                   0771   *                                                    PMN07690
                   0772   * CONSTRUCT THE 2ND WORD OF ALL IOCC'S               PMN07700
                   0773   *                                                    PMN07710
0178 00 660000F2   0774   DZ300 LDX  L2 DZ000      ADDR OF DZ000 TO XR2        PMN07720
017A 0  C23D       0775         LD    2 DZ913-X2   FETCH SECTOR ADDRESS        PMN07730
017B 0  E249       0776         AND   2 DZ955-X2   'AND' OUT ALL BUT SECTOR NO PMN07740
017C 0  D250       0777         STO   2 DZ990-X2   SAVE SECTOR NO.             PMN07750
017D 00 C40C0000   0778   DZ330 LD   L  *-*        FETCH AREA CODE             PMN07760
017F 0  EA4E       0779         OR    2 DZ980-X2   'OR' IN SEEK FUNCTION CODE  PMN07770
0180 0  D23A       0780         STO   2 DZ910-X2   SEEK IOCC MINUS DIRECTION   PMN07780
0181 0  EA43       C781         OR    2 DZ925-X2   'OR' IN READ FUNCTION CODE  PMN07790
0182 0  D239       0782         STO   2 DZ909-X2   IOCC FOR READ-AFTER-SEEK    PMN07800
0183 0  EA5C       0783         OR    2 DZ990-X2   'OR' IN SECTOR NO.          PMN07810
0184 0  9247       0784         S     2 DZ945-X2   COMPLETE READ/WRITE CODE    PMN07820
0185 0  D237       0785         STO   2 DZ907-X2   2ND WD OF READ/WRITE IOCC   PMN07830
0186 0  FA42       C786         OR    2 DZ920-X2   'OR' IN READ CHECK BIT      PMN07840
0187 0  8247       0787         A     2 DZ945-X2                               PMN07850
0188 0  D24D       0788         STO   2 DZ975-X2   2ND WD OF READ CHECK IOCC   PMN07860
0189 0  EA48       0789         OR    2 DZ950-X2   'OR' IN SENSE IOCC BITS     PMN07870
018A 0  D23B       0790         STO   2 DZ911-X2   COMPLETED SENSE IOCC        PMN07880
018B 0  CA3C       0791         LDD   2 DZ912-X2   1+SCTR ADDR TO EXTENSION    PMN07890
018C 0  0A3A       0792   DZ340 XIO   2 DZ910-X2   SENSE FOR DISK READY        PMN07900
018D 0  D2EB       0793         STO   2 $DDSW-X2   SAVE THE DSW                PMN07910
018E 0  4828       C794         BSC     Z+         SKIP UNLESS POWER UNSAFE OR PMN07920
018F 0  70BC       0795         MDX     DZ215      *WRITE SELECT, BR OTHERWISE PMN07930
0190 0  1002       C796         SLA     2          BR TO PREOPERATIVE ERR TRAP PMN07940
0191 0  4828       C797         BSC     Z+         *IF DISK NOT READY, SKIP    PMN07950
```

```
DISKZ

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

0192 0   70BC       0798         MDX      DZ220       *OTHERWISE                  PMN07960
                    0799    *                         STATEMENTS REMOVED      2-1 PMN07970
0193 0   1002       0800         SLA      2           CHECK FOR ARM AT HOME  2-11 PMN07980
0194 0   4828       0801         BSC      +Z          SKIP IF NOT HOME       2-11 PMN07990
0195 0   7C1C       0802         MDX      DZ390+1     BR TO VERIFY ARM AT HOM2-11 PMN08000
                    0803    *                                                2-11 PMN08010
                    0804    *                         3 INSTRUCTIONS REMOVED 2-11 PMN08020
                    0805    *                                                2-11 PMN08025
0196 0   C101       0806         LD     1 1           FETCH DESIRED CYLINDER ADDR PMN08030
0197 00  94000000   0807    DZ350 S     L  *-*         SUBTRACT ARM POSITION       PMN08040
0199 0   4818       0808         BSC      +-          SKIP IF SEEK NECESSARY      PMN08050
019A 0   701B       0809         MDX      DZ400       BRANCH TO PERFORM OPERATION PMN08060
                    0810    *                                                     PMN08070
                    0811    * SEEK                                                PMN08080
                    0812    *                                                     PMN08090
019B 0   1893       0813         SRT      19          PUT NO. CYLINDERS IN EXT    PMN08100
019C 0   180F       0814         SRA      15          + OR - SIGN TO BIT 15       PMN08110
019D 0   1002       0815         SLA      2           SHIFT SIGN TO BIT 13        PMN08120
019E 0   EA3A       0816         OR     2 DZ910-X2    "OR" IN REMAINDER OF IOCC   PMN08130
019F 0   18D0       0817         RTE      16                                      PMN08140
01A0 0   4810       0818         BSC      -           SKIP IF SEEK TOWARD HOME    PMN08150
01A1 0   7002       0819         MDX      DZ380       BRANCH IF SEEK TOWARD CENTR PMN08160
01A2 0   F251       0820         EOR    2 DZ995-X2    COMPLEMENT NO. CYLS TO BE   PMN08170
01A3 0   823C       0821         A      2 DZ900-X2    *SOUGHT TO GET POSITIVE NO. PMN08180
01A4 0   DA34       0822    DZ380 STD   2 DZ904-X2                                PMN08190
01A5 0   42CF       0823    DZ390 BSI   2 DZ070-1-X2 START SEEK              2-1 PMN08200
                    0824    *                                                     PMN08210
                    0825    * SEEK COMPLETE INTERRUPT PROCESSING                  PMN08220
                    0826    *                                                     PMN08230
01A6 0   CA38       0827         LDD    2 DZ908-X2    SET UP IOCC FOR             PMN08240
01A7 0   DA34       0828         STD    2 DZ904-X2    *READ AFTER SEEK            PMN08250
01A8 0   420F       0829         BSI    2 DZ070-1-X2 START READ-AFTER-SEEK        PMN08260
                    0830    *                                                     PMN08270
                    0831    * READ-AFTER-SEEK COMPLETE INTERRUPT PROCESSING       PMN08280
                    0832    *                                                     PMN08290
01A9 0   C231       0833         LD     2 DZ9C1-X2    FETCH ADR OF SCTR JUST READ PMN08300
01AA 00  D48C0198   0834         STO    I  DZ350+1     UPDATE ARM POSITION         PMN08310
01AC 0   91C1       0835         S      1 1           SUB DESIRED SCTR ADDR       PMN08320
01AD 00  4C1801B6   0836         BSC    L  DZ400,+-    BR IF SEEK SUCCESSFUL  2-11 PMN08330
01AF 00  74FFC0EF   0837         MDX    L  $DBSY,-1    SKIP IF NO MORE RETRIES2-11 PMN08332
01B1 0   70F4       0838         MDX       DZ350-1     BR TO CALC NEW SEEK    2-11 PMN08334
01B2 00  740100FF   0839         MDX    L  $DBSY,1     PREVENT A MINUS $DBSY  2-11 PMN08336
01B4 00  4C000119   0840         BSC    L  DZ185       BR TO TRAP OUT         2-11 PMN08338
                    0841    *                                                     PMN08340
                    0842    *                                                     PMN08350
                    0843    * READ/WRITE                                          PMN08360
                    0844    *                                                     PMN08370
01B6 0   CA3C       0845    DZ400 LDD   2 DZ912-X2    FETCH INTERMEDIATE WD CNT   PMN08380
01B7 0   48C8       0846         BSC      +           SKIP, WD CNT NOT EXHAUSTED  PMN08390
01B8 0   7011       0847         MDX      DZ410       BR IF WD CNT EXHAUSTED 2-11 PMN08400
01B9 0   8A4C       0848         AD     2 DZ916-X2    DECREMENT WORD COUNT AND    PMN08410
01BA 0   DA3C       0849         STD    2 DZ912-X2    *INCREMENT SECTOR ADDRESS   PMN08420
01BB 0   4830       0850         BSC      Z-          SKIP IF THIS IS LAST SECTOR PMN08430
01BC 0   181C       0851         SRA      16          CLEAR ACCUMULATOR           PMN08440
01BD 0   824F       0852         A      2 DZ985-X2    ADD BACK 321 TO WD CNT      PMN08450
01BE 0   D1CC       0853         STO    1 0           STORE RESULT IN I/O AREA    PMN08460
01BF 0   CA36       0854         LDD    2 DZ906-X2    RESTORE IOCC FOR ORIGINALLY PMN08470
01C0 0   DA34       0855         STD    2 DZ904-X2    *REQUESTED OPERATION        PMN08480
01C1 0   C101       0856         LD     1 1           ADD SECTOR NO. TO SECTOR    PMN08490
01C2 0   FA5C       0857         OR     2 DZ990-X2    *ADDRESS                    PMN08500
01C3 0   D101       0858         STO    1 1                                       PMN08510
01C4 0   42CF       0859         BSI    2 DZ070-1-X2 START READ/WRITE OPERATION   PMN08520
                    0860    *                                                     PMN08530
                    0861    * READ/WRITE COMPLETE INTERRUPT PROCESSING            PMN08540
                    0862    *                                                     PMN08550
01C5 0   C24D       0863         LD     2 DZ975-X2    SET UP FOR READ CHECK       PMN08560
01C6 0   D235       0864         STO    2 DZ905-X2                                PMN08570
01C7 0   C247       0865         LD     2 DZ945-X2    FETCH FUNCTION INDICATOR    PMN08580
01C8 0   4820       0866         BSC      Z           SKIP IF READ REQUESTED      PMN08590
01C9 0   420F       0867         BSI    2 DZ070-1-X2 START READ CHECK OPERATION   PMN08600
01CA 0   CA32       0868    DZ410 LDD   2 DZ902-X2    RESTORE LST 2 WDS, SEC-2-11 PMN08610
01CB 0   D900       0869         STD    1 0           *TOR PREVIOUSLY READ        PMN08620
01CC 0   C23C       0870         LD     2 DZ912-X2    FETCH INTERMEDIATE WD CNT   PMN08630
                    0871    * SHORT BSC AND MDX CHANGED TO FOLLOWING BSC L 2-11   PMN08640
01CD 00  4C080146   0872         BSC    L  DZ210,+     BR IF WD CNT EXHAUSTED 2-11 PMN08650
01CF 00  750C0140   0873         MDX    L1 320         POINT XR1 TO NEW I/O AREA   PMN08660
01D1 0   C900       0874         LDD    1 0           SAVE LAST 2 WDS OF SECTOR   PMN08670
01D2 0   DA32       0875         STD    2 DZ902-X2    *JUST READ/WRITTEN          PMN08680
01D3 0   CA3C       0876         LDD    2 DZ912-X2    WD CNT, SCTR ADDR NEXT OP   PMN08690
01D4 0   D900       0877         STD    1 0           STORE BOTH IN NEW I/O AREA  PMN08700
01D5 0   7087       0878         MDX      DZ240       BACK TO SET UP NEXT OPERATN PMN08710
                    0879    *                                                     PMN08720
                    0880    *                                                     PMN08730
01E0 0              0881    $ZEND EQU     /01E0       1 + END OF DISKZ       2-11 PMN08735
01D6     C004       0882         BSS      $ZEND-*-6   PATCH AREA            2-11 PMN08740
                    0883    *                                                     PMN08750
                    0884    *                                                     PMN08760
```

```
DISKZ

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

01DA 0  00A0      0885            DC    'CIL1      ID NO. OF CORE IMAGE LDR,P1 PMN08770
01DB 0  0000      0886     $CIDN  DC    *-*        CORE ADDR/CID NO.           PMN08780
01DC 0  0000      0887            DC    *-*        WORD COUNT                  PMN08790
01DD 0  0000      0888            DC    *-*        SCTR ADDR                   PMN08800
01DE    0002      0889            BSS   2          WD CNT, SCTR ADDR CORE LDS  PMN08810
                  0890     *                       $ZEND EQUATE MOVED     2-11 PMN08820   ——$ZEND


EQUIVALENCES

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

                  0892     *                                                   PMN08840
                  0893     * EQUIVALENCES FOR DCOM PARAMETERS                  PMN08850
                  0894     *                                                   PMN08860
0004 0            0895     #NAME EQU    4     NAME OF PROGRAM/CORE LOAD        PMN08870
0006 0            0896     #DBCT EQU    6     BLOCK CT OF PROGRAM/CORE LOAD    PMN08880
0007 0            0897     #FCNT EQU    7     FILES SWITCH                     PMN08890
0008 0            0898     #SYSC EQU    8     SYSTEM/NON-SYSTEM CARTRIDGE INDR PMN08900
0009 0            0899     #JBSW EQU    9     JOBT SWITCH                      PMN08910
000A 0            0900     #CBSW EQU    10    CLB-RETURN SWITCH                PMN08920
000B 0            0901     #LCNT EQU    11    NO. OF LOCALS                    PMN08930
000C 0            0902     #MPSW EQU    12    CORE MAP SWITCH                  PMN08940
000D 0            0903     #MDF1 EQU    13    NO. DUP CTRL RECORDS (MODIF)     PMN08950
000E 0            0904     #MDF2 EQU    14    ADDR OF MODIF BUFFER             PMN08960
000F 0            0905     #NCNT EQU    15    NO. OF NOCALS                    PMN08970
0010 0            0906     #ENTY EQU    16    RLTV ENTRY ADDR OF PROGRAM       PMN08980
0011 0            0907     #RP67 EQU    17    1442-5 SWITCH                    PMN08990
0012 0            0908     #TODR EQU    18    OBJECT WORK STORAGE DRIVE CODE   PMN09000
0014 0            0909     #FHOL EQU    20    ADDR LARGEST HOLE IN FIXED AREA  PMN09010
0015 0            0910     #FSZE EQU    21    BLK CNT LARGEST HOLE IN FXA      PMN09020
0016 0            0911     #UHOL EQU    22    ADDR LAST HOLE IN USER AREA 2-10 PMN09030
0017 0            0912     #USZE EQU    23    BLK CNT LAST HOLE IN UA     2-10 PMN09040
0018 0            0913     #DCSW EQU    24    DUP CALL SWITCH                  PMN09050
0019 0            0914     #PIOD EQU    25    PRINCIPAL I/O DEVICE INDICATOR   PMN09060
001A 0            0915     #PPTR EQU    26    PRINCIPAL PRINT DEVICE INDICATOR PMN09070
001B 0            0916     #CIAD EQU    27    RLTV ADDR IN 'STRT OF CIL ADDR   PMN09080
001C 0            0917     #ACIN EQU    28    AVAILABLE CARTRIDGE INDICATOR    PMN09090
001D 0            0918     #GRPH EQU    29    2250 INDICATOR               2G2 PMN09100
001E 0            0919     #GCNT EQU    30    NO. G2250 RECORDS            2G2 PMN09110
001F 0            0920     #LOSW EQU    31    LOCAL-CALLS-LOCAL SWITCH     2-2 PMN09120
0020 0            0921     #X3SW EQU    32    SPECIAL ILS SWITCH           2-2 PMN09130
0021 0            0922     #ECNT EQU    33    NO. OF *EQUAT RCDS           2-4 PMN09140
0023 0            0923     #ANDU EQU    35    1+BLK ADDR END OF UA (ADJUSTED)  PMN09150
0028 0            0924     #BNDL EQU    40    1+BLK ADDR END OF UA (BASE)      PMN09160
002D 0            0925     #FPAD EQU    45    FILE PROTECT ADDR                PMN09170
0032 0            0926     #PCID EQU    50    CARTRIDGE ID, PHYSICAL DRIVE     PMN09180
0037 0            0927     #CIDN EQU    55    CARTRIDGE ID, LOGICAL DRIVE      PMN09190
003C 0            0928     #CIBA EQU    60    SCTR ADDR OF CIB                 PMN09200
0041 0            0929     #SCRA EQU    65    SCTR ADDR OF SCRA                PMN09210
0046 0            0930     #FMAT EQU    70    FORMAT OF PROG IN WORKING STG    PMN09220
004B 0            0931     #FLET EQU    75    SCTR ADDR 1ST SCTR OF FLET       PMN09230
0050 0            0932     #ULET EQU    80    SCTR ADDR 1ST SCTR OF LET        PMN09240
0055 0            0933     #WSCT EQU    85    BLK CNT OF PROG IN WORKING STG   PMN09250
005A 0            0934     #CSHN EQU    90    NO. SCTRS IN CUSHION AREA        PMN09260
                  0935     *                                                   PMN09270
                  0936     * EQUIVALENCES FOR PHASE ID NUMBERS                 PMN09280
                  0937     *                                                   PMN09290
006E 0            0938     'MCRA EQU    110        PHASE ID FOR MCRA           PMN09300
0073 0            0939     'SUP6 EQU    115        PHASE ID FOR DUMP PROG 2-10 PMN09310
0074 0            0940     'SUP7 EQU    116        PHASE ID FOR AUX SUPV  2-10 PMN09320
0078 0            0941     'CLB0 EQU    120        PHASE ID FOR CLB, PHASE 0/1 PMN09330
008C 0            0942     '1403 EQU    140        PHASE ID FOR SYS 1403 SUBR  PMN09340
008D 0            0943     '1132 EQU    141        PHASE ID FOR SYS 1132 SUBR  PMN09350
008E 0            0944     'CPTR EQU    142        PHASE ID FOR SYS CP SUBR    PMN09360
008F 0            0945     '2501 EQU    143        PHASE ID FOR SYS 2501 SUBR  PMN09370
0090 0            0946     '1442 EQU    144        PHASE ID FOR SYS 1442 SUBR  PMN09380
0091 0            0947     '1134 EQU    145        PHASE ID FOR SYS 1134 SUBR  PMN09390
0092 0            0948     'KBCP EQU    146        PHASE ID FOR SYS KB/CP SUBR PMN09400
0093 0            0949     'CDCV EQU    147        PHASE ID FOR SYS CD   CONV  PMN09410
0094 0            0950     'PTCV EQU    148        PHASE ID FOR SYS 1134 CONV  PMN09420
0095 0            0951     'KBCV EQU    149        PHASE ID FOR SYS KB   CONV  PMN09430
0096 0            0952     'DZID EQU    150        PHASE ID FOR DISKZ          PMN09440
0097 0            0953     'D1ID EQU    151        PHASE ID FOR DISK1          PMN09450
0098 0            0954     'DNID EQU    152        PHASE ID FOR DISKN          PMN09460
00A0 0            0955     'CIL1 EQU    160        PHASE ID FOR CI LOADER,PH 1 PMN09470
00A1 0            0956     'CIL2 EQU    161        PHASE ID FOR CI LOADER,PH 2 PMN09480
                  0957     *                                                   PMN09490
                  0958     * EQUIVALENCES FOR RESIDENT MONITOR                 PMN09500
                  0959     *                                                   PMN09510
0014 0            0960     $LKNM EQU    $HASH      SAVE AREA FOR NAME OF LINK  PMN09520
0016 0            0961     $RMSW EQU    $HASH+2    EXIT-LINK-DUMP SW(-1,0,+1)  PMN09530
0017 0            0962     $CXR1 EQU    $HASH+3    SAVE AREA FOR XR1           PMN09540
0018 0            0963     $CLSW EQU    $HASH+4    SW FOR CORE IMAGE LDR,PH 2  PMN09550
0019 0            0964     $DMPF EQU    $HASH+5    DUMP FORMAT CODE            PMN09560
001A 0            0965     $ACEX EQU    $HASH+6    ACC AND EXT WHEN ENTER DUMP PMN09570
005A 0            0966     $CILA EQU    $S150+1    ADDR OF END OF DK I/O - 3   PMN09580
00B9 0            0967     $IBT2 EQU    $I205+1    ADR OF SERVICE PART OF DKIO PMN09590
00D4 0            0968     $IBT4 EQU    $I405+1    ADDR OF THE IBT             PMN09600
00FF 0            0969     $SNLT EQU    $DBSY+1    SENSE LIGHT INDICATOR       PMN09610
00F0 0            0970     $PAUS EQU    DZ000-2    PAUSE,INTERRUPT INDICATOR   PMN09620
00F1 0            0971     $RWCZ EQU    DZ000-1    READ/WRITE SWITCH (CARDZ)   PMN09630
00F4 0            0972     $XR3X EQU    $I496      XR3 SETTING DURING XEQ  2-2 PMN09640
```

EQUIVALENCES

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

                   0973     *                                                   PMN09650
                   0974     * EQUIVALENCES FOR ABSOLUTE SECTOR ADDRESSES        PMN09660
                   0975     *                                                   PMN09670
0000 0             0976     *ICAD EQU     0    ADDR OF SCTR WITH ID,DEF CYL ADR PMN09680
0001 0             0977     *DCOM EQU     1    ADDR OF SCTR CONTAINING DCOM     PMN09690
0002 0             0978     *RIAD EQU     2    ADDROOF SCTR CONTAINING RES IMGE PMN09700
0003 0             0979     *SLET EQU     3    ADDR OF SCTR CONTAINING SLET     PMN09710
0006 0             0980     *RTBL EQU     6    ADDR OF SCTR CONTAINING RELC TBL PMN09720
0007 0             0981     *HDNG EQU     7    ADDR OF SCTR CONTAINING PAGE HDR PMN09730
0000 0             0982     *STRT EQU     0    ADDR OF SCTR W/ COLD START PROG  PMN09740
                   0983     *                                                   PMN09750
                   0984     * EQUIVALENCES FOR THE CORE IMAGE HEADER            PMN09760
                   0985     *                                                   PMN09770
0000 0             0986     *XEQA EQU     0    RLTV ADDR OF CORE LOAD EXEC ADDR PMN09780
0001 0             0987     *CMON EQU     1    RLTV ADDR OF WD CNT OF COMMON    PMN09790
0002 0             0988     *DREQ EQU     2    RLTV ADDR OF DISK I/O INDICATOR  PMN09800
0003 0             0989     *FILE EQU     3    RLTV ADDR OF NO. FILES DEFINED   PMN09810
0004 0             0990     *HWCT EQU     4    RLTV ADDR OF WD CNT OF CI HEADER PMN09820
0005 0             0991     *LSCT EQU     5    SCTR CNT OF FILES IN WK STORAGE  PMN09830
0006 0             0992     *LDAD EQU     6    RLTV ADDR OF LOAD ADDR CORE LOAD PMN09840
0007 0             0993     *XCTL EQU     7    RLTV ADDR DISK1/DISKN EXIT CTRL  PMN09850
0008 0             0994     *TVWC EQU     8    RLTV ADDR OF WD CNT OF TV        PMN09860
0009 0             0995     *WCNT EQU     9    RLTV ADDR OF WD CNT OF CORE LOAD PMN09870
000A 0             0996     *XR3X EQU     10   RLTV ADDR OF EXEC SETTING OF XR3 PMN09880
000B 0             0997     *ITVX EQU     11   RLTV ADDR OF 1ST WD OF ITV       PMN09890
0011 0             0998     *ILS4 EQU     17   RLTV ADDR OF 1ST WD OF IBT4      PMN09900
001A 0             0999     *OVSW EQU     26   RLTV ADDR OF LOCAL/SOCAL SWITCH  PMN09910
001B 0             1000     *CORE EQU     27   CORE SIZE OF BUILDING SYST  2-10 PMN09920
001D 0             1001     *HEND EQU     29   RLTV ADDR OF LAST WD OF CI HDR   PMN09930
                   1002     *                                                   PMN09940
                   1003     * EQUIVALENCES FOR LET/FLET                         PMN09950
                   1004     *                                                   PMN09960
0005 0             1005     *LFHD EQU     5    WORD COUNT OF LET/FLET HEADER    PMN09970
0003 0             1006     *LFEN EQU     3    NO OF WDS PER LET/FLET ENTRY     PMN09980
0000 0             1007     *SCTN EQU     0    RLTV ADDR OF LET/FLET SCTR NO.   PMN09990
0001 0             1008     *UAFX EQU     1    RLTV ADDR OF SCTR ADDR OF UA/FXA PMN10000
0003 0             1009     *WCSA EQU     3    RLTV ADDR OF WDS AVAIL IN SCTR   PMN10010
0004 0             1010     *NEXT EQU     4    RLTV ADDR OF ADDR NEXT SCTR      PMN10020
0000 0             1011     *LFNM EQU     0    RLTV ADDR OF LET/FLET ENTRY NAME PMN10030
0002 0             1012     *BLCT EQU     2    RLTV ADDR OF LET/FLET ENTRY DBCT PMN10040
                   1013     *                                                   PMN10050
                   1014     * MISCELLANEOUS EQUIVALENCES                        PMN10060
                   1015     *                                                   PMN10070
0033 0             1016     *ISTV EQU     51   ISS NO. ADJUSTMENT FACTOR    2-1 PMN10080
0005 0             1017     *MXDR EQU     5    MAX NO. DRIVES SUPPORTED         PMN10090
0380 0             1018     *CCMZ EQU     896  LOW COMMON LIMIT FOR DISKZ       PMN10100
04C0 0             1019     *CCM1 EQU     1216 LOW COMMON LIMIT FOR DISK1       PMN10110
0600 0             1020     *CCM2 EQU     1536 LOW COMMON LIMIT OF DISKN        PMN10120
0011 0             1021     *TCNT EQU     17   NO. TRIES BEFORE DISK ERROR      PMN10130
00F9 0             1022     *DKEP EQU     DZ000+7 LIBF ENTRY TO DISK1/N         PMN10140
00F7 0             1023     *DKIP EQU     DZ000+5 DISK I/O INTERRUPT ENTRY PT   PMN10150
0010 0             1024     *SCIB EQU     16   CIB SECTOR COUNT             2-2 PMN10160
0003 0             1025     *HCIB EQU     3    HIGH COMMON SECTOR COUNT     2-2 PMN10170
1000 0             1026     *MCOR EQU     4096       SIZE OF MINIMUM CORE   2-2 PMN10180
007F 0             1027     Y     EQU     127                                   PMN10190
                   1028     *                                                   PMN10200
0004 0             1029     *CIDN EQU     4    RLTV ADDR CARTRIDGE ID       2-2 PMN10210
0005 0             1030     *COPY EQU     5    RLTV ADDR COPY INDICATOR     2-2 PMN10220
0001 0             1031     *DCTB EQU     1    RLTV ADDR DEFECTIV CYL TBL   2-2 PMN10230
0008 0             1032     *DTYP EQU     8    RLTV ADDR DISK TYPE INDR     2-2 PMN10240
```

COLD START PROGRAM

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

                   1034     ****************************************************** PMN10260
                   1035     *                                                 * PMN10270
                   1036     *STATUS - VERSION 2, MODIFICATION 11              * PMN10280
                   1037     *                                                 * PMN10290
                   1038     *FUNCTION/OPERATION -                             * PMN10300
                   1039     *    THIS PROGRAM IS READ INTO CORE FROM SECTOR 0 * PMN10310
                   1040     *    OF THE SYSTEM CARTRIDGE AND TRANSFERRED TO BY * PMN10320
                   1041     *    THE COLD START CARD.  DEFECTIVE CYLINDER     * PMN10330
                   1042     *    ADDRESSES, CARTRIDGE ID AND DISKZ ARE ALSO ON * PMN10340
                   1043     *    SECTOR 0 AND ARE READ IN AT THE SAME TIME.   * PMN10350
                   1044     *    ALL THAT REMAINS FOR THE COLD START PROGRAM IS* PMN10360
                   1045     *    TO READ IN THE RESIDENT IMAGE, SAVE THE      * PMN10370
                   1046     *    CARTRIDGE ID AND TRANSFER TO THE AUXILIARY   * PMN10380
                   1047     *    SUPERVISOR THROUGH $DUMP IN THE RESIDENT     * PMN10390
                   1048     *    MONITOR.                                     * PMN10400
                   1049     *                                                 * PMN10410
                   1050     *ENTRY - CR010-2                                  * PMN10420
                   1051     *    ENTER PROGRAM BY TRANSFER FROM COLD START CARD* PMN10430
                   1052     *                                                 * PMN10440
                   1053     *INPUT -                                          * PMN10450
                   1054     *    THE CARTRIDGE ID OF LOGICAL DRIVE ZERO (THE  * PMN10460
                   1055     *    SYSTEM CARTRIDGE) IS READ IN FROM SECTOR 0   * PMN10470
                   1056     *    WITH THE COLD START PROGRAM.                 * PMN10480
                   1057     *                                                 * PMN10490
```

```
COLD START PROGRAM

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

                   1058    *OUTPUT -                                        * PMN105C0
                   1059    *   * THE RESIDENT IMAGE IS READ INTO CCRE FROM  * PMN10510
                   1060    *     THE DISK.                                  * PMN10520
                   1061    *   * IN COMMA-                                  * PMN10530
                   1062    *      $ACDE                                     * PMN10540
                   1063    *      $CIBA-1                                   * PMN10550
                   1C64    *      $CIDN                                     * PMN10560
                   1065    *      $CYLN                                     * PMN10570
                   1066    *      $DBSY                                     * PMN10580
                   1067    *      $IOCT                                     * PMN1059C
                   1068    *                                                * PMN10600
                   1069    *EXTERNAL REFERENCES -                           * PMN1061C
                   1070    *   DZCOC  SUBROLTINE TC PERFORM CISK I/C.       * PMN10620
                   1071    *                                                * PMN10630
                   1072    *EXITS -                                         * PMN10640
                   1073    *   THE ONLY EXIT IS TO THE AUXILIARY SUPERVISOR * PMN10650
                   1074    *   AS FOLLOWS-                                  * PMN10660
                   1075    *     BSI    $DUMP                               * PMN10670
                   1076    *     DC     -1                                  * PMN10680
                   1077    *                                                * PMN10690
                   1078    *TABLES/WORK AREAS - N/A                         * PMN10700
                   1C79    *                                                * PMN10710
                   1080    *ATTRIBUTES -                                    * PMN1072C
                   1081    *   THIS PROGRAM IS NOT NATURALLY RELOCATABLE.   * PMN10730
                   1082    *                                                * PMN10740
                   1083    *NOTES -                                         * PMN10750
                   1084    *   DISK ERRORS RESULT IN A WAIT AT $PST2.       * PMN10760
                   1C85    ************************************************** PMN10770
                   1087    *                                                  PMN10790
                   1088    * READ THE RESIDENT IMAGE INTO CORE                PMN1080C
                   1C89    *                                                  PMN10810
01E0 0  617F       1C90          LDX   1 Y                                    PMN10820
01E1 0  C824       1091          LDD     CR920     SET UP WORD COUNT AND SCTR PMN10830
01E2 00 DC000004   1C92    CR010 STD   L $CIBA-1   *ADDR OF RESIDENT IMAGE    PMN10840
01E4 0  D125       1093          STO   1 $DCYL-Y   *INITIALIZE DEF CYL NO. 1  PMN10850
01E5 0  C184       1C94          LD    1 3-Y       FETCH LOG DRIVE 0 AREA CODE PMN10860
01E6 0  D120       1095          STO   1 $ACDE-Y   *AND STORE IT IN COMMA     PMN10870
01E7 0  D01F       1C96          STO     CR920+1   SAVE THE AREA CODE         PMN10880
01E8 0  C156       1097          LD    1 DZ000-2-27-Y FETCH AND SAVE THE      PMN10890
01E9 0  D0F1       1098          STO     $CIDN     *CARTRIDGE ID              PMN10900
01EA 00 660001FE   1C99          LDX  L2 CR020     SET UP TEMPORARY      2-11 PMN10902
01EC 00 6E00000A   1100          STX  L2 $LEV2     *ILS02                2-11 PMN10904
01EE 0  C0F4       11C1          LD      CR010+1   FETCH CORE ADDR OF RESIDENT PMN10910
01EF 0  1890       1102          SRT     16        *IMAGE AND PUT IN EXTENSION PMN10920
01F0 0  D16F       11C3          STO   1 $DBSY-Y   CLEAR DISK BUSY INDICATOR  PMN10930
01F1 0  D118       1104          STO   1 $CYLN-Y   INITIALIZE ARM POSITION    PMN10940
01F2 0  4173       1105          BSI   1 DZ000-Y   FETCH RESIDENT IMAGE       PMN10950
01F3 C  300C       1106          WAIT              WAIT OUT THE INTERRUPT     PMN10960
                   1107    *                                                  PMN10970
                   11C8    * INITIALIZE ITEMS IN COMMA                        PMN1098C
                   1109    *                                                  PMN10990
01F4 0  1810       1110          SRA     16                                   PMN110CC
01F5 0  D1B3       1111          STO   1 $IOCT-Y   CLEAR IOCS COUNTER         PMN11010
01F6 0  C80D       1112          LDD     CR910                                PMN11020
01F7 0  D985       1113          STD   1 $CIBA-1-Y *FOR SAVING CORE ON THE CIB PMN11030
01F8 0  C00E       1114          LD      CR920+1   FETCH AREA CODE            PMN11040
01F9 0  D120       1115          STO   1 $ACDE-Y   RESET AREA CODE            PMN11050
01FA 0  C00D       1116          LD      CR905     INITIALIZE WD ZERO TO      PMN11060
01FB 0  D181       1117          STO   1 0-Y       *AN 'MDX   *-1' LOOP       PMN11070
                   1118    *                                                  PMN11080
                   1119    * TRANSFER TO THE AUXILIARY SUPERVISOR             PMN1109C
                   1120    * TO COMPLETE INITIALIZATION                       PMN11100
                   1121    *                                                  PMN1111C
01FC 0  41C0       1122          BSI   1 $DUMP-Y   BR TO AUXILLIARY SUPERVISOR PMN11120
01FD 0  FFFF       1123          DC      -1        *FOR JOB PROCESSING        PMN11130
                   1124    *                                                  PMN11140
01FE 0  0000       1125    CRC20 DC      *-*                             2-11 PMN11142
01FF 0  4178       1126          BSI   1 DZ010-Y   BR TO SERVICE INTERRUPT2-11 PMN11144
02C0 00 74FFC1FE   1127          MDX   L CR020,-1                        2-11 PMN11146
02C2 00 4CCCC1FE   1128          BOSC  I CR020     RETURN                2-11 PMN11148
                   1129    *                                                  PMN11160
                   1130    * CCNSTANTS AND WORK AREAS                         PMN11170
                   1131    *                                                  PMN11180
02C4    0C00       1132          BSS   E 0         ASSURE EVEN BOUNDARY  2-11 PMN11190
0204 0  0CC0       1133    CR910 DC      0         WD CNT.SCTR ADDR OF    2-5 PMN11200
0205 0  0007       1134          DC      'HDNG     *HARMLESS WRITE TO DISK    PMN11210
02C6 0  00E8       1135    CR920 DC      $DBSY-$CH12 WD CNT AND SCTR          PMN11220
0207 0  0002       1136          DC      'RIAD     *ADDR OF RESIDENT IMAGE    PMN11230
02C8 C  70FF       1137    CR905 MDX     *-1       TO BE PUT AT ADDR 0000 2-11 PMN11231
0209    0C09       1138          BSS     /0212-*   PATCH AREA            2-11 PMN11232
020A    020A       1139          END     *
```

CROSS-REFERENCE

SYMBOL	VALUE	REL	DEFN	REFERENCES											
CRC10	01F2	0	1092	1101R											
CR020	01FF	0	1125	1099R	1127M	1128R									
CR905	0208	0	1137	1116R											
CR910	02C4	0	1133	1112R											
CR920	02C6	0	1135	1091R	1096M	1114R									
DZ000	00F2	0	0617	0327B	0348B	0645R	0655R	0714R	0774R	0970R	0971R	1022R	1023R	1097R	1105B
DZ010	00F7	0	0624	0644R	0648M	1126B									
DZ020	00F9	0	0626	0620B											
DZ060	01CC	0	0633	0659B											
DZ070	0102	0	0637	0823B	0829B	0855B	0867B								
DZ100	0104	0	0642	0626M	0627M	0722B									
DZ110	01CF	0	0649	0646M											
DZ180	0111	0	0654	0625B	0637M										
DZ185	0119	0	0660	0840B											
DZ190	012C	0	0665	0729B											
DZ210	0146	0	0718	0872B											
DZ215	014C	0	0726	0795B											
DZ220	015C	0	0733	0798B											
DZ230	0154	0	0739	0632B											
DZ232	0156	0	0740	0670B											
DZ235	0158	0	0742	0631M	0663B										
DZ240	015D	0	0746	0878B											
DZ250	016C	0	0761	0769B											
DZ280	016D	0	0762	0748M	0754R	0755R	0756M	0767M							
DZ300	0178	0	0774	0764B											
DZ330	017D	0	0778	0750M											
DZ340	018C	0	0792	0735B											
DZ350	0197	0	0807	0752M	0834R	0838B									
DZ380	01A4	0	0822	0819B											
DZ390	01A5	0	0823	0802B											
DZ400	01B6	0	0845	0809B	0836B										
DZ410	01CA	0	0868	0847B											
DZ900	0122	0	0676	0684R	0821R										
DZ901	0123	0	0677	0833R											
DZ902	0124	0	0678	0660R	0726R	0744M	0868R	0875M							
DZ904	0126	0	0680	0638R	0822M	0828M	0855M								
DZ905	0127	0	0681	0864M											
DZ906	0128	0	0682	0758M	0854R										
DZ907	0129	0	0683	0785M											
DZ908	012A	0	0684	0827R											
DZ909	012B	0	0685	0782M											
DZ910	012C	0	0686	0656R	0780M	0792R	0816R								
DZ911	012D	0	0687	0790M											
DZ912	012E	0	0688	0667R	0745M	0791R	0845R	0849M	0870R	0876R					
DZ913	012F	0	0689	0775R											
DZ914	0130	0	0690	0728R											
DZ915	0131	0	0691	0668R											
DZ916	0132	0	0692	0848R											
DZ920	0134	0	0694	0786R											
DZ925	0135	0	0695	0781R											
DZ930	0136	0	0696	0766R											
DZ935	0137	0	0697	0733R											
DZ940	0138	0	0698	0760R											
DZ945	0139	0	0699	0629M	0784R	0787R	0865R								
DZ950	013A	0	0700	0789R											
DZ955	013B	0	0701	0776R											
DZ960	013C	0	0702	0753R											
DZ965	013D	0	0703	0749R											
DZ970	013E	0	0704	0751R											
DZ975	013F	0	0705	0788M	0863R										
DZ980	0140	0	0706	0779R											
DZ985	0141	0	0707	0852R											
DZ990	0142	0	0708	0777M	0783R	0857R									
DZ995	0143	0	0709	0820R											
$ACDF	009F	0	0446	0703R	0704R	1095M	1115M								
$ACEX	001A	0	0965	0320M											
$CH12	00C6	0	0255	1135R											
$CIBA	00C5	0	0254	0313R	1092M	1113M									
$CIDN	01CB	0	0886	1098M											
$CILA	005A	0	0966	0346R											
$CPTR	007E	0	0383												
$CXR1	0017	0	0962	0323M											
$CYLN	009A	0	0436	0702R	0704R	1104M									
$DBSY	00EF	0	0559	0618M	0662M	0719M	0741M	0837M	0839M	0969R	1103M	1135R			
$DCYL	00A4	0	0457	0702R	1093M										
$DDSW	00DD	0	0539	0657M	0793M										
$DMPF	0019	0	0964	0325M											
$DUMP	003F	0	0319	0324R	0555B	1122B									
$FLSH	0071	0	0369												
$GCOM	0063	0	0352												
$GRIN	0064	0	0353												
$HASH	0014	0	0274	0960R	0961R	0962R	0963R	0964R	0965R						
$IOCT	0032	0	0294	0358R	0720M	0739M	1111M								
$IRFO	002C	0	0288	0523R											
$I200	0083	0	0483	0262R	0494R										
$I205	00B8	0	0489	0967R											
$I210	00EA	0	0490	0484M	0485M	0486M									
$I290	00C2	0	0495	0487M	0493R										
$I400	00C4	0	0515	0264R	0534R										

CROSS-REFERENCE

SYMBOL	VALUE	REL	DEFN	REFERENCES
$I403	00DC	0	0525	0522B
$I405	C0D3	C	C529	C968R
$I410	00D5	0	0530	C517M 0518M 0519M
$I425	00EA	0	0555	C554B
$I490	00DE	0	C540	C516M 0533R
$I492	00EC	0	0541	C520R
$I494	00E2	C	C544	0526R
$I496	C0E4	C	0549	0553R 0972R
$I499	006F	C	0365	0361R
$LEV2	C00A	0	0262	1100M
$LINK	0039	C	0307	0338R
$LKNM	0014	C	C960	C344M
$NDUP	0034	C	C296	
$NXFQ	CC35	C	0297	
$PAUS	00FC	0	C970	
$PRFT	0028	C	C283	C285R C734B
$PST1	0081	0	C389	C391R
$PST2	0C85	C	C395	C397R 0669B
$PST3	0C89	0	0401	0403R
$PST4	CC8D	C	C407	0409R
$RMSW	0016	0	C961	0337M
$RWCZ	00F1	0	C971	
$SCAT	0011	0	0271	0359R
$SNLT	00EF	0	C969	
$SSTS	006D	0	C364	C322M
$STOP	0091	0	0414	C265R 0416R
$SYSC	C0EC	0	C542	
$S000	0052	C	0336	C303B
$S100	0053	0	0337	0309B 0329B
$S150	0C59	0	C343	C966R
$S250	0065	C	C357	0321B 0345B 0349B 0362R
$S300	CC66	0	0358	C360B
$S900	003C	0	0311	C326R 0328R
$S910	003E	0	0314	0336R
$UFDR	CC7D	C	C382	
$UFIO	0079	0	C378	
$ZEND	01E0	0	C881	C611R 0613R 0882R
X2	CCF2	C	0714	C669B C775R 0776R 0777M 0779R 0780M 0781R 0782M 0783R 0784R C785M 0786R C787R 0788M C789R 0790M 0791R C792R C793M 0816R 0820R 0821R 0822M 0823B C827R 0828M 0829B 0833R 0845R 0848R 0849M 0852R 0854R C855M C857R 0859B C863R 0864M C865R 0867B 0868R 0870R 0875M 0876R
Y	C07F	0	1027	1090R 1093M 1094R 1095M 1097R 1103M 1104M 1105B 1111M 1113M 1115M 1117M 1122B 1126B
#ACIN	001C	0	0917	
#ANDU	C023	C	C923	
#BNDU	C028	0	C924	
#CBSW	000A	0	0900	
#CIAD	001B	0	C916	
#CIBA	003C	0	C928	
#CIDN	0037	0	C927	
#CSHN	C05A	C	C934	
#DBCT	0006	0	C896	
#DCSW	0018	0	0913	
#ECNT	0021	0	C922	
#ENTY	0010	0	0906	
#FCNT	0007	0	C897	
#FHOL	0014	C	C909	
#FLET	004B	0	C931	
#FMAT	0046	0	C930	
#FPAD	002D	0	C925	
#FSZE	0015	C	C910	
#GCNT	001F	0	C919	
#GRPH	CC1D	0	C918	
#JBSW	00C9	0	C899	
#LCNT	000B	0	0901	
#LOSW	001F	0	C920	
#MDF1	000D	0	C903	
#MDF2	CCCE	C	0904	
#MPSW	0CCC	0	C902	
#NAME	0004	0	0895	
#NCNT	000F	C	C905	
#PCID	0032	C	C926	
#PTOD	0C19	0	C914	
#PPTR	CC1A	C	C915	
#RP67	0011	0	C907	
#SCRA	C041	0	0929	
#SYSC	00C8	0	C898	
#TODR	0012	0	C908	
#UHOL	0016	0	C911	
#ULFT	C05C	0	C932	
#USZE	0017	0	C912	
#WSCT	0055	C	0933	
#X3SW	C02C	0	C921	
*BLCT	0002	0	1012	
*CIDN	0004	0	1029	
*CIL1	00AC	0	0955	0885R
*COPY	0005	0	1030	
*DCOM	0001	0	C977	
*DCTR	00C1	0	1031	

CROSS-REFERENCE

SYMBOL	VALUE	REL	DEFN	REFERENCES
'DTYP	00C8	0	1032	
'D7ID	CC96	C	C952	0612R
'FILE	CC03	0	0989	
'HDNG	0CC7	C	C981	1134R
'HWCT	CCC4	C	C990	
'ICAD	0CCC	0	C976	
'ISTV	0033	0	1C16	
'OVSW	001A	0	C999	
'RIAD	0002	C	C978	1136R
'RTBL	CCC6	0	C980	
'SLET	0003	C	C979	
'STRT	CCCC	C	C982	
'TCNT	0011	0	1021	0740R

Appendix H. Monitor System Sample Programs

Sample programs 1, 2, and 3 are provided with the monitor system. The first is a FORTRAN compilation, the second is an assembly, and the third is an RPG compilation (RPG is available on the Disk Monitor System, Version 2, card system only). All 3 programs are loaded, listed on the principal printer, and processed as monitor jobs.

The output of the FORTRAN program is printed on the printer specified on the IOCS control record. The output of the assembler program is printed on the console printer. The output of the RPG program is printed on the printer specified as the output device on a file description coding sheet.

Sample programs 4, 5, 6, and 7 are not provided with the monitor system. These programs illustrate techniques described in Chapter 6. "Programming Tips and Techniques."

1. FORTRAN SAMPLE PROGRAM

The FORTRAN sample program is listed as it runs on a 4K and an 8K system (the LIST ALL control record is removed for the 8K run). This program reads data cards supplied with the program and builds 3 files on disk; one in the user area, and 2 in working storage. The core and file maps for the program are described in Chapter 6.

The FORTRAN card sample program as supplied uses a 1442-6, or -7, and 1132 Printer, and disk. The paper tape sample program uses an 1134 Paper Tape Reader, a console printer, and disk. If your system does not have the required configuration, you must make the following changes to the program:

card SMFOR006 — If printed output is to a 1403 Printer, change the IOCS entry from 1132 PRINTER to 1403 PRINTER.

If printed output is to the console printer, change the IOCS entry from 1132 PRINTER to TYPEWRITER.

card SMFOR007 — If card input is from a 2501 Reader, change the IOCS entry from CARD to 2501 PRINTER.

card SMFOR023 — If card input is from a 2501 Reader, change M=2 to M=8.

card SMFOR024 — If printer output is to a 1403 Printer, change L=3 to L=5.

If the printer output is on a console printer, change L=3 to L=1.

FORTRAN Sample Program Run on 4K

```
// JOB T                                                SAMPLE              SMFOR000

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  C000        0ED0        0ED0         C000

// DUP                                                                      SMFOR001

*STOREDATA  WS  UA  FILEA    2                                              SMFOR002
CART ID 0ED0   DB ADDR  2EA0   DB CNT    0020

// * IBM 1130 FORTRAN SAMPLE PROGRAM                                        SMFOR003

// FOR                                                                      SMFOR004
*ONE WORD INTEGERS                                                          SMFOR005
*IOCS(DISK,1132 PRINTER)                                                    SMFOR006
*IOCS(CARD)                                                                 SMFOR007
*LIST ALL                                                                   SMFOR008
C     IBM 1130 FORTRAN SAMPLE PROGRAM                                       SMFOR009
C     SIMULTANEOUS EQUATION PROGRAM                                         SMFOR010
C                                                                           SMFOR011
      INTEGER V1,V2,V3                                                      SMFOR012
      DIMENSION A(10,10),X(10),B(126)                                       SMFOR013
      DEFINE FILE 101(1,100,U,V1),102(1,10,U,V2),103(1,100,U,V3)            SMFOR014
  301 FORMAT (1H1,20X15HINCOMPATIBILITY)                                    SMFOR015
  302 FORMAT (1H 20X41HMORE EQUATIONS THAN UNKNOWNS-NO SOLUTIONS)           SMFOR016
  303 FORMAT (1H 20X46HMORE UNKNOWNS THAN EQUATIONS-SEVERAL SOLUTIONS)      SMFOR017
  304 FORMAT (1H 20X15HSOLUTION MATRIX)                                     SMFOR018
  305 FORMAT (1H 20X8HMATRIX A)                                             SMFOR019
  306 FORMAT (1H 20X8HMATRIX B)                                             SMFOR020
  307 FORMAT (1H 20X10H A-INVERSE)                                          SMFOR021
  308 FORMAT (1H 20X24HDIAGONAL ELEMENT IS ZERO)                            SMFOR022
      M=2                                                                   SMFOR023
      L=3                                                                   SMFOR024
      READ (M,10)                                                           SMFOR025
   10 FORMAT(80H           SPACE FOR TITLE                                  SMFOR026
     1                             )                                        SMFOR027
      WRITE (L,10)                                                          SMFOR028
   12 FORMAT (6I10,20X)                                                     SMFOR029
      READ (M,12) M1,M2,L1,L2,N1,N2                                         SMFOR030
C                                                                           SMFOR031
C     M1 = NO. OF ROWS OF A                                                 SMFOR032
C     M2 = NO. OF COLS OF A                                                 SMFOR033
C     L1 = NO. OF ROWS OF X                                                 SMFOR034
C     L2 = NO. OF COLS OF X                                                 SMFOR035
C     N1 = NO. OF ROWS OF B                                                 SMFOR036
C     N2 = NO. OF COLS OF B                                                 SMFOR037
C                                                                           SMFOR038
   13 FORMAT (7F10.4,10X)                                                   SMFOR039
   17 FORMAT (10F10.4)                                                      SMFOR040
      IF (N2-1)63,64,63                                                     SMFOR041
   64 IF (L2-1)63,65,63                                                     SMFOR042
   65 IF (L1-M2)63,66,63                                                    SMFOR043
   66 IF (M1-N1)63,11,63                                                    SMFOR044
   63 WRITE (L,301)                                                         SMFOR045
      GO TO 2                                                               SMFOR046
   11 N=M1                                                                  SMFOR047
      N=M2                                                                  SMFOR048
      IF (M1-M2) 91,14,93                                                   SMFOR049
   91 WRITE (L,302)                                                         SMFOR050
      GO TO 2                                                               SMFOR051
   93 WRITE (L,303)                                                         SMFOR052
      GO TO 2                                                               SMFOR053
   14 WRITE (L,305)                                                         SMFOR054
      DO 70 I=1,N                                                           SMFOR055
      READ  (M,13) (A(I,J), J=1,N)                                          SMFOR056
      WRITE (L,17) (A(I,J), J=1,N)                                          SMFOR057
      WRITE (101'1)(A(I,J), J=1,N)                                          SMFOR058
   70 CONTINUE                                                              SMFOR059
   89 FORMAT (F10.4,70X)                                                    SMFOR060
      WRITE (L,306)                                                         SMFOR061
      READ  (M,89) (B(I), I=1,N)                                            SMFOR062
      WRITE (L,89) (B(I), I=1,N)                                            SMFOR063
      WRITE (102'1)(B(I), I=1,N)                                            SMFOR064
C                                                                           SMFOR065
C     INVERSION OF A                                                        SMFOR066
C                                                                           SMFOR067
      DO 120 K=1,N                                                          SMFOR068
      D=A(K,K)                                                              SMFOR069
      IF(D)40,200,40                                                        SMFOR070
   40 A(K,K)=1.0                                                            SMFOR071
      DO 60 J=1,N                                                           SMFOR072
   60 A(K,J)=A(K,J)/D                                                       SMFOR073
      IF(K-N)80,130,130                                                     SMFOR074
   80 IK=K+1                                                                SMFOR075
      DO 120 I=IK,N                                                         SMFOR076
      D=A(I,K)                                                              SMFOR077
      A(I,K)=0.0                                                            SMFOR078
      DO 120 J=1,N                                                          SMFOR079
  120 A(I,J)=A(I,J)-(D*A(K,J))                                              SMFOR080
```

```
C                                                                     SMFOR081
C     BACK SCLUTICN                                                   SMFOR082
C                                                                     SMFOR083
  130 IK=N-1                                                          SMFOR084
      DO 180 K=1,IK                                                   SMFOR085
      I1=K+1                                                          SMFOR086
      DC 180 I=I1,N                                                   SMFOR087
      D=A(K,I)                                                        SMFOR088
      A(K,I)=0.0                                                      SMFOR089
      DO 180 J=1,N                                                    SMFOR090
  180 A(K,J)=A(K,J)-(D*A(I,J))                                        SMFOR091
      GC TC 202                                                       SMFOR092
  200 WRITE (L,308)                                                   SMFOR093
      GC TC 2                                                         SMFOR094
  202 WRITE (L,307)                                                   SMFOR095
      DC 201 I=1,N                                                    SMFOR096
      WRITE (L,17)  (A(I,J), J=1,N)                                   SMFOR097
      WRITE (103'1) (A(I,J), J=1,N)                                   SMFOR098
  201 CONTINUE                                                        SMFOR099
      DO 21 I=1,N                                                     SMFOR100
      X(I)=0.0                                                        SMFOR101
      DO 21 K=1,N                                                     SMFOR102
   21 X(I)=X(I)+A(I,K)*B(K)                                           SMFOR103
      WRITE (L,304)                                                   SMFOR104
      WRITE (L,89) (X(I), I=1,N)                                      SMFOR105
    2 CALL EXIT                                                       SMFOR106
      END                                                             SMFOR107

VARIABLE ALLOCATICNS
     A(R )=00DC-0016    X(R )=00F0-00DE    B(R )=01EC-00F2    D(R )=01EE       V1(I )=01F0      V2(I )=01F1
    V3(I )=01F2         M(I )=01F3         L(I )=01F4        M1(I )=01F5       M2(I )=01F6      L1(I )=01F7
    L2(I )=01F8        N1(I )=01F9        N2(I )=01FA         N(I )=01FB        I(I )=01FC       J(I )=01FD
     K(I )=01FE        IK(I )=01FF        I1(I )=02C0

STATEMENT ALLCCATICNS
 301  =020E  302  =021B  303  =0235  304  =0251  305  =025E  306  =0267  307  =0270  308  =027A  10   =028B  12   =02B5
 13   =02B9  17   =02BD  89   =02C0  64   =0300  65   =0306  66   =03CC  63   =0312  11   =0318  91   =0328  93   =032E
 14   =0334  70   =0386  40   =03EB  60   =03FA  80   =0416  120  =0435  130  =0468  180  =0491  200  =04C6  202  =04CC
 201  =05C6  21   =0520  2    =056C

FEATURES SUPPCRTED
 CNE WCRD INTEGERS
 ICCS

CALLEC SUBPRCGRAMS
 FACDX   FMPYX   FDIV    FLD     FLDX    FSTO    FSTOX   FSBRX   CARDZ   PRNTZ   SRED    SWRT    SCOMP   SFIO    SIOFX
 SIOI    SUBSC   SDFIC   SDWRT   SDCOM   SDFX

REAL CCNSTANTS
 .100000E 01=0204     .000000E 00=0206

INTEGER CCNSTANTS
     2=0208       3=0209       1=020A     101=020B     102=020C     103=020D

CCRE REQUIREMENTS FOR
 COMMCN       0  VARIABLES     516  PROGRAM      874

ENC OF CCMPILATICN
```

FORTRAN Sample Program
run on 4K

```
// XEC        L  2                                                    SMFOR108

*LCCAL,FLCAT,FARC,IFIX,PAUSE,HOLEZ                                    SMFOR109

*FILES(103,FILEA)                                                     SMFOR110
FILES ALLCCATICN
  1C3 02EA  0C01  0EDC FILEA
  101 C000  0C01  0ED0  02EC
  102 C001  0C01  0ECO  02EC
STORAGE ALLCCATION
R 4C  03BF (HEX) ADDITICNAL CCRE REQUIRD
R 43  0124 (HEX) ARITH/FUNC SCCAL WD CNT
R 44  06B2 (HEX) FI/C, I/O SOCAL WD CNT
R 45  0?B6 (HEX) DISK FI/C SOCAL WD CNT
R 41  C004 (HEX) WCS UNUSED BY CORE LCAD
LIBF TRANSFER VECTOR
 XMCS   09AA SCCAL 1
 EBCTB  0F51 SCCAL 2
 HCLTB  0F15 SCCAL 2
 GETAC  0ED2 SCCAL 2
 NORM   07C0
 FACDX  0955 SCCAL 1
 FSBRX  092C SCCAL 1
 FMPYX  08F8 SOCAL 1
 FDIV   08A6 SOCAL 1
 FSTCX  076C
 FLCX   0788
 SCCOM  0978 SOCAL 3
 SDFX   08E3 SCCAL 3
 SDWRT  0901 SCCAL 3
 SICFX  09A6 SOCAL 2
 SUBSC  07A2
 SICI   09AA SCCAL 2
 SCCMP  0983 SCCAL 2
 SWRT   08A2 SCCAL 2
 SRED   08A7 SCCAL 2
 FSTO   0770
 FLD    078C
 PRNTZ  0CF8 SOCAL 2
 CARDZ  0C48 SCCAL 2
 SFIO   09BF SCCAL 2
 SDFIC  0960 SCCAL 3
 HCLEZ  086A LOCAL
 PAUSE  086A LOCAL
 IFIX   086A LOCAL
 FARC   086A LOCAL
 FLCAT  086A LOCAL
SYSTEM SUBROUTINES
 ILS04  0CC4
 ILS02  00B3
 ILS01  0F56
 ILS00  0F6F
 FLIPR  0804
      04C1 (HEX) IS THE EXECUTION ADDR

     IBM 1130 FORTRAN          SAMPLE PROGRAM                         SMFOR111
                          MATRIX A
   4.2150   -1.2120    1.1050
  -2.1200    3.5050   -1.6320
   1.1220   -1.3130    3.9860
                          MATRIX B
   3.2160
   1.2470
   2.3456
                           A-INVERSE
   0.2915    0.0833   -0.0467
   0.1631    0.3836    0.1118
  -0.0283    0.1029    0.3008
                          SOLUTION MATRIX
   0.9321
   1.2654
   0.7429
```

FORTRAN Sample Program Run on 8K

```
// JOB T                                            SAMPLE              SMFOR000

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0ED0        0ED0        0000
                          0ED4        0001

V2 M09   ACTUAL  8K  CONFIG  8K

// DUP                                                                  SMFOR001

*STOREDATA  WS  UA  FILEA    2                                          SMFOR002
CART ID 0ED0   DB ADDR  2EA0   DB CNT   0020

// * IBM 1130 FORTRAN SAMPLE PROGRAM                                    SMFOR003

// FOR                                                                  SMFOR004
*ONE WORD INTEGERS                                                      SMFOR005
*IOCS(DISK,1132 PRINTER)                                                SMFOR006
*IOCS(CARD)                                                             SMFOR007

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR
 COMMON       0  VARIABLES    516   PROGRAM    874

END OF COMPILATION

// XEQ        L  2                                                      SMFOR108

*LOCAL,FLOAT,FARC,IFIX                                                  SMFOR109

*FILES(103,FILEA)                                                       SMFOR110
FILES ALLOCATION
  103 02EA  0001  0ED0 FILEA
  101 0000  0001  0ED0  02EC
  102 0001  0001  0ED0  02EC
STORAGE ALLOCATION
R 41  6C08 (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 EBCTB  12BF
 HOLTB  1283
 GETAD  1240
 XMDS   1224
 HOLEZ  11EE
 PAUSE  11D8
 NORM   11AE
 FADDX  1159
 FSBRX  1130
 FMPYX  10FC
 FDIV   10AA
 FSTOX  1052
 FLDX   106E
 SDCOM  0842
 SDFX   07AD
 SDWRT  07CB
 SIOFX  0B26
 SUBSC  1088
 SIOI   0B2A
 SCOMP  0B03
 SWRT   0A22
```

```
 SRED    0A27
 FSTO    1056
 FLD     1072
 PRNTZ   0F78
 CARDZ   0EC8
 SFIO    0B3F
 SDFIO   082A
 IFIX    1338 LOCAL
 FARC    1338 LOCAL
 FLOAT   1338 LOCAL
SYSTEM  SUBROUTINES
 ILS04   00C4
 ILS02   00B3
 ILS01   1366
 ILS00   137F
 FLIPR   12D2
      04C1 (HEX) IS THE EXECUTION ADDR

     IBM 1130 FORTRAN          SAMPLE PROGRAM                              SMFOR111
                         MATRIX A
   4.2150   -1.2120     1.1050
  -2.1200    3.5050    -1.6320
   1.1220   -1.3130     3.9860
                         MATRIX B
   3.2160
   1.2470
   2.3456
                          A-INVERSE
   0.2915    0.0833    -0.0467
   0.1631    0.3836     0.1118
  -0.0283    0.1029     0.3008
                         SOLUTION MATRIX
   0.9321
   1.2654
   0.7429
```

2. ASSEMBLER SAMPLE PROGRAM

The core map printed with the assembler sample program is described in Chapter 6. "Programing Tips and Techniques."

output on
the principal
printer

```
// JOB                                                                        SMASM101

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000         0ED0        0ED0         0000

V2 M09   ACTUAL 32K   CONFIG 32K

// ASM                                                                        SMASM102
*LIST                                                                         SMASM103
*PRINT SYMBOL TABLE                                                           SMASM104

                                COMPUTE THE SQUARE ROOT OF 64

                    00001   ***************************************************** SMASM106
                    00002   *                                                   * SMASM107
                    00003   *   THIS PROGRAM COMPUTES THE SQUARE ROOT OF 64     * SMASM108
                    00004   *   *AND PRINTS THE RESULT IN THE CONSOLE PRINTER.* SMASM109
                    00005   *                                                   * SMASM110
                    00006   ***************************************************** SMASM111
0000 0  C030        00007   BEGIN LD     D64       INPUT TO THE SQUARE ROOT    SMASM112
0001 20 064D6063    00008         LIBF   FLOAT     INTEGER TO FLOATING PT.     SMASM113
0002 30 06898640    00009         CALL   FSQR      FLOATING PT. SQRT.          SMASM114
0004 20 091899C0    00010         LIBF   IFIX      FLOATING PT. TO INTEGER     SMASM115
0005 0  1008        00011         SLA    8                                     SMASM116
                    00012   *            MASK TO BUILD EBCDIC INTEGER          SMASM117
                    00013   *            RESULT AND EBCDIC BLANK IN WORD1.     SMASM118
0006 0  E829        00014         OR     MASK                                  SMASM119
0007 0  D01B        00015         STO    WORD1     CONVERSION INPUT AREA       SMASM120
                    00016   *            CONVERT MESSAGE FROM EBCDIC           SMASM121
                    00017   *            TO ROTATE/TILT CODE.                  SMASM122
0008 20 05097663    00018         LIBF   EBPRT     CALL CONVERSION SUBROUTINE  SMASM123
0009 0  0000        00019         DC     0         CONTROL PARAMETER           SMASM124
000A 1  0023        00020         DC     WORD1     INPUT AREA                  SMASM125
000B 1  0015        00021         DC     TYPE+1    OUTPUT AREA                 SMASM126
000C 0  001A        00022         DC     26        CHARACTER COUNT             SMASM127
000D 20 23A17170    00023         LIBF   TYPE0     TYPE MESSAGE                SMASM128
000E 0  2000        00024         DC     /2000     CONTROL PARAMETER           SMASM129
000F 1  0014        00025         DC     TYPE      I/O AREA                    SMASM130
0010 20 23A17170    00026   BUSY  LIBF   TYPE0     WAIT FOR TYPING COMPLETE    SMASM131
0011 0  0000        00027         DC                                           SMASM132
0012 0  7CFD        00028         MDX    BUSY      BR TO WAIT FOR COMPLETION   SMASM133
0013 0  6038        00029         EXIT             RETURN TO MONITOR CONTROL   SMASM134
0014 0  000E        00030   TYPE  DC     14        I/O AREA WORD COUNT         SMASM135
0015    000D        00031         BSS    13        RESERVE AS PRINT BUFFER     SMASM136
0022 0  8181        00032         DC     /8181     TWO CARRIAGE RETURNS        SMASM137
0023 0  0000        00033   WORD1 DC     *-*       CONVERSION INPUT AREA       SMASM138
0024    0018        00034         EBC    .IS THE SQUARE ROOT OF 64.            SMASM139
0030 0  F040        00035   MASK  DC     /F040     EBCDIC INTEGER MASK         SMASM140
0031 0  0040        00036   D64   DC     64        CONSTANT FOR SQUARE ROOT    SMASM141
0032    0000        00037         END    BEGIN                                 SMASM142
```

```
                    *SYMBOL TABLE*

  BEGIN  0000     BUSY   0010     D64    0031     MASK   0030     TYPE   0014
  WORD1  0023

  000 OVERFLCW SECTCRS SPECIFIED
  000 OVERFLCW SECTCRS REQUIRED
  006 SYMBCLS DEFINEC
   NC ERRCR(S) ANC   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// XEQ        L                                                          SMASM143
R 41   7908 (HEX) WCS UNUSED BY CORE LOAC
CALL TRANSFER VECTOR
 FSQR   0248
LIBF TRANSFER VECTOR
 FARC   069A
 XMCS   067E
 HCLL   062E
 PRTY   05DE
 EBPA   058E
 FACD   04CD
 FDIV   053C
 FLD    0488
 FACDX  04E3
 FMPYX  049E
 FSTO   046C
 FGETP  0452
 NORM   0428
 TYPEO  0312
 EBPRT  02AC
 IFIX   0280
 FLOAT  0230
SYSTEM SUBRCUTINES
 ILS04  0CC4
 ILS02  00B3
    01FE (HEX) IS THE EXECUTION ADDR
```

output on the console printer

```
8 IS THE SQUARE ROOT OF 64
```

3. RPG SAMPLE PROGRAM

The RPG program as supplied, uses 1442 input and 1132 output. If your system does not have the required configuration, you must make the following changes to the program:

card RGS009 — If card input is from a 2501 Card Reader, change READ42 to READ01.

card RGS010 — If printed output is to a 1403 Printer, change PRINTER to PRINT03. If printer output is on the console printer, change PRINTER to CONSOLE.

output on principal printer

```
// JOB

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0ED0        0ED0        0000
                          0ED4        0001

V2 M09   ACTUAL 32K  CONFIG 32K

// RPG

                                   V1-3  1130  RPG   RGSPL

SEQ NO  PG LIN    SPECIFICATIONS COL 6 - 74                                        ERRORS

                  H                                                                RGSPL
                  F*  1130 RPG SAMPLE PROGRAM.                                     RGS001
                  F*  THIS PROGRAM PRINTS AN ACCOUNTS RECEIVABLE REGISTER WITH     RGS002
                  F*  INVOICE TOTALS . CUSTOMER TOTALS PRINT AS A RESULT OF A CONTROL RGS003
                  F*  BREAK IN COLUMNS 39-43 OF THE INPUT CARD. CORRECT OUTPUT     RGS004
                  F*  APPEARS IN ACCOMPANYING DOCUMENTATION. CARDS ARE SORTED ON   RGS005
                  F*  COLUMNS 39-43 AND ARE IDENTIFIED BY AN ELEVEN PUNCH IN CARD  RGS006
                  F*  COLUMN 1.                                                    RGS007
                  F*                                                               RGS008
 0001   01 010    FINPUT   IPE F      80          READ42                          RGS009
 0002   01 020    FOUTPUT  O   F     120      OF  PRINTER                         RGS010
 0003   02 010    IINPUT   AA  01    1 Z-                                          RGS011
 0004   02 020    I                                      8  29 NAME                RGS012
 0005   02 030    I                                     30  310MONTH               RGS013
 0006   02 040    I                                     32  330DAY                 RGS014
 0007   02 050    I                                     34  380INVNO               RGS015
 0008   02 060    I                                     39  430CUSTNOL1            RGS016
 0009   02 070    I                                     44  450STATE               RGS017
 0010   02 080    I                                     46  480CITY                RGS018
 0011   02 090    I                                     74  802INVAMT              RGS019
 0012   03 010    C   01      INVAMT    ADD  TOTAL     TOTAL   82                  RGS020
 0013   03 020    C   01      INVAMT    ADD  GRPTOT    GRPTOT  82                  RGS021
 0014   04 010    OOUTPUT  H  201    1P                                            RGS022
 0015   04 020    O       OR         OF                                            RGS023
 0016   04 030    O                                 53 '      A C C O U N T S  R'  RGS024
 0017   04 040    O                                 77 ' E C E I V A B L E  R E '  RGS025
 0018   04 050    O                                 88 'R E G I S T E R'           RGS026
 0019   04 060    O        H  1      1P                                            RGS027
 0020   04 070    O       OR         OF                                            RGS028
 0021   04 080    O                                 25 'CUSTOMER'                  RGS029
 0022   04 090    O                                 80 'LOCATION         INVOICE'  RGS030
 0023   04 100    O                                109 'INVOICE DATE     INVOICE'  RGS031
 0024   04 110    O        H  2      1P                                            RGS032
 0025   04 120    O       OR         OF                                            RGS033
 0026   04 130    O                                 42 'NUMBER          CUSTOMER '  RGS034
 0027   04 140    O                                 46 'NAME'                      RGS035
 0028   04 150    O                                 79 'STATE   CITY       NUMBER' RGS036
 0029   04 160    O                                108 'MO      DAY      AMOUNT'   RGS037
 0030   05 010    O        D  2      01                                            RGS038
 0031   05 020    O                        CUSTNOZ  22                             RGS039
 0032   05 030    O                        NAME     53                             RGS040
 0033   05 040    O                        STATE Z  59                             RGS041
 0034   05 050    O                        CITY  Z  67                             RGS042
 0035   05 060    O                        INVNO Z  79                             RGS043
 0036   05 070    O                        MONTH Z  89                             RGS044
 0037   05 080    O                        DAY   Z  97                             RGS045
 0038   05 090    O                        INVAMT  109 '$    ,  0.   '             RGS046
 0039   05 100    O        T  2      L1                                            RGS047
 0040   05 110    O                        GRPTOT B 109 '$    ,  0.   '            RGS048
 0041   05 120    O                                110 '*'                         RGS049
 0042   05 130    O        T  2      LR                                            RGS050
 0043   05 140    O                        TOTAL   109 '$    ,  0.   '             RGS051
 0044   05 150    O                                111 '**'                        RGS052
```

INDICATORS

IND	DISP	IND	DISP	IND	DISP	IND	DISP	IND	DISP	IND	DISP
MR	0150	00	0151	OF	0152	OV	0153	1P	0154	L0	0155
L1	0156	L2	0157	L3	0158	L4	0159	L5	015A	L6	015B
L7	015C	L8	015D	L9	015E	LR	015F	H1	0160	H2	0161
H3	0162	H4	0163	H5	0164	H6	0165	H7	0166	H8	0167
H9	0168	01	0169								

FIELD NAMES

FIELD	DISP	L	T	D	FIELD	DISP	L	T	D	FIELD	DISP	L	T	D	FIELD	DISP	L	T	D
NAME	016A	022	A		MONTH	0181	002	N	0	DAY	0184	002	N	0	INVNO	0187	005	N	0
CUSTNO	018D	005	N	0	STATE	0193	002	N	0	CITY	0196	003	N	0	INVAMT	019A	007	N	2
TOTAL	01A2	008	N	2	GRPTOT	01AB	008	N	2										

LITERALS

LITERAL	LENGTH	TYPE	DISP	LITERAL	LENGTH	TYPE	DISP
A C C O U N T S R	24	A	01B4	E C E I V A B L E R E	24	A	01CD
R E G I S T E R	15	A	01E6	CUSTOMER	8	A	01F6
LOCATION INVOICE	22	A	01FF	INVOICE DATE INVOICE	23	A	0216
NUMBER CUSTOMER	24	A	022E	NAME	4	A	0247
STATE CITY NUMBER	24	A	024C	MO DAY AMOUNT	21	A	0265
, .	11	E	027B	*	1	A	0287
**	2	A	0289				

KEY ADDRESSES OF OBJECT PROGRAM

NAME OF ROUTINE	HEX DISP	NAME OF ROUTINE	HEX DISP
H + D LINES	04DE	TOTAL LINES	04EC
DETAIL CALCS	046E	TOTAL CALCS	047D
CHAIN ROUT 1	03D2	CONTROL FLD	03F5
LOW FIELD	042E	EXCPT LINES	04FA
CLOSE FILES	067B	FILE SEQ 1	02EC
FILE SEQ 2	0377		

END OF COMPILATION

```
// XEQ        L              R
R 41  6D16 (HEX) WDS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
 RGERR  0C24
 HLEBC  0A1A
LIBF TRANSFER VECTOR
 RGSI5  11E4
 RGBLK  11AA
 RGEDT  105A
 RGMV2  0FA6
 RGADD  0DDD
 RGSI1  0D80
 RGMV5  0C72
 RGMV3  0D50
 RGCMP  0CFE
 RGMV1  0C6A
 PRNT1  0A9A
 ZIPCO  097A
 CARDO  087C
SYSTEM SUBROUTINES
 ILSX4  1249
 ILSX2  126D
 ILSX1  1286
 ILSX0  12A3
     020F (HEX) IS THE EXECUTION ADDR
```

output on specified output device

ACCOUNTS RECEIVABLE REGISTER

CUSTOMER NUMBER	CUSTOMER NAME	LOCATION STATE	CITY	INVOICE NUMBER	INVOICE DATE MO	DAY	INVOICE AMOUNT
10712	AMALGAMATED CORP	33	61	11603	11	10	$ 389.25
							$ 389.25*
11315	BROWN WHOLESALE	30	231	12324	12	28	$ 802.08
11315	BROWN WHOLESALE	30	231	99588	12	14	$ 261.17
							$ 1,063.25*
11897	FARM IMPLEMENTS	47	77	10901	10	18	$ 27.63
							$ 27.63*
18530	BLACK OIL	16	67	11509	11	8	$ 592.95
18530	BLACK OIL	16	67	12292	12	23	$ 950.97
							$ 1,543.92*
20716	LEATHER BELT CO	36	471	11511	11	8	$ 335.63
20716	LEATHER BELT CO	36	471	12263	12	17	$ 121.75
							$ 457.38*
29017	GENERAL MFG CO	6	63	11615	11	14	$ 440.12
29017	GENERAL MFG CO	6	63	11676	11	23	$ 722.22
							$ 1,162.34*
29054	A-B-C DIST CO	25	39	9689	9	11	$ 645.40
29054	A-B-C DIST CO	25	39	11605	11	11	$ 271.69
29054	A-B-C DIST CO	25	39	12234	12	14	$ 559.33
							$ 1,476.42*
							$ 6,120.19**

4. USING FORTRAN UNFORMATTED I/O

This program is referred to under "Initializing $$$$$ Data Files for Use with FORTRAN Unformatted I/O" in Chapter 6.

```
// JCB     OECO

LCG DRIVE    CART SPEC    CART AVAIL    PHY DRIVE
  CCOO         OECO         OECO          COOC

V2 MO9    ACTUAL 32K   CCNFIG 32K

// CUP

*STCREDATA  WS   FX   $$$$$ 001C
CART IC CECO    CB ACCR   1F70    DB CNT    CCAO

// FCR
*ICCS(LCISK)
*LIST ALL
*NAME UNFCX
      CIMENSICN A(200),B(24),C(3CO),E(12),F(3CO)
      CATA A/200*4.0/,B/24*5.C/,C/300*6.0/
      WRITE (10)A
      WRITE (10)B
      WRITE (10)C
      ENC FILE 10
      BACKSPACE 1C
      BACKSPACE 1C
      REAC(1C)F
      REWINC 10
      REAC(1C)
      REAC(1C)E
      PAUSE 9999
      CALL EXIT
      END
VARIABLE ALLCCATICNS
     A(R )=018E-CCCO      B(R )=01BE-0190     C(R )=0416-01CO     E(R )=042E-0418     F(R )=0686-0430

FEATURES SUPPCRTEC
 ICCS

CALLEC SUBPRCGRAMS
 URED     UWRT     UCCMP    BCKSP    ECF      REWNC    PAUSE    UFIO     UIOAF

INTEGER CCNSTANTS
    1C=0688     9999=0689  -26215=068A

CCRE REQUIREMENTS FCR UNFCX
 CCMMCN       C  VARIABLES    1672  PROGRAM      52

ENC CF CCMPILATICN

// CUP

*STCRE       WS   UA   UNFCX
CART IC OECO    CB ACCR   2E50    DB CNT    CC41

// XEQ UNFCX
```

5. PROCESSING ON ONE DISK DRIVE A FILE THAT EXTENDS OVER TWO CARTRIDGES

This program is referred to under "Reeling" in the section "SYSUP" in Chapter 6.

```
// JOB    OEDO

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0EDO        0EDO        0000

V2 M09   ACTUAL 32K  CONFIG 32K

// FOR
*NAME LINK2
*IOCS(1132 PRINTER)
*IOCS(DISK)
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      DIMENSION J(320)
      DEFINE FILE 2(200,320,U,K)
      K = 1
      L = 0
      DO 5 I = 1, 199
      L = L + 1
      DO 4 N = 1, 320
    4 J(N) = L
    5 WRITE (2'K) J
      L = 999
      DO 6 N = 1, 320
    6 J(N) = L
      WRITE (2'K) J
      WRITE (3,10)
   10 FORMAT(/' LINK NO. 2 EXECUTED.'/)
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR LINK2
 COMMON       0  VARIABLES    334  PROGRAM    142

END OF COMPILATION

// DUP

*DUMP        WS   CD   LINK2
CART ID 0EDO  DB ADDR 4530   DB CNT 000B

// FOR
*NAME LINK1
*IOCS(DISK,1132 PRINTER)
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      DIMENSION J(320)
      DIMENSION L(2)
      DEFINE FILE 1(210,320,U,K)
      K = 1
      L(2) = 3796
      L(1) = 0
      M = 0
      DO 5 I = 1, 209
      M = M + 1
      DO 4 N = 1, 320
    4 J(N) = M
    5 WRITE (1'K) J
      M = 999
      DO 6 N = 1, 320
    6 J(N) = M
```

```
   WRITE (1'K) J
   WRITE (3,40)
40 FORMAT (40HOLINK NO. 1 EXECUTED. CHANGE CARTRIDGES.///)
   PAUSE 1111
   CALL SYSUP (L(2))
   CALL LINK (LINK2)
   END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR LINK1
 COMMON      0  VARIABLES     336  PROGRAM     180

END OF COMPILATION

// DUP

*STORECI     WS  UA   LINK1 0001
*FILES(1,DATA,0ED0)
FILES ALLOCATION
    1 0206  00D2  0ED0 DATA
STORAGE ALLOCATION
R 41  6B6C (HEX) WDS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
 FSYSU  13F1
 FSLEN  1205
 SYSUP  0CA2
LIBF TRANSFER VECTOR
 NORM   1418
 FLOAT  11FA
 IFIX   11CE
 PAUSE  0C8C
 SCOMP  0799
 SWRT   06B8
 SDCOM  04D8
 SDAI   043A
 SDWRT  0461
 SUBSC  0C6E
 FSTO   0C3C
 FLD    0C58
 PRNTZ  0B5E
 SFIO   07D5
 SDFIO  04C0
SYSTEM SUBROUTINES
 ILS04  00C4
 ILS02  00B3
 ILS01  1444
      0370 (HEX) IS THE EXECUTION ADDR
CART ID 0ED0   DB ADDR  4530   DB CNT   00F0

// PAUS                         CHANGE TO CARTRIDGE 0ED4

// JOB    0ED4

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0ED4        0ED4        0000

V2 M09   ACTUAL 32K  CONFIG 32K

// DUP

*STORECI     CD  FX  LINK2 0001
*FILES(2,DATA2,0ED4)
FILES ALLOCATION
    2 01F7  00C8  0ED4 DATA2
STORAGE ALLOCATION
R 41  72D8 (HEX) WDS UNUSED BY CORE LOAD
```

```
LIBF TRANSFER VECTOR
 NORM    0CB0
 FLOAT   0CA6
 IFIX    0C7A
 PAUSE   0C64
 SCOMP   0771
 SWRT    0690
 SDCOM   04B0
 SDAI    0412
 SDWRT   0439
 SUBSC   0C46
 FSTO    0C14
 FLD     0C30
 PRNTZ   0B36
 SFIO    07AD
 SDFIO   0498
SYSTEM SUBROUTINES
 ILS04   00C4
 ILS02   00B3
 ILS01   0CDC
      0362 (HEX) IS THE EXECUTION ADDR
CART ID 0ED4   DB ADDR  3230   DB CNT   00A0

// PAUS                          CHANGE TO CARTRIDGE 0ED0

// JOB    0ED0

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0ED0        0ED0         0000

V2 M09   ACTUAL 32K   CONFIG 32K

// XEQ LINK1

LINK NO. 1 EXECUTED. CHANGE CARTRIDGES.

LOG DRIVE   CART SPEC   CART AVAIL   PHY DRIVE
  0000        0ED4        0ED4         0000

LINK NO. 2 EXECUTED.
```

6. PROCESSING ON TWO DISK DRIVES A FILE THAT EXTENDS OVER TWO CARTRIDGES

This program is referred to under "Reeling" in the section "SYSUP" in Chapter 6.

```
// JOB    0ED0 0ED4

LOG DRIVE   CART SPEC   CART AVAIL  PHY DRIVE
  0000        0ED0        0ED0        0000
  0001        0ED4        0ED4        0001

V2 M09   ACTUAL 32K  CONFIG 32K

// FOR
*NAME MDEX1
*IOCS (DISK)
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM
      DIMENSION J(320)
      DEFINE FILE 1(210,320,U,K)
      DEFINE FILE 2(200,320,U,KK)
      M = 111
      K = 1
      KK = 1
      DO 2 N = 1, 320
    2 J(N) = M
      DO 3 I = 1, 209
    3 WRITE (1'K) J
      M = 999
      DO 5 N = 1, 320
    5 J(N) = M
      WRITE (1'K) J
      M = 222
      DO 7 N = 1, 320
    7 J(N) = M
      DO 8 I = 1, 199
    8 WRITE (2'KK) J
      M = 999
      DO 9 N = 1,320
    9 J(N) = M
      WRITE (2'KK) J
      CALL EXIT
      END

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CORE REQUIREMENTS FOR MDEX1
 COMMON       0  VARIABLES     340  PROGRAM     178

END OF COMPILATION

// DUP

*STORE      WS  UA  MDEX1
CART ID 0ED0    DB ADDR  4515   DB CNT   000D

// XEQ MDEX1 L  2

*FILES(1,DATA,0ED0)

*FILES(2,DATA2,0ED4)
FILES ALLOCATION
    1 0206  00D2  0ED0 DATA
    2 01F7  00C8  0ED4 DATA2
STORAGE ALLOCATION
R 41  78FA (HEX) WDS UNUSED BY CORE LOAD
LIBF TRANSFER VECTOR
 PAUSE  06D8
 SDCOM  04DA
 SDAI   043C
 SDWRT  0463
 SUBSC  06BA
 SDFIO  04C2
SYSTEM SUBROUTINES
 ILS04  00C4
 ILS02  00B3
      035A (HEX) IS THE EXECUTION ADDR
```

7. CALCULATING ISAM FILE PARAMETERS

This program is referred to under "Indexed Sequential Access Method" in the section "Calculating Sequentially Organized and ISAM File Sizes" in Chapter 6. This program does no error checking.

For this program, you are requested to enter the first 4 values. The input fields are 5 characters long; enter right-justified decimal numbers (leading zeros are required). Press EOF on the console keyboard after each entry. The requests for your entries are as follows:

```
            ISAM FILE LOAD CALCULATIONS


 INDEX ENTRY LENGTH IN WORDS =
RECORD LENGTH IN WORDS =
NUMBER OF RECORDS TO BE LOADED =
NUMBER OF OVERFLOW SECTORS =
NUMBER OF INDEXES PER SECTOR =
NUMBER OF RECORDS PER SECTOR =
NUMBER OF PRIME DATA CYLINDERS =
NUMBER OF PRIME DATA SECTORS =
NUMBER OF INDEX SECTORS =

TOTAL NUMBER OF SECTORS =
```

After you enter the number of overflow sectors, the program calculates the file size. The following is a sample of the program output:

```
            ISAM FILE LOAD CALCULATIONS


 INDEX ENTRY LENGTH IN WORDS = 00010
RECORD LENGTH IN WORDS = 00100
NUMBER OF RECORDS TO BE LOADED = 00250
NUMBER OF OVERFLOW SECTORS = 00009
NUMBER OF INDEXES PER SECTOR = 00032
NUMBER OF RECORDS PER SECTOR = 00003
NUMBER OF PRIME DATA CYLINDERS = 00011
NUMBER OF PRIME DATA SECTORS = 00084
NUMBER OF INDEX SECTORS =00001 .

TOTAL NUMBER OF SECTORS =00095 .
```

The program that computes file size is listed as follows:

```
// JCB

LCG CRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  CCOC          OEC1         OEC1         COOC

// ASM
*XREF

CCCC 20 23A1717C CCC01     START LIBF     TYPEO                                     ISM00010
0CC1 C  2CC0     CC002           DC       /2C00        TYPE HEADING LINE            ISM00020
0002 1  CC1B     C0003           DC       HEAC                                      ISM00030
CCC3 20 23A17170 CCCC4     WAIT4 LIBF     TYPEO                                     ISMC0040
0004 0  CC00     CC005           DC       /CCCO        WAIT FOR CCNSOLE             ISMC0050
0C05 C  7CFC     CC006           B        WAIT4                                     ISM00060
C006 CC 65C00C04 CC007           LCX  L1 4             SET COUNT                    ISMC0070
0C08 01 C5C00016 CC008     IN    LC   L1 BTAB1         LOAD ADDR CF MESSAGE         ISM00080
0C0A 0  D002     C0009           STO      MESS         STCRE FOR SUBROUTINE         ISM00090
0C0B 20 23A1717C CC010           LIBF     TYPEO                                     ISM00100
0CCC C  2CC0     CC011           DC       /2CC0        TYPE MESSAGE                 ISMC0110
CCCC C  CCC0     CC012     MESS  DC       *-*                                       ISM00120
CCCE 2C 23A1717C CC013     WAIT1 LIBF     TYPEO                                     ISM00130
0C0F C  CCC0     CC014           DC       /CC00        WAIT FOR CCNSOLE             ISM00140
C010 0  7CFC     CC015           B        WAIT1                                     ISM00150
C011 C0 6600C005 CC016           LCX  L2 5             SET CHARACTER                ISM00160
0013 01 6EC0C08C CC017           STX  L2 IC            *CCUNT FOR TYPEO             ISMC0170
0C15 C  7C5B     CC018           B        CCNV                                      ISM00180
0016 0  1CC0     CC019     BTAB1 NOP                   TABLE OF ACCRESSES           ISM00190
0C17 1  0C61     C0020           DC       MESS4        *FCR MESSAGES                ISM00200
C018 1  CC4F     CC021           DC       MESS3        *FCR INPUT                   ISM00210
CC19 1  CC41     CC022           DC       MESS2                                     ISM00220
001A 1  0C30     C0023           DC       MESS1                                     ISM00230
001B 0  CC14     C0024     HEAC  DC       2C           WORD COUNT FOR HEACING       ISM00240
001C    CC28     CC025           DMES     'R'1OSISAM FILE LCAD CALCULATIONS'R'E     ISM00250
C03C C  CC10     CC026     MESS1 DC       16                                        ISM00260
0031    0C20     00027           DMES     '2RINCEX ENTRY LENGTH IN WORDS = 'E       ISM00270
C041 C  CCOC     CC028     MESS2 DC       13                                        ISM00280
C042    CC1A     C0029           DMES     'RRECCRD LENGTH IN WORDS = 'E             ISM00290
C04F C  CC11     CC030     MESS3 DC       17                                        ISM00300
C050    CC22     C0031           DMES     'RNUMBER CF RECORDS TC BE LCACED = 'E     ISM00310
0061 C  CC0F     C0032     MESS4 DC       15                                        ISM00320
0062    0C1E     C0033           DMES     'RNUMBER CF OVERFLOW SECTCRS = 'E         ISM00330
0071 20 23A1717C CC034     CONV  LIBF     TYPEO                                     ISM00340
0072 0  1CC0     CC035           DC       /1C00        READ IN VALUE                ISM00350
0073 1  0080     C0036           DC       IC                                        ISM00360
0074 20 23A17170 CC037     WAIT  LIBF     TYPEO                                     ISM00370
0C75 0  CCC0     CC038           DC       /CCC0        WAIT ON KEYBCARD             ISM00380
0076 C  7CFC     C0039           B        WAIT                                      ISM00390
0C77 C  CC0E     CC040           LC       OUT1         MOVE PLUS SIGN TO            ISM00400
0078 C  CC07     CC041           STC      IC           *IC AREA FCR DCBIN           ISMC0410
0079 20 040C2255 CC042           LIBF     DCBIN        CONVERT FROM IBM CARD CODE   ISMC0420
0C7A 1  CC80     CC043           DC       IC           *TC BINARY VALUE IN ACC      ISM00430
007B 01 C5C0008C CC044           STC  L1 VTAB1         STCRE IN VALUE TABLE         ISMC0440
007C C  71FF     CC045           MCX   1 -1            DECREMENT COUNT              ISMC0450
C07E C  7C89     CC046           B        IN           BRANCH IF COUNT NON-ZERO     ISMC0460
CC7F C  7C11     CC047           B        CAL          OTHERWISE TAKE THIS BRANCH   ISM00470
0080 C  CC05     CC048     IO    DC       5            INPUT AREA FOR KEYBOARD      ISM00480
0081    CCC5     CC049           BSS      5                                         ISM00490
0086 C  8CA0     CC050     OUT1  DC       /80A0        CONVERSION AREA              ISM00500
0087    C005     CC051     OUT   BSS      5                                         ISM00510
C08C 0  1CC0     CC052     VTAB1 NOP                   TABLE OF INPUT VALUES        ISM00520
008C 0  0CC0     CC053     OVRSC DC       *-*          NO. CF OVERFLOW SECTORS      ISM00530
C08E C  CC00     CC054     RECRD DC       *-*          NO. OF RECCRDS               ISM00540
008F 0  0C00     C0055     LENCR DC       *-*          RECORD LENGTH                ISM00550
0090 C  CCC0     C0056     LENGI DC       *-*          INCEX ENTRY LENGTH           ISM00560
0091 C  CC22     CC057     CAL   LD       SCTLG        DIVID SECTCR LENGTH BY       ISM00570
0092 C  1890     CC058           SRT      16           *INDEX ENTRY LENGTH TO       ISM00580
0093 0  A8FC     C0059           D        LENGI        *CALCULATE THE NUMBER OF     ISM00590
```

```
0094 0  DC27     00060         STO      IEPS    *INDEX ENTRIES PER SECTOR  ISM00600
0095 0  C0F9     00061         LD       LENGR   CREATE DIVISOR BY ADDING   ISM00610
0096 0  801B     00062         A        TWO     *TWO TO THE RECORD SIZE AN ISM00620
0097 0  D01D     00063         STO      WORK    *STORING IN HOLD AREA      ISM00630
0098 0  C01B     00064         LD       SCTLG   DIVIDE RECORD LENGTH+2     ISM00640
0099 0  1890     00065         SRT      16      *INTO THE SECTOR LENGTH    ISM00650
009A 0  A81A     00066         D        WORK    *TO CALCULATE THE NUMBER   ISM00660
009B 0  D01F     00067         STO      RCDPS   *OF RECORDS PER SECTOR     ISM00670
009C 0  C0F1     00068         LD       RECRD   DIVIDE THE TOTAL NUMBER OF ISM00680
009D 0  801D     00069         A        RCDPS   *RECORDS PLUS NO. OF REC.  ISM00690
009E 0  9012     00070         S        ONE     *PER SECTOR MINUS ONE      ISM00700
009F 0  1890     00071         SRT      16      *BY THE NUMBER OF          ISM00710
00A0 0  A81A     00072         D        RCDPS   *REC. PER SECTOR TO FIND   ISM00720
00A1 0  D017     00073         STO      NOPDS   *NO. OF PRIME DATA SECTORS ISM00730
00A2 0  8010     00074         A        SEVEN   ADD CONSTANT OF SEVEN      ISM00740
00A3 0  1803     00075         SRA      3       DIVIDE BY 8 TO DETERMINE   ISM00750
00A4 0  D015     00076         STO      NOPDC   *NO. OF PRIME DATA CYLNDRS ISM00760
00A5 0  8016     00077         A        IEPS    ADD INDEX ENTRIES/SECTOR   ISM00770
00A6 0  900A     00078         S        ONE     MINUS ONE                  ISM00780
00A7 0  1890     00079         SRT      16      DIVIDE BY NO. OF INDEX     ISM00790
00A8 0  A813     00080         D        IEPS    *ENTRIES/SECTOR TO FIND NO ISM00800
00A9 0  D00E     00081         STO      NOISC   *OF INDEX SECTORS          ISM00810
00AA 0  800E     00082         A        NOPDS   ADD NO OF INDEX SECTORS+   ISM00820
00AB 0  80E1     00083         A        OVRSC   *NO. OF PRIME DATA SECTORS ISM00830
00AC 0  8004     00084         A        ONE     * + NO OF OVERFLOW SECTORS ISM00840
00AD 0  D009     00085         STO      TOTSC   * + 1 FOR LABEL            ISM00850
00AE 00 65000006 00086         LDX   L1 6       SET COUNT                  ISM00860
00B0 0  7013     00087         B        ROUT                               ISM00870
00B1 0  0001     00088 ONE     DC       1       CONSTANT OF ONE            ISM00880
00B2 0  0002     00089 TWO     DC       2       CONSTANT OF TWO            ISM00890
00B3 0  0007     00090 SEVEN   DC       7       CONSTANT OF SEVEN          ISM00900
00B4 0  0140     00091 SCTLG   DC       320     NO. WORDS PER SECTOR       ISM00910
00B5 0  0000     00092 WORK    DC       *-*     TEMPORARY HOLD AREA        ISM00920
00B6 0  1000     00093 VTAB2   NOP              TABLE OF OUTPUT VALUES     ISM00930
00B7 0  0000     00094 TOTSC   DC       *-*     TOTAL NO. OF SECTORS       ISM00940
00B8 0  0000     00095 NOISC   DC       *-*     NO. OF INDEX SECTORS       ISM00950
00B9 0  0000     00096 NOPDS   DC       *-*     TOT NO. OF PRIME DATA SCTR ISM00960
00BA 0  0000     00097 NOPDC   DC       *-*     NO. OF PRIME DATA CYLINDRS ISM00970
00BB 0  0000     00098 RCDPS   DC       *-*     NO. OF RECORDS PER SECTOR  ISM00980
00BC 0  0000     00099 IEPS    DC       *-*     NO. OF INDEX ENTRIES/SECTR ISM00990
00BD 0  1000     00100 MTAB    NOP              MESSAGE TABLE CONTAINS     ISM01000
00BE 1  00F4     00101         DC       MS5P    *ADDRESS IN MESSAGES       ISM01010
00BF 1  0105     00102         DC       MS6P    *WHERE THE VALUES ARE TO   ISM01020
00C0 1  0119     00103         DC       MS7P    *BE INSERTED               ISM01030
00C1 1  012E     00104         DC       MS8P                               ISM01040
00C2 1  0142     00105         DC       MS9P                               ISM01050
00C3 1  0156     00106         DC       MS10P                              ISM01060
00C4 01 C50000BD 00107 ROUT    LD    L1 MTAB    MOVE ADDR OF CORRECT       ISM01070
00C6 01 D40000CF 00108         STO   L  ADDR    *MSG TO CONVERT ROUTINE    ISM01080
00C8 01 C50000B6 00109         LD    L1 VTAB2   LOAD A VALUE TO CONVERT    ISM01090
00CA 20 02255103 00110         LIBF     BINDC   GO CONVERT FROM BINARY TO  ISM01100
00CB 1  0086     00111         DC       OUT1    *IBM CARD CODE             ISM01110
00CC 20 292570C6 00112         LIBF     ZIPCO   CONVERT FROM IBM CARD CODE ISM01120
00CD 0  1100     00113         DC       /1100   *TO CONSOLE CODE AND       ISM01130
00CE 1  0087     00114         DC       OUT     *PLACE VALUE IN            ISM01140
00CF 0  0000     00115 ADDR    DC       *-*     *MESSAGE                   ISM01150
00D0 0  0005     00116         DC       5                                  ISM01160
00D1 30 085930C7 00117         CALL     HOLCP                              ISM01170
00D3 01 C50000CF 00118         LD    L1 BTAB2   LOAD ADDR OF MESSAGE       ISM01180
00D5 0  D002     00119         STO      MESSP   STORE ADDR FOR SUBROUTINE  ISM01190
00D6 20 23A17170 00120         LIBF     TYPE0                              ISM01200
00D7 0  2000     00121         DC       /2000   TYPE OUTPUT MESSAGE        ISM01210
00D8 0  0000     00122 MESSP   DC       *-*                                ISM01220
00D9 20 23A17170 00123 WAIT3   LIBF     TYPE0                              ISM01230
00DA 0  0000     00124         DC       /0000   WAIT FOR CONSOLE           ISM01240
00DB 0  70FD     00125         B        WAIT3                              ISM01250
```

```
00DC 0   71FF      00126           MDX    1 -1         DECREMENT COUNT             ISM01260
00DD 0   70E6      00127           B       RCUT        IF COUNT NON-ZERO BRANCH    ISM01270
00DE 0   6C38      00128           EXIT                OTHERWISE, CALL EXIT        ISM01280
00DF 0   1000      00129    BTAB2  NOP                 TABLE OF ADDRESSES          ISM01290
00E0 1   00E6      00130           DC      MS5                                     ISM01300
00E1 1   00F7      00131           DC      MS6                                     ISM01310
00E2 1   0108      00132           DC      MS7                                     ISM01320
00E3 1   011C      00133           DC      MS8                                     ISM01330
00E4 1   0131      00134           DC      MS9                                     ISM01340
00E5 1   0145      00135           DC      MS10                                    ISM01350
00E6 0   0011      00136    MS5    DC      17                                      ISM01360
00E7     001B      00137           DMES    'RTOTAL NUMBER OF SECTORS = '           ISM01370
00F4     0005      00138    MS5P   DMES        'E                                  ISM01380
00F7 0   0011      00139    MS6    DC      17                                      ISM01390
00F8     001B      00140           DMES    'RNUMBER OF INDEX SECTORS = '           ISM01400
0105     0005      00141    MS6P   DMES        'E                                  ISM01410
0108 0   0013      00142    MS7    DC      19                                      ISM01420
0109     0020      00143           DMES    'RNUMBER OF PRIME DATA SECTORS = '      ISM01430
0119     0006      00144    MS7P   DMES        'E                                  ISM01440
011C 0   0014      00145    MS8    DC      20                                      ISM01450
011D     0022      00146           DMES    'RNUMBER OF PRIME DATA CYLINDERS = '    ISM01460
012E     0006      00147    MS8P   DMES         E                                  ISM01470
0131 0   0013      00148    MS9    DC      19                                      ISM01480
0132     0020      00149           DMES    'RNUMBER OF RECORDS PER SECTOR = '      ISM01490
0142     0006      00150    MS9P   DMES        'E                                  ISM01500
0145 0   0013      00151    MS10   DC      19                                      ISM01510
0146     0020      00152           DMES    'RNUMBER OF INDEXES PER SECTOR = '      ISM01520
0156     0006      00153    MS10P  DMES        'E                                  ISM01530
015A     0000      00154           END     START                                   ISM01540
```

```
                         CROSS-REFERENCE

SYMBOL      VALUE  REL DEFN     REFERENCES

ADDR        00CF   1   00115    00108,M
BTAB1       0016   1   00019    00008,R
BTAB2       00DF   1   00129    00118,R
CAL         0091   1   00057    00047,B
CONV        0071   1   00034    00018,B
HEAD        001B   1   00024    00003,R
IEPS        00BC   1   00099    00060,M 00077,R 00080,R
IN          0008   1   00008    00046,B
IO          0080   1   00048    00017,M 00036,R 00041,M 00043,R
LENGI       0090   1   00056    00059,R
LENGR       008F   1   00055    00061,R
MESS        000D   1   00012    00009,M
MESSP       00D8   1   00122    00119,M
MESS1       003C   1   00026    00023,R
MESS2       0041   1   00028    00022,R
MESS3       004F   1   00030    00021,R
MESS4       0061   1   00032    00020,R
MS10        0145   1   00151    00135,R
MS10P       0156   1   00153    00106,R
MS5         00E6   1   00136    00130,R
MS5P        00F4   1   00138    00101,R
MS6         00F7   1   00139    00131,R
MS6P        0105   1   00141    00102,R
MS7         0108   1   00142    00132,R
MS7P        0119   1   00144    00103,R
MS8         011C   1   00145    00133,R
MS8P        012E   1   00147    00104,R
MS9         0131   1   00148    00134,R
MS9P        0142   1   00150    00105,R
MTAB        00BD   1   00100    00107,R
NOISC       00B8   1   00095    00081,M
NOPCC       00BA   1   00097    00076,M
NOPDS       00B9   1   00096    00073,M 00082,R
```

```
CNE      C0B1   1   CC088    COC70,R CC078,R CC084,R
CLT      C087   1   C0051    C0114,R
CLT1     C086   1   C005C    CC040,R 00111,R
CVRSC    C08C   1   C0053    C0083,R
RCCPS    C0BB   1   C0098    C0067,M CC069,R CC072,R
RECRD    008E   1   C0054    CC068,R
RCLT     CCC4   1   CC1C7    C0087,B CC127,B
SCTLG    C0B4   1   00091    00057,R CC064,R
SEVEN    00B3   1   00090    CC074,R
START    0CCC   1   CC001    C0154,R
TCTSC    C0B7   1   CC094    C0085,M
TWC      C0B2   1   00089    C0062,R
VTAB1    008C   1   CC052    CC044,M
VTAB2    C0B6   1   CC093    C01C9,R
WAIT     C074   1   C0037    C0C39,B
WAIT1    000E   1   CC013    C0015,B
WAIT3    COC9   1   C0123    00125,B
WAIT4    0C03   1   CC004    CCC06,B
WCRK     C0B5   1   C0092    CC063,M C0066,R

   CCC OVERFLCW SECTCRS SPECIFIEC
   COC OVERFLCW SECTCRS REQUIRED
   052 SYMBCLS DEFINEC

NC ERRCR(S) ANC   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY
```

Appendix I. Formats

The general formats in which information is stored and dumped by the monitor system are:

- Disk
- Card
- Paper tape
- Data

Programs and subroutines are assigned type and subtype numbers that are placed in the program or subroutine header. The program types are defined as follows:

Type	Type of program
1	Mainline (absolute)
2	Mainline (relocatable)
3	Subprogram, not an ISS, referenced by an LIBF statement
4	Subprogram, not an ISS, referenced by a CALL statement
5	Interrupt service subroutine (ISS), referenced by an LIBF statement
6	Interrupt service subroutine (ISS), referenced by a CALL statement
7	Interrupt level subroutine (ILS)

Subtypes are defined for program types 3, 4, 5, and 7 only. When not used, the subtype indicator in a program header contains a zero. Program subtypes are defined as follows:

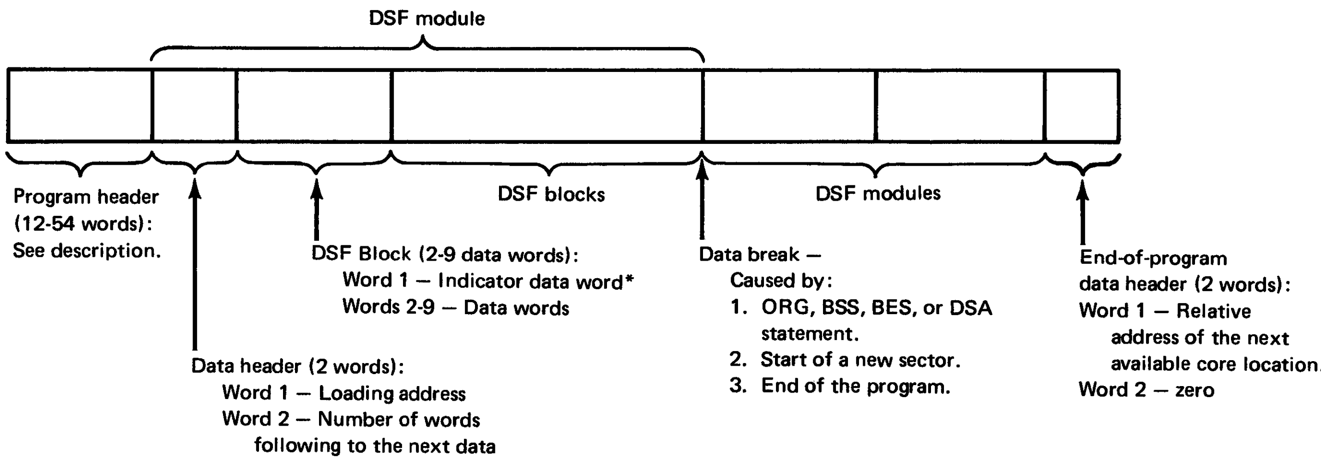
Subtype	Type	Description
0	3, 4	In-core subprograms
1	3	Disk FORTRAN I/O subroutines
2	3	Arithmetic subroutines
3	3	Nondisk FORTRAN I/O and "Z" conversion subroutines
3	5	"Z" device subroutines
8	4	Function subprogram
1	7	Dummy ILS02, ILS04

Monitor system formats are described in the following text.

DISK FORMATS

DSF format

Disk system format is the format in which absolute and relocatable programs (mainlines and subroutines) are stored on disk. The layout of a program stored in DSF format is shown in Figure I-1.



*The bits of the indicator data word describe the corresponding data word as follows:

00	Absolute
01	Relocatable
1000	LIBF
1100	CALL
1101	DSA

Figure I-1. Disk system format

The format of words 1 through 12 of the program header is the same for all program types. The following shows the contents of words 1 through 12 of a program header:

Word	Contents
1	Zero
2	Checksum, if the source was cards; otherwise, a zero.
3	Subtype (bits 0 through 3) Program type (bits 4 through 7) Precision bits: Integer Precision ❶ ❷ — 3RD HEX DIGIT unspecified — 0 matches real — 8 one word — 9 Real Precision ❶ — 4TH HEX DIGIT unspecified — 0 standard — 1 extended — 2
4	Effective program length, the terminal address in the program
5	Length of COMMON (in words)
6	Length of the program header (in words) minus 9
7	Zero
8	Length of the program, including the program header (in disk blocks)
9	FORTRAN indicator (bits 0 through 7), number of files defined (bits 8 through 15)
10 and 11	Name of entry point 1 (in name code)
12	Address of entry point 1 (absolute for type 1 programs, relative for all others)

❶ All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.

❷ Two words for standard precision, 3 for extended.

The format of words 13 through 54 of the program header varies according to the program type. For program types 1 and 2, the program header consists of words 1 through 12 only. For program types 3 and 4, the program header, in addition to words 1 through 12, includes:

Word	Contents
13 and 14	Name of entry point 2 (in name code)
15	Relative address of entry point 2
16 and 17	Name of entry point 3 (in name code)
18	Relative address of entry point 3
19 through 51	Name and relative addresses of entry points 4 through 14, as required, in the format shown above. The program header ends following the relative address of the last entry point defined; hence, it is of variable length.

For program types 5 and 6, the program header, in addition to words 1 through 12, contains the following information:

Word	Contents
13	ISS number plus 50
14	ISS number
15	Number of interrupt levels required ❶
16	Interrupt level number associated with the primary interrupt ❶
17	Interrupt level number associated with the secondary interrupt ❶

❶ The 1442 Card Read/Punch is the only device requiring more than one interrupt level.

For type 7 programs, the program header, in addition to words 1 through 12, contains the associated interrupt level number in word 13.

DDF format

Disk data format (DDF) is the format in which data files are stored on disk. DDF consists of 320 binary words per sector. Information such as headers, trailers, and indicator words is not included in DDF format.

DCI format

Disk core image (DCI) format is the format in which a core image program is stored on disk. A core image program consists of the core image header, the mainline program, all subroutines referenced in the mainline program or other subroutines (except the disk I/O subroutine), the transfer vector, and any LOCALs or SOCALs that are required. A layout of a stored DCI program is shown under "Construction of a Core Load" in Chapter 3. The contents of the core image header are:

Word: Symbol	Word: Relative address	Contents	
@XEQA	0	Execution address of the core load	
@CMON	1	Length of COMMON (in words)	
@DREQ	2	Disk I/O subroutine indicator – /FFFF for DISKZ, /0000 for DISK1, /0001 for DISKN	
@FILE	3	Number of files defined	
@HWCT	4	Length of the core image header (in words)	
@LSCT	5	Sector count of files in system WS	
@LDAD	6	Loading address of the core load	
@XCTL	7	Exit control address for DISK1/N	
@TVWC	8	Length of the transfer vector (in words)	
@WCNT	9	Length, in words, of the core load, core image header, and the transfer vector	
@XR3X	10	Setting for the index register 3 during execution of the core load	
@ITVX	11	Contents of word 8 during execution	ITV
	12	Contents of word 9 during execution	ITV
	13	Contents of word 10 during execution	ITV
	14	Contents of word 11 during execution	ITV
	15	Contents of word 12 during execution	ITV
	16	Contents of word 13 during execution	ITV
	17	Reserved	IBT for ILS04
	18 through 20	Interrupt entry to 1231 ISS	IBT for ILS04
	21	Interrupt entry to 1403 ISS	IBT for ILS04
	22	Interrupt entry to 2501 ISS	IBT for ILS04
	23	Interrupt entry to 1442 ISS	IBT for ILS04
	24	Interrupt entry to keyboard/ console printer ISS	IBT for ILS04
	25	Interrupt entry to 1134/1055 ISS	IBT for ILS04
@OVSW	26	Sector count of LOCALs/SOCALs	
@CORE	27	Core size of system on which core load built	
	28 and 29	Define file table checksum work area	
@HEND	29	Last word of core image header	

CARD FORMATS

card sequencing and ID field

In card formats, the file name and card sequence number are punched in columns 73 through 80. The file name is in columns 73 through 77, and 3-column sequence number in columns 78 through 80. Names of less than 5 characters use columns 73 through 76 and 4-column sequence number in columns 77 through 80. The only exception to this convention is that card decks punched by DUMPDATA E do not contain the ID field and sequence number.

CDS format

Card system format (CDS) is the format in which absolute and relocatable programs (mainlines and subroutines) are punched into cards. Each deck in card system format consists of (1) a header card, (2) data cards, and (3) an end-of-program card.

mainline header card

The mainline header card is the first card of every type 1 or 2 program in CDS format. This card contains:

Word	Contents	
1	Reserved	
2	Checksum	
3	Type code (first 8 bits): 0000 0001 absolute 0000 0010 relocatable	
	Precision bits:	
	Integer Precision ❶❷	3RD HEX DIGIT
	unspecified	0
	matches real	8
	one word	9
	Real Precision ❶	4TH HEX DIGIT
	unspecified	0
	standard	1
	extended	2
4	Reserved	
5	Length of COMMON, in words (FORTRAN mainline program only)	
6	0000 0000 0000 0011	
7	Length of the work area required, in words (FORTRAN only)	
8	Reserved	
9	Define file count	
10 and 11	Name	
12	Relative entry point	
13 through 54	Reserved	

❶ All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified, unless an EPR or SPR card is used.

❷ Two words for standard precision, three for extended.

subprogram header card

The subprogram header card is the first card of every type 3 or 4 program in card system format. This card contains:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0011 to be called by an LIBF statement only 0000 0100 to be called by a CALL statement only Precision bits: Integer Precision ❶ ❷ — 3RD HEX DIGIT unspecified — 0 matches real — 8 one word — 9 Real Precision ❶ — 4TH HEX DIGIT unspecified — 0 standard — 1 extended — 2
4 and 5	Reserved
6	Number of entry points times three
7 through 9	Reserved
10 and 11	Name of entry point 1 (in name code)
12	Relative address of entry point 1
13 through 51	Names and relative addresses of entry point 2 through 14, as required
52 through 54	Reserved

❶ All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.

❷ Two words for standard precision, three for extended.

ISS header card

The ISS header card is the first card of every type 5 or 6 program in CDS format, and contains:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0101 to be called by an LIBF statement only 0000 0110 to be called by a CALL statement only Precision bits: Integer Precision ❶ ❷ — 3RD HEX DIGIT unspecified — 0 matches real — 8 one word — 9 Real Precision ❶ — 4TH HEX DIGIT unspecified — 0 standard — 1 extended — 2
4 and 5	Reserved
6	Number of interrupt levels required plus 6
7 through 9	Reserved
10 and 11	Subroutine name (in name code)
12	Relative entry point address
13 and 14	Reserved for parameters used by the 1130 Card/Paper Tape System
15	Number of interrupt levels required ❸
16	Interrupt level number associated with the primary interrupt ❸
17	Interrupt level associated with the secondary interrupt level ❸
18 through 29	Reserved
30	One
31 through 54	Reserved

❶ All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.

❷ Two words for standard precision, three for extended.

❸ The 1442 Card Read Punch is the only device requiring more than one interrupt level.

ILS header card

The ILS header card is the first card of every type 7 program in CDS format, and contains:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0111 Reserved (last 8 bits)
4 and 5	Reserved
6	0000 0000 0000 0100
7 through 9	Reserved
10 through 12	Reserved
13	Interrupt level number
14 through 54	Reserved

format of data cards

In all types of programs, data cards contain the instructions and data that comprise the machine language program. The format of each data card is:

Word	Contents
1	The loading address of the first data word in the card. Succeeding words go into higher numbered core locations. The relocation factor must be added to this address to obtain the actual load address. For an absolute program the relocation factor is zero.
2	Checksum
3	Type code (first 8 bits): 0000 1010 Count of data words, excluding indicator data words, in these cards (last 8 bits)
4 through 9	Relocation indicator data words (2 bits for each following data word): 00 absolute 01 relocatable 10 LIBF (next two bits 00) 11 CALL (next two bits 00) 11 DSA (next two bits 01)
10 through 54	Data words 1 through 45

end-of-program card

The end-of-program card is the last card of all programs in CDS format, and contains:

Word	Contents
1	Effective length of the program. This number is always even and is assigned by the assembler, FORTRAN compiler, or RPG compiler.
2	Checksum
3	Type code (first 8 bits): 0000 1111 Last 8 bits: 0000 0000
4	Execution address (mainline program only)
5 through 54	Reserved

sector break cards

Sector break cards are binary cards used by the system loader to cause programs or phases of programs to start loading at the beginning of a sector. The monitor system uses type 1 header cards as sector break cards. The sector break cards are not checksummed. Columns 5 through 72 of the sector break cards may contain information identifying the program phase being loaded. The card sequence number appears in columns 73 through 80. Columns 5 through 80 are punched in IBM Card Code.

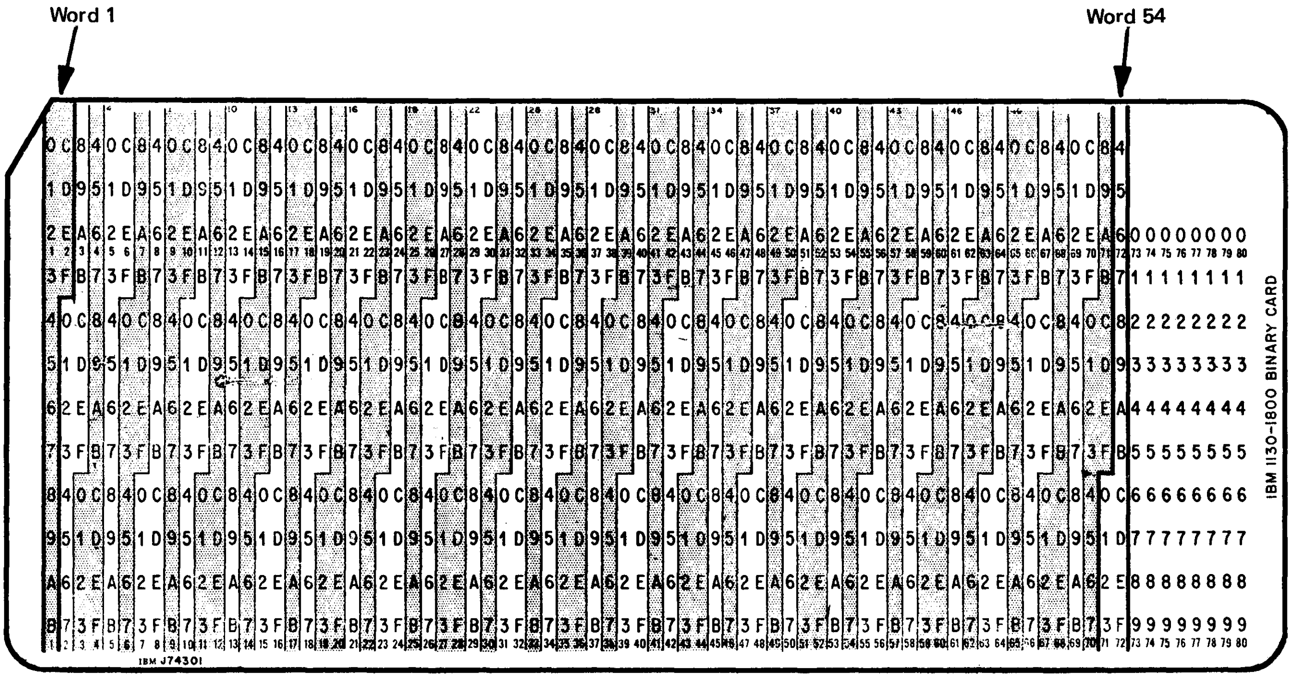
Type 1 cards are identified by a 1 punch in column 4 (binary word 3). A type 1 card indicates to the system loader that it should check word 11 of the first data card that follows. For the resident image, Cold Start Program, and phase 1 of the system loader, word 11 contains the absolute starting sector address. For all other monitor programs or phases, word 11 contains the phase ID. Recognition of a phase ID during initial load causes the system loader to load the program or phase starting at the next sequential sector. During a reload, the phase ID is matched with the ID in SLET and the phase is loaded to the sector address indicated in SLET.

On an initial load, phase 1 of DUP starts loading at sector 8.

A type 2 (relocatable starting sector address) sector break card is processed by the monitor system as a type 1 sector break card.

CDD format

Card data format (CDD) is the format in which data files are punched into cards. CDD format consists of 54 binary words per card. Each binary word occupies 1-1/3 columns. Information such as headers, trailers, and indicator words is not included in CDD format. CDD format is illustrated by the following:



CDC format

Card core image (CDC) format is the format in which core image programs are punched into cards. CDC format is identical to card data format (CDD), that is, one binary word occupies 1-1/3 columns and 54 binary words can be punched per card.

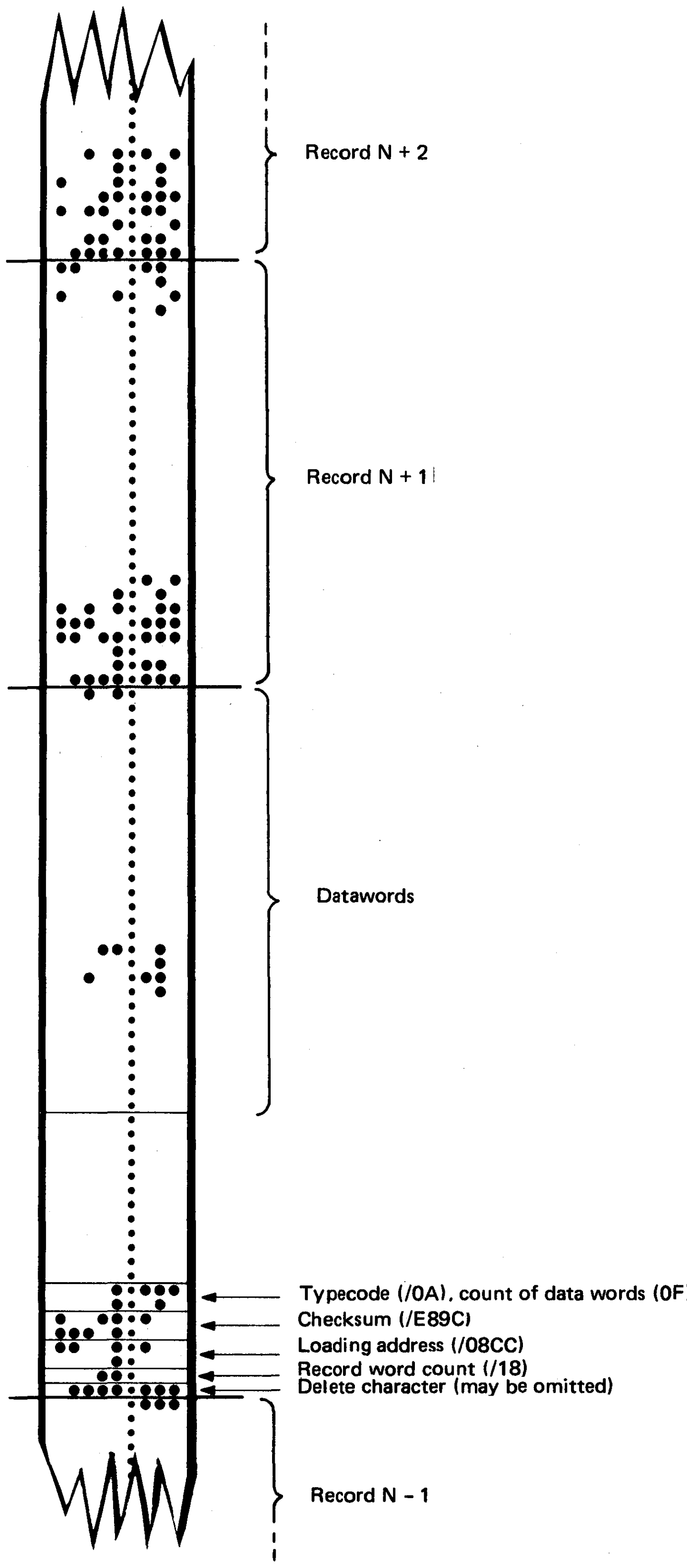
PAPER TAPE FORMATS

The paper tape formats—paper tape system format (PTS), paper tape data format (PTD), and paper tape core image format (PTC)—are analogous to the corresponding card formats (see preceding).

Two frames in paper tape (data or core image) format contain one binary word and are equivalent to 1-1/3 columns in card (data or core image) format. A data record in paper tape (data or core image) format differs from a data record in card (data or core image) format in that the record is preceded by any number (normally zero) of delete characters (/7F) and a frame containing the word count, one-half the number of frames in this data record. A data record in paper tape (data or core image) format contains a maximum of 108 frames (54 binary words) plus the 2 special frames.

Information that would appear in columns 73 through 80 in card format must not appear in paper tape format.

The following is an example of paper tape data (PTD) format:



PRINT FORMATS

PRD format

Print data format (PRD) is the format in which DUP prints a DSF program, core image program, or data file on a print device (1403, 1132, or console printer). The following are printouts of dumps of a DSF program and a DCI program:

DSF Program

*DUMP UA PR SAMPL

ADDR	***0	***1	***2	***3	***4	***5	***6	***7	***8	***9	***A	***B	***C	***D	***E	***F
0000	0CC0	0000	0100	0372	0000	0C03	CC0C	0015	0C00	2205	45D3	0218	0218	009B	8080	23A1
0C1C	717C	2C0C	0233	23A1	717C	C000	70FC	0020	6500	0004	C500	022E	D002	23A1	7170	2000
0020	2C00	CC00	23A1	7170	CC0C	7CFD	6600	0C05	6E00	CCCC	0298	705B	1000	0279	0267	0259
0030	0248	0C14	0000	8121	2121	2121	2121	2121	2120	983C	7021	0000	1020	5C34	215C	503C
0C40	3021	1C3C	5C1C	805C	C000	3C9C	205C	7498	8121	0010	8181	2074	3034	0000	9421	3474
CC5C	9C60	A421	5C34	7414	9C24	2120	C00C	7421	9C50	6030	9821	C221	000D	8160	341C	0000
0060	5C60	3021	5C34	7414	9C24	2120	7421	9C50	0CCC	6030	9821	C221	0011	8174	8070	1834
CC70	6021	000C	5010	216C	341C	5C60	3098	219C	5021	1834	C000	215C	503C	3034	3021	C221
0080	0C0F	8174	B070	C00C	1834	6021	5010	2150	B434	6010	5C50	9021	0020	9834	1C9C	5060
0C9C	9821	C221	23A1	717C	1000	2C02	0298	23A1	7170	0C00	70FD	C00E	D007	040C	0000	2255
0CA0	0298	D50C	C2A4	71FF	7089	7C11	CC05	029E	0004	0000	80A0	02A4	0096	0000	1000	0000
0CBC	0000	CCC0	CC00	C022	1890	A8FC	0000	D027	C0F9	801B	D01D	C01B	1890	A81A	D01F	0000
0CC0	C0F1	801D	9012	1890	A81A	DC17	801C	1803	C000	D015	8016	900A	1890	A813	D00E	800E
0CDC	80E1	CC0C	8C04	D009	65C0	0C06	7013	CC01	0C02	0007	C000	0140	0000	1000	0000	0000
0CEC	CCC0	CC00	CCC0	0000	C000	1C00	030C	031D	0331	0346	035A	036E	0008	C500	02D5	D400
0CFC	02E7	C5CC	C2CE	0225	5103	2C03	029E	2925	70D6	1100	029F	0000	0005	0859	0080	30D7
0100	C5C0	02F7	D002	23A1	717C	2C00	CC0C	8CC0	23A1	717C	0000	70FD	71FF	70E6	6038	1000
C11C	CCC0	02FE	030F	032C	C334	0349	035D	0011	819C	C000	509C	3C5C	2174	B070	1834	6021
012C	5C10	2198	CCC0	341C	9C5C	6098	21C2	2121	2121	2121	0011	C000	8174	B070	1834	6021
C130	5C10	2120	7430	3494	C000	2198	341C	9C50	6C98	21C2	2121	2121	2121	0000	0013	8174
C14C	BC7C	0323	005A	C00C	1834	6C21	5C1C	2154	6020	7034	2130	3C9C	0000	3C21	9834	1C9C
C15C	5C60	9821	C221	2121	2121	CC00	2121	0014	8174	B07C	1834	6021	5010	2154	0000	6020
0160	7034	2130	3C9C	3C21	1CA4	5C20	7430	0000	3460	9821	C221	2121	2121	2121	0013	8174
0170	0CC0	807C	1834	6021	5010	2160	341C	5C60	3098	000C	2154	3460	2198	341C	9C50	6021
018C	C221	2121	CC00	2121	2121	0C13	8174	B070	1834	6021	5010	0000	2120	7430	3494	3498
C19C	2154	346C	2198	341C	C000	9C50	6021	C221	2121	2121	2121	0372	0000	8160	341C	0000
01A0	5C60	3021	5C34	7414												

CART ID 0EC1 CB ADDR 2EAD CB CNT CC15

Disk block on sector. For Data Files, this position will always be 0 (Data Files must start on sector boundary).

Sector

Core Image Program (note that the actual starting address is **/01FA**)

*DUMP FX PR CISAM

ADDR	***0	***1	***2	***3	***4	***5	***6	***7	***8	***9	***A	***B	***C	***D	***E	***F
C1FC											0218	0000	FFFF	0000	001E	0000
02C0	C1FA	CC0C	CC1A	C508	7F7E	0091	0091	0CB3	0091	00C4	0091	0091	0091	0091	0091	0091
C21C	0C91	CC91	0376	0091	C000	8C00	CC0C	0000	4377	200C	0233	4377	0000	70FD	6500	0004
0220	C5C0	022E	D002	4377	2C00	CCC0	4377	CC00	7CFD	6600	0005	6E00	0298	705B	1000	0279
C230	0267	0259	0248	C014	8121	2121	2121	2121	2121	2120	983C	7021	1020	5C34	215C	503C
C240	3C21	1C3C	5C1C	805C	3C9C	2C50	7498	8121	C01C	8181	2074	3034	9421	3474	9C60	A421
C25C	5C34	7414	9C24	212C	7421	9C50	6C3C	9821	C221	000D	8160	341C	5060	3021	5C34	7414
0260	9C24	2120	7421	905C	6030	9821	C221	0011	8174	807C	1834	6021	5010	2160	341C	5060
C270	3C98	219C	5021	1834	215C	5C3C	3034	3021	C221	000F	8174	B070	1834	6021	5010	2150
C28C	B434	6C1C	5C50	9021	9834	1C9C	5060	9821	C221	4377	10C0	0298	4377	0000	70FD	C00E
029C	DC07	4374	0298	D50C	C2A4	71FF	7C89	7C11	CCC5	0658	0658	0000	0000	0000	80A0	0000
C2A0	CC00	000C	0000	C00C	1000	0C00	0C00	0C00	0000	C022	1890	A8FC	D027	C0F9	801B	D01D
C2B0	CC1B	189C	A81A	C01F	C0F1	801D	9012	1890	A81A	D017	8010	1803	D015	8016	900A	1890
C2CC	A813	C00E	8C0E	80E1	8004	DC09	650C	0CC6	7013	C001	0002	0007	0140	0000	1000	0000
C2DC	CCC0	0CC0	C000	0C0C	C000	1CC0	C30C	031D	0331	0346	035A	036E	C500	02D5	D400	02E7
C2E0	C5C0	02CE	4371	029E	436E	11C0	029F	0C00	0CC5	448C	7FFF	C500	02F7	D002	4377	2000
C2F0	CCC0	4377	C00C	70FC	71FF	7CE6	6C38	1CC0	02FE	030F	0320	0334	0349	035D	0011	819C
03CC	5C9C	3C5C	2174	B07C	1834	6021	5C1C	2198	341C	9C5C	6098	21C2	2121	2121	2121	0011
C31C	8174	B07C	1834	6021	5010	212C	7430	3494	2198	341C	9C50	6098	21C2	2121	2121	2121
C32C	CC13	8174	BC70	1834	6021	5010	2154	6020	7C34	2130	3C9C	3C21	9834	1C9C	5060	9821
C33C	C221	2121	2121	2121	C014	8174	807C	1834	6C21	501C	2154	602C	7034	2130	3C9C	3C21
C340	1CA4	5C2C	7430	346C	9821	C221	2121	2121	2121	0013	8174	B070	1834	6021	5010	2160
C350	341C	5060	3098	2154	3460	2198	341C	9C50	6C21	C221	2121	2121	2121	0013	8174	B070
C360	1834	6021	5C10	212C	7430	3494	3498	2154	3460	2198	341C	9C50	6021	C221	2121	2121
C37C	2121	C400	6914	658C	7FF5	7C03	CC0C	4C00	03D5	6A0F	280F	D83A	C100	180C	4C20	038D
C380	CC34	4818	7101	C832	7101	6906	650C	0C00	6600	C000	2000	4C00	0000	C027	4C20	038D
0390	C1C0	180C	906C	4C30	03AB	8045	D011	0820	1C05	4C28	03AD	1810	C05D	D058	D05E	D059
C3AC	C1C1	805C	D051	D057	C580	CC01	4C08	03AB	7C00	7012	701F	C00E	7003	4C02	0397	C008
C3BC	71FF	6C0C	0028	6128	70CF	CCC0	D235	C247	2C00	0F00	2001	0F01	D03A	D03E	08F9	1006
03CC	4C28	03AF	4C00	047C	C0FC	7401	0032	1000	0827	70B8	1001	C02B	D0E8	7401	0032	1000
C3DC	C480	03F4	D01D	C81E	70AC	08E4	C021	1C01	4C28	0423	4802	7001	7011	C01C	4C20	0462
C3EC	CC15	4C2C	0455	C0C4	4804	7C1A	740C	C3F7	7C05	181C	D0CA	74FF	0032	1000	4C80	0376

ADDR	***0	***1	***2	***3	***4	***5	***6	***7	***8	***9	***A	***B	***C	***D	***E	***F
C3FC	7CE6	0CCC	03F0	09CC	C000	0A00	CCCC	CC00	CC00	000C	0000	0000	0000	0000	0001	0002
C400	74FF	03F7	7001	70E5	C0F4	4C18	C40E	1810	D0F0	7401	03F4	C48C	03F4	7005	C0EF	D0E9
C41C	C480	C3F4	1008	D0DC	08A3	1C05	4C28	041E	089F	1005	4C28	041E	08D5	70D0	C400	03B8
C420	4400	0C8C	70F1	08CC	C480	03F4	1C0C	4C02	043F	4C3C	0443	4C2C	044A	614B	C480	03F4
C430	F5C0	0CCC	4C18	0436	71FF	7CF8	C50C	0C00	DCB7	7401	03F4	74FF	03F7	08B2	70D5	C02E
C440	D480	03F4	70A6	D086	C0B6	DCAE	C0B5	DCAF	C027	70C9	D0AB	C0B0	90AA	4C08	0453	74FF
C450	C3F4	7401	03F7	C019	70BE	CCA7	4C2C	045C	C0A5	D0A3	C015	70B7	1810	D098	D09E	D09A
C460	088F	7CBB	C09A	4C18	C458	CC90	4C2C	045C	C095	D08C	C004	70A7	0003	1100	4110	8100
C470	BCC0	1C00	1008	1888	6B04	8C03	8088	D001	C400	000C	80F1	4C80	0471	C007	40F2	D0B1
C480	CC05	4CEF	DC84	4C0C	03C4	436B	4368	6ADC	6936	6580	7FF2	6A35	282F	D850	C100	D003
C49C	7101	6931	6500	C001	C040	DC3E	181C	DC46	C101	4C18	04A7	E03F	4C20	04BA	C101	620C
C4A0	1240	9C38	4C20	04BA	6A37	CC30	9C35	8C36	74FF	04D4	701C	D032	4C01	04CE	C1FC	4C18
C4B0	04BC	9C27	4C18	04C4	9024	4C18	C4BC	9020	4C18	04BC	2001	7001	2000	C820	6500	0000
C4C0	66C0	CC00	4C00	C00C	9019	DC18	70F5	AC0F	4C20	04BA	108F	D012	7101	70CA	F00B	4C20
C4D0	04BA	C1FC	FC06	7CE4	C000	CC05	CC0A	0C14	0CA0	400C	8000	C00F	0000	DB05	7401	0037
04EC	6926	658C	7FEF	2824	C83D	DC40	C10C	D01B	7101	6921	6105	C03A	4C28	050C	C02F	D480
04F0	0503	CC34	A027	D031	A02B	1C90	4801	8020	802D	802A	D006	901D	4C10	0513	C026	D026
050C	CC1A	1800	D500	CC0C	71FF	7CEB	650C	0000	2C00	C818	4C00	0000	1810	9018	4808	C00D
051C	DC15	CC0A	70DC	8010	D0EC	7401	0525	7CE6	0C01	180A	1999	2000	4000	7FFF	80A0	A000
C52C	FFF6	D032	0833	C82E	18C0	0C0C	CC0C	1002	695D	6580	7FEC	6A58	10A0	D02F	C100	18D0
0530	1084	D027	1084	D026	1010	1C84	D024	1084	D023	C101	D00B	C102	D040	C103	D07E	C580
C540	CC05	D011	7106	6944	10A0	C400	CC0C	18D0	1081	D076	1010	7400	0559	7045	1087	D06E
C55C	6580	05BE	C5C0	C00C	7400	05C0	7007	1C08	7006	0000	0000	0000	0000	0000	E062	D05E
C560	C061	740C	055D	101C	D0F8	C058	7400	055C	7C10	740C	055D	7001	700A	D051	74FF	05BD
057C	7C18	C480	057D	E052	E84A	7C06	7C09	1808	E846	740C	055B	7028	D400	0000	7401	057D
0580	74FF	05BD	7C06	6600	C000	6500	CC0C	4C00	0C00	1010	7400	055A	7003	7400	055D	70B8
C59C	7401	0546	7CB1	1082	1005	DC2D	1010	1087	4C18	05A1	620F	1240	72F9	1000	6A1F	1010
C5AC	9C1D	1082	E820	70AB	18D0	D01F	1010	1083	100D	D019	1083	4C18	05B6	9016	D00F	1010
05B0	9C0D	D00C	6680	05BE	C00D	12C0	E80C	1806	1086	18DC	C00A	18D0	70BF	0000	0000	0000
05C0	0CC0	FF00	C001	000C	0C0D	0CC0	C0FF	CC20	4421	20E0	24E4	2121	3CFC	2121	2121	2121
C5DC	18D8	2121	0282	2121	1CDC	2121	00C0	2121	30F0	2121	DE04	2121	34F4	4109	FEE6	2121
C5EC	1CD0	2121	DAC2	2121	14D4	2121	C6E2	2121	21C4	21A0	21A4	2121	21BC	2121	2121	2121
C5F0	2198	2121	2121	2121	219C	2121	218C	2121	21B0	2121	2106	2121	21B4	2103	21BE	2121
060C	2190	2121	2146	2121	2194	2121	2186	2121	2184	2160	2164	2121	217C	2121	2121	2121
0610	2158	2121	2142	2121	215C	2121	2140	2121	2170	2105	2106	2121	2174	2181	21F6	2121
062C	2150	2111	21D2	2121	2154	2121	21F2	2121	2121	2121	2121	2121	2121	2121	2121	2121
0630	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121	2121
064C	2121	2121	2121	2121	2121	2121	2121	2121	0000	0000	7FE9	0000	0000	0000	8110	8090
0650	8050	4210	4110	409C	2110	2090	0110	0090	8820	8220	8060	4820	40A0	4060	2220	2120
0660	20A0	2060	0820	0420	0120	0060	842C	8120	4420	422C	4120	3000	2420	0220	00A0	2010
C670	2C20	2040	2080	210C	2200	2400	280C	4C10	4020	404C	4080	4100	4200	4400	4800	5000
0680	8010	802C	8040	808C	8100	8200	840C	8800	9C00	0010	0020	0040	0080	0100	0200	0400
0690	0800	1C00	2000	4000	8000	80A0	000C	0282	0000	0000	7FE6	217F	217F	217F	417F	217F
06A0	217F	057F	817F	117F	037F	217F	097F	217F	027F	DE7F	C67F	427F	D27F	F27F	067F	BE7F
06B0	467F	867F	827F	C07F	E60B	E27F	C06E	FE57	4062	D623	F62F	BC4C	8016	047F	C24A	A054
06CC	A413	9452	9051	B410	B04F	9C0E	980C	6020	641F	545E	505D	741C	705B	5C1A	5819	7C58
06D0	202C	246B	142A	1029	3468	3067	1C26	1825	3C64	E008	E407	D446	D045	F404	F043	DC02
06E0	D801	FC40	C449	8461	4415	DA6D	217F	C1C0	0000	4C00	0698	0000	4C00	0648	0000	4C00
06FC	0528	0000	4C00	04E0	0000	4C00	0488	0000	4C00	0372	0000	0000	0000	0000	0000	0000
0700	0000	05C8														

CART ID 0ED1 DB ADDR 1F00 DB CNT 0050

The address that precedes each printed line is the core address of word 1 on that line when a core image program is being printed. If a DSF program or data file is being printed, the address is the address of word 1 on that line relative to the start of the DSF program or data file. Each word printed is 4 hexadecimal characters long, and represents one binary word.

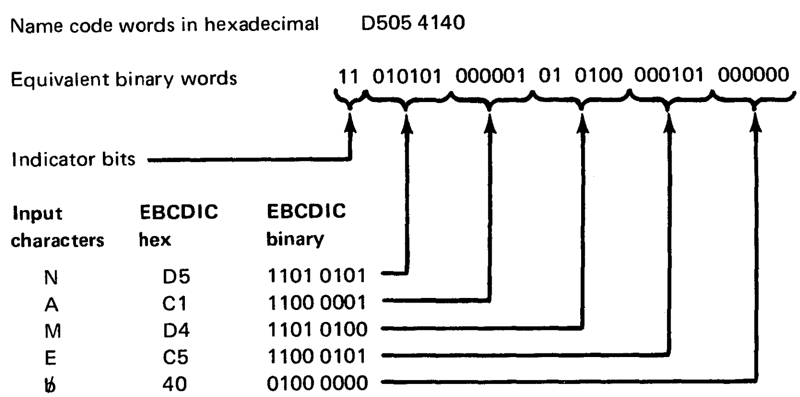
DATA FORMATS

NCF format

Name code format is the format in which names of subprograms, entry points, labels, etc., are stored into 2 binary words for use by monitor programs. The name consists of 5 characters, with the terminal characters possibly being blanks. Each EBCDIC character has the 2 leftmost bits dropped, and the remaining 6-bit blocks are packed to fill the following 30 bits of the 2 words. The 2 left bits of the 2-word name code representation are used for various purposes by different parts of the monitor system. For example, in the LET/FLET entry, these bits specify the format of the file (see Appendix D "LET/FLET").

The name-data words, used internally by the FORTRAN compiler, are similarly packed but the leftmost bit of each word is used as the indicator bit. This bit is set to zero if the word contains a constant; otherwise, it is set to one.

The following is an example of name code format:



Appendix J. Field Type Examples for DFCNV

The following is a description of each field type supported by the program. In each of these specification descriptions, the column and field length indicators may vary from 1 to 3 digits in length; all other numeric indicators must be one digit in length.

I-FIELD TYPE

This field type describes FORTRAN integer conversion; input is an integer field. The specification is:

m-Iw.t (P)

where

m is the column of the RPG record in which the converted field begins (1 through 640).
I identifies the field type.
w is the field length of the converted field (maximum of 14).
t is the number of positions to the right of the decimal point reserved in the RPG field (maximum of 9).
(P) is optional and is present only if the RPG field is to be packed.

Note. Since the FORTRAN integer field is regarded as a whole number with no decimal places, up to 5 positions to the left of the decimal should be reserved in the converted field to hold the largest possible integer value. Alignment is at the decimal point; if 5 positions are not reserved, high-order truncation occurs (see "DFCNV Messages and Error Messages" in Appendix A).

Example 1: The integer field /3A7E (14974 decimal) is converted using the field specification 15-I8.2 to the following RPG field.

Record word	8	9	10	11
Content	F0F1	F4F9	F7F4	F0F0

Example 2 (truncation): The integer field of Example 1 is converted using the field specification 15-I6.2 to the following RPG field.

Record word	8	9	10
Content	F4F9	F7F4	F0F0

Example 3 (packed format): The integer field of example 1 is converted using the field specification 15-I8.2(P) to the following RPG field. The number is converted as in Example 1. The zone portions of each character are then removed and the digit portions are packed 2 per byte. The sign is added as a trailing hexadecimal digit (F=positive; D=negative).

Record word	8	9	10
Content	0014	9740	0F40

Note. Since field length does not account for sign, incorrect alignment exists if packed mode is specified and field length is an even number. In order to align the data correctly, a leading zero is added to the field. This is true in all field types that accept packed mode conversion.

Example 4: The integer field /C582 (−14974 decimal) is converted using the field specification 15-I8.2 to the following RPG field.

Record word	8	9	10	11
Content	F0F1	F4F9	F7F4	F0D0

J-FIELD TYPE

This field type describes 2-word integer conversion; input is a 2-word integer. The specification is:

m-Jw.t (P)

where

m is the column of the RPG record in which the converted field begins (1 through 640).
J identifies the field type.
w is the field length of the converted field (maximum of 14).
t is the number of positions to the right of the decimal point reserved in the RPG field (maximum of 9).
(P) is optional and is present only if the RPG field is to be packed.

Note. Since a 2-word integer is regarded as a whole number with no decimal places, up to 10 positions to the left of the decimal point should be reserved in the converted field to hold the largest possible integer value. Alignment is at the decimal point; if 10 positions are not reserved, high-order truncation occurs (see "DFCNV Messages and Error Messages" in Appendix A). If a file contains 2-word integers, standard precision must be specified on the file description card. If extended precision is specified, any J-field type specification is invalid.

Example: The 2-word integer field /7FFF/FFFF is converted using the field specification 7-J13.(P) to the following RPG field.

Record word	4	5	6	7
Content	0021	4748	3647	0F40

R-FIELD TYPE

This field type describes FORTRAN real-variable conversion. The specification is:

m-Rw.t (P)

where

m is the column of the RPG record in which the converted field begins (1 through 640).
R identifies the field type.
w is the field length of the converted field (maximum of 14).
t is the number of positions to the right of the decimal point reserved in the RPG field (maximum of 9).
(P) is optional and is present only if the RPG field is to be packed.

Note. If the real number of the input field is too small to yield any significant digits in the RPG field, the RPG field is set to zeros. If the real number is too large to yield any significant digits in the RPG field, the RPG field is set to nines (see "DFCNV Messages and Error Messages" in Appendix A).

Example 1: The standard precision real field /BC00/0080 (–0.53125 decimal) is converted using the field specification 25-R7.5 (P) to the following RPG field.

Record word	13	14
Content	0053	125D

Example 2: The real field of Example 1 is converted using the field specification 25-R7.5 to the following RPG field.

Record word	13	14	15	16
Content	F0F0	F5F3	F1F2	D540

Example 3: The standard precision real field /7A12/0097 (eight million decimal) is converted using the field specification 39-R7.0 (P) to the following RPG field.

Record word	20	21
Content	8000	000F

Example 4: If the field specification in Example 3 were 39-R7.2 (P) then the resulting RPG field would be set to nines since the input field is too large to yield any significant digits in the RPG field.

Record word	20	21
Content	9999	999F

If column 33 of the file description card contained a W, a warning message would be printed when the preceding conversion took place.

Example 5: The extended precision real field /0047/6250/0000 (10^{-12} decimal) is converted using the field specification 17-R9.9 to the following RPG field.

Record word	9	10	11	12	13
Content	F0F0	F0F0	F0F0	F0F0	F040

The RPG field is set to zeros since the input field is too small to yield any significant digits in the RPG field. A number whose first significant digit is more than 9 decimal places to the right of the decimal point cannot be expressed in RPG. If column 33 of the file description card contained a W, a warning message would be printed when above conversion took place.

B-FIELD TYPE

This field type describes FORTRAN A-conversion for integer data and CSP A1 and A2 conversion. The specification is:

m-Bw.n

where

m is the column of the RPG record in which the converted field begins (1 through 640).
B identifies the field type.
w is the number of characters in the field (maximum of 255).
n is the number of characters in each unit of the input field (n=1 or 2).

Note. If CSP A1 or A2 format is converted, one word integers must be specified on the file description card; however, no diagnostic check is made for this condition.

Example: The CSP field POSITIVE appears on a disk record in A2 format as follows:

Record word	n	n+ 1	n+ 2	n+ 3
Content	E5C5	E3C9	E2C9	D7D6
	VE	TI	SI	PO

This field is converted using the field specification 21-B8.2 to the following RPG field.

Record word	11	12	13	14
Content	D7D6	E2C9	E3C9	E5C5
	PO	SI	TI	VE

C-FIELD TYPE

This field type describes FORTRAN A-conversion for real data. The specification is:

m-Cw.n

where

m is the column of the RPG record in which the converted field begins (1 through 640).
C identifies the field type.
w is the number of characters in the field (maximum of 255).
n is the number of characters in each unit (2 or 3 words) of the input field. For standard precision, n may range from 1 through 4; for extended precision, from 1 through 6.

Example: The FORTRAN field WASHINGTON, D. C. appears on a disk record in A4 format, extended precision, beginning at word 221 as follows:

Record word	210	211	212	213	214	215
Content	4BC3	4B40	4040	D6D5	6BC4	4040
	.C	.		ON	,D	

Record word	216	217	218	219	220	221
Content	C9D5	C7E3	4040	E6C1	E2C8	4040
	IN	GT		WA	SH	

D-FIELD TYPE

This field type describes CSP D1 conversion. The specification is

m-D1.j=1_2.K(P)

where

m is the column of the RPG record in which the converted field begins (1 through 640).

D identifies the field type.

l_1 is the length of the CSP field (maximum of 255).

j is the number of positions to the right of the decimal point in the CSP field.

l_2 is the length of the RPG field (maximum of 14).

k is the number of positions to the right of the decimal point in the RPG field (maximum of 9).

(P) is optional and is present only if the RPG field is to be packed.

Note. Alignment is at the decimal point. If, for example, $1_1 = 1_2$ and k>j, then k–j high order positions of the CSP field are truncated in the RPG field (see "DFCNV Messages and Error Messages" in Appendix A).

Example: The CSP D1 format field +00946.88 appears on a disk record beginning at word 78 as shown.

Record word	72	73	74	75	76	77	78
Content	0008	0008	0006	0004	0009	0000	0000

This field is converted using the field specification 35-C15.4 to the following RPG field.

Record word	18	19	20	21	22	23
Content	E6C1	E2C8	C9D5	C7E3	D6D5	6BC4
	WA	SH	IN	GT	ON	, D

Record word	24	25
Content	4BC3	4B40
	.C.	

This field is converted using the field specification 25-D7.2=6.3 to the following RPG field.

Record word	13	14	15
Content	F9F4	F6F8	F8F0

E-FIELD TYPE

This field describes CSP D4 conversion. The specification is:

$m{-}El_1.j{=}l_2.k(P)$

where

m is the column of the RPG record in which the converted field begins (1 through 640).
E identifies the field type.
l_1 is the length of the CSP field (maximum of 255).
j is the number of positions to the right of the decimal point in the CSP field.
l_2 is the length of the RPG field (maximum of 14).
k is the number of positions to the right of the decimal point in the RPG field (maximum of 9).
(P) is optional and is present only if the RPG field is to be packed.

Note. For E-field type conversion, alignment is also performed at the decimal point; high order truncation is possible (see "DFCNV Messages and Error Messages" in Appendix A).

Example: The CSP D4 format field −00946.88 appears on a disk record beginning at word 103 as follows:

Record word	101	102	103
Content	FFF7	68FF	0094

This field is converted using the field specification 25-E7.2=7.2 (P) to the following RPG field.

Record word	13	14
Content	0094	688D

F-FIELD TYPE

This field type describes CSP A3 conversion, and requires a 40 character translation table. The specification is:

m-Fw

where

m is the column of the RPG record in which the converted field begins (1 through 640).
F identifies the field type.
w is the number of characters in the field (not to exceed the input record size in characters).

Example: Suppose that a 40 character translation table with W as the 23rd position relative to the last position (card column 40) of the A3 table, H as the eighth relative position, and Y as the 25th relative position, is used to form the CSP field WHY in A3 format. This field is represented on a disk record by the integer /1419 that is derived using the following formula.

$$I=1600\,(N_1-20) + 40N_2 + N_3$$

where

N_1, N_2 *and* N_3 represent the positions relative to card column 40 in the table of the 1st, 2nd and 3rd characters, respectively.

/1419 is converted using the field specification 21-F4 to the following RPG field.

Record word	11	12
Content	E6C8	E840
	WH	Y

X-FIELD TYPE

This field type allows fields on the input record to be bypassed. The specification is:

Xw

where

X identifies the field type.
w is the number of words to be bypassed (not to exceed input record size).

Example: The field specification used to bypass an array of 10 real numbers when standard precision (each real number is 2 words in length) is specified as X20.

Appendix K. Decimal and Hexadecimal Disk Addresses

SECTOR ADDRESS BASE 10	SECTOR ADDRESS BASE 16	CYLINDER ADDRESS BASE 10	CYLINDER ADDRESS BASE 16
+00000	0000	+00000	0000
+00008	0008	+00001	0001
+00016	0010	+00002	0002
+00024	0018	+00003	0003
+00032	0020	+00004	0004
+00040	0028	+00005	0005
+00048	0030	+00006	0006
+00056	0038	+00007	0007
+00064	0040	+00008	0008
+00072	0048	+00009	0009
+00080	0050	+00010	000A
+00088	0058	+00011	000B
+00096	0060	+00012	000C
+00104	0068	+00013	000D
+00112	0070	+00014	000E
+00120	0078	+00015	000F
+00128	0080	+00016	0010
+00136	0088	+00017	0011
+00144	0090	+00018	0012
+00152	0098	+00019	0013
+00160	00A0	+00020	0014
+00168	00A8	+00021	0015
+00176	00B0	+00022	0016
+00184	00B8	+00023	0017
+00192	00C0	+00024	0018
+00200	00C8	+00025	0019
+00208	00D0	+00026	001A
+00216	00D8	+00027	001B
+00224	00E0	+00028	001C
+00232	00E8	+00029	001D
+00240	00F0	+00030	001E
+00248	00F8	+00031	001F
+00256	0100	+00032	0020
+00264	0108	+00033	0021
+00272	0110	+00034	0022
+00280	0118	+00035	0023
+00288	0120	+00036	0024
+00296	0128	+00037	0025
+00304	0130	+00038	0026
+00312	0138	+00039	0027
+00320	0140	+00040	0028
+00328	0148	+00041	0029
+00336	0150	+00042	002A
+00344	0158	+00043	002B
+00352	0160	+00044	002C
+00360	0168	+00045	002D
+00368	0170	+00046	002E
+00376	0178	+00047	002F
+00384	0180	+00048	0030
+00392	0188	+00049	0031
+00400	0190	+00050	0032
+00408	0198	+00051	0033
+00416	01A0	+00052	0034
+00424	01A8	+00053	0035
+00432	01B0	+00054	0036
+00440	01B8	+00055	0037
+00448	01C0	+00056	0038
+00456	01C8	+00057	0039
+00464	01D0	+00058	003A
+00472	01D8	+00059	003B
+00480	01E0	+00060	003C
+00488	01E8	+00061	003D
+00496	01F0	+00062	003E
+00504	01F8	+00063	003F
+00512	0200	+00064	0040
+00520	0208	+00065	0041
+00528	0210	+00066	0042

SECTOR ADDRESS BASE 10	SECTOR ADDRESS BASE 16	CYLINDER ADDRESS BASE 10	CYLINDER ADDRESS BASE 16
+00536	0218	+00067	0043
+00544	0220	+00068	0044
+00552	0228	+00069	0045
+00560	0230	+00070	0046
+00568	0238	+00071	0047
+00576	0240	+00072	0048
+00584	0248	+00073	0049
+00592	0250	+00074	004A
+00600	0258	+00075	004B
+00608	0260	+00076	004C
+00616	0268	+00077	004D
+00624	0270	+00078	004E
+00632	0278	+00079	004F
+00640	0280	+00080	0050
+00648	0288	+00081	0051
+00656	0290	+00082	0052
+00664	0298	+00083	0053
+00672	02A0	+00084	0054
+00680	02A8	+00085	0055
+00688	02B0	+00086	0056
+00696	02B8	+00087	0057
+00704	02C0	+00088	0058
+00712	02C8	+00089	0059
+00720	02D0	+00090	005A
+00728	02D8	+00091	005B
+00736	02E0	+00092	005C
+00744	02E8	+00093	005D
+00752	02F0	+00094	005E
+00760	02F8	+00095	005F
+00768	0300	+00096	0060
+00776	0308	+00097	0061
+00784	0310	+00098	0062
+00792	0318	+00099	0063
+00800	0320	+00100	0064
+00808	0328	+00101	0065
+00816	0330	+00102	0066
+00824	0338	+00103	0067
+00832	0340	+00104	0068
+00840	0348	+00105	0069
+00848	0350	+00106	006A
+00856	0358	+00107	006B
+00864	0360	+00108	006C
+00872	0368	+00109	006D
+00880	0370	+00110	006E
+00888	0378	+00111	006F
+00896	0380	+00112	0070
+00904	0388	+00113	0071
+00912	0390	+00114	0072
+00920	0398	+00115	0073
+00928	03A0	+00116	0074
+00936	03A8	+00117	0075
+00944	03B0	+00118	0076
+00952	03B8	+00119	0077
+00960	03C0	+00120	0078
+00968	03C8	+00121	0079
+00976	03D0	+00122	007A
+00984	03D8	+00123	007B
+00992	03E0	+00124	007C
+01000	03E8	+00125	007D
+01008	03F0	+00126	007E
+01016	03F8	+00127	007F
+01024	0400	+00128	0080
+01032	0408	+00129	0081
+01040	0410	+00130	0082
+01048	0418	+00131	0083
+01056	0420	+00132	0084
+01064	0428	+00133	0085

SECTOR ADDRESS BASE 10	SECTOR ADDRESS BASE 16	CYLINDER ADDRESS BASE 10	CYLINDER ADDRESS BASE 16
+01072	0430	+00134	0086
+01080	0438	+00135	0087
+01088	0440	+00136	0088
+01096	0448	+00137	0089
+01104	0450	+00138	008A
+01112	0458	+00139	008B
+01120	0460	+00140	008C
+01128	0468	+00141	008D
+01136	0470	+00142	008E
+01144	0478	+00143	008F
+01152	0480	+00144	0090
+01160	0488	+00145	0091
+01168	0490	+00146	0092
+01176	0498	+00147	0093
+01184	04A0	+00148	0094
+01192	04A8	+00149	0095
+01200	04B0	+00150	0096
+01208	04B8	+00151	0097
+01216	04C0	+00152	0098
+01224	04C8	+00153	0099
+01232	04D0	+00154	009A
+01240	04D8	+00155	009B
+01248	04E0	+00156	009C
+01256	04E8	+00157	009D
+01264	04F0	+00158	009E
01272	04F8	+00159	009F
01280	0500	+00160	00A0
+01288	0508	+00161	00A1
+01296	0510	+00162	00A2
+01304	0518	+00163	00A3
+01312	0520	+00164	00A4
+01320	0528	+00165	00A5
+01328	0530	+00166	00A6
+01336	0538	+00167	00A7
+01344	0540	+00168	00A8
+01352	0548	+00169	00A9
+01360	0550	+00170	00AA
+01368	0558	+00171	00AB
+01376	0560	+00172	00AC
+01384	0568	+00173	00AD
+01392	0570	+00174	00AE
+01400	0578	+00175	00AF
+01408	0580	+00176	00B0
+01416	0588	+00177	00B1
+01424	0590	+00178	00B2
+01432	0598	+00179	00B3
+01440	05A0	+00180	00B4
+01448	05A8	+00181	00B5
+01456	05B0	+00182	00B6
+01464	05B8	+00183	00B7
+01472	05C0	+00184	00B8
+01480	05C8	+00185	00B9
+01488	05D0	+00186	00BA
+01496	05D8	+00187	00BB
+01504	05E0	+00188	00BC
+01512	05E8	+00189	00BD
+01520	05F0	+00190	00BE
+01528	05F8	+00191	00BF
+01536	0600	+00192	00C0
+01544	0608	+00193	00C1
+01552	0610	+00194	00C2
+01560	0618	+00195	00C3
+01568	0620	+00196	00C4
+01576	0628	+00197	00C5
+01584	0630	+00198	00C6
+01592	0638	+00199	00C7

Appendix L. Disk Storage Unit Conversion Factors

No. of ↓ Per →	Word	Disk block	Sector	Track	Cylinder	Disk
Bits	16	320	5,112	20,480	40,960	8,192,000
Data words		20	320 ❶	1,280	2,560	512,000
Disk blocks			16	64	128	25,600
Sectors				4	8	1,600
Tracks					2	400
Cylinders						200

❶ These follow the first actual word of each sector, which is used for the address.

Ref no.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM card code Rows 12	11	0	9	8	7–1	IBM card code Hex	Graphics and control names		1132 Printer EBCDIC subset hex	PTTC/8 hex U-upper case L-lower case	Console printer hex	notes	1403 Printer hex
0	0000	0000	00	12		0	9	8	1	B030	NUL						
1		0001	01	12			9		1	9010							
2		0010	02	12			9		2	8810							
3		0011	03	12			9		3	8410							
4		0100	04	12			9		4	8210	PF	Punch Off					
5*		0101	05	12			9		5	8110	HT	Horiz.Tab		6D (U/L)	41	①	
6		0110	06	12			9		6	8090	LC	Lower Case		6E (U/L)			
7*		0111	07	12			9		7	8050	DEL	Delete		7F (U/L)			
8		1000	08	12			9	8		8030							
9		1001	09	12			9	8	1	9030							
10		1010	0A	12			9	8	2	8830							
11		1010	0B	12			9	8	3	8430							
12		1100	0C	12			9	8	4	8230							
13		1101	0D	12			9	8	5	8130							
14		1110	0E	12			9	8	6	80B0							
15		1111	0F	12			9	8	7	8070							
16	0001	0000	10	12	11		9	8	1	D030							
17		0001	11		11		9		1	5010							
18		0010	12		11		9		2	4810							
19		0011	13		11		9		3	4410							
20*		0100	14		11		9		4	4210	RES	Restore		4C (U/L)	05	②	
21*		0101	15		11		9		5	4110	NL	New Line		DD (U/L)	81	③	
22*		0110	16		11		9		6	4090	BS	Backspace		5E (U/L)	11		
23		0111	17		11		9		7	4050	IDL	Idle					
24		1000	18		11		9	8		4030							
25		1001	19		11		9	8	1	5030							
26		1010	1A		11		9	8	2	4830							
27		1011	1B		11		9	8	3	4430							
28		1100	1C		11		9	8	4	4230							
29		1101	1D		11		9	8	5	4130							
30		1110	1E		11		9	8	6	40B0							
31		1111	1F		11		9	8	7	4070							
32	0010	0000	20		11	0	9	8	1	7030							
33		0001	21			0	9		1	3010							
34		0010	22			0	9		2	2810							
35		0011	23			0	9		3	2410							
36		0100	24			0	9		4	2210	BYP	Bypass					
37*		0101	25			0	9		5	2110	LF	Line Feed		3D (U/L)	03		
38*		0110	26			0	9		6	2090	EOB	End of Block		3E (U/L)			
39		0111	27			0	9		7	2050	PRE	Prefix					
40		1000	28			0	9	8		2030							
41		1001	29			0	9	8	1	3030							
42		1010	2A			0	9	8	2	2830							
43		1011	2B			0	9	8	3	2430							
44		1100	2C			0	9	8	4	2230							
45		1101	2D			0	9	8	5	2130							
46		1110	2E			0	9	8	6	20B0							
47		1111	2F			0	9	8	7	2070							
48	0011	0000	30	12	11	0	9	8	1	F030							
49		0001	31				9		1	1010							
50		0010	32				9		2	0810							
51		0011	33				9		3	0410							
52		0100	34				9		4	0210	PN	Punch On					
53*		0101	35				9		5	0110	RS	Reader Stop		0D (U/L)	09	④	
54*		0110	36				9		6	0090	UC	Upper Case		0E (U/L)			
55		0111	37				9		7	0050	EOT	End of Trans.					
56		1000	38				9	8		0030							
57		1001	39				9	8	1	1030							
58		1010	3A				9	8	2	0830							
59		1011	3B				9	8	3	0430							
60		1100	3C				9	8	4	0230							
61		1101	3D				9	8	5	0130							
62		1110	3E				9	8	6	00B0							
63		1111	3F				9	8	7	0070							

Notes. Typewriter output

① Tabulate ③ Carrier return

② Shift to black ④ Shift to red

* Recognized by all conversion subroutines
Codes that are not asterisked are recognized by the SPEED subroutine.
The ZIPCO subroutine also recognizes these codes in conjunction with the appropriate code tables, notably EBHOL and HLEBC.

Ref no.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM card code Rows 12	11	0	9	8	7-1	IBM card code Hex	Graphics and control names	1132 Printer EBCDIC subset hex	PTTC/8 hex U-upper case L-lower case	Console printer hex	1403 Printer hex
64*	0100	0000	40	no punches						0000	blank	40 **	10 (U/L)	21	7F
65		0001	41	12		0	9		1	B010					
66		0010	42	12		0	9		2	A810					
67		0011	43	12		0	9		3	A410					
68		0100	44	12		0	9		4	A210					
69		0101	45	12		0	9		5	A110					
70		0110	46	12		0	9		6	A090					
71		0111	47	12		0	9		7	A050					
72		1000	48	12		0	9	8		A030					
73		1001	49	12				8	1	9020					
74*		1010	4A	12				8	2	8820	¢		20 (U)	02	
75*		1011	4B	12				8	3	8420	. (period)	4B	6B (L)	00	6E
76*		1100	4C	12				8	4	8220	<		02 (U)	DE	
77*		1101	4D	12				8	5	8120	(	4D	19 (U)	FE	57
78*		1110	4E	12				8	6	80A0	+	4E	70 (U)	DA	6D
79*		1111	4F	12				8	7	8060	| (logical OR)		3B (U)	C6	
80*	0101	0000	50	12						8000	&	50	70 (L)	44	15
81		0001	51	12	11		9		1	D010					
82		0010	52	12	11		9		2	C810					
83		0011	53	12	11		9		3	C410					
84		0100	54	12	11		9		4	C210					
85		0101	55	12	11		9		5	C110					
86		0110	56	12	11		9		6	C090					
87		0111	57	12	11		9		7	C050					
88		1000	58	12	11		9	8		C030					
89		1001	59		11			8	1	5020					
90*		1010	5A		11			8	2	4820	!		5B (U)	42	
91*		1011	5B		11			8	3	4420	$	5B	5B (L)	40	62
92*		1100	5C		11			8	4	4220	*	5C	08 (U)	D6	23
93*		1101	5D		11			8	5	4120	)	5D	1A (U)	F6	2F
94*		1110	5E		11			8	6	40A0	;		13 (U)	D2	
95*		1111	5F		11			8	7	4060	¬ (logical NOT)		6B (U)	F2	
96*	0110	0000	60		11					4000	- (dash)	60	40 (L)	84	61
97*		0001	61			0			1	3000	/	61	31 (L)	BC	4C
98		0010	62		11	0	9		2	6810					
99		0011	63		11	0	9		3	6410					
100		0100	64		11	0	9		4	6210					
101		0101	65		11	0	9		5	6110					
102		0110	66		11	0	9		6	6090					
103		0111	67		11	0	9		7	6050					
104		1000	68		11	0	9	8		6030					
105		1001	69			0		8	1	3020					
106		1010	6A	12	11					C000					
107*		1011	6B			0		8	3	2420	, (comma)	6B	3B (L)	80	16
108*		1100	6C			0		8	4	2220	%		15 (U)	06	
109*		1101	6D			0		8	5	2120	_ (underscore)		40 (U)	BE	
110*		1110	6E			0		8	6	20A0	>		07 (U)	46	
111*		1111	6F			0		8	7	2060	?		31 (U)	86	
112	0111	0000	70	12	11	0				E000					
113		0001	71	12	11	0	9		1	F010					
114		0010	72	12	11	0	9		2	E810					
115		0011	73	12	11	0	9		3	E410					
116		0100	74	12	11	0	9		4	E210					
117		0101	75	12	11	0	9		5	E110					
118		0110	76	12	11	0	9		6	E090					
119		0111	77	12	11	0	9		7	E050					
120		1000	78	12	11	0	9	8		E030					
121		1001	79					8	1	1020					
122*		1010	7A					8	2	0820	:		04 (U)	82	
123*		1011	7B					8	3	0420	#		0B (L)	C0	
124*		1100	7C					8	4	0220	@		20 (L)	04	
125*		1101	7D					8	5	0120	' (apostrophe)	7D	16 (U)	E6	0B
126*		1110	7E					8	6	00A0	=	7E	01 (U)	C2	4A
127*		1111	7F					8	7	0060	"		0B (U)	E2	

** Any code other than those defined for 1132 is interpreted by the PRNT1 subroutine as a blank.

Ref no.	EBCDIC Binary 0123	4567	EBCDIC Hex	IBM card code Rows 12	11	0	9	8	7–1	IBM card code Hex	Graphics and control names	1132 Printer EBCDIC subset hex	PTTC/8 hex U-upper case L-lower case	Console printer hex	1403 Printer hex
128	1000	0000	80	12		0		8	1	B020					
129		0001	81	12		0			1	B000	a				
130		0010	82	12		0			2	A800	b				
131		0011	83	12		0			3	A400	c				
132		0100	84	12		0			4	A200	d				
133		0101	85	12		0			5	A100	e				
134		0110	86	12		0			6	A080	f				
135		0111	87	12		0			7	A040	g				
136		1000	88	12		0		8		A020	h				
137		1001	89	12		0	9			A010	i				
138		1010	8A	12		0		8	2	A820					
139		1011	8B	12		0		8	3	A420					
140		1100	8C	12		0		8	4	A220					
141		1101	8D	12		0		8	5	A120					
142		1110	8E	12		0		8	6	A0A0					
143		1111	8F	12		0		8	7	A060					
144	1001	0000	90	12	11			8	1	D020					
145		0001	91	12	11				1	D000	j				
146		0010	92	12	11				2	C800	k				
147		0011	93	12	11				3	C400	l				
148		0100	94	12	11				4	C200	m				
149		0101	95	12	11				5	C100	n				
150		0110	96	12	11				6	C080	o				
151		0111	97	12	11				7	C040	p				
152		1000	98	12	11			8		C020	q				
153		1001	99	12	11		9			C010	r				
154		1010	9A	12	11			8	2	C820					
155		1011	9B	12	11			8	3	C420					
156		1100	9C	12	11			8	4	C220					
157		1101	9D	12	11			8	5	C120					
158		1110	9E	12	11			8	6	C0A0					
159		1111	9F	12	11			8	7	C060					
160	1010	0000	A0		11	0		8	1	7020					
161		0001	A1		11	0			1	7000					
162		0010	A2		11	0			2	6800	s				
163		0011	A3		11	0			3	6400	t				
164		0100	A4		11	0			4	6200	u				
165		0101	A5		11	0			5	6100	v				
166		0110	A6		11	0			6	6080	w				
167		0111	A7		11	0			7	6040	x				
168		1000	A8		11	0		8		6020	y				
169		1001	A9		11	0	9			6010	z				
170		1010	AA		11	0		8	2	6820					
171		1011	AB		11	0		8	3	6420					
172		1100	AC		11	0		8	4	6220					
173		1101	AD		11	0		8	5	6120					
174		1110	AE		11	0		8	6	60A0					
175		1111	AF		11	0		8	7	6060					
176	1011	0000	B0	12	11	0		8	1	F020					
177		0001	B1	12	11	0			1	F000					
178		0010	B2	12	11	0			2	E800					
179		0011	B3	12	11	0			3	E400					
180		0100	B4	12	11	0			4	E200					
181		0101	B5	12	11	0			5	E100					
182		0110	B6	12	11	0			6	E080					
183		0111	B7	12	11	0			7	E040					
184		1000	B8	12	11	0		8		E020					
185		1001	B9	12	11	0	9			E010					
186		1010	BA	12	11	0		8	2	E820					
187		1011	BB	12	11	0		8	3	E420					
188		1100	BC	12	11	0		8	4	E220					
189		1101	BD	12	11	0		8	5	E120					
190		1110	BE	12	11	0		8	6	E0A0					
191		1111	BF	12	11	0		8	7	E060					

Ref no.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM card code Rows 12	11	0	9	8	7-1	IBM card code Hex	Graphics and control names	1132 Printer EBCDIC subset hex	PTTC/8 hex U-uppercase L-lowercase	Console printer hex	1403 Printer hex
192	1100	0000	C0	12		0				A000	(+ zero)				
193*		0001	C1	12					1	9000	A	C1	61 (U)	3C or 3E	64
194*		0010	C2	12					2	8800	B	C2	62 (U)	18 or 1A	25
195*		0011	C3	12					3	8400	C	C3	73 (U)	1C or 1E	26
196*		0100	C4	12					4	8200	D	C4	64 (U)	30 or 32	67
197*		0101	C5	12					5	8100	E	C5	75 (U)	34 or 36	68
198*		0110	C6	12					6	8080	F	C6	76 (U)	10 or 12	29
199*		0111	C7	12					7	8040	G	C7	67 (U)	14 or 16	2A
200*		1000	C8	12				8		8020	H	C8	68 (U)	24 or 26	6B
201*		1001	C9	12			9			8010	I	C9	79 (U)	20 or 22	2C
202		1010	CA	12		0	9	8	2	A830					
203		1011	CB	12		0	9	8	3	A430					
204		1100	CC	12		0	9	8	4	A230					
205		1101	CD	12		0	9	8	5	A130					
206		1110	CE	12		0	9	8	6	A0B0					
207		1111	CF	12		0	9	8	7	A070					
208	1101	0000	D0		11	0				6000	(- zero)				
209*		0001	D1		11				1	5000	J	D1	51 (U)	7C or 7 E	58
210*		0010	D2		11				2	4800	K	D2	52 (U)	58 or 5A	19
211*		0011	D3		11				3	4400	L	D3	43 (U)	5C or 5E	1A
212*		0100	D4		11				4	4200	M	D4	54 (U)	70 or 72	5B
213*		0101	D5		11				5	4100	N	D5	45 (U)	74 or 76	1C
214*		0110	D6		11				6	4080	O	D6	46 (U)	50 or 52	5D
215*		0111	D7		11				7	4040	P	D7	57 (U)	54 or 56	5E
216*		1000	D8		11			8		4020	Q	D8	58 (U)	64 or 66	1F
217*		1001	D9		11		9			4010	R	D9	49 (U)	60 or 62	20
218		1010	DA	12	11		9	8	2	C830					
219		1011	DB	12	11		9	8	3	C430					
220		1100	DC	12	11		9	8	4	C230					
221		1101	DD	12	11		9	8	5	C130					
222		1110	DE	12	11		9	8	6	C0B0					
223		1111	DF	12	11		9	8	7	C070					
224	1110	0000	E0			0		8	2	2820					
225		0001	E1		11	0	9		1	7010					
226*		0010	E2			0			2	2800	S	E2	32 (U)	98 or 9A	0D
227*		0011	E3			0			3	2400	T	E3	23 (U)	9C or 9E	0E
228*		0100	E4			0			4	2200	U	E4	34 (U)	B0 or B2	4F
229*		0101	E5			0			5	2100	V	E5	25 (U)	B4 or B6	10
230*		0110	E6			0			6	2080	W	E6	26 (U)	90 or 92	51
231*		0111	E7			0			7	2040	X	E7	37 (U)	94 or 96	52
232*		1000	E8			0		8		2020	Y	E8	38 (U)	A4 or A6	13
233*		1001	E9			0	9			2010	Z	E9	29 (U)	A0 or A2	54
234		1010	EA		11	0	9	8	2	6830					
235		1011	EB		11	0	9	8	3	6430					
236		1100	EC		11	0	9	8	4	6230					
237		1101	ED		11	0	9	8	5	6130					
238		1110	EE		11	0	9	8	6	60B0					
239		1111	EF		11	0	9	8	7	6070					
240*	1111	0000	F0			0				2000	0	F0	1A (L)	C4	49
241*		0001	F1						1	1000	1	F1	01 (L)	FC	40
242*		0010	F2						2	0800	2	F2	02 (L)	D8	01
243*		0011	F3						3	0400	3	F3	13 (L)	DC	02
244*		0100	F4						4	0200	4	F4	04 (L)	F0	43
245*		0101	F5						5	0100	5	F5	15 (L)	F4	04
246*		0110	F6						6	0080	6	F6	16 (L)	D0	45
247*		0111	F7						7	0040	7	F7	07 (L)	D4	46
248*		1000	F8					8		0020	8	F8	08 (L)	E4	07
249*		1001	F9				9			0010	9	F9	19 (L)	E0	08
250		1010	FA	12	11	0	9	8	2	E830					
251		1011	FB	12	11	0	9	8	3	E430					
252		1100	FC	12	11	0	9	8	4	E230					
253		1101	FD	12	11	0	9	8	5	E130					
254		1110	FE	12	11	0	9	8	6	E0B0					
255		1111	FF	12	11	0	9	8	7	E070					

$$$$$ data files, initializing for use with FORTRAN unformatted I/O 6-27
$DUMP entry (skeleton supervisor) use by CIL 3-13
$EXIT entry (skeleton supervisor) use by CIL 3-13
$LINK entry (skeleton supervisor) use by CIL 3-14
$PRET address, indicates preoperative error 3-2
$PST1 address, indicates postoperative error 3-2
$PST2 address, indicates postoperative error 3-2
$PST3 address, indicates postoperative error 3-2
$PST4 address, indicates postoperative error 3-2
** header information FORTRAN control record
  format 5-69
  printing a header on each page 5-69
*ARITHMETIC TRACE (*see also* ARITHMETIC TRACE FORTRAN control record) 5-70
*COMMON (*see also* COMMON assembler control record) 5-63
*DEFINE (*see also* DEFINE DUP control record) 5-45
*DELETE (*see also* DELETE DUP control record) 5-44
*DFILE (*see also* DFILE DUP control record) 5-48
*DUMP (*see also* DUMP DUP control record) 5-22
*DUMPDATA (*see also* DUMPDATA DUP control record) 5-24
*DUMPDATA E (*see also* DUMPDATA E DUP control record) 5-26
*DUMPFLET (*see also* DUMPFLET DUP control record) 5-29
*DUMPLET (*see also* DUMPLET DUP control record) 5-28
*DWADR (*see also* DWADR DUP control record) 5-47
*END (*see also* END MODSF patch control record) 4-19
*EQUAT (*see also* EQUAT supervisor control record) 5-17
*EXTENDED PRECISION (*see also* EXTENDED PRECISION FORTRAN control record) 5-68
*FILES (*see also* FILES supervisor control record) 5-15
*G2250 (*see also* G2250 supervisor control record) 5-16
*IOCS (*see also* IOCS FORTRAN control record) 5-65
*LEVEL (*see also* LEVEL assembler control record) 5-61
*LIST (*see also* LIST assembler control record) 5-53
*LIST ALL (*see also* LIST ALL FORTRAN control record) 5-67
*LIST DECK (*see also* LIST DECK assembler control record) 5-57
*LIST DECK E (*see also* LIST DECK E assembler control record) 5-59
*LIST SOURCE PROGRAM (*see also* LIST SOURCE PROGRAM FORTRAN control record) 5-66
*LIST SUBPROGRAM NAMES (*see also* LIST SUBPROGRAM NAMES FORTRAN control record) 5-66
*LIST SYMBOL TABLE (*see also* LIST SYMBOL TABLE FORTRAN control record) 5-67
*LOCAL (*see also* LOCAL supervisor control record) 5-13
*MACLIB (*see also* MACLIB assembler control record) 5-63
*MACRO UPDATE (*see also* MACRO UPDATE DUP control record) 5-49
*MON (*see also* MON MODIF patch control record) 4-9
*NAME (*see also* NAME FORTRAN control record) 5-69
*NOCAL (*see also* NOCAL supervisor control record) 5-14
*ONE WORD INTEGERS (*see also* ONE WORD INTEGERS FORTRAN control record) 5-68
*ORIGIN (*see also* ORIGIN FORTRAN control record) 5-71
*OVERFLOW SECTORS (*see also* OVERFLOW SECTORS assembler control record) 5-61
*PRINT SYMBOL TABLE (*see also* PRINT SYMBOL TABLE assembler control record) 5-59
*PRO (*see also* PRO MODSF control record) 4-15
*PUNCH SYMBOL TABLE (*see also* PUNCH SYMBOL TABLE assembler control record) 5-59
*SAVE SYMBOL TABLE (*see also* SAVE SYMBOL TABLE assembler control record) 5-60
*STORE (*see also* STORE DUP control record) 5-30
*STORECI (*see also* STORECI DUP control record) 5-38
*STOREDATA (*see also* STOREDATA DUP control record) 5-33
*STOREDATACI (*see also* STOREDATACI DUP control record) 5-37
*STOREDATAE (*see also* STOREDATAE DUP control record) 5-34
*STOREMOD (*see also* STOREMOD DUP control record) 5-42
*SUB (*see also* SUB MODIF patch control record) 4-12
*SYSTEM SYMBOL TABLE (*see also* SYSTEM SYMBOL TABLE assembler control record) 5-60
*TRANSFER TRACE (*see also* TRANSFER TRACE FORTRAN control record) 5-70
*TWO PASS MODE (*see also* TWO PASS MODE assembler control record) 5-52
*XREF (*see also* XREF assembler control record) 5-56
// * (comments) monitor control record (*see also* comments monitor control record) 5-9
// ASM (*see also* ASM monitor control record) 5-5
// CEND (*see also* CEND monitor control record) 5-11
// COBOL (*see also* COBOL monitor control record) 5-6
// CPRNT (*see also* CPRNT monitor control record) 5-11
// DEND (*see also* DEND MODIF patch control record) 4-13
// DUP (*see also* DUP monitor control record) 5-6
// EJECT (*see also* EJECT monitor control record) 5-11
// FOR (*see also* FOR monitor control record) 5-6
// JOB (*see also* JOB monitor control record) 5-2
// PAUS (*see also* PAUS monitor control record) 5-10
// records read during FORTRAN program execution 6-51
// RPG (*see also* RPG monitor control record) 5-6
// TEND (*see also* TEND monitor control record) 5-10
// TYP (*see also* TYP monitor control record) 5-10
// XEQ (*see also* XEQ monitor control record) 5-7
@DCOM (*see* DCOM)
@HDNG, on a system cartridge 2-6
@IDAD (*see* IDAD)
@RIAD, on a system cartridge 2-6
@RTBL, on a system cartridge 2-6

A-conversion, FORTRAN 6-50
absolute address 3-10
  An address that indicates the exact storage location where data is found or stored.
absolute program 3-10
  A DSF program to which you assign an origin so that the program can be executed from that core location only.
absolute starting address, defining with *ORIGIN FORTRAN control record 5-71
acronyms used in DUP operations 5-20
adding system library subroutines (MODIF) 4-12
address formats used in this publication
  hexadecimal 1-2
  symbolic 1-2
addresses, appendix K, decimal and hexadecimal disk K-1
ADRWS disk maintenance program 4-8

allocation addresses, locating FORTRAN 6-18
altering LET and FLET with DUP control records 5-20
altering the contents of a core location 7-14
analyzing disk cartridges, operating procedure 9-36
appendixes
  A, monitor system operational and error messages A-1
  B, monitor system error wait codes B-1
  C, monitor system library listing C-1
  D, LET/FLET D-1
  E, system location equivalence table (SLET) E-1
  F, core dump F-1
  G, resident monitor G-1
  H, monitor system sample programs H-1
  I, formats I-1
  J, field type examples for DFCNV J-1
  K, decimal and hexadecimal disk addresses K-1
  L, disk storage unit conversion factors L-1
  M, character code set M-1
ARITHMETIC TRACE FORTRAN control record
  format 5-70
  tracing variable values during execution 5-70
arithmetic tracing, how to stop 5-70
ASM monitor control record
  format 5-5
  general function 5-5
assembler
  CALL TSTOP 5-70
  CALL TSTRT 5-70
  deleting with the *DEFINE DUP control record 5-46
  description 3-5
  error codes A-2
  error messages A-2
  FILE statement 3-10
  listing of error codes A-3
  listing of error messages A-5
  loading 8-8
  monitor system program 3-5
assembler control records 5-50
  *COMMON 5-63
  *LEVEL 5-61
  *LIST 5-53
  *LIST DECK 5-57
  *LIST DECK E 5-59
  *MACLIB 5-63
  *OVERFLOW SECTORS 5-61
  *PRINT SYMBOL TABLE 5-59
  *PUNCH SYMBOL TABLE 5-59
  *SAVE SYMBOL TABLE 5-60
  *SYSTEM SYMBOL TABLE 5-60
  *TWO PASS MODE 5-52
  *XREF 5-56
  coding for keyboard and paper tape input 5-51
  general functions 5-50
  how to code 5-51
  where placed in the input stream 5-50
assembler core load
    A core load that is built from a mainline written in assembler language.
assembler core map, reading an 6-14
assembler disk file organization and processing 6-27
assembler error codes
  description of A-2
  assembler error codes listing of A-3
assembler error messages
  description of A-2
  listing of A-5
assembler INT REQ service subroutine 6-45
assembler language programmers, tips for 6-35
assembler listing
  printing with a cross reference symbol table 5-57
  printing with statement numbers 5-57
  reading an 5-54 and 55
assembler mnemonics, grouping of 6-35
assembler program 5-53
  assigning a core load origin 3-10
  calling sequence for SYSUP 6-20
  double buffering 6-35
  grouping of mnemonics 6-35
  using index register 3 6-35
  using the LOCAL-calls-a-LOCAL option 6-10
  using the 1403 conversion subroutines 6-37
assembler sample program H-7
assembler two pass mode, how to use 5-52
assignment of the origin of a core load, core load construction 3-9
auxiliary supervisor
  disk resident supervisor program 3-3
  error messages A-58

B-field type, DFCNV J-4
backspace key (←)
  use of 7-12
  when TYPEZ is used 7-13
  when TYPE0 is used 7-12
binary patch data record format 4-12
binary patch data records, MODIF 4-12
bits, number is a word 2-3

C-field type, DFCNV J-4
calculating
  ISAM file parameters (sample program) H-17
  ISAM file sizes 6-29
  sequentially organized file sizes 6-29
CALL, restriction on number in a core load 3-13
call system print subroutine 4-4
CALL subroutine
    A subroutine that must be referenced with a CALL statement. The type codes for subroutines in this category are 4 and 6.
CALL TSTOP statement, assembler 5-70
CALL TSTRT statement, assembler 5-70
CALL TV 3-13
    The transfer vector through which CALL subroutines are entered during execution.
calling the macro update program 5-49
CALPR, system library utility subroutine 4-4
card core image loader wait code B-9
card formats I-6
  card core image format (CDC) I-11
    The format in which a program stored in disk core image format is dumped to cards.
  card data format (CDD) I-11
    The format in which a data file is dumped to cards.
  card system format (CDS) I-6
    The format in which absolute and relocatable programs are punched into cards. In this format, columns 73 through 80 contain the card ID and sequence number.
card input at the RJE work station 10-2
Card Punch, readying the 1442 Model 5 7-5
card read errors, 1442 B-6
Card Read Punch, readying the 1442 Model 6 and 7 7-5
card reader, entering jobs from the 7-12
Card Reader, readying the 2501 7-6
card sequencing and ID field I-6

card subroutine errors
1442 B-5
2501 B-8
card system cold start procedure 7-11
card system initial load
materials needed for 8-15
operating procedure 8-15
organization of cards for 8-15
card system preload
materials needed for 8-24
operating procedure 8-24
card system reload
materials needed for 8-25
operating procedure 8-26
organization of cards for 8-20
cartridge (see disk)
cartridge assignments, specifying logical 5-4
cartridge copy code (COPY), in sector @IDAD 2-5
cartridge ID (CIDN), in sector @IDAD 2-5
cartridge IDs
changing 4-7
printing 4-5
cartridges
restoring destroyed 6-5
summary of the contents of disk 2-14
CDC program card format I-11
CDD data file card format I-11
CDS data card format I-9
CDS program
end-of-program card I-10
format I-6
ILS header card I-9
ISS header card I-8
mainline header card I-6
subprogram header card I-7
CEND monitor control record
format 5-11
reassigns principal printer 5-11
Central Processing Unit, readying the 1131 7-2
changed LOGON affect on RJE input 10-3
changing cartridge IDs 4-7
changing cartridges during a job, how to use SYSUP when 6-20
changing the size of the fixed area, *DEFINE DUP control record 5-45
character code set, appendix M M-1
checksum 4-12

The two's complement of the logical sum of the record count and the data words in the record. Before the monitor system computes a checksum when data word 2 contains a value, data word 2 is saved and changed to zero. The logical sum is obtained by arithmetically summing the record number and the contents of each of the 54 data words in the record. Each time a carry occurs out of the high-order position, one is added before the addition of the next data word. The two's complement of this logical sum is the checksum.

The term record number (count) should not be confused with the sequence number that appears in columns 73 through 80. A card is a record. The first record (a type 1 or 2 header card) is record one (not zero). The beginning of each program or program phase starts a new record count.

CIB (*see* core image buffer)
CIL (*see* core image loader)
CLB (*see* core load builder)
COBOL monitor control record
format 5-6
general function 5-6
codes
card core image loader wait B-9
FORTRAN codes A-7
FORTRAN I/O wait B-10
codes (continued)
monitor system error wait B-1
PTUTL error wait B-9
RPG object program wait B-12 and 13
coding an INT REQ service subroutine, rules for 6-45
cold start
error waits B-1
in sector @IDAD 2-5
messages printed during 7-10
procedure 7-10
procedure for card system 7-11
procedure for paper tape system 7-11
to stop the DCIP operating procedures 9-10
when changing cartridges on a one drive system 2-3
cold start card 8-24

The card that contains the coding necessary for initial program loading (IPL), that is, calling the cold start program.

cold start procedure 7-10
card system 7-11
paper tape system 7-11
cold start program 2-4

The disk resident program that initializes the monitor system by reading the resident monitor into core from the disk.

cold start program error waits B-1
cold start program in sector @IDAD 2-5
COMMA (*see* core communications area)
comment 5-9

The text contained on a monitor control record with an asterisk in column 4, an assembler language source record with an asterisk in column 21, a FORTRAN source record with a C in column 1, or an RPG specification with an asterisk in column 7.

comments monitor control record, // * 5-9
format 5-9
general function 5-9
COMMON, defining the length of (*COMMON assembler control record) 5-63
COMMON, low, how processed by core image loader 3-14
COMMON assembler control record
defining the length of common 5-63
format 5-63
communication considerations, RJE 10-1
comparing disk cartridges, operating procedure (DCIP) 9-40
compilation error messages, FORTRAN A-7
compilation messages, FORTRAN A-7
compiler
FORTRAN 3-6
RPG 3-6
compiler error notes
listing of RPG A-40
RPG A-38
compiler messages, RPG A-38
compilers
deleting the 5-46
loading the 8-8
components of disk storage 2-2
computing
file lable for ISAM files 6-30
file sizes for DFCNV 4-23
index sectors for ISAM files 6-30
overflow sectors for ISAM files 6-30
prime data sectors of ISAM files 6-30
number of sectors to be reserved by *DFILE 6-9
configuration, minimum system iii
console entry switches, RJE 10-12
console keyboard, entering jobs from the 7-12
console keyboard procedures, RJE 10-10
console operator keys, functions during monitor system control 7-13
console printer
FORTRAN program control of 6-50
subroutine errors B-8

console printer core dump stand-alone utility program
format 9-1
materials needed to use the 9-1
operating procedure 9-2
console printer subroutine errors B-8
construction of a core load 3-7
construction of the core image header 3-9
contents of an ISAM file, assembler and RPG disk file organization and processing 6-31
contents of CLB during core load construction 3-7
contents of disk cartridges, summary of the 2-14
contents of ISAM file
index 6-32
labels 6-31
overflow area 6-33
prime data area 6-33
contents of the CLB during core load construction 3-8
contents of working storage during core load construction 3-8
continuing RJE output 10-7
control character types for RJE user-exit data 10-8
control record
*One of the records (card or paper tape) that directs the activities of the monitor system. For example, the // DUP monitor control record directs the monitor to initialize DUP; the *DUMPLET DUP control record directs DUP to initialize the DUMPLET program; the *EXTENDED PRECISION FORTRAN control record directs the FORTRAN compiler to allot 3 words instead of 2 for the storage of variables.*
control record analyzer monitor 3-3
control record analyzer supervisor 3-3
control records 5-1
** header information, FORTRAN 5-69
*ARITHMETIC TRACE, FORTRAN 5-70
*COMMON, assembler 5-63
*DEFINE, DUP 5-45
*DELETE, DUP 5-44
*DFILE, DUP 5-48
*DUMP, DUP 5-22
*DUMPDATA, DUP 5-24
*DUMPDATA E, DUP 5-26
*DUMPFLET, DUP 5-29
*DUMPLET, DUP 5-28
*DWADR, DUP 5-47
*END patch, MODSF 4-19
*EQUAT, supervisor 5-17
*EXTENDED PRECISION, FORTRAN 5-67
*FILES, supervisor 5-15
*G2250, supervisor 5-16
*IOCS, FORTRAN 5-65
*LEVEL, assembler 5-61
*LIST ALL, FORTRAN 5-67
*LIST, assembler 5-53
*LIST DECK, assembler 5-57
*LIST DECK E, assembler 5-59
*LIST SOURCE PROGRAM, FORTRAN 5-66
*LIST SUBPROGRAM NAMES, FORTRAN 5-66
*LIST SYMBOL TABLE, FORTRAN 5-67
*LOCAL, supervisor 5-13
*MACLIB, assembler 5-63
*MACRO UPDATE, DUP 5-49
*MON patch, MODIF 4-9
*NAME, FORTRAN 5-69
*NOCAL, supervisor 5-14
*ONE WORD INTEGERS, FORTRAN 5-68
*ORIGIN, FORTRAN 5-71
*OVERFLOW SECTORS, assembler 5-61
*PRINT SYMBOL TABLE, assembler 5-59
control record (continued)
*PRO patch, MODSF 4-15
*PUNCH SYMBOL TABLE, assembler 5-59
*SAVE SYMBOL TABLE, assembler 5-60
*STORE, DUP 5-30
*STORECI, DUP 5-38
*STOREDATA, DUP 5-33
*STOREDATACI, DUP 5-37
*STOREDATAE, DUP 5-34
*STOREMOD, DUP 5-42
*SUB patch, MODIF 4-12
*SYSTEM SYMBOL TABLE, assembler 5-60
*TRANSFER TRACE, FORTRAN 5-70
*TWO PASS MODE, assembler 5-52
*XREF assembler 5-56
// * (comments), monitor 5-9
// ASM, monitor 5-5
// CEND, monitor 5-11
// COBOL, monitor 5-6
// CPRNT, monitor 5-11
// DEND patch 4-13
// DUP, monitor 5-6
// EJECT, monitor 5-11
// FOR, monitor 5-6
// JOB, monitor 5-2
// PAUS, monitor 5-10
// RPG, monitor 5-6
// TEND, monitor 5-10
// TYP, monitor 5-10
// XEQ, monitor 5-7
control card, RPG 5-74
core system loader that you punch 8-10
DUP 5-18
end-of-file, DFCNV 4-25
end-of-file control card, RPG 5-74
field specification, DFCNV 4-24
file description, DFCNV 4-21
FORTRAN 5-64
IBM-supplied system loader 8-2
load mode, system loader that you punch 8-8
mainline program, DFCNV 4-21
MODIF patch 4-9
MODSF patch 4-15
monitor 5-1
optional DFCNV 4-24
PHID, IBM-supplied system loader 8-2
RPG 5-71
SCON, IBM-supplied system loader 8-2
supervisor 5-12
system configuration system loader 8-9
system loader that you punch 8-7
TERM, IBM-supplied system loader 8-2
tips for using *EQUAT 6-48
type 81, IBM-supplied system loader 8-7
conversion and transfer operations, summary of DUP 5-21
conversion factors, appendix L, disk storage unit L-1
conversion of the mainline program during core load construction 3-10
conversion subroutines, assembler program use of 1403 6-37
converting FORTRAN data files to RPG data files 4-20
COPY
(*see also* cartridge copy code)
disk maintenance program 4-7
error messages A-60
messages A-60
copying cartridges
COPY disk maintenance program 4-7
operating procedure (DCIP) 9-20

core communications area (COMMA) 3-2
The part of core that is reserved for the work area and parameters required by the monitor programs. In general a parameter is found in COMMA if it is required by 2 or more monitor programs and is required to load a program stored in disk core image format. Otherwise, a parameter is found in DCOM. COMMA is initialized by the supervisor during the processing of a JOB monitor control record.
core dump printout, appendix F F-1
core dump programs
console printer 9-1
supervisor 3-3
1403 Printer 9-4
1132 Printer 9-1
core image buffer (CIB) 2-8
The buffer on which most of the first 4K of core is saved while a core load is being built. The CIB is also used to save any part of COMMON defined below location 4096 during a link-to-link transfer of control.
contents during core load construction 3-7
deleting 4-8
on a nonsystem cartridge 2-13
on a system cartridge 2-8
specifying for current job 5-4
core image header
The part of a core image program that includes such parameters as the word count of the core load, the ITV, and the setting for index register 3.
construction during core load build 3-9
contents of I-5
core image header storage area, restriction on use by FORTRAN subroutines 3-9
core image loader 3-13
processing of low COMMON 3-14
transfers core load into core 3-8
use of $DUMP entry in skeleton supervisor 3-13
use of $EXIT entry in skeleton supervisor 3-13
use of $LINK entry in skeleton supervisor 3-14
core image loader wait code, card B-9
core image program 3-9
A mainline that has been converted, along with all of its required subroutines, to disk core image format. Included in the core image program are any LOCALs and/or SOCALs that are required. This term should not be confused with core load, *which refers to the part of a core image program that is read into core storage just prior to execution.*
core load
A mainline, its required subroutines, and its interrupt, CALL and LIBF transfer vectors. This term should not be confused with core image program.
assignment of origin 3-9
construction of a 3-7
layout in core ready for execution 3-14
origin locations used by the core load builder 3-9
restriction on number of CALLs in a 3-13
restriction on number of LIBFs in a 3-13
specifying I/O devices for FORTRAN 5-65
core load builder (CLB) 3-7
construction of a core load 3-7
core load origin locations assigned by 3-9
how called 3-7
messages A-55
provision for LOCALs 3-11
provision for SOCALs 3-11
use of core storage 3-8
use of the CIB 3-7
use of working storage 3-8
core load construction
assignment of core load origin 3-9
contents of CLB during 3-7
contents of core storage during 3-8
contents of working storage during 3-8
conversion of mainline during 3-10
equating subroutines during 3-11
incorporating subroutines during 3-10
of core image header 3-9
processing data files during 3-10
processing of define file tables 3-10
processing of transfer vector during 3-13
processing the SCRA during 3-10
substituting subroutines during 3-11
use of FLIPR 3-11
core load origin
how assigned by an assembler programmer 3-10
how assigned by a FORTRAN programmer 3-10
locations assigned by the core load builder 3-9
core location
altering the contents of a 7-14
displaying the contents of a 7-14
selecting a 7-14
core map 6-13
how to specify printing of 5-8
reading a FORTRAN 6-15 and 16
reading an assembler 6-14
reading an RPG 6-18
core storage
contents during core load construction 3-8
core load builder use of 3-8
mainline programs that use all of 6-9
manual dump of 7-14
CORE system loader control record
for card system 8-10
for paper tape system 8-10
format 8-10
CPRNT monitor control record
format 5-11
how to avoid overprinting when using 6-5
prints monitor and supervisor control records on console printer 5-11
cross reference symbol table, how to read 5-57
CSF block
A group of data words, not more than 51, of a program in card system format. In this format, the first 6 data words of every CSF block are indicator words. These 6 words are always present, even though all 6 are not needed. A CSF block is equivalent to words 4 through 54 of the CSF module (data card) of which the block is a part.
CSF module
A group of words consisting of a data header and CSF blocks for a program in card system format. A CSF module is equivalent to a data card in card system format. A new CSF module is created for every data break. A data break occurs (1) for an ORG, BSS, BES, or DSA statement, (2) when a new data card is required to store the words of a program, and (3) at the end of a program.
cushion area (in IBM system area) 2-6
An area immediately following the system programs on disk that provides for expansion of the monitor system programs in a reload operation. The cushion area is initialized in an initial load to occupy the sectors remaining on the cylinder occupied by the system programs, plus one complete cylinder.
cylinder, description of 2-2
cylinder 0
on a nonsystem cartridge 2-12
on a system cartridge 2-3

D-field type, DFCNV J-5
DATA command, RJE 10-5
data card format, CDS I-9
data file
A collection of data. Also, an area in either the user area or the fixed area in which data is stored.
data file, DDF format I-4
data file names, duplicate 6-6
data file processing
assembler and RPG disk file organization and processing 6-27
calculating ISAM file sizes 6-29
calculating sequentially organized file sizes 6-29
FORTRAN disk file organization and processing 6-23
programming tips and techniques 6-23
data files
dumping and restoring 6-8
FORTRAN converting to RPG data files 4-20
initializing $$$$$ for use with FORTRAN unformatted I/O 6-27
processing 6-23
processing during core load construction 3-10
reserving disk space for (*DFILE DUP control record) 5-48
RPG, converting FORTRAN data files to 4-20
data formats NCF name code I-15
data header I-2
The first pair of words in a module for a program in disk system format. The first word contains the loading address of the module; the second the total number of words contained in the module. The data header for the last module contains the effective program length, followed by a word count of zero.
data records
keyboard input during FORTRAN program execution 6-50
MODIF binary patch 4-12
MODIF hexadecimal patch (hex) 4-11
MODIF patch 4-11
MODSF D-mode patch 4-19
MODSF P-mode patch 4-17
MODSF patch 4-17
DATA statement, length of FORTRAN 6-51
data words, number in a sector 2-3
DCI format
disadvantages of storing a program in 6-7
storing a program in, *STORECI 5-38
DCI program
format I-5
header I-5
DCIP
disk analysis subroutine 9-9
disk compare subroutine 9-9
disk copy subroutine 9-8
disk dump subroutine 9-9
disk patch subroutine 9-9
initialization subroutine 9-8
operating procedures 9-9
stand-alone utility program 9-8
DCIP operating procedures 9-9
disk analysis 9-36
disk compare 9-40
disk copy 9-20
disk dump 9-28
disk initialization 9-12
disk patch 9-32
how to stop 9-10
materials needed for 9-10
preparation for 9-9
DCIP program, loading the 9-11
DCOM
disk communications area 2-6
on a nonsystem cartridge 2-6
on a system cartridge 2-6
update error messages A-37
DCOM-SYSUP update error messages A-37
DCYL defective cylinder table in sector @IDAD 2-5
DDF data file format I-4
decimal and hexadecimal disk addresses, appendix K K-1
defective cylinder table (DCYL), in sector @IDAD 2-5
DEFINE DUP control record
changing the size of the fixed area 5-45
DEFINE FIXED AREA examples 5-46
DEFINE FIXED AREA format 5-45
DEFINE VOID format 5-46
defining the fixed area 5-45
deleting the assembler or compilers 5-46
define file table
The table at the beginning of every mainline that refers to defined files. This table contains one 7-word entry for each file that is defined.
how processed during core load construction 3-10
defined files, use of 6-9
defining an absolute starting address, *ORIGIN FORTRAN control record 5-71
defining the fixed area, *DEFINE DUP control record 5-45
defining the length of COMMON, *COMMON assembler control record 5-63
DELETE DUP control record
deleting information from the UA or FX 5-44
examples 5-44
format 5-44
deleting
duplicate records caused by a disk error during an ISAM add operation 6-34
I/O device subroutines 4-3
information from the UA or FX 5-44
the assembler or compilers 5-46
the CIB 4-8
DEND MODIF patch control record
description 4-13
format 4-13
destroyed cartridges
patching with DCIP 6-5
restoring 6-5
restoring sector addresses with DCIP 6-5
restoring with a system reload 6-5
device and machine requirements, RJE 10-1
DFCNV disk maintenance program 4-20
B-field type J-4
C-field type J-4
control records 4-21
converting FORTRAN data files to RPG 4-20
D-field type J-5
E-field type J-6
end-of-file control record 4-25
error messages A-68
example 4-25
F-field type J-6
field specification control record 4-24
field type examples for J-1
file description control record 4-21
I-field type J-1
J-field type J-2
messages A-67
optional control record 4-24
R-field type J-2
X-field type J-7

DFILE DUP control record
computing the number of sectors to be reserved 6-9
format 5-48
reserving disk space for data files or macro libraries 5-48
disadvantages of storing a program in DCI format 6-7
DISC disk maintenance program 4-6
error messages A-59
initializing satellite cartridges 4-6
messages A-59
discontinuing RJE output 10-7
disk 1-2.1
A single disk in an IBM 2315 Disk Cartridge or any one of 3 or 5 usable disks in an IBM 1316 Disk Pack, Model 12 or 11.
disk addresses, appendix K, decimal and hexadecimal K-1
disk analysis subroutine, DCIP 9-9
disk block I-2
One-sixteenth of a disk sector, that is, 20 disk words. A disk block is the smallest distinguishable increment for programs stored in disk system format. Thus, the monitor system permits packing of disk system format programs at smaller intervals than the hardware otherwise allows.
disk cartridge (see disk)
disk cartridge initialization program (*see also* DCIP) 9-8
disk cartridges, summary of the contents of 2-14
disk communications area (DCOM) 2-6
The disk sector of cylinder 0 that contains the work areas and parameters for the monitor programs.
disk files, organization and processing of FORTRAN 6-23
disk format, LET/FLET D-1
disk formats I-2
disk core image format (DCI) I-5
The format in which core image programs are stored on the disk prior to execution.
disk data format (DDF) I-4
The format in which a data file is stored in either the user area or the fixed area.
disk system format (DSF) I-2
The format in which mainlines and subprograms are stored on the disk as separate entities. A program in disk system format cannot be executed; it must first be converted to disk core image format with either an XEQ monitor control record or a STORECI DUP control record.
disk I/O, how to specify FORTRAN unformatted 5-5
disk I/O subroutine in the resident monitor 3-3
disk I/O subroutines
how to specify 5-8
using the 6-4
disk initialization, satellite 4-6
disk input at the RJE work station 10-2
disk maintenance programs
ADRWS 4-8
COPY 4-7
DFCNV 4-20
DISC 4-6
DLCIB 4-8
DSLET 4-7
ID 4-7
IDENT 4-5
MODIF 4-8
MODSF 4-14
system library mainline programs 4-5
Disk Monitor System, operating the 1130 7-1
disk organization 2-1
disk patch subroutine, DCIP 9-9
disk placement of monitor system programs 3-1
disk storage, components of 2-2
Disk Storage Drive, readying the 2310 7-3
disk storage unit conversion factors, appendix L L-1
disk system format program I-2
A program that is stored in disk system format; sometimes called a DSF program.
disk type (DTYP), in sector @IDAD 2-5
disk utility program (DUP)
description 3-4
monitor system program 3-4
DISKN, in the resident monitor 3-3
disk-resident supervisor programs
auxiliary supervisor 3-3
monitor control record analyzer 3-3
supervisor control record analyzer 3-3
supervisor core dump program 3-3
DISKZ
in the resident monitor 3-3
subroutine in sector @IDAD 2-5
where stored on a system cartridge 2-7
DISK1 in the resident monitor 3-3
displaying the contents of a selected core location 7-14
DLCIB disk maintenance program
deleting the CIB 4-8
error messages A-61
messages A-61
D-mode patch data record format 4-19
double buffering example 6-36
double buffering in assembler programs 6-35
drive number, physical 1-2.3
DSF block I-2
A group of data words, not more than 9, of a program in disk system format. In this format, the first data word of every DSF block is an indicator word. Normally every DSF block in a DSF module consists of 9 data words, including an indicator word; but if the DSF module contains a number of data words that is not a multiple of 9, then the next-to-last DSF block contains less than 9 data words.
DSF module I-2
A group of words consisting of a data header and DSF blocks for a program in disk system format. A new DSF module is created for every data break. A data break occurs (1) for every ORG, BSS, BES, or DSA statement, (2) when a new sector is required to store the words of a program, and (3) at the end of the program.
DSF program
format I-2
header I-3
DSLET dump SLET program 4-7
DTYP, disk type in sector @IDAD 2-5
DUMP DUP control record
additional field information 5-24
examples 5-24
format 5-23
transferring stored information to working storage or I/O devices 5-22
dump format, LET/FLET D-2
dump of core storage, manual 7-14
dump program, supervisor core 3-3
DUMPDATA DUP control record
additional field information 5-26
examples 5-26
format 5-25
transferring stored information to working storage or I/O devices 5-24
DUMPDATA E DUP control record
additional field information 5-27
examples 5-28
format 5-27
transferring stored information to working storage or I/O devices 5-26
DUMPFLET DUP control record
additional field information 5-29
examples 5-30
format 5-29
printing the contents of FLET 5-29

DUMPFLET listing D-6
dumping and restoring data files, tips on monitor control and usage 6-8
dumping disk cartridges, operating procedure 9-28
dumping FORTRAN DSF programs to cards 6-52
dumping SLET, DSLET disk maintenance program 4-7
DUMPLET DUP control record
  additional field information 5-28
  examples 5-29
  format 5-28
  printing the contents of LET 5-28
DUMPLET listing D-3
DUP control records 5-18
  *DEFINE 5-45
  *DELETE 5-44
  *DFILE 5-48
  *DUMP 5-22
  *DUMPDATA 5-24
  *DUMPDATA E 5-26
  *DUMPFLET 5-29
  *DUMPLET 5-28
  *DWADR 5-47
  *MACRO UPDATE 5-49
  *STORE 5-30
  *STORECI 5-38
  *STOREDATA 5-33
  *STOREDATACI 5-37
  *STOREDATAE 5-34
  *STOREMOD 5-42
  altering LET and FLET with 5-20
  general coding information 5-18 and 19
  general functions 5-18
  information transfer and format conversion with 5-20
  where placed in input stream 5-18
DUP error messages
  description of A-13
  listing of A-14
DUP messages A-13
DUP monitor control record
  format 5-6
  general function 5-6
DUP transfer and conversion operations, summary of 5-21
duplicate program and data file names 6-6
duplicate program names, example 6-6
duplicate records, deleting when caused by a disk error during an ISAM add operation 6-34
DWADR DUP control record
  format 5-47
  writing sector addresses in WS 5-47

E-field type, DFCNV J-6
effective program length
  The ending address of a relocatable program. For example, in assembler language programs, this address is the last value used by the location assignment counter during assembly. This value is assigned to the END statement.
EJECT monitor control record
  example 6-5
  format 5-11
  skips the printer to a new page 5-11
  usage of 6-5
END MODSF patch control record
  description 4-19
  format 4-19
end-of-data indicator for RJE user exit 10-9
end-of-file control card, RPG 5-74
end-of-file control record, DFCNV 4-25
end-of-file indicators, RJE 10-6
end-of-program card, CDS program I-10
ending address of RJE user exit data 10-8
entering jobs
  from the card reader 7-12
  from the console keyboard 7-12
  from the paper tape reader 7-12
entry point
  Either (1) the symbolic address (name) where a program is entered, (2) the absolute core address where a program is entered, or (3) the address, relative to the address of the first word of a subroutine, where a subroutine is entered.
entry points to the skeleton supervisor 3-2
EQUAT supervisor control record
  additional field information 5-17
  for substituting subroutines 5-17
  format 5-17
  maximum number of substitutions 5-17
  tips for using 6-48
equatable FORTRAN I/O subroutines 6-48
ERASE FIELD key 7-13
error codes
  assembler, description A-2
  FORTRAN, description A-7
  listing of assembler A-3
  listing of FORTRAN A-9
error messages
  assembler, description A-2
  auxiliary supervisor A-58
  COPY A-60
  DCOM update A-37
  DFCNV A-68
  DISC A-59
  DLCIB A-61
  DUP, description A-13
  FORTRAN compilation A-7
  ID A-60
  listing of assembler A-5
  listing of DUP A-14
  listing of MUP A-14
  MODIF A-62
  MODSF A-66
  MUP, description A-13
  RJE A-28
  SGJP A-26
  supervisor A-36
  system loader A-22
  SYSUP update A-37
error notes
  listing of RPG compiler A-40
  RPG compiler, description A-38
error recovery procedures, RJE 10-11
error statistics, RJE 10-12
error traps
  postoperative 3-2
  preoperative 3-2
  PROGRAM STOP key 3-2
error wait codes
  appendix B, monitor system B-1
  PTUTL B-9
error waits
  cold start program B-1
  ISS subroutine preoperative B-2
  listing of ISS subroutine B-3
errors
  console printer subroutine B-8
  FORTRAN I/O 6-51
  I/O device subroutine B-5

errors (continued)
paper tape subroutine B-9
1442 card subroutine B-6
2501 card subroutine B-8
2501 card subroutine feed check B-8
2501 card subroutine read check B-8
execution
The execution of a program specified on an XEQ monitor control record and any subsequent links executed via CALL LINK statements. The execution is complete when a CALL EXIT is executed.
EXTENDED PRECISION FORTRAN control record
format 5-68
specifying extended FORTRAN precision 5-68

F-field type, DFCNV J-6
feed check error, 2501 card subroutine B-8
field specification control record
description 4-24
example 4-24
specifying repeated fields 4-24
field type examples for DFCNV, appendix J J-1
file description DFNCV control record
additional field information 4-23
description 4-21
format 4-22
file index, contents of an ISAM 6-32
file label, computing for ISAM files 6-30
file labels, contents of ISAM 6-31
file map, reading a 6-13
file organization, indexed sequential (ISAM) 6-28
file processing
assembler and RPG disk file organization and processing 6-28
sequential 6-28
file records
how FORTRAN formatted are written in sectors 6-25
how unformatted FORTRAN are written in sectors 6-26
file sizes
calculating ISAM 6-29
calculating sequentially organized 6-29
computing for DFCNV 4-23
FILE statement, assembler 3-10
FILES supervisor control record
additional field information 5-15
continuing to another FILES control record 5-15
equating program file numbers to stored data files 5-15
format 5-15
how processed 6-9
maximum number of equated data files 5-15
fixed area (FX)
The area on disk in which you store core image programs and data files if you want them to always occupy the same sectors. Packing never occurs in the fixed area. Programs in disk system format cannot be stored in this area.
changing the size of 5-45
defining 5-45
deleting information from 5-44
description 2-9
storing information in
*STOREDATA 5-33
*STOREDATACI 5-37
*STOREDATAE 5-34
FLET
fixed location equivalence table 2-8
printing the contents of 5-29
FLET and LET, altering with DUP control records 5-20
FLIPR
core load construction use of 3-11
system library utility subroutines 4-4
flowchart blocks in operating procedures, functions of 1-1
FOR monitor control record
format 5-6
general function 5-6
format conversion and information transfer, DUP control records 5-20
format of hexadecimal addresses used in this publication 1-2
format of symbolic addresses used in this publication 1-2
formats
appendix I I-1
card I-6
CDC program card I-11
CDD data file card I-11
CDS data card I-9
CDS program I-6
data I-15
DCI program I-5
DDF data file I-4
disk I-2
DSF program I-2
NCF I-15
paper tape I-12
PRD print I-13
print I-13
formatted disk file
The organization of a FORTRAN disk data file to allow random accessing of fixed length records. Data conversion is not possible.
formatted file records, how FORTRAN writes in sectors 6-25
formatted FORTRAN I/O statements 6-25
FORTRAN
calling sequence for SYSUP 6-21
compilation error messages A-7
compilation messages A-7
error codes, description A-7
I/O wait codes B-10
listing of error codes A-9
messages A-7
FORTRAN A-conversion 6-50
FORTRAN allocation addresses, locating 6-18
FORTRAN compiler, description 3-6
FORTRAN control records 5-64
**(header information) 5-69
*ARITHMETIC TRACE 5-70
*EXTENDED PRECISION 5-68
*IOCS 5-65
*LIST ALL 5-67
*LIST SOURCE PROGRAM 5-66
*LIST SUBPROGRAM NAMES 5-66
*LIST SYMBOL TABLE 5-67
*NAME 5-69
*ONE WORD INTEGERS 5-68
*ORIGIN 5-71
*TRANSFER TRACE 5-70
general functions 5-64
how to code 5-64
where placed in the input stream 5-64
FORTRAN core load
A core load that is built from a mainline written in the FORTRAN language.
specifying I/O devices for 5-65
FORTRAN core map, reading a 6-15 and 16
FORTRAN data files, converting to RPG data files 4-20
FORTRAN DATA statement, length of 6-51

FORTRAN disk files
data file processing 6-23
formatted FORTRAN I/O statements 6-25
initializing $$$$$ data files 6-27
organization and processing 6-23
unformatted FORTRAN I/O statements 6-26
FORTRAN DSF programs, dumping to cards 6-52
FORTRAN formatted file records, how written in sectors 6-25
FORTRAN I/O errors, tips for FORTRAN programmers 6-51
FORTRAN I/O statements
formatted 6-25
unformatted 6-26
FORTRAN I/O subroutines, equatable 6-48
FORTRAN IOCS control record, I/O subroutines called by 6-49
FORTRAN logical unit numbers 6-24
FORTRAN object program paper tape data record format 6-49
FORTRAN program
control of the console printer 6-50
listing a 5-66
listing with subprogram names and symbol table 5-67
FORTRAN program control of the console printer 6-50
FORTRAN program execution
// records read during 6-51
keyboard input of data records during 6-50
FORTRAN programmer, how to assign core load origin 3-10
FORTRAN programmers, tips for 6-48
FORTRAN READ and WRITE statements, maximum record sizes used in 6-24
FORTRAN sample program H-1
FORTRAN source cards, invalid characters in 6-49
FORTRAN subprogram names, listing with a program and symbol table 5-67
FORTRAN subroutines, restriction on use of core image header storage area 3-9
FORTRAN symbol table, listing 5-67
FORTRAN unformatted file records, how written in sectors 6-26
FORTRAN unformatted I/O
initializing $$$$$ data files for use with 6-27
sample program using H-12
FSLEN system library utility subroutines 4-4
function
A subprogram that evaluates a mathematical relationship between a number of variables. In FORTRAN, a FUNCTION is a subprogram that is restricted to a single value for the result. This type of subprogram is called by direct reference.
functions of console operator keys during monitor system control 7-13
functions of flowchart blocks in operating procedures 1-1
header, construction of the core image 3-9

generation of the 1130 RJE work station program 10-3
grouping of assembler mnemonics 6-35
G2250 supervisor control record
examples 5-17
format 5-16
specifying use of graphic subroutine package 5-16

halt codes (see wait codes)
header
DCI program I-5
DSF program I-3
heading, how to specify a page 5-5
hexadecimal address format used in this publication 1-2
hexadecimal disk addresses, appendix K K-1
hexadecimal MODIF patch data record (hex), format 4-11

I-field type, DFCNV J-1
I/O device subroutine errors B-5
I/O devices
list of principal 8-9
specifying for FORTRAN core loads 5-65
I/O errors, FORTRAN 6-51
I/O statements, unformatted FORTRAN 6-26
I/O subroutines
called by FORTRAN IOCS control record 6-49
equatable FORTRAN 6-48
how to specify disk 5-8
using the disk 6-4
I/O wait codes FORTRAN B-10
IBM area 2-8
That part of disk storage that is composed of DCOM, the CIB and the monitor programs. This area is also known as the IBM system area or system area.
IBM-supplied system loader control records 8-2
PHID 8-3
SCON 8-2
system program sector break cards 8-4
TERM 8-2
type 81 8-7
IBM system area
CIB 2-8
cushion area 2-6
FLET 2-8
LET 2-8
SCRA in the 2-7
system device subroutine area 2-7
IBM system area on a nonsystem cartridge 2-13
IBM system area on a system cartridge 2-6
ID, disk maintenance program 4-7
error messages A-60
messages A-60
IDAD 2-5
IDENT, disk maintenance program 4-5
messages A-59
ILS branch table (IBT, *see* interrupt branch table)
A table consisting of the addresses of the interrupt entry points for each ISS used for the interrupt level. An IBT is required by the ILS for an interrupt level with which more than one device is associated.
ILS header card CDS program I-9
ILS subroutine example 6-44
ILS subroutines, rules for writing 6-42
ILSs
how to specify special 5-8
writing by assembler language programmers 6-42
ILS02 in the skeleton supervisor 3-2
ILS04 in the skeleton supervisor 3-2
IMM STOP key (immediate stop) 7-13
incore subprogram
A subprogram that remains in core storage during the entire execution of the core load of which it is a part. ILSs are always incore subprograms, whereas LOCALs and SOCALs never are.
incorporating subroutines, core load construction 3-10
index register 3, assembler program use of 6-35
index sectors, computing for ISAM files 6-30
indexed sequential access method files, calculating size of 6-29
indexed sequential file organization, ISAM 6-28

indicator word
The first word of a DSF block indicating which of the following data words should be incremented (relocated) when relocating a program in disk system format. This word also indicates which words are LIBF, CALL, and DSA names and the graphic instruction GSE, GBE, or GBCE. Programs in disk system format all contain indicator words. Each pair of bits in the indicator word is associated with one of the following data words; the first pair with the first data word following the indicator word, etc.
information transfer and format conversion, DUP control records 5-20
initial load, monitor system 8-1
initial load operating procedure
  card system 8-15
  paper tape system 8-28
initial program load
The action that occurs when the PROGRAM LOAD key is pressed. One record is read into core, starting at location zero, from the hardware device that is physically wired to perform this function. The record read, usually a loader, then instructs the system as to the next action to be performed; such as, load more records.
initialization, satellite disk (DISC) 4-6
initialization subroutine of DCIP 9-8
initializing $$$$$ data files for use with FORTRAN unformatted I/O 6-27
initializing disk cartridges, operating procedure (DCIP) 9-12
input
  changed LOGON effect on RJE input 10-3
  stacked job arrangement 1-3
input at the work station, RJE 10-2
input of data records, from the keyboard during FORTRAN program execution 6-50
INT REQ key (interrupt request) 7-13
INT REQ service subroutine
  assembler 6-45
  for any core load example 6-46
  for core load using TYPEZ, WRTYZ, TYPE0 or WRTY0 6-47
  rules for coding an 6-45
interrupt branch table 6-42
interrupt level subroutine (ILS) 6-42
A subroutine that analyzes all interrupts on a given level, that is, it determines which device on a given level caused the interrupt and branches to a servicing subroutine (ISS) for the processing of that interrupt.
interrupt level 2, skeleton supervisor 3-2
interrupt level 4, skeleton supervisor 3-2
interrupt levels, specifying for ISSs (*LEVEL assembler control record) 5-61
interrupt request key (INT REQ) 7-13
interrupt service subroutine (ISS) 6-37
A subroutine that (1) manipulates a given I/O device and (2) services all interrupts for that device after they are detected by an ILS.
interrupt transfer vector (ITV, *see* transfer vector)
The contents of words 8 through 13 or core, which are the automatic BSI instructions which occur with each interrupt. In other words, if an interrupt occurs on level zero and if core location 8 contains 500, an automatic BSI to core location 500 occurs. Similarly, interrupts on levels 1 through 5 cause BSIs to the contents of core locations 9 through 13, respectively.
I/O device subroutines, deleting 4-3
IOAR header
The words required by an I/O device subroutine (ISS). They must be the first or the first and second words of the I/O buffer.
IOCS FORTRAN control record
  format 5-65
  I/O subroutines called by 6-49
  specifying I/O devices for FORTRAN core loads 5-65
ISAM add operation, deleting duplicate records caused by a disk error during an 6-34
ISAM file, contents of 6-31
ISAM file index, contents of 6-32
ISAM file label, contents of 6-31
ISAM file parameters, sample program to calculate H-17
ISAM files
  calculating size of 6-29
  computing file label for 6-30
  computing index sectors 6-30
  computing overflow sectors 6-30
  computing prime data sectors 6-30
  random processing of 6-28
  sequential processing of 6-28
ISAM indexed sequential file organization 6-28
ISAM overflow area, contents of 6-33
ISAM prime data area, contents of 6-32
ISS, specifying interrupt level for (*LEVEL assembler control record) 5-61
ISS, subroutines in system library 4-2
ISS branch table 6-42
ISS counter
A counter in COMMA (word $IOCT) that is incremented by one upon the initiation of every I/O operation and decremented by one upon completion of the I/O operation.
ISS header card, CDS program I-8
ISS subroutine error waits, listing of B-3
ISS subroutine example 6-38
ISS subroutine preoperative error waits B-2
ISS subroutines
  nameN 4-3
  nameZ 4-2
  name0 4-2
  name1 4-2
ISS subroutines in system library 4-2
ISSs, writing by assembler language programmers 6-37

J-field type, DFCNV J-2
JECL for the 1130 RJE work station 10-5
job
A group of tasks (subjobs) that are performed by the monitor system and are interdependent; that is, the successful execution of any given subjob (after the first) depends on the successful execution of at least one of those that precede it.
  how to specify a temporary 5-4
  how to use SYSUP when changing cartridges during a 6-20
JOB monitor control record
  additional field information 5-4
  examples 5-5
  format 5-2 and 3
  general function 5-2
jobs
  entering from the card reader 7-12
  entering from the console keyboard 7-12
  entering from the paper tape reader 7-12
  restrictions on temporary 5-4
  stacked input arrangement 6-1

keyboard, entering jobs from the console 7-12
keyboard input at the RJE work station 10-2

keyboard input of data records during FORTRAN program execution 6-50
keyboard operation, starting 7-12
keyboard operation, stopping 7-13
keyboard procedures, RJE console 10-10
keys
  backspace (←) 7-12
  ERASE FIELD 7-13
  IMM STOP 7-13
  INT REQ 7-13
  PROGRAM START 7-13
  PROGRAM STOP 7-13
  REST KB 7-13

layout of core load ready for execution 3-14
length of FORTRAN DATA statement 6-51
LET
  location equivalence table 2-8
  printing the contents of *DUMPLET 5-28
LET and FLET, altering with DUP control records 5-20
LET/FLET
    The location equivalence table (LET) for the user area and the fixed location equivalence table (FLET) for the fixed area. These are disk resident tables through which the disk addresses of programs and data files stored in the user area or fixed area are found. On a system cartridge, LET occupies the cylinder preceding the user area. If a fixed area is defined, FLET occupies the cylinder preceding it; otherwise, there is no FLET.
  appendix D D-1
  disk format D-1
  dump format D-2
  entry format D-1
  sector header format D-1
  1DUMY entry D-1
LET in IBM system area 2-7
LEVEL assembler control record, specifying interrupt levels for ISSs 5-61
LIBF subroutine
    A subroutine that must be referenced with an LIBF statement. The type codes for subroutines in this category are 3 and 5.
LIBF TV 3-13
    The transfer vector through which LIBF subroutines are entered at execution time.
LIBFs, restriction on number in a core load 3-13
library, monitor system 4-1
library maintenance, system 4-14
link
    A link is a core image program that is read into core for execution as a result of the execution of a CALL LINK statement.
linking between programs, how to avoid overprinting when 6-5
LIST ALL FORTRAN control record 5-67
LIST assembler control record, listing an assembler program 5-53
list deck
  punching (*LIST DECK) 5-57
  punching with error flags (*LIST DECK E) 5-59
  reading a punched 5-58
LIST DECK assembler control record, punching a list deck 5-57
LIST DECK E assembler control record, punching a list deck with error flags 5-59
LIST SOURCE PROGRAM FORTRAN control record, listing a FORTRAN program 5-66
LIST SUBPROGRAM NAMES FORTRAN control record 5-66
LIST SYMBOL TABLE FORTRAN control record 5-67
listings
  DUMPFLET D-6
  DUMPLET D-3
  SLET E-1
  monitor system library, appendix C C-1
  resident monitor G-1
load-although-not-called (NOCAL) subroutine 6-11
    A subroutine included in a core image program although it is not referenced in the core image program by an LIBF or CALL statement. Debugging aids such as a trace or a dump fall into this category.
load mode control record (system loader) 8-8
  for card system 8-8
  for paper tape system 8-8
  format 8-8
load mode control tape (system loader) 8-10
  materials needed for preparation of 8-10
  preparation of 8-10
load-on-call (LOCAL) subroutine 3-11
    A subroutine that is a part of a core image program, but resides on disk when not in use during execution. A LOCAL is read from the disk into a special overlay area incore when called during execution. LOCALs, which are specified for any given execution by the user, are a means of gaining core storage at the expense of execution time. The core load builder constructs the LOCALs and all linkages to and from them.
loading
    The process of reading information into core storage, usually from disk.
loading address
    The address at which a mainline, subroutine, core load, or DSF module is to begin. For mainlines and DSF modules, the loading address is either absolute or relative. For subroutines, it is always relative, whereas, for core loads, it is always absolute.
loading the assembler and compilers 8-8
loading the DCIP stand-alone utility program 9-11
LOCAL and NOCAL control record usage, tips on monitor control and usage 6-10
LOCAL-call-LOCAL, how to specify 5-8
LOCAL-calls-a-LOCAL, usage 6-10
LOCAL supervisor control record
  additional field information 5-13
  coding for linked programs 5-13
  continuing to another LOCAL control record 5-13
  format 5-13
  specifying LOCAL subroutines 5-13
  when mainline program is in working storage 5-14
LOCALs, core load builder provision for 3-11
LOCALs, using 6-9
locating FORTRAN allocation addresses 6-18
location assignment counter
    A counter maintained in the assembler for assigning addresses to the instructions it assembles. A similar counter is maintained in the core load builder for loading purposes.
location equivalence table (LET) 2-8
logic flow of the monitor system 3-15
logical cartridge assignments, specifying 5-4
logical record length of RJE user-exit data 10-8
logical unit numbers, FORTRAN 6-24
LOGON, effect on RJE input of changed 10-3
long instruction
    An assembler instruction that occupies two core storage locations.

low COMMON, how processed by core image loader 3-14
The words of core that are saved in the core image buffer when linking from program to program. This area exists even if there is no COMMON.

machine and device requirements, RJE 10-1
MACLIB assembler control record
format 5-63
specifying the use of the macro library 5-63
macro libraries, reserving disk space for 5-48
macro library, specifying the use of 5-63
macro overflow, specifying WS sectors for 5-61
MACRO UPDATE DUP control record
calling the macro update program 5-49
format 5-49
macro update program (MUP), calling 5-49
mainline
A program about which a core image program is built. The mainline is normally the program in control and calls subroutines to perform various functions.
mainline header card, CDS program I-6
mainline program, conversion during core load construction 3-10
mainline programs
error messages for monitor system library A-59
messages for monitor system library A-59
system library 4-5
mainline programs that use all of core 6-9
maintenance
system library (MODIF) 4-8
system library (MODSF) 4-14
maintenance programs, disk 4-5
manual dump of core storage 7-14
master cartridge 2-3
The cartridge residing on logical drive zero. A master cartridge must be a system cartridge.
maximum record sizes used in FORTRAN READ and WRITE statements 6-24
merging assembler symbol tables 5-60
messages
auxiliary supervisor error A-58
COPY A-60
core load builder A-54
DCOM update error A-37
DFCNV A-67
DFCNV error A-68
DISC A-59
DLCIB A-61
DUP A-13
FORTRAN A-7
ID A-60
IDENT A-59
listing of DUP error A-14
listing of MUP error A-14
MODIF A-61
MODIF error A-62
MODSF A-65
MODSF error A-66
monitor system library mainline programs A-59
MUP error A-13
printed during cold start 7-10
RJE A-33
RJE error A-28
RPG compiler A-38
SGJP error A-26
supervisor A-35
supervisor error A-36
system loader A-22
SYSUP update error A-37
messages printed during cold start 7-10
messages sent to RJE work stations 10-12
minimum system configuration iii
mnemonics, grouping of assembler 6-35
MODIF disk maintenance program 4-8
*MON patch control record 4-9
*SUB patch control record 4-12
// DEND patch control record 4-13
adding subroutines to the system library 4-12
disk maintenance programs 4-8
error messages A-62
example 4-14
messages A-61
patch control records 4-9
patch data records 4-11
modified EBCDIC code (*see also* name code format)
A 6-bit code used internally by the monitor programs. In converting from EBCDIC to modified EBCDIC, the leftmost 2 bits are dropped.
MODSF disk maintenance program 4-14
*END patch control record 4-19
*PRO patch control record 4-15
error messages A-66
example 4-20
messages A-65
MODSF patch control and data records 4-15
MODSF patch data records
D-mode 4-19
P-mode 4-17
MON MODIF patch control record
additional field information 4-11
description 4-9
format 4-10
monitor
A synonym for the entire 1130 Disk Monitor System, Version 2, which is also known as the monitor system or the disk monitor.
monitor control, tips on 6-1
monitor control record analyzer, disk-resident supervisor programs 3-3
monitor control records 5-1
// *(comments) 5-9
// ASM 5-5
// CEND 5-11
// COBOL 5-6
// CPRNT 5-11
// DUP 5-6
// EJECT 5-11
// FOR 5-6
// JOB 5-2
// PAUS 5-10
// RPG 5-6
// TEND 5-10
// TYP 5-10
// XEQ 5-7
coding of 5-1
functions of 5-1
usage of EJECT 6-5
monitor mode, RJE 10-1
monitor program (*see also* monitor system programs)
One of the following parts of the monitor system: supervisor (SUP), core image loader (CIL), core load builder (CLB), disk utility program (DUP), assembler (ASM), FORTRAN compiler (FOR), RPG compiler (RPG), or COBOL compiler.
monitor system
error wait codes B-1
logic flow of 3-15
using the 1130 with the 7-12

Monitor System, operating the 1130 Disk 7-1
monitor system control, functions of console operator keys during 7-13
monitor system error wait codes, appendix B B-1
monitor system initial load and system reload 8-1
monitor system library 4-1
monitor system library listing, appendix C C-1
monitor system library mainline programs, messages A-59
monitor system operational and error messages, appendix A A-1
monitor system programs 3-1
  assembler 3-5
  core image loader 3-13
  core load builder 3-7
  disk placement of 3-1
  disk utility program 3-4
  FORTRAN compiler 3-6
  RPG compiler 3-6
  supervisor 3-2
monitor system sample programs, appendix H H-1
monitor system sector break cards, listing of 8-5
monitor usage, tips on 6-1
MUP error messages A-13
MUP error messages, listing of A-14

name code format (NCF) I-15
  The format in which the names of subroutines, entry points, labels, etc., are stored for use in the monitor programs. The name consists of 5 characters, terminal blanks are added if necessary to make 5 characters. Each character is in modified EBCDIC code, and the entire 30-bit representation is right-justified in two 16-bit words. The leftmost 2 bits are used for various purposes by the monitor.
name data words
  The format in which constants and the names of variables and subprograms are stored for internal use by the FORTRAN compiler. The first bit of each name data word is set to zero to indicate that the word contains a constant and is set to one of the word contains a name. In either case, the remainder of the word is packed with the characters in modified EBCDIC code.
NAME FORTRAN control record
  format 5-69
  printing the program name on each printed page 5-69
nameN ISS subroutines 4-3
nameZ ISS subroutines 4-2
name0 ISS subroutines 4-2
name1 ISS subroutines 4-2
naturally relocatable program 2-6
  A program that can be executed from any core storage location without first being relocated. The only absolute addresses in such a program refer to parts of the resident monitor, which are fixed.
NOCAL and LOCAL control record usage 6-10
NOCAL example 6-12
NOCAL supervisor control record
  format 5-14
  specifying NOCAL subroutines 5-14
NOCALs, the use of 6-11
nonsystem cartridge 2-3
  A cartridge that does not contain the monitor programs, although it does contain DCOM, LET, and working storage. A nonsystem cartridge can be used only as a satellite cartridge.
  CIB on a 2-13
  cylinder 0 on a 2-12
  description 2-12
  IBM system area on a 2-13
  sector @DCOM on a 2-13
  sector @IDAD on a 2-13
notes
  listing of RPG compiler error A-40
  RPG compiler error, description A-38
null command, RJE 10-6 and 10

object program
  The output from either the assembler, or the FORTRAN, RPG, or COBOL compiler.
object program considerations, RPG 6-52
object program paper tape data record format, FORTRAN 6-49
one word integers FORTRAN control record
  format 5-68
  specifying one word of core for integers 5-68
operating procedures
  analyzing disk cartridges 9-36
  card system initial load 8-15
  card system preload 8-25
  card system reload 8-19
  console printer core dump 9-2
  copying disk cartridges 9-20
  DCIP 9-9
  disk compare 9-40
  dumping disk cartridges 9-28
  functions of flowchart blocks in 1-1
  initializing disk cartridges 9-12
  paper tape reproducing 9-42
  paper tape system initial load 8-28
  paper tape system reload 8-33
  patching disk cartridges 9-32
  preparation for DCIP 9-9
  printer core dump program 9-5
  PTUTL program 9-46
  RJE 10-9
operating the 1130 Disk Monitor System 7-1
operator keys, console 7-13
Optical Mark Page Reader, readying the 1231 7-9
organization, disk 2-1
ORIGIN FORTRAN control record
  defining an absolute starting address 5-71
  format 5-71
origin locations, used by the CLB during core load construction 3-9
origin of a core load
  assignment during core load construction 3-9
  how assigned by assembler programmer 3-10
  how assigned by FORTRAN programmer 3-10
output
  continuing RJE 10-7
  discontinuing RJE 10-7
output to the RJE work station 10-6
overflow area, contents of the ISAM 6-34
overflow sectors, computing ISAM file 6-30
OVERFLOW SECTORS assembler control record
  format 5-61
  specifying WS sectors for symbol table overflow 5-61
  specifying WS sectors for macro overflow 5-61
overlays, subroutines included in SOCAL 3-12
overprinting
  how to avoid when linking between programs 6-5
  how to avoid when using // CPRNT 6-5

P-mode patch data record format 4-19
packing 2-11
  The process of storing programs in the user area to the nearest disk block, thus reducing the average wasted disk space from 160 words per program to 10 disk words per program. This process of moving programs toward the beginning of the user area makes additional space available in working storage.

padding
Areas in the user or fixed area required to start core image programs and data files on a sector boundary. The length of the padding, which is reflected in LET or FLET by a 1DUMY entry, is from one to 15 disk blocks.
page heading, how to specify 5-5
paper tape data record format, FORTRAN object program 6-49
paper tape formats I-12
paper tape input, PTUTL 4-25
paper tape output, PTUTL 4-25
Paper Tape Punch, readying the 1055 7-7
paper tape reader, entering jobs from the 7-12
Paper Tape Reader, readying the 1134 7-6
paper tape reproducing program, stand-alone utility 9-42
paper tape subroutine errors B-9
paper tape system cold start procedure 7-11
paper tape system initial load
  materials needed for 8-28
  operating procedure 8-28
  organization of tapes for 8-29
paper tape system reload
  materials needed for 8-33
  operating procedure 8-33
  organization of tapes for 8-33
paper tape utility program (*see* PTUTL)
patch control and data records
  MODIF 4-9
  MODSF 4-15
patch control records
  MODIF 4-9
  MODIF *MON 4-9
  MODIF *SUB 4-12
  MODIF // DEND 4-13
  MODSF 4-15
  MODSF *END 4-19
  MODSF *PRO 4-15
patch data records
  MODIF 4-11
  MODIF binary 4-12
  MODIF hexadecimal (hex) 4-11
  MODSF 4-15
  MODSF D-mode 4-19
  MODSF P-mode 4-17
patching disk cartridges, operating procedure 9-32
PAUS monitor control record
  format 5-10
  general function 5-10
phase identification control record (*see also* PHID control record)
PHID control record (IBM-supplied system loader)
  format of first 8-3
  format of second 8-4
physical drive number 1-2.3
Plotter, readying the 1627 7-8
postoperative error traps, skeleton supervisor 3-2
PRD print format I-13
preload operating procedure, card system 8-25
preoperative error trap, skeleton supervisor 3-2
preoperative error waits, ISS subroutine B-2
preparation of a load mode control tape 8-10
preparation of a system configuration control tape 8-10
prime data area, contents of the ISAM 6-33
prime data sectors computing for ISAM files 6-30
principal I/O device
The device used for stacked job input to the monitor system. The 2501, 1442, or 1134 can be assigned as the principal I/O device. The keyboard can be temporarily assigned as the principal input device (see "// TYP" under "Monitor Control Records" in Chapter 5). The system loader considers the fastest device defined on the REQ system configuration records to be the principal I/O device.
principal I/O devices, list of 8-9
principal print device
The device used by the monitor system for printing system messages. Either the 1403, 1132, or console printer can be assigned as the principal print device. The system loader considers the fastest print device defined on the REQ system configuration records to be the principal print device.
print format, PRD I-13
print formats I-13
PRINT SYMBOL TABLE assembler control record 5-59
printer, FORTRAN program control of the console 6-50
Printer, readying the 1132 7-4
Printer, readying the 1403 7-4
printer core dump program
  format 9-4
  materials needed to use the 9-4
  operating procedure 9-5
  stand-alone utility 9-4
printing a header on each page, ** header information FORTRAN control record 5-69
printing an assembler listing with a cross reference symbol table, *XREF 5-57
printing an assembler listing with statement numbers, *XREF 5-57
printing cartridge IDs, IDENT disk maintenance program 4-5
printing the contents of FLET, *DUMPFLET 5-29
printing the contents of LET, *DUMPLET 5-28
printing the program name on each printed page, *NAME FORTRAN control record 5-69
PRO MODSF patch control record
  additional field information 4-17
  description 4-15
  format 4-16
procedures
  card system cold start 7-11
  cold start 7-10
  paper tape system cold start 7-11
  RJE console keyboard 10-10
  RJE error recovery 10-11
  RJE operating 10-9
  RJE restart 10-11
processing data files during core load construction 3-10
processing on one disk drive a file that extends over two cartridges, sample program H-13
processing on two disk drives a file that extends over two cartridges, sample program H-16
processing the contents of the SCRA, core load construction 3-10
Processing Unit, readying the 1131 Central 7-2
program
The highest level in the hierarchy describing various types of code. Subprograms and mainlines are subsets of this set.
program header record I-3
The part of a program stored in disk system format that precedes the first DSF module. Its contents vary with the type of program with which it is associated. The program header record contains the information necessary to identify the program, to describe its properties, and to convert it from DSF format to disk core image format.
program names, duplicate 6-6
PROGRAM START key 7-13
PROGRAM STOP key 7-13
PROGRAM STOP key error trap, skeleton supervisor 3-2
program types and subtypes I-1
programmers
  tips for assembler language 6-35
  tips for FORTRAN 6-48
programming tips and techniques 6-1
  data file processing 6-23
  RPG object program considerations 6-52

programming tips and techniques (continued)
tips for assembler language programmers 6-35
tips for FORTRAN programmers 6-48
tips on monitor control and usage 6-1
programs
assembler sample H-7
calculating ISAM file parameters sample H-17
disk maintenance 4-5
disk-resident supervisor 3-3
FORTRAN sample H-1
generation of the RJE work station 10-3
how to avoid overprinting when linking between 6-5
monitor system 3-1
processing on one disk drive a file that extends over two cartridges sample H-13
processing on two disk drives a file that extends over two cartridges sample H-16
RPG sample H-9
size discrepancies in stored 6-7
stand-alone utility 9-1
system library mainline 4-5
types and subtypes I-1
use of FORTRAN unformatted I/O sample H-12
use of reeling sample for multidrive systems H-16
use of reeling sample for one drive systems H-13
use of SYSUP sample for multidrive systems H-16
use of SYSUP sample for one drive systems H-13
PTUTL
error wait codes B-9
example 9-51
materials needed to use the 9-46
operating procedure 9-46
paper tape input 4-25
paper tape output 4-25
publication, how to use this 1-1
publications, related 1130 iii
PUNCH SYMBOL TABLE assembler control record 5-59
punching a list deck *LIST DECK 5-57
punching a list deck with error flags, *LIST DECK E 5-59

R-field type, DFCNV J-2
random processing of ISAM files 6-28
random processing of sequential files 6-28
RDREC, system library utility subroutine 4-4
read check error, 2501 card subroutine B-8
reading
a core map and a file map, tips on monitor control and usage 6-13
a cross reference symbol table 5-57
a punched list deck 5-58
an assembler listing 5-54 and 55
the transfer vector 6-19
readying the
1055 Paper Tape Punch 7-7
1131 Central Processing Unit 7-2
1132 Printer 7-4
1134 Paper Tape Reader 7-6
1231 Optical Mark Page Reader 7-9
1403 Printer 7-4
1442 Model 5 Card Punch 7-5
1442 Model 6 and 7 Card Read Punch 7-5
1627 Plotter 7-8
2310 Disk Storage Drive 7-3
2311 Disk Storage Drive 7-3.1
2501 Card Reader 7-6
receive mode, RJE 10-2
record format of RJE user-exit data 10-9
record sizes, maximum used in FORTRAN READ and WRITE statements 6-24

recovery procedures, RJE error 10-11
reeling
how to use on a multidrive system 6-22
how to use on a one drive system 6-22
how to use with SYSUP 6-22
sample program that uses for multidrive systems H-16
sample program that uses for one drive systems H-13
related 1130 publications iii
reload, monitor system 8-1
reload operating procedure
card system 8-19
paper tape system 8-33
reload table, sector @RTBL on a system cartridge 2-5
A table occupying one sector of the system cartridge. It contains a 3-word entry for each monitor phase that requests SLET information. This entry specifies where the SLET information is to be placed in the requesting phase and the number of SLET entries to be inserted.
relocatable program 3-9
A program that can be executed from any core location. Such a program is stored on the disk in DSF format. The program is relocated by the core load builder.
relocation
The process of adding a relocation factor to address constants and to long instructions whose second words are not (1) invariant quantities, (2) absolute core addresses, or (3) symbols defined as absolute core addresses. The relocation factor for any program is the absolute core address where the first word of that program is found.
relocation indicator 5-55
The second bit in a pair of bits in an indicator word. If the relocation indicator is set to one, the associated data word is to be relocated unless the word is a LIBF, CALL, DSA name, or one of the graphic instructions: GSB, GBE, or GBCE. Pairs of relocation indicators indicate the exceptions as follows: 1000 for LIBF, 1100 for CALL, 1101 for DSA names, 1110 for GBE, and 1010 for GBCE. GBS has indicator bits 11.
remark
An explanation of the use or function of a statement or statements. A remark is a part of a statement, whereas a comment is a separate statement.
remote job entry, machine and device requirements 10-1
remote job entry program, RJE 10-1
REQ system configuration control record 8-9
reserved areas of sector @IDAD 2-5
reserving disk space for data files or macro libraries, *DFILE DUP control record 5-48
resident image, sector @RIAD 2-6
The mirror-image of the resident monitor minus the disk I/O subroutine. The resident image resides on disk and is read into core by the cold start program.
resident monitor 3-2
The area required in core by the monitor system for its operation. This area is generally unavailable for your use. The resident monitor consists of COMMA, the skeleton supervisor, and one of the disk I/O subroutines (normally DISKZ).
COMMA 3-2
disk I/O subroutine 3-3
DISKN 3-3
DISKZ 3-3
DISK1 3-3
listing of G-1
skeleton supervisor 3-2
supervisor 3-2
resident monitor including table of equivalences, appendix G G-1
REST KB (restore keyboard) key 7-13
restart procedures, RJE 10-11
restore keyboard key (REST KB) 7-13

restoring destroyed cartridges, tips on monitor control and usage 6-5
restrictions
caused by temporary mode 5-22
maximum number of substituted subroutines EQUAT control record 5-17
on number of CALLs in a core load 3-13
on number of LIBFs in a core load 3-13
on temporary jobs 5-4
on use of core image header storage area by FORTRAN subroutines 3-9
RJE
.. DATA command 10-5
.. null command 10-6
card input at the work station 10-2
changed LOGON affect on input 10-3
communication considerations 10-1
communication considerations for switched lines 10-2
console entry switches 10-12
console keyboard procedures 10-10
continuing output 10-7
discontinuing output 10-7
disk input at the work station 10-2
error messages A-28
error recovery procedures 10-11
error statistics 10-12
generation of the 1130 RJE work station program 10-3
input at the work station 10-2
JECL for the 1130 work station 10-5
keyboard input at the work station 10-2
machine and device requirements 10-1
messages A-33
messages sent to work stations 10-12
monitor mode 10-1
null command 10-10
operating procedures 10-9
output to the work station 10-6
receive mode 10-2
restart procedures 10-11
transmit mode 10-2
user-exit subroutine 10-8
work station startup 10-9
RJE end-of-file indicators 10-6
RJE messages and error messages A-27
RJE output discontinuing 10-7
RJE user-exit
control characters for data 10-8
data record format 10-9
end-of-data indicator 10-9
ending address of data 10-8
logical record length of data 10-8
parameter list 10-8
starting address of data 10-8
RJE work station, output to the 10-6
RJE work station program, generation of the 1130 10-3
RPG compiler 3-6
error notes A-38
error notes, listing of A-40
messages A-35
monitor system program 3-6
RPG control records 5-71
end-of-file control card 5-74
general functions 5-71
RPG control card 5-74
where placed in input stream 5-71
RPG core load
A core load that is built from a mainline written in the RPG language.
RPG core map reading an 6-18
RPG data files, converting FORTRAN data files to 4-20
RPG disk file organization and processing, data file processing 6-27
RPG monitor control record
format 5-6
general function 5-6
RPG object program considerations 6-52
RPG object program wait codes B-12 and 13
RPG sample program H-9
rules
for coding an INT REQ service subroutine 6-45
for writing ILS subroutines 6-42
for writing ISS subroutines 6-37

sample programs
appendix H, monitor system H-1
assembler H-7
calculating ISAM file parameters H-17
FORTRAN H-1
processing on one disk drive a file that extends over two cartridges H-13
processing on two disk drives a file that extends over two cartridges H-16
RPG H-9
use of FORTRAN unformatted I/O H-12
use of reeling for multidrive systems H-16
use of reeling for one drive systems H-13
use of SYSUP for multidrive systems H-16
use of SYSUP for one drive systems H-13
satellite cartridge 2-3
A cartridge residing on a drive other than logical drive zero. A satellite cartridge can be either a system or a non-system cartridge.
satellite disk initialization, DISC disk maintenance program 4-6
satellite graphic job processor error messages A-26
SAVE SYMBOL TABLE assembler control record
format 5-60
saving the symbol table on disk 5-60
SCON control record, IBM supplied system loader 8-2
SCRA
in the IBM system area 2-7
processing during core load construction 3-10
sector, description of 2-2
sector @DCOM
on a nonsystem cartridge 2-13
on a system cartridge 2-6
sector @HDNG on a system cartridge 2-6
sector @IDAD
cartridge copy code 2-5
cartridge ID in 2-5
cold start program in 2-5
defective cylinder table in 2-5
description 2-5
disk type in 2-5
DISKZ subroutine in 2-5
on a nonsystem cartridge 2-13
on a system cartridge 2-5
reserved areas of 2-5
sector @RIAD resident image on a system cartridge 2-6
sector @RTBL reload table on a system cartridge 2-6
sector addresses
writing in WS, ADRWS disk maintenance program 4-8
writing in WS, *DWADR DUP control record 5-47
sector break cards
listing of monitor system 8-5
used by system loader I-10
system program 8-4
sector header format, LET/FLET D-1
selecting a core location 7-14

sequential file, calculating size of 6-29
sequential file organization, assembler and RPG 6-28
sequential file processing 6-28
sequential files
  random processing of 6-28
  sequential processing of 6-28
sequential processing of ISAM files 6-28
sequential processing of sequential files 6-28
short instruction
    An instruction that occupies only one core storage location.
size discrepancies in stored programs, tips on monitor control and usage 6-7
skeleton supervisor
    The part of the supervisor that is always in core. The skeleton supervisor processes CALL DUMP, CALL EXIT, and CALL LINK statements. Certain error traps are also considered part of the skeleton supervisor.
  entry points to the 3-2
  ILS02 and ILS04 3-2
  interrupt level 2 3-2
  interrupt level 4 3-2
  postoperative error traps 3-2
  preoperative error trap 3-2
  PROGRAM STOP key error trap 3-2
  resident monitor 3-2
  $DUMP entry, use by CIL 3-13
  $EXIT entry, use by CIL 3-13
  $LINK entry, use by CIL 3-13
SLET (system location equivalence table) 2-6
  dumping 4-7
  listing E-1
SOCAL options 3-12
SOCAL overlays, subroutines included in 3-12
SOCALs
  core load builder provision for 3-11
  using 6-13
special ILSs, how to specify 5-8
stacked job input arrangement 6-1
stacked job input example 6-3
stand-alone paper tape utility program (*see also* PTUTL) 9-46
stand-alone utility programs 9-1
  console printer core dump 9-1
  disk cartridge initialization 9-8
  paper tape reproducing 9-42
  printer core dump 9-4
starting address of RJE user-exit data 10-8
starting keyboard operation (// TYP) 7-12
starting up the RJE work station 10-9
stopping DCIP operating procedures 9-10
stopping keyboard operation (// TEND) 7-13
storage, components of disk 2-2
storage area (core image header), restriction on use by FORTRAN subroutines 3-9
storage unit conversion factors, appendix L, disk L-1
STORE DUP control record
  additional field information 5-32
  examples 5-32
  format 5-31
  storing information in WS or UA 5-30
STORECI DUP control record
  additional field information 5-40
  examples 5-41
  format 5-39
  storing a program in DCI format 5-38
stored programs, size discrepancies in 6-7
STOREDATA DUP control record
  calculating card count after operations other than DUMPDATA 6-8
STOREDATA DUP control record (continued)
  examples 5-34
  format 5-33
  storing information in WS, UA, or FX 5-33
STOREDATACI DUP control record
  examples 5-38
  format 5-37
  storing information in WS, UA, or FX 5-37
STOREDATAE DUP control record
  additional field information 5-36
  examples 5-36
  format 5-35
  storing information in WS, UA, or FX 5-34
STOREMOD DUP control record
  examples 5-43
  format 5-42
  replacing stored information 5-42
storing a program in DCI format
  *STORECI 5-38
  disadvantages of 6-7
storing information in WS or UA, *STORE 5-30
storing information in WS, UA, or FX
  *STOREDATA 5-33
  *STOREDATACI 5-37
  *STOREDATAE 5-34
SUB MODIF patch control record
  additional field information 4-13
  description 4-12
  format 4-13
subjob
    A monitor operation performed during a job. Each subjob is initiated by a monitor control record such as ASM or XEQ. A subjob can also be initiated by a CALL LINK statement.
subprogram
    A synonym used mainly in FORTRAN for both FUNCTIONs and SUBROUTINEs.
subprogram header card, CDS program 1-7
subroutine
    A subset of the set program. In FORTRAN, a SUBROUTINE is a type of subprogram that is not restricted to a single value for the result and is called with a CALL statement.
  disk I/O in the resident monitor 3-3
  how to specify disk I/O 5-8
  RJE user-exit 10-8
subroutine errors
  console printer B-8
  I/O device B-5
  paper tape B-9
  1442 card B-5
  2501 card B-8
subroutine types and subtypes I-1
subroutines
  adding to system library 4-12
  assembler INT REQ service 6-45
  assembler program use of 1403 conversion 6-37
  deleting I/O device 4-3
  equatable FORTRAN I/O 6-48
  equating during core load construction 3-11
  example of an INT REQ service for any core load 6-46
  example of an INT REQ service for core load using TYPEZ, WRTYZ, TYPE0, or WRTY0 6-47
  included in IBM system area 2-7
  included in SOCAL overlays 3-12
  incorporating during core load construction 3-10
  ISS in system library 4-2
  ISS preoperative error waits B-2
  listing of ISS error waits B-3

subroutines (continued)
nameN ISS 4-3
nameZ ISS 4-2
name0 ISS 4-2
name1 ISS 4-2
restriction on use of core image header storage area by FORTRAN 3-9
rules for coding an INT REQ service 6-45
rules for writing ILS 6-42
rules for writing ISS 6-37
substituting during core load construction 3-11
system library utility 4-4
types and subtypes I-1
using the disk I/O 6-4
substituting subroutines during core load construction 3-11
summary of the contents of disk cartridges 2-12
supervisor
auxiliary 3-3
disk-resident programs 3-3
error messages A-36
messages A-35
monitor system programs 3-2
resident monitor 3-2
skeleton 3-2
supervisor control record analyzer, disk-resident supervisor programs 3-3
supervisor control record area (SCRA) 2-6
The disk cylinder in which the supervisor control records are written. Sectors zero and one are reserved for LOCAL control records, sectors 2 and 3 are reserved for NOCAL control records, 4 and 5 for FILES control records, 6 is reserved for G2250 information and 7 is reserved for EQUAT information.
(*see also* SCRA)
supervisor control records 5-12
*EQUAT 5-17
*FILES 5-15
*G2250 5-16
*LOCAL 5-13
*NOCAL 5-14
coding of 5-12
functions of 5-12
maximum number 5-12
where placed in input stream 5-12
supervisor core dump program, disk-resident supervisor program 3-3
switched lines, communication considerations for RJE 10-2
switches, RJE console entry 10-12
symbol table, listing a FORTRAN 5-67
symbol table, saving on disk the assembler 5-60
symbol table overflow, specifying WS sectors for assembler 5-61
symbol tables merging assembler 5-60
symbolic address format used in this publication 1-2
system
operating the 1130 disk monitor 7-1
using the 1130 with the monitor 7-12
system area on a system cartridge, IBM 2-5
system cartridge 2-4
A cartridge that contains the monitor programs. A system cartridge may be used as either a master or a satellite cartridge.
cylinder 0 on a 2-4
description 2-4
IBM system area on a 2-5
resident image in sector @RIAD on a 2-5
sector @DCOM on a 2-5
sector @HDNG on a 2-5
sector @RTBL on a 2-5
user area on a 2-8
system configuration, minimum iii
system configuration control record (system loader)
for card system 8-9
for paper tape system 8-9
format 8-9
system core image buffer
The core image buffer used by the monitor system programs during a job. System CIB need not be on the master cartridge. The JOB monitor control record defines the cartridge that contains the CIB to be used for the job.
system configuration control tape preparation of 8-10
system control, functions of console opertor keys during monitor 7-13
system device subroutine area (*see also* SCRA) 2-6
system initial load, monitor 8-1
system library
adding subroutines 4-12
ISS subroutines in 4-2
mainline programs 4-5
mainline programs error messages for monitor A-59
mainline programs messages for monitor A-59
monitor 4-1
utility subroutines 4-4
system library listing, appendix C, monitor C-1
common arithmetic/function LIBFs C-5
common FORTRAN CALLs C-1
common plot CALL C-7
common plot LIBFs C-8
conversion subroutines C-6
conversion tables C-6
disk FORTRAN I/O C-3
disk I/O C-8
disk maintenance programs C-1
extended arithmetic/function CALLs C-2
extended arithmetic/function LIBFs C-4
extended plot CALLs C-7
extended plot LIBFs C-8
flipper for LOCAL/SOCAL subprograms C-3
FORTRAN common LIBFs C-3
FORTRAN find subroutines C-3
FORTRAN I/O and conversion subroutines C-4
FORTRAN sign transfer CALLs C-2
FORTRAN trace subroutines C-3
interrupt level subroutines C-7
interrupt service subroutines C-5
log subroutine C-6
mainlines C-1
nondisk FORTRAN format I/O C-3
paper tape utility program C-1
RPG compare subroutine C-8
RPG decimal arithmetic subroutines C-8
RPG indicators C-9
RPG miscellaneous subroutines C-9
RPG move subroutines C-8
RPG sterling and edit subroutines C-8
special interrupt level subroutines C-7
standard arithmetic/function CALLs C-2
standard arithmetic/function LIBFs C-4
standard plot CALLs C-7
standard plot LIBFs C-7
subroutines C-1
unformatted FORTRAN disk I/O C-3
utility call subroutines C-1
ZIPCO conversion tables C-6
system library mainline programs 4-5
disk maintenance programs 4-5
PTUTL 4-25

system library maintenance
MODIF disk maintenance program 4-8
MODSF disk maintenance program 4-14
system library utility subroutines 4-4
CALPR 4-4
FLIPR 4-4
FSLEN 4-4
RDREC 4-4
SYSUP 4-4
system loader
error messages A-22
messages A-22
sector break cards used by I-10
system loader control records
IBM supplied 8-2
that you punch 8-7
system location equivalence table (*see* SLET)
system maintenance MODIF disk maintenance program 4-8
system-overlay-to-be-loaded-on-call (SOCAL) 3-11
One of 2 or 3 overlays the core load builder automatically prepares, under certain conditions, when a core load is too large to fit into core storage.
system program sector break cards, IBM supplied system loader 8-4
system programs, monitor 3-1
system reload, monitor 8-1
SYSTEM SYMBOL TABLE assembler control record
format 5-60
merging symbol tables 5-60
system working storage
The working storage area used by the monitor programs during a job. System working storage need not be on the master cartridge. The JOB monitor control record defines the cartridge that contains working storage to be used for a job.
SYSUP
assembler calling sequence 6-20
FORTRAN calling sequence 6-21
how to use reeling 6-22
how to use when changing cartridges during a job 6-20
sample program for one drive systems that uses H-13
sample program for multidrive systems that uses H-16
system library utility subroutine 4-4
tips on monitor control and usage 6-20
update error messages A-37
SYSUP-DCOM update error messages A-37

techniques and tips, programming 6-1
temporary job, how to specify 5-4
temporary job mode, how to use 4-6
temporary jobs, restrictions on 5-4
temporary mode, restrictions on DUP caused by temporary mode 5-22
TEND monitor control record reassigns card or paper tape reader as principal input device 5-10
TERM control record, IBM-supplied system loader 8-2
tips and techniques, programming 6-1
tips for assembler language programmers
assembler INT REQ service subroutine 6-45
assembler program use of index register 3 6-35
assembler program use of 1403 conversion subroutines 6-37
grouping of assembler mnemonics 6-35
programming tips and techniques 6-35
writing ISSs and ILSs 6-37
tips for FORTRAN programmers 6-48
// records read during FORTRAN program execution 6-51
dumping FORTRAN DSF programs to cards 6-52
tips for FORTRAN programmers (continued)
FORTRAN I/O errors 6-51
FORTRAN object program paper tape data record format 6-49
FORTRAN program control of the console printer 6-50
invalid characters in FORTRAN source cards 6-49
keyboard input of data records during FORTRAN program execution 6-50
length of FORTRAN DATA statement 6-51
tips for use of the EQUAT control record 6-48
tips on monitor control and usage
disadvantages of storing a program in DCI format 6-7
duplicate program and data file names 6-6
how to avoid overprinting when using // CPRNT 6-5
how to avoid overprinting when linking between programs 6-5
how to use temporary job mode 6-6
LOCAL and NOCAL control record usage 6-10
LOCAL-calls-a-LOCAL 6-10
locating FORTRAN allocation addresses 6-18
mainline programs that use all of core 6-9
programming tips and techniques 6-1
reading a core map and a file map 6-13
reading the transfer vector 6-19
restoring destroyed cartridges 6-5
size discrepancies in stored programs 6-7
stacked job input arrangement 6-1
SYSUP 6-20
the use of LOCALs 6-9
the use of NOCALs 6-11
the use of SOCALs 6-13
usage of the EJECT monitor control record 6-5
use of defined files 6-9
using the disk I/O subroutines 6-4
tracing branch statement execution sequence, *TRANSFER TRACE FORTRAN control record 5-70
tracing variable values during execution, *ARITHMETIC TRACE FORTRAN control record 5-70
track, description of 2-1
transfer and conversion operations, summary of DUP 5-21
TRANSFER TRACE FORTRAN control record, tracing branch statement execution sequence 5-70
transfer tracing, how to stop 5-70
transfer vector (TV)
A collection of both the LIBF TV and the CALL TV.
processing during core load construction 3-13
reading 6-19
transmit mode, RJE 10-2
two pass mode, how to use 5-52
TWO PASS MODE assembler control record 5-52
TYP monitor control record, assigns console keyboard as principal input device 5-10
type 81 control record, IBM-supplied system loader 8-7

UA (*see* user area)
unformatted disk file 6-26
The organization by FORTRAN of a disk data file to simulate processing of a magnetic tape file with variable length records. Data conversion is not possible.
unformatted disk I/O, how to specify 5-5
unformatted file records, how FORTRAN writes in sectors 6-26
unformatted FORTRAN I/O statements 6-26
unformatted I/O
FORTRAN sample program using H-12
initializing $$$$$ data files for use with FORTRAN 6-27
usage of the EJECT monitor control record 6-5
use of defined files 6-9
use of 1403 conversion subroutines, assembler program 6-37

user area (UA)
The area on disk in which all of your programs in disk system format and all IBM-supplied programs are stored. Core image programs and data files can also be stored in this area. The user area occupies as many sectors as are required to contain the programs and files stored in it.
deleting information from *DELETE 5-44
packing of 2-11
storing information in
*STORE 5-30
*STOREDATA 5-33
*STOREDATACI 5-37
*STOREDATAE 5-34
on a system cartridge 2-9
user area and working storage
during delete operations 2-11
during store operations 2-10
user-exit data
control character types for RJE 10-8
end-of-data indicator for RJE 10-9
ending address of RJE 10-8
logical record length of RJE 10-8
record format of RJE 10-9
starting address of RJE 10-8
user-exit parameter list, RJE 10-8
user-exit subroutine, RJE 10-8
user programs 2-14
Mainlines, subroutines, or core loads that you have written and stored in the user or fixed area.
utility programs, stand-alone 9-1
utility subroutines, system library 4-4

variable values, tracing during FORTRAN execution 5-70

wait codes
card core image loader B-9
FORTRAN I/O B-10
monitor system error B-1
PTUTL error B-9
RPG object program B-12 and 13
waits
cold start program error B-1
ISS subroutine preoperative error B-2
listing of ISS subroutine error B-3
words, number in a sector 2-2
work station
JECL for the 1130 RJE 10-5
output to the RJE 10-6
RJE card input at the 10-2
RJE disk input at the 10-2
RJE input at the 10-2
RJE keyboard input at the 10-2
work station startup, RJE 10-9
work stations, messages sent to RJE 10-12
working storage (WS) 2-11
The area on disk immediately following the last sector occupied by the user area. This is the only one of three major divisions of disk storage (IBM system area, user/fixed area, working storage) that does not begin at a cylinder boundary.
working storage (WS) (continued)
contents during core load construction 3-8
core load builder use of 3-8
specifying for current job 5-4
specifying sectors for assembler symbol table overflow 5-61
specifying sectors for macro overflow 5-61
working storage and user area
during delete operations 2-11
during store operations 2-10
writing ISSs and ILSs 6-37
writing sector addresses in WS
*DWADR DUP control record 5-47
ADRWS disk maintenance program 4-8
WS (*see* working storage)

X-field type, DFCNV J-7
XEQ monitor control record
additional field information 5-7
examples 5-9
format 5-7
general function 5-7
XREF assembler control record
format 5-56
printing an assembler listing with statement numbers 5-57
printing an assembler listing with a cross reference symbol table 5-57
XR3, assembler program use of 6-35

1DUMY entry format, LET/FLET D-1
1055 Paper Tape Punch, readying the 7-7
1130 Disk Monitor System, operating the 7-1
1130 publications, related iii
1130 RJE work station program generation of the 10-3
1130 work station, JECL for the RJE 10-5
1131 Central Processing Unit, readying the 7-2
1132 Printer, readying the 7-4
1134 Paper Tape Reader, readying the 7-6
1231 Optical Mark Page Reader, readying the 7-9
1316 Disk Pack 2-1.1
1403 conversion subroutines, assembler program use of 6-37
1403 Printer, readying the 7-4
1442 card read errors B-6
1442 card subroutine errors B-5
1442 Model 5 Card Punch, readying the 7-5
1442 Model 6 and 7 Card Read Punch, readying the 7-5
1627 Plotter
omitting subroutines 8-7
readying the 7-8
2310 Disk Storage Drive, readying the 7-3
2311 Disk Storage Drive, readying the 7-3.1
2315 Disk Cartridge 2-1.1
2501 Card Reader, readying the 7-6
2501 card subroutine errors
feed check error B-8
read check error B-8

IBM 1130 Disk Monitor System, Version 2,
Programmer's and Operator's Guide

GC26-3717-9

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ______________________________

Number of latest Technical Newsletter (if any) concerning this publication: ____________

Please indicate your name and address in the space below if you wish a reply.

____

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC26-3717-9

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold Fold

IBM

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

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

Cut Along Line

IBM 1130 Disk Monitor System, Version 2, Programmer's and Operator's Guide (1130-36) Printed in U.S.A. GC26-3717-9

GC26-3717-9

IBM 1130 Disk Monitor System, Version 2, Programmer's and Operator's Guide (1130-36) Printed in U.S.A. GC26-3717-9

IBM®

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

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