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

Program Numbers: 1130-0S-005
1130-0S-006
Eleventh Edition (June 1974)

This is a reprint of GC26-3717-9 incorporating changes released in Technical Newsletter GN34-0183 dated February 1974.

This edition applies to version 2, modification 12, 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 control number and is not related to the subject matter.

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

Address comments concerning the contents of this publication to IBM Corporation, General Systems Division, Department 27T, P.O. Box 1328, Boca Raton, Florida 33432. Comments become the property of IBM.

Preface

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 1130-0S-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 2501 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 1130-0S-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 JOCS 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: 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

Preface  iii
Chapter 6. Programming Tips and Techniques

Tips on Monitor Control and Usage

Chapter 7. Operating the 1130 Disk Monitor System

Readying the 131 Optical Mark Page Reader

Tips for Assembly Language Programmers

Tips for FORTRAN Programmers

Tips for Use of the EQUAT Control Record

Tips for Use of the EJECT Control Record

RPG Program Considerations

Chapter 8. Monitor System Initial Load and System Reload

IBM-Supplied System Loader Control Records

Chapter 9. Stand-alone Utility Programs

Stand-alone Tape Utility Program (PTUTL)

Chapter 10. Remote Job Entry Program

Remote Job Entry Program

Machine and Device Requirements

Communication Considerations

Error Statistics

Wallboard Error Recovery Procedures

User-Exit Subroutine

Displaying or Altering the Contents of a Selected Core Location

Cold Start Procedure

Card System Cold Start Procedure

Paper Tape System Cold Start Procedure

Using the 1130 with the Monitor System

Entering Jobs from the Card Reader

Entering Jobs from the Paper Tape Reader

Entering Jobs from the Console Keyboard

Functions of Console Operator Keys During Monitor System Control

Displaying or Altering the Contents of a Selected Core Location

Manual Dump of Core Storage

Contents
# Appendix A. Monitor System Operational and Error Messages

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler Error Codes and Messages</td>
<td>A-1</td>
</tr>
<tr>
<td>FORTRAN Messages and Error Codes</td>
<td>A-2</td>
</tr>
<tr>
<td>DUP and MUP Messages and Error Messages</td>
<td>A-7</td>
</tr>
<tr>
<td>System Loader Messages and Error Messages</td>
<td>A-13</td>
</tr>
<tr>
<td>Satellite Graphic Job Processor Error Messages</td>
<td>A-22</td>
</tr>
<tr>
<td>RJE Messages and Error Messages</td>
<td>A-26</td>
</tr>
<tr>
<td>Supervisor Messages and Error Messages</td>
<td>A-27</td>
</tr>
<tr>
<td>DUP and MUP Messages and Error Messages</td>
<td>A-35</td>
</tr>
<tr>
<td>Core Load Builder Messages</td>
<td>A-38</td>
</tr>
<tr>
<td>Auxiliary Supervisor Error Messages</td>
<td>A-54</td>
</tr>
<tr>
<td>Monitor System Library Mainline Programs Messages and Error Messages</td>
<td>A-58</td>
</tr>
<tr>
<td>IDENT Messages</td>
<td>A-59</td>
</tr>
<tr>
<td>DISC Messages and Error Messages</td>
<td>A-59</td>
</tr>
<tr>
<td>ID Messages and Error Messages</td>
<td>A-60</td>
</tr>
<tr>
<td>COPY Messages and Error Messages</td>
<td>A-60</td>
</tr>
<tr>
<td>DLCIB Messages and Error Messages</td>
<td>A-61</td>
</tr>
<tr>
<td>MODIF Messages and Error Messages</td>
<td>A-61</td>
</tr>
<tr>
<td>MODSF Messages and Error Messages</td>
<td>A-63</td>
</tr>
<tr>
<td>DFCNV Messages and Error Messages</td>
<td>A-67</td>
</tr>
</tbody>
</table>

# Appendix B. Monitor System Error Wait Codes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold Start Program Error Waits</td>
<td>B-1</td>
</tr>
<tr>
<td>ISS Subroutine Preoperative Error Waits</td>
<td>B-2</td>
</tr>
<tr>
<td>I/O Device Subroutine Errors</td>
<td>B-5</td>
</tr>
<tr>
<td>1442 Card Subroutine Errors</td>
<td>B-5</td>
</tr>
<tr>
<td>2501 Card Subroutine Errors</td>
<td>B-8</td>
</tr>
<tr>
<td>Console Printer Subroutine Errors</td>
<td>B-8</td>
</tr>
<tr>
<td>Paper Tape Subroutine Errors</td>
<td>B-9</td>
</tr>
<tr>
<td>Paper Tape Utility Program (PTUTL) Error Wait Codes</td>
<td>B-9</td>
</tr>
<tr>
<td>FORTRAN I/O Wait Codes</td>
<td>B-10</td>
</tr>
<tr>
<td>RPG Object Program Wait Codes</td>
<td>B-12</td>
</tr>
</tbody>
</table>

# Appendix C. Monitor System Library Listing

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LET/FLET Disk Format</td>
<td>C-1</td>
</tr>
<tr>
<td>LET/FLET Dump Format</td>
<td>C-1</td>
</tr>
</tbody>
</table>

# Appendix D. LET/FLET

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LET/FLET Disk Format</td>
<td>D-1</td>
</tr>
<tr>
<td>LET/FLET Dump Format</td>
<td>D-1</td>
</tr>
<tr>
<td>Appendix E. System Location Equivalence Table (SLET)</td>
<td>D-2</td>
</tr>
</tbody>
</table>

# Appendix F. Core Dump

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

# Appendix G. Resident Monitor (Including Table of Equivalences)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

# Appendix H. Monitor System Sample Programs

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN Sample Program Run on 4K</td>
<td>H-1</td>
</tr>
<tr>
<td>FORTRAN Sample Program Run on 8K</td>
<td>H-2</td>
</tr>
<tr>
<td>FORTRAN Sample Program Run on 16K</td>
<td>H-5</td>
</tr>
<tr>
<td>FORTRAN Sample Program Run on 32K</td>
<td>H-7</td>
</tr>
<tr>
<td>FORTRAN Sample Program Run on 64K</td>
<td>H-9</td>
</tr>
</tbody>
</table>

# Appendix I. Formats

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Formats</td>
<td>I-1</td>
</tr>
<tr>
<td>Card Formats</td>
<td>I-2</td>
</tr>
<tr>
<td>Paper Tape Formats</td>
<td>I-6</td>
</tr>
<tr>
<td>Print Formats</td>
<td>I-11</td>
</tr>
<tr>
<td>Data Formats</td>
<td>I-13</td>
</tr>
</tbody>
</table>

# Appendix J. Field Type Examples for DFCNV

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-Field Type</td>
<td>J-1</td>
</tr>
<tr>
<td>J-Field Type</td>
<td>J-1</td>
</tr>
<tr>
<td>R-Field Type</td>
<td>J-2</td>
</tr>
<tr>
<td>B-Field Type</td>
<td>J-4</td>
</tr>
<tr>
<td>C-Field Type</td>
<td>J-4</td>
</tr>
<tr>
<td>D-Field Type</td>
<td>J-5</td>
</tr>
<tr>
<td>E-Field Type</td>
<td>J-6</td>
</tr>
<tr>
<td>F-Field Type</td>
<td>J-6</td>
</tr>
<tr>
<td>X-Field Type</td>
<td>J-7</td>
</tr>
</tbody>
</table>

# Appendix K. Decimal and Hexadecimal Disk Addresses

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

# Appendix L. Disk Storage Unit Conversion Factors

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

# Appendix M. Character Code Set

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

# Glossary-Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>
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:

<table>
<thead>
<tr>
<th>Physical drive number</th>
<th>1131 CPU</th>
<th>2310 Disk Storage or 2311 Disk Storage Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal disk</td>
<td>First 2310, first disk</td>
</tr>
<tr>
<td>1</td>
<td>First 2310, second disk</td>
<td>First 2311, second disk</td>
</tr>
<tr>
<td>2</td>
<td>Second 2310, first disk</td>
<td>First 2311, third disk</td>
</tr>
<tr>
<td>3</td>
<td>Second 2310, second disk</td>
<td>First 2311, fourth disk</td>
</tr>
<tr>
<td>4</td>
<td>Second 2310, third disk</td>
<td>First 2311, fifth disk</td>
</tr>
<tr>
<td>5</td>
<td>Second 2310, fourth disk</td>
<td>Second 2311, first disk</td>
</tr>
<tr>
<td>6</td>
<td>Second 2310, fifth disk</td>
<td>Second 2311, second disk</td>
</tr>
<tr>
<td>7</td>
<td>Second 2311, third disk</td>
<td>Second 2311, third disk</td>
</tr>
<tr>
<td>8</td>
<td>Second 2311, fourth disk</td>
<td>Second 2311, fourth disk</td>
</tr>
<tr>
<td>9</td>
<td>Second 2311, fifth disk</td>
<td>Second 2311, fifth disk</td>
</tr>
</tbody>
</table>

*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.

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.

<table>
<thead>
<tr>
<th>No. of</th>
<th>Word</th>
<th>Disk block</th>
<th>Sector</th>
<th>Track</th>
<th>Cylinder</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>16</td>
<td>320</td>
<td>5,112</td>
<td>20,480</td>
<td>40,960</td>
<td>8,192,000</td>
</tr>
<tr>
<td>Data words</td>
<td>20</td>
<td>320</td>
<td>1,280</td>
<td>2,560</td>
<td>512,000</td>
<td></td>
</tr>
<tr>
<td>Disk blocks</td>
<td>16</td>
<td>64</td>
<td>128</td>
<td>25,600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sectors</td>
<td>4</td>
<td>8</td>
<td>1,600</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracks</td>
<td>2</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinders</td>
<td></td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
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:

<table>
<thead>
<tr>
<th>Sector</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>@ IDAD</td>
</tr>
<tr>
<td>1</td>
<td>@ DCOM</td>
</tr>
<tr>
<td>2</td>
<td>@ RIAD</td>
</tr>
<tr>
<td>3, 4, 5</td>
<td>@ SLET</td>
</tr>
<tr>
<td>6</td>
<td>@ RTBL</td>
</tr>
<tr>
<td>7</td>
<td>@ HDNG</td>
</tr>
</tbody>
</table>

Sector | 0 | 1 | 2 | 3, 4, 5 | 6 | 7 |

ID and cold start prog. | DCOM | Resident image | SLET | Reload table | Page heading |

---

2-4
The following is a discussion of each sector.

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 cartridge 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 G. 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 /FFFE.

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:

- DUP FOR COS SUP CLB
- System device subroutines, DISK1, DISK1, DISKZ CIL RPG DUP part 2 ASM Cushion area SCRA FLET
- IBM system area

1 Program product
2 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.

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
  DISKI
  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.

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.

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

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 beginning 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:

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.
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:

```
Cyl 0  Fixed area  User area  Working storage
```

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:

```
Cyl 0
```

```
ID and error message program  DCOM  LET
Sector 0 1 2-7
```
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 Diagram]

- **DCYL**: (defective cylinder table)
- **CI DN**: (cartridge ID)
- **COPY**: (copy code)
- **Reserved**
- **DTYP**: (disk type)
- **Reserved**
- **Error message and error message program**

Words: 0 1 2 3 4 5 6 7 269 270 319

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:

![IBM System Area Diagram]

- **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.

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

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.

![Diagram of Monitor System Programs]

Program product

IBM system area
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.

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.

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 (ILSO2 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.
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.

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.

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.

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.

The *Supervisor Core Dump Program* provides a hexadecimal printout and an EBCDIC translation 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.
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.
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.
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.

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:

<table>
<thead>
<tr>
<th>That part of core load below 4096 (or 5056)</th>
<th>Not used</th>
<th>COMMON saved from last core load</th>
</tr>
</thead>
</table>

Core image buffer
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:

![Working storage diagram]

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:

![Core storage diagram]

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:

![Core load layout diagram]
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.

<table>
<thead>
<tr>
<th>Disk I/O subroutine in core</th>
<th>Core load origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKZ</td>
<td>610 /01FE</td>
</tr>
<tr>
<td>DISK1</td>
<td>690 /0282</td>
</tr>
<tr>
<td>DISKN</td>
<td>980 /03C0</td>
</tr>
</tbody>
</table>
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. SOCA Ls 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, SOCA Ls are created. An area as large as the largest SOCA L overlay (usually SOCA L 2) is reserved in the core load. SOCA L overlays will be read by flipper into this area as required during execution. The SOCA L overlays are placed in working storage following any data files and LOCALs stored there. If the core load is executed immediately, each SOCA L overlay is read, as it is called, from working storage into the SOCA L area by flipper. If the core load is stored in disk core image format before it is executed, SOCA Ls are stored following the core load and any LOCALs. SOCA Ls 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:

<table>
<thead>
<tr>
<th>SOCAL overlay</th>
<th>Subroutine class</th>
<th>Type</th>
<th>Subtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Arithmetic</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Function</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Nondisk FORTRAN I/O and “Z” conversion subroutines</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>“Z” device subroutines</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Disk FORTRAN I/O</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

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 SOCA Ls” 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 SOCALLs are required, the CALL TV entries for function subroutines contain the address of the special SOCALL 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 SOCALLs are required, the LIBF TV entry for a SOCALL subroutine contains a branch to a special entry in the LIBF TV for the SOCALL of which the subroutine is a part. This special entry provides the linkage to the desired SOCALL.

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.

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.

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).
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.
Cold start record

Cold start program (sets negative parameter for DUMP entry)

DUMP entry

EXIT entry

Skeleton supervisor

Core image loader determines where skeleton supervisor was entered

EXIT entry

LINK entry, DUMP entry

Monitor control record analyzer

Monitor control record analyzer

EXIT entry

DUMP entry, DSF program

Auxiliary supervisor

DUMP entry, positive parameter

LINK entry, DCI program

USER EXECUTION

Subroutine library

Terminal dump

Dynamic dump

Core load builder

DCI program

LINK DCI program

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

LINK DCI program

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT

EXIT
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.

<table>
<thead>
<tr>
<th>I/O device</th>
<th>I/O device subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1442 Card Read Punch</td>
<td>CARDF, CARDO, or CARD1</td>
</tr>
<tr>
<td>2501 Card Reader</td>
<td>READZ, READ0, or READ1</td>
</tr>
<tr>
<td>1442 Card Punch</td>
<td>PNCHZ, PNCH0, or PNCH1</td>
</tr>
<tr>
<td>Disk</td>
<td>DISKZ, DISK1, or DISKN</td>
</tr>
<tr>
<td>1132 Printer</td>
<td>PRNTZ, PRNT1, PRNT2</td>
</tr>
<tr>
<td>1403 Printer</td>
<td>PRNZ, or PRNT3</td>
</tr>
<tr>
<td>Console keyboard/printer</td>
<td>TYPEZ, or TYPEO</td>
</tr>
<tr>
<td>Console printer</td>
<td>WRTYZ, or WRTYO</td>
</tr>
<tr>
<td>1134/1055 Paper Tape Reader Punch</td>
<td>PAPTZ, PAPT1, PAPT1, or PAPT1</td>
</tr>
<tr>
<td>1627 Plotter</td>
<td>PLOT1, or PLOTX</td>
</tr>
<tr>
<td>1231 Optical Mark Page Reader</td>
<td>OMPR1</td>
</tr>
<tr>
<td>Synchronous Communications Adapter</td>
<td>SCAT1, SCAT2, or SCAT3</td>
</tr>
</tbody>
</table>

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.
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:

<table>
<thead>
<tr>
<th>Device not in configuration</th>
<th>I/O device subroutines that can be deleted</th>
<th>Disk blocks gained (hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1442 Card Read Punch (input/output)</td>
<td>CARD0, CARD1, CARDZ</td>
<td>/4E</td>
</tr>
<tr>
<td>2501 Card Reader</td>
<td>READ0, READ1, READZ</td>
<td>/62</td>
</tr>
<tr>
<td>1442 Card Punch</td>
<td>PNCHO, PNCH1, PNCHZ</td>
<td>/22</td>
</tr>
<tr>
<td>1134/1055 Paper Tape Reader/Punch</td>
<td>PAPT1, PAPTN, PAPTZ, PAPEB, PAPPR, PAPHL</td>
<td>/75</td>
</tr>
<tr>
<td>1132 Printer</td>
<td>PRNT1, PRNT2, PRTZ2, PRNTZ, DMD01</td>
<td>/69</td>
</tr>
<tr>
<td>1403 Printer</td>
<td>PRNT3, PRNZ, EBP3, CPPT3, HLPT3, PT3E8, PT3CP, PT3HOL</td>
<td>/40</td>
</tr>
<tr>
<td>1627 Plotter</td>
<td>PLOT1, PLOTI, PLOTX, FCHRX, ECHRX, SCALF, SCALE, FGRID, EGRID, FCHAR, ECHAR, FPLT, EPLT, FRULE, ERULE, POINT, XYPLT</td>
<td>/80</td>
</tr>
<tr>
<td>Synchronous Communications Adapter</td>
<td>SCAT1, SCAT2, SCAT3, PRNT2, PRTZ2, IOLOG, EBC48, HOL48, HXCV, STRTB, HOLCA</td>
<td>/FA</td>
</tr>
<tr>
<td>1231 Optical Mark Page Reader</td>
<td>OMPR1</td>
<td>/15</td>
</tr>
<tr>
<td>MTCA</td>
<td>MTCA0, MTCAZ, TSM41, TSTTY, FEB41</td>
<td>/9A</td>
</tr>
</tbody>
</table>
Utility Subroutines

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:

<table>
<thead>
<tr>
<th>Device not in configuration</th>
<th>Mainline programs that can be deleted</th>
<th>Disk blocks gained (hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1134/1055 Paper Tape Reader/Punch</td>
<td>PTUTL</td>
<td>/OA</td>
</tr>
<tr>
<td>2310 Disk Storage or 2311 Disk Storage Drive</td>
<td>DLCIB, ID, COPY, DISC, IDENT</td>
<td>/9D</td>
</tr>
</tbody>
</table>

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.
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.

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.

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 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:

*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:

```plaintext
/ / 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:

```plaintext
/ / XEQ  ID
```

*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:

```plaintext
/ / XEQ  COPY
```
Disk Maintenance Programs

| ADRWS | DLCIB | MODIF |

*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
1/ XEQ DLCIB
1/ 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

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:

```
// ...
// DEND
```

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.
The table below describes the MODIF control record format for disk maintenance programs:

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>*MON</td>
<td>These characters identify a patch to any of the monitor system programs and/or the system device subroutines.</td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td>A hexadecimal number; v is the monitor version, and mm is the monitor modification level.</td>
</tr>
<tr>
<td>6 through 8</td>
<td>vmm</td>
<td>0 indicates system modification update. G indicates general temporary fix. R indicates restricted temporary fix.</td>
</tr>
<tr>
<td>9</td>
<td>0 or G or R</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Blank</td>
<td>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).</td>
</tr>
<tr>
<td>11 through 14</td>
<td>xxxx</td>
<td>The numbers (in hexadecimal) of patch data records that follow this control record.</td>
</tr>
<tr>
<td>15</td>
<td>Blank</td>
<td>This character identifies the format of the patch data records that follow. B indicates binary system format. H indicates hexadecimal patch format.</td>
</tr>
<tr>
<td>20</td>
<td>Blank</td>
<td>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.</td>
</tr>
<tr>
<td>23 through 36</td>
<td>pppp</td>
<td>A hexadecimal number; d is the disk drive code, and ass is the sector address of the program being patched. Use this field only when columns 11 through 14 contain 0000.</td>
</tr>
</tbody>
</table>
Disk Maintenance Programs

MODIF data records

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>33 through 36</td>
<td>cccc</td>
<td>A hexadecimal number that specifies the absolute core address of the first word of the sector specified in columns 28 through 31. Use this field only when columns 11 through 14 contain 0000.</td>
</tr>
<tr>
<td>37 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*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 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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>aaaa</td>
<td>The beginning core address (in hexadecimal) of the patch. Each patch data record must contain the core address.</td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>6 through 9, 11 through 14, 16 through 19, ...</td>
<td>Blank</td>
<td>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.</td>
</tr>
<tr>
<td>66 through 69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70 through 72</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

Additional field information

Monitor System Library 4-11
Hexadecimal patch records can contain ID/sequence numbers in columns 73 through 80. Zeros must be punched; leading blanks are not assumed.

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

*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**

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

**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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 7</td>
<td>/&amp;DEND</td>
<td>&amp; indicates blank.</td>
</tr>
<tr>
<td>8 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
MODIF Example

This example illustrates how to change an instruction in the Disk Utility Program (DUP). The following data is used to make the change:

- The SLET phase ID of the subroutine is 0009.
- Hexadecimal patch format is used.
- The instruction address (from an assembly listing) is 03B6.
- The instruction is D7F0.
- The instruction is to be changed to D7D6.
- The new modification level is 12.
- 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:

```
/ /  /09
/ / XEQ MODIF
*MON D7F0 0009 0001 H 001
03B6 D7D6
// DEND
```

The following is printed on the console printer when the example is executed:

MODIF EXECUTION 020B

```
MON 20C0 0009 0001 H 001
DAAA REL—WD ADDR OLD NEW
002B 0096 03B6 D7F0 D7D6
SW 0 OFF=PATCH
SW 0 ON =ABORT
```

MODIF COMPLETED 020C

Where:

- MODIF EXECUTION 020B: Execution of MODIF starts on DM2, Version 11.
- DAAA: Drive code and sector address of the patch.
- REL—WD: Relative word within the sector that is to be patched.
- ADDR: Instruction address (from an assembly listing).
- OLD: Original instruction.
- NEW: New instruction.
- SW 0 OFF=PATCH: The system waits after these 2 lines are printed for operator intervention. Set data entry switch 0 to OFF and press PROGRAM START to write the patch to disk or set data entry switch 0 to ON to prevent the patch from being made.

Note. To prevent the printing of patch information, set data entry switch 1 to ON.

MODIF COMPLETED 020C: The patch is installed, and the new level is 12.

To direct printout to principal printer, set data entry switch 2 to ON.
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.

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.
### MODSF control records

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>*PRO</td>
<td>These characters identify a MODSF patch control record.</td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td>A hexadecimal number; v is the current monitor version, and mm is the current monitor modification level.</td>
</tr>
<tr>
<td>6 through 8</td>
<td>vmm</td>
<td>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.)</td>
</tr>
<tr>
<td>16 through 19</td>
<td>nnnn</td>
<td>The number (in hexadecimal) of patch data records that follow this control record.</td>
</tr>
<tr>
<td>21</td>
<td>m</td>
<td>Indicates addressing mode, where m is: P for program-address mode, or D for disk-displacement mode.</td>
</tr>
<tr>
<td>23 through 26</td>
<td>xxxx</td>
<td>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.)</td>
</tr>
<tr>
<td>28 through 31</td>
<td>aaaa</td>
<td>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.</td>
</tr>
<tr>
<td>33 through 36</td>
<td>vvvv</td>
<td>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.</td>
</tr>
<tr>
<td>67 through 72</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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.

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.
### Disk Maintenance Programs
#### MODSF data records

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>aaaa</td>
<td>The address (in hexadecimal) in the program of the first word being changed.</td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td>Relocation code of the first word being changed; enter:</td>
</tr>
<tr>
<td>6</td>
<td>r</td>
<td>$A$ 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).</td>
</tr>
<tr>
<td>7</td>
<td>Blank</td>
<td>The value (in hexadecimal) that is being inserted in the first location.</td>
</tr>
<tr>
<td>8 through 11</td>
<td>xxxx</td>
<td>Relocation code of the second word being changed (see column 6).</td>
</tr>
<tr>
<td>12</td>
<td>Blank</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>15 through 18</td>
<td>xxxx</td>
<td></td>
</tr>
<tr>
<td>64 through 67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68 through 72</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>aaaa</td>
<td>Disk displacement (in hexadecimal) of the first word being changed with this data record.</td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td>The value (in hexadecimal) that is being inserted in the location specified by columns 1 through 4.</td>
</tr>
<tr>
<td>6 through 9</td>
<td>xxxx</td>
<td>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.</td>
</tr>
<tr>
<td>10</td>
<td>Blank</td>
<td>Reserved</td>
</tr>
<tr>
<td>11 through 14</td>
<td>xxxx</td>
<td>Not used</td>
</tr>
<tr>
<td>66 through 69</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>70 through 72</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*END patch control record

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

*END control record format

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>*END</td>
<td>These characters signify the end of input for MODSF.</td>
</tr>
<tr>
<td>5 through 72</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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 | 10| 15| 20| 25| 30| 35| 40| 45| 50| 55| 60| 65| 72|   |   |   |   |   |   |   |
| **11 JOB** | | | | | | | | | | | | | | | | | | | | |
| **11 XEQ MODSF** | | | | | | | | | | | | | | | | | | | | |
| **11020 200 FADD 0001 P** | 001B C900 001C D839 001D 18D0 | | | | | | | | | | | | | | | | | | |
| **0A1B A 1000 A 1000 A 1000** | | | | | | | | | | | | | | | | | | | | |
| **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 unforeseen 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:

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

**DFCNV Control Records**

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

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

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:

```
// XEQ DFCNV 1
```
The file description control record contains the following information.

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

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.
The second required control record, field specification control record, 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
--- --- --- --- --- --- --- --- --- --- --- --- --- ---  
   0   1
```

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.

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:

```
15-15.2(P)R3
```

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

<table>
<thead>
<tr>
<th>Word:</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents:</td>
<td>XXX0</td>
<td>0FYY</td>
<td>Y00F</td>
<td>ZZZ0</td>
<td>0F40</td>
</tr>
</tbody>
</table>

where

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

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.
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 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Content: 3A7E D64B 40D5 D540 D4C1 BC 00 00 80 03C8 0000 0083

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

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 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Content: FOFO D440 F9F6 F8F0 4040 40F0 FOF5 F3F

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:

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.
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.

Note. The System 2501/1442 conversion routine interprets the following character punches as equal: ' and @, + and &, = and #, ) and t, ( and %.

The characters ', +, =, ), and ( are printed. The conversion routine is used during analysis of control records, source input for language processors, and DUP input/output data. This routine provides uniformity for 024 and 029 prepared input.

**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 5 and reserved card columns indicate that the columns must be blank. You can replace card columns shown as not used with comments.
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)

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>// JOB</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Temporary mode indicator</td>
<td>T or blank. A T indicates that temporary mode is desired for this job.</td>
</tr>
<tr>
<td>9 through 10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11 through 14</td>
<td>First ID</td>
<td>This is the ID of the master cartridge (logical drive 0).</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16 through 19</td>
<td>Second ID</td>
<td>This is the ID of the cartridge on logical drive 1.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 24</td>
<td>Third ID</td>
<td>This is the ID of the cartridge on logical drive 2.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26 through 29</td>
<td>Fourth ID</td>
<td>This is the ID of the cartridge on logical drive 3.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>Fifth ID</td>
<td>This is the ID of the cartridge on logical drive 4.</td>
</tr>
<tr>
<td>35</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>36 through 39</td>
<td>CIB ID</td>
<td>This is the ID of the cartridge containing the CIB to be used during this job.</td>
</tr>
<tr>
<td>Card column</td>
<td>Contents</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>40</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>41 through 44</td>
<td>Working storage ID</td>
<td>This is the ID of the cartridge containing the working storage to be used by the monitor during this job. See <code>*FILES</code>, for details on working storage for your programs.</td>
</tr>
<tr>
<td>45</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>46 through 49</td>
<td>Unformatted disk I/O ID</td>
<td>This is the ID of the cartridge containing the unformatted disk I/O area to be used during this job.</td>
</tr>
<tr>
<td>50</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>51 through 58</td>
<td>Date, name, etc.</td>
<td>This information is printed at the top of every page of the listing on the principal print device during this job.</td>
</tr>
<tr>
<td>59</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>60 and 61</td>
<td>EQUAT record count</td>
<td>This number specifies how many EQUAT records follow this JOB record.</td>
</tr>
<tr>
<td>62 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
**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 following chart:

<table>
<thead>
<tr>
<th>DUP operations</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMP</td>
<td>None</td>
</tr>
<tr>
<td>DUMPDATA, DUMPDATAE</td>
<td>None</td>
</tr>
<tr>
<td>STORE</td>
<td>None</td>
</tr>
<tr>
<td>STORECI</td>
<td>To UA only</td>
</tr>
<tr>
<td>STOREDATA, STOREDATAE</td>
<td>To UA and WS only</td>
</tr>
<tr>
<td>STOREDATACI</td>
<td>To UA only</td>
</tr>
<tr>
<td>STOREMOD</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DUMPLET</td>
<td>None</td>
</tr>
<tr>
<td>DUMPFLET</td>
<td>None</td>
</tr>
<tr>
<td>DWADR</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DELETE</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE FIXED AREA</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID ASSEMBLER</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID FORTRAN</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID RPG</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID COBOL</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DFILE</td>
<td>To UA only</td>
</tr>
<tr>
<td>MACRO UPDATE</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>// JOB</td>
<td>// JOB T</td>
<td>// JOB 1004 1005 1006</td>
</tr>
</tbody>
</table>

- **This is all that is necessary for a one-drive system.**
- **This specifies temporary mode for the current job, a heading for each printed page, and that 2 EQUAT control records follow.**
- **This specifies disk IDs 1004, 1005, and 1006 on logical drives 0, 1, and 2, respectively, and that 1005 contains the CIB and 1006 contains working storage for this job.**

---

### // 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.

<table>
<thead>
<tr>
<th>format</th>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>general function</td>
<td>1 through 6</td>
<td>//ASM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
Monitor Control Records

// FOR // RPG // COBOL // DUP

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>//FOR</td>
<td></td>
</tr>
<tr>
<td>7 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// RPG

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>//RPG</td>
<td></td>
</tr>
<tr>
<td>7 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// COBOL

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>//COBOL</td>
<td></td>
</tr>
<tr>
<td>9 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// DUP

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>//DUP</td>
<td></td>
</tr>
<tr>
<td>7 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
### Monitor Control Records

#### // XEQ

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>//XEQ</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8 through 12</td>
<td>Name</td>
<td>This is the name (left-justified) of the DSF program or DCI program to be executed.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Core map indicator</td>
<td>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.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16 and 17</td>
<td>Count</td>
<td>A decimal number (right-justified) that indicates the number of supervisor control records that follow.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Disk I/O subroutine indicator</td>
<td>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.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 24</td>
<td>Cartridge ID</td>
<td>The ID of the cartridge that contains the mainline program in its working storage; blanks in this field indicate that the program is in system working storage.</td>
</tr>
<tr>
<td>25</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>LOCAL-call-LOCAL indicator</td>
<td>A punch in this column enables a LOCAL subroutine to call another LOCAL.</td>
</tr>
<tr>
<td>27</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Special ILS indicator</td>
<td>A punch in this column indicates that ILSs for this core load should be chosen from the special ILSs.</td>
</tr>
<tr>
<td>29</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>XEQ working storage indicator</td>
<td>A punch in this column allows the execution of a DSF or DCI program from working storage if a LET/FLET search fails to find the program named in the name field. When the name field is blank, it allows the execution of a DCI program from working storage.</td>
</tr>
<tr>
<td>31 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**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, EQUAT, 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.

<table>
<thead>
<tr>
<th>Column 19</th>
<th>Disk I/O subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank or Z</td>
<td>DISKZ</td>
</tr>
<tr>
<td>0 or 1</td>
<td>DISK1</td>
</tr>
<tr>
<td>N</td>
<td>DISKN</td>
</tr>
</tbody>
</table>

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 Working Storage Indicator. A punch (any character) in column 30 allows the execution of any type program from working storage.

Working storage will be checked for a DSF or DCI program if the system fails to find a program in LET/FLET as indicated by the name field. Without this indicator set, the system will search only LET/FLET for the named program.

Working storage will be checked for a DSF or DCI type program when the name field is blank. Without this indicator set, the system will check working storage only for a DSF type program.
Monitor Control Records

// XEQ
// * (comments)

// XEQ Examples

1: // XEQ
2: // XEQ NAME 02 XXX
3: // XEQ L 1004

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 executed.

// * (Comments)
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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>//N*</td>
<td>Any alphameric characters can be used.</td>
</tr>
<tr>
<td>5 through 80</td>
<td>Comments</td>
<td></td>
</tr>
</tbody>
</table>
Monitor Control Records
// PAUS   // TYP
// TEND

// PAUS

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 7</td>
<td>//PAUS</td>
<td></td>
</tr>
<tr>
<td>8 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// TYP

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>//TYP</td>
<td></td>
</tr>
<tr>
<td>7 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// TEND

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 7</td>
<td>//TEND</td>
<td></td>
</tr>
<tr>
<td>8 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
// 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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>//EJECT</td>
<td></td>
</tr>
<tr>
<td>9 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// 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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>//CPRNT</td>
<td></td>
</tr>
<tr>
<td>9 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

// CEND

general function

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

format

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 7</td>
<td>//CEND</td>
<td></td>
</tr>
<tr>
<td>8 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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, EQUAT, 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.

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 through column 80. A blank is used as a termination character for each record; therefore, embedded blanks cannot be coded within the characters in a record. A comma (,) followed by a blank is used as a continuation indicator, which means the information is continued on the following record. If sequence numbers are applied to the records, at least one blank must appear before the sequence information.

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**

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
HLOCALMAIN1, SUB1, SUB2, ... , SUBn
```

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

**Additional field information**

`MAINI`. You replace `MAINI` 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
HLOCALMAIN1, SUB1, SUB2, ...
HLOCALSUB3,
  ...
HLOCALSUBn
```

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

```
1 5 10 15 20 25 30 35 40 45 50
HLOCALMAIN1, SUB1
HLOCALMAIN1, SUB2
  ...
HLOCALMAIN1, 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.

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

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

When the mainline program is to be executed from working storage and the name specified is blank, the name of the mainline program is omitted from LOCAL control records.

When LOCALs are used in conjunction with a STORECI operation, the name of the mainline must be specified.

**NOCAL**

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.

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

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

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>FILE1, NAME1</td>
<td>...</td>
<td>FILEn, NAMEn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE1, NAME1, CAR1</td>
<td>...</td>
<td>FILEn, NAMEn, CARn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE1, CAR1</td>
<td>...</td>
<td>FILEn, CARn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>FILE1, NAME1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE1, NAME1, CAR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE1, CAR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILE1, NAME1, CAR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**G2250**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 11</td>
<td>*G2250mimne</td>
<td>Specifies that graphic support is required for the named mainline program.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You replace <em>mimne</em> with the name of the program. If the program being</td>
</tr>
<tr>
<td></td>
<td></td>
<td>executed is in working storage, the program name is omitted.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>U, blank, or N</td>
<td><em>U</em> indicates the character stroke subroutine containing upper case,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>numeric, and special characters is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Blank</em> indicates the character stroke subroutine containing upper case,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower case, numeric, and special characters is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>N</em> indicates that a character stroke subroutine is not loaded.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Blank or N</td>
<td><em>Blank</em> indicates the scissoring subroutine is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>N</em> indicates the scissoring subroutine is not loaded.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Blank or N</td>
<td><em>Blank</em> indicates the ICA area expansion subroutine is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>N</em> indicates the ICA area expansion subroutine is not loaded.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Blank or N</td>
<td><em>Blank</em> indicates the index controlled entity subroutine is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>N</em> indicates the index controlled entity subroutine is not loaded.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Blank or N</td>
<td><em>Blank</em> indicates the level controlled direct entry subroutine is loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>N</em> indicates the level controlled direct entry subroutine is not loaded.</td>
</tr>
<tr>
<td>22 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
*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, or included in the count field of the preceding // XEQ monitor control record or *STORECI DUP 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, SUBm represents the name of the old subroutine. SUB2, SUBn represents the name of the new subroutine. SUB2 is substituted for SUB1. 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. Additional substitutions are indicated by supplying *EQUAT records to the subjobs (// XEQ, *STORECI).
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:

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 (STOREDATACI); columns 7 through 12 (or 21) identify the extended operation (STOREDATACI). Where shown in the control record format, a blank character (\$) 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:

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

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.

A file protection facility exists which is enabled by multi-punching one or more of the name field card columns during program or file definition. Column 21 must not be multi-punched. A delete of that file is not allowed unless the same column is multi-punched on the delete card. Assume that the following control card has an 'A' and an 4*5 punched in column 25.

```
1 5 10 15 20 25 30 35 40 45 50
AF 1L A
```

On the printer, it appears as:

```
1 5 10 15 20 25 30 35 40 45 50
tic LI
```

A disk dump of the LET sector containing this entry shows its name code to be:

C104 106F
Name code is derived in the following manner:

1. Find the low order 6-bit equivalent of each alphabetic character (the multi-punched column is represented as /EF).

2. Place these 30 bits in the order of the 5-character name field.

   C 1 0 4 1 0 6 F
   00 0001 0000 0100 0001 0000 0110 1111
   1100 0001 1100 0001 1100 0001 1100 0001 1110 1111
   C 1 C 1 C 1 E F
   A A A

   The fifth character is a multi-punched character. Any multi-punching in the corresponding column in a DUP control card name field will access that file if the other characters are the same.

Add 2 high-order bits to denote file type:

11 - Data File
10 - DCI
00 - DSF

4. Mark off bits into 8 groups of 4 bits each.
5. Convert each group to its hexadecimal equivalent.

4. Mark off bits into 8 groups of 4 bits each.
5. Convert each group to its hexadecimal equivalent.

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 information from one device or disk area to another device or disk area. In addition, the format conversions 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:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSF</td>
<td>Disk system format</td>
</tr>
<tr>
<td>DDF</td>
<td>Disk data format</td>
</tr>
<tr>
<td>DCI</td>
<td>Disk core image format</td>
</tr>
<tr>
<td>CDS</td>
<td>Card system format</td>
</tr>
<tr>
<td>CDD</td>
<td>Card data format</td>
</tr>
<tr>
<td>CDC</td>
<td>Card core image format</td>
</tr>
<tr>
<td>PTS</td>
<td>Paper tape system format</td>
</tr>
<tr>
<td>PTD</td>
<td>Paper tape data format</td>
</tr>
<tr>
<td>PTC</td>
<td>Paper tape core image format</td>
</tr>
<tr>
<td>PRD</td>
<td>Printer data format</td>
</tr>
<tr>
<td>NCF</td>
<td>Name code format</td>
</tr>
</tbody>
</table>

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 programs is allowed, such as dumping a program to cards and later storing it back to disk.
<table>
<thead>
<tr>
<th>From Area Symbols, with Formats</th>
<th>UA</th>
<th>FX</th>
<th>WS</th>
<th>CD</th>
<th>PT</th>
<th>NR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DDF</td>
<td>DDF</td>
<td>DDF</td>
<td>DDF</td>
<td>DDF</td>
<td>DDF</td>
</tr>
<tr>
<td>DDF</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
</tr>
<tr>
<td>DDF</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
</tr>
<tr>
<td>DDF</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
</tr>
<tr>
<td>DDF</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
<td>DUMP</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>DUP operations</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMP</td>
<td>None</td>
</tr>
<tr>
<td>DUMPDATA, DUMPDATAE</td>
<td>None</td>
</tr>
<tr>
<td>STORE</td>
<td>None</td>
</tr>
<tr>
<td>STORECI</td>
<td>To UA only</td>
</tr>
<tr>
<td>STOREDATA, STOREDATAE</td>
<td>To UA and WS only</td>
</tr>
<tr>
<td>STOREDATA1</td>
<td>To UA only</td>
</tr>
<tr>
<td>STOREMOD</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DUMPLET</td>
<td>None</td>
</tr>
<tr>
<td>DUMPFLET</td>
<td>None</td>
</tr>
<tr>
<td>DWADR</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DELETE</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE FIXED AREA</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID ASSEMBLER</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID FORTRAN</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID RPG</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DEFINE VOID COBOL</td>
<td>Not allowed</td>
</tr>
<tr>
<td>DF1</td>
<td>To UA only</td>
</tr>
<tr>
<td>MACRO UPDATE</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

*DUMP

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.

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

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

<table>
<thead>
<tr>
<th>*DUMP summary chart</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>From symbols, including formats</strong></td>
</tr>
<tr>
<td>UA (DSF)</td>
</tr>
<tr>
<td>UA or WS (DSF)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>UA or FX (DDF)</td>
</tr>
<tr>
<td>UA, FX, or WS (DDF)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>UA or FX (DCI)</td>
</tr>
<tr>
<td>UA, FX, or WS (DCI)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
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

1. XDUMP WS PR
   This dumps a program from working storage to the printer.

2. XDUMP UA WS MAIN
   This dumps a program named MAIN from the user area to working storage.

3. XDUMP FX WS MAIN1 1003
   This dumps a program named MAIN1 from the fixed area on disk 1003 to system working storage.

*DUMPDATA

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 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, if present, of the DUMPDATA control record or the block count of the program or data file.
<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 10</td>
<td>&quot;DUMPDATA&quot;</td>
<td></td>
</tr>
<tr>
<td>11 and 12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 26</td>
<td>Name</td>
<td>A name is required except when the dump is from working storage to the printer.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>The count (a right-adjusted decimal number) specifies the number of sectors to be dumped. If this field is blank, the working storage indicator or disk block count in LET or FLET is used.</td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td></td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To cartridge ID</td>
<td></td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>*DUMPDATA summary chart</th>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA(DFS)</td>
<td>WS(DDF)</td>
<td></td>
</tr>
<tr>
<td>UA or WS(DFS)</td>
<td>CD(CDD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PT(PTD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR(PRD)</td>
<td></td>
</tr>
<tr>
<td>UA or FX(DDF)</td>
<td>WS(DDF)</td>
<td></td>
</tr>
<tr>
<td>UA, FX, or WS(DDF)</td>
<td>CD(CDD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PT(PTD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR(PRD)</td>
<td></td>
</tr>
<tr>
<td>UA(DICI) or FX(DDF)</td>
<td>WS(DDF)</td>
<td></td>
</tr>
<tr>
<td>UA, FX, or WS(DICI)</td>
<td>CD(CDD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PT(PTD)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PR(PRD)</td>
<td></td>
</tr>
</tbody>
</table>
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 SPRET with /100F displayed in the ACCUMULATOR.

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

*DUMPDATA Examples

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DUMPDATA</td>
<td>IA</td>
<td>CD</td>
<td>DATA</td>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DUMPDATA</td>
<td>IX</td>
<td>WS</td>
<td>DATA1</td>
<td>1003</td>
<td>1007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DUMPDATA</td>
<td>WS</td>
<td>DT</td>
<td>DATA2</td>
<td>1002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 This dumps a data file named DATA from the user area to cards.
2 This dumps a data file named DATA1 from the fixed area on cartridge 1003 to working storage on cartridge 1007.
3 This dumps a data file named DATA2 from working storage on cartridge 1002 to paper tape.

*DUMPDATA E

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 11</td>
<td>*DUMPDATA E</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>A name is required except when the dump is from working storage to the printer.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>The count (a right-adjusted decimal number) specifies the number of sectors to be dumped. If this field is blank, the working storage indicator or disk block count in LET or FLET is used.</td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td></td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To cartridge ID</td>
<td></td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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

* DUMPDATA E summary chart

<table>
<thead>
<tr>
<th>From symbols</th>
<th>To symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA or FX</td>
<td>WS</td>
</tr>
<tr>
<td>UA, FX, or WS</td>
<td>CD PR</td>
</tr>
</tbody>
</table>

* DUMPDATA E additional field information

From 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. If present, the count overrides the contents of the working storage indicator or the disk block count in the LET or FLET entry; when present, this number of sectors is dumped regardless of the length of the program or data file.
DUP Control Records
*DUMPPDATA E
*DUMPLET

*DUMPPDATA E Examples

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>*DUMPPDATA E</td>
<td>This dumps a data file named DATA from the user area to working storage.</td>
</tr>
<tr>
<td>9 through 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>This dumps a data file named DATA1 from the fixed area to cards.</td>
</tr>
<tr>
<td>26 through 30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td>This dumps a data file from working storage to the printer.</td>
</tr>
<tr>
<td>35 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*DUMPLET

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.

**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**

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

**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**

1. **DUMPLET**
   - This dumps LET (and FLET) from the disks defined by the current JOB monitor control record.

2. **DUMPLET**
   - This dumps LET (and FLET) from cartridge 1004.

3. **DUMPLET**
   - This dumps the LET (or FLET) entry for the program named MAIN.

---

**DUMPFLET**

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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 10</td>
<td><em>DUMPFLET</em></td>
<td></td>
</tr>
<tr>
<td>11 through 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>Name specifies that only the FLET entry for that program or data file is printed.</td>
</tr>
<tr>
<td>26 through 30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td>The cartridge ID specifies that only the FLET on that cartridge is printed.</td>
</tr>
<tr>
<td>35 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*DUMPFLET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>*DUMPFLET</td>
<td>MAIN1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>*DUMPFLET</td>
<td>MAIN2</td>
<td>1002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. This dumps FLET from the disks defined by the current JOB monitor control record.
2. This dumps the FLET entry for the program named MAIN1.
3. This dumps the FLET entry for the program named MAIN2 from cartridge 1002.

*STORE

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

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

<table>
<thead>
<tr>
<th>*STORE summary chart From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS(DSF)</td>
<td>UA(DSF)</td>
</tr>
<tr>
<td>CD(CDS)</td>
<td>UA or WS(DSF)</td>
</tr>
<tr>
<td>PT(PTS)</td>
<td>UA or WS(DSF)</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Subroutine description</th>
<th>Type</th>
<th>Code in subtype field</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-core subroutines</td>
<td>3, 4</td>
<td>0</td>
</tr>
<tr>
<td>Disk FORTRAN I/O subroutines</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Arithmetic subroutines</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Nondisk FORTRAN I/O and &quot;Z&quot;</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>&quot;Z&quot; device subroutines</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Function subroutines</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Dummy ILS02, ILS04 stored in monitor system library</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>User-written ILS02, ILS04 that replace dummy ILS02, ILS04</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

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. XSTORE CD WS
   This reads a program from cards and stores it in working storage.

2. XSTORE WS UA MAIN
   This names a program in working storage MAIN and stores it in the user area.

3. XSTORE CD UA ILS04
   This reads from cards an ILS04 you have written and stores it in the user area.
### *STOREDATA*

This control record (1) transfers information from working storage to the user area or fixed area, 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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 10</td>
<td><em>STOREDATA</em></td>
<td></td>
</tr>
<tr>
<td>11 and 12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>A name is not required when the STOREDATA operation is from cards or paper tape to working storage.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>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 count field is blank, the contents of the working storage indicator are used. 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.</td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td></td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To cartridge ID</td>
<td></td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
The following chart is a summary of the information transfers and format conversions performed by STOREDATA.

**STOREDATA summary chart**

<table>
<thead>
<tr>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS(DSF, DDF, DCI)</td>
<td>UA or FX(DDF)</td>
</tr>
<tr>
<td>CD(CDS, CDD, CDC)</td>
<td>UA, FX, or WS(DDF)</td>
</tr>
<tr>
<td>PT(PTS, PTD, PTC)</td>
<td>UA, FX, or WS(DDF)</td>
</tr>
</tbody>
</table>

*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**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>STOREDATA</em></td>
<td>PT WS</td>
</tr>
<tr>
<td>2</td>
<td><em>STOREDATA</em></td>
<td>WS UA FILE1 0005 1005</td>
</tr>
<tr>
<td>3</td>
<td><em>STOREDATA</em></td>
<td>CD UA FILE2 0200</td>
</tr>
<tr>
<td>4</td>
<td><em>STOREDATA</em></td>
<td>WS UA FILE3</td>
</tr>
</tbody>
</table>

1. This reads a data file from paper tape, and stores it in system working storage.
2. This transfers a data file named FILE1 that occupies 5 sectors from system working storage to the user area on cartridge 1005.
3. This reads a data file named FILE2 from cards, and stores it in the user area. 200 cards are read.
4. This transfers a data file named FILE3 from working storage to the user area. Count is in the working storage indicator.

**STOREDATAE**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 11</td>
<td>*STOREDATAE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>A name is not required when the STOREDATAE operation is from cards to working storage.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>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.</td>
</tr>
<tr>
<td>31 through 34</td>
<td>From</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>*STOREDATAE summary chart</th>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS</td>
<td>UA or FX</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>UA, FX, or WS</td>
<td></td>
</tr>
</tbody>
</table>

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.
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. STOREDATAE WS FILE'S 00402
2. STOREDATAE CD WS 0056 1003

1 This transfers a data file named FILE5 from working storage to the fixed area. The file occupies 2 sectors.
2 This reads a data file of 56 cards into working storage on cartridge 1003.
**STOREDATACI**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 12</td>
<td>*STOREDATACI</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td>A name is not required when the STOREDATACI operation is to working storage.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>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.</td>
</tr>
<tr>
<td>31 through 34</td>
<td>From</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used.</td>
<td></td>
</tr>
</tbody>
</table>
The following chart is a summary of the information transfers and format conversions performed by STOREDATACI.

*STOREDATACI

**summary chart**

<table>
<thead>
<tr>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS(DCI)</td>
<td>UA or FX(DCI)</td>
</tr>
<tr>
<td>CD(CDC, CDD)</td>
<td>UA, FX, or WS(DCI)</td>
</tr>
<tr>
<td>PT(PTC, PTD)</td>
<td>UA, FX, or WS(DCI)</td>
</tr>
</tbody>
</table>

**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*

1

<table>
<thead>
<tr>
<th>1</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

This transfers the data file, FILE5, from working storage on cartridge 1001 to the fixed area on the system cartridge.

2

This reads a data file from cards and stores it in working storage. 108 cards are read.

*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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>*STORECI</td>
<td>This column specifies the disk I/O subroutine to be used by the core load during execution.</td>
</tr>
<tr>
<td>9</td>
<td>Disk I/O subroutine indicator</td>
<td>A punch (any character) in this column enables a LOCAL subroutine to call another LOCAL.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LOCAL-call-LOCAL indicator</td>
<td>A decimal number (right-justified) that indicates the number of supervisor control records (FILES, LOCAL, NOCAL, EQUAT, and G2250) that follow.</td>
</tr>
<tr>
<td>12</td>
<td>Special ILS indicator</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>From</td>
<td></td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Core map indicator</td>
<td>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.</td>
</tr>
<tr>
<td>43 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
The following chart is a summary of the information transfers and format conversions performed by STORECI.

<table>
<thead>
<tr>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS(DSF)</td>
<td>UA or FX(DCI)</td>
</tr>
<tr>
<td>CD(CDS)</td>
<td>UA or FX(DCI)</td>
</tr>
<tr>
<td>PT(PTS)</td>
<td>UA or FX(DCI)</td>
</tr>
</tbody>
</table>

*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.

**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 be used on the LOCAL and NOCAL record. It must not be used on the G2250 record. 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

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 | XSTORECI | W5 | UA | MAIN1 |
| 2 | XSTORECI1 | XCD | FX | MAIN7 | 1003 | N |
| 3 | XSTORECI1 | XPT | UA | MAIN2 0002 |

1. This converts the DSF program, MAIN1, into DCI format and transfers it from working storage to the user area.
2. This specifies that DISK1 is to be used by this core load, and that special ILSs are to be used. The program, MAIN7, is read from cards and stored in the fixed area on cartridge 1003. N in column 42 suppresses the printing of a core map.
3. This reads program MAIN2 from paper tape and stores it in the user area. The X in column 11 indicates that a LOCAL calls another, and 0002 in 27-30 indicates that two supervisor control records follow.
**STOREMOD**

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, the contents of working storage are stored by STORE, STOREDATA, or STOREDATACI, when the respective format is DSF, DDF, or DCI. The STOREMOD operation is not allowed when temporary mode is indicated in the current JOB monitor control record.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 10</td>
<td><em>STOREMOD</em></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Subtype</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13 and 14</td>
<td>From symbol</td>
<td>The source is always working storage.</td>
</tr>
<tr>
<td>15 and 16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17 and 18</td>
<td>To symbol</td>
<td>See the following summary chart.</td>
</tr>
<tr>
<td>19 and 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>26 through 30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>From</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>35 and 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>To</td>
<td>cartridge ID</td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
The following chart is a summary of the information transfers and format conversions performed by STOREMOD.

<table>
<thead>
<tr>
<th>From symbols, including formats</th>
<th>To symbols, including formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS(DSF)</td>
<td>UA(DSF)</td>
</tr>
<tr>
<td>WS(DDF)</td>
<td>UA or FX(DDF)</td>
</tr>
<tr>
<td>WS(DCI)</td>
<td>UA or FX(DCI)</td>
</tr>
</tbody>
</table>

*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**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>STOREMOD</td>
<td>WS</td>
</tr>
<tr>
<td>STOREMOD</td>
<td>WS</td>
</tr>
</tbody>
</table>

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*

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 contiguous, 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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td><em>DELETE</em></td>
<td>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.</td>
</tr>
<tr>
<td>9 through 20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21 through 25</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>26 through 30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>31 through 34</td>
<td>From cartridge ID</td>
<td></td>
</tr>
</tbody>
</table>

DUP Control Records

*DELETE Examples*

1. DELETE

   MAIN1

   This deletes LET or FLET entry for the program, MAIN1, from the cartridge on the first logical drive where the name is found.

2. DELETE

   FILE1 1004

   This deletes the data file, FILE1, from cartridge 1004.
*DEFINE

This control record performs 4 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 fixed area and FLET.
- 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.

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.

The fixed area is increased in cylinder increments. It is decreased in cylinder increments by deleting 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 (by using *DELETE), the fixed area and FLET can be deleted by specifying a number in the count field that reduces the fixed area and FLET to one cylinder or less.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>*DEFINE</td>
<td></td>
</tr>
<tr>
<td>9 through 18</td>
<td>FIXEDAREA</td>
<td></td>
</tr>
<tr>
<td>19 through 26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27 through 30</td>
<td>Count</td>
<td>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.</td>
</tr>
<tr>
<td>31</td>
<td>Sign</td>
<td>Blank if the fixed area is being increased; a minus sign if the fixed area is being decreased.</td>
</tr>
<tr>
<td>32 through 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>Cartridge ID</td>
<td>This ID specifies the cartridge that is being altered; when omitted, the system cartridge is assumed.</td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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. This defines a 5 cylinder fixed area on the master cartridge.
2. This decreases the size of the fixed area on cartridge 1002 by 2 cylinders.

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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 8</td>
<td>*DEFINEx</td>
<td></td>
</tr>
<tr>
<td>9 through 13</td>
<td>VOIDx</td>
<td></td>
</tr>
<tr>
<td>14 through 22</td>
<td>ASSEMBLER or FORTRANx or RPGx or COBOLx</td>
<td></td>
</tr>
<tr>
<td>23 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 6</td>
<td>*DWADR</td>
<td></td>
</tr>
<tr>
<td>7 through 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 40</td>
<td>Cartridge ID</td>
<td>This ID specifies the cartridge on which the working storage sector addresses are to be rewritten.</td>
</tr>
<tr>
<td>41 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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

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**

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

*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**

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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 13</td>
<td>*MACROUPDATE</td>
<td></td>
</tr>
<tr>
<td>14 through 36</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>37 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*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 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.
All assembler control records have the following format:

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>Option</td>
<td>Replace <code>option</code> with the keywords for the control record being used.</td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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.

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**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>TWO PASS MODE</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>LIST</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
The format of a printed listing for an 8K or larger system is shown by:

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, \( \text{\text{a}} \) and \( \text{\text{a}} \) are the entry label in name code
   e. For LIBF and CALL statements, \( \text{\text{a}} \) and \( \text{\text{a}} \) 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, \( \text{\text{a}} \) and \( \text{\text{a}} \) are the entry label in name code
   c. For LIBF and CALL statements, \( \text{\text{a}} \) and \( \text{\text{a}} \) 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:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100F</td>
<td>0</td>
<td>A814</td>
<td>M</td>
<td>D</td>
<td>SAVE+2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>0</td>
<td>8012</td>
<td>M</td>
<td>A</td>
<td>DIVIDE BY (I+J)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>0</td>
<td>3000</td>
<td>W AIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td>0</td>
<td>6038</td>
<td>EXIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1014</td>
<td>00</td>
<td>0000</td>
<td>BSS</td>
<td>E</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1014</td>
<td>00</td>
<td>0000C000</td>
<td>B</td>
<td>DEC</td>
<td>49152</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1016</td>
<td>00</td>
<td>0000E000</td>
<td>F</td>
<td>DEC</td>
<td>57344</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, \( i \) and \( j \) are the entry label in name code
   e. For LIBF and CALL statements, \( i \) and \( j \) 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, \( i \) and \( j \) are the entry label in name code
   c. For LIBF and CALL statements, \( i \) and \( j \) 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**

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 comment 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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>XREF</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value of the symbol</th>
<th>Relocation indicator</th>
<th>Statement number of statement that defines the symbol</th>
<th>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</th>
</tr>
</thead>
<tbody>
<tr>
<td>K1</td>
<td>105D</td>
<td>0</td>
<td>00071</td>
<td>00007&gt;R 00013&gt;R 00038&gt;R 00057&gt;R 00063&gt;R 00123&gt;R</td>
</tr>
<tr>
<td>K16</td>
<td>106C</td>
<td>0</td>
<td>00083</td>
<td>00065&gt;R</td>
</tr>
<tr>
<td>K20</td>
<td>105E</td>
<td>0</td>
<td>00072</td>
<td>00003&gt;R 00019&gt;R</td>
</tr>
<tr>
<td>K32</td>
<td>105F</td>
<td>0</td>
<td>00073</td>
<td>00044&gt;R 00116&gt;R 00117&gt;R 00121&gt;R</td>
</tr>
<tr>
<td>KA0</td>
<td>1060</td>
<td>0</td>
<td>00074</td>
<td>00062&gt;R 00064&gt;M 00068&gt;M</td>
</tr>
<tr>
<td>K640</td>
<td>159F</td>
<td>0</td>
<td>00131</td>
<td>00040&gt;B</td>
</tr>
<tr>
<td>LINE</td>
<td>1661</td>
<td>0</td>
<td>00075</td>
<td>00007&gt;R 00013&gt;R 00038&gt;R 00057&gt;R 00063&gt;R 00123&gt;R</td>
</tr>
<tr>
<td>LINES</td>
<td>1064</td>
<td>0</td>
<td>00078</td>
<td>00065&gt;R</td>
</tr>
<tr>
<td>LOOP</td>
<td>1022</td>
<td>0</td>
<td>00026</td>
<td>00003&gt;R 00019&gt;R</td>
</tr>
</tbody>
</table>

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**

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.

**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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>LIST DECK</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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:

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

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>LIST DECK E</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**PRINT SYMBOL TABLE**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>PRINT SYMBOL TABLE</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**PUNCH SYMBOL TABLE**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>PUNCH SYMBOL TABLE</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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.

*SAVE SYMBOL TABLE

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.

*SYSTEM SYMBOL TABLE

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.
**LEVEL**

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.

**OVERFLOW SECTORS**

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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>LEVEL(n)</td>
<td>(n) is an interrupt level number (decimal)</td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

### Note

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

#### 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, \(n2\),
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:

<table>
<thead>
<tr>
<th>Size of core storage (in decimal words)</th>
<th>Approximate maximum number of symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>3500</td>
</tr>
<tr>
<td>8192</td>
<td>4165</td>
</tr>
<tr>
<td>16384</td>
<td>6895</td>
</tr>
<tr>
<td>32768</td>
<td>12355</td>
</tr>
</tbody>
</table>

#### \(n2\)

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:

\[
\text{Number of words} = 3 + N + \sum_{i=1}^{N} \frac{1}{2}(m_i + 1)
\]

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 \(3 + 3 + \frac{1}{2}(5 + 1) + \frac{1}{2}(4 + 1) + \frac{1}{2}(1 + 1) = 12\) 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:

\[
n2 = \frac{1}{100}(x - 100)
\]

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**

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. Variables within COMMON must be assigned specific addresses with EQU statements. Assignment of those variables with code generating instructions causes a R40 error message to be generated.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 71</td>
<td>COMMONnnnn</td>
<td>nnnnn is the number (in decimal) of words of COMMON to be saved between links.</td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**MACLIB**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 8</td>
<td>MACLIB</td>
<td></td>
</tr>
<tr>
<td>9 through 13</td>
<td>Macro library name</td>
<td></td>
</tr>
<tr>
<td>14 through 71</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**NOSET**

This control record allows object code from assembly to be built and stored although errors are included. If NOSET is an included control record, $NDUP is not set when errors occur. The setting of $NXEQ remains unchanged. Thus, with a lengthy assembly, it is possible to patch the incorrect statements via MODSF until a final assembly is desired.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 6</td>
<td>NOSET</td>
<td></td>
</tr>
<tr>
<td>7 through 71</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>72 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
FORTRAN CONTROL RECORDS

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:

```
// FOR
// JOB
```

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.

All FORTRAN control records have the following format:

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>Option</td>
<td>Replace option with the keywords for the control record being used.</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**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**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>2 through 72</td>
<td>IOCS (d, d, . . . , d)</td>
<td>d is a valid device name selected from the following list.</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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**
  - 1442 Card Read/Punch, Model 6 or 7
  - 2501 Card Reader
  - 1442 Card Punch, Model 5 (1442 Model 6 or 7 if used as a punch only)
  - Console printer
  - Keyboard
  - 1132 Printer
  - 1403 Printer
  - 1134/1055 Paper Tape Reader/Punch
  - 1627 Plotter
  - Disk
  - Disk (unformatted disk I/O)

- **IOCS device name**
  - CARD
  - 2501 READER
  - 1442 PUNCH
  - TYPEWRITER
  - KEYBOARD
  - 1132 PRINTER
  - 1403 PRINTER
  - PAPER TAPE
  - PLOTTER
  - DISK
  - UDISK

- **Subroutine called**
  - CARDZ
  - READZ
  - PNCHZ
  - WRTYZ
  - TYPEZ
  - PRNTZ
  - PRNZ
  - PAPTZ
  - PLOTX
  - DISKZ
  - 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.
**FORTRAN Control Records**

---

*LIST SOURCE PROGRAM*

This control record causes the source program, as it is entered, to be listed on the principal print device.

**Format**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>LIST SOURCE PROGRAM</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*LIST SUBPROGRAM NAMES*

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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>LIST SUBPROGRAM NAMES</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

---

*IOCS Examples*

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 10 15 20 25 30 35 40 45 50</td>
<td>I O C S ( C A R D , 1 4 0 3 P R I N T E R , D I S K )</td>
</tr>
<tr>
<td></td>
<td>I O C S ( P A P E R T A P E , 1 1 3 2 P R I N T E R , D I S K )</td>
</tr>
</tbody>
</table>

Page of GC26-3717-9, 10
As Updated October 22, 1976
By TNL GN34-0353

5-66
**Lists Symbol Table**

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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>LIST SYMBOL TABLE</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**List All**

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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>LIST ALL</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

The FORTRAN sample program in Appendix H is listed by a LIST ALL control record.
**EXITED PRECISION**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>EXTENDED PRECISION</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

**ONE WORD INTEGERS**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisks</td>
</tr>
<tr>
<td>2 through 72</td>
<td>ONE WORD INTEGERS</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
### *NAME*

This control record causes the specified program name to be printed at the end of the program listing.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>NAMExxxx</td>
<td>*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.</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

### ** (Header Information)

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>**</td>
<td>Asterisks</td>
</tr>
<tr>
<td>3 through 72</td>
<td>Any string of characters</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
*ARITHMETIC TRACE

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>ARITHMETIC TRACE</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

*TRANSFER TRACE

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>TRANSFER TRACE</td>
<td></td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
FORTRAN Control Records
*ORIGIN
RPG Control Cards

*ORIGIN
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

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>2 through 72</td>
<td>ORIGIN/ddddd or ORIGIN/xxxx</td>
<td>This is the starting core address expressed as a decimal number (ddddd) of 3 to 5 digits or as a hexadecimal number (/xxxx) of 1 to 4 digits preceded by a slash.</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Disk I/O subroutine in core</th>
<th>Core load origin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>DISKZ</td>
<td>510</td>
</tr>
<tr>
<td>DISK1</td>
<td>690</td>
</tr>
<tr>
<td>DISKN</td>
<td>960</td>
</tr>
</tbody>
</table>

RPG CONTROL CARDS

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:

```plaintext
// JOB
// RPG
// DUP
*STORE WS UA PROGN
// XEQ PROGN X
/*
Data file
End of file signals end of data card input file
RPG data file (if file not already stored on disk)
Execute the program. X or any other entry in column 28 will bring in the special ILS routines required by RPG.
Get PROGN (program name) from working storage and store it in the user area.
Disk utility program call
End of file card for source deck
RPG source deck (specification statements)
RPG control card
Monitor control card to call the RPG compiler
Monitor control card to initiate the job
```
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**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 5</td>
<td>Described in <em>IBM 1130 RPG Language</em></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td>Identifies this card as an RPG control card</td>
</tr>
<tr>
<td>7 through 10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Blank, B, or D</td>
<td>Blank indicates compilation with a listing of the program. B indicates compilation only. D indicates a listing only.</td>
</tr>
<tr>
<td>12 through 80</td>
<td>Described in <em>IBM 1130 RPG Language</em></td>
<td></td>
</tr>
</tbody>
</table>

**End-of-File Control Card**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>/*</td>
<td></td>
</tr>
<tr>
<td>3 through 80</td>
<td>Not used</td>
<td></td>
</tr>
</tbody>
</table>
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
- Using EQUATS
- 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.
Monitor Control

// JOB
// XEQ C
*STORE C
// DUP

Source program C

FORTRAN control records

// FOR
// PAUS
// *comments
// JOB

Object program B

// FOR
// PAUS
// *comments
// JOB

// STORE B
// DUP
// PAUS

Job B

// ASM
// PAUS
// *comments
// JOB

Source program A

Assembler control records

Cold start card (see cold start operating procedure)

Job A
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.

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.

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

DISK 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 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.)

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 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.
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.

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.

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.

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.

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.

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. Use DUMPDATA to dump a file to cards and STOREDATA to store these cards back on disk.

DUMPDATA dumps by sector count. For example, the control record:

\[
\begin{array}{cccccccccc}
1 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\
+DUMPDATA & WS & CD & NAMEF & 029 \\
\end{array}
\]

causes 3 sectors to be dumped to 18 cards; 17 cards of 54 words and one card of 42 words. The last 12 words of card 18 do not contain data.

STOREDATA stores by card count. To store the cards in this example, the control record:

\[
\begin{array}{cccccccccc}
1 & 5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 \\
+STOREDATA & CD & UAS & NAMEF & 018 \\
\end{array}
\]

causes the contents of these 18 cards, excluding the contents of the 12 unused words on card 18, to be stored back in 3 sectors. Note that if you use DUMPDATA to dump to cards, the number of cards (same as the last—highest—sequence number in cc 78-80 of the cards dumped) is the number to enter in the count field of the STOREDATA card.

STOREDATA for Cards Not Processed by DUMPDATA

If you use STOREDATA to store cards produced by a function other than DUMPDATA, some of the words in the last card may not be stored. To prevent this, use the following formula (based on the number of cards) to determine the card count to specify in the count field of the STOREDATA control record:

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 rewriting 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.

For a FORTRAN program, the core load builder cannot detect a LOCAL-calls-a-LOCAL condition between FORTRAN format I/O routines and the I/O subroutines that they call. Therefore:

- A FORTRAN format I/O routine and any routine that it calls cannot both be specified as LOCALs in the same core load.
- A user subroutine that contains I/O statements and the FORTRAN I/O routines that are used to execute those statements cannot both be specified as LOCALs in the same core load.

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 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:

\[3M + 2S \leq 640\]

where

- \(M\) is the total number of mainlines specified in the LOCAL control records.
- \(S\) is the total number of subroutines 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.
- Although a subroutine's instructions or data areas may be altered during execution, later LOCAL/SOCAL reloading may put the subroutine back into its original state.
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.

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.

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.
<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>FT</th>
<th>T</th>
<th>Operands &amp; Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td></td>
<td>25</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>40</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>55</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*EXAMPLE OF I.S.S. SUBROUTINE FOR LEVEL 5.*

*TO ALLOW TRACING DURING INTERRUPT, RUN MODE.*

**ADDR DC: */.0047*

**CORE LD BLDR PUTS ACTUAL ENT. DT. OF I.S.S. IN ADDR.*

**INT DC: X-X.*

**STD TEMP SAVE ACC-EXT.*

**STS RETUR SAVE STATUS.*

**STX 1 XR1+1 SAVE XR1.*

**STX 2 XR2+1 SAVE XR2.*

**BSI 1 ADDR.*

**RETURN LDS 0 RESTORE STATUS.*

**LDD TEMP RESTORE ACC-EXT.*

**XR1 LDX L1 X-X RESTORE XR1.*

**XR2 LDX L2 X-X RESTORE XR2.*

**BOSC 1 INT EXIT-TURN OFF INT LEVEL.*

**TEMP BSS E 2 (ACC-EXT).*

**END.*

*EXAMPLE OF I.S.S. TRACE SUBROUTINE.*

ENTERED AFTER EACH MAINLINE INSTRUCTION.*

**ISS 20 TRACY  ISS NO. 13 20.*

**TRACY DC: X-X.*

**LOC L LOC SUSPECTED LOCATION.*

**BSC + WAIT AND WAIT FOR OPERATOR ACTION WHEN FIRST ZERO.*

**BSC 1 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

A subroutine that is included in one SOCAI overlay must not call a subroutine included in another SOCAI overlay or cause another SOCAI overlay to be loaded into core before execution of the current SOCAI is complete. This restriction is required because the IBM-supplied 1130 subroutines that are used in SOCAIs are not re-enterable.

Note that disk I/O is used every time a SOCAI 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 SOCAI 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

The use of SOCAIs 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 SOCAIs are called into core. Ideally, your program should be written in sections, each employing a single SOCAI; input, computation, and output. Plan input and output carefully so as to separate disk and nondisk operations whenever possible.

The Use of EQUAT

EQUAT records can be applied to both the job and the subjob (XEQ or STORECI). Job EQUAT remain in effect throughout the entire job. Subjob EQUAT remain in effect only for the subjob operation. Subjob EQUAT take precedence over job EQUAT. They can be used to add additional EQUAT, override job EQUAT or nullify job EQUAT as shown in the following examples.

```
1 5
1/ JOB
X(EQUAT(SUB1, SUBA), (SUB2, SUBB))
1/X EQ
X(EQUAT(SUB3, SUBC))
1/DUP
X(STORECI) WS UA MAIN 1
X(EQUAT(SUB3, SUBC))
1/X EQ
X(EQUAT(SUB2, SUBD))
1/DUP
X(STORECI) WS UA MAIN 1
X(EQUAT(SUB2, SUBD))
```

Programming Tips and
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):

```
// XEQ L
R 41 7908 (HEX) WCS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
FSR 0248
LIBF TRANSFER VECTOR
FARC 069A
XMSC 067E
HCLL 062E
PRTY 05DE
EBPA 058E
FACD 04CD
FDIV 053C
FLD 0488
FACDX 04E3
FMPYX 049E
FSTO 046C
FGETP 0452
NCRM 0428
TYPEO 0312
EBPRT 02AC
IFIX 0280
FLCAT 0230
SYSTEM SUBROUTINES
ILSO4 00C4
ILSO2 0083
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 ILSO2 and ILSO4 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):
// XEQ L 2

*LCCAL,FLCAT,FARC,IFIX,PAUSE,HOLEZ

*FILES(103,FILEA)
FILES ALLCCATION
  103 02EA 0001 OEDO FILEA
  101 C000 0001 OEDO 02EC
  102 0001 0001 OEDO 02EC

STORAGE ALLOCATION
R 40 03BF (HEX) ADDITIONAL CORE REQUIRED
R 43 0124 (HEX) ARITH/FUNC SCAL WD CNT
R 44 06B2 (HEX) FI/O, I/O SCAL WD CNT
R 45 02B6 (HEX) DISK FI/C SCAL WD CNT
R 41 0004 (HEX) WDS UNUSED BY CORE LOAD

LIBF TRANSFER VECTOR
XMCX 09AA SCAL 1
EBCTB 0F51 SCAL 2
HCLTB 0F15 SCAL 2
GETAC 0ED2 SCAL 2
NORM 07CO
FADDX 0955 SCAL 1
FSBRX 092C SCAL 1
FMPYX 08F8 SCAL 1
FDIV 08A6 SCAL 1
FSTOX 076C
FLDX 0788
SDCOM 0978 SCAL 3
SDFX 08E3 SCAL 3
SDWRX 0901 SCAL 3
SICFX 09A6 SCAL 2
SUBSC 07A2
SICI 09AA SCAL 2
SCCMP 0983 SCAL 2
SWRT 08A2 SCAL 2
SRED 08A7 SCAL 2
FSTO 0770
FLD 078C
PRNTZ 0DF8 SCAL 2
CARDZ 0D48 SCAL 2
SFIO 09BF SCAL 2
SDFIO 0960 SCAL 3
HCLEZ 086A LOCAL
PAUSE 086A LOCAL
IFIX 086A LOCAL
FARC 086A LOCAL
FLCAT 086A LOCAL
SYSTEM SUBROUTINES
ILSO4 00C4
ILSO2 0083
ILSO1 OF56
ILSO0 OF6F
FL1PR 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 OED0. 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 in the 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 SOCA(Ls). 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 SOCA(L subroutines, and these subroutines are identified by the term SOCA(L followed by a SOCA(L overlay number in the core map. SOCA(L option 2 is used for this program because all 3 SOCA(L overlay numbers are used. SOCA(L option 1 uses SOCA(L overlay 1 and 2 only.

Under STORAGE ALLOCATION, message R40 indicates that the core load exceeds the capacity of core storage before SOCA(Ls are employed by /03BF words. Messages R43, R44, and R45 indicate that SOCA(Ls 1, 2, and 3 require /0124, /06B2, and /02B6 words of core, respectively. This information indicates that since SOCA(L 2 is much larger than SOCA(L 1, more arithmetic and function subprograms can be called at little extra cost in core. Message R41 indicates that after SOCA(Ls 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:

```
// XEG   L 2

*LCAL,FLCAT,FARC,IFIX

*FILES(103,FILEA)
FILES ALLCATION
103 02EA 0001 OECO FILEA
101 000C 0001 OECO 02EC
102 0001 0001 OECO 02EC

STORAGE ALLCATION
R 41 0008 11-EX) WCS UNUSED BY CORE LOAD

LHIF TRANSFER VECTOR
EBCTB 128F
HCLTB 1283
GETAC 1240
XMCS 1224
HCLTB 11EE
PAUSE 11C8
NCRM 11AE
FACDX 1159
FSBRX 1130
FPPXY 10FC
FCIV 10AA
FSTCX 1052
FLDX 106E
SCCGW 0842
SCFX 07AC
SCWRT 07CB
SICFX 0E26
SUBSC 1088
SICI 082A
SCCPX 0803
SWRT 0A22
SREC 0A27
FSTO 1056
FLD 1072
PRATZ 0F7B
CARDZ 0ECB
SFIO 0B3F
SCFLC 082A
IFIX 1338 LOCAL
FARC 1338 LOCAL
FLCAT 1338 LOCAL
SYSTEM SUBRCUTINES
ILS04 0CC4
ILS02 0C83
ILS01 1366
ILSOC 137F
FLIPR 12D2
04C1 (HEX) IS THE EXECUTION ADDR
```

Note that fewer LOCALs are specified, and that SLOCALs 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):

```
// XEQ   / 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
RGS11  0D80
RGMV9  0C72
RGMV3  0D50
RGCMP  0CFE
RGMV1  0C6A
PRNT1  0A9A
ZIPCO  097A
CARD0  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:

<table>
<thead>
<tr>
<th>Disk I/O subroutine in core</th>
<th>Core load origin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Decimal</td>
</tr>
<tr>
<td>DISKZ</td>
<td>510</td>
</tr>
<tr>
<td>DISK1</td>
<td>690</td>
</tr>
<tr>
<td>DISKN</td>
<td>960</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>ALLOCATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(R)</td>
<td>00CC-0016</td>
</tr>
<tr>
<td>V3(I)</td>
<td>01F2</td>
</tr>
<tr>
<td>L2(I)</td>
<td>01F8</td>
</tr>
<tr>
<td>K(I)</td>
<td>01FE</td>
</tr>
<tr>
<td>O(R)</td>
<td>01EE</td>
</tr>
<tr>
<td>M1(I)</td>
<td>01F5</td>
</tr>
<tr>
<td>N(I)</td>
<td>01FB</td>
</tr>
<tr>
<td>X(R)</td>
<td>00FO-00DE</td>
</tr>
<tr>
<td>M(I)</td>
<td>01F3</td>
</tr>
<tr>
<td>N1(I)</td>
<td>01F9</td>
</tr>
<tr>
<td>I(K)</td>
<td>01FF</td>
</tr>
<tr>
<td>V1(I)</td>
<td>01F0</td>
</tr>
<tr>
<td>M2(I)</td>
<td>01F6</td>
</tr>
<tr>
<td>I(I)</td>
<td>01FC</td>
</tr>
<tr>
<td>B(R)</td>
<td>01EC-00F2</td>
</tr>
<tr>
<td>L(I)</td>
<td>01F4</td>
</tr>
<tr>
<td>N(I)</td>
<td>01FA</td>
</tr>
<tr>
<td>I1(I)</td>
<td>02C0</td>
</tr>
<tr>
<td>V2(I)</td>
<td>01F1</td>
</tr>
<tr>
<td>L1(I)</td>
<td>01F7</td>
</tr>
<tr>
<td>J(I)</td>
<td>01FD</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>F T</th>
<th>Label</th>
<th>Operation</th>
<th>F T</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>LD</td>
<td>27</td>
<td>35</td>
<td>CHNG</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>WAIT</td>
<td>30</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
<td>33</td>
<td></td>
<td>SYSUP</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>29</td>
<td></td>
<td>LIST</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>CHNG</td>
<td>25</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>30</td>
<td></td>
<td>LIST</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>30</td>
<td></td>
<td>DC</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>30</td>
<td></td>
<td>DC</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>30</td>
<td></td>
<td>DC</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>30</td>
<td></td>
<td>DC</td>
<td>33</td>
</tr>
</tbody>
</table>

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).
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:

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>F</th>
<th>T</th>
<th>Operands &amp; Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>25</td>
<td>27</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>LIST</td>
<td>DC</td>
<td></td>
<td></td>
<td>(Assume master cartridge).</td>
</tr>
<tr>
<td>DC</td>
<td></td>
<td></td>
<td></td>
<td>(Cart. ID of logical 1).</td>
</tr>
<tr>
<td>DC</td>
<td>2222</td>
<td></td>
<td></td>
<td>(Cart. ID of logical 2).</td>
</tr>
<tr>
<td>DC</td>
<td>0000</td>
<td></td>
<td></td>
<td>(Indicates end of list).</td>
</tr>
</tbody>
</table>

The FORTRAN calling sequence for SYSUP is:

```
PAUSE 1234
(C)harge cartridges)
CALL SYSUP(a)
(P)ress PROGRAM START)
```

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.

**Note.** The entry in the array must be the decimal equivalent of the cartridge ID in hexadecimal.
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.

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.

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.

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.
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 OED0 on logical drive 0 and cartridge OED4 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 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Console Printer</td>
<td>Output only</td>
<td>120</td>
</tr>
<tr>
<td>2</td>
<td>1442 Card Read/ Punch</td>
<td>Input/output</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>1132 Printer</td>
<td>Output only</td>
<td>120 + carriage control + 120</td>
</tr>
<tr>
<td>4</td>
<td>1134/1055 Paper Tape Reader Punch</td>
<td>Input/output</td>
<td>120, plus max. of 80 case shifts for PTTC/8 code, plus NL code</td>
</tr>
<tr>
<td>5</td>
<td>1403 Printer</td>
<td>Output only</td>
<td>120 + carriage control + 120</td>
</tr>
<tr>
<td>6</td>
<td>Keyboard</td>
<td>Input only</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>1627 Plotter</td>
<td>Output only</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>2501 Card Reader</td>
<td>Input only</td>
<td>80</td>
</tr>
<tr>
<td>9</td>
<td>1442 Card Punch</td>
<td>Output only</td>
<td>80</td>
</tr>
<tr>
<td>10</td>
<td>UDISK</td>
<td>Unformatted input/output</td>
<td>320* input/output without data conversion</td>
</tr>
</tbody>
</table>

*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:

```FORTRAN
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.

This example assumes a FORTRAN program with the following specification statements:

```
1  5 10 15 20 25 30 35 40 45 50
DEFINE FILE 1(100,4.0,16)
DIMENSION R(5),T(5)
DATA R/1.0,2.0,3.0,4.0,5.0/,T/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 area) by the following *FILES control record:

```
1  5 10 15 20 25 30 35 40 45 50
*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.)

<table>
<thead>
<tr>
<th>Precision specified</th>
<th>I/O statements executed</th>
<th>Record 2 of DATA1</th>
<th>Record 1 of DATA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>* ONE WORD INTEGERS</td>
<td>DO 5 J = 1,2</td>
<td>FFFF FFFF FFFF 0002</td>
<td>FFFF FFFF FFFF 0001</td>
</tr>
<tr>
<td></td>
<td>5 WRITE (1'J)I(J)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* ONE WORD INTEGERS</td>
<td>DO 5 J = 1,2</td>
<td>0002 4000 0082 0002</td>
<td>0001 4000 0081 0001</td>
</tr>
<tr>
<td></td>
<td>5 WRITE (1'J)I(J),R(J),I(J)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* ONE WORD INTEGERS</td>
<td>WRITE (1'1)(I(J),J=1,5)</td>
<td>FFFF FFFF FFFF 0005</td>
<td>0004 0003 0002 0001</td>
</tr>
<tr>
<td>None</td>
<td>DO 5 J = 1,2</td>
<td>FFFF FFFF 0002 XXXX</td>
<td>FFFF FFFF 0001 XXXX</td>
</tr>
<tr>
<td></td>
<td>5 WRITE (1'J)I(J)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* EXTENDED PRECISION</td>
<td>DO 5 J = 1,2</td>
<td>FFFF 0002 XXXX XXXX</td>
<td>FFFF 0001 XXXX XXXX</td>
</tr>
<tr>
<td></td>
<td>5 WRITE (1'J)I(J)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* EXTENDED PRECISION</td>
<td>DO 5 J = 1,2</td>
<td>FFFF 0082 4000 0000</td>
<td>FFFF 0081 4000 0000</td>
</tr>
<tr>
<td></td>
<td>5 WRITE (1'J)R(J)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* EXTENDED PRECISION</td>
<td>WRITE (1'1)(I(1),R(1),I(2))</td>
<td>FFFF FFFF FFFF 0002</td>
<td>0081 4000 0000 0001</td>
</tr>
</tbody>
</table>

**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 MLI 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

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.
Data File Processing

**sequential 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

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.

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.

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.

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.
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.

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:

$$E = \frac{320}{X} \text{ (disregard any remainder)}$$

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 of words; that is, 49 characters require 25 words.

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.

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:

<table>
<thead>
<tr>
<th>File label</th>
<th>Index</th>
<th>Prime data area</th>
<th>Overflow area</th>
</tr>
</thead>
</table>

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:

<table>
<thead>
<tr>
<th>Word number</th>
<th>Label entry description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Key length</td>
</tr>
<tr>
<td>2</td>
<td>Record length</td>
</tr>
<tr>
<td>3</td>
<td>Number of index entries per sector</td>
</tr>
<tr>
<td>4</td>
<td>Index entry length</td>
</tr>
<tr>
<td>5</td>
<td>Number of records per sector</td>
</tr>
<tr>
<td>6</td>
<td>Record number of last prime data record</td>
</tr>
<tr>
<td>7</td>
<td>Index entry number of last entry in file</td>
</tr>
<tr>
<td>8</td>
<td>Sector address of last prime data record</td>
</tr>
<tr>
<td>9</td>
<td>Sector address of last index entry</td>
</tr>
<tr>
<td>10</td>
<td>Sector address of next overflow record</td>
</tr>
<tr>
<td>11</td>
<td>Record number of next overflow record</td>
</tr>
</tbody>
</table>
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.
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:

<table>
<thead>
<tr>
<th>Key</th>
<th>Zeros</th>
<th>Zeros</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The last data record on a prime data cylinder that has overflowed is

<table>
<thead>
<tr>
<th>Key</th>
<th>Zeros</th>
<th>X'FFFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>520</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Key 154</th>
<th>Zeros</th>
<th>Zeros</th>
<th>Key 153</th>
<th>Zeros</th>
<th>Zeros</th>
<th>Overflow sector address</th>
<th>Reg 0001</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 FORAssembler 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.
### Double Buffering

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>Operation</th>
<th>Operands &amp; Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>L.I.BF</td>
<td>READ0</td>
<td>PRIME, DOUBLE-BUFFERED</td>
</tr>
<tr>
<td>DC</td>
<td>/1000</td>
<td>/<em>CARD READING, THIS READ PERFORMED ONCE ONLY</em>/</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>L.I.BF</td>
<td>READ0</td>
<td>THIS READ WILL NOT START</td>
</tr>
<tr>
<td>DC</td>
<td>/1000</td>
<td><em>/UNTIL PREVIOUS READ</em></td>
<td></td>
</tr>
<tr>
<td>SET1</td>
<td>DC</td>
<td>BUF1</td>
<td><em>IS COMPLETED</em></td>
</tr>
<tr>
<td>SET2</td>
<td>DC</td>
<td>BUF1+1</td>
<td>INPUT AREA ADDRESS</td>
</tr>
<tr>
<td>SET3</td>
<td>DC</td>
<td>BUF1+1</td>
<td>OUTPUT AREA ADDRESS</td>
</tr>
<tr>
<td>CALL</td>
<td>HLEB,C</td>
<td>CONVERSION TABLE FOR ZIPOCO</td>
<td></td>
</tr>
<tr>
<td>LDD</td>
<td>BFADR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>SET1</td>
<td>CHANGE READ BUFFER ADDRESS</td>
<td></td>
</tr>
<tr>
<td>RTE</td>
<td>16</td>
<td>EXCHANGE BUFFER ADDRESSES</td>
<td></td>
</tr>
<tr>
<td>STD</td>
<td>BFADR</td>
<td>FOR NEXT TIME THRU LOOP</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>SET2</td>
<td>CHANGE INPUT AND OUTPUT BFR</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>SET3</td>
<td><em>ADDRESSES FOR CONVERSION</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODING FOR REQUIRED PROCESSING SHOULD FOLLOW</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BCE</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONE</td>
<td>DC</td>
<td>1</td>
<td>CONSTANT VALUE OF 1</td>
</tr>
<tr>
<td>BUF1</td>
<td>DC</td>
<td>80</td>
<td>WORD COUNT FOR CARD BFR 1</td>
</tr>
<tr>
<td>BSS</td>
<td>80</td>
<td>CARD BUFFER 1</td>
<td></td>
</tr>
<tr>
<td>BUF2</td>
<td>DC</td>
<td>80</td>
<td>WORD COUNT FOR CARD BFR 2</td>
</tr>
<tr>
<td>BSS</td>
<td>80</td>
<td>CARD BUFFER 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>THE FOLLOWING PAIR OF ADDRESSES ARE EXCHANGED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EACH TIME THROUGH THE CARD READING LOOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSS</td>
<td>E</td>
<td>MAKE NEXT LOCATION EVEN</td>
</tr>
<tr>
<td>BFADR</td>
<td>DC</td>
<td>BUF1</td>
<td>ADDRESS OF CARD BUFFER 1</td>
</tr>
<tr>
<td>DC</td>
<td>BUF2</td>
<td>ADDRESS OF CARD BUFFER 2</td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** The above code snippet is a simple example of how to use double buffering in an assembler program to read from a card reader. The double buffering technique helps in managing the data flow between the card reader and the main memory, ensuring smooth and efficient data processing.

---

** assembler programmer tips **

---

**double buffering**

---

**label**

---

**operation**

---

**operands & remarks**

---

**Example:**

- **READ L.I.BF**
  - **READ0**
  - **PRIME, DOUBLE-BUFFERED**

- **DC /1000**
  - **CARD READING, THIS READ**
  - **PERFORMED ONCE ONLY**

- **READ L.I.BF**
  - **READ0**
  - **THIS READ WILL NOT START**

- **DC /1000**
  - **UNTIL PREVIOUS READ**

- **SET1 DC BUF1**
  - **IS COMPLETED**

- **SET2 DC BUF1+1**
  - **INPUT AREA ADDRESS**

- **SET3 DC BUF1+1**
  - **OUTPUT AREA ADDRESS**

- **CALL HLEB,C**
  - **CONVERSION TABLE FOR ZIPOCO**

- **LDD BFADR**
- **STO SET1**
- **RTE 16**
- **STD BFADR**
- **A ONE**
- **STO SET2**
- **STO SET3**
- **B READ**
- **ONE DC 1**
- **BUF1 DC 80**
- **BSS 80**
- **BUF2 DC 80**
- **BSS 80**
- **BSS E 0**
- **MAKE NEXT LOCATION EVEN**
- **BFADR DC BUF1**
- **DC BUF2**
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.

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.

<table>
<thead>
<tr>
<th>ISS number</th>
<th>Device</th>
<th>Device interrupt level assignments</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1442 Card Reader Punch</td>
<td>0, 4</td>
<td>+4, +7</td>
</tr>
<tr>
<td>2</td>
<td>Input keyboard/console printer</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>3</td>
<td>1134/1055 Paper Tape Reader/Punch</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>4</td>
<td>2501 Card Reader</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>5</td>
<td>Disk storage</td>
<td>2</td>
<td>+5</td>
</tr>
<tr>
<td>6</td>
<td>1132 Printer</td>
<td>1</td>
<td>+4</td>
</tr>
<tr>
<td>7</td>
<td>1627 Plotter</td>
<td>3</td>
<td>+4</td>
</tr>
<tr>
<td>8</td>
<td>Synchronous Communications Adapter</td>
<td>1</td>
<td>+4</td>
</tr>
<tr>
<td>9</td>
<td>1403 Printer</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>10</td>
<td>1231 Optical Mark Page Reader</td>
<td>4</td>
<td>+4</td>
</tr>
<tr>
<td>11</td>
<td>2250 Display Unit</td>
<td>3</td>
<td>+4</td>
</tr>
</tbody>
</table>

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.

```
CC001  ********************************************
CC002  TITLE- READC
CC003  FUNCTION/OPERATION-
CC004  THIS 1130 SUBROUTINE OPERATES THE PRIMARY
CC005  2501 CARD READER, IT INITIATES REQUESTED
CC006  OPERATIONS, PROCESSES OPERATION COMPLETELY
CC007  INTERRUPTS, AND AUTOMATICALLY INITIATES
CC008  ERROR RECOVERY PROCEDURES.
CC009  ENTRY POINTS-
CC010  1. READC CALL ENTRANCE FOR TEST OR READ
CC011  OPERATIONS. E.G. LIBF READO
CC012  DC /1000
CC013  DC ICBUF
CC014  DC ICHUF
CC015  I
CC016  2. RE048 OPERATION COMPLETE INTERRUPT ENTRY
CC017  CALL. 
CC018  INPUT- NONE OTHER THAN FROM THE PARAMETERS IN
CC019  LIBF CALLING SEQUENCE.
CC020  OUTPUT- Routines will transfer 0 to 80 cols from
CC021  CARD TC I/C Helper area as specified by calling
CC022  SEQUENCE. FORMAT IS 12 BITS PER BUFFER WORD
CC023  LEFT JUSTIFIED.
CC024  EXTERNAL SUBROUTINES- NONE.
CC025  EXITS- 
CC026  NORMAT-
CC027  1. RE180 IF NC PRE-CP ERROR HAS BEEN DETECTED, THE EXIT FROM RE180 IS
CC028  TC THE CALLER AFTER THE REQUESTED
CC029  2501 operation has been initiated
CC030  2. RE348 THE EXIT FROM RE348 IS BACK TO THE
CC031  CALLER VIA ILSO4 AFTER CP-COMPLETE,
CC032  PROCESSING HAS BEEN FINISHED.
CC033  ERROR-
CC034  1. RE180 IF A PRE-CP ERROR OR NOT READY
CC035  CONDITION IS DETECTED, SUBROUTINE
CC036  WILL BRANCH TO HEX 0029 VIA RE180
CC037  AND DISPLAY ONE OF TWO CODES IN
CC038  ACCUMULATOR.
CC039  4000 IS DISPLAYED IF 2501 IS NOT
CC040  READY, 4001 IS DISPLAYED IF AN
CC041  ERROR IS DISCOVERED IN CALLING
CC042  PARAMETERS OR AREAS REFERENCED BY
CC043  THEM.
CC044  TABLES/WORK AREAS- NONE.
CC045  ATTRIBUTES- RELASABLE, CAN READ UP TO 80 COLUMNS
CC046  IF BINARY DATA. IF A WORD COUNT
CC047  CF ZEROC IS SPECIFIED, THE READ
CC048  CFPFRACT ACTS AS A FEED.
CC049  /*ASM
CC050  *XREF
CC051  *LEVEL 4
```

6-38
<table>
<thead>
<tr>
<th>SYMBOLS</th>
<th>VALUE</th>
<th>REL</th>
<th>CEFA</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE60</td>
<td>CC0C</td>
<td>1</td>
<td>CO055</td>
<td>CCC51,R</td>
</tr>
<tr>
<td>RE68</td>
<td>CC04</td>
<td>1</td>
<td>CO058</td>
<td>CO154,B</td>
</tr>
<tr>
<td>RE74</td>
<td>CC07</td>
<td>1</td>
<td>CO072</td>
<td>CCC57,B</td>
</tr>
<tr>
<td>RE79</td>
<td>CO12</td>
<td>1</td>
<td>CO082</td>
<td>CCC77,B</td>
</tr>
<tr>
<td>RE84</td>
<td>CO15</td>
<td>1</td>
<td>CO084</td>
<td>CCC78,B</td>
</tr>
<tr>
<td>RE56</td>
<td>CO1P</td>
<td>1</td>
<td>CO086</td>
<td>CCC1C,B</td>
</tr>
<tr>
<td>RE108</td>
<td>CO24</td>
<td>1</td>
<td>CO094</td>
<td>CCC90,B</td>
</tr>
<tr>
<td>RE120</td>
<td>CO28</td>
<td>1</td>
<td>CO1C0</td>
<td>CO01,B</td>
</tr>
<tr>
<td>RE132</td>
<td>CO2C</td>
<td>1</td>
<td>CO1C2</td>
<td>CO115,B</td>
</tr>
<tr>
<td>RE144</td>
<td>CO2E</td>
<td>1</td>
<td>CO1C3</td>
<td>CO055,M</td>
</tr>
<tr>
<td>RE156</td>
<td>CO30</td>
<td>1</td>
<td>CO1C4</td>
<td>CO074,M</td>
</tr>
<tr>
<td>RE168</td>
<td>CO32</td>
<td>1</td>
<td>CO1C5</td>
<td>CCC73,M</td>
</tr>
<tr>
<td>RE180</td>
<td>CO33</td>
<td>1</td>
<td>CO1C6</td>
<td>CO1C2,M</td>
</tr>
<tr>
<td>RE192</td>
<td>CO35</td>
<td>1</td>
<td>CO1C7</td>
<td>CO083,B</td>
</tr>
<tr>
<td>RE204</td>
<td>CC37</td>
<td>1</td>
<td>CO1C9</td>
<td>CO087,B</td>
</tr>
<tr>
<td>RE216</td>
<td>CC3B</td>
<td>1</td>
<td>CO112</td>
<td>CO108,B</td>
</tr>
<tr>
<td>RE228</td>
<td>CO4C</td>
<td>1</td>
<td>CO119</td>
<td>CCC78,R</td>
</tr>
<tr>
<td>RE264</td>
<td>CO48</td>
<td>1</td>
<td>CO121</td>
<td>CO082,R</td>
</tr>
<tr>
<td>RE252</td>
<td>CO43</td>
<td>1</td>
<td>CO122</td>
<td>CO147,R</td>
</tr>
<tr>
<td>RE264</td>
<td>CO44</td>
<td>1</td>
<td>CO123</td>
<td>CO096,M</td>
</tr>
<tr>
<td>RE276</td>
<td>CO46</td>
<td>1</td>
<td>CO125</td>
<td>CO092,R</td>
</tr>
<tr>
<td>RE288</td>
<td>CO47</td>
<td>1</td>
<td>CO126</td>
<td>CO086,R</td>
</tr>
<tr>
<td>RE300</td>
<td>CO48</td>
<td>1</td>
<td>CO127</td>
<td>CO111,R</td>
</tr>
<tr>
<td>RE312</td>
<td>CO49</td>
<td>1</td>
<td>CO128</td>
<td>CO107,R</td>
</tr>
<tr>
<td>RE324</td>
<td>CO4A</td>
<td>1</td>
<td>CO129</td>
<td>CCC72,M</td>
</tr>
<tr>
<td>RE336</td>
<td>CO4B</td>
<td>1</td>
<td>CO147</td>
<td>CO059,B</td>
</tr>
<tr>
<td>RE348</td>
<td>CO54</td>
<td>1</td>
<td>CO154</td>
<td>CO158,B</td>
</tr>
<tr>
<td>RE360</td>
<td>CO56</td>
<td>1</td>
<td>CO155</td>
<td>CO149,B</td>
</tr>
<tr>
<td>RE385</td>
<td>CO58</td>
<td>1</td>
<td>CO159</td>
<td>CO156,B</td>
</tr>
<tr>
<td>$1CCT</td>
<td>CO32</td>
<td>0</td>
<td>CO131</td>
<td>CO097,M</td>
</tr>
<tr>
<td>$PRET</td>
<td>CO20</td>
<td>0</td>
<td>CO130</td>
<td>CO113,M</td>
</tr>
<tr>
<td>$PSBT</td>
<td>CO08C</td>
<td>0</td>
<td>CO132</td>
<td>CO16C,B</td>
</tr>
</tbody>
</table>

COC OVERFLOW SECTORS SPECIFIED
COC OVERFLOW SECTORS REQUIRED
032 SYMBOLS DEFINED
NC ERROR(S) AND NO WARNING(S) FLAGGED IN ABOVE ASSEMBLY
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)
  ...
  ...
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.
Assembler Programmer Tips

// ASM
*XREF

00001 00020
00002 00030
00003 00040
00004 00050
00005 00060
00006 00070
00007 00080
00008 00090
00009 00100
00010 00110
00011 00120
00012 00130
00013 00140
00014 00150
00015 00160
00016 00170
00017 00180
00018 00190
00019 00200
00020 00210
00021 00220
00022 00230
00023 00240
00024 00250
00025 00260
00026 00270
00027 00280
00028 00290
00029 00300
00030 00310
00031 00320
00032 00330
00033 00340
00034 00350
00036 00370
00037 00380
00038 00390
00039 00400
00040 00410
00041 00420
00042 00430
00043 00440
00044 00450
00045 00460
00046 00470
00047 00480
00048 00490
00049 00500
00050 00510
00051 00520
00052 00530
00053 00540
00054 00550
00055 00560
00056 00570
00057 00580
00058 00590
00059 00600
00060 00610
00061 00620
00062 00630
00063 00640
00064 00650
00065 00660
00066 00670
00067 00680
00068 00690
00069 00700
00070 00710
00071 00720

ILS 04

1X410 DC /0033 DEVICO ** AND Iss NO. ** U1J00320
DC /0033 DEVICO ** AND Iss NO. ** U1J00330
DC /0033 DEVICO ** AND Iss NO. ** U1J00340
DC /0430 1231 +4 AND Iss NO. 10 U1J00350
DC /0430 1403 +4 And Iss NO. 9 U1J00360
DC /0430 2501 +4 And Iss NO. 4 U1J00370
DC /0734 1442 +7 AND Iss NO. 1 U1J00380
DC /0435 Console +4 AND Iss NO. 2 U1J00390
DC /0436 1134/1055 +4 AND Iss NO. 3 U1J00400

~IX420 DC 0 INTERRUPT ENTRY U1J00420

STD IX480 SAVE ACC AND EXTENSION, U1J00430
STS IX430 *STATUS, U1J00440
STX 1 IX441+1 *XR1, U1J00450
STX 2 IX442+1 *XR2, U1J00460
STX 3 IX443+1 *XR3, U1J00470
LDX 13 $XR3X POINT TO TRANSFER VECTOR U1J00490
XIO IX495-1 SENSE KEYSBOARD U1J00490
SLA 2 IS IT INTERRUPT REQUEST U1J00500
BSI 1 $IREQ,+2 *KEY, BR IF YES U1J00510

Nop U1J00520

LDX 1 9 NUMBER OF DEVICES ON LEVEL, U1J00530
XIO IX490-1 SENSE ILSW U1J00540
SLCA 1 0 SHIFT AND DECREMENT XR1 U1J00550
BS1 11 IX410-1 BR TO DEVICE ISS U1J00560

LDS 0 RESTORE STATUS, U1J00580
IX441 LDIX L1 ** *XR1, U1J00590
IX442 LDIX L2 ** *XR2, U1J00600
IX443 LDIX L3 ** *XR3, U1J00610

LDD IX480 *ACC AND EXTENSION U1J00620

BSC 1 IX420 TURN OFF INTERRUPT, RETURN U1J00630
IX420 BSS 2 ACCUMULATOR AND EXTENSION U1J00650

LOCC TO SENSE ILSW U1J00660
DC 0

IX490 DC /0300 I/OCC TO SENSE ILSW U1J00670
DC 0

IX495 DC /0F00 SENSE I/OCC FOR KEYBOARD U1J00680
SIREQ EQU /002C ADD OF ISS FOR INT REQ U1J00690
$XR3X EQU /00E4 ADDR OF TRANSFER VECTOR U1J00710

END
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 $1420 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, TYPEO, or WRTYO 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, TYPEO, or WRTYO.

*These programs may not be LOCALed when $IREQ has been modified.*
### Assembler Programmer Tips

**INT REQ service subroutine**

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>F</th>
<th>T</th>
<th>Operands &amp; Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td></td>
<td>27</td>
<td>30</td>
<td>XXXXXXXXXXXXXXXXXX</td>
</tr>
<tr>
<td></td>
<td>A CALL TO THIS SUBROUTINE WILL CHANGE THE CONTENTS OF $1.REQ IN THE RESIDENT MONITOR. IF THE INTERRUPT REQUEST KEY IS PRESSED AFTER A CALL TO THIS SUBROUTINE, ANY CORE LOAD AND WILL PREVENT FLUSHING TO THE NEXT JOB IF THE INT REQ KEY IS PRESSED.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>35</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>45</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>55</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>65</td>
<td>70</td>
<td></td>
</tr>
</tbody>
</table>

- **ENT**: I.REQ
- **EREQ**: ENTRY POINT
- **STX**: 1 R0,0+1, SAVE XR1
- **LDX**: L1 INTR, SET XR1-ADDR OF INTR PORTN
- **STX**: L1 $1.REQ, SET INTERRUPT BRANCH ADDR
- **IRO10**: LDX L1 X-X, RESET XR1
- **BSC**: I $1.REQ, RETURN TO CALLING PGM
- **$1.REQ EQU**: /0,02C
- **INT**: ENTRY POINT FROM ILS
- **X.IO**: IN9.0, READ THE CONSOLE SWITCHES
- **LD**: IN9.30, LD WHAT HAS BEEN READ
- **BS1**: L $1420,Z+ FLUSH TO NEXT JOB IF NEG
- **X.IO**: IN9.20, SENSE KEYBOARD, WITH RESET
- **MDX**: L INTR,6, INCREMENT, RETURN ADDR
- **BSC**: I INTR, RETURN TO ILS
- **BSS**: E 0 CREATE, EVEN, ADDR
- **IN91.0**: DC, IN9.30, TO READ CONSOLE
- **DC**: 1,3400, XSWITCHES
- **IN92.0**: DC, X-X, TO SENSE KEYBOARD, WITH
- **DC**: 1,0E0Z, XRESET
- **IN93.0**: EQU IN9.20, VALUE READ FROM CONSOLE SW
- **$1420**: EQU /0,02C
- **END**: 1,0E0Z

**Figure 6-3. INT REQ service subroutine for any core load**
### Assembler Programmer Tips

**INT REQ service subroutine**

<table>
<thead>
<tr>
<th>Label</th>
<th>Operation</th>
<th>F</th>
<th>T</th>
<th>Operands &amp; Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>A CALL TO THE SUBROUTINE WILL CHANGE THE CONTENTS OF $.REQ IN THE RESIDENT MONITOR...</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>THE INTERRUPT REQUEST KEY IS PRESSED, AFTER A CALL TO THIS SUBROUTINE HAS BEEN EXECUTED...</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>BRANCH TO THE SECOND PART OF THE SUBROUTINE WILL TAKE PLACE... THIS SUBROUTINE CAN ONLY BE...</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>THE CORE LOAD, AND WILL PREVENT FLUSHING TO THE NEXT JOB IF THE INT. REQ. KEY IS PRESSED...</td>
<td>65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*ENT* 1,REQ

*THIS PORTION WILL BE ENTERED WHEN A CALL 1,REQ IS EXECUTED...*

1,REQ DC  X, Entry Point

STX 1,IR010 1, Save XR1

LDX L1 INTR SET XR1: ADDR OF INTR PORTN

STX L1 $1,REQ SET INTERRUPT BRANCH ADDR

IR010  LDX L1 X,  Reset XR1

BSC 1, I,REQ RETURN TO CALLING PGM

$1,REQ 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  X, Entry Point FROM ILS

XI0 IN910 READ THE CONSOLE SWITCHES

LD IN920 LD WHAT HAS BEEN READ

BSI L $1420, Z+ FLUSH TO NEXT JOB IF NEG

MDX L INTR, 1 INCREMENT THE RETURN ADDR

BSC 1 INTR RETURN TO ILS

BSS 0  CREATE EVEN ADDR

IN910 DC  IN920 LOCC TO READ CONSOLE

DC /3A00 *SWITCHES

IN920 DC  $1420 VALUE READ FROM CONSOLE SW

$1420 EQU /0026

END

---

Figure 6-4. INT REQ service subroutine for core load using TYPEZ, WRTYZ, TYPE0, or WRTY0

Programming Tips and Techniques 6-47
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:

<table>
<thead>
<tr>
<th>1132 Printer</th>
<th>1403 Printer</th>
<th>2501 Card Reader</th>
<th>1442 Card Reader Punch</th>
<th>Console printer keyboard</th>
<th>1055 Punch 1134 Reader</th>
<th>1627 Plotter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRNTZ</td>
<td>PRNZ</td>
<td>READZ</td>
<td>CARDZ</td>
<td>TYPEZ</td>
<td>*VCHRI,WCHRI</td>
<td>Input only</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TYPEZ</td>
<td>PAPTZ</td>
<td>Output only</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WRTYZ</td>
<td>PAPTZ</td>
<td>*VCHRI,WCHRI</td>
<td>Output only</td>
</tr>
</tbody>
</table>

*VCHRI — extended precision
  WCHRI — standard precision
The following lists the possible entries in a FORTRAN *IOCS control record and the subroutine each entry implies:

<table>
<thead>
<tr>
<th>*IOCS entry</th>
<th>Subroutine called</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD</td>
<td>CARDZ</td>
</tr>
<tr>
<td>2501 READER</td>
<td>READZ</td>
</tr>
<tr>
<td>1442 PUNCH</td>
<td>PNCHZ</td>
</tr>
<tr>
<td>TYPEWRITER</td>
<td>WRTYZ</td>
</tr>
<tr>
<td>KEYBOARD</td>
<td>TYPEZ</td>
</tr>
<tr>
<td>1132 PRINTER</td>
<td>PRNTZ</td>
</tr>
<tr>
<td>1403 PRINTER</td>
<td>PRNZ</td>
</tr>
<tr>
<td>PAPER TAPE</td>
<td>PAPTZ</td>
</tr>
<tr>
<td>PLOTTER</td>
<td>PLOTX</td>
</tr>
<tr>
<td>DISK</td>
<td>DISKZ</td>
</tr>
<tr>
<td>UDISK</td>
<td>DISKZ</td>
</tr>
</tbody>
</table>

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.

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.

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:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backspace</td>
<td>5696</td>
</tr>
<tr>
<td>Carrier return</td>
<td>5440</td>
</tr>
<tr>
<td>Line feed</td>
<td>9536</td>
</tr>
<tr>
<td>Shift to print black</td>
<td>5184</td>
</tr>
<tr>
<td>Shift to print red</td>
<td>13632</td>
</tr>
<tr>
<td>Space</td>
<td>16448</td>
</tr>
<tr>
<td>Tabulate</td>
<td>1344</td>
</tr>
</tbody>
</table>

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:

```
WRITE (L,3)X,1,X,Y,K
L=1
3 FORMAT (F12.6,Z14,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 //16 record read by CARDZ, READZ, or PAPTZ during a FORTRAN program execution causes an immediate CALL EXIT. Only the //16 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 //16 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. 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).
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
Finding Devices

1131 CPU

READYING THE 1131 CENTRAL PROCESSING UNIT (with an internal disk)

Operator 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.

2. Insert a cartridge in the single disk drive.

3. Move the DISK switch on the disk drive to ON. The disk drive requires approximately 90 seconds to reach operating speed.

System response or Error indicator and corrective action

If the FORMS CHECK light comes on, insert or adjust the paper in the console printer. If the DISK UNLOCK light comes on, it indicates that the DISK switch on the disk drive is set to OFF. See step 3.

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.

READYING THE 1131 CENTRAL PROCESSING UNIT (without an internal disk)

Operator 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.

2. Ready the 2311 Disk Storage Drives as described under "Readying the 2311 Disk Storage Drive" in this chapter.

System response or Error indicator and corrective action

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.

*These indicators are blank on an 1131 CPU that does not contain an internal single disk drive.
READYING THE 2310 DISK STORAGE DRIVE

Operator action

1. Be sure system power is turned on.

2. Insert the disk cartridges

3. Move the START/STOP switch to START.

4. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.

5. Move the START/STOP switch to START position for the cartridges being used. The drives require approximately 90 seconds to reach operating speed.

6. Move the ENABLE/DISABLE switch on the disk storage drive to ENABLE.

System response or Error indicator and corrective action

If the CARTRIDGE UNLOCKED light comes on, it indicates that the START/STOP switch is set to STOP. See step 3.

The READY light on the 1133 is on.

The indicators showing the drive numbers come on when the disks reach operating speed.

READYING THE 2311 DISK STORAGE DRIVE

Operator 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.

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.

5. Move the ENABLE/DISABLE switch on the disk storage drive to the ENABLE position.

System response or Error indicator and corrective action

The READY light on the 1133 is on.

The green indicator showing the drive number comes on when the disks reach operating speed.
READYING THE 1132 PRINTER

Operator action

1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

System response or Error indicator and corrective action
The printer POWER ON light comes on.
If the printer FORMS CHECK light comes on, insert or adjust the paper in the printer.

The READY light comes on.

READYING THE 1403 PRINTER

Operator 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.
3. Press the CARRIAGE RESTORE key on the printer.
4. Press START.

System response or Error indicator and corrective action
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).
The READY light on the 1133 is on.
The PRINT READY light comes on.
READYING THE 1442 MODEL 6 AND 7 CARD READ PUNCH

Operator action

1. Be sure system power is turned on.

2. Press the NPRO key.

3. Place the cards to be processed in the hopper, face down, 9-edge first.

4. Press the START key.

System response or Error indicator and corrective action
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).
The HOPR light goes off.

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.
ATTENTION
READ
CHECK
Readying Devices
2501 Card Reader
1134 Paper Tape Reader

READYING THE 2501 CARD READER

Operator action
1. Be sure system power is turned on.
2. Press NPRO.
3. Place cards to be processed in the hopper, face down, S-edge first.
4. Press START.

System response or Error indicator and corrective action
The card reader POWER ON and FEED CHECK lights are on.
If any other indicators are on, correct the condition (see Appendix B).
The FEED CHECK light goes off.

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.

Delete codes
Feed holes

System response or Error indicator and corrective action
Tape movement
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 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.
5. Press RESET.
6. Press START.
7. Press START again.

The PROGRAM LOAD light comes on.
The hopper is raised to the ready position. The RESET light goes off and the START light comes on.
The PROGRAM LOAD light goes off.
The START light goes off. All indicator lights should be off, with one exception: the SYSTEM STOP light can be on.

Operating the 1130 Disk Monitor System 7-9
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:

<table>
<thead>
<tr>
<th>LOG DRIVE</th>
<th>CART SPEC</th>
<th>CART AVAIL</th>
<th>PHY DRIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXX</td>
<td>XXXX</td>
<td>XXXX</td>
<td>XXXX</td>
</tr>
</tbody>
</table>

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.
Cold Start
Card System
Paper Tape System

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
   - 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. 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.
Entering Jobs from the Card Reader

1. Place the cards to be processed in the card hopper, face down, 9-edge first, and press \texttt{START} on the card reader.
2. Check that the console mode switch is set to \texttt{RUN}.
3. Press \texttt{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 \texttt{START} on the card reader and \texttt{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 \texttt{RUN}.
3. Press \texttt{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."

When the // TYP control record is read, the console printer performs a carriage return and the \texttt{KB SELECT} light on the keyboard operator's panel comes on. The system is now in a \texttt{WAIT} state at $\texttt{PRET}$ with /2002 in the accumulator, awaiting keyboard input.

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 \texttt{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 \texttt{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 *DELETA, d want to change it to *DEFINE.

1. Press the backspace key 3 times. (The T is slashed: *DELET.)
2. Enter the correct characters. (The corrected line appears as *DELETFINE 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.

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 TYPEZ 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.

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.

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

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.

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.

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. Because the keyboard remains selected during interrupt request processing when in // TYP mode, you must be careful not to press any keys until the /2002 halt at SPRET is displayed.

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

To select a specific core location to be displayed or altered:

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.

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.

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.

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, $DSBY, 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

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 unusable
- The assembler and/or any of the compilers are to be loaded onto a system cartridge

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, and 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**

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>SCON or TERM</td>
</tr>
<tr>
<td>5 through 80</td>
<td>Blanks</td>
</tr>
</tbody>
</table>
Phase Identification (PHID) Control Records

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.

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>PHID</td>
</tr>
<tr>
<td>6 through 8</td>
<td>IDs of the first and last phases of DUP and 10 through 12</td>
</tr>
<tr>
<td>14 through 16</td>
<td>IDs of the first and last phases of the FORTRAN compiler and 18 through 20</td>
</tr>
<tr>
<td>22 through 24</td>
<td>IDs of the first and last phases of the COBOL compiler and 26 through 28</td>
</tr>
<tr>
<td>30 through 32</td>
<td>IDs of the first and last phases of the supervisor and 34 through 36</td>
</tr>
<tr>
<td>38 through 40</td>
<td>IDs of the first and last phases of the core load builder and 42 through 44</td>
</tr>
<tr>
<td>46 through 48</td>
<td>IDs of the first and last phases of the system I/O device subroutines and 50 through 52</td>
</tr>
<tr>
<td>54 through 56</td>
<td>IDs of the first and last phases of the core image loader and 58 through 60</td>
</tr>
<tr>
<td>64</td>
<td>1 (indicates continuation to the second PHID card)</td>
</tr>
<tr>
<td>66 through 68</td>
<td>Vxx (where xx is the disk monitor system version number)</td>
</tr>
<tr>
<td>70 through 72</td>
<td>Mxx (where xx is the version modification number)</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Card identification and sequence number</td>
</tr>
</tbody>
</table>

Note: All card columns omitted in this format contain blanks.
<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>PHID</td>
</tr>
<tr>
<td>6 through 8 and 10 through 12</td>
<td>IDs of the first and last phases of the RPG compiler</td>
</tr>
<tr>
<td>14 through 16 and 18 through 20</td>
<td>IDs of the first and last phases of DUP, part 2</td>
</tr>
<tr>
<td>22 through 24 and 26 through 28</td>
<td>IDs of the first and last phases of the macro assembler</td>
</tr>
<tr>
<td>29 through 65</td>
<td>Blanks</td>
</tr>
<tr>
<td>66 through 68</td>
<td>Vxx (where xx is the disk monitor system version number)</td>
</tr>
<tr>
<td>70 through 72</td>
<td>Mxx (where xx is the version modification number)</td>
</tr>
<tr>
<td>73 through 80</td>
<td>Card identification and sequence number</td>
</tr>
</tbody>
</table>

**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.
<table>
<thead>
<tr>
<th>Phase number</th>
<th>Program or program phase name</th>
<th>ID starting in column 73</th>
<th>Phase number</th>
<th>Program or program phase name</th>
<th>ID starting in column 73</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td>RES SKELETON SUPY, Part of COMMA, DISKZ, COLD system START PROGRAM</td>
<td>30 FOR EXPANDER II PHASE K18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EMN loader</td>
<td>31 FOR DATA ALLOCATION PHASE K19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td>SYS LDR—PHASE 2—OVERLAY 0</td>
<td>32 FOR COMPILATION ERROR PHASE K20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td>SYS LDR—PHASE 2—OVERLAY 1</td>
<td>33 FOR STATEMENT ALLOCATION PHASE K21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td>SYS LDR—PHASE 2—OVERLAY 2</td>
<td>34 FOR LIST STATEMENT ALLOCATION PHASE K22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX</td>
<td>SYS LDR—PHASE 2—OVERLAY 3</td>
<td>35 FOR LIST SYMBOL TABLE PHASE K23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>36 FOR LIST CONSTANTS PHASE K24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 FOR OUTPUT I PHASE K25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>38 FOR OUTPUT II PHASE K26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>39 FOR RECOVERY (EXIT) PHASE K27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>51 PHASE NUMBERS USED BY THE COBOL COMPILER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dup</td>
<td>DUP COMMON SUBROUTINES, CCAT</td>
<td>J01 SUP PHASE 1—MONITOR CONTROL RECORD ANALYZER NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>DUP CTRL RECORD PROCESSOR</td>
<td>J02 SUP PHASE 2—JOB CONTROL RECORD PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>DUP STORE PHASE</td>
<td>J03 SUP PHASE 3—DELETE TEMPORARILY STORED PROGRAM LET N01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>DUP *FILES, *LOCAL, *NOCAL PHASE</td>
<td>J04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>DUP DUMP PHASE</td>
<td>J05 SUP PHASE 4—XEQ CONTROL RECORD PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>DUP DUMP LET/FLET PHASE</td>
<td>J06 SUP PHASE 5—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>DUP DELETE PHASE</td>
<td>J07 SUP PHASE 6—XEQ CONTROL RECORD PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>DUP DEFINE PHASE</td>
<td>J08 SUP PHASE 7—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>DUP EXIT PHASE</td>
<td>J09 SUP PHASE 8—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>DUP CARD I/O INTERFACE</td>
<td>J10 SUP PHASE 9—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>DUP PAPER TAPE I/O INTERFACE</td>
<td>J11 SUP PHASE 10—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0B</td>
<td>DUP UPCOR PHASE SAVED BY DUMP</td>
<td>J12 SUP PHASE 11—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>DUMP DUMP PHASE</td>
<td>J17 SUP PHASE 12—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>DUMP EXIT PHASE</td>
<td>J17 SUP PHASE 13—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>DUMP PRINCIPAL INPUT WITH KEYBOARD</td>
<td>J17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>DUMP PRINCIPAL W/O KEYBOARD</td>
<td>J17 SUP PHASE 14—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DUMP PAPER TAPE I/O</td>
<td>J17 SUP PHASE 15—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>DUMP STORE CI</td>
<td>J17 SUP PHASE 16—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DUMP MODIF DUMMY PHASE</td>
<td>J17 SUP PHASE 17—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORTRAN</td>
<td>FOR INPUT PHASE</td>
<td>K01 SUP PHASE 18—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>compiler</td>
<td>FOR CLASSIFIER PHASE</td>
<td>K02 SUP PHASE 19—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>FOR CHECK ORDER/STMNT NO. PHASE</td>
<td>K03 SUP PHASE 20—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>FOR COMMON SUBR OR FUNCTION PHASE</td>
<td>K04 SUP PHASE 21—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>FOR DIMENSION, REAL, INTEGER</td>
<td>K05 SUP PHASE 22—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>FOR REAL CONSTANT PHASE</td>
<td>K06 SUP PHASE 23—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>FOR DEFINE FILE, CALL LINK EXIT</td>
<td>K07 SUP PHASE 24—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>FOR VARIABLE, STMNT FUNC PHASE</td>
<td>K08 SUP PHASE 25—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>FOR DATA STATEMENT PHASE</td>
<td>K09 SUP PHASE 26—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>FOR FORMAT STATEMENT PHASE</td>
<td>K10 SUP PHASE 27—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>FOR SUBTRACT DECOMPOSITION PHASE</td>
<td>K11 SUP PHASE 28—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>FOR ASCAN I PHASE</td>
<td>K12 SUP PHASE 29—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>FOR ASCAN II PHASE</td>
<td>K13 SUP PHASE 30—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2B</td>
<td>FOR DO, CONTINUE, ETC. PHASE</td>
<td>K14 SUP PHASE 31—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>FOR SUBSCRIPT OPTIMIZE PHASE</td>
<td>K15 SUP PHASE 32—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>FOR SCAN PHASE</td>
<td>K16 SUP PHASE 33—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>FOR EXPANDER I PHASE</td>
<td>K17 SUP PHASE 34—SUPERVISOR CONTROL RECORDS PROCESSOR NO1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td></td>
<td>8-5 Monitor System Initial Load and System Reload</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase number</td>
<td>Program or program phase name</td>
<td>ID starting in column 73</td>
<td>Phase number</td>
<td>Program or program phase name</td>
<td>ID starting in column 73</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>8C</td>
<td>SYS 1403</td>
<td>PMN</td>
<td>CF</td>
<td>ASM INITIALIZATION PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>8D</td>
<td>SYS 1132</td>
<td>PMN</td>
<td>D0</td>
<td>ASM CARD CONVERSION PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>8E</td>
<td>SYS CONSOLE PRINTER</td>
<td>PMN</td>
<td>D1</td>
<td>ASM DSF OUTPUT PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>8F</td>
<td>SYS 2501</td>
<td>PMN</td>
<td>D2</td>
<td>ASM INTERMEDIATE INPUT PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>90</td>
<td>SYS 1442</td>
<td>PMN</td>
<td>D3</td>
<td>ASM END STATEMENT PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>91</td>
<td>SYS 1134</td>
<td>PMN</td>
<td>D4</td>
<td>ASM ASSEMBLY ERROR PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>92</td>
<td>SYS KEYBOARD</td>
<td>PMN</td>
<td>D5</td>
<td>ASM CONTROL CARDS 1</td>
<td>PTM</td>
</tr>
<tr>
<td>93</td>
<td>SYS 2501/1442 CONVERSION</td>
<td>PMN</td>
<td>D6</td>
<td>ASM CONTROL CARDS 2</td>
<td>PTM</td>
</tr>
<tr>
<td>94</td>
<td>SYS 1134 CONVERSION</td>
<td>PMN</td>
<td>D7</td>
<td>ASM DUMMY PHASE (SYST SYMBOL TBL)</td>
<td>PTM</td>
</tr>
<tr>
<td>96</td>
<td>DISKZ</td>
<td>PMN</td>
<td>D8</td>
<td>ASM SYMBOL TABLE OPTIONS PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>97</td>
<td>DISK1</td>
<td>PMN</td>
<td>D9</td>
<td>ASM EXIT PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>98</td>
<td>DISKN</td>
<td>PMN</td>
<td>DA</td>
<td>ASM PROG HEADER MNEMONICS PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>A0</td>
<td>CORE IMAGE LOADER, PHASE 1</td>
<td>PMN</td>
<td>DB</td>
<td>ASM FILE STATEMENT PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>A1</td>
<td>CORE IMAGE LOADER, PHASE 2</td>
<td>PMN</td>
<td>DC</td>
<td>ASM COMMON SUBROUTINES, ASCOM</td>
<td>PTM</td>
</tr>
<tr>
<td>B0</td>
<td>RESIDENT</td>
<td>PR1</td>
<td>DE</td>
<td>ASM IMPERATIVE STATEMENTS PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B1</td>
<td>ENTER FILES</td>
<td>PR2</td>
<td>DF</td>
<td>ASM DECML EFLC PROCESSING PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B2</td>
<td>ENTER INPUT</td>
<td>PR3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>ENTER CALCULATION</td>
<td>PR4</td>
<td>E0</td>
<td>ASM DECIMAL CONVERSION PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B4</td>
<td>ENTER OUTPUT</td>
<td>PR5</td>
<td>E1</td>
<td>ASM PROG LINKING PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B5</td>
<td>ASSIGN INDICATORS</td>
<td>PR6</td>
<td>E2</td>
<td>ASM DMES PROCESSING PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B6</td>
<td>ASSIGN FIELD NAMES</td>
<td>PR7</td>
<td>E3</td>
<td>ASM PUNCH CONVERSION PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>B7</td>
<td>ASSIGN LITERALS</td>
<td>PR8</td>
<td>E4</td>
<td>ASM INTERMEDIATE DISK OUTPUT</td>
<td>PTM</td>
</tr>
<tr>
<td>B8</td>
<td>EXTENDED FILE AND INPUT DIAGNOSTIC</td>
<td>PR9</td>
<td>E5</td>
<td>ASM SYMBOL TABLE OVERFLOW</td>
<td>PTM</td>
</tr>
<tr>
<td>B9</td>
<td>EXTENDED CALCULATION AND OUTPUT DIAGNOSTIC</td>
<td>PR9</td>
<td>E6</td>
<td>ASM G2250 PH1</td>
<td>PTM</td>
</tr>
<tr>
<td>BA</td>
<td>DIAGNOSTIC MESSAGE 1</td>
<td>PRB</td>
<td>E7</td>
<td>ASM DIVISION OPERATOR PHASE</td>
<td>PTM</td>
</tr>
<tr>
<td>BB</td>
<td>DIAGNOSTIC MESSAGE 2</td>
<td>PRC</td>
<td>E8</td>
<td>ASM CONTROL CARDS 3</td>
<td>PTM</td>
</tr>
<tr>
<td>BC</td>
<td>DIAGNOSTIC MESSAGE 3</td>
<td>PRD</td>
<td>E9</td>
<td>ASM MACRO PHASE 1—SPECIAL OP AND PREPROCESSING</td>
<td>PTM</td>
</tr>
<tr>
<td>BD</td>
<td>ASSEMBLE 1 I/O</td>
<td>PRE</td>
<td>EA</td>
<td>ASM MACRO PHASE 1A—SPECIAL PSEUDO OPS</td>
<td>PTM</td>
</tr>
<tr>
<td>BE</td>
<td>ASSEMBLE 2 I/O</td>
<td>PRF</td>
<td>EB</td>
<td>ASM MACRO PHASE 1B—CONDITIONAL ASSEMBLY</td>
<td>PTM</td>
</tr>
<tr>
<td>BF</td>
<td>ASSEMBLE 3 I/O</td>
<td>PRG</td>
<td>EC</td>
<td>ASM MACRO PHASE 2—MACRO DEFINITION</td>
<td>PTM</td>
</tr>
<tr>
<td>C0</td>
<td>ASSEMBLE 4 I/O</td>
<td>PRH</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>ASSEMBLE TABLES</td>
<td>PRJ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>ASSEMBLE CHAIN AND RAF</td>
<td>PRK</td>
<td>ED</td>
<td>ASM MACRO PHASE 2A—MACRO DEFINITION</td>
<td>PTM</td>
</tr>
<tr>
<td>C3</td>
<td>ASSEMBLE INPUT FIELDS</td>
<td>PRL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>ASSEMBLE CONTROL LEVELS</td>
<td>PRM</td>
<td>EE</td>
<td>ASM MACRO PHASE 2B—MACRO DEFINITION</td>
<td>PTM</td>
</tr>
<tr>
<td>C5</td>
<td>ASSEMBLE MULTI FILE LOGIC</td>
<td>PRN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>ASSEMBLE GET ROUTINES</td>
<td>PRO</td>
<td>EF</td>
<td>ASM MACRO PHASE 3—EXPANSION</td>
<td>PTM</td>
</tr>
<tr>
<td>C7</td>
<td>ASSEMBLE CALCULATIONS 1</td>
<td>PRP</td>
<td>F0</td>
<td>ASM MACRO PHASE 3A—EXPANSION</td>
<td>PTM</td>
</tr>
<tr>
<td>C8</td>
<td>ASSEMBLE CALCULATIONS 2</td>
<td>PRQ</td>
<td>F1</td>
<td>ASM MACRO PHASE 3B—EXPANSION</td>
<td>PTM</td>
</tr>
<tr>
<td>C9</td>
<td>ASSEMBLE OUTPUT FIELDS</td>
<td>PRR</td>
<td>F2</td>
<td>ASM CROSS REFERENCE—PART 1</td>
<td>PTM</td>
</tr>
<tr>
<td>CA</td>
<td>ASSEMBLE PUT ROUTINES</td>
<td>PRS</td>
<td>F3</td>
<td>ASM CROSS REFERENCE—PART 2A</td>
<td>PTM</td>
</tr>
<tr>
<td>CB</td>
<td>ASSEMBLE FIXED DRIVER</td>
<td>PRT</td>
<td>F4</td>
<td>ASM CROSS REFERENCE—PART 2B</td>
<td>PTM</td>
</tr>
<tr>
<td>CC</td>
<td>TERMINATE-compilation</td>
<td>PRU</td>
<td>F5</td>
<td>ASM CROSS REFERENCE—PART 2C</td>
<td>PTM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F6</td>
<td>ASM CROSS REFERENCE—PART 3</td>
<td>PTM</td>
</tr>
</tbody>
</table>

DUP part 2

| CD | DUP CTRL—PART 2 | PS0 |
|    | MACRO UPDATE PROGRAM | PS1 |
**Type 81 Control Record**

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.

**General Function**

The general function of the type 81 control record is to define the end of the loading and/or phases of the system.

**Format of Type 81 Control Record**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>Blanks</td>
</tr>
<tr>
<td>3</td>
<td>A 6 punch</td>
</tr>
<tr>
<td>4</td>
<td>A 1 punch</td>
</tr>
<tr>
<td>5 through 80</td>
<td>Blanks</td>
</tr>
</tbody>
</table>

*Note.* These punches are /8100 in card data format (CDC) 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, 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

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.

**general function**

**format**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>MODE</td>
<td></td>
</tr>
<tr>
<td>5 through 7</td>
<td>Blanks</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>I or R</td>
<td>I indicates initial load. R indicates reload.</td>
</tr>
<tr>
<td>9 through 11</td>
<td>Blanks</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>A or blank</td>
<td>A indicates the assembler is not being loaded or is not on the system being reloaded. Blank indicates the assembler is being loaded or is on the system being reloaded.</td>
</tr>
<tr>
<td>13</td>
<td>F or blank</td>
<td>F indicates the FORTRAN compiler is not being loaded or FORTRAN is not on the system being reloaded. Blank indicates the FORTRAN compiler is being loaded or FORTRAN is on the system being reloaded.</td>
</tr>
<tr>
<td>14</td>
<td>R or blank</td>
<td>R indicates RPG is not being loaded or RPG is not on the system being reloaded. Blank indicates the RPG compiler is being loaded or RPG is on the system being reloaded.</td>
</tr>
<tr>
<td>15</td>
<td>C or blank</td>
<td>C indicates the COBOL compiler (a program product) is being loaded or COBOL is on the system being reloaded. Blank indicates COBOL is not being loaded or COBOL is not on the system being reloaded.</td>
</tr>
<tr>
<td>16 through 80</td>
<td>Blanks</td>
<td></td>
</tr>
</tbody>
</table>

**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

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.

### General Function

Device cards:

<table>
<thead>
<tr>
<th>Device Description</th>
<th>Card Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card Reader/Punch</td>
<td>2501, 1442, Paper Tape</td>
</tr>
<tr>
<td>Paper Tape Reader</td>
<td>1132, 1403</td>
</tr>
<tr>
<td>Printer</td>
<td>1403</td>
</tr>
</tbody>
</table>

### Format

<table>
<thead>
<tr>
<th>Device Description</th>
<th>Card columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1442 Card Reader/Punch</td>
<td>REQ, 1442-5, 1442-7</td>
</tr>
<tr>
<td>Paper Tape Reader and/or Punch</td>
<td>REQ, 1134</td>
</tr>
<tr>
<td>2501 Card Reader</td>
<td>REQ, 2501</td>
</tr>
<tr>
<td>1132 Printer</td>
<td>REQ, 1132</td>
</tr>
<tr>
<td>1403 Printer</td>
<td>REQ, 1403</td>
</tr>
</tbody>
</table>

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.

### 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.

<table>
<thead>
<tr>
<th>Device specified on REQ control records</th>
<th>Principal I/O device</th>
</tr>
</thead>
<tbody>
<tr>
<td>2501, 1442, paper tape</td>
<td>2501 input, 1442 output</td>
</tr>
<tr>
<td>1442, paper tape</td>
<td>1442 input/output</td>
</tr>
<tr>
<td>Paper tape</td>
<td>Paper tape input/output</td>
</tr>
<tr>
<td>1403, 1132</td>
<td>1403 output</td>
</tr>
</tbody>
</table>
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**

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**

<table>
<thead>
<tr>
<th>Card column</th>
<th>Contents</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 through 4</td>
<td>CORE</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>6 through 8</td>
<td>04K, 08K, 16K, or 32K</td>
<td>The entry chosen specifies the core size you are defining.</td>
</tr>
<tr>
<td>9 through 80</td>
<td>Blanks</td>
<td></td>
</tr>
</tbody>
</table>

**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 star-wheels 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.
7. Press PROGRAM START to finish the reading of PTUTL.
8. Press PROGRAM START again.
9. Turn console entry switches 2 and 3 on.

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.

The core image loader is read into core storage, and the system waits with /006C displayed in the ACCUMULATOR.

When the reading of BP17 is complete, the system waits with /00C9 in the ACCUMULATOR.

The system waits again with /1111 in the ACCUMULATOR.

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.

Figure 8-1 (Part 1 of 4). Preparation of paper tape load and reload control tapes
Prepare the paper tape load mode control record

1. Enter MODE through the keyboard.
2. Press the SPACE BAR three times.
3. Enter an I or an R.
4. Press the SPACE BAR three times.
5. Enter an A or press the SPACE BAR once.
6. Enter an F or press the SPACE BAR once.
7. Enter an R so that references to the RPG compiler are ignored during loading.

\[ / \text{indicates initial load.} \]
\[ R \text{ indicates reload.} \]
\[ A \text{ indicates that the assembler is not being loaded.} \]
\[ F \text{ indicates that the FORTRAN compiler is not being loaded.} \]

Were mistakes made during entry?

Yes

Prepare another load mode control record
Punch another leader of delete codes.

No

Finish the load mode control record
1. Press EOF.
2. Punch a trailer of delete codes with the DELETE key on the paper tape punch.

Figure 8-1 (Part 2 of 4). Preparation of paper tape load and reload control tapes
Prepare the system configuration tape
1. Punch a leader of delete codes with the DELETE key on the paper tape punch.
2. Enter SCON through the keyboard.
3. Press EOF to end the SCON record.

Define the system I/O devices
1. Enter REQ.
2. Press the SPACE BAR six times (5 times if a 2-digit ISS number is being entered in Step 6).
3. Enter the ISS number for an I/O device being configured.
4. Press EOF.

If errors are made during the preparation of this tape, repeat from here.

I/O device	ISS number
1442 Card Read	1
Punch/Card Reader	3
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.

Are you defining the size of core storage?

Yes

Define core storage size
1. Enter CORE.
2. Press the SPACE BAR once.
3. Enter the size of core that you are defining.
4. Press EOF.

Finish the system configuration tape
1. Enter TERM.
2. Press EOF.
3. Punch a trailer of delete codes with the DELETE key on the paper tape punch.

Finish the preparation of the control record tapes
1. Remove the tapes from the paper tape reader and punch.
2. Separate the load mode control record from the system configuration control record by cutting them apart at the delete codes that separate them on the tape just punched.

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 initialized 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.
Turn on system power
Move the console POWER switch to ON.

Is DM2 on cards or cartridge?

Cartridge

Perform the "Card System Preload Procedure" in this chapter.

Cards

Ready a cartridge for initialization
1. Place a disk cartridge or pack in a disk drive.
2. Ready the disk drive (see Chapter 7 for information about readying disk drives).

Is cartridge freshly initialized?

Yes

Ready the 1132 Printer
1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

No

Initialize the cartridge using the DCIP program (see Chapter 9).

Note: If your system has 2311 Disk Storage Drives, be sure all disks in a disk pack are initialized.

Is the principal printer 1132 or 1403?

1132

Ready the 1132 Printer
1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

1403

A2

B2

Figure 8-3 (Part 1 of 3). Card system initial load procedure
1. Press NPRO.
2. Place the first of the IBM system cards in the hopper, face down, 9-edge first.
3. Press START.

Ready the 1403 Printer

1. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.
2. Press CARRIAGE RESTORE on the printer.
3. Press START on the printer.

Ready the IBM system deck

Place the user-punched control cards in the IBM system deck where indicated in Figure 8-2.

Is input from 1442, Model 6 or 7, or 2501?

Ready the 2501 Card Reader
1. Press NPRO.
2. Place the first of the IBM system cards 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).

Ready the 1442, Model 6 or 7
1. Press NPRO.
2. Place the first of the IBM system cards in the hopper, face down, 9-edge first.
3. Press START.

Figure 8-3 (Part 2 of 3). Card system initial load procedure
Card System Initial Load operating procedure

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).

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.

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.

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

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.

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.

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.
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.
Move the console POWER switch to ON.

Is reload information on cards or cartridge?

Cards

Ready the system cartridge to be used
1. Place a system cartridge in a disk drive.
2. Ready the disk drive (see Chapter 7 for information about readying disk drives).

Cartridge

Perform the "Card System Preload Procedure" in this chapter.

Ready the 1132 Printer
1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

1132

1403

Ready the 1403 Printer
1. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.
2. Press CARRIAGE RESTORE on the printer.
3. Press START on the printer.

A2

Figure 8-5 (Part 1 of 4). Card system reload procedure
Ready the IBM system deck

Place the load mode and REQ (and CORE, if used) cards that you punched in the system deck where indicated in Figure 8-4.

Figure 8-5 (Part 2 of 4). Card system reload procedure
Figure 8-5 (Part 3 of 4). Card system reload procedure
Finish procedure
Continue placing IBM system cards in the reader hopper until all of the cards have been placed in the hopper.

The system prints END OF RELOAD on the console printer when the reload is complete.

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational again.

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.
Card System Preload operating procedure

Ready the preload cartridge
1. Place the preload cartridge in a disk drive.
2. Ready the disk drive (see Chapter 7 for information about readying disk drives).
3. Set the console entry switches 12 through 15 to indicate the physical drive number of the drive that contains the preload 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 6 — switches 13 and 14 on
*If your preload cartridge is on a 1316 Disk Pack, the DM2 system is on either physical drive 1 or 6.

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 6 — switches 13 and 14 on
*If your preload cartridge is on a 1316 Disk Pack, the DM2 system is on either physical drive 1 or 6.

Start cold start
1. Turn the console mode switch to RUN.
2. Press IMM STOP on the console.
3. Press RESET on the console.
4. Press PROGRAM LOAD on the console.

Figure 8-6 (Part 1 of 2). Card system preload procedure
System waits before punching begins?

Yes

No

Punching of the monitor system begins.

Finish the procedure

1. Continue adding cards to the card punch hopper until the system halts with hexadecimal /03CC displayed in the ACCUMULATOR on the console display panel.

2. Remove the 9s cards (a 9 punched in columns 1 through 80) that separate the object decks punched by this procedure. The stand-alone utility programs, sample programs, and the DISKN subroutine for the 2311 Disk Storage Drives are the last decks punched during an initial load; remove these.

If the system halts (halt codes displayed in the ACCUMULATOR on the console display panel) refer to Appendix B.

If a 1442, Model 5, punch is being used, the first card through the punch is blank. Throw this card away.

Initial load or reload?

Reload

Initial load

Return to "Card System Initial Load Procedure" in this chapter.

Return to "Card System Reload Procedure" in this chapter.

Figure 8-6 (Part 2 of 2). Card system preload procedure
The materials that you need to perform a paper tape system initial load procedure are:

- An initialized 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 as shown in Figure 8-8.
Figure 8-7. Paper tape system load tapes
Turn on system power
Move the POWER switch on the console to ON.

Ready a cartridge for initialization
1. Place a cartridge in the single disk drive (the cartridge can be placed on any drive on the system).
2. Move the DISK switch on the disk drive to ON. The drive requires approximately 90 seconds to reach operating speed (see Chapter 7 for readying the 2310 Disk Storage Drive).

Is cartridge freshly initialized?
Yes

Is the principal printer 1132 or 1403?
1132

1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

1403

Is the principal printer 1132 or 1403?

Ready the 1403 Printer
1. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.
2. Press CARRIAGE RESTORE on the printer.
3. Press START on the printer.

Figure 8-8 (Part 1 of 3). Paper tape system initial load procedure
Figure 8-8 (Part 2 of 3). Paper tape system initial load procedure
Continue the reading of the Monitor System

10. Insert BP02, system loader, Part 2, in the paper tape reader.

11. Press PROGRAM START.

12. Insert the user-punched system configuration tape in the reader.

13. Press PROGRAM START.

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.

Load tapes BP03 through BP14

1. Insert the next higher numbered tape in the paper tape reader.

2. Press PROGRAM START.

The system waits with /3000 displayed in the ACCUMULATOR when reading of each tape is complete.

Was last tape BP14?

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational.

Figure 8-8 (Part 3 of 3). Paper tape system initial load 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.
Turn on system power
Move the POWER switch on the console to ON.

Ready the system cartridge to be used
1. Place a system cartridge in the single disk drive (the cartridge can be placed on any drive on the system).
2. Move the DISK switch on the disk drive to ON. The drive requires approximately 90 seconds to reach operating speed (see Chapter 7 for readying the 2310 Disk Storage Drive).

Is the principal printer 1132 or 1403?

1132

Ready the 1132 Printer
1. Move the printer MOTOR switch to ON.
2. Press CARRIAGE RESTORE.
3. Press START.

1403

Ready the 1403 Printer
1. Be sure the ENABLE/DISABLE switch on the 1133 Multiplex Control Enclosure is in the ENABLE position.
2. Press CARRIAGE RESTORE on the printer.
3. Press START on the printer.

Figure 8-10 (Part 1 of 4). Paper tape system reload procedure
Perform a cold start

1. Insert tape BP15, cold start paper tape record, in the paper tape reader.

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).

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.

A cold start is recommended prior to a reload operation in order to restore certain parameters in DCOM on the system cartridge.

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

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.

Figure 8-10 (Part 2 of 4). Paper tape system reload procedure
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.
3. Press PROGRAM START again to finish the reading of Part 1 of the system loader.
4. Press PROGRAM START again.
5. Place the user-punched load mode control record tape in the reader.
6. Press PROGRAM START.
7. Insert tape BP02, system loader, Part 2, in the paper tape reader.
8. Press PROGRAM START.

Configure system

1. Insert the user-punched system configuration tape in the reader.
2. Press PROGRAM START.

The core image loader is read into core storage from BP01, and the system waits with /006C displayed in the ACCUMULATOR.

When reading of BP01 is complete, the system waits with /00C9 displayed in the ACCUMULATOR.

The system waits again with /3000 displayed in the ACCUMULATOR.

The system waits with /3000 in the ACCUMULATOR when reading of the tape is complete.

The system waits with /3000 in the ACCUMULATOR when reading of BP02 is complete.

The system waits with /3000 in the ACCUMULATOR when reading of the system configuration tape is complete.

Figure 8-10 (Part 3 of 4). Paper tape system reload procedure
Load tapes BP03, revised programs or phases, and BP08

1. Insert the next tape.
2. Press PROGRAM START.

The system waits with /3000 in the ACCUMULATOR when reading of each tape is complete.

Was last tape BP08?

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational.

Figure 8-10 (Part 4 of 4). Paper tape system reload procedure
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.

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
- Console Printer Core Dump Program paper tape, BP20

Figure 9-1 is the operating procedure for the stand-alone Console Printer Core Dump Program.
Ready the 2501 card reader
1. Press NPRO.
2. Place the console printer core dump program card 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).

Ready the paper tape reader
1. Insert tape BP20, console printer core dump, in the paper tape reader.
2. Position under the read star-wheels one of the delete codes beyond the program ID.

Ready the console
1. Press IMM STOP.
2. Press RESET.
3. Turn the console mode switch to RUN.
4. Set the margins on the console printer.
5. Set the address (in binary) of the starting core location in the console entry switches.

Set the number of print positions to a multiple of 5 so each line will be printed in the same format.

Ready the 1442, Model 6 or 7.
1. Press NPRO.
2. Place the console printer core dump program card in the hopper, face down, 9-edge first.
3. Press START.

Figure 9-1 (Part 1 of 2). Console printer core dump operating procedure
Start the dump program
Press PROGRAM LOAD on the console keyboard.

Do you want to interrupt the dump program?

Yes
Temporarily interrupt the dump
1. Press IMM STOP on the console keyboard.
2. Press PROGRAM START on the console keyboard to continue

No
The dump continues until complete.

Perform a "Cold Start Procedure" to continue monitor system operations (see Chapter 7).

Figure 9-1 (Part 2 of 2). Console printer core dump operating procedure
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.

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
- Printer Core Dump Program paper tape, BP19

Figure 9-2 is the operating procedure for the stand-alone Printer Core Dump Program.
Ready the paper tape reader
1. Insert tape BP19, printer core dump program, in the paper tape reader.
2. Position under the read star-wheels one of the delete codes beyond the program ID.

Ready the console
1. Press IMM STOP.
2. Press RESET.
3. Turn the console mode switch to RUN.

Ready the printer
1. Press CARRIAGE RESTORE.
2. Press START.

Is the program on cards or paper tape?

Card

Tape

A2

B2

Figure 9-2 (Part 1 of 3). Printer Core Dump Program operating procedure
1. Press NPRO.

2. Place the printer core dump program card 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).

Start the dump program
Press PROGRAM LOAD on the console keyboard.

Ready the 1442, Model 6 or 7
1. Press NPRO.
2. Place the printer core dump program card deck in the hopper, face down, 9-edge first.
3. Press START.

Ready the 2501 card reader
1. Press NPRO.
2. Place the printer core dump program card 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).

Figure 9-2 (Part 2 of 3). Printer Core Dump Program operating procedure
Do you want to interrupt the dump program?

1. Press IMM STOP on the console keyboard.
2. Press PROGRAM START on the console to continue.

Perform a "Cold Start Procedure" to continue monitor system operations (see Chapter 7).

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.

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).

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.

Figure 9-3 (Part 1 of 4). Load DCIP operating procedure
Ready the paper tape reader

1. Insert tape BP16, Disk Cartridge Initialization Program, in the paper tape reader.
2. Position under the read starwheel one of the delete codes beyond the program ID.

Ready the 2501 Card Reader

1. Press NPRO.
2. Place the Disk Cartridge Initialization Program card 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).

Ready the 1442, Model 6 or 7

1. Press NPRO.
2. Place the Disk Cartridge Initialization Program card deck in the hopper, face down, 9-edge first.
3. Press START.
Does the system halt with /006C in the ACCUMULATOR?

Yes

Continue the reading of tape BP16

1. Press PROGRAM START on the console.
2. Press PROGRAM START again when the system waits with /00C9 in the ACCUMULATOR.

No

Start the loading of DCIP

Press PROGRAM LOAD on the console.

DCIP is read into core storage and the system prints:

TURN ON:
SW0 TO INITLZ
SW1 TO COPY
SW2 TO DUMP
SW3 TO PATCH
SW4 TO ANALYZE
SW5 TO CMP
SW6 TO INITLZ NEW DISK

Figure 9-3 (Part 3 of 4). Load DCIP operating procedure
Select the DCIP function to be performed

1. Turn off all console entry switches.
2. Turn on the console entry switch that corresponds to the DCIP function you are doing.
3. Press PROGRAM START.

If switch 0 or 6 is on, Figure 9-4
If switch 1 is on, Figure 9-5
If switch 2 is on, Figure 9-6
If switch 3 is on, Figure 9-7
If switch 4 is on, Figure 9-8
If switch 5 is on, Figure 9-9

Figure 9-3 (Part 4 of 4). Load DCIP operating procedure
Stand-alone Utilities
DCIP initialize procedure

ENTER DR NO. (BITS 12-15) is printed on the console printer.

Initialize a cartridge
1. Turn off console entry switch 0.
2. Enter through console entry switches 12 through 15 the physical drive number (in binary) of the drive that contains the cartridge being initialized.
3. Press PROGRAM START.
4. Turn off all console entry switches.
5. If the pack has not been previously initialized, be sure that the access arm is at sector 0. On a 2311 cartridge, turn off the disk drive and turn it on again before initializing each disk surface.
6. Enter through the console entry switches the cartridge ID (binary representation).
7. Press PROGRAM START.

ENTER CART ID is printed.

A valid cartridge ID is a hexadecimal number between 0001 and 7FFF.

The cartridge ID you entered is printed, and the system waits with the cartridge ID in hexadecimal displayed in the ACCUMULATOR.

Are switches correct for the physical drive number and cartridge ID?

Yes

Start initialization
Press PROGRAM START again.

The cartridge is initialized. The entire surface is cleared, disk addresses are written, and three distinct patterns are written on each sector and read back for testing purposes.

Interrupt the initialize function to reenter the physical drive number and the cartridge ID.

Press INT REG.

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
SW6 TO INITLZ NEW DISK

Restart the initialize function
1. Turn off all console entry switches.
2. Turn on switch 0.
3. Press PROGRAM START.

A1

Figure 9-4 (Part 1 of 5). Operating procedure for DCIP initialize function
Any of the following messages are printed:

NO DEF CYLS
(if all of the cartridge is usable)

or

DEF CYLS:
XXXX . . .
(if defective cylinders are found)

and/or

DEF CART
(if more than 3 defective cylinders are printed in the previous message. This message is also printed if cylinder 0 is defective, or if sector addresses cannot be written on every sector.)

After any of the previous messages are printed, this message is printed:

TURN ON SW0 FOR MORE TESTING

Do you want to do more testing?

Yes  A3

No  A4
Continue testing the cartridge

1. Turn off all console entry switches.
2. Turn on switch 0.
3. Press PROGRAM START.
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.)
5. Press PROGRAM START.

A maximum of 31 (decimal) can be entered.

Figure 9-4 (Part 3 of 5). Operating procedure for DCIP initialize function
Stand-alone Utilities
DCIP initialize procedure

Continue or finish DCIP operations
1. Turn off all console entry switches.
2. Press PROGRAM START.

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
SW6 TO INITLZ NEW DISK

Note. If your system has 2311 Disk Storage Drives, be sure all disks in a disk pack are initialized.

Do you want to continue DCIP?
Yes
No

Do one of the following:
Perform a "Cold Start Procedure" (see Chapter 7) from a system cartridge.
Perform an initial system load on the cartridge just initialized (see Chapter 8).

Figure 9-4 (Part 4 of 5). Operating procedure for DCIP initialize function
Select the DCIP function to be performed

1. Change cartridges or packs and ready the disk drives, if necessary.
2. Turn off all console entry switches.
3. Turn on the console entry switch that corresponds to the DCIP function you are doing.
4. Press PROGRAM START.

If switch 0 or 6 is on  
Figure 9-4

If switch 1 is on  
Figure 9-5

If switch 2 is on  
Figure 9-6

If switch 3 is on  
Figure 9-7

If switch 4 is on  
Figure 9-8

If switch 5 is on  
Figure 9-9

Figure 9-4 (Part 5 of 5). Operating procedure for DCIP initialize function
Copy a cartridge

1. Turn off console entry switch 1.
2. Enter through console entry switches 0 through 3 the physical drive number (in binary) of the drive that contains the cartridge being copied (source drive).
3. Enter through console switches 12 through 15 the physical drive number (in binary) of the drive that contains the cartridge onto which information is being copied (object drive).
4. Press PROGRAM START.

The source and object cartridge ID’s are printed as follows:

SRC ID = XXXX
OBJ ID = YYYY

A system wait follows.

Are switches correct for the physical drive numbers for the source and object drives?

No

Interrupt the copy function to reenter the physical drive numbers for the source and object drives.
Press INT REQ.

Yes

Start the copy function
Press PROGRAM START again.

Figure 9-5 (Part 1 of 8). Operating procedure for DCIP copy function
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
SW6 TO INITLZ NEW DISK

Restart the copy function

1. Turn off all console entry switches.
2. Turn on switch 1.
3. Press PROGRAM START.

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
SW6 TO INITLZ NEW DISK

Is X CART NOT INITLZED printed?

Yes

No

SOURCE or OBJECT is printed in place of X.

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
SW6 TO INITLZ NEW DISK

Figure 9-5 (Part 2 of 8). Operating procedure for DCIP copy function
The object cartridge is a DM2 UCART master cartridge.

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
SW6 TO INITLZ NEW DISK

Do you want to continue with copy or initialize the cartridge?

Init
Continue

Figure 9-5 (Part 3 of 8). Operating procedure for DCIP copy function
Interrupt the copy function to begin initialization
Press INT REQ.

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
SW6 TO INITLZ NEW DISK

Initialize the cartridge
1. Turn off all console entry switches.
2. Turn on switch 0.
3. Press PROGRAM START.

Figure 9-4 (Part 4 of 8). Operating procedure for DCIP copy function
The contents (less defective cylinder data and the cartridge ID) of the source cartridge are copied onto the object cartridge.

Word 5 of sector @IDAD of the source cartridge is incremented by one when written on the object cartridge. Because of this, the copy number of the object cartridge is always one greater than the copy number of the source cartridge.

A disk read, write, or seek error has occurred.

/5003 is displayed in the ACCUMULATOR

The ACCUMULATOR EXTENSION contains /XYYY, where X is the drive code, and YYY is the address of the sector in error.

Figure 9-5 (Part 5 of 8). Operating procedure for DCIP copy function
Do you want the sector in error rewritten or reread?

Yes

Have sector in error rewritten or reread

1. Turn off all console entry switches.
2. Turn on console entry switch 0.
3. Press PROGRAM START.

The sector in error is rewritten or reread and the copy function continued.

No

Continue copy function

1. Turn off all console entry switches.
2. Press PROGRAM START.

The read or write error is ignored, and the contents of the object cartridge reflect the last attempt to copy the sector in error.

B4

B4

Figure 9-5 (Part 6 of 8). Operating procedure for DCIP copy function
The copy function is completed and 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
SW6 TO INITLZ NEW DISK

Figure 9-5 (Part 7 of 8). Operating procedure for DCIP copy function
Select the DCIP function to be performed

1. Change cartridges or packs and ready the disk drives, if necessary.
2. Turn off all console entry switches.
3. Turn on the console entry switch that corresponds to the DCIP function you are doing.
4. Press PROGRAM START.

If switch 0 or 6 is on

If switch 1 is on

If switch 2 is on

If switch 3 is on

If switch 4 is on

If switch 5 is on

Figure 9-4
Figure 9-5
Figure 9-6
Figure 9-7
Figure 9-8
Figure 9-9

Figure 9-5 (Part 8 of 8). Operating procedure for DCIP copy function
Stand-alone Utilities
DCIP dump procedure

**DCIP dump procedure**

**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. Turn off console entry switch 2.
2. 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.
3. Enter through the console entry switches 4 through 15 the physical address of the first sector being dumped.
4. Press PROGRAM START.
5. Enter through the console entry switches the number of consecutive sectors to be dumped.
6. Press PROGRAM START.

**Figure 9-6 (Part 1 of 4). Operating procedure for DCIP dump function**
Stand-alone Utilities

DCIP dump procedure

Is DISK ERR ... TURN ON SW 0 TO RETRY printed?

A disk read error has occurred.

Do you want the sector in error reread?

Yes

Have sector in error reread
1. Turn off all console entry switches.
2. Turn on switch 0.
3. Press PROGRAM START.

The sector in error is reread.

A2

Continue dump function
1. Turn off all console entry switches.
2. Press PROGRAM START.

A3

B3

Figure 9-6 (Part 2 of 4). Operating procedure for DCIP dump function
The read error is ignored. The sector in error is printed as it was last read, and the dump continues.

The dump continues until all the specified sectors are dumped.

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
SW6 TO INITLZ NEW DISK

Do you want to continue DCIP?

Yes

No

Perform a "Cold Start Procedure" (see Chapter 7) to make the system operational.

Figure 9-6 (Part 3 of 4). Operating procedure for DCIP dump function
Select the DCIP function to be performed

1. Change cartridges or packs and ready the disk drives, if necessary.
2. Turn off all console entry switches.
3. Turn on the console entry switch that corresponds to the DCIP function you are doing.
4. Press PROGRAM START.

If switch 0 or 6 is on
- Figure 9-4

If switch 1 is on
- Figure 9-5

If switch 2 is on
- Figure 9-6

If switch 3 is on
- Figure 9-7

If switch 4 is on
- Figure 9-8

If switch 5 is on
- Figure 9-9
Start the patch function

1. Turn off console entry switch 3.
2. Enter through console entry switches 0 through 3 the physical drive number (in binary) of the drive that contains the cartridge being patched.
3. Enter through console entry switches 4 through 15 the address of the sector being patched.
4. Press PROGRAM START.
5. Enter through the console entry switches the relative address of the sector word being changed.
6. Press PROGRAM START.

The sector address is a right-adjusted hexadecimal number, maximum /0657.

The specified sector is dumped, and the following message is printed:
ENTER RLTV ADDR OF SCTR WD TO CHANGE.

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).

The KEYBOARD SELECT indicator on the console keyboard is turned on.

Figure 9-7 (Part 1 of 4). Operating procedure for DCIP patch function
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.
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.

1. Enter through the keyboard the 4 hexadecimal characters that comprise the new word to be stored.

2. Use any of the special control characters previously listed to perform the desired patch functions.

Is the patch function terminated by the use of the o or * keys?

No

Yes

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
SW6 TO INITLZ NEW DISK

Figure 9-7 (Part 3 of 4). Operating procedure for DCIP patch function
Select the DCIP function to be performed

1. Change cartridges or packs and ready the disk drives, if necessary.
2. Turn off all console entry switches.
3. Turn on the console entry switch that corresponds to the DCIP function you are doing.
4. Press PROGRAM START.

If switch 0 or 6 is on

If switch 1 is on

If switch 2 is on

If switch 3 is on

If switch 4 is on

If switch 5 is on

Figure 9-4

Figure 9-5

Figure 9-6

Figure 9-7

Figure 9-8

Figure 9-9

Figure 9-7 (Part 4 of 4). Operating procedure for DCIP patch function
A disk read error has occurred.

Do you want to dump the sector in error?

Yes

No
DCIP analysis procedure

A2

Dump the sector in error
1. Turn all console entry switches off.
2. Turn on console entry switch 0.
3. Press PROGRAM START.

The sector containing the error is dumped, and sector analysis continues.

B2

Continue disk analysis
1. Turn off all console entry switches.
2. Press PROGRAM START.

Figure 9-8 (Part 2 of 4). Operating procedure for DCIP analysis function
An incorrect sector address has been read.

The system writes the correct sector address on the disk, and sector analysis continues.

Disk analysis continues until all sectors have been analyzed.

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
SW6 TO INITLZ NEW DISK

Do you want to continue DCIP?

No

Perform a "Cold Start Procedure" (see Chapter 7) to make the system operational.

Yes

Figure 9-8 (Part 3 of 4). Operating procedure for DCIP analysis function
Select the DCIP function to be performed

1. Change cartridges or packs and ready the disk drives, if necessary.
2. Turn off all console entry switches.
3. Turn on the console entry switch that corresponds to the DCIP function you are doing.
4. Press PROGRAM START.

Figure 9-8 (Part 4 of 4). Operating procedure for DCIP analysis function
Stand-alone Utilities
DCIP compare procedure

ENTER:
SOURCE DR (BITS 0-3)
OBJECT DR (BITS 12-15)
is printed on the console printer.

Start the compare function

1. Turn off console entry switch 5.
2. 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).
3. 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).
4. 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 continues printed.

TURN ON:
SW0 TO INITLZ
SW1 TO COPY
SW2 TO DUMP
SW3 TO PATCH
SW4 TO ANALYZE
SW5 TO CMP
SW6 TO INITLZ NEW DISK

Figure 9-9 (Part 1 of 2). Operating procedure for DCIP compare function
Do you want to compare more cartridges?

- Yes: Change cartridges or packs and ready the disk drives, if necessary.
- No: Turn off all console entry switches.
- Yes: Turn on console entry switch 5.
- No: Press PROGRAM START.

Perform a "Cold Start Procedure" (see Chapter 7) to make the system operational.

Reload the DCIP Program (Figure 9-3).

If one of the DCIP functions other than CMP is selected, this message is printed:

CMP OPTION USED ... RELOAD DCIP

Figure 9-9 (Part 2 of 2). Operating procedure for DCIP compare function
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.
7. Remove BP18 from the paper tape reader.

The program is read into core storage, and the system waits with /1111 displayed in the ACCUMULATOR.

**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.

Figure 9-10 (Part 1 of 4). Paper tape reproducing operating procedure
Do you want to interrupt the operation?

Yes

1. Press PROGRAM STOP on the console.
2. Press PROGRAM START on the console to continue.

No

Stand-alone Utilities

Paper tape reproducing procedure

Figure 9-10 (Part 2 of 4). Paper tape reproducing operating procedure
Ready the paper tape reader
1. Insert the next tape to be reproduce.
2. Position, under the read star-wheels, one of the delete codes.
3. Press PROGRAM START on the console keyboard.

/2222 displayed in the ACCUMULATOR indicates that the paper tape reader is not ready.
/3333 displayed in the ACCUMULATOR indicates that the paper tape punch is not ready.

Ready the paper tape punch
1. Insert a blank tape in the paper tape punch.
2. Punch several inches of delete codes and a header.
3. Press PROGRAM START on the console keyboard.

Are more tapes to be reproduced?
Yes

No

Figure 9-10 (Part 3 of 4). Paper tape reproducing operating procedure
Finish procedure

1. Punch a trailer of delete codes in the output tape.
2. Remove the tapes from the paper tape reader and punch.

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational.

An unlimited number of tapes can be reproduced with this procedure. Be sure to punch a trailer and header in the output tape after each tape is reproduced. This is done so that the output tape can be cut apart after the reproducing procedure is completed.
Stand-alone Utilities

Loading PTUTL

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.
7. Press PROGRAM START to finish the reading of PTUTL.
8. Press PROGRAM START again.

The core image program is read into core storage, and the system waits with /006C displayed in the ACCUMULATOR.

When the reading of BP17 is complete, the system waits with /00C9 in the ACCUMULATOR.

The system waits with /1111 displayed in the ACCUMULATOR.

Figure 9-11. Loading the stand-alone PTUTL tape
Read system library version of PTUTL into core
1. Insert the tape that includes the \( // \) XEQ PTUTL control record.
2. Press PROGRAM START on the console keyboard.

The system halts with /1111 displayed in the ACCUMULATOR on the console display panel.

Is input from paper tape or the keyboard?
1. Insert the source tape.
2. Position under the read star-wheels one of the delete codes.

Ready the paper tape reader
1. Insert a blank tape in the paper tape punch.
2. Punch several inches of delete codes and a header.

Is output to paper tape or the console printer?

Ready the paper tape punch

Figure 9-12 (Part 1 of 4). PTUTL operating procedure
Make changes and/or additions

1. Turn on the appropriate console entry switches to perform the PTUTL functions you want.

2. Press PROGRAM START.

<table>
<thead>
<tr>
<th>Console entry switch on</th>
<th>PTUTL function performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Print record after reading</td>
</tr>
<tr>
<td>1</td>
<td>Read records from the paper tape reader</td>
</tr>
<tr>
<td>2</td>
<td>Accept keyboard input</td>
</tr>
<tr>
<td>3</td>
<td>Punch records on the paper tape punch</td>
</tr>
<tr>
<td>14</td>
<td>Wait after punching with /3333 in the ACCUMULATOR</td>
</tr>
<tr>
<td>15</td>
<td>Wait after printing with /2222 in the ACCUMULATOR</td>
</tr>
</tbody>
</table>

All other console entry switches must be off.

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
Are all changes and additions complete?

Yes

Finish procedure
1. Turn off all console entry switches.
2. Press PROGRAM START.

No

Was PTUTL stand-alone or in the system library?

Stand-alone

Perform the "Cold Start Procedure" in Chapter 7 to make the monitor system operational.

System library

Continue with next monitor job
1. Insert the tape for the next monitor job to be processed.
2. Press PROGRAM START.

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:

<table>
<thead>
<tr>
<th>Your action</th>
<th>System response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Load into core storage and start execution of PTUTL.</td>
<td>The system waits with <code>/1111</code> displayed in the ACCUMULATOR on the console display panel.</td>
</tr>
<tr>
<td>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.</td>
<td>The <code>// JOB</code> control record is read, punched in the output tape, and the system waits with <code>/3333</code> in the ACCUMULATOR.</td>
</tr>
<tr>
<td>3. Turn on console entry switches 1, 3, and 14.</td>
<td>The comments record is read and printed on the console printer. The system waits with <code>/2222</code> in the ACCUMULATOR.</td>
</tr>
<tr>
<td>4. Press PROGRAM START.</td>
<td>The K.B. SELECT indicator on the console keyboard turns on and <code>/3333</code> is displayed in the ACCUMULATOR.</td>
</tr>
<tr>
<td>5. In addition to the console entry switches already turned on, turn on 0, 2, and 15.</td>
<td>The new comments record is punched in the output tape; the system waits with <code>/2222</code> in the ACCUMULATOR.</td>
</tr>
<tr>
<td>6. Press PROGRAM START.</td>
<td>The K.B. SELECT indicator turns on, and <code>/3333</code> is displayed in the ACCUMULATOR.</td>
</tr>
<tr>
<td>7. Press PROGRAM START again.</td>
<td>The <code>// PAUSE</code> control record is punched in the output tape; the system waits with <code>/2222</code> in the ACCUMULATOR.</td>
</tr>
<tr>
<td>8. Enter the new comments record in the proper format.</td>
<td></td>
</tr>
<tr>
<td>9. Press EOF.</td>
<td></td>
</tr>
<tr>
<td>10. Turn off console entry switch 1.</td>
<td></td>
</tr>
<tr>
<td>11. Press PROGRAM START.</td>
<td></td>
</tr>
<tr>
<td>12. Enter the <code>// PAUSE</code> control record.</td>
<td></td>
</tr>
<tr>
<td>13. Press EOF.</td>
<td></td>
</tr>
<tr>
<td>14. Turn off console entry switches 0, 2, and 15.</td>
<td></td>
</tr>
</tbody>
</table>
Your action | System response
---|---
15. Turn on console entry switch 1. (Switches 3 and 14 should still be on.) | The //ASM control record is read and punched in the output tape; the system waits with /3333 in the ACCUMULATOR.
16. 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.
17. Turn off all console entry switches except 1. | The next input record is read into the I/O buffer, overlaying the //DUP control record. (The //DUP control record is deleted.)
18. Turn on console entry switches 0 and 15. | 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.
19. Press PROGRAM START. | A CALL EXIT is executed.
20. Press PROGRAM START again. | 
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. | 
24. Turn off all console entry switches. | 
25. Press PROGRAM START. | 
Chapter 10. Remote Job Entry Program

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.

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.

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:

\[
\begin{align*}
\text{LINE}=P \quad & \text{LINE}=S \quad & \text{LINE}=M (x,y) \quad & \text{UEXIT}=(address 1, address 2) \quad & \text{UEXIT}=\text{USER} \quad & \text{UEXIT}=\text{USER} \\
\text{COMPRESS}=\text{NO} \quad & \text{COMPRESS}=\text{YES}
\end{align*}
\]

- **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
- \( aaaa \) 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.

The following example shows the coding for generating the 1130 Work Station Program:

```
/* JOB */
/* EXEC RJE00 */
LINE=M(A,B), UEXIT=(21B0,22B0), 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 \{, C
                   \}
                   \{, D, \xa\xa\xa\xa [, \xb\xb\xb\xb]\}
```

. 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,
\xa\xa\xa\xa is the disk sector address (hexadecimal), and
\xb\xb\xb\xb 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.

```
<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>Starting address</td>
</tr>
<tr>
<td>+1</td>
<td>Ending address</td>
</tr>
<tr>
<td>+2</td>
<td>Logical record length</td>
</tr>
<tr>
<td>+3</td>
<td>Control character type</td>
</tr>
<tr>
<td>+4</td>
<td>Record format</td>
</tr>
<tr>
<td>+5</td>
<td>End of data</td>
</tr>
</tbody>
</table>
```

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**

This 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

<table>
<thead>
<tr>
<th>Device</th>
<th>I/O Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1132 Printer</td>
<td>PRNT2</td>
</tr>
<tr>
<td>1403 Printer</td>
<td>PRNT3</td>
</tr>
<tr>
<td>1442, Model 6 or 7,</td>
<td>CAR1</td>
</tr>
<tr>
<td>Card Read/Punch</td>
<td>READ1</td>
</tr>
<tr>
<td>2501 Card Reader</td>
<td>PNCH1</td>
</tr>
<tr>
<td>Console Keyboard</td>
<td>TYPE0</td>
</tr>
<tr>
<td>Disk</td>
<td>DISK2</td>
</tr>
</tbody>
</table>

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:

```
/ / JOB
/ / RJE RJE
... RASTART

JECL statements and operating system job

... RJEEND
```
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

<table>
<thead>
<tr>
<th>Console Entry Switch</th>
<th>Console Entry Switch Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (off)</td>
<td>Indicates that RJE messages from the central system are printed on the console printer</td>
</tr>
<tr>
<td>0 (on)</td>
<td>Indicates that RJE messages from the central system are printed on the line printer</td>
</tr>
<tr>
<td>1</td>
<td>If on when the console printer becomes not ready, the operation is retried.</td>
</tr>
<tr>
<td>2</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.
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:

<table>
<thead>
<tr>
<th>Error code prefix</th>
<th>Figure number</th>
<th>Figure title including program name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>A-1</td>
<td>Assembler error detection codes</td>
</tr>
<tr>
<td>A</td>
<td>A-2</td>
<td>Assembler error messages</td>
</tr>
<tr>
<td>C</td>
<td>A-3</td>
<td>FORTRAN error codes</td>
</tr>
<tr>
<td>C</td>
<td>A-4</td>
<td>FORTRAN error messages</td>
</tr>
<tr>
<td>D</td>
<td>A-5</td>
<td>DUP/MUP error messages</td>
</tr>
<tr>
<td>E</td>
<td>A-6</td>
<td>System loader error messages</td>
</tr>
<tr>
<td>G</td>
<td>A-7</td>
<td>SGJP error messages</td>
</tr>
<tr>
<td>J</td>
<td>A-8</td>
<td>RJE work station error messages</td>
</tr>
<tr>
<td>J</td>
<td>A-9</td>
<td>RJE work station messages</td>
</tr>
<tr>
<td>M</td>
<td>A-10</td>
<td>Phase 1. System control record program error messages</td>
</tr>
<tr>
<td>M</td>
<td>A-11</td>
<td>Phase 2. System control record program error messages</td>
</tr>
<tr>
<td>-</td>
<td>A-12</td>
<td>SYSUP - DCOM update error messages</td>
</tr>
<tr>
<td>Note</td>
<td>A-13</td>
<td>RPG compiler error notes</td>
</tr>
<tr>
<td>R</td>
<td>A-14</td>
<td>Core load builder error messages</td>
</tr>
<tr>
<td>S</td>
<td>A-15</td>
<td>Auxiliary supervisor error messages</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>Monitor system mainline programs messages</td>
</tr>
</tbody>
</table>
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 | The displacement is set to zero.
An attempt has been made to specify a displacement field, directly or indirectly, outside the range of -128 to +127.
C | Condition code error | The displacement is set to zero.
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.
F | Format code error | 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.
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.
L | Label error | The label is ignored.
An invalid symbol is detected in the label field.
M | Multiply defined label error | 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).
A duplicate symbol is encountered in the label field.
O | Operation code error | The statement is ignored and the address counter is incremented by 2. If the op code is punched beginning in column 26, the character punched in column 26 will not appear in the listing.
An operation code is not valid.
An ISS, ILS, ENT, LIBR, SPR, EPR, or ABS is incorrectly placed.
Q | Warning flag | A possible problem code is detected; that is, a modify memory statement with a displacement of zero.
R | Relocation error | The expression is set to zero.
An expression does not have a valid relocation.
The displacement is set to zero.
The specified origin is ignored.
The operand is assumed to be zero.
An absolute displacement is not specified.
The specified origin is ignored.
The operand is assumed to be zero.
An absolute origin is specified in a relocatable program.
The operand is assumed to be zero.
An absolute operand is not specified in a BSS or BES statement.
A relocatable operand is not in an END statement of a relocatable mainline program.
The operand of an ENT statement is not relocatable.
Columns 9 through 12 are left blank; the entry is assumed to be relative zero.
The statement is ignored.

Figure A-1 (Part 1 of 2). Assembler error detection codes
### Assembler Error Codes

<table>
<thead>
<tr>
<th>Flag</th>
<th>Coding error</th>
<th>Assembler action</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Syntax error</td>
<td>The expression is set to zero.</td>
</tr>
<tr>
<td></td>
<td>An invalid expression (that is, an invalid symbol, adjacent operators, invalid constant) is used.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An invalid character is used in a record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The main program entry point is not specified as the operand in an END statement.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The syntax of an EBC statement is incorrect (that is, a delimiter is not in column 35, a zero character count).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An invalid label is used as an operand in an ENT or ISS statement.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An operand label occurs in more than one ENT statement.</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Tag error</td>
<td>A tag of zero is assumed.</td>
</tr>
<tr>
<td></td>
<td>Column 33 contains a character other than blank, 0, 1, 2, or 3 instruction statement.</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>Undefined symbol</td>
<td>The value of the expression is set to absolute zero.</td>
</tr>
<tr>
<td></td>
<td>A symbol used in an expression is not defined.</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>An x- or y-coordinate, or both, is not within the specified range; or an operand is invalid.</td>
<td>The operand is set to zero.</td>
</tr>
<tr>
<td>X</td>
<td>A character other than R or I is in column 32; or a character other than D or N is in column 33.</td>
<td>The field is set to zero.</td>
</tr>
<tr>
<td>Z</td>
<td>An invalid condition is in a conditional branch or interrupt order.</td>
<td>The condition bits in the first word are set to zero.</td>
</tr>
</tbody>
</table>

Figure A-1 (Part 2 of 2). Assembler error detection codes
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>A01 MINIMUM W.S. NOT AVAILABLE ... ASSEMBLY TERMINATED</td>
<td>The available working storage is less than the specified number of overflow sectors plus nine.</td>
<td>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.</td>
</tr>
<tr>
<td>A02 SYMBOL TABLE OVERFLOW ... ASSEMBLY TERMINATED</td>
<td>The number of sectors of symbol table overflow is greater than the number of overflow sectors available.</td>
<td>Use an *OVERFLOW SECTORS control record to increase the number of overflow sectors for this assembly (maximum 32 sectors).</td>
</tr>
<tr>
<td>A03 DISK OUTPUT EXCEEDS W.S.</td>
<td>Intermediate output (pass 1) or final DSF output (pass 2) exceeds the capacity of working storage less the specified number of overflow sectors.</td>
<td>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.</td>
</tr>
<tr>
<td>A04 SAVE SYMBOL TABLE INHIBITED</td>
<td>One of the following occurs when an *SAVE SYMBOL TABLE control record is used: 1. The program is relocatable. 2. The program contains assembly errors. 3. The source program contains more than 100 symbols.</td>
<td>Add an ABS statement to your program and reassemble. Correct the program errors and reassemble. Reduce the number of symbols and reassemble.</td>
</tr>
<tr>
<td>A05 XXX ERRONEOUS ORG, BSS, OR EQU STATEMENTS</td>
<td>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.</td>
<td>When forward references are attempted, correct them and reassemble the program.</td>
</tr>
<tr>
<td>A06 LOAD BLANK CARDS</td>
<td>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).</td>
<td>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. <strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>

Figure A-2 (Part 1 of 2). Assembler error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>A07 ABOVE CONTROL STATEMENT INVALID</td>
<td>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.</td>
<td>The control record is ignored.</td>
</tr>
<tr>
<td>A08 MACLIB UNDEFINED</td>
<td>An attempt is made to define a stored macro when a macro library is not associated with this assembly.</td>
<td>Reassemble specifying a valid macro library.</td>
</tr>
<tr>
<td>A09 PARAMETER LIST OVERFLOW . . . ASSEMBLY TERMINATED</td>
<td>The disk parameter-list spill area is undefined or exceeded.</td>
<td>Reassemble specifying a larger parameter-list disk area (see &quot;OVERFLOW SECTORS&quot; in Chapter 5).</td>
</tr>
<tr>
<td>A10 MACRO AREA OVERFLOW . . . ASSEMBLY TERMINATED</td>
<td>The disk area for macro definitions is undefined or exceeded.</td>
<td>Reassemble specifying a larger macro-definition disk area (see &quot;OVERFLOW SECTORS&quot; in Chapter 5).</td>
</tr>
<tr>
<td>A12 NEST LEVEL EXCEEDS 20 . . . ASSEMBLY TERMINATED</td>
<td>An attempt is made to nest more than 20 macro calls.</td>
<td>Redefine the macro nest and reassemble.</td>
</tr>
<tr>
<td>A21 *LEVEL CONTROL STATEMENT MISSING</td>
<td>A program is assembled as an ISS subroutine without the required *LEVEL control record.</td>
<td>Reassemble using an *LEVEL control record.</td>
</tr>
<tr>
<td>A22 INVALID LIST DECK OPTION . . . ASSEMBLY TERMINATED</td>
<td>LIST DECK or LIST DECK E option is specified when macros are called.</td>
<td>Reassemble and do not specify either LIST DECK or LIST DECK E options.</td>
</tr>
</tbody>
</table>

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.
 YYYYYY 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),J(5,8)  (recall that the 1 in column 6 indicates a continuation line)
DIMENSION J(3,2,6))  (error C 16)
FORMAT (150,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.
<table>
<thead>
<tr>
<th>Error code</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C01</td>
<td>Nonnumeric character in statement number</td>
</tr>
<tr>
<td>C02</td>
<td>More than 5 continuation cards, or continuation card out of sequence</td>
</tr>
<tr>
<td>C03</td>
<td>Syntax error in CALL LINK or CALL EXIT statement</td>
</tr>
<tr>
<td>C04</td>
<td>Unrecognizable, misspelled, or incorrectly formed statement</td>
</tr>
<tr>
<td>C05</td>
<td>Statement out of sequence</td>
</tr>
<tr>
<td>C06</td>
<td>A statement follows a STOP, RETURN, CALL LINK, CALL EXIT, or GO TO statement, or an IF statement does not have a statement number</td>
</tr>
<tr>
<td>C07</td>
<td>Name longer than 5 characters, or name not starting with an alphabetic character</td>
</tr>
<tr>
<td>C08</td>
<td>Incorrect or missing subscript within dimension information (DIMENSION, COMMON, REAL, or INTEGER)</td>
</tr>
<tr>
<td>C09</td>
<td>Duplicate statement number</td>
</tr>
<tr>
<td>C10</td>
<td>Syntax error in COMMON statement</td>
</tr>
<tr>
<td>C11</td>
<td>Duplicate name in COMMON statement</td>
</tr>
<tr>
<td>C12</td>
<td>Syntax error in FUNCTION or SUBROUTINE statement</td>
</tr>
<tr>
<td>C13</td>
<td>Parameter (dummy argument) appears in COMMON statement</td>
</tr>
<tr>
<td>C14</td>
<td>Name appears twice as a parameter in SUBROUTINE or FUNCTION statement</td>
</tr>
<tr>
<td>C15</td>
<td>*IOCS control record in a subprogram</td>
</tr>
<tr>
<td>C16</td>
<td>Syntax error in DIMENSION statement</td>
</tr>
<tr>
<td>C17</td>
<td>Subprogram name in DIMENSION statement</td>
</tr>
<tr>
<td>C18</td>
<td>Name dimensioned more than once, or not dimensioned on first appearance of name.</td>
</tr>
<tr>
<td>C19</td>
<td>Syntax error in REAL, INTEGER, or EXTERNAL statement</td>
</tr>
<tr>
<td>C20</td>
<td>Subprogram name in REAL or INTEGER statement, or a FUNCTION subprogram containing its own name in an EXTERNAL statement</td>
</tr>
<tr>
<td>C21</td>
<td>Name in EXTERNAL that is also in a COMMON or DIMENSION statement</td>
</tr>
<tr>
<td>C22</td>
<td>IFIX or FLOAT in EXTERNAL statement</td>
</tr>
<tr>
<td>C23</td>
<td>Invalid real constant</td>
</tr>
<tr>
<td>C24</td>
<td>Invalid integer constant</td>
</tr>
<tr>
<td>C25</td>
<td>More than 15 dummy arguments, or duplicate dummy argument in statement function argument list</td>
</tr>
<tr>
<td>C26</td>
<td>Right parenthesis missing from a subscript expression</td>
</tr>
<tr>
<td>C27</td>
<td>Syntax error in FORMAT statement</td>
</tr>
<tr>
<td>C28</td>
<td>FORMAT statement without statement number</td>
</tr>
<tr>
<td>C29</td>
<td>Field width specification greater than 145</td>
</tr>
<tr>
<td>C30</td>
<td>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</td>
</tr>
<tr>
<td>C31</td>
<td>Subscript error in EQUIVALENCE statement</td>
</tr>
<tr>
<td>C32</td>
<td>Subscripted variable in a statement function</td>
</tr>
<tr>
<td>C33</td>
<td>Incorrectly formed subscript expression</td>
</tr>
<tr>
<td>C34</td>
<td>Undefined variable in subscript expression</td>
</tr>
<tr>
<td>C35</td>
<td>Number of subscripts in a subscript expression, and/or the range of the subscripts does not agree with the dimension information</td>
</tr>
<tr>
<td>C36</td>
<td>Invalid arithmetic statement or variable; or, in a FUNCTION subprogram the left side of an arithmetic statement is a dummy argument or in COMMON</td>
</tr>
<tr>
<td>C37</td>
<td>Syntax error in IF statement</td>
</tr>
<tr>
<td>C38</td>
<td>Invalid expression in IF statement</td>
</tr>
</tbody>
</table>

Figure A-3 (Part 1 of 3). FORTRAN error codes
<table>
<thead>
<tr>
<th>Error code</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C39</td>
<td>Syntax error or invalid simple argument in CALL statement</td>
</tr>
<tr>
<td>C40</td>
<td>Invalid expression in CALL statement</td>
</tr>
<tr>
<td>C41</td>
<td>Invalid expression to the left of an equal sign in a statement function</td>
</tr>
<tr>
<td>C42</td>
<td>Invalid expression to the right of an equal sign in a statement function</td>
</tr>
<tr>
<td>C43</td>
<td>In an IF, GO TO, or DO statement, a statement number is missing, invalid, incorrectly placed, or is the number of a FORMAT statement</td>
</tr>
<tr>
<td>C44</td>
<td>Syntax error in READ, WRITE or FIND statement</td>
</tr>
<tr>
<td>C45</td>
<td>*10CS record missing with a READ or WRITE statement (mainline program only)</td>
</tr>
<tr>
<td>C46</td>
<td>FORMAT statement number missing or incorrect in a READ or WRITE statement</td>
</tr>
<tr>
<td>C47</td>
<td>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</td>
</tr>
<tr>
<td>C48</td>
<td>Syntax error in GO TO statement</td>
</tr>
<tr>
<td>C49</td>
<td>Index of a computed GO TO is missing, invalid, or not preceded by a comma</td>
</tr>
<tr>
<td>C50</td>
<td>*TRANSFER TRACE or *ARITHMETIC TRACE control record or CALL PDUMP statement present, with no *10CS control record in a mainline program</td>
</tr>
<tr>
<td>C51</td>
<td>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</td>
</tr>
<tr>
<td>C52</td>
<td>More than 25 nested DO statements</td>
</tr>
<tr>
<td>C53</td>
<td>Syntax error in DO statement</td>
</tr>
<tr>
<td>C54</td>
<td>Initial value in DO statement is zero</td>
</tr>
<tr>
<td>C55</td>
<td>In a FUNCTION subprogram the index of DO is a dummy argument or in COMMON</td>
</tr>
<tr>
<td>C56</td>
<td>Syntax error in BACKSPACE statement</td>
</tr>
<tr>
<td>C57</td>
<td>Syntax error in REWIND statement</td>
</tr>
<tr>
<td>C58</td>
<td>Syntax error in END FILE statement</td>
</tr>
<tr>
<td>C59</td>
<td>Syntax error in STOP statement</td>
</tr>
<tr>
<td>C60</td>
<td>Syntax error in PAUSE statement</td>
</tr>
<tr>
<td>C61</td>
<td>Integer constant in STOP or PAUSE statement greater than 9999</td>
</tr>
<tr>
<td>C62</td>
<td>Last executable statement before END statement is not a STOP, GO TO, IF, CALL LINK, CALL EXIT, or RETURN statement</td>
</tr>
<tr>
<td>C63</td>
<td>Statement contains more than 15 different subscript expressions</td>
</tr>
<tr>
<td>C64</td>
<td>Statement too long to be scanned, because of compiler expansion of subscript expressions or compiler addition of generated temporary storage locations</td>
</tr>
<tr>
<td>C65*</td>
<td>All variables undefined in an EQUIVALENCE list</td>
</tr>
<tr>
<td>C66*</td>
<td>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</td>
</tr>
<tr>
<td>C67*</td>
<td>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</td>
</tr>
<tr>
<td>C68</td>
<td>Syntax error in an EQUIVALENCE statement; or an illegal variable name in an EQUIVALENCE list</td>
</tr>
<tr>
<td>C69</td>
<td>Subprogram does not contain a RETURN statement, or a mainline program contains a RETURN statement</td>
</tr>
<tr>
<td>C70</td>
<td>No DEFINE FILE statement in a mainline program that has disk READ, WRITE, or FIND statements</td>
</tr>
<tr>
<td>C71</td>
<td>Syntax error in DEFINE FILE statement</td>
</tr>
<tr>
<td>C72</td>
<td>Duplicate DEFINE FILE statement, more than 75 DEFINE FILES, or DEFINE FILE statement in subprogram</td>
</tr>
</tbody>
</table>

Figure A-3 (Part 2 of 3), FORTRAN error codes
<table>
<thead>
<tr>
<th>Error code</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C73</td>
<td>Syntax error in record number of disk READ, WRITE, or FIND statement</td>
</tr>
<tr>
<td>C74</td>
<td>Defined file exceeds disk storage size</td>
</tr>
<tr>
<td>C75</td>
<td>Syntax error in DATA statement</td>
</tr>
<tr>
<td>C76</td>
<td>Names and constants in a DATA statement not in a one-to-one correspondence</td>
</tr>
<tr>
<td>C77</td>
<td>Mixed mode in DATA statement</td>
</tr>
<tr>
<td>C78</td>
<td>Invalid hollerith constant in a DATA statement (see &quot;Length of FORTRAN DATA Statement&quot; in Chapter 6)</td>
</tr>
<tr>
<td>C79</td>
<td>Invalid hexadecimal specification in a DATA statement</td>
</tr>
<tr>
<td>C80</td>
<td>Variable in a DATA statement not used elsewhere in the program or dummy variable in DATA statement</td>
</tr>
<tr>
<td>C81</td>
<td>COMMON variable loaded with a DATA specification</td>
</tr>
<tr>
<td>C82</td>
<td>DATA statement too long to compile, due to internal buffering. Refer to the section TIPS FOR FORTRAN PROGRAMMERS</td>
</tr>
</tbody>
</table>

* 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
FORTRAN Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C85 ORIGIN IN SUBPROGRAM</td>
<td>An ORIGIN control record was detected in a subprogram compilation.</td>
</tr>
<tr>
<td>C86 INVALID ORIGIN</td>
<td>An attempt has been made to relocate a word at an address exceeding 7FFF (hexadecimal).</td>
</tr>
<tr>
<td>C96 WORKING STORAGE EXCEEDED</td>
<td>The working storage area on disk is too small to accommodate the compiled program in disk system format.</td>
</tr>
<tr>
<td>C97 PROGRAM LENGTH EXCEEDS CAPACITY</td>
<td>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.</td>
</tr>
<tr>
<td>C98 SUBROUTINE INITIALIZE TOO LARGE</td>
<td>During compilation of subprograms a subroutine initialize statement (CALL SUBIN) is generated.</td>
</tr>
<tr>
<td></td>
<td>The CALL SUBIN statement initializes all references to dummy variables contained within the subprogram to the appropriate core location in the calling program.</td>
</tr>
<tr>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>[ S = 5 + \text{ARG} + \text{N} ]</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>C99 CORE REQUIREMENTS EXCESSIVE</td>
<td>The error occurs when the total core requirements exceed 32767 words.</td>
</tr>
</tbody>
</table>

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
- 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.
### DUP/MUP Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D01 NAME IS NOT PRIME ENTRY</td>
<td>The primary entry point name of the program in working storage does not match the name on the DUP control record.</td>
<td></td>
</tr>
<tr>
<td>D02 INVALID HEADER RECORD TYPE</td>
<td>One of the following is detected: 1. A non-DSF program 2. A mispositioned header 3. Foreign data 4. An erroneous subtype</td>
<td></td>
</tr>
<tr>
<td>D03 INVALID HEADER LENGTH</td>
<td>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.</td>
<td>Delete the specified entry point name before storing this subroutine.</td>
</tr>
<tr>
<td>D06 SECONDARY ENTRY XXXXX IN LET</td>
<td>The specified secondary entry point name is already in LET.</td>
<td>Delete the specified name from LET or FLET before storing this program or data file.</td>
</tr>
<tr>
<td>D06 ENTRY POINT NAME ALREADY IN LET/FLET</td>
<td>The specified name is already in LET or FLET.</td>
<td></td>
</tr>
<tr>
<td>D12 INVALID DISK I/O SPECIFIED</td>
<td>The disk I/O subroutine coded (column 9) on the STORECI control record is other than 0, 1, N, Z, or blank.</td>
<td></td>
</tr>
<tr>
<td>D13 INVALID FUNCTION FIELD</td>
<td>An invalid DUP function is specified on the DUP control record.</td>
<td></td>
</tr>
<tr>
<td>D14 INVALID FROM (CC 13-14)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D15 INVALID TO FIELD (CC 17-18)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D16 INVALID NAME FIELD (CC 21-25)</td>
<td>One of the following: 1. A required name is not specified. 2. The specified name contains a syntax error.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-5 (Part 1 of 8). DUP/MUP error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D17 INVALID COUNT FIELD (CC 27-30)</td>
<td>Columns 27 through 30 are blank or include alphabetic characters. The count field requires a decimal number.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D18 INVALID FUNCTION DURING TEMPORARY JOB</td>
<td>This function is not allowed during the JOB T mode.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D19 CARTRIDGE NOT ON SYSTEM</td>
<td>The cartridge specified as the TO or FROM cartridge is not specified on the JOB control record as being used for this job.</td>
<td>Correct the cartridge ID and retry.</td>
</tr>
<tr>
<td>D20 CARTRIDGE ID OUTSIDE VALID RANGE (0001-7FFF)</td>
<td>The replacement version of the program or data file is larger than the current stored version.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D21 INVALID STOREMOD. SIZE OF REPLACEMENT EXCEEDS SIZE OF ORIGINAL</td>
<td>The system overlay subtype indicator (column 11) on a STORE control record is not in the range 0 through 9.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D22 PROGRAM NOT IN WORKING STORAGE</td>
<td>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.</td>
<td>Correct the cartridge ID and retry.</td>
</tr>
<tr>
<td>D23 INVALID SYSTEM OVERLAY SUBTYPE SPECIFIED</td>
<td>The system overlay subtype indicator (column 11) on a STORE control record is not in the range 0 through 9.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D24 COUNT FIELD TOO LARGE</td>
<td>One of the following: 1. The program not in working storage. 2. Column 31 is not a minus sign.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D25 REQUIRED FORMAT NOT IN W.S.</td>
<td>During a STOREMOD, the format of the LET or FLET entry does not agree with the format in working storage.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D26 NAME NOT FOUND IN LET/FLET</td>
<td>The name specified on a DELETE or DUMP control record is not in LET or FLET.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D27 SOURCE NOT IN DSF</td>
<td>The format indicator of the FROM cartridge indicates that working storage on this cartridge does not contain a DSF program.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
<tr>
<td>D30 INVALID RECORD TYPE</td>
<td>An invalid type binary record has been read when storing from cards or paper tape.</td>
<td>Delete the old version of the program or data file and retry.</td>
</tr>
</tbody>
</table>

Figure A-5 (Part 2 of 8). DUP/MUP error messages
### DUP/MUP Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D31 PROGRAM OR DATA</td>
<td>The number of disk blocks required to store a program or data file exceeds the</td>
<td>You must delete a program with a LET or FLET entry of similar size before this program can be stored.</td>
</tr>
<tr>
<td>DESTINATION DISK AREA</td>
<td>amount of space available in the specified TO field.</td>
<td></td>
</tr>
<tr>
<td>D32 INVALID CORE IMAGE</td>
<td>The core load builder has inhibited the continuation of STORECI. The specific</td>
<td></td>
</tr>
<tr>
<td>CONVERSION</td>
<td>reason has been printed by the core load builder (see “Core Load Builder Error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Messages” in this appendix).</td>
<td></td>
</tr>
<tr>
<td>D33 LET/FLET OVERFLOW.</td>
<td>A ninth sector of LET/FLET is required (or a seventh sector of LET on a non-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>system cartridge) for the LET/FLET entry.</td>
<td></td>
</tr>
<tr>
<td>D41 INVALID STORECI</td>
<td>A control record read after a STORECI is not a LOCAL, NOCAL, FILES, or G2250</td>
<td></td>
</tr>
<tr>
<td>CONTROL RECORD</td>
<td>record, or a mainline name is not specified on a LOCAL or NOCAL record, or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a mainline name is specified on a G2250 record, or the name specified on the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOCAL or NOCAL record does not match the name on the STORECI card.</td>
<td></td>
</tr>
<tr>
<td>D42 STORECI CONTROL</td>
<td>LOCAL, NOCAL, FILES, and G2250 control records are intermixed.</td>
<td>All records of a given type must be loaded together.</td>
</tr>
<tr>
<td>RECORDS INCORRECTLY ORDERED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D43 INCORRECT</td>
<td>A comma at the end of a record indicates continuation to the next record;</td>
<td></td>
</tr>
<tr>
<td>CONTINUATION</td>
<td>however, it is not continued.</td>
<td></td>
</tr>
<tr>
<td>D44 ILLEGAL CHARACTER IN</td>
<td>An illegal character, probably a blank, is in the record.</td>
<td></td>
</tr>
<tr>
<td>RECORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D45 ILLEGAL FILE NUMBER</td>
<td>One of the following:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. A nonnumeric character is in a file number.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. A file number is more than 5 characters long.</td>
<td></td>
</tr>
<tr>
<td>D46 ILLEGAL NAME</td>
<td>One of the following:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. A name is more than 5 characters long.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. A name contains characters other than A through Z, 0 through 9, or $.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. A name contains embedded blanks.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-5 (Part 3 of 8). DUP/MUP error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D47 ILLEGAL CARTRIDGE ID</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D48 SCRA BUFFER OVERFLOW</td>
<td>The supervisor control record area (SCRA) cannot contain all the LOCAL, NOCAL, FILES, or G2250 information.</td>
<td></td>
</tr>
<tr>
<td>D70 LAST ENTRY IN LET/FLET NOT 1DUMY</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D71 1DUMY ENTRY IN LET/FLET IS FOLLOWED BY A SECONDARY ENTRY POINT</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D72 FIRST ENTRY IN LET/FLET SECTOR IS A SECONDARY ENTRY POINT</td>
<td>The LET/FLET table is improperly constructed; the first entry is not a primary entry.</td>
<td></td>
</tr>
<tr>
<td>D80 FIXED AREA PRESENT</td>
<td>The FORTRAN compiler, RPG compiler, or assembler cannot be eliminated if a fixed area is defined on the disk.</td>
<td></td>
</tr>
<tr>
<td>D81 ASSEMBLER NOT IN SYSTEM</td>
<td>The assembler has been previously deleted from the system.</td>
<td></td>
</tr>
<tr>
<td>D82 FORTRAN NOT IN SYSTEM</td>
<td>The FORTRAN compiler has been previously deleted from the system.</td>
<td></td>
</tr>
<tr>
<td>D83 INCREASE VALUE IN COUNT FIELD (CC 27-30)</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D84 DEFECTIVE SLET</td>
<td>The cartridge must be reloaded.</td>
<td></td>
</tr>
<tr>
<td>D85 FIXED AREA NOT PRESENT</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>
### DUP/MUP Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D86 DECREASE VALUE IN COUNT FIELD</td>
<td>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). If a large program is in working storage this error may also occur. If you do not need the contents of working storage, precede the //DUP card with a //JOB card to reinitialize #WSCT in DCOM. If the contents of working storage are needed, save the required information, then run with the following cards: //JOB, //DUP, and *DEFINE FIXED AREA. 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.</td>
<td>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 *DFILE or *STOREDATA control record.</td>
</tr>
<tr>
<td>D87 RPG NOT IN SYSTEM</td>
<td>The RPG compiler has been previously deleted from the system.</td>
<td></td>
</tr>
<tr>
<td>D88 COBOL NOT IN SYSTEM</td>
<td>The COBOL compiler (a program product) has been previously deleted from the system.</td>
<td></td>
</tr>
<tr>
<td>D90 CHECK SUM ERROR</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D92 INVALID DISKZ CALL.</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D93 CARTRIDGE OVERFLOW</td>
<td>While performing a DUP function, an attempt has been made to read or write a sector beyond 1599 decimal.</td>
<td></td>
</tr>
<tr>
<td>D100 LIBRARY NOT FOUND</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-5 (Part 5 of 8). DUP/MUP error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
</table>
| 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
### DUP/MUP Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D104 MACRO NOT FOUND</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>D105 SEQUENCE NUMBER NOT FOUND</td>
<td>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.</td>
<td>Place a correct sequence number on the statement in error, and rerun the portion of the job that is affected.</td>
</tr>
<tr>
<td>D106 LIBRARY NOT SPECIFIED</td>
<td>An attempt was made to operate on a macro without specifying a macro library. Processing continues with the next LIB, BUILD, or ENDUP statement.</td>
<td>Place a LIB or BUILD statement before the statement before the statement in error, and rerun the portion of the job that is affected.</td>
</tr>
<tr>
<td>D107 SPILL OVERFLOW</td>
<td>Macro text insertions have caused the capacity of working storage spill to be exceeded.</td>
<td>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.</td>
</tr>
<tr>
<td>D108 CONTROL STATEMENT READ</td>
<td>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.</td>
<td>Insert a NAME statement, and rerun the portion of the job that is affected.</td>
</tr>
<tr>
<td>D109 NAME STATEMENT NOT FOUND</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>D110 INVALID NAME</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

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 | 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.
Note: N may be truncated if the field size is exceeded.

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.

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: In addition to the DUP error messages just listed, the following message:
NO SUCH ERROR MESSAGE 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.

Note. Your response
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.

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.

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.
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>From phases 1 and 2</td>
<td></td>
</tr>
<tr>
<td>E01 CHECKSUM ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>E02 INVALID RECORD OR BLANK</td>
<td>Follow procedure A or restart initial load.</td>
</tr>
<tr>
<td>E03 SEG ERROR OR MISSING RECORDS</td>
<td>Follow procedure A or restart initial load. The missing record may be end-of-program record.</td>
</tr>
<tr>
<td>E04 ORG BACKWARD</td>
<td>Inspect the deck for records missing or out of sequence. Correct the deck and restart from the record in error.</td>
</tr>
<tr>
<td>E05 INITIALIZE THE CARTRIDGE</td>
<td>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.</td>
</tr>
<tr>
<td>From phase 1 only</td>
<td></td>
</tr>
<tr>
<td>E11 INVALID DRIVE NO.</td>
<td>Set all bit switches off. Set bit switches to select physical drive number and press PROGRAM START.</td>
</tr>
<tr>
<td></td>
<td>Drive 0—All switches off</td>
</tr>
<tr>
<td></td>
<td>Drive 1—Switch 15 on</td>
</tr>
<tr>
<td></td>
<td>Drive 2—Switch 14 on</td>
</tr>
<tr>
<td></td>
<td>Drive 3—Switches 14 and 15 on</td>
</tr>
<tr>
<td></td>
<td>Drive 4—Switch 13 on</td>
</tr>
<tr>
<td></td>
<td>Drive 5—Switches 13 and 15 on</td>
</tr>
<tr>
<td>E12 ID SECTOR DATA INVALID</td>
<td>Initialize using DCIP or DISC and follow with an initial load.</td>
</tr>
<tr>
<td>E13 CONFIG DECK ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>E14 FILE PROTECT ADDR TOO HIGH</td>
<td>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.</td>
</tr>
<tr>
<td>E15 PHID RECORD ERROR</td>
<td>Follow procedure A or reload and restart.</td>
</tr>
<tr>
<td>E16 INITIAL LOAD THE CARTRIDGE</td>
<td>The ID sector indicates that this cartridge has not been loaded since initialization by DCIP or DISC. Only an initial load may be performed.</td>
</tr>
<tr>
<td>E17 ERROR IN LOAD MODE RECORD</td>
<td>Follow procedure A or restart load.</td>
</tr>
<tr>
<td>E18 PAPER TAPE ERROR</td>
<td>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.</td>
</tr>
<tr>
<td>E19 INVALID SLET/RELOAD TABLE CHECKSUM</td>
<td>System loader will ignore the checksum and continue if PROGRAM START is pressed. However, the cartridge should be initialized and an initial load performed.</td>
</tr>
<tr>
<td>From phase 2 only</td>
<td></td>
</tr>
<tr>
<td>E20 FIXED AREA PRESENT</td>
<td>Programs may not be added to a cartridge with a fixed area defined. Press PROGRAM START to restore the resident image and DCOM.</td>
</tr>
<tr>
<td>E21 SYSTEM DECK ERROR</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 processing of the reload table, 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 B1 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
SGJP 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.

<table>
<thead>
<tr>
<th>Error number (if any) and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>G01 INITIALIZATION FAILURE</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>G02 LINE ERROR</td>
<td>An attempt to transmit data to the System/360 is unsuccessful because of an I/O error; standard retries are unsuccessful.</td>
<td>Using the console keyboard, type either an R to retry the operation or a C to cancel SGJP.</td>
</tr>
<tr>
<td>G03 SYNCHRONIZATION ERROR</td>
<td>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.</td>
<td>Using the console keyboard, type either an R to retry the operation or a C to cancel SGJP.</td>
</tr>
<tr>
<td>IKyxxxz message text THE SATELLITE GRAPHIC JOB PROCESSOR MUST RESTART</td>
<td>SGJP is terminated because an internal error occurred. If the error recurs, refer to the publication, <em>IBM System/360 Operating System Messages and Codes</em>, GC28-8631, under the message code (IKyxxxz) for further explanation of the error condition.</td>
<td>Perform the END function, which causes the LOG ON frame to reappear. Perform the LOG ON operation again.</td>
</tr>
<tr>
<td>IKyxxxz message text THE SATELLITE GRAPHIC JOB PROCESSOR MUST TERMINATE</td>
<td>SGJP must be terminated because an internal error occurred. If the error recurs, refer to the publication, <em>IBM System/360 Operating System Messages and Codes</em>, GC28-8631, under the message code (IKyxxxz) for further explanation of the error condition.</td>
<td>Perform the END function. This returns SGJP to the state it was in before the initial (CANCEL key) attention.</td>
</tr>
</tbody>
</table>

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.
## RJE Error Messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J01 INVALID CARD</td>
<td>The control card that contains the work station generation information is invalid or contains invalid information (see &quot;Generation of the 1130 RJE Work Station Program&quot; in Chapter 10).</td>
<td>The work station prepares to read a new data card.</td>
<td>Enter a valid data card.</td>
</tr>
<tr>
<td>J10 INVALID PRINTER</td>
<td>Information from the disk monitor system indicates that the principal print device is not an 1132 Printer or a 1403 Printer.</td>
<td>The work station program exits to the disk monitor supervisor.</td>
<td>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).</td>
</tr>
<tr>
<td>J11 INVALID READER</td>
<td>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.</td>
<td>The work station program exits to the disk monitor supervisor.</td>
<td>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).</td>
</tr>
<tr>
<td>J12 LOGICAL DRIVE X NOT IN SYSTEM</td>
<td>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.</td>
<td>The work station program exits to the disk monitor supervisor.</td>
<td>Change your exit parameters or ready the requested logical drive, and reload the RJE Work Station Program.</td>
</tr>
<tr>
<td>J13 TOO MANY EQUATS</td>
<td>The number of subroutines equated by you and the RJE program in the current job is more than 25.</td>
<td>The work station program exits to the disk monitor supervisor.</td>
<td>Reload the RJE Work Station Program with a smaller number of subroutines specified in the *EQUAT control record.</td>
</tr>
</tbody>
</table>

*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

<table>
<thead>
<tr>
<th>J14 DISK ERROR OCR</th>
<th>A permanent error is encountered while attempting to read data from disk during the initialization part of the RJE program.</th>
<th>The program continues according to your response.</th>
<th>Enter one of the following codes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCR</td>
<td></td>
<td></td>
<td>T — Indicates exit to the disk monitor supervisor requesting a terminating dump of the contents of core storage on the printer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X — Indicates exit to the disk monitor supervisor without printing the contents of core storage on the printer.</td>
</tr>
</tbody>
</table>

Figure A-8 (Part 1 of 5). RJE Work Station Program error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J20 RJSTART MISSING</td>
<td>The requirement for an RJSTART command is not satisfied.</td>
<td>The program waits for your response.</td>
<td>Enter an RJSTART command through the card reader, and press PROGRAM START on the console to resume processing.</td>
</tr>
<tr>
<td>J21 .. DATA INVALID</td>
<td>A .. DATA command contains invalid parameter. Note: This message is also printed if the requested logical disk drive is not present.</td>
<td>The program waits for your response.</td>
<td>Use the operator communication request facility (see message J90 in Figure A-9).</td>
</tr>
<tr>
<td>J22 INVALID INPUT</td>
<td>The input entered from the console keyboard does not start with the JECL identifier (..) followed by at least one blank.</td>
<td>The program waits for more input from the keyboard.</td>
<td>Enter a work station command or press EOF.</td>
</tr>
<tr>
<td>J23 INPUT ABORTED BY CENTRAL</td>
<td>The central system has terminated input from the work station and sends a message that explains why input was terminated (see &quot;Messages Sent to Work Stations&quot; in IBM System/360 Operating System Remote Job Entry, GC30-2006, for a list of the messages).</td>
<td>The program waits for input from the line.</td>
<td>When the message from the central system is printed, take the indicated action. To resume input, follow the procedures described under &quot;Console Keyboard Procedures&quot; in Chapter 10.</td>
</tr>
<tr>
<td>J51 LINE ERROR OCR=</td>
<td>An unrecoverable error is encountered while reading or writing on the communication line, or the line cannot be opened.</td>
<td>The RJE program closes the communication line, if it is open, and waits for your response.</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure A-8 (Part 2 of 5). RJE Work Station Program error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J52 DISK ERROR INPUT OCR=</td>
<td>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.</td>
<td>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.</td>
<td>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 keyboard. 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.</td>
</tr>
<tr>
<td>J53 DISK ERROR OUTPUT OCR=</td>
<td>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.</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>J54 DISK ERROR OCR=</td>
<td>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.</td>
<td>The program continues according to your response.</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure A-8 (Part 3 of 5). RJE Work Station Program error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J56 END OF DISK</td>
<td>You did not reserve space or reserved too little space on disk for user-exit output data sets.</td>
<td>Output from the central system is discontinued. The system continues as directed by your response.</td>
<td>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.</td>
</tr>
<tr>
<td>J56 CARD READER ERROR OCR=</td>
<td>An error has occurred on the card reader that requires your intervention.</td>
<td>The system waits for your response.</td>
<td>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.</td>
</tr>
<tr>
<td>J57 CARD PUNCH ERROR OCR=</td>
<td>An error has occurred on the card punch that requires your intervention.</td>
<td>The system waits for your response.</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure A-8 (Part 4 of 5). RJE Work Station Program error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J58 PRINTER ERROR OCR=</td>
<td>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.</td>
<td>The system waits for your response.</td>
<td>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.</td>
</tr>
<tr>
<td>J59 PREOPERATIVE ERROR CODE XXXX OCR=</td>
<td>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.</td>
<td>The system waits for your response.</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure A-8 (Part 5 of 5). RJE Work Station Program error messages
### RJE Messages

<table>
<thead>
<tr>
<th>Message number and message</th>
<th>Reason for message</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J90 OCR=</td>
<td>You have indicated that you want to communicate with the system by pressing PROGRAM STOP and PROGRAM START on the console keyboard.</td>
<td>The system waits for your response.</td>
<td>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):</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A — Indicates that input is available at the card reader.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B — Indicates that commands are to be submitted from the console keyboard.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D — Indicates that receiving output is to be discontinued.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N — Indicates that the system ignore the request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T — Indicates exit to the disk monitor supervisor, requesting a terminating dump of the contents of core storage on the printer.</td>
</tr>
</tbody>
</table>

**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
### RJE Messages

<table>
<thead>
<tr>
<th>Message number and message</th>
<th>Reason for message</th>
<th>System action</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>J92 (Continued)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>f — The IBM System/360 Operating System record format, where f may have the following values:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 — Fixed unblocked records</td>
<td>The K.B. SELECT light on the console keyboard is turned on, and the program waits for input from the keyboard.</td>
<td>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.)</td>
</tr>
<tr>
<td></td>
<td>2 — Fixed blocked records</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 — Variable unblocked records</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 — Variable blocked records</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 — Undefined records</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>x — The logical disk drive number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>aaa — The starting sector address of the data set in hexadecimal.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>J93</td>
<td>PROCEED</td>
<td>The system waits for your action.</td>
<td>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.</td>
</tr>
<tr>
<td>J94</td>
<td>PUNCHED OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A-9 (Part 2 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:

\[
\text{LOG DRIVE CART SPEC CART AVAIL PHY DRIVE} \\
\quad XXXX \quad XXXX \quad XXXX \quad XXXX
\]

where

- \text{LOG DRIVE} is the drive number specified on the JOB control record (or in the calling sequence of the SYSUP subroutine).
- \text{CART SPEC} is the specified cartridge ID.
- \text{CART AVAIL} is the available cartridge ID.
- \text{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:

\[
\text{V2MXX ACTUAL XXK CONFIG XXK}
\]

where

- \text{V2MXX} is the current version and modification level of the 1130 Disk Monitor System.
- \text{ACTUAL XXK} indicates the physical core size.
- \text{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.
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>M11 INVALID MONITOR CONTROL RECORD</td>
<td>A // record was not recognized as a valid monitor control record.</td>
</tr>
<tr>
<td>M12 EXECUTION SUPPRESSED</td>
<td>$NXEQ was set upon detection of an error that would prevent successful execution by the system. Execution is bypassed.</td>
</tr>
<tr>
<td>M13 DUP SUPPRESSED</td>
<td>$NDUP was set upon detection of an error that would prevent successful DUP operation. DUP is bypassed.</td>
</tr>
<tr>
<td>M14 SYSTEM PROGRAM DETECTED MONITOR CONTROL RECORD</td>
<td>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.</td>
</tr>
<tr>
<td>M15 ILLEGAL CARTRIDGE ID</td>
<td>A cartridge ID contains an illegal character or is a negative number. The job is terminated.</td>
</tr>
<tr>
<td>M16 PROGRAM VOIDED</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

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 (*) precedes 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:

<table>
<thead>
<tr>
<th>Seq. No.</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTENDED FILE DEF.</td>
<td>xxxxx</td>
</tr>
<tr>
<td>EXT. AND/OR INPUT DIAGNOSTICS</td>
<td></td>
</tr>
<tr>
<td>EXTENDED CALCULATION SPECIFICATION DIAGNOSTICS</td>
<td>xxxxx</td>
</tr>
<tr>
<td>EXTENDED OUTPUT SPECIFICATION DIAGNOSTICS</td>
<td>xxxxx</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>FILE TYPE COL 15 INVALID</td>
<td>File Type entry is not I, O, U, or C, or is blank.</td>
<td>I is assumed.</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>PROC MODE COL 28 INVALID</td>
<td>Mode of Processing entry is not L, R, or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>REC ADDR COL 29-30 INVALID</td>
<td>Length of Record Address Field (or key length) entry is invalid or is blank.</td>
<td>08 is assumed.</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>REC ADDR TYPE COL 31 INVALID. CORRECT ENTRY ASSUM</td>
<td>Warning only. The correct value for the file type (column 32) is assumed.</td>
<td>Blank is assumed for sequential files. K is assumed for ISAM files.</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>TABLE FILE COL 16 REQ E COL 39. E ASSUM</td>
<td>Extension Code entry must be E if File Designation entry is T (table file).</td>
<td>E is assumed.</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>FILE DESIGN INVALID WITH INPUT FILE</td>
<td>File Designation entry column 16 is not P, S, R, C, or T with an input file (I in column 15).</td>
<td>P is assumed.</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>OF IND COL 33-34 INVALID BLK ASSUM</td>
<td>Overflow Indicator entry is invalid for the device type specified.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>FILE TYPE COL 15 INVALID 0 ASSUM</td>
<td>File Type entry is invalid with a printer device in columns 40 through 46.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>MULT PRI FILES DEF. SEC ASSUM</td>
<td>Only one primary file (P in column 16) is allowed. Other input files are designated as secondary (S in column 16).</td>
<td>Secondary is assumed for all but first input file.</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>FILE ORG COL 32 INVALID</td>
<td>File Organization entry is not I, numeric (1 through 9), or blank; or, two I/O areas are specified for a table file.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td>EXT CODE COL 39 NOT BLK BLK ASSUM</td>
<td>Extension Code must be blank for output files.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td>EOF COL 17 INVALID E ASSUM</td>
<td>End of File entry is not E or blank.</td>
<td>E is assumed.</td>
</tr>
<tr>
<td>13</td>
<td>F</td>
<td>SEQ COL 18 INVALID A ASSUM</td>
<td>Sequence entry not A, D, or blank.</td>
<td>A is assumed.</td>
</tr>
<tr>
<td>14</td>
<td>F</td>
<td>FILE DESIG COL 16 NOT BLK. BLK ASSUM</td>
<td>File Designation entry is not blank for an output file.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>C IN FILE TYPE COL 15 INVALID WITH DEVICE</td>
<td>File Type entry C requires card read punch in device columns 40 through 46.</td>
<td>READ 42 is assumed.</td>
</tr>
<tr>
<td>17</td>
<td>F</td>
<td>FILE FMT INVALID. F ASSUM</td>
<td>File Format (column 19) is not F. 1130 RPG uses fixed length records only.</td>
<td>F is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 1 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>F</td>
<td>BLOCK LNG COL 20-23 NOT BLK. BLK ASSUM</td>
<td>Block Length must be blank for 1130 RPG.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>20</td>
<td>F</td>
<td>REC LNG COL 24-27 INVALID. 120 ASSUM PRINTER. ALL ELSE 80</td>
<td>Record Length is improperly specified or is blank.</td>
<td>120 is assumed for printer. 80 is assumed otherwise.</td>
</tr>
<tr>
<td>* 21</td>
<td>F</td>
<td>U IN FILE TYPE COL 15 INVALID WITH DEVICE</td>
<td>File Type entry U requires disk I/O in device columns 40 through 46.</td>
<td>DISK is assumed.</td>
</tr>
<tr>
<td>22</td>
<td>F</td>
<td>COL 17-18 INVALID WITH PRINTER. BLK ASSUM</td>
<td>End-of-File and Sequence entries are invalid with a printer.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>23</td>
<td>F</td>
<td>COL 28 INVALID WITH CHAIN FILE, R ASSUME</td>
<td>Mode of processing must be random for chain file.</td>
<td>R is assumed.</td>
</tr>
<tr>
<td>* 24</td>
<td>F</td>
<td>MORE THAN 8 SEC FILES DEF</td>
<td>The number of secondary files (S1 in column 16) exceeds the maximum allowable 8.</td>
<td>8 is assumed.</td>
</tr>
<tr>
<td>25</td>
<td>F</td>
<td>OF IND COL 33-34 INVALID BLK ASSUM</td>
<td>Overflow indicator not OF on OV.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>27</td>
<td>F</td>
<td>EOF COL 17 NOT BLK WITH OUTPUT. BLK ASSUM</td>
<td>End-of-File entry must be blank with output files.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>29</td>
<td>F</td>
<td>EXT CODE 39 INVALID. E ASSUM</td>
<td>Extension Code entry is not E or blank with input file.</td>
<td>E is assumed.</td>
</tr>
<tr>
<td>* 30</td>
<td>E</td>
<td>FROM FILENAME COL 11-18 INVALID</td>
<td>From Filename entry is missing or not left-justified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 31</td>
<td>E</td>
<td>FROM FILENAME COL 11-18 INVALID</td>
<td>From Filename entry was not defined on a File Description Specification form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 32</td>
<td>E</td>
<td>FROM FILENAME COL 11-18 INVALID</td>
<td>From Filename entry requires an E in Extension Code column on the File Description Specifications form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 33</td>
<td>E</td>
<td>CHAINING FLD COL 9-10 INVALID</td>
<td>Chaining Field entry is not C1, C2, or C3 for chaining file (same entry as columns 61 and 62 of Input Specifications form).</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 34</td>
<td>E</td>
<td>SEQ COL 7-8 INVALID</td>
<td>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).</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 35</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry is missing or not left-justified on RAF or chaining type specifications.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 36</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry was not defined on RAF or chaining type specifications on a File Description Specifications form.</td>
<td>Specification is dropped.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 2 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 37</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry is not the same as the filename defined as a RAF or chaining type specification on a File Description Specifications form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>38</td>
<td>E</td>
<td>COL 33-57 NOT BLK. BLK ASSUM</td>
<td>Columns 33 through 57 of the Extension Specifications form must be blank for all chaining type specifications.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>39</td>
<td>E</td>
<td>COL 7-10 NOT BLK. BLK ASSUM</td>
<td>Columns 7 through 10 of the Extension Specifications form must be blank for all RAF type specifications.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>40</td>
<td>E</td>
<td>COL 33-57 NOT BLK. BLK ASSUM</td>
<td>Columns 33 through 57 of the Extension Specifications form must be blank for all RAF type specifications.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>41</td>
<td>E</td>
<td>COL 7-10 NOT BLK. BLK ASSUM</td>
<td>Columns 7 through 10 of the Extension Specifications form must be blank for all table type specifications.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>* 42</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry is missing or not left-justified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 43</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry was not defined on a File Description Specifications form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 44</td>
<td>E</td>
<td>TO FILENAME COL 19-26 INVALID</td>
<td>To Filename entry is not defined as an output file on a File Description Specifications form.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>* 45</td>
<td>E</td>
<td>TBL NAME COL 27-32 OR 46-51 INVALID</td>
<td>Table Name entries missing or not left-justified. Columns 46-51 are required for alternating input formats only.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>* 46</td>
<td>E</td>
<td>COL 27-29 OR 46-48 NOT TAB</td>
<td>First 3 characters of table names must be TAB. Columns 46 through 48 are required for alternating input formats only.</td>
<td>TAB is assumed.</td>
</tr>
<tr>
<td>* 47</td>
<td>E</td>
<td>NO OF TBL ENTRIES COL 33-35 NOT NUMERIC</td>
<td>Number of table entries per record. These columns must contain a right-justified decimal number.</td>
<td>10 is assumed.</td>
</tr>
<tr>
<td>* 48</td>
<td>E</td>
<td>NO OF TBL ENTRIES COL 36-39 NOT NUMERIC</td>
<td>Number of table entries per table. These columns must contain a right-justified decimal number.</td>
<td>100 is assumed.</td>
</tr>
<tr>
<td>* 49</td>
<td>E</td>
<td>TBL ENTRY LNG COL 40-42 OR 52-54 NOT NUMERIC</td>
<td>Length of table entry. These columns must contain a right-justified decimal number. Columns 52 through 54 are required for alternating input formats only.</td>
<td>8 is assumed.</td>
</tr>
<tr>
<td>50</td>
<td>E</td>
<td>PACKED ENTRY COL 43 OR 55 INVALID. BLK ASSUM</td>
<td>Packed entry is not P or blank, or invalid for specified device.</td>
<td>Blank is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 3 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>E</td>
<td>NUM DEC POS COL 44 OR 56 INVALID</td>
<td>Decimal positions is not blank or a number.</td>
<td>Zero is assumed.</td>
</tr>
<tr>
<td>52</td>
<td>E</td>
<td>TBL SEQ COL 45 OR 57 INVALID. BLK ASSUM</td>
<td>Sequence entry is not A, D, or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>53</td>
<td>E</td>
<td>FORM TYPE COL 6 NOT VALID</td>
<td>The next specification should have been an E or I specification.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>56</td>
<td>F</td>
<td>COL 47-65, 67-70 MUST BE BLK FOR 1130 RPG</td>
<td>Specified columns are not used with 1130 RPG except for ISAM load files.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>57</td>
<td>F</td>
<td>ISAM NUMBER OF RECORDS INVALID</td>
<td>The number of records specified for an ISAM load (columns 47 through 52) is not numeric or left-justified.</td>
<td>One is assumed.</td>
</tr>
<tr>
<td>60</td>
<td>H</td>
<td>NO RPG CONTROL CARD. BLK ASSUM</td>
<td>Warning only. A compilation and listing will be performed for this run.</td>
<td>Blanks are assumed for all entries.</td>
</tr>
<tr>
<td>61</td>
<td>H</td>
<td>COL 11 INVALID. BLK ASSUM</td>
<td>Type of run. This entry should be B, D, or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>63</td>
<td>H</td>
<td>COL 17-20 INVALID. BLK ASSUM</td>
<td>Sterling entries are not blank, 0, 1, or 2, as required.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>64</td>
<td>H</td>
<td>COL 21 INVALID. BLK ASSUM</td>
<td>Inverted print option entry is not I or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>65</td>
<td>H</td>
<td>COL 26 INVALID. BLK ASSUM</td>
<td>Alternating collating sequence entry is not A or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>67</td>
<td>H</td>
<td>PROG NAME COL 75-80 INV. RPGOBJ ASSUM</td>
<td>Program Name entry on RPG Control Card is invalid.</td>
<td>RPGOBJ is assumed.</td>
</tr>
<tr>
<td>71</td>
<td>C</td>
<td>RSLT FLD COL 43-48 REQUIRED</td>
<td>Result Field name is required but is missing.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>72</td>
<td>C</td>
<td>RSLT FLD COL 43-48 MUST BE BLK. BLK ASSUM</td>
<td>Result Field must be blank for COMP, GOTO, EXIT, TAG, SETOF, SETON, CHAIN, BEGSR, ENDSR, EXSR, and EXCPT.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>73</td>
<td>C</td>
<td>FACT1, COL 18-27 INVALID</td>
<td>Factor 1 requires a fieldname, label, or literal with the specified operation.</td>
<td>Numeric literal 1 is assumed.</td>
</tr>
<tr>
<td>74</td>
<td>C</td>
<td>FACT2 COL 33-42 INVALID</td>
<td>Factor 2 requires a fieldname, label, or literal with the specified operation.</td>
<td>Numeric literal 1 is assumed.</td>
</tr>
<tr>
<td>75</td>
<td>C</td>
<td>RSLT IND COL 54-59 INVALID. 00 ASSUM</td>
<td>Resulting Indicator is not 01 through 99, H1 through H9, L1 through L9, OF, or OV.</td>
<td>00 is assumed for indicator in error.</td>
</tr>
<tr>
<td>76</td>
<td>C</td>
<td>FACT1 COL 18-27 MUST BE BLK. BLK ASSUM</td>
<td>Factor 1 entry must be blank for the operation being performed.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>77</td>
<td>C</td>
<td>FACT2 COL 33-42 MUST BE BLK. BLK ASSUM</td>
<td>Factor 2 entry must be blank for the operation being performed.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>Note</td>
<td>Spec type</td>
<td>Error message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 78</td>
<td>C</td>
<td>CTRL LEVEL COL 7-8 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 79</td>
<td>C</td>
<td>DETAIL CALC DOES NOT PRECEDE TOTAL CALC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 80</td>
<td>C</td>
<td>FACT1 COL 18-27 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 81</td>
<td>C</td>
<td>FACT2 COL 33-42 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 82</td>
<td>C</td>
<td>FACT1 COL 18-27 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 83</td>
<td>C</td>
<td>FACT2 COL 33-42 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 84</td>
<td>C</td>
<td>FACT1 COL 18-27 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 85</td>
<td>C</td>
<td>FACT2 COL 33-42 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 86</td>
<td>C</td>
<td>OPER CODE COL 28-32 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>C</td>
<td>CTRL LEV COL 7-8 INVALID. L0 ASSUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 89</td>
<td>C</td>
<td>RSLT FLD COL 43-48 REQUIRED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 94</td>
<td>C</td>
<td>RSLT FLD LNG COL 49-51 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 95</td>
<td>C</td>
<td>DEC POS COL 52 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>C</td>
<td>HLF ADJ COL 53 INVALID. H ASSUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 97</td>
<td>C</td>
<td>RSLT IND COL 54-59 REQUIRED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 98</td>
<td>C</td>
<td>IND COL 9-17 INVALID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*100</td>
<td>I</td>
<td>STERL COL 71-74 INVALID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cause of error**

- Control Level column 7 is not L or blank. Blank is assumed.
- A detail calculation, columns 7 and 8 blank, follows a total calculation, columns 7 and 8 L0 through L9 or LR.
- Factor 1 entry is not left-justified. Numeric literal 1 is assumed.
- Factor 2 entry is not left-justified. Numeric literal 1 is assumed.
- Factor 1 entry is an improperly stated literal or field name. Numeric literal 1 is assumed.
- Factor 2 entry is an improperly stated literal or field name. Numeric literal 1 is assumed.
- Factor 1 entry is a field name of more than 6 characters. First six characters are assumed.
- Factor 2 entry is a field name of more than 6 characters. First six characters are assumed.
- Operation code is missing or unrecognizable. MOVE operation code is assumed.
- Column 7 is L but column 8 is not 0 through 9 or R. L0 is assumed.
- Result Field entry is improperly defined. Specification is dropped.
- 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.
- Decimal Position entry is not blank or numeric. 0 is assumed.
- Half adjust entry is not H or blank. H is assumed.
- A resulting indicator is required for this operation. Internal indicator is assigned.
- Indicator entry improperly defined. Indicator is dropped.
- 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
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>I</td>
<td>FLD REC RELATION IND COL 63-64 INVALID</td>
<td>Field Record Relation Indicator</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>102</td>
<td>I</td>
<td>PLUS, MINUS, ZERO/BLK IND COL 65-70 INVALID</td>
<td>Indicator columns 65 through 70</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>103</td>
<td>I</td>
<td>OVER 60 REC TYPE SPEC'S</td>
<td>Input has more than 60 record</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>identification columns 6 through 42.</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>I</td>
<td>INPUT OR OUTPUT SPECS MISSING OR INVALID</td>
<td>Input or output specifications are</td>
<td>Job is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required.</td>
<td>terminated.</td>
</tr>
<tr>
<td>110</td>
<td>I</td>
<td>FORM TYPE COL 6 INVALID</td>
<td>Form Type is not I, C, or O and</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>column 7 does not contain an *.</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>I</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename entry is not defined.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>112</td>
<td>I</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename entry is not correctly defined on the Description form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>113</td>
<td>I</td>
<td>'AND' CD OUT OF SEQ</td>
<td>'AND' card is first card in deck, first specification after field name, or invalid file type.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>114</td>
<td>I</td>
<td>NO RECORD ID IN CARD BEFORE 'AND' CARD</td>
<td>Record ID entry columns 21 through 41 of Input Specifications form required in card before 'AND' card.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>115</td>
<td>I</td>
<td>'OR' CD OUT OF SEQ</td>
<td>'OR' card is first card in deck, first specification after field name, or invalid file type.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>116</td>
<td>I</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename entry not left-justified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>117</td>
<td>I</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename entry begins with a numeric character.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>118</td>
<td>I</td>
<td>FILE AND FLD NAME ARE BOTH ON SAME SPEC</td>
<td>File and field names cannot both appear on same specification.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>119</td>
<td>I</td>
<td>SEQ COL 15-16 BLK. AA ASSUM</td>
<td>Sequence entry must be 2 alpha or 2 numeric characters.</td>
<td>AA is assumed.</td>
</tr>
<tr>
<td>120</td>
<td>I</td>
<td>SEQ COL 15-16 ALPHA SEQ AFTER NUM SEQ</td>
<td>Alpha sequence entries must appear before numeric sequence entries.</td>
<td>Numeric sequence last used is assumed.</td>
</tr>
<tr>
<td>121</td>
<td>I</td>
<td>SEQ COL 15-16 IS INVALID</td>
<td>Ascending numeric sequence is required, or the first entries must begin with 01.</td>
<td>Numeric sequence last used is assumed.</td>
</tr>
<tr>
<td>122</td>
<td>I</td>
<td>NUMBER ENTRY COL 17 INVALID. N ASSUM</td>
<td>Sequence is numeric and the number entry column is not N or 1.</td>
<td>N is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 6 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>I</td>
<td>OPTION ENTRY COL 18 INVALID. 0 ASSUM</td>
<td>Sequence is numeric, and the option entry column is not 0 or blank.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>124</td>
<td>I</td>
<td>RECORD IDENTIFYING IND COL 19-20 INVALID. BLK ASSUM</td>
<td>Record Identifying Indicator entry is not 01 through 99.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>125</td>
<td>I</td>
<td>STKR SEL COL 42 INVALID. BLK ASSUM</td>
<td>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.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>*126</td>
<td>I</td>
<td>INVALID INPUT FILE</td>
<td>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.</td>
<td>No immediate action taken.</td>
</tr>
<tr>
<td>*127</td>
<td>I</td>
<td>POSITION ENTRY COL 21-24, 28-31, 35-38 INVALID</td>
<td>Position entry contains a non-numeric character.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>128</td>
<td>I</td>
<td>'NOT' ENTRY COL 25, 32 OR 39 INVALID. N ASSUM</td>
<td>'NOT' entry not N or blank.</td>
<td>N is assumed.</td>
</tr>
<tr>
<td>129</td>
<td>I</td>
<td>C/Z/D ENTRY COL 26, 33 OR 40 INVALID. C ASSUM</td>
<td>Combined/Zone/Digit entry is not C, Z, or D.</td>
<td>C is assumed.</td>
</tr>
<tr>
<td>130</td>
<td>I</td>
<td>FIELD NAME SPEC OUT OF SEQ</td>
<td>Field Name Type specification is first in deck, after invalid filename or invalid AND or OR specification.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*131</td>
<td>I</td>
<td>FLD NAME COL 53-58 INVALID</td>
<td>Field Name entry is not left-justified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*132</td>
<td>I</td>
<td>FLD NAME COL 53-58 INVALID</td>
<td>Field Name entry does not begin with an alphabetic character.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*133</td>
<td>I</td>
<td>FROM OR TO COL 44-51 INVALID</td>
<td>From or To columns are blank.</td>
<td>0001 is assumed.</td>
</tr>
<tr>
<td>*134</td>
<td>I</td>
<td>FROM OR TO COL 44-51 INVALID</td>
<td>From or To columns contain a non-numeric character.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>*135</td>
<td>I</td>
<td>TO COL 48-51 LESS THAN FROM COL 44-47</td>
<td>Defined field length less than 1.</td>
<td>1 is assumed.</td>
</tr>
<tr>
<td>*136</td>
<td>I</td>
<td>PACKED INPUT FLD INVALID</td>
<td>Packed input field length defined by From and To fields is greater than 8, or packed field is invalid for input device.</td>
<td>8 is assumed.</td>
</tr>
<tr>
<td>137</td>
<td>I</td>
<td>PACKED ENTRY COL 43 INVALID. P ASSUM</td>
<td>Packed entry is not P or blank.</td>
<td>P is assumed.</td>
</tr>
<tr>
<td>*138</td>
<td>I</td>
<td>DEC POS COL 52 INVALID</td>
<td>Decimal Positions are not numeric.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>*139</td>
<td>I</td>
<td>NUMERIC FLD GT 14</td>
<td>Numeric field length is greater than 14 characters.</td>
<td>Field length of 14 is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 7 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*140</td>
<td>I</td>
<td>CTRL LEV COL 59-60 INVALID</td>
<td>Column 59 not L.</td>
<td>L in column 59 is assumed.</td>
</tr>
<tr>
<td>*141</td>
<td>I</td>
<td>CTRL LEV COL 59-60 INVALID</td>
<td>Column 60 not numeric.</td>
<td>1 in column 60 is assumed.</td>
</tr>
<tr>
<td>*142</td>
<td>I</td>
<td>MATCH OR CHAIN ENTRY COL 61-62 INVALID</td>
<td>Column 61 not M or C.</td>
<td>M in column 61 is assumed.</td>
</tr>
<tr>
<td>*143</td>
<td>I</td>
<td>MATCH OR CHAIN ENTRY COL 61-62 INVALID</td>
<td>Column 62 not numeric.</td>
<td>1 in column 62 is assumed.</td>
</tr>
<tr>
<td>*144</td>
<td>I</td>
<td>MATCH ENTRY COL 61-62 NOT M1-M9</td>
<td>Match entry is invalid.</td>
<td>M9 is assumed.</td>
</tr>
<tr>
<td>145</td>
<td>I</td>
<td>RSLT IND COL 65-68 SPECIFIED FOR NON-NUM FLD. INDIGN</td>
<td>Plus and minus indicators cannot be used with an alphameric field.</td>
<td>Indicator is ignored.</td>
</tr>
<tr>
<td>*146</td>
<td></td>
<td>ALPHA FLD GT 256</td>
<td>Alphameric field length is more than 256 characters.</td>
<td>Field length of 256 is assumed.</td>
</tr>
<tr>
<td>*147</td>
<td>I</td>
<td>STERL FLD INVALID</td>
<td>Sterling field has more than 3 decimal positions specified.</td>
<td>3 is assumed.</td>
</tr>
<tr>
<td>*148</td>
<td>I</td>
<td>STERL FLD INVALID</td>
<td>Sterling field has no decimal positions specified.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>149</td>
<td>I</td>
<td>REC ID SPEC OUT OF SEQ OR NO FIELDS FOR GIVEN REC</td>
<td>Warning only. Record ID specification is out of order, or no fields are indicated for a given record.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*150</td>
<td>I</td>
<td>PACKED FLD MUST BE NUMERIC</td>
<td>Decimal Position entry column 52 is blank.</td>
<td>0 is assumed.</td>
</tr>
<tr>
<td>*151</td>
<td>I</td>
<td>FROM TO OR RECORD ID ZERO</td>
<td>From, To, or Position entries are zero.</td>
<td>0001 is assumed.</td>
</tr>
<tr>
<td>*152</td>
<td>I</td>
<td>FLD REC POS BLK, BUT TEST CHAR PRESENT</td>
<td>Position entry 27, 34, or 41 contains a valid test character.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*155</td>
<td>F</td>
<td>KEY SIZE EXCEEDS REC LNG</td>
<td>Key length columns 29 and 30 (ISAM file) is greater than record length.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*158</td>
<td>F</td>
<td>KEY LNG EXCEEDS 50</td>
<td>Key length columns 29 and 30 (ISAM file) is more than 50 characters.</td>
<td>50 is assumed.</td>
</tr>
<tr>
<td>*159</td>
<td></td>
<td>FLD NAME BEGINS WITH 'TAB' BUT IS NOT TBL NAME</td>
<td>Field name beginning with TAB is not a table name. Tables are defined on Extension Specifications form columns 27 through 32.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*160</td>
<td></td>
<td>FORM TYPE COL 6 INVALID</td>
<td>Next Form Type entry should have been 0.</td>
<td>Specification is dropped.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 8 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>161</td>
<td>O</td>
<td>INVALID OUTPUT SPEC</td>
<td>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.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>162</td>
<td>O</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename entry is missing, improperly defined, or undefined.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>163</td>
<td>O</td>
<td>H/D/T/E ENTRY COL 15 OUT OF SEQ</td>
<td>Output lines must be sequenced as follows: H/D/T/E.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>164</td>
<td>O</td>
<td>LINE TYPE COL 15 INVALID</td>
<td>Line Type entry must be H, D, T, or E.</td>
<td>H is assumed.</td>
</tr>
<tr>
<td>165</td>
<td>O</td>
<td>IND COL 23-31 MISSING ON 'OR' SPEC. 00 ASSUM</td>
<td>'OR' specification requires conditioning indicators in columns 23 through 31.</td>
<td>Indicator 00 is assumed.</td>
</tr>
<tr>
<td>166</td>
<td>O</td>
<td>IND COL 23-31 MISSING ON 'AND' SPEC. SPEC DROPPED</td>
<td>'AND' specification requires conditioning indicators in columns 23 through 31.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>167</td>
<td>O</td>
<td>COL 32-70 MUST BE BLK ON LINE SPEC. BLK ASSUME</td>
<td>File ID and CONTROL specification requires columns 32 through 70 blank.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>168</td>
<td>O</td>
<td>FIELD NAME COL 32-37 INVALID. SPEC DROPPED</td>
<td>Field Name entry is not left-justified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>169</td>
<td>O</td>
<td>IND COL 23-25, 26-28, OR 29-31, INVALID OR OF OR OV NOT IN 33-34 OF FDS. SPEC DROPPED</td>
<td>Output Indicator entry is incorrect.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>170</td>
<td>O</td>
<td>CARD OUT OF ORDER</td>
<td>'OR' or 'AND' card is out of sequence.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>171</td>
<td>O</td>
<td>CARD OUT OF ORDER</td>
<td>Field type specification with column 15 blank is not preceded by a valid line type specification.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>172</td>
<td>O</td>
<td>OUTPUT FLD SPEC WITH ENTRIES IN COL 7-22</td>
<td>Output field specification requires columns 7 through 22 blank.</td>
<td>Entries in columns 7 through 22 are ignored.</td>
</tr>
<tr>
<td>173</td>
<td>O</td>
<td>LEAD OR CLOSE QUOTE COL 45-70 MISSING. NO EDIT</td>
<td>Edit word must be enclosed by apostrophes.</td>
<td>No editing is performed.</td>
</tr>
<tr>
<td>174</td>
<td>O</td>
<td>EDIT CODE COL 38 INVALID OR USED WITH ALPHA FLD. BLK ASSUM</td>
<td>Edit code used is invalid or an edit code has been specified with an alpha field.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>175</td>
<td>O</td>
<td>BLANK AFTER COL 39 INVALID. BLK ASSUM</td>
<td>Blank after entry not B or blank.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>176</td>
<td>O</td>
<td>PACKED ENTRY COL 44 INVALID. BLK ASSUM</td>
<td>Packed entry not P or blank, field is not numeric, or packed field is invalid.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>177</td>
<td>O</td>
<td>COL 17-22 NON-BLK ON 'AND' SPEC. BLK ASSUM</td>
<td>Columns 17 through 22 are not blank on 'AND' specification.</td>
<td>Blanks are assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 9 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td>O</td>
<td>END POS COL 40-43 INVALID. SPEC DROPPED</td>
<td>End position in Output Record entry is blank, alphabetic, or is incompatible with constant or edit word.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>179</td>
<td>O</td>
<td>LEAD OR CLOSE QUOTE COL 45-70 MISSING. SPEC DROPPED</td>
<td>Constant must be enclosed by apostrophes.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*180</td>
<td>C</td>
<td>FLD NAMED COL 43-48 GT 14</td>
<td>On an arithmetic operation, the field named in columns 43 through 48 is longer than 14 characters.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*181</td>
<td>C</td>
<td>MOVE ZONE OPER INVALID</td>
<td>Incorrect alphabetic or numeric fields have been specified for this Move Zone operation. Only the low zone of a numeric field can be referred to.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*183</td>
<td>C</td>
<td>FIELD NAME UNDEF</td>
<td>The field name in Factor 1, Factor 2, or Result Field is undefined.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>184</td>
<td></td>
<td>FLD NAME UNREF</td>
<td>Warning only. Field Name entry is unreferenced field or table name.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*185</td>
<td></td>
<td>FLD NAME MULT-DEF</td>
<td>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.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*186</td>
<td>C</td>
<td>ARITH OPER SPECIFIED WITH ALPHA FLD</td>
<td>Arithmetic operation specified in operation columns 28 through 32 with an alphameric field specified in Factor 1, Factor 2, or Result field.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*187</td>
<td>C</td>
<td>COMP OPER SPECIFIED WITH ALPHA AND NUM FLD</td>
<td>Alphameric and numeric field being compared. Compare operations are valid only between like fields.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>188</td>
<td>C</td>
<td>RSLT FLD LNG COL 49-51 MAY NOT BE LARGE ENOUGH</td>
<td>Warning only. The Result Field may not be long enough to contain the true result.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*189</td>
<td>C</td>
<td>FACT2 OR RSLT FLD NOT TBL NAME</td>
<td>LOKUP requires table names in Factor 2 columns 33 through 42, and Result Field columns 43 through 48 (if specified).</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*190</td>
<td>C</td>
<td>EXSR OPER CALLS ITSELF</td>
<td>Name in Factor 2 is the name of the subroutine of which the EXSR operation is a part (a subroutine may not call itself).</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*191</td>
<td>C</td>
<td>TESTZ OPER INVALID</td>
<td>Result Field entry columns 43 through 48 is numeric. TESTZ tests for a high-order zone punch of an alpha field.</td>
<td>Specification is dropped.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 10 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>C</td>
<td>GOTO AND TAG OPERS ARE NOT IN SAME CALC SECTION</td>
<td>Label of the TAG operation and the corresponding GOTO are not in Detail or Total calculations.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>193</td>
<td>C</td>
<td>HLF ADJ COL 53 IS INCOMPATIBLE. BLK ASSUM</td>
<td>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.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>*194</td>
<td>C</td>
<td>LOKUP OPER INVALID DUE TO UNEQUAL LNGS</td>
<td>Length of Factor 1 columns 18 through 27 and Factor 2 columns 33 through 42 are not equal.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*196</td>
<td>C</td>
<td>MVR OPER NOT PRECEDED BY DIV</td>
<td>There is no remainder to move.</td>
<td>MVR operation is ignored.</td>
</tr>
<tr>
<td>*197</td>
<td>C</td>
<td>MVR OPER PRECEDED BY DIV WITH HLF ADJ</td>
<td>Half-adjust effectively removes any remainder.</td>
<td>MVR operation is ignored.</td>
</tr>
<tr>
<td>*198</td>
<td>C</td>
<td>LOKUP OPER SPECIFIED WITH ALPHA AND NUM FLD</td>
<td>Factor 1 columns 18 through 27 and Factor 2 columns 33 through 42 must both be alpha or numeric.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*199</td>
<td>C</td>
<td>HIGH AND LOW RSLT IND SPEC FOR LOKUP OPER</td>
<td>High and Low Resulting indicators are both specified for LOKUP operation.</td>
<td>Low indicator is ignored.</td>
</tr>
<tr>
<td>*200</td>
<td>F</td>
<td>NO PRIMARY FILE SPECIFIED</td>
<td>No P in column 16 of File Description Specifications form. One file must be defined as primary.</td>
<td>Job is terminated.</td>
</tr>
<tr>
<td>*201</td>
<td></td>
<td>FORM TYPE COL 6 INVALID</td>
<td>Next Form Type entry should have been F, E or I.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*202</td>
<td>F</td>
<td>FILENAME COL 7-14 INVALID</td>
<td>Filename incorrectly specified.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*203</td>
<td>F</td>
<td>MORE THAN 10 FILENAMES SPEC</td>
<td>More than the maximum of 10 files are specified.</td>
<td>Only the first 10 are processed.</td>
</tr>
<tr>
<td>204</td>
<td>F</td>
<td>UNREF FILENAME</td>
<td>Warning only. A file defined on the File Description Specifications form has not been used in the program.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>205</td>
<td>F</td>
<td>FILE TYPE COL 15 INVALID WITH READ01</td>
<td>Device entry READ01 requires an I in File Type column 15.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*206</td>
<td>F</td>
<td>DEVICE COL 40-46 INVALID</td>
<td>Device name is unrecognizable.</td>
<td>Job is terminated.</td>
</tr>
<tr>
<td>207</td>
<td>F</td>
<td>FILENAME COL 7-14 MULT-REF</td>
<td>The filename is specified on the Input or Output Format Specifications form more than once.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*208</td>
<td>F</td>
<td>FILENAME COL 7-14 MULT-DEF</td>
<td>The same filename is defined on two File Description Specifications forms.</td>
<td>Second specification is dropped.</td>
</tr>
<tr>
<td>Note</td>
<td>Spec type</td>
<td>Error message</td>
<td>Cause of error</td>
<td>System action</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>---------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>*210</td>
<td></td>
<td>NO IND OR ONLY PREDEF IND SPEC FOR INPUT REC</td>
<td>At least one indicator is required on input specifications.</td>
<td>Job is terminated.</td>
</tr>
<tr>
<td>*212</td>
<td></td>
<td>UNDEFINED RESULT IND</td>
<td>Result indicator used but not defined.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>213</td>
<td></td>
<td>UNREFERENCED IND</td>
<td>Warning only. Indicator specified but not used.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>215</td>
<td>F</td>
<td>FILE DESCR SPEC WITH E COL 39 NOT REF ON EXT SPEC</td>
<td>File description specification with E in column 39 is not used on an extension specification.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>219</td>
<td>O</td>
<td>FLD NAME COL 32-37 UNDEFINED. SPEC DROPPED</td>
<td>Name must be defined on Input or Calculation Specifications form.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*221</td>
<td>I</td>
<td>MATCH FLD LNGS INCOMPATIBLE</td>
<td>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.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*222</td>
<td>E</td>
<td>TBL NAME MULT-DEF</td>
<td>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.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*223</td>
<td>I</td>
<td>FLD IS OUTSIDE THE REC</td>
<td>The input field specified in columns 44-51 is outside the physical record specified in columns 24-27 of the file description specification.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*224</td>
<td>I</td>
<td>SPLIT CHAIN FLDS IMPROPER</td>
<td>Split chain fields are improperly specified.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*225</td>
<td>I</td>
<td>SPLIT CTRL FLDS IMPROPER</td>
<td>Split control fields are improperly specified.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*226</td>
<td>I</td>
<td>SPLIT MATCH FLDS INVALID</td>
<td>Split matching fields are not allowed.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*227</td>
<td>I</td>
<td>MATCH FLD LNGS INCOMPATIBLE</td>
<td>All match fields of the same level must be the same length on all record types.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*228</td>
<td>I</td>
<td>CTRL FLD LNG INCOMPATIBLE</td>
<td>The control field on a given control level must be the same length for all record types.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*229</td>
<td>I</td>
<td>CHAIN FLD LNG INCOMPATIBLE</td>
<td>All fields using the same chaining indicator must be the same length on all record types.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*230</td>
<td>I</td>
<td>CTRL FLD LNG GT 247</td>
<td>The sum of the control fields on all levels used on a record type cannot exceed 247 characters.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*231</td>
<td>I</td>
<td>FLD AREA GT REC SIZE</td>
<td>Input field area size exceeds input record length.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>232</td>
<td>O</td>
<td>PRINTER FILE BLK COL 17-22. SPACE 1 AFTER ASSUM</td>
<td>Entry required in columns 17 through 22 for printer carriage control.</td>
<td>Single space after is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 12 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>233</td>
<td>O</td>
<td>STKR SEL COL 16 INVALID. BLK ASSUM.</td>
<td>Stacker select invalid for output device, or entry is incorrect (not 1 or 2).</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>234</td>
<td>O</td>
<td>SPACE BEFORE, COL 17, INVALID. 1 ASSUM</td>
<td>There is an entry in column 17, but it is not 0, 1, 2, or 3.</td>
<td>Single space before is assumed.</td>
</tr>
<tr>
<td>235</td>
<td>O</td>
<td>SPACE AFTER, COL 18, INVALID. 1 ASSUM</td>
<td>There is an entry in column 18, but it is not 0, 1, 2, or 3.</td>
<td>Single space after is assumed.</td>
</tr>
<tr>
<td>236</td>
<td>O</td>
<td>SKIP BEFORE, COL 19-20 INVALID. BLK ASSUM</td>
<td>There is an entry in 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.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>237</td>
<td>O</td>
<td>SKIP AFTER, COL 21-22, INVALID. BLK ASSUM</td>
<td>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.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>238</td>
<td>O</td>
<td>PACKED FLD COL 44 NOT NUM. BLK ASSUM</td>
<td>Output field is alpha.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>239</td>
<td>O</td>
<td>EDIT CODE COL 38 SPECIFIED ON ALPHA FLD. BLK ASSUM</td>
<td>Alpha fields cannot be edited with an edit code.</td>
<td>Blank is assumed.</td>
</tr>
<tr>
<td>240</td>
<td>O</td>
<td>STERL SPECIFIED ON NON-NUM FLD, NO STERL ASSUM</td>
<td>Sterling option columns 71 through 74 requested for alpha field.</td>
<td>No sterling is assumed.</td>
</tr>
<tr>
<td>*241</td>
<td>O</td>
<td>EDIT WD TOO SMALL</td>
<td>Edit word is too small for field.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*242</td>
<td>O</td>
<td>EDIT FLD NOT NUM</td>
<td>Alpha fields not edited.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*243</td>
<td>O</td>
<td>DOLLAR SIGN INVALID</td>
<td>Both fixed and floating dollar sign have been specified.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*244</td>
<td>O</td>
<td>BOTH CR AND — USED</td>
<td>Both CR and minus are used for credit.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*245</td>
<td>O</td>
<td>OUTPUT SPEC INVALID</td>
<td>Output specifications are missing or are invalid for this program.</td>
<td>Job is terminated.</td>
</tr>
<tr>
<td>*246</td>
<td>O</td>
<td>PAGE FLD IS DEF AS ALPHA</td>
<td>PAGE defined on Input Specifications form with no decimal position in column 52.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*247</td>
<td>O</td>
<td>FLD LNG GT END POS COL 40-43</td>
<td>Output field length is greater than the indicated End Position in Output Record columns 40 through 43.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*248</td>
<td>F</td>
<td>INDEX SEQ FILE ADDITION COL 68 INVALID</td>
<td>Column 68 must contain an A for ISAM ADD functions.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*250</td>
<td>F</td>
<td>INDEX SEQ KEY LNG COL 29-30 INVALID</td>
<td>Key Length entry columns 29 and 30 is not numeric.</td>
<td>8 is assumed.</td>
</tr>
</tbody>
</table>

Figure A-13 (Part 13 of 14). RPG compiler error notes
<table>
<thead>
<tr>
<th>Note</th>
<th>Spec type</th>
<th>Error message</th>
<th>Cause of error</th>
<th>System action</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td>F</td>
<td>INDEX SEQ KEY START POS COL 35-38 INVALID. 1 ASSUM</td>
<td>Key field must start in position one of record.</td>
<td>0001 is assumed.</td>
</tr>
<tr>
<td>*252</td>
<td>O</td>
<td>END POS GT RCD LNG</td>
<td>Output field length is greater than Record length (columns 40 through 43).</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*254</td>
<td>O</td>
<td>'ADD' COL 16-18 MUST BE SPEC</td>
<td>'ADD' must be specified if records are added to an ISAM file.</td>
<td>Specification is dropped.</td>
</tr>
<tr>
<td>*255</td>
<td>C</td>
<td>FACT2 COL 33-42 INVALID</td>
<td>Entry in Factor 2 must be filename described as a chained file on the File Description Specifications forms.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>256</td>
<td>C</td>
<td>CTRL LEV COL 7-8 INVALID. SR ASSUM</td>
<td>Closed subroutine must follow total calculations.</td>
<td>SR is assumed.</td>
</tr>
<tr>
<td>*257</td>
<td>C</td>
<td>ERROR IN SEQ OF ENDSR-BEGSR</td>
<td>BEGSR operation must come first.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>*258</td>
<td>C</td>
<td>BEGSR OR EXSR FACTORS INVALID</td>
<td>BEGSR—Subroutine name must appear in Factor 1 columns 18 through 27. EXSR—Subroutine name must appear in Factor 2 columns 33 through 42.</td>
<td>No immediate action is taken.</td>
</tr>
<tr>
<td>259</td>
<td>C</td>
<td>COL 49-59 MUST BE BLK WITH EXSR OR EXCPT. BLK ASSUM</td>
<td>EXSR or EXCPT operation codes require columns 49 through 59 blank.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>260</td>
<td>C</td>
<td>COL 9-17 MUST BE BLK WITH BEGSR. BLK ASSUM</td>
<td>BEGSR operation code requires columns 9 through 17 blank.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>262</td>
<td>C</td>
<td>COL 49-53 MUST BE BLK WITH CHAIN. BLK ASSUM</td>
<td>CHAIN operation code requires columns 49 through 53 blank.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>263</td>
<td>C</td>
<td>IND COL 56-57 MUST BE THE SAME AS IND COL 54-55 HIGH ASSUM</td>
<td>The same indicator must be specified as high and low indicator.</td>
<td>High indicator is assumed for high and low.</td>
</tr>
<tr>
<td>264</td>
<td>O</td>
<td>CHAIN SPECIFIED WITH IND IN COL 58-59. BLK ASSUM</td>
<td>Equal indicator cannot be specified on chaining operation.</td>
<td>Blanks are assumed.</td>
</tr>
<tr>
<td>*265</td>
<td>C</td>
<td>PAGE FLD INVALID</td>
<td>Page field must be numeric. Field length must be 4 with zero decimal positions.</td>
<td>Field length of 4 and zero decimal positions are assumed.</td>
</tr>
<tr>
<td>270</td>
<td>O</td>
<td>SKIP INVALID FOR CONSOLE PRINTER. BLK ASSUM</td>
<td>Console printer has no provisions for forms skipping. Columns 19 through 22 must be blank.</td>
<td>Blanks are assumed.</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R00 LOCALS/SOCALS OVERFLOW WORK STORAGE</td>
<td>Enough working storage is not available to accommodate the LOCAL and/or SOCAL overlays required by the core load.</td>
<td>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.</td>
</tr>
<tr>
<td>R01 ORIGIN BELOW 1ST WORD OF MAINLINE</td>
<td>The core load builder has been instructed to load a word into an address lower than the first word of the mainline program.</td>
<td>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.</td>
</tr>
<tr>
<td>R02 DEFINE-FILE(S) OVERFLOW WORK STORAGE</td>
<td>Enough working storage is not available to accommodate any records of the defined file(s).</td>
<td>See the options for error message R00.</td>
</tr>
<tr>
<td>R03 NO DSF PROGRAM IN WORKING STORAGE</td>
<td>Working storage does not contain a program when the core load builder is called.</td>
<td>Load the desired program into working storage.</td>
</tr>
<tr>
<td>R05 INVALID LOADING ADDR FOR ILS02</td>
<td>ILS02 has been loaded into low COMMON. If error message R48 is also printed, see R48. If ILS02 (or ILSX2) can be relocated, this is a warning message only.</td>
<td>Make the mainline program longer so that ILS02 can be loaded in a higher address. If the mainline program is a system program, restore the system ILSs and store the program in core image format.</td>
</tr>
<tr>
<td>R06 FILE(S) TRUNCATED (SEE FILE MAP)</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>R07 TOO MANY ENTRIES IN LOAD</td>
<td>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.</td>
<td>Divide the core load into 2 or more links.</td>
</tr>
<tr>
<td>R08 CORE LOAD EXCEEDS 32K</td>
<td>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.</td>
<td>Divide the core load into 2 or more links.</td>
</tr>
<tr>
<td>R09 LIBF TV REQUIRES 84 OR MORE ENTRIES</td>
<td>At least 82 different entry points are referenced in the core load by LIBF statements.</td>
<td></td>
</tr>
<tr>
<td>R16 XXXXX IS NOT IN LET OR FLET</td>
<td>The program name or data file name printed cannot be found in LET or FLET.</td>
<td>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).</td>
</tr>
</tbody>
</table>

Figure A-14 (Part 1 of 4). Core load builder error messages.
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R17 XXXXX CANNOT BE A LOCAL/NOCAL</td>
<td>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.</td>
<td>Choose another area for the storage of this file.</td>
</tr>
<tr>
<td>R18 XXXXX LOADING HAS BEEN TERMINATED</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>R19 XXXXX IS NOT A DATA FILE</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>R20 XXXXX COMMON EXCEEDS THAT OF ML</td>
<td>The length of COMMON for the subroutine named in this message is longer than that of the mainline program.</td>
<td>Define more COMMON for the mainline program.</td>
</tr>
<tr>
<td>R21 XXXXX PRECISION DIFFERENT FROM ML</td>
<td>The precision, both real and integer, for the subroutine named in this message is incompatible with that of the mainline program.</td>
<td>Make *EXTENDED PRECISION or *ONE WORD INTEGERS the same in the named subroutine and the mainline program.</td>
</tr>
<tr>
<td>R22 XXXXX AND ANOTHER VERSION RE’FER’ENCED</td>
<td>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.)</td>
<td>Change the references so that the core load uses only one version of any given I/O subroutine.</td>
</tr>
<tr>
<td>R23 XXXXX SHOULD BE IN THE FIXED AREA</td>
<td>The area named in this message is in the user area.</td>
<td>References in DEFINE FILE and DSA statements for *STORECI functions must be to the fixed area.</td>
</tr>
<tr>
<td>R39 XXXX is not CURRENTLY MOUNTED</td>
<td>XXXX is a cartridge ID specified on an *FILES card, but not the ID of a cartridge currently mounted.</td>
<td>Change *FILES card to reference an available cartridge or mount the requested cartridge and restart the job.</td>
</tr>
<tr>
<td>R40 XXXX (HEX) = ADDITIONAL CORE REQUIRED</td>
<td>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.</td>
<td>For the second case, create more links or LOCALS.</td>
</tr>
<tr>
<td>R41 XXXX (HEX) WORDS UNUSED BY CORE LOAD</td>
<td>Not an error. /XXXX is the number of words of core storage not used by this core load.</td>
<td></td>
</tr>
<tr>
<td>R42 XXXX (HEX) IS THE EXECUTION ADDR</td>
<td>Not an error. This message follows every successful conversion from DSF to DCI when a core map is requested.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-14 (Part 2 of 4). Core load builder error messages
<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R43 XXXX (HEX) = ARITH/FUNC SOCAL WD CNT</td>
<td>Not an error. Special overlays (SOCALs) are required. /XXXX is the length of the arithmetic/function overlay (see &quot;Incorporating Subroutines&quot; in Chapter 3).</td>
<td></td>
</tr>
<tr>
<td>R44 XXXX (HEX) = FI/O, I/O SOCAL WD CNT</td>
<td>Not an error. Special overlays (SOCALs) are required. /XXXX is the length of the FORTRAN I/O, I/O, and conversion subroutine overlay (see &quot;Incorporating Subroutines&quot; in Chapter 3).</td>
<td></td>
</tr>
<tr>
<td>R45 XXXX (HEX) = DISK FI/O SOCAL WD CNT</td>
<td>Not an error. Special overlays (SOCALs) are required. /XXXX is the length of the disk FORTRAN I/O overlay, including the 320 word buffer.</td>
<td></td>
</tr>
<tr>
<td>R46 XXXX (HEX) = AN ILLEGAL ML ADDR</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>R47 XXXX (HEX) TOO MANY WDS IN COMMON</td>
<td>The length of COMMON specified in the mainline program plus the length of the core load exceeds core storage by /XXXX words. Defined COMMON for this coreload overlaps low COMMON by /XXXX words.</td>
<td>Do one of the following: 1. Decrease the size of COMMON, 2. Request a shorter version of disk I/O.</td>
</tr>
<tr>
<td>R48 XXXX (HEX)</td>
<td>This message is printed with message R05.</td>
<td></td>
</tr>
<tr>
<td>R64 XXXXXXX IS BOTH A LIBF AND A CALL</td>
<td>The subroutine named in this message is either improperly referenced; that is, a CALL instead of a LIBF or vice versa, or has been referenced in both CALL and LIBF statements.</td>
<td></td>
</tr>
<tr>
<td>R66 XXXXXXX HAS MORE THAN 14 ENTRY POINTS</td>
<td>This message usually means that the subroutine has been destroyed since a subroutine is not stored if it contains more than 14 entry points.</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-14 (Part 3 of 4). Core load builder error messages
### CLB Error Messages

#### Auxiliary Supervisor error messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R66 XXXXX HAS AN INVALID TYPE</td>
<td>One of the following:</td>
<td>Change the file numbers on the *FILES control record to point to unique data files.</td>
</tr>
<tr>
<td></td>
<td>1. The subroutine named in this message:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Has been designated on an XEQ control record and is not a mainline program, or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 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</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Has been stored with an appropriate subtype.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>R67 XXXX HAS AN INVALID GSB ADDRESS</td>
<td>The subroutine named has a Graphic Short Branch order address that is larger than 8191 after relocation.</td>
<td></td>
</tr>
<tr>
<td>R68 XXXXX FILE NUMBER PREVIOUSLY USED</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

---

**AUXILIARY SUPERVISOR ERROR MESSAGES**

The auxiliary supervisor does not print informational messages. Figure A-15 lists the auxiliary supervisor error messages.

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>S00 INVALID FUNCTION CODE</td>
<td>The auxiliary supervisor received an illegal parameter.</td>
</tr>
<tr>
<td>S01 XXXXX IS NOT IN LET/FLET</td>
<td>The core image loader is unable to find the name specified in this message in LET or FLET.</td>
</tr>
<tr>
<td>S02 XXXXX IS A DATA FILE</td>
<td>The specified name cannot be executed since it is a data file, not a program.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Error message</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARTRIDGE XXXX INVALID</td>
<td>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.</td>
</tr>
<tr>
<td>... LOGICAL 0 ID</td>
<td></td>
</tr>
<tr>
<td>CARTRIDGE XXXX NEW LABEL IS INVALID</td>
<td>The new label XXXX is outside the range /0001 through /7FFF.</td>
</tr>
<tr>
<td>CARTRIDGE XXXX IS NOT AVAILABLE</td>
<td>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.</td>
</tr>
<tr>
<td>CARTRIDGE XXXX IS DEFECTIVE</td>
<td>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).</td>
</tr>
</tbody>
</table>
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 OWXX
MODIF COMPLETED OYZZ
```

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.
<table>
<thead>
<tr>
<th>Error number</th>
<th>Description</th>
<th>Operator's switch option</th>
<th>Operator recovery procedure</th>
<th>Remarks</th>
<th>First hexadecimal number printed</th>
<th>Second hexadecimal number printed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Invalid patch control record (*MON or *SUB)</td>
<td>No switches on</td>
<td>Correct error and reread from corrected patch control record. (If the error has occurred on the first patch control record, restart the modification.)</td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification.</td>
</tr>
<tr>
<td>2</td>
<td>Checksum error on binary patch data record</td>
<td>No switches on</td>
<td>Rechecksum and reread from preceding patch control record. (If the error has occurred on the first patch control record, restart the modification.)</td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification. Amount of checksum difference Number of binary records read after patch header (including record in error)</td>
</tr>
<tr>
<td>3</td>
<td>Invalid hex data record</td>
<td>No switches on</td>
<td>Correct error and reread from preceding patch control record.</td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification.</td>
</tr>
<tr>
<td>4</td>
<td>Modification level error in system modification update</td>
<td>No switches on</td>
<td>Correct error and reread from corrected patch control record.</td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification.</td>
</tr>
<tr>
<td>Error number</td>
<td>Description</td>
<td>Operator’s switch option</td>
<td>Operator recovery procedure</td>
<td>Remarks</td>
<td>First hexadecimal number printed</td>
<td>Second hexadecimal number printed</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>New modification level lower than current level in system modification update</td>
<td>No switches on</td>
<td>Correct error and reread from corrected patch control record.</td>
<td></td>
<td>Present version and modification level (from DCOM on disk)</td>
<td>Level of version and modification (from patch control record)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch 15 on</td>
<td>Press START to continue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Monitor control record or // DEND card read before required number of patches read</td>
<td>No switches on</td>
<td>Press START to continue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DCOM configuration indicators do not agree with SLET or required system I/O routine missing</td>
<td>No switches on</td>
<td>Press START to call EXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Switch 0 on</td>
<td>Press START to call EXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>DUP control record errors (DELETES or STORES)</td>
<td>No switches on</td>
<td>Press START to continue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SLET ID not found</td>
<td>No switches on</td>
<td>Press START to continue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Patch exceeds space allotted on disk for this phase</td>
<td>No switches on</td>
<td>Press START to continue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>// DEND card not found (patches completed but version and modification level in DCOM not updated)</td>
<td>No switches on</td>
<td>Press START to call EXIT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error number</td>
<td>Description</td>
<td>Operator's switch option</td>
<td>Operator recovery procedure</td>
<td>Remarks</td>
<td>First hexadecimal number printed</td>
<td>Second hexadecimal number printed</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>--------------------------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>C</td>
<td>Modification level error in general temporary fix</td>
<td>No switches on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification. Preceding patches in this MODIF JOB have been installed.</td>
<td>Present version and modification level (from DCOM on disk)</td>
<td>Version and modification level (from patch control record)</td>
</tr>
<tr>
<td>D</td>
<td>Modification level error in restricted temporary fix</td>
<td>No switches on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification. Preceding patches in this MODIF JOB have been installed.</td>
<td>Present version and modification level (from DCOM on disk)</td>
<td>Version and modification level (from patch control record)</td>
</tr>
<tr>
<td>E</td>
<td>System modification update mixed with temporary fixes</td>
<td>No switches on</td>
<td>Press START to call EXIT</td>
<td>This terminates modification. Preceding patches in this MODIF JOB have been installed.</td>
<td>Present version and modification level (from DCOM on disk)</td>
<td>Version and modification level (from patch control record)</td>
</tr>
</tbody>
</table>
**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.
<table>
<thead>
<tr>
<th>Error number</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>MODSF cannot be run in a temporary job mode.</td>
</tr>
<tr>
<td>02</td>
<td>MODSF cannot be run with DUP suppressed.</td>
</tr>
<tr>
<td>03</td>
<td>First card is not *PRO.</td>
</tr>
<tr>
<td>04</td>
<td>Last card was encountered before *END card.</td>
</tr>
<tr>
<td>05</td>
<td>Monitor control record was encountered.</td>
</tr>
<tr>
<td>06</td>
<td>*Card neither *PRO nor *END.</td>
</tr>
<tr>
<td>07</td>
<td>Column which must be blank was not blank in patch control record.</td>
</tr>
<tr>
<td>08</td>
<td>Version/modification (columns 6 through 8) invalidly specified or omitted.</td>
</tr>
<tr>
<td>09</td>
<td>Version/modification (columns 6 through 8) does not match system cartridge.</td>
</tr>
<tr>
<td>10</td>
<td>Program name (columns 10 through 14) is invalid or omitted.</td>
</tr>
<tr>
<td>11</td>
<td>Number of patch data records (columns 18 through 19) is not a valid positive hexadecimal value.</td>
</tr>
<tr>
<td>12</td>
<td>Cartridge ID (columns 23 through 26) is not validly specified.</td>
</tr>
<tr>
<td>13</td>
<td>Cartridge specified (columns 23 through 26) is not online.</td>
</tr>
<tr>
<td>14</td>
<td>Program specified (columns 10 through 14) cannot be found on requested cartridge.</td>
</tr>
<tr>
<td>15</td>
<td>Name specified in columns 10 through 14 is a secondary entry point.</td>
</tr>
<tr>
<td>16</td>
<td>Name specified in columns 10 through 14 is a core-image program.</td>
</tr>
<tr>
<td>17</td>
<td>Name specified in columns 10 through 14 is a data file.</td>
</tr>
<tr>
<td>18</td>
<td>Addressing mode (column 21) is neither D nor P.</td>
</tr>
<tr>
<td>19</td>
<td>Invalid address is specified for verification (columns 28 through 31, 38 through 41, 48 through 51, 58 through 61).</td>
</tr>
<tr>
<td>20</td>
<td>Invalid value is specified for verification (columns 33 through 36, 43 through 46, 53 through 56, 63 through 66).</td>
</tr>
<tr>
<td>21</td>
<td>During verification, a nonmatch was detected.</td>
</tr>
<tr>
<td>22</td>
<td>Number of patch data records does not match number specified.</td>
</tr>
<tr>
<td>23</td>
<td>Patch address is an invalid hexadecimal value.</td>
</tr>
<tr>
<td>24</td>
<td>Column in patch data record which must be blank was not blank.</td>
</tr>
<tr>
<td>25</td>
<td>In addressing mode P, relocation mode indicator is not A, R, L, or C.</td>
</tr>
<tr>
<td>26</td>
<td>Patch record contains an invalid hexadecimal value.</td>
</tr>
<tr>
<td>27</td>
<td>Patch address is within BSS or area skipped by ORG.</td>
</tr>
</tbody>
</table>

Figure A-17 (Part 1 of 2). MODSF error codes
### MODSF Error Codes

<table>
<thead>
<tr>
<th>Error number</th>
<th>Cause of error</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Attempt was made to change relocation mode of an LIBF.</td>
</tr>
<tr>
<td>29</td>
<td>Relocation mode of second word of LIBF is not A.</td>
</tr>
<tr>
<td>30</td>
<td>Attempt to patch in an LIBF where non-LIBF appears in program.</td>
</tr>
<tr>
<td>31</td>
<td>Program requiring expansion is not followed by *END patch control record.</td>
</tr>
<tr>
<td>32</td>
<td>More than 31 words are to be updated.</td>
</tr>
<tr>
<td>33</td>
<td>Insufficient working storage for expansion.</td>
</tr>
<tr>
<td>34</td>
<td>Address specified for verification beyond end of program or in area skipped by BSS or ORG.</td>
</tr>
</tbody>
</table>

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.
### Mainline Program Messages

#### DFCNV error messages

<table>
<thead>
<tr>
<th>Error number and message</th>
<th>Cause of error</th>
</tr>
</thead>
</table>
| F01 INVALID DESCRIPTION CARD FIELD—COL. XX | 1. Numeric field at card column XX outside allowable field range  
2. Unrecognizable character in field at card column XX |
| F02 FILE NAME NOT IN LET/FLET—Y | 1. LET/FLET entry not found for file named on File Description card  
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. 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 | 1. Unrecognizable card precedes / * card (column 72 not D, S, or A).  
2. Multiple File Description cards read  
3. File Description card out of order  
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. |

**Figure A-18. DFCNV error messages**
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:

<table>
<thead>
<tr>
<th>INSTRUCTION ADDRESS</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>/001F</td>
<td>Invalid disk drive number in console entry switches</td>
</tr>
<tr>
<td></td>
<td>Indicated disk drive not ready</td>
</tr>
<tr>
<td>/0046</td>
<td>Disk read error</td>
</tr>
<tr>
<td></td>
<td>Waiting for interrupt from seek operation</td>
</tr>
<tr>
<td>/0048</td>
<td>Waiting for interrupt from reading sector @IDAD</td>
</tr>
</tbody>
</table>

*Note.* When any of these errors occur, perform another cold start.
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.
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Device causing wait</th>
<th>Cause of wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>/1000</td>
<td>1442 Card Read/Punch or 1442 Card Punch</td>
<td>Device is not ready, or last card indicator is on or read. Illegal device, device is not in system, illegal function, word count is over +80, or word count is zero or negative. This wait occurs in a DUP operation after a D112 error message has been printed.</td>
</tr>
<tr>
<td>/1001</td>
<td>Keyboard/Console Printer</td>
<td>Device is not ready. Device is not in system, illegal function, or word count is zero or negative. Keyboard input is expected (TYPEZ only).</td>
</tr>
<tr>
<td>/100F</td>
<td>1134/1055 Paper Tape Reader/Punch</td>
<td>Device is not ready. Illegal device, illegal function, word count is zero or negative, or illegal check digit.</td>
</tr>
<tr>
<td>/2000</td>
<td>2501 Card Reader</td>
<td>Device is not ready. Illegal function, word count is over +80, or word count is zero or negative.</td>
</tr>
<tr>
<td>/2001</td>
<td>Disk</td>
<td>Device is not ready. Make device ready and press PROGRAM START. 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). Write select/power unsafe. Turn the cartridge off, then on again, to reset the error condition.</td>
</tr>
<tr>
<td>/2002</td>
<td>1132 Printer</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/3000</td>
<td>1132 Printer</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/3001</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/4000</td>
<td>Card Reader</td>
<td>Device is not ready. Make device ready and press PROGRAM START. 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). Write select/power unsafe. Turn the cartridge off, then on again, to reset the error condition.</td>
</tr>
<tr>
<td>/4001</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/5000</td>
<td>1132 Printer</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/5001</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/5002</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/5003</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/5004</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/6000</td>
<td>1132 Printer</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
<tr>
<td>/6001</td>
<td>Disk</td>
<td>Device is not ready or end of forms. Illegal function, word count is over +60, or word count is zero or negative.</td>
</tr>
</tbody>
</table>

Figure B-1 (Part 1 of 2), ISS subroutine WAITS Monitor System Error Wait Codes B-3
### ISS subroutine WAITs

<table>
<thead>
<tr>
<th>Accumulator display</th>
<th>Device causing wait</th>
<th>Cause of wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>/7000</td>
<td>1627 Plotter</td>
<td>Device is not ready. Ready the device and press PROGRAM START. Illegal function, or word count is zero or negative. If PROGRAM START is pressed, the operation is retried (PLOT1 only).</td>
</tr>
<tr>
<td>/8001</td>
<td>SCA (STR mode) (SCAT1)</td>
<td>Invalid function code or invalid word count. Receive or transmit operation is not completed. Failure to establish synchronization before attempting to perform some transmit or receive operation, or attempting to receive before receiving INQ sequence.</td>
</tr>
<tr>
<td>/8001</td>
<td>SCA (BSC mode) (SCAT2 or SCAT3)</td>
<td>Invalid function code, word count, or subfunction code. Invalid start characters in the I/O area for a transmit operation. Invalid number of identification characters for an identification specification operation (SCAT2 only).</td>
</tr>
<tr>
<td>/9000</td>
<td>1403 Printer</td>
<td>Device is not ready or end of forms. Make device ready and press PROGRAM START. Illegal function, word count is over +60, zero or negative. To retry operation, press PROGRAM START (PRINT3 only). Parity check, scan check, or ring check. Reset check and press PROGRAM START. The operation is not retried (PRNZ only).</td>
</tr>
<tr>
<td>/A000</td>
<td>1231 Optical Mark Page Reader</td>
<td>Device is not ready. Illegal function. Feed check, last document is processed. Clear jam, make ready, do not refeed. Feed check, last document is not processed. Clear jam, make ready, refeed last document. If error was caused by double feed, refeed both documents.</td>
</tr>
</tbody>
</table>

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 PNCHO 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. | Punch station | 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. | Punch station | When the program halts, empty the hopper and clear the 1442 card path. If reading, place card 2 in the hopper before card 3 and ready the 1442. If punching, 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. | Punch station | 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. | Punch station | When the program halts, empty the hopper, clear the 1442 card path, place 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. | Punch station | 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
READ REG

Read registration indicates
incorrect card registration or
a difference between the
first and second reading
of a column.

Card positions after error

Punch station

Corner ➔ 1 ➔ Read station

Stacker ➔ 2 ➔ Hopper

PUNCH

Punch check indicates an
error in output punching.

Card positions after error

Punch station

Corner ➔ 1 ➔ Read station

Stacker ➔ 2 ➔ Hopper

Hopper ➔ 3 ➔ Hopper

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 READO 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 reinitializing the read function whenever the card reader is readied.

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.

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 TYPEO and WRTYO 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.

<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Error condition</th>
<th>Your response</th>
</tr>
</thead>
<tbody>
<tr>
<td>/3005</td>
<td>Paper tape reader not ready</td>
<td>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.</td>
</tr>
<tr>
<td>/3004</td>
<td>Paper tape punch not ready</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure B-3. PTUTL error wait codes
When a FORTRAN I/O error occurs, the system waits at SPRET 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.

<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Cause of error</th>
<th>Type of FORTRAN I/O</th>
<th>System action if you press PROGRAM START</th>
</tr>
</thead>
<tbody>
<tr>
<td>F000</td>
<td>No *IOCS card appeared with the mainline program and I/O was attempted in a subroutine.</td>
<td>SFIO 1</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F001</td>
<td>Logical unit defined incorrectly, or no *IOCS control record for specified I/O device.</td>
<td>SFIO 1</td>
<td>Execution continues with next FORTRAN statement.</td>
</tr>
<tr>
<td>F002</td>
<td>Requested record exceeds allocated buffer size.</td>
<td>SFIO 1</td>
<td>All the variables in the I/O list, following the one which has the erroneous format specification, will also be treated as errors.</td>
</tr>
<tr>
<td>F003</td>
<td>Illegal character encountered in input record.</td>
<td>SFIO 1</td>
<td>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.</td>
</tr>
<tr>
<td>F004</td>
<td>Exponent too large or too small in input field.</td>
<td>SFIO 1</td>
<td>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.</td>
</tr>
<tr>
<td>F005</td>
<td>More than one exponent field encountered in input field.</td>
<td>SFIO 1</td>
<td>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.</td>
</tr>
<tr>
<td>F006</td>
<td>More than one sign encountered in input field.</td>
<td>SFIO 1</td>
<td>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.</td>
</tr>
<tr>
<td>F007</td>
<td>More than one decimal point encountered in input field.</td>
<td>SFIO 1</td>
<td>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.</td>
</tr>
<tr>
<td>F008</td>
<td>Read of output-only device, or write of input-only device.</td>
<td>SFIO 1</td>
<td>Execution continues with next FORTRAN statement.</td>
</tr>
<tr>
<td>F009</td>
<td>Real variable transmitted with an I format specification or integer variable transmitted with an E or F format specification.</td>
<td>SFIO 1</td>
<td>The actual format specifications will be effectuated.</td>
</tr>
</tbody>
</table>

Figure B-4 (Part 1 of 2). FORTRAN I/O errors
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Cause of error</th>
<th>Type of FORTRAN I/O</th>
<th>System action if you press PROGRAM START</th>
</tr>
</thead>
<tbody>
<tr>
<td>F020</td>
<td>Illegal unit reference.</td>
<td>UFIO ²</td>
<td>UFIO not updated.</td>
</tr>
<tr>
<td>F021</td>
<td>Read list exceeds length of write list.</td>
<td>UFIO ³</td>
<td>UFIO updated.</td>
</tr>
<tr>
<td>F022</td>
<td>Record not existing for read list element.</td>
<td>UFIO ³</td>
<td>UFIO updated.</td>
</tr>
<tr>
<td>F023</td>
<td>Maximum length of $$$$$$ area on the disk has been exceeded. This error is unrecoverable and results in a call exit.</td>
<td>UFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F024</td>
<td>UFIO has not been initialized: there is no *IOCS (UDISK) record in the mainline program.</td>
<td>UFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F100</td>
<td>File not defined by DEFINE FILE statement.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F101</td>
<td>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.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F103</td>
<td>Disk F10 has not been initialized: there is no *IOCS (DISK) record in the mainline program.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F105</td>
<td>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.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F107</td>
<td>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.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
<tr>
<td>F10A</td>
<td>Subscripting has destroyed the define file table and/or core image header. This occurs when a subscript exceeds the specification in a DIMENSION.</td>
<td>SDFIO ³</td>
<td>CALL EXIT</td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Type of processing</th>
<th>Meaning</th>
<th>Your response</th>
<th>Console entry switch settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C000</td>
<td>Sequential file: random processing</td>
<td>Record number is not within the assigned limits of the file.</td>
<td>One of the following: Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If chaining, correct the card and reinsert it in the input stream, or bypass the chaining record and read the next card.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td>C001*</td>
<td>Sequential file: random processing</td>
<td>Record size is not within limits (maximum 640 characters).</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C002*</td>
<td>Sequential file: random processing</td>
<td>Records per sector is not maximum.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C003</td>
<td>Sequential file: random processing</td>
<td>No record was found. The record number is not a positive number.</td>
<td>One of the following: Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If chaining, correct the card and reinsert it in the input stream, or bypass the chaining record and read the next card.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td>C004</td>
<td>Sequential file: random processing</td>
<td>Write before read on an update file.</td>
<td>One of the following: Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td>C005*</td>
<td>Sequential file: random processing</td>
<td>File was accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C006*</td>
<td>Sequential file: random processing</td>
<td>I/O buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C010</td>
<td>Sequential file: sequential processing</td>
<td>Disk file is full.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C011*</td>
<td>Sequential file: sequential processing</td>
<td>A write is requested on an input file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C012*</td>
<td>Sequential file: sequential processing</td>
<td>A read is requested on an output file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
</tbody>
</table>

Figure B-5 (Part 1 of 5), RPG Object Program error messages
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Type of processing</th>
<th>Meaning</th>
<th>Your response</th>
<th>Console entry switch settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C013*</td>
<td>Sequential file: sequential processing</td>
<td>Record size is not within limits (maximum 640 characters).</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C014*</td>
<td>Sequential file: sequential processing</td>
<td>Number of records per sector is not maximum.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C015*</td>
<td>Sequential file: sequential processing</td>
<td>File was accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C016*</td>
<td>Sequential file: sequential processing</td>
<td>I/O buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C017</td>
<td>Sequential file: sequential processing</td>
<td>Write before read requested on an update file.</td>
<td>One of the following: Terminate the job.</td>
<td>Bypass the record and continue processing. All off. Press console START.</td>
</tr>
<tr>
<td>C020</td>
<td>ISAM load processing</td>
<td>Invalid type of processing on load function.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C021*</td>
<td>ISAM load processing</td>
<td>One of the following: Record size not within limits (maximum 636 characters). Number of records per sector is not maximum.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C022*</td>
<td>ISAM load processing</td>
<td>Key length is greater than maximum.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C023*</td>
<td>ISAM load processing</td>
<td>Index entry length is not same as length computed from key length.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C024*</td>
<td>ISAM load processing</td>
<td>Number of index entries per sector does not permit maximum number of records per sector.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C025</td>
<td>ISAM load processing</td>
<td>Prime data area is full.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C026</td>
<td>ISAM load processing</td>
<td>Index area is full.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C027*</td>
<td>ISAM load processing</td>
<td>File was accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C028*</td>
<td>ISAM load processing</td>
<td>Index buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
</tbody>
</table>

Figure B-5 (Part 2 of 5). RPG Object Program error messages
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Type of processing</th>
<th>Meaning</th>
<th>Your response</th>
<th>Console entry switch settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C029*</td>
<td>ISAM load processing</td>
<td>Prime data buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C02A</td>
<td>ISAM load processing</td>
<td>Input record is out of sequence.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td>C030*</td>
<td>ISAM add processing</td>
<td>Invalid type of processing on add function.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C031*</td>
<td>ISAM add processing</td>
<td>File was accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C032</td>
<td>ISAM add processing</td>
<td>Key length for this job is not same as key length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C033</td>
<td>ISAM add processing</td>
<td>Record length for this job is not same as record length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C034</td>
<td>ISAM add processing</td>
<td>Attempt was made to add record already on file.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td></td>
</tr>
<tr>
<td>C035</td>
<td>ISAM add processing</td>
<td>Overflow area is full. The file must be resequenced, or the data area must be made larger before another add run can be made.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C036*</td>
<td>ISAM add processing</td>
<td>Index buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C040*</td>
<td>ISAM file: sequential processing</td>
<td>Invalid type of processing on retrieve or update function.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C041*</td>
<td>ISAM file: sequential processing</td>
<td>Index buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C042*</td>
<td>ISAM file: sequential processing</td>
<td>Prime data buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C043</td>
<td>ISAM file: sequential processing</td>
<td>Key length for this job is not same as key length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
</tbody>
</table>

Figure B-5 (Part 3 of 5). RPG Object Program error messages

Monitor System Error Wait Codes  B-15
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Type of processing</th>
<th>Meaning</th>
<th>Your response</th>
<th>Console entry switch settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C044</td>
<td>ISAM file:</td>
<td>Record length for this job is not same as record length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>sequential processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C045^</td>
<td>ISAM file:</td>
<td>File accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>sequential processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C046</td>
<td>ISAM file:</td>
<td>Write before read requested on update file.</td>
<td>One of the following:</td>
<td>Bypass the record and continue processing.</td>
</tr>
<tr>
<td></td>
<td>sequential processing</td>
<td></td>
<td>Terminate the job.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISAM file:</td>
<td>Invalid type of processing on retrieve or update function.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td>C050^</td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISAM file:</td>
<td>Index buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISAM file:</td>
<td>Prime data buffer is not on even-word boundary.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C063</td>
<td>ISAM file:</td>
<td>Key length for this job is not same as key length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C064</td>
<td>ISAM file:</td>
<td>Record length for this job is not same as record length in file.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C065^</td>
<td>ISAM file:</td>
<td>File accessed when not open.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C066</td>
<td>ISAM file:</td>
<td>Write before read requested on update.</td>
<td>One of the following:</td>
<td>Bypass the record and continue processing.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td>Terminate the job.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISAM file:</td>
<td>Record not on file.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td>random processing</td>
<td></td>
<td>Terminate the job.</td>
<td>Bypass the record and continue processing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15 on, all others off. Press console START.</td>
</tr>
<tr>
<td>C111</td>
<td></td>
<td>Numeric records or matching fields out of sequence, or record is an undefined type.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>Bypass the record and continue processing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15 on, all others off. Press console START.</td>
</tr>
</tbody>
</table>

Figure B-5 (Part 4 of 5). RPG Object Program error messages
<table>
<thead>
<tr>
<th>ACCUMULATOR display</th>
<th>Type of processing</th>
<th>Meaning</th>
<th>Your response</th>
<th>Console entry switch settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>C12n</td>
<td>Halt switch set by object program (n = 1-9)</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set the halt switches off and continue processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C400</td>
<td>Write before read requested on combined file.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C430</td>
<td>Attempt to divide by zero.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continue processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The quotient will be set to zero.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C450</td>
<td>Results of multiply over 14 positions.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continue processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The result is set to zero.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C500</td>
<td>Monitor control card is read while punching on the 1442 Reader/Punch.</td>
<td>One of the following:</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminate the job.</td>
<td>15 on, all others off. Press console START.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bypass the record and continue processing.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C998</td>
<td>Table fields are out of sequence.</td>
<td>Terminate the job.</td>
<td>All off. Press console START.</td>
<td></td>
</tr>
</tbody>
</table>

Figure B-5 (Part 5 of 5). RPG Object Program error messages
### Appendix C. Monitor System Library Listing

**System library programs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINLINES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk Maintenance Programs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk initialization</td>
<td>DISC</td>
<td>2, None</td>
<td>SYSUP, RDREC, DISKZ</td>
</tr>
<tr>
<td>Print cartridge ID</td>
<td>IDENT</td>
<td>2, None</td>
<td>CALPR, DISKZ</td>
</tr>
<tr>
<td>Change cartridge ID</td>
<td>ID</td>
<td>2, None</td>
<td>RDREC, CALPR, DISKZ</td>
</tr>
<tr>
<td>Disk copy</td>
<td>COPY</td>
<td>2, None</td>
<td>RDREC, DISKZ</td>
</tr>
<tr>
<td>Write sector addresses in WS</td>
<td>ADRWS (cannot be called)</td>
<td>2, None</td>
<td>Linked from DUP DWADR</td>
</tr>
<tr>
<td>Delete CIB</td>
<td>DLCIB</td>
<td>2, None</td>
<td>RDREC, DISKZ</td>
</tr>
<tr>
<td>Dump system location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equivalence table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Library maintenance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System maintenance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk data file conversion</td>
<td>DFCNV</td>
<td>2, None</td>
<td>DISK1, ELD, FLD, NORM</td>
</tr>
</tbody>
</table>

**Paper Tape Utility**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTUTL</td>
<td>2, None</td>
<td>PAPHL, PAPPR, PAPT1, TYPE0</td>
<td>UBJ</td>
</tr>
</tbody>
</table>

**SUBROUTINES**

**Utility Calls**

<table>
<thead>
<tr>
<th>Name</th>
<th>4, 0</th>
<th>WRTYO</th>
<th>U5B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selective dump on console printer</td>
<td>DMTD0, DMTX0</td>
<td>4, 0</td>
<td>WRTYO</td>
</tr>
<tr>
<td>Dump 80</td>
<td>DMP80</td>
<td>4, 0</td>
<td>None</td>
</tr>
<tr>
<td>Update DCOM</td>
<td>SYSUP</td>
<td>4, 0</td>
<td>FSLEN, FSYSU</td>
</tr>
<tr>
<td>Call system print</td>
<td>CALPR</td>
<td>4, 0</td>
<td>FSLEN</td>
</tr>
<tr>
<td>Read *ID record</td>
<td>RDREC</td>
<td>4, 0</td>
<td>FSLEN</td>
</tr>
<tr>
<td>Fetch phase IDs or fetch system subroutine</td>
<td>FSLEN, FSYSU</td>
<td>4, 0</td>
<td>DISKZ</td>
</tr>
<tr>
<td>Dummy log subroutine for SCA subroutines</td>
<td>IOLOG/CPLOG</td>
<td>4, 0</td>
<td>None</td>
</tr>
</tbody>
</table>

**Common FORTRAN Calls**

<table>
<thead>
<tr>
<th>Name</th>
<th>4, 8</th>
<th>None</th>
<th>T3A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test data entry switches</td>
<td>DATSW</td>
<td>4, 8</td>
<td>None</td>
</tr>
<tr>
<td>Divide check test</td>
<td>DVCHK</td>
<td>4, 8</td>
<td>None</td>
</tr>
<tr>
<td>Functional error test</td>
<td>FCTST</td>
<td>4, 8</td>
<td>None</td>
</tr>
<tr>
<td>Overflow test</td>
<td>OVERF</td>
<td>4, 8</td>
<td>None</td>
</tr>
<tr>
<td>Selective dump</td>
<td>PDUMP</td>
<td>4, 0</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>4, 8</th>
<th>None</th>
<th>T3G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sense light control and test</td>
<td>SLITE, SLITT</td>
<td>4, 8</td>
<td>None</td>
</tr>
<tr>
<td>FORTRAN trace stop</td>
<td>TSTOP</td>
<td>4, 8</td>
<td>TSET</td>
</tr>
<tr>
<td>FORTRAN trace start</td>
<td>TSTRT</td>
<td>4, 8</td>
<td>TSET</td>
</tr>
<tr>
<td>Integer transfer of sign</td>
<td>ISIGN</td>
<td>4, 8</td>
<td>None</td>
</tr>
</tbody>
</table>

---

1. Not distributed to papertape users.
<table>
<thead>
<tr>
<th>System Library Programs</th>
<th>Names</th>
<th>Type and Subtype</th>
<th>Subroutines Required</th>
<th>ID Field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extended Arithmetic/Function Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended precision hyperbolic tangent</td>
<td>ETANH, ETNH</td>
<td>4, 8</td>
<td>EEXP, EADD, EDIV,</td>
<td>S2I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EGETP, EL/D/ESTO</td>
<td></td>
</tr>
<tr>
<td>Extended precision A**B function</td>
<td>EAXB, EAXBX</td>
<td>4, 8</td>
<td>EEXP, ELN, EMPY</td>
<td>S2C</td>
</tr>
<tr>
<td>Extended precision natural logarithm</td>
<td>ELN, EALOG</td>
<td>4, 8</td>
<td>XMD, EADD, EMPY,</td>
<td>S2E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EDIV, NORM, EGETP</td>
<td></td>
</tr>
<tr>
<td>Extended precision exponential</td>
<td>EEXP, EXPN</td>
<td>4, 8</td>
<td>XMD, FARC, EGETP</td>
<td>S2D</td>
</tr>
<tr>
<td>Extended precision square root</td>
<td>ESQR, ESQRT</td>
<td>4, 8</td>
<td>EADD, EMPY, EDIV,</td>
<td>S2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EGETP, NORM, XMD,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EGETP</td>
<td></td>
</tr>
<tr>
<td>Extended precision sine-cosine</td>
<td>ESIN, ESINE,</td>
<td>4, 8</td>
<td>EADD, EMPY, EDIV,</td>
<td>S2G</td>
</tr>
<tr>
<td></td>
<td>ECOS, ECOSN</td>
<td></td>
<td>EGETP, NORM, XMD,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EGETP</td>
<td></td>
</tr>
<tr>
<td>Extended precision arctangent</td>
<td>EATN, EATAN</td>
<td>4, 8</td>
<td>EADD, EMPY, EDIV,</td>
<td>S2B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMD, EGETP, NORM</td>
<td></td>
</tr>
<tr>
<td>Extended precision absolute value function</td>
<td>EABS, EA VL</td>
<td>4, 8</td>
<td>ESUB, ELD</td>
<td>S2A</td>
</tr>
<tr>
<td><strong>FORTRAN Sign Transfer Calls</strong></td>
<td></td>
<td></td>
<td>FSIGN</td>
<td></td>
</tr>
<tr>
<td>Extended precision transfer of sign</td>
<td>ESIGN</td>
<td>4, 8</td>
<td>ESUB, ELD</td>
<td>S2F</td>
</tr>
<tr>
<td>Standard precision transfer of sign</td>
<td>FSIGN</td>
<td>4, 8</td>
<td>FLD</td>
<td>R2F</td>
</tr>
<tr>
<td><strong>Standard Arithmetic/Function Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard precision hyperbolic tangent</td>
<td>FTANH, FTNH</td>
<td>4, 8</td>
<td>FEXP, FADD, FDIV,</td>
<td>R2I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FGETP, FLD/STO</td>
<td></td>
</tr>
<tr>
<td>Standard precision A**B function</td>
<td>FAXB, FAXBX</td>
<td>4, 8</td>
<td>FEXP, FLN, FMPY</td>
<td>R2C</td>
</tr>
<tr>
<td>Standard precision natural logarithm</td>
<td>FLN, FA LOG</td>
<td>4, 8</td>
<td>FSTO, XMDS, FADD,</td>
<td>R2E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FMPY, FDIV, NORM,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FGETP, XMDS, FARC,</td>
<td></td>
</tr>
<tr>
<td>Standard precision exponential</td>
<td>FEXP, FXPN</td>
<td>4, 8</td>
<td>FADD, FMPY, EDIV,</td>
<td>R2D</td>
</tr>
<tr>
<td>Standard precision square root</td>
<td>FSQR, FSQRT</td>
<td>4, 8</td>
<td>FGETP, NORM, XMD,</td>
<td>R2H</td>
</tr>
<tr>
<td>Standard precision sine-cosine</td>
<td>FSIN, FSINE,</td>
<td>4, 8</td>
<td>FSTO, FGGETP</td>
<td>R2G</td>
</tr>
<tr>
<td></td>
<td>FCOS, FCOSN</td>
<td></td>
<td>FDADD, EMPY, EDIV,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMD, FGGETP</td>
<td></td>
</tr>
<tr>
<td>Standard precision arctangent</td>
<td>FATN, FATA N</td>
<td>4, 8</td>
<td>FDADD, EMPY, EDIV,</td>
<td>R2B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XMD, FGGETP</td>
<td></td>
</tr>
<tr>
<td>Standard precision absolute value function</td>
<td>FABS, FA VL</td>
<td>4, 8</td>
<td>FGETP</td>
<td>R2A</td>
</tr>
<tr>
<td><strong>Common Arithmetic/Function Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed point (fractional) square root</td>
<td>XSQR</td>
<td>4, 8</td>
<td>None</td>
<td>T1C</td>
</tr>
<tr>
<td>Integer absolute function</td>
<td>IABS</td>
<td>4, 8</td>
<td>None</td>
<td>T1B</td>
</tr>
<tr>
<td>Floating binary/EBC decimal conversions</td>
<td>FBT D</td>
<td>4, 0</td>
<td>None</td>
<td>T1A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C-2
<table>
<thead>
<tr>
<th>System library programs</th>
<th>Names</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flipper for LOCAL/SOCAL Subprograms</strong></td>
<td>FLIPR</td>
<td>4, 0</td>
<td>DISKZ, DISK1, or DISKN</td>
<td>USD</td>
</tr>
<tr>
<td><strong>FORTRAN Trace Subroutines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended floating variable trace</td>
<td>SEAR, SEARX</td>
<td>3, 0</td>
<td>ESTO, TTEST, SWRT, SIOF, SCOMP</td>
<td>S2J</td>
</tr>
<tr>
<td>Fixed variable trace</td>
<td>SIAR, SIARX</td>
<td>3, 0</td>
<td>TTEST, SWRT, SIOI, SCOMP</td>
<td>T6B</td>
</tr>
<tr>
<td>Standard floating IF trace</td>
<td>SFI F</td>
<td>3, 0</td>
<td>FSTO, TTEST, SWRT, SIOF, SCOMP</td>
<td>R2K</td>
</tr>
<tr>
<td>Extended floating IF trace</td>
<td>SEIF</td>
<td>3, 0</td>
<td>FSTO, TTEST, SWRT, SIOF, SCOMP</td>
<td>S2K</td>
</tr>
<tr>
<td>Fixed IF trace</td>
<td>SIIF</td>
<td>3, 0</td>
<td>TTEST, SWRT, SIOI, SCOMP</td>
<td>T6C</td>
</tr>
<tr>
<td>Standard floating variable trace</td>
<td>SFAR, SFARX</td>
<td>3, 0</td>
<td>FSTO, TTEST, SWRT, SIOF, SCOMP</td>
<td>R2J</td>
</tr>
<tr>
<td>GO TO trace</td>
<td>SGOTO</td>
<td>3, 0</td>
<td>TTEST, SWRT, SIOI, SCOMP</td>
<td>T6A</td>
</tr>
<tr>
<td><strong>Nondisk FORTRAN Format I/O</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORTRAN format subroutine</td>
<td>SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX</td>
<td>3, 3</td>
<td>FLOAT, IFIX, ELD/ESTO or FLD/FSTO, PAUSE</td>
<td>T4C</td>
</tr>
<tr>
<td><strong>FORTRAN Find Subroutines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDFND</td>
<td>3, 1</td>
<td>DISKZ, DISK1, or DISKN</td>
<td>T4B</td>
<td></td>
</tr>
<tr>
<td><strong>Disk FORTRAN I/O</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDF, SDI, SDIX, SDFX, SDAI</td>
<td>3, 1</td>
<td>DISKZ, DISK1, or DISKN, PAUSE</td>
<td>T4A</td>
<td></td>
</tr>
<tr>
<td><strong>Unformatted FORTRAN Disk I/O</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UFIO, URED, UWRT, UIOI, UIOF, UIOAI, UIOAF, UIOFX, UIOIX, UCOMP, BCKSP, EOF, REWIND</td>
<td>3, 1</td>
<td>DISKZ, DISK1, or DISKN, PAUSE</td>
<td>T4D</td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN Common LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORTRAN pause</td>
<td>PAUSE</td>
<td>3, 0</td>
<td>None</td>
<td>T2A</td>
</tr>
<tr>
<td>FORTRAN stop</td>
<td>STOP</td>
<td>3, 2</td>
<td>None</td>
<td>T2B</td>
</tr>
<tr>
<td>FORTRAN subscript displacement calculation</td>
<td>SUBSC</td>
<td>3, 0</td>
<td>None</td>
<td>T2D</td>
</tr>
<tr>
<td>FORTRAN subroutine initialization</td>
<td>SUBIN</td>
<td>3, 0</td>
<td>None</td>
<td>T2C</td>
</tr>
<tr>
<td>FORTRAN trace test and set</td>
<td>TTEST, TSET</td>
<td>3, 0</td>
<td>None</td>
<td>T2E</td>
</tr>
<tr>
<td>System Library Listing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN I/O and Conversion Subroutines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN 1442 input/output subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARDZ 5, 3 HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04 T5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN 1442 output subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNCHZ 5, 3 HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04 T5G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN 2501 input subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READZ 5, 3 HOLEZ, GETAD, EBCTB, HOLTB, ILS04 T5J</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Disk I/O routine (part of supervisor)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKZ 5, 3 HOLEZ, GETAD, EBCTB, HOLTB, ILS04 T5C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN printer subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAPTZ 5, 3 ILS01 T5F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN 1403 printer subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRNTZ 5, 3 ILS04 T5H</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN keyboard-typewriter subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPEZ 5, 3 GETAD, EBCTB, HOLTZ, ILS04 T5I</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN typewriter subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRTYZ 5, 3 GETAD, EBCTB, HOLTZ, ILS04 T5L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN 1627 plotter subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOTX 5, 0 HOLEZ T5D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN hollerith to EBCDIC conversion</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETAD 3, 3 None T5C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN EBCDIC table</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBCTB 3, 3 None T5B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FORTRAN multiple terminal communications adapter (MTCA) call interface</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTCAZ 4, 0 None W5C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended Arithmetic/Function LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision get parameter subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EGETP 3, 2 ELD S1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision A</strong> <strong>I function</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAXI, EAXIX 3, 2 ELD/ESTO, EMD, FLVR S1B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision divide reverse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDVR, EDVRX 3, 2 ELD/ESTO, EMD, FLVR S1C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision float divide</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDIV, EDIVX 3, 2 EMD, FARC S1D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision float multiply</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMD, EMDX 3, 2 EMD, FARC S1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended precision subtract reverse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESBR, ESBRX 3, 2 EMD, FARC S1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended add-subtract</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EADD, ESUB, EADDX, ESUBX 3, 2 FARC, NORM S1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Extended load-store</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELD, ELDX, ESTO, ESTOX 3, 0 None S1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard Arithmetic/Function LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision get parameter subroutine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FGETP 3, 2 FLD R1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision A</strong> <strong>I function</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAXI, FAXIX 3, 2 FLD/FSTO, FMD, FLVR R1B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision divide reverse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDVR, FDVRX 3, 2 FLD/FSTO, FMD, FLVR R1C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision float divide</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDIV, FDIVX 3, 2 FMD, FARC R1D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision float multiply</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMPY, FMPYX 3, 2 FMD, FARC R1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision subtract reverse</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSRB, FSRBX 3, 2 FMD, FARC R1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard add-subtract</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FADD, FSUB, FADDX, FSUBX 3, 2 NORM, FARC R1G</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard load-store</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLD, FLDX, FSTO, FSTOX 3, 0 None R1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Standard precision fractional multiply</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMDS 3, 2 None S1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Common Arithmetic/Function LIBFs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDD</td>
<td>3, 2</td>
<td>XMD</td>
<td>S3G</td>
</tr>
<tr>
<td>XMD</td>
<td>3, 2</td>
<td>None</td>
<td>S3H</td>
</tr>
<tr>
<td>SNR</td>
<td>3, 2</td>
<td>None</td>
<td>S3F</td>
</tr>
<tr>
<td>FLOAT</td>
<td>3, 0</td>
<td>NORM</td>
<td>S3C</td>
</tr>
<tr>
<td>IFIX</td>
<td>3, 0</td>
<td>None</td>
<td>S3D</td>
</tr>
<tr>
<td>SNR</td>
<td>3, 2</td>
<td>None</td>
<td>S3B</td>
</tr>
<tr>
<td>FLOAT</td>
<td>3, 0</td>
<td>None</td>
<td>S3E</td>
</tr>
<tr>
<td>IFIX</td>
<td>3, 2</td>
<td>None</td>
<td>S3E</td>
</tr>
<tr>
<td>FIXI</td>
<td>3, 2</td>
<td>None</td>
<td>S3A</td>
</tr>
</tbody>
</table>

### Interrupt Service Subroutines

<table>
<thead>
<tr>
<th>Name</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD0</td>
<td>5, 0</td>
<td>ILS00, ILS04</td>
<td>U2A</td>
</tr>
<tr>
<td>CARD1</td>
<td>5, 0</td>
<td>ILS00, ILS04</td>
<td>U2B</td>
</tr>
<tr>
<td>READ0</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2L</td>
</tr>
<tr>
<td>READ1</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2M</td>
</tr>
<tr>
<td>PNCHO</td>
<td>5, 0</td>
<td>ILS00, ILS04</td>
<td>U2H</td>
</tr>
<tr>
<td>PNCH1</td>
<td>5, 0</td>
<td>ILS00, ILS04</td>
<td>U2I</td>
</tr>
<tr>
<td>DISK1</td>
<td>None</td>
<td>ILS02</td>
<td></td>
</tr>
<tr>
<td>DISKN</td>
<td>None</td>
<td>ILS02</td>
<td></td>
</tr>
<tr>
<td>SCAT1</td>
<td>5, 0</td>
<td>IOLOG/CPLOG, ILS01</td>
<td>W1F</td>
</tr>
<tr>
<td>SCAT2</td>
<td>5, 0</td>
<td>IOLOG/CPLOG, ILS01</td>
<td>W1H</td>
</tr>
<tr>
<td>SCAT3</td>
<td>5, 0</td>
<td>IOLOG/CPLOG, ILS01</td>
<td>W1I</td>
</tr>
<tr>
<td>PAPT1</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2D</td>
</tr>
<tr>
<td>PAPTN</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2E</td>
</tr>
<tr>
<td>PAPTX</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2F</td>
</tr>
<tr>
<td>PLOT1</td>
<td>5, 0</td>
<td>ILS03</td>
<td>U2G</td>
</tr>
<tr>
<td>PLOTX</td>
<td>5, 0</td>
<td>ILS03</td>
<td>V1L</td>
</tr>
<tr>
<td>PRNT1</td>
<td>5, 0</td>
<td>ILS01</td>
<td>U2J</td>
</tr>
<tr>
<td>PRNT2</td>
<td>5, 0</td>
<td>ILS01</td>
<td>W1E</td>
</tr>
<tr>
<td>PRNT3</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2K</td>
</tr>
<tr>
<td>TYPE0</td>
<td>5, 0</td>
<td>HOLL, PRTY, ILS04</td>
<td>U2N</td>
</tr>
<tr>
<td>WRTY0</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2O</td>
</tr>
<tr>
<td>OMPT1</td>
<td>5, 0</td>
<td>ILS04</td>
<td>U2C</td>
</tr>
<tr>
<td>MTCA0</td>
<td>5, 0</td>
<td>ILS03, TSM41, TSTTY</td>
<td>W5B</td>
</tr>
<tr>
<td>TSM41</td>
<td>4, 0</td>
<td>None</td>
<td>W5D</td>
</tr>
<tr>
<td>TSTTY</td>
<td>4, 0</td>
<td>None</td>
<td>W5E</td>
</tr>
</tbody>
</table>
**System Library Listing**

<table>
<thead>
<tr>
<th>Conversion Subroutines</th>
<th>Names</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73–75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary word to 6 decimal characters (card code)</td>
<td>BINDC</td>
<td>3, 0</td>
<td>None</td>
<td>U4B</td>
</tr>
<tr>
<td>Binary word to 4 hexadecimal characters (card code)</td>
<td>BINHX</td>
<td>3, 0</td>
<td>None</td>
<td>U4C</td>
</tr>
<tr>
<td>6 decimal characters (card code) to binary word</td>
<td>DCBIN</td>
<td>3, 0</td>
<td>None</td>
<td>U4G</td>
</tr>
<tr>
<td>EBCDIC to console printer output code</td>
<td>EBPRT</td>
<td>3, 0</td>
<td>EBPA, PRTY</td>
<td>U3A</td>
</tr>
<tr>
<td>Card code to EBCDIC-EBCDIC to card code</td>
<td>HOLEB</td>
<td>3, 0</td>
<td>EBPA, HOLL</td>
<td>U3B</td>
</tr>
<tr>
<td>Card code to console printer output code</td>
<td>HOLFR</td>
<td>3, 0</td>
<td>HOLL, PRTY</td>
<td>U3C</td>
</tr>
<tr>
<td>4 hexadecimal characters (card code) to binary word</td>
<td>HXBIN</td>
<td>3, 0</td>
<td>None</td>
<td>U3D</td>
</tr>
<tr>
<td>PTTC/8 to EBCDIC to PTTC/8</td>
<td>PAPEB</td>
<td>3, 0</td>
<td>EBPA</td>
<td>U3E</td>
</tr>
<tr>
<td>PTTC/8 to card code to card code</td>
<td>PAPHL</td>
<td>3, 0</td>
<td>EBPA, HOLL</td>
<td>U3F</td>
</tr>
<tr>
<td>PTTC/8 to console printer output code</td>
<td>PAPFR</td>
<td>3, 0</td>
<td>EBPA, PRTY</td>
<td>U3G</td>
</tr>
<tr>
<td>Card code to EBCDIC-EBCDIC to card code</td>
<td>SPEED</td>
<td>3, 0</td>
<td>None</td>
<td>U3H</td>
</tr>
<tr>
<td>4 of 8 code to EBCDIC to 4 of 8 code</td>
<td>EBC48</td>
<td>3, 0</td>
<td>HXCV, STRTB</td>
<td>W1A</td>
</tr>
<tr>
<td>4 of 8 code to IBM card code to 4 of 8 code</td>
<td>HOL48</td>
<td>3, 0</td>
<td>HXCV, HOLCA, STRTB</td>
<td>W1B</td>
</tr>
<tr>
<td>4 of 8 code to table of displacements</td>
<td>HXCV</td>
<td>3, 0</td>
<td>None</td>
<td>W1D</td>
</tr>
<tr>
<td>32-bit binary value to IBM card code</td>
<td>BIDEC</td>
<td>3, 0</td>
<td>None</td>
<td>U4A</td>
</tr>
<tr>
<td>IBM card code decimal value to 32-bit binary value</td>
<td>DECBI</td>
<td>3, 0</td>
<td>None</td>
<td>U4H</td>
</tr>
<tr>
<td>Supplement to all standard conversions except those involving PTTC/8</td>
<td>ZIPCO</td>
<td>3, 0</td>
<td>Any ZIPCO Conversion Table</td>
<td>U3I</td>
</tr>
<tr>
<td>MTCA code conversion</td>
<td>FEB41, B4B41, F41EB, 841EB, QEB41, Q41EB</td>
<td>4, 0</td>
<td>None</td>
<td>W5A</td>
</tr>
</tbody>
</table>

**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**

<p>| Dummy log subroutine called by SCAT1, SCAT2, SCAT3 | IOLOG, CPLOG | 4, 0 | None | W1J |</p>
<table>
<thead>
<tr>
<th>System Library Programs</th>
<th>Names</th>
<th>Type and Subtype</th>
<th>Subroutines Required</th>
<th>ID Field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interrupt Level Subroutines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt level zero subroutine</td>
<td>ILS00</td>
<td>7, 0</td>
<td>None</td>
<td>U1A</td>
</tr>
<tr>
<td>Interrupt level one subroutine</td>
<td>ILS01</td>
<td>7, 0</td>
<td>None</td>
<td>U1B</td>
</tr>
<tr>
<td>Interrupt level two subroutine (part of supervisor)</td>
<td>ILS02</td>
<td>7, 1</td>
<td>None</td>
<td>U1C</td>
</tr>
<tr>
<td>Interrupt level three subroutine</td>
<td>ILS03</td>
<td>7, 0</td>
<td>None</td>
<td>U1D</td>
</tr>
<tr>
<td>Interrupt level four subroutine (part of supervisor)</td>
<td>ILS04</td>
<td>7, 1</td>
<td>None</td>
<td>U1E</td>
</tr>
<tr>
<td><strong>Special Interrupt Level Subroutines</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(restores index register 3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt level zero subroutine</td>
<td>ILSX0</td>
<td>7, 0</td>
<td>None</td>
<td>U1F</td>
</tr>
<tr>
<td>Interrupt level one subroutine</td>
<td>ILSX1</td>
<td>7, 0</td>
<td>None</td>
<td>U1G</td>
</tr>
<tr>
<td>Interrupt level two subroutine</td>
<td>ILSX2</td>
<td>7, 0</td>
<td>None</td>
<td>U1H</td>
</tr>
<tr>
<td>Interrupt level three subroutine</td>
<td>ILSX3</td>
<td>7, 0</td>
<td>None</td>
<td>U1I</td>
</tr>
<tr>
<td>Interrupt level four subroutine</td>
<td>ILSX4</td>
<td>7, 0</td>
<td>None</td>
<td>U1J</td>
</tr>
<tr>
<td><strong>Standard Plot Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard precision character</td>
<td>FCHAR</td>
<td>4, 0</td>
<td>FSIN, FCOS,</td>
<td>V1F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FPLOT, FCHRX,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FLD, FSTO,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FSTO</td>
<td></td>
</tr>
<tr>
<td>Standard precision scale</td>
<td>SCALF</td>
<td>4, 0</td>
<td>FRULE</td>
<td>V1O</td>
</tr>
<tr>
<td>Standard precision grid</td>
<td>FGRID</td>
<td>4, 0</td>
<td>FPLOT, POINT,</td>
<td>V1H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FADD, FLD,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FSTO, SNR</td>
<td></td>
</tr>
<tr>
<td>Standard precision plot</td>
<td>FPLOT</td>
<td>4, 0</td>
<td>FMOVE, XYPLT,</td>
<td>V1I</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PLOTI</td>
<td></td>
</tr>
<tr>
<td><strong>Extended Plot Calls</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended precision character</td>
<td>ECHAR</td>
<td>4, 0</td>
<td>ESIN, ECOS,</td>
<td>V1A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EPLOT, ECHRX,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ELD, ESTO,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTOX</td>
<td></td>
</tr>
<tr>
<td>Extended precision scale</td>
<td>SCALE</td>
<td>4, 0</td>
<td>ERULE</td>
<td>V1N</td>
</tr>
<tr>
<td>Extended precision grid</td>
<td>EGRID</td>
<td>4, 0</td>
<td>EPLOT, POINT,</td>
<td>V1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EADD, ELD,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTO, SNR</td>
<td></td>
</tr>
<tr>
<td>Extended precision plot</td>
<td>EPLOT</td>
<td>4, 0</td>
<td>EMOVE, XYPLT,</td>
<td>V1D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PLOTI</td>
<td></td>
</tr>
<tr>
<td><strong>Common Plot Call</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point characters</td>
<td>POINT</td>
<td>4, 0</td>
<td>PLOTI</td>
<td>V1M</td>
</tr>
<tr>
<td><strong>Standard Plot LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard precision annotation</td>
<td>FCHRX, FCHRI, WCHRI</td>
<td>3, 0</td>
<td>FLOAT, FMPY, IFIX, FADD, FLDX, FINC, XYPLT, PLOTI, FSTO, FLD</td>
<td>V1G</td>
</tr>
<tr>
<td>Standard precision plot scaler</td>
<td>FRULE, FMOVE, FINC</td>
<td>3, 0</td>
<td>FLDX, FSUEX, FMPYX, FLD, FSTO, FMPY, IFIX, FADD</td>
<td>V1J</td>
</tr>
<tr>
<td>System Library Listing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Names</strong></td>
<td><strong>Type and subtype</strong></td>
<td><strong>Subroutines required</strong></td>
<td><strong>ID field (73-75)</strong></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>------------------------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Extended Plot LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended precision annotation</td>
<td>ECHRX, ECHRI, VCHR1</td>
<td>3, 0</td>
<td>FLOAT, FMPY, IFIX, EADD, ELDX, EINC, XYPLT, PLOTI, ESTOX, ELDX, ESUX, ELYX, ELD, ESTOX, ELY, IFIX, EADD, ESTO</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3, 0</td>
<td>ERULE, EMOVE, EINC</td>
<td></td>
</tr>
<tr>
<td><strong>Common Plot LIBFs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pen mover</td>
<td>XYPLT</td>
<td>3, 2</td>
<td>PLOTI</td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td>PLOTI</td>
<td>3, 2</td>
<td>PLOTX</td>
<td></td>
</tr>
<tr>
<td>Interrupt service</td>
<td>PLOTX</td>
<td>5, 0</td>
<td>ILS03</td>
<td></td>
</tr>
<tr>
<td><strong>Disk I/O</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential access</td>
<td>SEQOP, SEQIO, SEQCL</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td>Direct access</td>
<td>DAOPN, DAIO, DACLS</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td>ISAM load</td>
<td>ISLDO, ISLD, ISLDC</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td>ISAM add</td>
<td>ISADO, ISAD, ISADC</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td>ISAM sequential</td>
<td>ISEQO, ISETL, ISEQ, ISEQC</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td>ISAM random</td>
<td>ISRDO, ISRD, ISRDC</td>
<td>3, 0</td>
<td>DISKZ</td>
<td></td>
</tr>
<tr>
<td><strong>RPG Decimal Arithmetic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add, subtract, and numeric compare</td>
<td>RGADD, RGSUB, RGNCP</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Multiply</td>
<td>RGMLT</td>
<td>3, 0</td>
<td>RGBTD, RGDTB, RGERR</td>
<td></td>
</tr>
<tr>
<td>Divide</td>
<td>RGDIV</td>
<td>3, 0</td>
<td>RGBTD</td>
<td></td>
</tr>
<tr>
<td>Move remainder</td>
<td>RGMVR</td>
<td>3, 0</td>
<td>RGBTD</td>
<td></td>
</tr>
<tr>
<td>Binary conversion</td>
<td>RGBRD, RGDTB</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>RPG Sterling and Edit</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sterling input conversion</td>
<td>RGSTI</td>
<td>3, 0</td>
<td>RGBTD, RGDTB, RGMV1</td>
<td></td>
</tr>
<tr>
<td>Sterling output conversion</td>
<td>RGSTO</td>
<td>3, 0</td>
<td>RGBTD, RGDTB, RGMV2</td>
<td></td>
</tr>
<tr>
<td>Edit</td>
<td>RGEDT</td>
<td>3, 0</td>
<td>RGMV2, RGSIS5</td>
<td></td>
</tr>
<tr>
<td><strong>RPG Move</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From I/O buffer to core</td>
<td>RGMV1, RGMV5</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>From core to I/O buffer</td>
<td>RGMV2</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVE operation</td>
<td>RGMV3</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVE operation</td>
<td>RGMV4</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>RPG Compare</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alphameric</td>
<td>RGCMP</td>
<td>3, 0</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

1 Not distributed to paper tape users.
<table>
<thead>
<tr>
<th>System library programs</th>
<th>Names</th>
<th>Type and subtype</th>
<th>Subroutines required</th>
<th>ID field (73-75)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RPG Indicators</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>RGSI1</td>
<td>3, 0</td>
<td>None</td>
<td>W2I</td>
</tr>
<tr>
<td>Set resulting on</td>
<td>RGSI2</td>
<td>3, 0</td>
<td>None</td>
<td>W2H</td>
</tr>
<tr>
<td>Set on, set off</td>
<td>RGSI3, RGSI4</td>
<td>3, 0</td>
<td>None</td>
<td>W2G</td>
</tr>
<tr>
<td>Test for 0 or blank</td>
<td>RGSI5</td>
<td>3, 0</td>
<td>None</td>
<td>W2E</td>
</tr>
<tr>
<td><strong>RPG Miscellaneous</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test zone</td>
<td>RGTSZ</td>
<td>4, 0</td>
<td>None</td>
<td>W2D</td>
</tr>
<tr>
<td>Convert to binary</td>
<td>RGCVB</td>
<td>3, 0</td>
<td>None</td>
<td>W2C</td>
</tr>
<tr>
<td>Object time error</td>
<td>RGERR</td>
<td>4, 0</td>
<td>None</td>
<td>W2B</td>
</tr>
<tr>
<td>Blank after</td>
<td>RGBLK</td>
<td>3, 0</td>
<td>None</td>
<td>W2A</td>
</tr>
<tr>
<td>Alternating sequence</td>
<td>ALTSE</td>
<td></td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

1 Not distributed to paper tape users.
LET/FLET entry format

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 word sector header. All entries in LET or FLET are 3 words long and consist of a name and disk block count.

### Sector Header Format

<table>
<thead>
<tr>
<th>Word</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relative sector address for this cartridge only. The first sector of LET is relative sector address 0000, the second 0001, etc. The first sector of FLET is relative sector address 0010, the second 0011, etc.</td>
</tr>
<tr>
<td>2</td>
<td>Sector address of the UA (or sector address of FX if FLET)</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Number of words available in this LET/FLET sector</td>
</tr>
<tr>
<td>5</td>
<td>Sector address of the next LET/FLET sector on this cartridge. If this is the last FLET sector on this cartridge, word 5 is zero. If this is the last LET sector on this cartridge, word 5 contains the address of the first FLET sector.</td>
</tr>
</tbody>
</table>

### LET/FLET Entry Format

<table>
<thead>
<tr>
<th>Word</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, bits 0–1</td>
<td>00—if DSF format (LET only)</td>
</tr>
<tr>
<td></td>
<td>10—if DCI format</td>
</tr>
<tr>
<td></td>
<td>11—if data format</td>
</tr>
<tr>
<td>1, bits 2–15</td>
<td>Program or data file name in name code</td>
</tr>
<tr>
<td>2</td>
<td>Disk block (DB) count of program or data file</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Word</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, bits 0–1</td>
<td>Reserved</td>
</tr>
<tr>
<td>1, bits 2–15</td>
<td>Name code for 1DUMY</td>
</tr>
<tr>
<td>and word 2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DB count of entry</td>
</tr>
</tbody>
</table>

The last entry of LET is a 1DUMY entry that reflects the current size of available working storage.
The DUP control records DUMPLET or DUMPLET 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 DUMPLET follow:
```plaintext
// 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
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
*STOREDATA CD FX CIMG 18
CART ID 4444 DB ADDR 1FP0 DB CNT 0030
*STOREDATA CD UA DATAZ 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

<table>
<thead>
<tr>
<th>PROG FOR DB</th>
<th>NAME</th>
<th>MAT</th>
<th>CNT</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>FADD DSF 002 2280</td>
<td>FATN DSF 000A 22AA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FADDX FSUB FSUBX FAXI DSF 0006 2288 FAXIX FDIV DSF 0007 228E FDIVX FDVR DSF 0003 2295 FDVRX FGETP DSF 0003 2298 FLD DSF 0005 2298 FLDX FSTO FNPy DSF 0005 22AO FNPyX FSBR DSF 0002 22A8 FAVL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
<tr>
<td>PROG FOR DB</td>
<td>NAME</td>
<td>MAT</td>
<td>CNT</td>
<td>ADDR</td>
</tr>
</tbody>
</table>

=LET/FLET D-3

LET/FLET DUMPLET listing
```
LET/FLET

DUMPLET listing

**CIDN**  $FPAD  $FPAD  #CIBA  #ULET  $FLET
4444  02E5  02E5  0210  0220  01E8

**SCTR NO.**  UA/FXA  **WORDS AVAIL.**  CHAIN ADDR.
0003  0228  0105  01E8

<table>
<thead>
<tr>
<th>PROGRAM FOR DB DB</th>
<th>NAME MAT CNT ADDR</th>
<th>PROGRAM FOR DB DB</th>
<th>NAME MAT CNT ADDR</th>
<th>PROGRAM FOR DB DB</th>
<th>NAME MAT CNT ADDR</th>
<th>PROGRAM FOR DB DB</th>
<th>NAME MAT CNT ADDR</th>
<th>PROGRAM FOR DB DB</th>
<th>NAME MAT CNT ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQOP DSF 001C 2040</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQIO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGSTO DSF 001A 2069</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGSTI DSF 000F 2083</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B58A1 DSF 0024 2092</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B41E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>041E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTCA0 DSF 0023 2086</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTCA1 DSF 000E 2089</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSMA1 DSF 003A 20E7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSTTY DSF 000B 2E21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDUMY 0004 2E2C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA2 DDF 0020 2E30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDUMY 3580 2E50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

End of DUMPLET/FLET
LET/FLET
DUMPFLET listing

#DUMPFLET

FLET

<CIDN 5PPAD 6FPAD 6CIBA 6ULET =FLET
4444 02E5 02E5 0210 0220 01E8
SCTR NO. UA/FXA. WORDS AVAIL. CHAIN ADDR.
0010 01F0 0192 0000

PROG FOR DB DB PROGRAM FOR DB DB NAME MAT CNT ADDR NAME MAT CNT ADDR
NAME MAT CNT ADDR NAME MAT CNT ADDR NAME MAT CNT ADDR NAME MAT CNT ADDR
DATA DDF 0020 1F00
CIDGE DCI 0030 1F20
IDUMY 0180 1F50

END OF DUMPFLET
Appendix G. Resident Monitor (Including Table of Equivalences)

The contents of this appendix are not to be construed as an external specification; that is, the locations in this listing may be changed. SPRET, SIREQ, SEXIT, SLINK, and SDUMP 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.

Resident Monitor (Including Table of Equivalences) G-1
Resident Monitor Listing

00064  # 48  *  FILE PROTECT ADDR, LOGICAL   PMN00640
00065  # 49  *  DRIVE 3 (BASE)   PMN00650
00066  # 49  *  FILE PROTECT ADDR, LOGICAL   PMN00660
00067  # 50  *  DRIVE 4 (BASE)   PMN00670
00068  # 51  =PCIO  CARTRIDGE ID, PHYSICAL DRIVE 0   PMN00680
00069  # 51  =PCIO  CARTRIDGE ID, PHYSICAL DRIVE 1   PMN00690
00070  # 52  *  DRIVE 5 (BASE)   PMN00700
00071  # 53  *  CARTRIDGE ID, PHYSICAL DRIVE 3   PMN00710
00072  # 54  *  CARTRIDGE ID, PHYSICAL DRIVE 4   PMN00720
00073  # 55  =CDLO  CARTRIDGE ID, LOGICAL DRIVE 0   PMN00730
00074  # 56  *  CARTRIDGE ID, LOGICAL DRIVE 1   PMN00740
00075  # 57  *  CARTRIDGE ID, LOGICAL DRIVE 2   PMN00750
00076  # 58  *  CARTRIDGE ID, LOGICAL DRIVE 3   PMN00760
00077  # 59  *  CARTRIDGE ID, LOGICAL DRIVE 4   PMN00770
00078  # 60  *  SCIA  SCRT ADDR OF CID, LOGICAL DR 0   PMN00780
00079  # 61  *  SCRT ADDR OF CID, LOGICAL DR 1   PMN00790
00080  # 62  *  SCRT ADDR OF CID, LOGICAL DR 2   PMN00800
00081  # 63  *  SCRT ADDR OF CID, LOGICAL DR 3   PMN00810
00082  # 64  *  SCRT ADDR OF CID, LOGICAL DR 4   PMN00820
00083  # 65  =SCRA  SCRT ADDR, LOGICAL DRIVE 0   PMN00830
00084  # 66  *  SCRA, LOGICAL DRIVE 1   PMN00840
00085  # 67  *  SCRA, LOGICAL DRIVE 2   PMN00850
00086  # 68  *  SCRA, LOGICAL DRIVE 3   PMN00860
00087  # 69  *  SCRA, LOGICAL DRIVE 4   PMN00870
00088  # 70  =FMAT  FORMAT OF PROG IN WS, DRIVE 0   PMN00880
00089  # 71  *  FORMAT OF PROG IN WS, DRIVE 1   PMN00890
00090  # 72  *  FORMAT OF PROG IN WS, DRIVE 2   PMN00900
00091  # 73  *  FORMAT OF PROG IN WS, DRIVE 3   PMN00910
00092  # 74  *  FORMAT OF PROG IN WS, DRIVE 4   PMN00920
00093  # 75  =FLET  FLET SCR ADDR, LOGICAL DR 0   PMN00930
00094  # 76  *  FLET SCR ADDR, LOGICAL DR 1   PMN00940
00095  # 77  *  FLET SCR ADDR, LOGICAL DR 2   PMN00950
00096  # 78  *  FLET SCR ADDR, LOGICAL DR 3   PMN00960
00097  # 79  *  FLET SCR ADDR, LOGICAL DR 4   PMN00970
00098  # 80  =ULET  LET SCR ADDR, LOGICAL DR 0   PMN00980
00099  # 81  *  LET SCR ADDR, LOGICAL DR 1   PMN00990
00100  # 82  *  LET SCR ADDR, LOGICAL DR 2   PMN01000
00101  # 83  *  LET SCR ADDR, LOGICAL DR 3   PMN01010
00102  # 84  *  LET SCR ADDR, LOGICAL DR 4   PMN01020
00103  # 85  =MSTC  BLK CNT OF PROG IN WS, DRIVE 0   PMN01030
00104  # 86  *  BLK CNT OF PROG IN WS, DRIVE 1   PMN01040
00105  # 87  *  BLK CNT OF PROG IN WS, DRIVE 2   PMN01050
00106  # 88  *  BLK CNT OF PROG IN WS, DRIVE 3   PMN01060
00107  # 89  *  BLK CNT OF PROG IN WS, DRIVE 4   PMN01070
00108  # 90  =CSHM  SCRT CNT CUSHION,LOGICAL DR 0   PMN01080
00109  # 91  *  SCRT CNT CUSHION,LOGICAL DR 1   PMN01090
00110  # 92  *  SCRT CNT CUSHION,LOGICAL DR 2   PMN01100
00111  # 93  *  SCRT CNT CUSHION,LOGICAL DR 3   PMN01110
00112  # 94  *  SCRT CNT CUSHION,LOGICAL DR 4   PMN01120
00113  # 95-319  *  RESERVED FOR FUTURE USE   PMN01130

RESIDENT MONITOR

00114  ******************************************   PMN01140
00115  STATUS -- VERSION 2, MODIFICATION 13   PMN01150
00116  *FUNCTION/OPERATION-   PMN01160
00117  **THIS SECTION ALWAYS REMAINS IN CORE. IT   PMN01170
00118  IS COMPRISED OF THE COMMUNICATIONS   PMN01180
00119  REGION (COMM), THE SKELETON SUPERVISOR, AND   PMN01190
00120  THE DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01200
00121  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01210
00122  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01220
00123  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01230
00124  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01240
00125  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01250
00126  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01260
00127  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01270
00128  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01280
00129  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01290
00130  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01300
00131  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01310
00132  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01320
00133  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01330
00134  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01340
00135  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01350
00136  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01360
00137  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01370
00138  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01380
00139  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01390
00140  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01400
00141  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01410
00142  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01420
00143  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01430
00144  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01440
00145  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01450
00146  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01460
00147  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01470
00148  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01480
00149  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01490
00150  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01500
00151  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01510
00152  PARAMETERS, THIS SECTION CONSISTS OF A SUBROUTINE FOR READING   PMN01520
00153  A DISK I/O SUBROUTINE, NOMINALLY DISKZONE (THE   PMN01530
00154  CALLING SEQUENCE IS DESCRIPTED IN DCOM, CARTRIDGE   PMN01540

DCOM cartridge parameters
As Updated October 22, 1976

00155 * DC FORMAT * PMN01550
00156 * DC LIMIT 1 * PMN01560
00157 * DC LIMIT 2 * PMN01570
00158 * FORMAT IS A CODE INDICATING THE FORMAT * PMN01580
00159 * OF THE DUMP. IF FORMAT IS NEGATIVE, * PMN01590
00160 * THE AUXILIARY SUPERVISOR IS FETCHED * PMN01600
00161 * AND CONTROL PASSED TO IT. * PMN01610
00162 * * DZO00-ENTERED WHEN THE CALLER WISHES TO * PMN01620
00163 * PERFORM A DISK I/O OPERATION. THE * PMN01630
00164 * CALLING SEQUENCE VARIES WITH THE * PMN01640
00165 * VERSION OF THE DISK I/O SUBROUTINE. * PMN01650
00166 * A 51200/$1400—ENTERED WHEN THE OPERATION— * PMN01660
00167 * COMPLETE INTERRUPT OCCURS ON * PMN01670
00168 * LEVEL 2/4. * PMN01680
00169 * INPUT—N/A * PMN01690
00170 * OUTPUT—WORDS 6-4090 SAVED ON THE CIB ON A CALL * PMN01700
00171 * DUMP * PMN01710
00172 * EXTERNAL REFERENCES—N/A * PMN01720
00173 * EXITS— * PMN01730
00174 * * THE EXITS FROM THE SUBROUTINES AT SPRET * PMN01740
00175 * AND SPST1, SPST2, SPST3, * PMN01750
00176 * SPST4, AND ESTOP ARE BRANCH INSTRUCTIONS FOLLOWING A * PMN01760
00177 * WAIT INSTRUCTION. ESTOP TURNS OFF IN— * PMN01770
00178 * TERRUPT LEVEL 5 AFTER THE START KEY IS * PMN01780
00179 * DEPRESSED. * PMN01790
00180 * * THE EXITS FROM SEXIT, SLINK, AND SDUMP ARE TO * PMN01800
00181 * THE CORE IMAGE LOADER, PHASE 1, AFTER THAT PHASE HAS BEEN FETCHED. * PMN01810
00182 * * THE EXITS FROM 07000 IS BACK TO THE * PMN01820
00183 * CALLER AFTER THE REQUESTED DISK OPERATION HAS BEEN INITIATED. * PMN01830
00184 * * THE EXITS FROM $1200/$1400 ARE BACK TO THE ADDRESSES FROM WHICH THE DISK OPERATION COMPLETE INTERRUPT OCCURRED AFTER THE INTERRUPT HAS BEEN SERVICED BY THE APPROPRIATE ISS. * PMN01840
00185 * ERROR—N/A * PMN01850
00186 * ATABLES/WORK AREAS— * PMN01860
00187 * * SCH12 * PMN01870
00188 * * SCILA * PMN01880
00189 * * SCLSW * PMN01890
00190 * * SCOMN * PMN01900
00191 * * SCORE * PMN01910
00192 * * SCADR * PMN01920
00193 * * SOBSY * PMN01930
00194 * * SCYL * PMN01940
00195 * * SCYFN * PMN01950
00196 * * SIBT2 * PMN01960
00197 * * SI8T4 * PMN01970
00198 * * SIBSY * PMN01980
00199 * * SIOCT * PMN01990
00200 * * SKCSW * PMN02000
00201 * * SLAST * PMN02010
00202 * * $LNXQ 2-9 * PMN02020
00203 * * NDUP * PMN02030
00204 * * SREQ * PMN02040
00205 * * $SR5 * PMN02050
00206 * * SREQ * PMN02060
00207 * * SFR5 * PMN02070
00208 * * SFR5 * PMN02080
00209 * * SFR5 * PMN02090
00210 * * SFR5 * PMN02100
00211 * * SFR5 * PMN02110
00212 * * SFR5 * PMN02120
00213 * * SFR5 * PMN02130
00214 * * SFR5 * PMN02140
00215 * * SFR5 * PMN02150
00216 * * SFR5 * PMN02160
00217 * * SFR5 * PMN02170
00218 * * SFR5 * PMN02180
00219 * * SFR5 * PMN02190
00220 * * SFR5 * PMN02200
00221 * * SFR5 * PMN02210
00222 * * SFR5 * PMN02220
00223 * * SFR5 * PMN02230
00224 * * SFR5 * PMN02240
00225 * * SFR5 * PMN02250
00226 * * SFR5 * PMN02260
00227 * * SFR5 * PMN02270
00228 * * SFR5 * PMN02280
00229 * * SFR5 * PMN02290
00230 * * SFR5 * PMN02300
00231 * * SFR5 * PMN02310
00232 * * SFR5 * PMN02320
00233 * * SFR5 * PMN02330
00234 * * SFR5 * PMN02340
00235 * * SFR5 * PMN02350
00236 * * SFR5 * PMN02360
00237 * * SFR5 * PMN02370
00238 * * SFR5 * PMN02380
00239 * * SFR5 * PMN02390
00240 * * SFR5 * PMN02400
00241 * * SFR5 * PMN02410
00242 * * SFR5 * PMN02420
00243 * * SFR5 * PMN02430
00244 * * SFR5 * PMN02440
00245 * * SFR5 * PMN02450
00246 * * SFR5 * PMN02460
00247 * * SFR5 * PMN02470

Resident Monitor (Including Table of Equivalences) G-3
ORG 4

$18A DC ** SCTR ADDR OF THE CIB
$100 DC ** ADDR OF CHANNEL 12 INDICATOR
$0257 DC ** LENGTH OF COMMON (11 WORDS)

*ULTIMATE RESIDENCE OF THE INTERRUPT TV

*LEVEL 0 BRANCH ADDRESS

*LEVEL 2 BRANCH ADDRESS

*LEVEL 1 BRANCH ADDRESS

*$1200 LEVEL 2 BRANCH ADDR

*$1400 LEVEL 4 BRANCH ADDR

*$STOPLEVEL 5 BRANCH ADDR

*** CALL ENTRY POINT TO SKELTON SUPERVISOR

*** FETCH CORE IMAGE LOADER, PHASE 1 PMNO3100

*** CALL LINK ENTRY POINT

*** SAVE 1ST 4K OF CORE ON THE CIB

*** SAVE I/O LINK ENTRY POINT

*** FETCH CORE IMAGE LOADER, PHASE 2

*** CALL EXIT INDICATOR

*** SAVE LAST 4K OF CORE ON THE CIB

*** CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR

*** FETCH CORE IMAGE LOADER, PHASE 1

*** CALL EXIT INDICATOR

*** CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR

*** CALL EXIT ENTRY POINT TO SKELETON SUPERVISOR

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT

**ENTRY POINT
Resident Monitor Listing

RESIDENT MONITOR (INCLUDING TABLE OF EQUIVALENCES) G-5

As Updated October 22, 1976

By TNL GN34-0353

Page of GC26-3717-9, 10

The image contains a detailed listing of computer code, likely for a disk controller or file system, with various operations and addresses

---

.security note: Due to the dense and technical nature of the content, it appears to be a listing of resident monitor code, possibly for a file system or disk controller, with entries like:

- **ENTRY POINT**: Code locations that serve as entry points for various operations.
- **TRAP FOR POSTOPERATIVE I/O ERRORS**: Code to handle errors after I/O operations.
- **FILE PROTECT ADDR, LOGICAL DR**: Addresses related to file protection and logical drives.
- **FILE PROTECT ADDR, LOGICAL DR 0**: Address for file protection and logical drive 0.
- **FILE PROTECT ADDR, LOGICAL DR 1**: Address for file protection and logical drive 1.
- **FILE PROTECT ADDR, LOGICAL DR 2**: Address for file protection and logical drive 2.
- **FILE PROTECT ADDR, LOGICAL DR 3**: Address for file protection and logical drive 3.
- **FILE PROTECT ADDR, LOGICAL DR 4**: Address for file protection and logical drive 4.
- **FILE PROTECT ADDR, LOGICAL DR 5**: Address for file protection and logical drive 5.
- **FILE PROTECT ADDR, LOGICAL DR 6**: Address for file protection and logical drive 6.
- **FILE PROTECT ADDR, LOGICAL DR 7**: Address for file protection and logical drive 7.
- **FILE PROTECT ADDR, LOGICAL DR 8**: Address for file protection and logical drive 8.
- **FILE PROTECT ADDR, LOGICAL DR 9**: Address for file protection and logical drive 9.
- **FILE PROTECT ADDR, LOGICAL DR 10**: Address for file protection and logical drive 10.
- **FILE PROTECT ADDR, LOGICAL DR 11**: Address for file protection and logical drive 11.
- **FILE PROTECT ADDR, LOGICAL DR 12**: Address for file protection and logical drive 12.
- **FILE PROTECT ADDR, LOGICAL DR 13**: Address for file protection and logical drive 13.
- **FILE PROTECT ADDR, LOGICAL DR 14**: Address for file protection and logical drive 14.
- **FILE PROTECT ADDR, LOGICAL DR 15**: Address for file protection and logical drive 15.

Additional notes:
- **FILE PROTECT ADDR, LOGICAL DR**: Addresses related to file protection and logical drives are provided, indicating how data is protected and stored across different drives.
- **TRAP FOR POSTOPERATIVE I/O ERRORS**: Indicates code to handle errors after I/O operations, ensuring data integrity and system reliability.
- **ENTRY POINT**: Code locations that serve as entry points for various operations, facilitating easy navigation and execution.

---

.security note: The code listing involves complex operations and addresses, indicative of a detailed system design, possibly for a file system or disk controller, with a focus on error handling and data protection. The entries for file protection and logical drives suggest a robust system design aimed at ensuring data integrity and security.
G-6

As Updated October 22, 1976

Page 6

* RESIDENT MONITOR LISTING *

00434 *** THE ARM POSITION IS UPDATED WHENEVER A SEEK OCCURS. PNM04180
00435
00436
00437 CYL DC 0 ARM POSITION FOR LOGICAL DRIVE 0 PNM04340
00438 DC 0 ARM POSITION FOR LOGICAL DRIVE 1 PNM04350
00439 DC 0 ARM POSITION FOR LOGICAL DRIVE 2 PNM04360
00440 DC 0 ARM POSITION FOR LOGICAL DRIVE 3 PNM04370
00441 DC 0 ARM POSITION FOR LOGICAL DRIVE 4 PNM04380
00442
00443 *** BELOW ARE THE DISK AREA CODES. A ZERO INDICATES THE CORRESPONDING DRIVE IS NOT ON THE SYSTEM. PNM04410
00444
00445 *** THE ADR OF THE CYLINDER IN WHICH A DEFECT OCCURS. ** PMN04420
00446
00447 AREA CODE FOR LOGICAL DRIVE 0 PNM04440
00448 AREA CODE FOR LOGICAL DRIVE 1 PNM04450
00449 AREA CODE FOR LOGICAL DRIVE 2 PNM04460
00450 AREA CODE FOR LOGICAL DRIVE 3 PNM04470
00451 AREA CODE FOR LOGICAL DRIVE 4 PNM04480
00452
00453 *** THE ADR IS STORED IN WORD 1205 BY PHASE 2 OF THE INTERRUPT SERVICE ROUTINE. ** PMN04490
00454 *** THE CORE IMAGE LOADER. WORD 10 ALWAYS CON TAINS THE ADDRESS OF $1400. * PMN04440
00455 *** CON TAINS THE ADDRESS OF $1200. PMN04450
00456 *** CON TAINS THE ADDRESS OF $1400. PMN04460
00457 *** CON TAINS THE ADDRESS OF $1400. PMN04470
00458 *** CON TAINS THE ADDRESS OF $1400. PMN04480
00459 *** CON TAINS THE ADDRESS OF $1400. PMN04490
00460 *** CON TAINS THE ADDRESS OF $1400. PMN04500
00461 *** CON TAINS THE ADDRESS OF $1400. PMN04510
00462 *** CON TAINS THE ADDRESS OF $1400. PMN04520
00463 *** CON TAINS THE ADDRESS OF $1400. PMN04530
00464 *** CON TAINS THE ADDRESS OF $1400. PMN04540
00465 *** CON TAINS THE ADDRESS OF $1400. PMN04550
00466 *** CON TAINS THE ADDRESS OF $1400. PMN04560
00467 *** CON TAINS THE ADDRESS OF $1400. PMN04570
00468 *** CON TAINS THE ADDRESS OF $1400. PMN04580
00469 *** CON TAINS THE ADDRESS OF $1400. PMN04590
00470 *** CON TAINS THE ADDRESS OF $1400. PMN04600
00471 *** CON TAINS THE ADDRESS OF $1400. PMN04610
00472 *** CON TAINS THE ADDRESS OF $1400. PMN04620
00473 *** CON TAINS THE ADDRESS OF $1400. PMN04630
00474 *** CON TAINS THE ADDRESS OF $1400. PMN04640
00475 *** CON TAINS THE ADDRESS OF $1400. PMN04650
00476 *** CON TAINS THE ADDRESS OF $1400. PMN04660
00477 *** CON TAINS THE ADDRESS OF $1400. PMN04670
00478 *** CON TAINS THE ADDRESS OF $1400. PMN04680
00479 *** CON TAINS THE ADDRESS OF $1400. PMN04690
00480 *** CON TAINS THE ADDRESS OF $1400. PMN04700
00481 *** CON TAINS THE ADDRESS OF $1400. PMN04710
00482 *** CON TAINS THE ADDRESS OF $1400. PMN04720
00483 *** CON TAINS THE ADDRESS OF $1400. PMN04730
00484 *** CON TAINS THE ADDRESS OF $1400. PMN04740
00485 *** CON TAINS THE ADDRESS OF $1400. PMN04750
00486 *** CON TAINS THE ADDRESS OF $1400. PMN04760
00487 *** CON TAINS THE ADDRESS OF $1400. PMN04770
00488 *** CON TAINS THE ADDRESS OF $1400. PMN04780
00489 *** CON TAINS THE ADDRESS OF $1400. PMN04790
00490 *** CON TAINS THE ADDRESS OF $1400. PMN04800
00491 *** CON TAINS THE ADDRESS OF $1400. PMN04810
00492 *** CON TAINS THE ADDRESS OF $1400. PMN04820
00493 *** CON TAINS THE ADDRESS OF $1400. PMN04830
00494 *** CON TAINS THE ADDRESS OF $1400. PMN04840
00495 *** CON TAINS THE ADDRESS OF $1400. PMN04850
00496 *** CON TAINS THE ADDRESS OF $1400. PMN04860
00497 *** CON TAINS THE ADDRESS OF $1400. PMN04870
00498 *** CON TAINS THE ADDRESS OF $1400. PMN04880
00499 *** CON TAINS THE ADDRESS OF $1400. PMN04890
00500 *** CON TAINS THE ADDRESS OF $1400. PMN04900
00501 *** CON TAINS THE ADDRESS OF $1400. PMN04910
00502 *** CON TAINS THE ADDRESS OF $1400. PMN04920
00503 *** CON TAINS THE ADDRESS OF $1400. PMN04930
00504 *** CON TAINS THE ADDRESS OF $1400. PMN04940
00505 *** CON TAINS THE ADDRESS OF $1400. PMN04950
00506 *** CON TAINS THE ADDRESS OF $1400. PMN04960
00507 *** CON TAINS THE ADDRESS OF $1400. PMN04970
00508 *** CON TAINS THE ADDRESS OF $1400. PMN04980
00509 *** CON TAINS THE ADDRESS OF $1400. PMN04990
00510 *** CON TAINS THE ADDRESS OF $1400. PMN05000
00511 *** CON TAINS THE ADDRESS OF $1400. PMN05010
00512 *** CON TAINS THE ADDRESS OF $1400. PMN05020
00513 *** CON TAINS THE ADDRESS OF $1400. PMN05030
00514 *** CON TAINS THE ADDRESS OF $1400. PMN05040
00515 *** CON TAINS THE ADDRESS OF $1400. PMN05050
00516 *** CON TAINS THE ADDRESS OF $1400. PMN05060
00517 *** CON TAINS THE ADDRESS OF $1400. PMN05070
00518 *** CON TAINS THE ADDRESS OF $1400. PMN05080
00519 *** CON TAINS THE ADDRESS OF $1400. PMN05090
00520 *** CON TAINS THE ADDRESS OF $1400. PMN05100
00521 *** CON TAINS THE ADDRESS OF $1400. PMN05110
00522 *** CON TAINS THE ADDRESS OF $1400. PMN05120
00523 *** CON TAINS THE ADDRESS OF $1400. PMN05130
00524 *** CON TAINS THE ADDRESS OF $1400. PMN05140
00525 *** CON TAINS THE ADDRESS OF $1400. PMN05150
00526 *** CON TAINS THE ADDRESS OF $1400. PMN05160
00527 *** CON TAINS THE ADDRESS OF $1400. PMN05170
00528 *** CON TAINS THE ADDRESS OF $1400. PMN05180
00529 *** CON TAINS THE ADDRESS OF $1400. PMN05190
00530 *** CON TAINS THE ADDRESS OF $1400. PMN05200
00531 *** CON TAINS THE ADDRESS OF $1400. PMN05210
00532 *** CON TAINS THE ADDRESS OF $1400. PMN05220
00533 *** CON TAINS THE ADDRESS OF $1400. PMN05230

G-6
Resident Monitor Listing

DISKZ

005562  0000  0010  00567  XIO  $1494  SENSE ILSW  PMNO5590
005563  0000  0010  00568  DC  0  SENSE ILSW  PMNO5590
005564  0000  0010  00569  DC  0  SENSE ILSW  PMNO5590
005565  0000  0010  00570  DC  0  SENSE ILSW  PMNO5590
005566  0000  0010  00571  DC  0  SENSE ILSW  PMNO5590
005567  0000  0010  00572  DC  0  SENSE ILSW  PMNO5590
005568  0000  0010  00573  DC  0  SENSE ILSW  PMNO5590
005569  0000  0010  00574  DC  0  SENSE ILSW  PMNO5590
005570  0000  0010  00575  DC  0  SENSE ILSW  PMNO5590
005571  0000  0010  00576  DC  0  SENSE ILSW  PMNO5590
005572  0000  0010  00577  DC  0  SENSE ILSW  PMNO5590
005573  0000  0010  00578  DC  0  SENSE ILSW  PMNO5590
005574  0000  0010  00579  DC  0  SENSE ILSW  PMNO5590
005580  0000  0010  00580  DC  0  SENSE ILSW  PMNO5590
005581  0000  0010  00581  DC  0  SENSE ILSW  PMNO5590
005582  0000  0010  00582  DC  0  SENSE ILSW  PMNO5590
005583  0000  0010  00583  DC  0  SENSE ILSW  PMNO5590
005584  0000  0010  00584  DC  0  SENSE ILSW  PMNO5590
005585  0000  0010  00585  DC  0  SENSE ILSW  PMNO5590
005586  0000  0010  00586  DC  0  SENSE ILSW  PMNO5590
005587  0000  0010  00587  DC  0  SENSE ILSW  PMNO5590
005588  0000  0010  00588  DC  0  SENSE ILSW  PMNO5590
005589  0000  0010  00589  DC  0  SENSE ILSW  PMNO5590
005590  0000  0010  00590  DC  0  SENSE ILSW  PMNO5590
005591  0000  0010  00591  DC  0  SENSE ILSW  PMNO5590
005592  0000  0010  00592  DC  0  SENSE ILSW  PMNO5590
005593  0000  0010  00593  DC  0  SENSE ILSW  PMNO5590
005594  0000  0010  00594  DC  0  SENSE ILSW  PMNO5590
005595  0000  0010  00595  DC  0  SENSE ILSW  PMNO5590
005596  0000  0010  00596  DC  0  SENSE ILSW  PMNO5590
005597  0000  0010  00597  DC  0  SENSE ILSW  PMNO5590
005598  0000  0010  00598  DC  0  SENSE ILSW  PMNO5590
005599  0000  0010  00599  DC  0  SENSE ILSW  PMNO5590
005600  0000  0010  00600  DC  0  SENSE ILSW  PMNO5590
005601  0000  0010  00601  DC  0  SENSE ILSW  PMNO5590
005602  0000  0010  00602  DC  0  SENSE ILSW  PMNO5590
005603  0000  0010  00603  DC  0  SENSE ILSW  PMNO5590
005604  0000  0010  00604  DC  0  SENSE ILSW  PMNO5590
005605  0000  0010  00605  DC  0  SENSE ILSW  PMNO5590
005606  0000  0010  00606  DC  0  SENSE ILSW  PMNO5590
005607  0000  0010  00607  DC  0  SENSE ILSW  PMNO5590
005608  0000  0010  00608  DC  0  SENSE ILSW  PMNO5590
005609  0000  0010  00609  DC  0  SENSE ILSW  PMNO5590
005610  0000  0010  00610  DC  0  SENSE ILSW  PMNO5590
005611  0000  0010  00611  DC  0  SENSE ILSW  PMNO5590
005612  0000  0010  00612  DC  0  SENSE ILSW  PMNO5590
005613  0000  0010  00613  DC  0  SENSE ILSW  PMNO5590
005614  0000  0010  00614  DC  0  SENSE ILSW  PMNO5590
005615  0000  0010  00615  DC  0  SENSE ILSW  PMNO5590
005616  0000  0010  00616  DC  0  SENSE ILSW  PMNO5590

006010  0000  0010  006010  DC  0  SENSE ILSW  PMNO5590
006011  0000  0010  006011  DC  0  SENSE ILSW  PMNO5590
006012  0000  0010  006012  DC  0  SENSE ILSW  PMNO5590
006013  0000  0010  006013  DC  0  SENSE ILSW  PMNO5590
006014  0000  0010  006014  DC  0  SENSE ILSW  PMNO5590
006015  0000  0010  006015  DC  0  SENSE ILSW  PMNO5590
006016  0000  0010  006016  DC  0  SENSE ILSW  PMNO5590

Resident Monitor (Including Table of Equivalences) G-7
Resident Monitor Listing

0075 * EQUIVALENCES FOR ABSOLUTE SECTOR ADDRESSES PMN09660
0076 *
0000 0077 * LOCAL EQU 0 ADDR OF SCRT WTH ID,DEF CYL ADR PMN09680
0001 0078 * OCM EQU 1 ADDR OF SCRT CONTAINING OCM PMN09690
0002 0079 * REAC EQU 2 ADDR INP SCRT CONTAINING RES IMGE PMN09700
0003 0080 * SLET EQU 3 ADDR OF SCRT CONTAINING SLT PMN09710
0006 0081 * RSTL EQU 6 ADDR OF SCRT CONTAINING RSLT TBL PMN09720
0007 0082 * SVCG EQU 7 ADDR OF SCRT CONTAINING PAGE HDR PMN09730
0008 0083 * SOSY EQU 2 ADDR OF SCRT W/COLD START PRG PMN09740
0084 *
0085 * EQUIVALENCES FOR THE CORE IMAGE HEADER PMN09750
0086 *
0000 0087 * XEQG EQU 0 RLTV ADDR OF CORE LOAD EXEC ADDR PMN09780
0001 0088 * XCOM EQU 1 RLTV ADDR OF WD CAT OF COMMON PMN09790
0002 0089 * OREK EQU 2 RLTV ADDR OF DISK I/O INDICATOR PMN09800
0003 0089 * FREL EQU 3 RLTV ADDR OF NO. FILES DEFINED PMN09810
0004 0091 * XWCT EQU 4 RLTV ADDR OF WD CRT OF CI HEADER PMN09820
0005 0092 * XLCT EQU 5 SCRT CNT OF FILES IN WK STORAGE PMN09830
0006 0093 * XLEA EQU 6 RLTV ADDR OF LOAD ADDR CORE LOAD PMN09840
0007 0094 * XSCT EQU 7 RLTV ADDR DISK/I/DK CNTRL CRTL PMN09850
0008 0095 * YVWC EQU 8 RLTV ADDR OF WD CRT OF TV PMN09860
0009 0096 * WCNT EQU 9 RLTV ADDR OF WD CRT OF CORE LOAD PMN09870
000A 0097 * RDCW EQU 10 RLTV ADDR OF EXEC SETTING OF X3 PMN09880
0008 0098 * XTVX EQU 11 RLTV ADDR OF 1ST WD OF TV PMN09890
0011 0099 * XLSA EQU 17 RLTV ADDR OF 1ST WD OF IB4 PMN09900
0014 009A * XWLY EQU 26 RLTV ADDR OF LOCAL/SOCIAL SWITCH PMN09910
0018 009B * CORE EQU 27 CORE SIZE OF BUILDING SYST 2-10 PMN09920
0010 0100 * XEND EQU 29 RLTV ADDR OF LAST WD OF CI HDR PMN09930
0013 0101 * XCOM EQU 0 RLTV ADDR OF LET/FLET ENTRY NAME PMN09940
0014 0102 * LET/LY EQU 1 RLTV ADDR OF LET/FLET ENTRY DBCT PMN09950
0015 0103 * LFNM EQU 0 RLTV ADDR OF LET/FLET ENTRY NAME PMN09960
0016 0104 * EQUIVALENCES FOR LET/FLET PMN09970
0019 0105 * LFTR EQU 5 WORD CNT OF LET/FLET ENTRY PMN09980
0020 0106 * LFPN EQU 3 NO OF WDTS PER LET/FLET ENTRY PMN09990
0021 0107 * SCTR EQU 0 RLTV ADDR OF LET/FLET SCTR NO. PMN10000
0022 0108 * UXBF EQU 1 RLTV ADDR OF SCTR ADDR OF UA/FA PMN10010
0023 0109 * WOSA EQU 3 RLTV ADDR OF WOS AVAIL IN SCTR PMN10020
0024 0110 * XTQX EQU 4 RLTV ADDR OF ADDR NEXT SCTR PMN10030
0025 0111 * XPFM EQU 0 RLTV ADDR DISK PMN10040
0026 0112 * XSFV EQU 2 RLTV ADDR OF LET/FLET ENTRY DBCT PMN10050
0027 0113 * XBSL EQU 0 RLTV ADDR OF ADVANCE ENTRY PMN10060
0028 0114 * XFRX EQU 0 RLTV ADDR OF ADVANCE ENTRY PMN10070
0033 0115 * MISCELLANEOUS EQUIVALENCES PMN10080
0034 0116 *
0005 0117 * ISTY EQU 51 ISS NO. ADJUSTMENT FACTOR 2-1 PMN10090
0006 0118 * XDOC EQU 5 MAX NO. DRIVES SUPPORTED PMN10100
0007 0119 * COMX EQU 896 LOW COMMON LIMIT FOR DISKZ PMN10110
0008 0120 * OFDL EQU 1216 LOW COMMON LIMIT FOR DISK1 PMN10120
0009 0121 * CDRX EQU 1536 LOW COMMON LIMIT OF DISKX PMN10130
0011 0122 * TCNT EQU 17 NO. TRIES BEFORE DISK ERROR PMN10140
0012 0123 * DTEX EQU 32000+7 DISK I/O INTERRUPT ENTRY PT PMN10150
0013 0124 * DPIK EQU 32000+4 DISK I/O INTERRUPT ENTRY PMN10160
0014 0125 * SCIR EQU 16 CIB SECTOR COUNT 2-2 PMN10170
0015 0126 * OCRX EQU 3 HIGH COMMON SECTOR COUNT 2-2 PMN10180
0016 0127 * MDOS EQU 4086 SIZE OF MINIMUM CORE 2-2 PMN10190
0017 0128 * Y EQU 127 PMN10200
0020 0129 * PMN10210
0029 0130 * CMDF EQU 4 RLTV ADDR CARTRIDGE ID 2-2 PMN10220
002A 0131 * XCPY EQU 5 RLTV ADDR COPY INDICATOR 2-2 PMN10230
002C 0132 * OCRX EQU 1 RLTV ADDR DISK TYPE IND 2-2 PMN10240
002F 0133 * DTRP EQU 8 RLTV ADDR DISK TYPE IND 2-2 PMN10250

COLD START PROGRAM

0135 0135 ****************************************************************** PMN10260
0136 *
0137 *
0138 *
0139 *
0140 *
0141 *
0142 *
0143 *
0144 *
0145 *
0146 *
0147 *
0148 *
0149 *
0150 *
0151 *
0152 *
0153 *
0154 *
0155 *
0156 *
0157 *
0158 *

By TNL GN34-0353

As Updated October 22, 1976

Page of GC26-3717-9, 10

By TNL GN34-0353
Resident Monitor Listing

01059 *OUTPUT - PMN10500
01060 * THE RESIDENT IMAGE IS READ INTO CORE FROM PMN10510
01061 * THE DISK - PMN10520
01062 * IN COMMA- PMN10530
01063 * $ACDE - PMN10540
01064 * $CIBA-1 - PMN10550
01065 * $CION - PMN10560
01066 * $CYLN - PMN10570
01067 * $DBSY - PMN10580
01068 * $DCT - PMN10590
01069 *
01070 *EXTERNAL REFERENCES - PMN10610
01071 * DZ000 SUBROUTINE TO PERFORM DISK I/O. PMN10620
01072 *
01073 *EXITS - PMN10640
01074 * THE ONLY EXIT IS TO THE AUXILIARY SUPERVISOR PMN10650
01075 * AS FOLLOWS- PMN10660
01076 *
01077 * $B$ SCUMP - PMN10670
01078 *
01079 *TABLES/WORK AREAS - N/A PMN10690
01080 *
01081 *ATTRIBUTES - PMN10720
01082 * THIS PROGRAM IS NOT NATURALLY RELOCATABLE. PMN10730
01083 *
01084 *NOTES - PMN10750
01085 * DISK ERRORS RESULT IN A WAIT AT SPST2. PMN10760
01086 *************************************************** PMN10770
01088 *
01089 * READ THE RESIDENT IMAGE INTO CORE PMN10800
01100 *
01101 01E0 0 617F 01102 LOX
01103 01E1 0 0824 01104 LOD	 CR920	 SET UP WORD COUNT AND SCTR PMN10830
01105 01E2 00 DC000004 01106 CR010 STD 	 L	 SCIBA-1	 *ADDR OF RESIDENT IMAGE PMN10840
01107 01E4 0 D125 01108 STO	 I	 SCCYL-Y 	 *INITIALIZE DEF CYL NO. 	 1 PMN10850
01109 01E5 0 0184 01110 LD	 1 3-Y	 FETCH LOG DRIVE 0 AREA CODE PMN10860
01111 01E6 0 0120 01112 STO	 1 SACDE-Y 	 *AND STORE IT IN COMMA PMN10870
01113 01E7 0 DO1F 01114 STO	 CR920+1 	 SAVE THE AREA CODE PMN10880
01115 01E8 0 0156 01116 LD	 1 CZ000-2-27-Y FETCH AND SAVE THE PMN10890
01117 01E9 0 00E1 01118 STO	 SCION	 *CARTRIDGE ID PMN10900
01119 (ILEA 00 660001FE 01120 LOX	 L2 CR020 	 SET UP TEMPORARY 	 2-11 PMN10920
01121 01EE 00 6E00000A 01122 STX	 L2	 SLEV2	 *ILSO2	 2-11 PMN10940
01123 01EE 0 COF4 01124 LO	 CR920+1 	 FETCH CORE ADDR OF RESIDENT PMN10950
01125 01EF 0 1890 01126 SRT	 16	 *IMAGE AND PUT 	 IN EXTENSION PMN10960
01127 01F0 0 016E 01128 STO	 1	 SDBSY-Y 	 CLEAR DISK BUSY INDICATOR PMN10970
01129 01F1 0 0118 01130 STO	 I	 SCYLN-Y 	 INITIALIZE 	 ARM	 POSITION PMN10980
01131 01F2 0 4173 01132 BSI	 1 DZ000-Y 	 FETCH RESIDENT IMAGE PMN10990
01133 01F3 0 3000 01134 WAIT	 WAIT OUT THE INTERRUPT PMN11000
01135 01F4 0 1810 01136 SRA	 16 PMN11010
01137 01F5 0 0183 01138 STO	 1 SIOCT-Y 	 CLEAR 10CS COUNTER PMN11020
01139 01F6 0 C800 01140 LCD	 CR910 PMN11030
01141 01F7 0 0985 01142 STD	 1	 SCIBA-1-Y *FOR SAVING CORE ON THE CIB PMN11040
01143 01F8 0 COOE 01144 LC	 CR920+1 	 FETCH AREA CODE PMN11050
01145 01F9 0 0120 01146 STO	 1	 SACDE-Y 	 RESET AREA CODE PMN11060
01147 01FA 0 COOD 01148 LO	 CR905	 INITIALIZE WO 	 ZERO TO PMN11070
01149 01FB 0 0000 01150 BSS	 E	 0	 ASSURE EVEN BOUNDARY 2-11 PMN11080
01151 01FC 0 1810 01152 LOX	 L2 CR020,-1 	 2-11 PMN11090
01153 01FD 0 0000 01154 DC 	 -1 M Compatible Processing PMN11100
01155 01FE 0 0000 01156 CR910 DC ++
01157 01FF 0 4178 01158 BSI	 1 DZ010-Y AR TO SERVICE INTERRUPT 2-11 PMN11140
01159 0200 0 76FF01FE 01160 MXD L CR020,-1 PMN11144
01161 0202 0 4C0001FE 01162 BSO	 C 1 CR020 RETURN 2-11 PMN11148
01163 01130 *
01164 01131 *
01165 01132 *
01166 0204 0 0000 01167 BSS E 0 ASSURE EVEN BOUNDARY 2-11 PMN11190
01168 0204 0 0000 01169 CR910 DC 0 WO CNT,SCR ADDR OF 2-5 PMN11200
01170 0205 0 0007 01171 DC *NON WAREHOUSE WRITE TO DISK PMN11210
01172 0206 0 0008 01173 CR920 DC $DBSY-$CH12 WO CNT AND SCR PMN11220
01174 0207 0 0002 01175 DC *SCI40 ADDR OF RESIDENT IMAGE PMN11230
01176 0208 0 70FF 01177 CH05 MXD =-1 TO BE PUT AT ADDR 0000 2-11 PMN11250
01178 0209 0 0009 01179 BSS /0212= PATCH AREA 2-11 PMN11292
01180 0212 0 0140 END *
### Resident Monitor Listing

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
<th>REL DEPN</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR010</td>
<td>01E2</td>
<td>01093</td>
<td>01102,R</td>
</tr>
<tr>
<td>CR020</td>
<td>01FE</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>CR030</td>
<td>0115</td>
<td>0117</td>
<td>0113,R</td>
</tr>
<tr>
<td>CR040</td>
<td>0181</td>
<td>01136</td>
<td>01104,R</td>
</tr>
<tr>
<td>CR050</td>
<td>01092</td>
<td>01128,M</td>
<td>01109,R</td>
</tr>
<tr>
<td>CR060</td>
<td>0100</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>CR070</td>
<td>0102</td>
<td>01138</td>
<td>01104,R</td>
</tr>
<tr>
<td>CR905</td>
<td>0208</td>
<td>01139</td>
<td>0113,R</td>
</tr>
<tr>
<td>CR910</td>
<td>0204</td>
<td>00117</td>
<td>01109,R</td>
</tr>
<tr>
<td>CR920</td>
<td>0206</td>
<td>01095,M</td>
<td>01115,R</td>
</tr>
<tr>
<td>DZ000</td>
<td>00F2</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>DZ010</td>
<td>00F7</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>DZ020</td>
<td>00F8</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>DZ030</td>
<td>00F9</td>
<td>01126</td>
<td>01100,R</td>
</tr>
<tr>
<td>DZ040</td>
<td>0100</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ050</td>
<td>0101</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ060</td>
<td>0102</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ070</td>
<td>0103</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ080</td>
<td>0104</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ090</td>
<td>0105</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ100</td>
<td>0106</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ110</td>
<td>0107</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ120</td>
<td>0108</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ130</td>
<td>0109</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ140</td>
<td>0110</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ150</td>
<td>0111</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ160</td>
<td>0112</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ170</td>
<td>0113</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ180</td>
<td>0114</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ190</td>
<td>0115</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ200</td>
<td>0116</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ210</td>
<td>0117</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ220</td>
<td>0118</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ230</td>
<td>0119</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ240</td>
<td>0120</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ250</td>
<td>0121</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ260</td>
<td>0122</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ270</td>
<td>0123</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ280</td>
<td>0124</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ290</td>
<td>0125</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ300</td>
<td>0126</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ310</td>
<td>0127</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ320</td>
<td>0128</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ330</td>
<td>0129</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ340</td>
<td>0130</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ350</td>
<td>0131</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ360</td>
<td>0132</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ370</td>
<td>0133</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ380</td>
<td>0134</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ390</td>
<td>0135</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ400</td>
<td>0136</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ410</td>
<td>0137</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ420</td>
<td>0138</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ430</td>
<td>0139</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ440</td>
<td>0140</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ450</td>
<td>0141</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ460</td>
<td>0142</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ470</td>
<td>0143</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ480</td>
<td>0144</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ490</td>
<td>0145</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>DZ500</td>
<td>0146</td>
<td>00118</td>
<td>00110,R</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>VALUE</td>
<td>REL</td>
<td>DEFN</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>S02Z1N</td>
<td>0076</td>
<td>0</td>
<td>00376</td>
</tr>
<tr>
<td>SEXIT</td>
<td>0038</td>
<td>0</td>
<td>00304</td>
</tr>
<tr>
<td>$FLSH</td>
<td>0071</td>
<td>0</td>
<td>00370</td>
</tr>
<tr>
<td>$FGDR</td>
<td>0095</td>
<td>0</td>
<td>00428</td>
</tr>
<tr>
<td>$GCOMM</td>
<td>0063</td>
<td>0</td>
<td>00353</td>
</tr>
<tr>
<td>$GRIN</td>
<td>0064</td>
<td>0</td>
<td>00354</td>
</tr>
<tr>
<td>$HASH</td>
<td>0014</td>
<td>0</td>
<td>00275</td>
</tr>
<tr>
<td>$18SY</td>
<td>0020</td>
<td>0</td>
<td>00274</td>
</tr>
<tr>
<td>$IBS2</td>
<td>0004</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$IBT4</td>
<td>0013</td>
<td>0</td>
<td>00274</td>
</tr>
<tr>
<td>$1880</td>
<td>0013</td>
<td>0</td>
<td>00274</td>
</tr>
<tr>
<td>$1200</td>
<td>0004</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1290</td>
<td>0004</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1400</td>
<td>0004</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1405</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1410</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1420</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1425</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1430</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1435</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1440</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1445</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1450</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1455</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1460</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1465</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1470</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1475</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1480</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1490</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1495</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1500</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1510</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1520</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1530</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1540</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1550</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1560</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>$1570</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1580</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1590</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1600</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1610</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1620</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1630</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1640</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1650</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1660</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1670</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1680</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1690</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1700</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1710</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1720</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1730</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1740</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1750</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1760</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1770</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1780</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1790</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1800</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1810</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1820</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1830</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1840</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1850</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1860</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1870</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1880</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1890</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1900</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1910</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1920</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1930</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1940</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1950</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1960</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1970</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1980</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>S1990</td>
<td>0005</td>
<td>0</td>
<td>00969</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>VALUE</td>
<td>REL</td>
<td>DEFN</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>#OBCT</td>
<td>0060</td>
<td>0</td>
<td>00897</td>
</tr>
<tr>
<td>#OCW</td>
<td>0018</td>
<td>0</td>
<td>00914</td>
</tr>
<tr>
<td>#ENT</td>
<td>0021</td>
<td>0</td>
<td>00923</td>
</tr>
<tr>
<td>#ENTY</td>
<td>0010</td>
<td>0</td>
<td>00907</td>
</tr>
<tr>
<td>#FCNT</td>
<td>0007</td>
<td>0</td>
<td>00898</td>
</tr>
<tr>
<td>#FHOL</td>
<td>0014</td>
<td>0</td>
<td>00910</td>
</tr>
<tr>
<td>#FLET</td>
<td>0048</td>
<td>0</td>
<td>00932</td>
</tr>
<tr>
<td>#FMAT</td>
<td>0046</td>
<td>0</td>
<td>00931</td>
</tr>
<tr>
<td>#FPAC</td>
<td>0020</td>
<td>0</td>
<td>00926</td>
</tr>
<tr>
<td>#FSZE</td>
<td>0015</td>
<td>0</td>
<td>00911</td>
</tr>
<tr>
<td>#GCNT</td>
<td>001E</td>
<td>0</td>
<td>00920</td>
</tr>
<tr>
<td>#GRPH</td>
<td>0010</td>
<td>0</td>
<td>00919</td>
</tr>
<tr>
<td>#JBSW</td>
<td>0009</td>
<td>0</td>
<td>00900</td>
</tr>
<tr>
<td>#LCNT</td>
<td>0008</td>
<td>0</td>
<td>00902</td>
</tr>
<tr>
<td>#LCSW</td>
<td>001F</td>
<td>0</td>
<td>00921</td>
</tr>
<tr>
<td>#MDF1</td>
<td>0000</td>
<td>0</td>
<td>00904</td>
</tr>
<tr>
<td>#MDF2</td>
<td>000E</td>
<td>0</td>
<td>00905</td>
</tr>
<tr>
<td>#MPSW</td>
<td>000C</td>
<td>0</td>
<td>00903</td>
</tr>
<tr>
<td>#NAME</td>
<td>0004</td>
<td>0</td>
<td>00896</td>
</tr>
<tr>
<td>#NCT</td>
<td>000F</td>
<td>0</td>
<td>00906</td>
</tr>
<tr>
<td>#PCID</td>
<td>0032</td>
<td>0</td>
<td>00927</td>
</tr>
<tr>
<td>#PIDO</td>
<td>0019</td>
<td>0</td>
<td>00915</td>
</tr>
<tr>
<td>#PRTB</td>
<td>001A</td>
<td>0</td>
<td>00916</td>
</tr>
<tr>
<td>#RP67</td>
<td>0011</td>
<td>0</td>
<td>00908</td>
</tr>
<tr>
<td>#SCRA</td>
<td>0041</td>
<td>0</td>
<td>00930</td>
</tr>
<tr>
<td>#SYSC</td>
<td>0008</td>
<td>0</td>
<td>00899</td>
</tr>
<tr>
<td>#THDC</td>
<td>0012</td>
<td>0</td>
<td>00909</td>
</tr>
<tr>
<td>#UHOL</td>
<td>0016</td>
<td>0</td>
<td>00912</td>
</tr>
<tr>
<td>#ULET</td>
<td>0030</td>
<td>0</td>
<td>00933</td>
</tr>
<tr>
<td>#USZE</td>
<td>0017</td>
<td>0</td>
<td>00913</td>
</tr>
<tr>
<td>#MSCT</td>
<td>0059</td>
<td>0</td>
<td>00934</td>
</tr>
<tr>
<td>#XSW</td>
<td>0020</td>
<td>0</td>
<td>00922</td>
</tr>
<tr>
<td>#BLCT</td>
<td>0002</td>
<td>0</td>
<td>01013</td>
</tr>
<tr>
<td>#CDCV</td>
<td>0093</td>
<td>0</td>
<td>00950</td>
</tr>
<tr>
<td>#CICK</td>
<td>0004</td>
<td>0</td>
<td>011030</td>
</tr>
<tr>
<td>JCIM</td>
<td>0004</td>
<td>0</td>
<td>01030</td>
</tr>
<tr>
<td>#CIL2</td>
<td>00A1</td>
<td>0</td>
<td>00957</td>
</tr>
<tr>
<td>#CLBO</td>
<td>0078</td>
<td>0</td>
<td>00942</td>
</tr>
<tr>
<td>#CMON</td>
<td>0001</td>
<td>0</td>
<td>00988</td>
</tr>
<tr>
<td>#COMZ</td>
<td>0380</td>
<td>0</td>
<td>00109</td>
</tr>
<tr>
<td>#CDM2</td>
<td>0600</td>
<td>0</td>
<td>00121</td>
</tr>
<tr>
<td>#CPYF</td>
<td>0005</td>
<td>0</td>
<td>00131</td>
</tr>
<tr>
<td>#CORE</td>
<td>001E</td>
<td>0</td>
<td>00101</td>
</tr>
<tr>
<td>#CPR</td>
<td>009E</td>
<td>0</td>
<td>00945</td>
</tr>
<tr>
<td>#DCMP</td>
<td>0001</td>
<td>0</td>
<td>00978</td>
</tr>
<tr>
<td>#DCTB</td>
<td>0001</td>
<td>0</td>
<td>00132</td>
</tr>
<tr>
<td>#DKP</td>
<td>00F9</td>
<td>0</td>
<td>00123</td>
</tr>
<tr>
<td>#DKP</td>
<td>00F7</td>
<td>0</td>
<td>00124</td>
</tr>
<tr>
<td>#DNID</td>
<td>0098</td>
<td>0</td>
<td>00955</td>
</tr>
<tr>
<td>#DREC</td>
<td>0002</td>
<td>0</td>
<td>00989</td>
</tr>
<tr>
<td>#DYP</td>
<td>0008</td>
<td>0</td>
<td>001033</td>
</tr>
<tr>
<td>#DJP</td>
<td>0096</td>
<td>0</td>
<td>00953</td>
</tr>
<tr>
<td>#DIID</td>
<td>0097</td>
<td>0</td>
<td>00954</td>
</tr>
<tr>
<td>#FILE</td>
<td>0003</td>
<td>0</td>
<td>00990</td>
</tr>
<tr>
<td>#HCI8</td>
<td>0033</td>
<td>0</td>
<td>01026</td>
</tr>
<tr>
<td>#HENG</td>
<td>0007</td>
<td>0</td>
<td>00982</td>
</tr>
<tr>
<td>#HEND</td>
<td>0010</td>
<td>0</td>
<td>01002</td>
</tr>
<tr>
<td>#HCCT</td>
<td>0004</td>
<td>0</td>
<td>00991</td>
</tr>
<tr>
<td>#IDAD</td>
<td>0000</td>
<td>0</td>
<td>00977</td>
</tr>
<tr>
<td>#IL54</td>
<td>0011</td>
<td>0</td>
<td>00999</td>
</tr>
<tr>
<td>#ISTV</td>
<td>0033</td>
<td>0</td>
<td>01017</td>
</tr>
<tr>
<td>#ITVX</td>
<td>0008</td>
<td>0</td>
<td>00998</td>
</tr>
<tr>
<td>#KBCP</td>
<td>0092</td>
<td>0</td>
<td>00649</td>
</tr>
<tr>
<td>#KBCV</td>
<td>0095</td>
<td>0</td>
<td>00952</td>
</tr>
<tr>
<td>#LOAD</td>
<td>0006</td>
<td>0</td>
<td>00993</td>
</tr>
<tr>
<td>#LFIN</td>
<td>0003</td>
<td>0</td>
<td>01007</td>
</tr>
<tr>
<td>#LPHM</td>
<td>0005</td>
<td>0</td>
<td>01006</td>
</tr>
<tr>
<td>#LFNM</td>
<td>0000</td>
<td>0</td>
<td>01012</td>
</tr>
<tr>
<td>#LSCD</td>
<td>0005</td>
<td>0</td>
<td>00992</td>
</tr>
<tr>
<td>#MCID</td>
<td>0000</td>
<td>0</td>
<td>01027</td>
</tr>
<tr>
<td>#MCRA</td>
<td>006E</td>
<td>0</td>
<td>00939</td>
</tr>
<tr>
<td>#MDCR</td>
<td>0005</td>
<td>0</td>
<td>01018</td>
</tr>
<tr>
<td>#NEXT</td>
<td>0004</td>
<td>0</td>
<td>01011</td>
</tr>
<tr>
<td>#CVSW</td>
<td>001A</td>
<td>0</td>
<td>01000</td>
</tr>
<tr>
<td>#PTCV</td>
<td>0094</td>
<td>0</td>
<td>00991</td>
</tr>
<tr>
<td>#RIAD</td>
<td>0002</td>
<td>0</td>
<td>00979</td>
</tr>
<tr>
<td>#RTBL</td>
<td>0030</td>
<td>0</td>
<td>00981</td>
</tr>
<tr>
<td>#SC18</td>
<td>0010</td>
<td>0</td>
<td>01025</td>
</tr>
<tr>
<td>#SCIN</td>
<td>0000</td>
<td>0</td>
<td>01008</td>
</tr>
<tr>
<td>#SLET</td>
<td>0003</td>
<td>0</td>
<td>00980</td>
</tr>
<tr>
<td>#STRT</td>
<td>0000</td>
<td>0</td>
<td>00983</td>
</tr>
<tr>
<td>#SUP6</td>
<td>0073</td>
<td>0</td>
<td>00940</td>
</tr>
<tr>
<td>#SUPT</td>
<td>0074</td>
<td>0</td>
<td>00941</td>
</tr>
<tr>
<td>#TCMT</td>
<td>0011</td>
<td>0</td>
<td>01022</td>
</tr>
<tr>
<td>#TVHC</td>
<td>0008</td>
<td>0</td>
<td>00995</td>
</tr>
<tr>
<td>#VAPX</td>
<td>0001</td>
<td>0</td>
<td>01009</td>
</tr>
</tbody>
</table>
### CROSS-REFERENCE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
<th>REL</th>
<th>DEFN</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>'WONT</td>
<td>0009</td>
<td>0</td>
<td>00996</td>
<td></td>
</tr>
<tr>
<td>'WDSA</td>
<td>0003</td>
<td>0</td>
<td>01010</td>
<td></td>
</tr>
<tr>
<td>'XCTL</td>
<td>0007</td>
<td>0</td>
<td>00994</td>
<td></td>
</tr>
<tr>
<td>'XECO</td>
<td>0000</td>
<td>0</td>
<td>00997</td>
<td></td>
</tr>
<tr>
<td>'XR3X</td>
<td>000A</td>
<td>0</td>
<td>00997</td>
<td></td>
</tr>
<tr>
<td>'1132</td>
<td>0080</td>
<td>0</td>
<td>00944</td>
<td></td>
</tr>
<tr>
<td>'1134</td>
<td>0091</td>
<td>0</td>
<td>00946</td>
<td></td>
</tr>
<tr>
<td>'1403</td>
<td>008C</td>
<td>0</td>
<td>00945</td>
<td></td>
</tr>
<tr>
<td>'1442</td>
<td>0090</td>
<td>0</td>
<td>00947</td>
<td></td>
</tr>
<tr>
<td>'2501</td>
<td>008F</td>
<td>0</td>
<td>00946</td>
<td></td>
</tr>
</tbody>
</table>

### ERROR STATEMENT LINE NUMBERS

00619
1000. OVERFLOW SECTORS SPECIFIED
2000 OVERFLOW SECTORS REQUIRED
3000 SYMBOLS DEFINED
4000 ERROR(S) AND 001 WARNING(S) FLAGGED IN ABOVE ASSEMBLY
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 READER.

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 1
LOG DRIVE CART SPEC CART AVALIB PPM DRIVE C000 OED0 OEDO C000

// DUP
*STCREGATA W5 UA FILEA Z CART IO OEDO DB ADDR ZEA0 DB CNT OC20

// = IBM 1130 FORTRAN SAMPLE PROGRAM

// FOR
*CNT WORD INTERGERS
*ECCS(DISK,1132 PRINTER)
*ECCS(CARD)
*LIST ALL
C IBM 1130 FORTRAN SAMPLE PROGRAM
C SIMULTANEOUS EQUATION PROGRAM
C INTEGER V1,V2,V3
DIMENSION A(10,10),X(10,10),U(126)
DEFINE FILE 101 (1,100,U,V1),102(1,100,U,V2),103(1,100,U,V3)
301 FORMAT (1H1,20X,15HINCOMPATIBILITY)
302 FORMAT (1H1,20X,41HMORE EQUATIONS THAN UNKNOWNS-NC SOLUTIONS)
303 FORMAT (1H1,20X,15HSOLUTION MATRIX)
304 FORMAT (1H1,20X,8HMATRIX A)
305 FORMAT (1H1,20X,108A-INVESWEIL ELEMENT IS ZERO)
306 FORMAT (1H1,20X,24MDIAGONAL ELEMENT IS ZERO)
307 FORMAT (1H1,20X,2M0)
L=3
READ (P,10)
10 FORMAT (80HSPACE FOR TITLE)
WRITE (41,10)
12 FORMAT (6110 920X)
REAC (P44,12) MI,P2,P1,L2,NI,N2
C MI = NO. OF ROWS OF A
C M2 = NO. OF COLS OF A
C LI = NO. OF ROWS OF X
C L2 = NO. OF COLS OF X
C NI = NO. OF ROWS OF B
C N2 = NO. OF COLS OF B
C 13 FORMAT (7F10.4,10X)
17 FORMAT (1OF10.4)
19 IF (N1-166,64,63)
64 IF (L1-166,65,63)
65 IF (L2-166,66,63)
66 IF (M1-166,67,63)
63 WRITE (L,301)
GC TO 2
14 WRITE (L,305)
DO 70 I=1,N
REAC (P,13) (AIJ,J=1,N)
WRITE (L,17) (AIJ,J=1,N)
WRITE (1011,14) (AIJ,J=1,N)
70 CONTINUE
89 FORMAT (F10.4,7DX)
WRITE (L,306)
REAC (P,89) (BI,J=1,N)
WRITE (L,89) (BI,J=1,N)
WRITE (1021,13) (BI,J=1,N)
C INVERSION OF A
C DO 120 K=1,N
D=AI(K,K)
IF (D=0.0) GO TO 120
A(K,K)=1.0
DO 60 J=1,N
60 AI(K,J)/=A(K,K)
DO 120 J=1,N
120 AI(J,J)=AI(J,J)-(D*AI(K,J))
FORTRAN Sample Program
run on 4K

C
BACK SOLUTION
C
130 IX=1-1
DO 180 K=1,IK
11=1
DC 180 I=1,IK
D=AK(1)
A(K)=0.0
DO 180 J=1,N
180 A(K,J)=A(K,J)+(DC(AI(1,J)))
GC TO 202
200 WRITE (L,308)
GC TC 2
202 WRITE (L,307)
DO 201 I=1,N
WRITE (L,17) (AI(1,J), J=1,N)
WRITE (10*17) (AI(1,J), J=1,N)
201 CONTINUE
DO 21 I=1,N
X(I)=0.0
DO 21 K=1,IK
21 X(I)=X(I)+AI(1,K)
WRITE (L,304) FRI
WRITE (1,303) YI(11, 11)
2 CALL EXIT
END

VARIABLE ALLOCATIONS
AIR )=000C-0016
XRI )=000F-000E
B1R )=0108-000E
DIR )=010C-000E
VRI )=0100-0010
V2I )=0101-0010

STATEMENT ALLOCATIONS
301 =010E 302 =0108 303 =0100 304 =0107 305 =0106 306 =0105 307 =0104 308 =0103 309 =0102 310 =0101 311 =0100
13 =010E 17 =0108 19 =0100 20 =0106 21 =0105 22 =0104 23 =0103 24 =0102 25 =0101 26 =0100
14 =010E 20 =0108 21 =0100 22 =0106 23 =0105 24 =0104 25 =0103 26 =0102 27 =0101 28 =0100
201 =010E 21 =0108 22 =0106 23 =0105 24 =0104 25 =0103 26 =0102 27 =0101 28 =0100

FEATURES SUPPORTED
CCE CCRO INTERGERS
CALLEC SUBPRCGRAMS
FACDX FMPVX FCIV FLD FLDX FSTO FSTOX FSBRX CARDZ PRNZ SRED SWRT SCOMP SFIO SF0X
SIDI SUBSC SDIFC SDWAR SDCOM SDFX

REAL CONSTANTS
+100000DE 01=0204
+000000DE 00=0204

INTEGER CONSTANTS
2=0208 3=0209 4=0210 101=02C8 102=020C 103=020D

CCRE REQUIREMENTS FOR
COPPCH O VARIABLES 516 PROGRAM 874

END OF COPPILATION
FORTRAN Sample Program
run on 4K

IBM 1130 FORTRAN SAMPLE PROGRAM

// XEC L 2
*LOCAL,FLCAT,PARC,IFIX,PAUSE,HOLEZ
*FILES(103,FILEA)
FILES ALLOCATIONS
  1C3 02EA OC01 OECD FILEA
  101 000C OC01 OECD 02EC
  102 0001 OC01 OECD 02EC

SIZE RATIONS ALLOCATIONS
  R 40 03BF (HEX) ADDITIONAL CORE REQUIRED
  R 43 0124 (HEX) ARITH/FUNC SCAL W/D CAT
  R 44 0622 (HEX) F/L/C, I/O SCAL W/D CNT
  R 45 0768 (HEX) DISK F/L/C SCAL W/D CAT
  R 41 0004 (HEX) W/C UNUSEC BY CORE LCAD

LIBR TRANSFER VECTOR
  XPCS 094A SCAL 1
  EBC18 0F51 SCAL 2
  HC17B 0F1E SCAL 2
  GETAC 0E02 SCAL 2
  HORM 0E0D
  FAD0X 0955 SCAL 1
  FSB0X 092C SCAL 1
  PARYX 0978 SCAL 1
  FCIV 0846 SCAL 1
  FST0X 076A
  FL0X 0788
  SGGP 0978 SCAL 3
  SDFX 0833 SCAL 3
  SDRT 0901 SCAL 3
  SIC0X 09A6 SCAL 2
  SUBST 07A2
  SIC1 094A SCAL 2
  SCMP 0983 SCAL 2
  SWRT 08A2 SCAL 2
  SNED 08A7 SCAL 2
  FST0 0770
  FLD 078C
  PANTZ 0CF8 SCAL 2
  CAR0Z 0C49 SCAL 2
  SPI0 098F SCAL 2
  SCP1C 0960 SCAL 3
  HCLIT 086A LOCAL
  PAUSE 0866 LOCAL
  IFIX 086A LOCAL
  PARC 086A LOCAL
  FLCAT 086A LOCAL
  SYSTEM SUBROUTINES
  ILS04 06C4
  ILS02 08E3
  ILS01 0F56
  ILS00 0F6F
  FIIPL 0804
  04C1 (HEX) IS THE EXECUTION ADDR

A-INVORSE

IBM 1130 FORTRAN SAMPLE PROGRAM

MATRICES

\[
\begin{bmatrix}
4.2150 & -2.1200 & 1.1220 \\
-2.2100 & 3.5050 & -0.6320 \\
1.1220 & -0.3100 & 3.9860
\end{bmatrix}
\]

A-INVORSE

\[
\begin{bmatrix}
0.2919 & 0.0833 & -0.0467 \\
0.1631 & 0.3836 & 0.1110 \\
-0.0203 & 0.1029 & 0.3005
\end{bmatrix}
\]

SOLUTION MATRIX

\[
\begin{bmatrix}
0.9321 & 1.6294 & 0.7429
\end{bmatrix}
\]
FORTRAN Sample Program Run on 8K

// JOB T

LOG DRIVE	CART SPEC	CART AVAIL	PHY DRIVE
0000	2222	2222	0002
V2 M11	ACTUAL 8K CONFIG 8K

// DUP

*STORED ATA WS UA FILE A 2
CART ID 2222	DB ADDR 5380	DB CNT 0020

// > FOR

*ONE WORD INTEGERS
*IOCS(1DISK,1132 PRINTER)
*IOCS(CARD)
*LIST ALL
C 11130 FORTRAN SAMPLE PROGRAM
C SIMULTANEOUS EQUATION PROGRAM

(INTEGER V1,V2,V3
DIMENSION X(10,10),X(10),A1(25)
DEFINE FILE 1011,I10,U,V111011,10,V211011,10,U,V311011,10,U,V3
301 FORMAT (130X41HMORE EQUATIONS THAN UNKNOWNS-NO SOLUTIONS)
302 FORMAT (130X41HPORE UNKNOWNS THAN EQUATIONS-SEVERAL SOLUTIONS)
303 FORMAT (130X41HDIAGONAL ELEMENT IS ZERO)
304 FORMAT (130X41H A-INVERSE)
305 FORMAT (130X41HREAC (M,10)
10 FORMAT(80H SPACE 1'TITLE
12 FORMAT(80H SPACE 2
13 FORMAT(80H SPACE 3
14 FORMAT(80H SPACE 4
15 FORMAT(80H SPACE 5
16 FORMAT(80H SPACE 6
17 FORMAT(80H SPACE 7
18 FORMAT(80H SPACE 8
19 FORMAT(80H SPACE 9
20 FORMAT(80H SPACE 10

Monitor System Sample Programs H-5
FORTRAN Sample Program
run on 8K

VARIABLE ALLOCATIONS
AIR 1=0000-0016
XIR 1=00FO-00DE
BIR 1=01EC-00F2
DIR 1=01EE
VII 1=01F8
V2 1=01F3
V3 1=01F2
M11 1=01FF
M12 1=01F8
KII 1=01FE
K11 1=01FF
K12 1=0200

STATEMENT ALLOCATIONS
301 G02E 302 0218 303 0245 304 0251 305 0256 306 0267 307 0270 308 027A 10 0280 12 0288
13 0289 17 0280 89 02C0 66 0300 65 0306 66 030C 63 0312 11 0318 91 0320 93 0330
14 0334 70 0336 4C 033B 60 039A 80 0416 120 0435 150 0446 250 0464 252 046E
201 0506 21 0520 2 054C

FEATURES SUPPORTED
ONE WORD INTEGERS

CORE REQUIREMENTS FOR
COMMON 0 VARIABLES 516 PROGRAM 874

END OF COMPIlATION
FORTRAN Sample Program
run on BK

// XEQ L 2
LOCAL, FLOAT, FARC, IFIX, PAUSE, HCLE?, FLD

*FILE(103, FILE4)
FILES ALLOCATION
103 053B 0001 2222 FILEA
101 0000 0001 2222 0530
102 0001 0001 2222 0530
STORAGE ALLOCATION
R 41 0C42 (HEX) WDS UNSEC BY CORE LCAD
LINK TRANSFER VECTOR
XMD5 1244
EBCT 1241
MOLT 12C5
GETAD 11C2
NORM 1198
FADDX 111A
FMPYX 10E6
FSTOX 12F4 LOCAL
FLO% 1310 LOCAL
SOCOM 0842
SOFX 07A0
SDMRT 07C8
SUBSC 1064
SID1 082E
SCOMP 0806
SWRT 0A22
SREW 0A27
FSTO 12F8 LOCAL
PRNTZ 0F7C
CAROZ 0ECC
SF10 0843
SOF10 082A
FLD 1314 LOCAL
HOLEZ 12F4 LOCAL
PAUSE 12F4 LOCAL
IFIX 12F4 LOCAL
FARC 12F4 LOCAL
FLOAT 12F4 LOCAL
SYSTEM SUBROUTINES
ILSO4 0083
ILSO2 132C
ILSO1 1345
FLIPR 128E
04C1 (HEX) IS THE EXECUTION ADDR
This page intentionally left blank
2. ASSEMBLER SAMPLE PROGRAM

The core map printed with the assembler sample program is described in Chapter 6, "Programming Tips and Techniques."

output on the principal printer

// JOB

LCG DRIVE CART SPEC CART AVAIL PHY DRIVE
O000 OEDO OEDO 0000

V2 W09 ACTUAL 32K CONFIG 32K

// ASW

*LST

*PRINT SYMBOL TABLE

*********** smasm102
*********** smasm103
*********** smasm104

BEGIN LC 064 INPUT TO THE SQUARE ROOT smasm111

LIBF FLOAT INTEGER TO FLOATING PT. smasm112
CALL FSQR FLOATING PT. SORT. smasm113
LIBF IFIX FLOATING PT. TO INTEGER smasm114
SLA 8 smasm115

* MASK TO BUILD EBCDIC INTEGER smasm116
* RESULT AND EBCDIC BLANK IN WORD1. smasm117

BEGIN LC 64 INPUT TO THE SQUARE ROOT smasm111

DC 0 CONTROL PARAMETER smasm112
DC 0 CONTROL PARAMETER smasm113
DC 0 CONTROL PARAMETER smasm114

DC 26 CHARACTER COUNT smasm115

LIBF TYPE0 TYPE MESSAGE smasm112
DC /2000 CONTROL PARAMETER smasm113
DC /2000 CONTROL PARAMETER smasm114
DC /2000 CONTROL PARAMETER smasm115

DC 0 CONVERSION INPUT AREA smasm112
DC 0 CONVERSION INPUT AREA smasm113
DC 0 CONVERSION INPUT AREA smasm114

DC 0 CONVERSION INPUT AREA smasm115

DC 0 CONVERSION INPUT AREA smasm116

DC 0 CONVERSION INPUT AREA smasm117

DC 0 CONVERSION INPUT AREA smasm118

DC 0 CONVERSION INPUT AREA smasm119

DC 0 CONVERSION INPUT AREA smasm120

DC 0 CONVERSION INPUT AREA smasm121

DC 0 CONVERSION INPUT AREA smasm122

DC 0 CONVERSION INPUT AREA smasm123

DC 0 CONVERSION INPUT AREA smasm124

DC 0 CONVERSION INPUT AREA smasm125

DC 0 CONVERSION INPUT AREA smasm126

DC 0 CONVERSION INPUT AREA smasm127

DC 0 CONVERSION INPUT AREA smasm128

DC 0 CONVERSION INPUT AREA smasm129

DC 0 CONVERSION INPUT AREA smasm130

DC 0 CONVERSION INPUT AREA smasm131

DC 0 CONVERSION INPUT AREA smasm132

DC 0 CONVERSION INPUT AREA smasm133

DC 0 CONVERSION INPUT AREA smasm134

DC 0 CONVERSION INPUT AREA smasm135

DC 0 CONVERSION INPUT AREA smasm136

DC 0 CONVERSION INPUT AREA smasm137

DC 0 CONVERSION INPUT AREA smasm138

DC 0 CONVERSION INPUT AREA smasm139

DC 0 CONVERSION INPUT AREA smasm140

DC 0 CONVERSION INPUT AREA smasm141

DC 0 CONVERSION INPUT AREA smasm142

Monitor System Sample Programs  H-7
Assembler Sample Program

*Ssymbol Table*

BEGIN 0000 BUSY 0010 D64 0031 MASK 0030 TYPE 0014

COO OVERFCW SECTORS SPECIFIED
COO OVERFCW SECTORS REQUIRED
CO6 SYMBOLS DEFINED
NC ERR(s) AND NO WARNING(S) FLAGGED IN ABOVE ASSEMBLY

// XEC
R 41 7908 (HEX) WCS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
FSCR 0248
LIBF TRANSFER VECTOR
FARC 069A
XMCS 067E
HCLL 062E
PRTY 05DE
EBPA 058E
FACD 04CD
FCIV 053C
FLD 0480
FACDX 04E3
FMPYX 049E
FSR 046C
FGETP 0452
NORM 0428
TYPEO 0312
EBPRST 02AC
IFIX 0280
FLOAT 0230
SYSTEM SUBROUTINES
ILSO4 OCC4
ILS02 0083
01FE (HEX) IS THE EXECUTION ADDR

8 IS THE SQUARE ROOT OF 64

output on
the console
printer
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:

If card input is from a 2501 Card Reader, change READ42 to READ01.

If printed output is to a 1403 Printer, change PRINTER to PRINT03. If printer output is on the console printer, change PRINTER to CONSOLE.
RPG Sample Program

INDICATORS

<table>
<thead>
<tr>
<th>IND</th>
<th>DISP</th>
<th>IND</th>
<th>DISP</th>
<th>IND</th>
<th>DISP</th>
<th>IND</th>
<th>DISP</th>
<th>IND</th>
<th>DISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>0150</td>
<td>00</td>
<td>0151</td>
<td>DF</td>
<td>0152</td>
<td>OV</td>
<td>0153</td>
<td>1P</td>
<td>0154</td>
</tr>
<tr>
<td>L1</td>
<td>0156</td>
<td>L2</td>
<td>0157</td>
<td>L3</td>
<td>0158</td>
<td>L4</td>
<td>0159</td>
<td>L5</td>
<td>015A</td>
</tr>
<tr>
<td>L7</td>
<td>015C</td>
<td>L8</td>
<td>015D</td>
<td>L9</td>
<td>015E</td>
<td>LR</td>
<td>015F</td>
<td>H1</td>
<td>0160</td>
</tr>
<tr>
<td>H3</td>
<td>0162</td>
<td>H4</td>
<td>0163</td>
<td>H5</td>
<td>0164</td>
<td>H6</td>
<td>0165</td>
<td>H7</td>
<td>0166</td>
</tr>
<tr>
<td>H9</td>
<td>0168</td>
<td>01</td>
<td>0169</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIELD NAMES

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DISP</th>
<th>L</th>
<th>T</th>
<th>D</th>
<th>FIELD</th>
<th>DISP</th>
<th>L</th>
<th>T</th>
<th>D</th>
<th>FIELD</th>
<th>DISP</th>
<th>L</th>
<th>T</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>016A</td>
<td>022</td>
<td>A</td>
<td>MONT</td>
<td>H</td>
<td>0181</td>
<td>002</td>
<td>N</td>
<td>0</td>
<td>DAY</td>
<td>0184</td>
<td>002</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>CUSTNO</td>
<td>0180</td>
<td>005</td>
<td>N</td>
<td>0</td>
<td>STATE</td>
<td>0193</td>
<td>002</td>
<td>N</td>
<td>0</td>
<td>CITY</td>
<td>0196</td>
<td>003</td>
<td>N</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>01A2</td>
<td>008</td>
<td>N</td>
<td>2</td>
<td>GRPTOT</td>
<td>01AB</td>
<td>008</td>
<td>N</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LITERALS

<table>
<thead>
<tr>
<th>LITERAL</th>
<th>LENGTH</th>
<th>TYPE</th>
<th>DISP</th>
<th>LITERAL</th>
<th>LENGTH</th>
<th>TYPE</th>
<th>DISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTS</td>
<td>R</td>
<td>24</td>
<td>A</td>
<td>RECEIvable</td>
<td>R</td>
<td>24</td>
<td>A</td>
</tr>
<tr>
<td>REGISTER</td>
<td>A</td>
<td>15</td>
<td>A</td>
<td>CUSTOMER</td>
<td>A</td>
<td>8</td>
<td>A</td>
</tr>
<tr>
<td>LOCATION</td>
<td>INVOICE</td>
<td>22</td>
<td>A</td>
<td>INVOICE DATE</td>
<td>INVOICE</td>
<td>23</td>
<td>A</td>
</tr>
<tr>
<td>NUMBER</td>
<td>CUSTOMER</td>
<td>24</td>
<td>A</td>
<td>NAME</td>
<td>A</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>STATE</td>
<td>CITY</td>
<td>NUMBER</td>
<td>24</td>
<td>NO</td>
<td>DAY</td>
<td>AMOUNT</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>027B</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>0289</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

KEY ADDRESSES OF OBJECT PROGRAM

<table>
<thead>
<tr>
<th>NAME OF ROUTINE</th>
<th>HEX DISP</th>
<th>NAME OF ROUTINE</th>
<th>HEX DISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>M + D LINES</td>
<td>04DE</td>
<td>TOTAL LINES</td>
<td>04EC</td>
</tr>
<tr>
<td>DETAIL CALCS</td>
<td>046E</td>
<td>TOTAL CALCS</td>
<td>047D</td>
</tr>
<tr>
<td>CHAIN ROUT 1</td>
<td>03D2</td>
<td>CONTROL FLD</td>
<td>03F3</td>
</tr>
<tr>
<td>LOW FIELD</td>
<td>042E</td>
<td>EXCEPT LINES</td>
<td>04FA</td>
</tr>
<tr>
<td>CLOSE FILES</td>
<td>067B</td>
<td>FILE SEQ 1</td>
<td>02EC</td>
</tr>
<tr>
<td>FILE SEQ 2</td>
<td>0377</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

END OF COMPILATION

// XEO
// L
// R 41 6D16 (HEX) WDS UNUSED BY CORE LOAD
CALL TRANSFER VECTOR
RGEPRR 022A
RLEBC 01A1
LIBF TRANSFER VECTOR
RGS19 11E4
RGRlk 11A8
RGEDE 105A
RGNV2 0FA6
RGAD0 00D0
RGS1 00D0
RGNV5 0C72
RGNV3 0D50
RGCMP 0CFE
RGNV1 0C6A
PRNT1 0A9A
ZICP0 097A
CARD0 0B7C
SYSTEM SUBROUTINES
ILSX4 1249
ILSX2 126D
ILSX1 12B6
ILS0 12A3

020F (HEX) IS THE EXECUTION ADDR
<table>
<thead>
<tr>
<th>CUSTOMER NUMBER</th>
<th>CUSTOMER NAME</th>
<th>LOCATION</th>
<th>INVOICE NUMBER</th>
<th>INVOICE MO</th>
<th>DATE</th>
<th>INVOICE AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10712</td>
<td>AMALGAMATED CORP</td>
<td>33 61</td>
<td>11603</td>
<td>11</td>
<td>10</td>
<td>$389.25</td>
</tr>
<tr>
<td>11315</td>
<td>BROWN WHOLESALE</td>
<td>30 231</td>
<td>12324</td>
<td>12</td>
<td>28</td>
<td>$602.08</td>
</tr>
<tr>
<td>11315</td>
<td>BROWN WHOLESALE</td>
<td>30 231</td>
<td>99988</td>
<td>12</td>
<td>14</td>
<td>$261.17</td>
</tr>
<tr>
<td>11897</td>
<td>FARM IMPLEMENTS</td>
<td>47 77</td>
<td>10901</td>
<td>10</td>
<td>18</td>
<td>$27.63</td>
</tr>
<tr>
<td>18530</td>
<td>BLACK OIL</td>
<td>16 67</td>
<td>11509</td>
<td>11</td>
<td>8</td>
<td>$592.95</td>
</tr>
<tr>
<td>18530</td>
<td>BLACK OIL</td>
<td>16 67</td>
<td>12292</td>
<td>12</td>
<td>27</td>
<td>$950.97</td>
</tr>
<tr>
<td>20716</td>
<td>LEATHER BELT CO</td>
<td>36 471</td>
<td>11511</td>
<td>11</td>
<td>8</td>
<td>$355.63</td>
</tr>
<tr>
<td>20716</td>
<td>LEATHER BELT CO</td>
<td>36 471</td>
<td>12263</td>
<td>12</td>
<td>17</td>
<td>$121.75</td>
</tr>
<tr>
<td>29017</td>
<td>GENERAL MFG CO</td>
<td>6 63</td>
<td>11619</td>
<td>11</td>
<td>14</td>
<td>$440.12</td>
</tr>
<tr>
<td>29017</td>
<td>GENERAL MFG CO</td>
<td>6 63</td>
<td>11676</td>
<td>11</td>
<td>23</td>
<td>$722.22</td>
</tr>
<tr>
<td>29054</td>
<td>A-B-C DIST CO</td>
<td>25 39</td>
<td>9689</td>
<td>9</td>
<td>11</td>
<td>$645.40</td>
</tr>
<tr>
<td>29054</td>
<td>A-B-C DIST CO</td>
<td>25 39</td>
<td>11605</td>
<td>11</td>
<td>11</td>
<td>$271.69</td>
</tr>
<tr>
<td>29054</td>
<td>A-B-C DIST CO</td>
<td>25 39</td>
<td>12234</td>
<td>12</td>
<td>14</td>
<td>$599.33</td>
</tr>
</tbody>
</table>

* * *
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.

// JC0 OEC0
LCC DRIVE CART SPEC CART AVAIL PHY DRIVE
CC00 OEC0 OEC0 COCC
V2 P09 ACTUAL 32K CONFIG 32K
// CLP
*IICRRECATA WS FX $$$$ O0IC
CART IC OEC0 CB ACCR 1F70 DB CNT CCA0
// FCR
*IICCS(IDCISK) LIST ALL
NAME UNFEX
CARD A/200/8/240/C13O/E(12)/F(3C0)
DATA A/200*+4.0/,B/24*5.0/,C/300*6.0/
WRITE (10)A
WRITE (10)C
READ FILE 10
BACKSPACE IC
REACt1ICF
REWINC 10
REACtICF
PAUSE 9999
CALL EXIT
END
VARIABLE ALLCCATICS
AIR )=018E—CCCO B(R )=018E-O190 C(R )=0416-O1C0 E(R )=042E-0418 F(R )=0686-0430
FEATURES SUPPCRTEC
ICCS
CALLED SUBPRGRAMS
URED LWP R LCCPP BKSP ECF REWNC PAUSE UFIO UICAF
INTEGER CCASTANTS
10=0688
9999=0689-26215=068A
CCRE REQUIREMENTS FOR UNFEX
CCPCPA C VARIABLES 1672 PROGRAM 52
ENC CF CCMPILATION
// CLP
*IICRRECATA WS LA UNFEX
CART IC OEC0 CB ACCR 2E50 DB CNT CC41
// XEC UNFEX

H-12
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 OEDO OEDO 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

// DUMP

"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
SYSUP Reeling Sample Program
for one drive systems

WRITE (1,K) J
WRITE (5,40)
40 FORMAT (4OMOLINK NO. 1 EXECUTED. CHANGE CARTRIDGES:///)
PAUSE 1111
CALL SYSUP (L(2))
CALL LINK (LINK2)
END

FEATURES SUPPORTED
ONE WORD INTEGERS
IOCS

CCRF REQUIRMENTS FOR LINK1
COMMON 0 VARIABLES 336 PROGRAM 180

END OF COMPILATION

// DUP

*STORECI WS UA LINK1 0001
*FILES(1,DATA,-OEDO)
FILES ALLOCATION
1 0206 0002 OEDO 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 1118
FLOAT 11FA
IFIX 11CE
PAUSE 0C8C
SCOMP 0799
SWRT 06B8
SDCOM 04D8
SDAI 043A
SDWRT 0461
SUBSC 0C6E
FSTO 0C9C
FLD 0C58
PRNTZ 0B5E
SFIO 07D5
SDFIO 04C0
SYSTEM SUBROUTINES
ILSO4 00C4
ILSO2 0083
ILSO1 1444
0370 (HEX) IS THE EXECUTION ADDR
CART ID OEDO DB ADDR 4530 DB CNT 00F0
// PAUS CHANGE TO CARTRIDGE OED4
// JOB OED4

LOG DRIVE CART SPEC CART AVAIL PHY DRIVE
0000 OED4 OED4 0000

V2 MO9 ACTUAL 32K CONFIG 92K

// DUP

*STORECI CD FX LINK2 0001
*FILES(2,DATA2,OED4)
FILES ALLOCATION
2 01F7 00C8 OED4 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
SCOMD 04B0
SDAI 0412
SDWRT 0439
SUBSC 0C46
FSTO 0C14
FLD 0C30
PRNTZ 0B36
SFI0 07AD
SDFI0 0498
SYSTEM SUBROUTINES
ILSO4 00C4
ILSO2 00B3
ILSO1 0CDC
0362 (HEX) IS THE EXECUTION ADDR
CART_ID OED4 DB ADDR 3230 DB CNT 00A0

// PAUS CHANGE TO CARTRIDGE OED0

// JOB OED0

LOG DRIVE CART SPEC CART AVAIL PHY DRIVE
0000 OED0 OED0 0000

V2 M09 ACTUAL 32K CONFIG 32K

// XEO LINK1

LINK NO. 1 EXECUTED, CHANGE CARTRIDGES.

LOG_DRIVE CART SPEC CART AVAIL PHY_DRIVE
0000 OED4 OED4 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.

```xml
// JOB OEDO OED4
LOG DRIVE CART SPEC CART AVAIL PIVY DRIVE
0000 OEDO OEDO OEDO 0000
0001 OED4 OED4 OED4 0001
V2 M09 ACTUAL 32K CONFIG 32K

// FOR
*NAMM MDEX1
*I0 CS (DISK)
*ONE WORD INTEGERS
*LIST SOURCE PROGRAM

DIMENSION J(320)
DEFINE FILE 1(210x320,U+K)
DEFINE FILE 2(200x320,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
  WRITE (2,KK) J
  M = 999
DO 9 N = 1, 320
  9 J(N) = M
  WRITE (2,IK) J
  CALL EXIT
END

FEATURES SUPPORTED
ONE WORD INTEGERS
I0CS

COMMON O VARIABLES P PROGRAM 34O
END OF COMPILATION

// DUP

*STORE WS UA MDEX1
CART ID OEDO DB ADDR 4515 DB CNT 0OOD

// XEO MDEX1 L 2

*FILES(1,DATA1,OEDO)
*FILES(2,DATA2,OED4)

FILES ALLOCATION
  1 O206 OOD2 OEDO DATA
  2 O1F7 O0C8 OED4 DATA2
STORAGE ALLOCATION
  R 41 76FA (HEX) WDS UNUSED BY CORE LOAD
  L1BF TRANSFER VECTOR
  PAUSE 0608
  SDCOM 04DA
  SDAT 043C
  SWRT 0463
  SUBSC 068A
  SDXIO 04C2
SYSTEM SUBRoutines
  ILS01 00C4
  ILS02 00B3
  O35A (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:
<table>
<thead>
<tr>
<th>CCCC</th>
<th>00</th>
<th>2C</th>
<th>START</th>
<th>LIBF</th>
<th>TYPEO</th>
<th>ISM0010</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCF1</td>
<td>C</td>
<td>00</td>
<td>CC</td>
<td>/2000</td>
<td>TYPE</td>
<td>ISM0020</td>
</tr>
<tr>
<td>OCO2</td>
<td>1</td>
<td>C</td>
<td>CC</td>
<td>SIGN</td>
<td>HEAD</td>
<td>ISM0030</td>
</tr>
<tr>
<td>OCO3</td>
<td>2</td>
<td>C</td>
<td>CC</td>
<td>/2000</td>
<td>COPY</td>
<td>ISM0040</td>
</tr>
<tr>
<td>OCO4</td>
<td>0</td>
<td>C</td>
<td>CC</td>
<td>L1</td>
<td>COPY</td>
<td>ISM0050</td>
</tr>
<tr>
<td>OCO5</td>
<td>0</td>
<td>7CFD</td>
<td>CC</td>
<td>B</td>
<td>WAIT4</td>
<td>ISM0060</td>
</tr>
<tr>
<td>OCO6</td>
<td>CC</td>
<td>65CCC004</td>
<td>CC</td>
<td>L1</td>
<td>SET</td>
<td>ISM0070</td>
</tr>
<tr>
<td>OCO7</td>
<td>C</td>
<td>C5CC0016</td>
<td>CC</td>
<td>L1</td>
<td>BTAB1</td>
<td>ISM0080</td>
</tr>
<tr>
<td>OCO8</td>
<td>C</td>
<td>C009</td>
<td>CC</td>
<td>STG</td>
<td>MESS</td>
<td>ISM0090</td>
</tr>
<tr>
<td>OCE9</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>/2CC0</td>
<td>TYPE</td>
<td>ISM0100</td>
</tr>
<tr>
<td>OCE10</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>WAIT1</td>
<td>ISM0110</td>
</tr>
<tr>
<td>OCE11</td>
<td>C</td>
<td>66CC0005</td>
<td>CC</td>
<td>L1</td>
<td>SET</td>
<td>ISM0120</td>
</tr>
<tr>
<td>OCE12</td>
<td>C</td>
<td>CCE0</td>
<td>CC</td>
<td>/CC0</td>
<td>WAIT1</td>
<td>ISM0130</td>
</tr>
<tr>
<td>OCE13</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>WAIT1</td>
<td>ISM0140</td>
</tr>
<tr>
<td>OCE14</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0150</td>
</tr>
<tr>
<td>OCE15</td>
<td>C</td>
<td>7CF8</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0160</td>
</tr>
<tr>
<td>OCE16</td>
<td>C</td>
<td>1CC0</td>
<td>CC</td>
<td>B</td>
<td>#CCNV</td>
<td>ISM0170</td>
</tr>
<tr>
<td>OCE17</td>
<td>C</td>
<td>CE01</td>
<td>CC</td>
<td>/CC0</td>
<td>#CCNV</td>
<td>ISM0180</td>
</tr>
<tr>
<td>OCE18</td>
<td>C</td>
<td>CC4F</td>
<td>CC</td>
<td>/CC0</td>
<td>#CCNV</td>
<td>ISM0190</td>
</tr>
<tr>
<td>OCE19</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0200</td>
</tr>
<tr>
<td>OCE20</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0210</td>
</tr>
<tr>
<td>OCE21</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0220</td>
</tr>
<tr>
<td>OCE22</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0230</td>
</tr>
<tr>
<td>OCE23</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0240</td>
</tr>
<tr>
<td>OCE24</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0250</td>
</tr>
<tr>
<td>OCE25</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0260</td>
</tr>
<tr>
<td>OCE26</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0270</td>
</tr>
<tr>
<td>OCE27</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0280</td>
</tr>
<tr>
<td>OCE28</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0290</td>
</tr>
<tr>
<td>OCE29</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0300</td>
</tr>
<tr>
<td>OCE30</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0310</td>
</tr>
<tr>
<td>OCE31</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0320</td>
</tr>
<tr>
<td>OCE32</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0330</td>
</tr>
<tr>
<td>OCE33</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0340</td>
</tr>
<tr>
<td>OCE34</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0350</td>
</tr>
<tr>
<td>OCE35</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0360</td>
</tr>
<tr>
<td>OCE36</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0370</td>
</tr>
<tr>
<td>OCE37</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0380</td>
</tr>
<tr>
<td>OCE38</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0390</td>
</tr>
<tr>
<td>OCE39</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0400</td>
</tr>
<tr>
<td>OCE40</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0410</td>
</tr>
<tr>
<td>OCE41</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0420</td>
</tr>
<tr>
<td>OCE42</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0430</td>
</tr>
<tr>
<td>OCE43</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0440</td>
</tr>
<tr>
<td>OCE44</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0450</td>
</tr>
<tr>
<td>OCE45</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0460</td>
</tr>
<tr>
<td>OCE46</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0470</td>
</tr>
<tr>
<td>OCE47</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0480</td>
</tr>
<tr>
<td>OCE48</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0490</td>
</tr>
<tr>
<td>OCE49</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0500</td>
</tr>
<tr>
<td>OCE50</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0510</td>
</tr>
<tr>
<td>OCE51</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0520</td>
</tr>
<tr>
<td>OCE52</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0530</td>
</tr>
<tr>
<td>OCE53</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0540</td>
</tr>
<tr>
<td>OCE54</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0550</td>
</tr>
<tr>
<td>OCE55</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0560</td>
</tr>
<tr>
<td>OCE56</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0570</td>
</tr>
<tr>
<td>OCE57</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0580</td>
</tr>
<tr>
<td>OCE58</td>
<td>C</td>
<td>CCC0</td>
<td>CC</td>
<td>B</td>
<td>CCNV</td>
<td>ISM0590</td>
</tr>
</tbody>
</table>
ISAM Sample Program

Calculating File Parameters

0C94 0 CC77 CC060
0C55 0 CC09 CC061
0096 0 BC1B CC062
CC67 0 CC1C CC063
0C5E 0 CC1B CC064
CC69 0 189D CC065
CA60 0 A81A CC066
0C82 0 CC1F CC067
0C5C 0 CCF1 CC068
0C6C 0 8C1C CC069
0C6E 0 9C12 CC070
0C6F 0 1890 CC071
CAAC 0 A81A CC072
CA00 0 BC17 CC073
0A22 0 8C10 CC074
0A32 0 1803 CC075
CA44 0 CC15 CC076
0A55 0 8C16 CC077
0A60 0 9C0A CC078
CA67 0 1890 CC079
0A86 0 A813 CC080
0A90 0 8C0E CC081
CAAB 0 RC1E CC083
0CAC 0 8C04 CC084
0CAE 0 CC09 CC085
0CAE 065CC0C6 CC086
0C6C 0 7C13 CC087
OC81 0 CC01 CC088
00P2 0 8C02 CC089
OC83 0 CC07 CC090
0C84 0 CI40 CC091
0C85 0 CCCC CC092
OCPE 0 ICCC CC093
OC87 0 CCCC CC094
0C88 0 CCCC CC095
0C89 0 CCCC CC096
0C8B 0 CCCC CC097
0C8C 0 CCCC CC098
0C8E 0 CCCC CC099
0CBE 0 CCCC CC100
0CBE 0 CCCC CC101
00F1 0 1C05 CC102
C001 0 0119 CC103
C0C1 0 012E CC104
0CC2 0 0142 CC105
0CC3 0 0156 CC106
CCCE 0 189C CC107
0CC6 0 18CC0CC CF108
0CC7 0 18CC0CC CF109
0CCA 0 2255103 CC110
0CCB 0 0086 CC111
0CCD 0 22557CE6 CC112
0CCF 0 110C CC113
0CCF 0 1C87 CC114
0CCF 0 1CCC CC115
0CCC 0 1CCC CC116
0CC1 0B8530C7 CC117
0CC3 0 1CCC0CC CF118
0CC5 0 1CCC0CC CF119
0CC6 0 23A17170 CC120
0CC7 0 2C00 CC121
0CC8 0 1CCC CC122
0CC9 0 23A17170 CC123
0CCE 0 0CC0 CC124
0CCE 0 7CFC CC125

STC IEP5
LC LKGR
A TWC
STC WRCR
LC SCLLG
SRT 16
C WRCR
STC RCPDS
LC RCRD
A RCPDS
S CNE
STC 16
D RCPDS
STC NCPS
STC NCPS
A IEP5
A SEVEN
S CNE
STC SCTLG
LC NCPS
STC RCPD
LC RCPD
STC NCPS
STC NCPS
IEPS
STC NCPS

RCUT
C LI MTAB
STC L ACCR
LC L VTA9
LIRF BICE
CC OLT2
LIRF ZIPO
CC //1100
CC OLT3
DC OLT1
ADDR DC *-
DC 5
CALL HCLCP
LC LI VTAB2
STC MESSP
LIRF TYPE0
CC //2C00
MESSP DC *-
WAIT3 LIRF TYPE0
CC //CC00
B WAIT3

INDEX ENTRIES PER SECTOR
CREATE DIVISOR BY ADDING
*TWO TO THE RECORD SIZE AN
STORING IN HCLD AREA
DIVIDE RECORD LENGTH+2
*INTO THE SECTOR LENGTH
TC CALCULATE THE NUMBER
OF RECORDS PER SECTOR
DIVIDE THE TOTAL NUMBER OF
*RECORDS PLUS NO. CF REC.
*PER SECTOR MINUS CNE
*BY THE NUMBER OF
RECORDS PER SECTOR TC FIND
NO. CF PRIME DATA SECTORS
ADD CONSTANT OF SEVEN
DIVIDE BY 8 TO DETERMINE
*NO. OF PRIME DATA CYLINDRS
ADD INDEX ENTRIES/SECTOR
MINUS ONE
DIVIDE BY NO. OF INDEX
ENTRIES/SECTOR TO FIND NO
*OF INDEX SECTORS
ADD NO OF INDEX SECTORS*
NO. OF PRIME DATA SECTORS
*+ NO OF OVERFLOW SECTORS
SET COUNT

CONSTANT OF ONE
CONSTANT OF TWO
CONSTANT OF SEVEN
NO. WORDS PER SECTOR
TEMPORARY HCLD AREA
TABLE OF OUTPUT VALUES
TOTAL NC. CF SECTORS
NO. OF INDEX SECTORS
TOL NC. CF PRIME DATA SECTORS
NO. OF PRIME CYLINDERS
NO. OF RECORDS PER SECTOR
NO. OF INDEX ENTRIES/SECTOR
MESSAGE TABLE CONTAINS
*ADDRESS IN MESSAGES
*WHERE THE VALUES ARE TO
*BE INSERTED

MOVE ADDR CF CORRECT
*MSG TO CONVERT ROLLINE
LOAD A VALUE TO CONVERT
GO CONVERT FROM BINARY TO
*IBM CARD CODE
CONVERT FROM IBM CARD CODE
*TC CONSOLE CODE AND
*PLACE VALUE IN
*MESSAGE

LOAD ADDR CF MESSAGE
STORE ADDR FCR SUBROUTINE
LOAD ADDR CF MESSAGE
LOAD ADDR CF MESSAGE
### ISAM Sample Program

**Calculating file parameters**

```
OCCC C 71FF CC126 MX 1 - 1 DECEND CCUN IF COUNT NCN-ZERO BRANCH ISM01260
OCCC 0 7CE6 CO127 B RCU T COTHERWISE, CALL EXIT ISM01270
OCEE 0 6C38 CO128 EXIT TABLE OF ACCRESSES ISM01280
OCF6 0 1CC0 C0129 BTAB2 NOP ISM01290
OCF7 0 7CE6 CO127 C0130 CC MS5 ISM01300
OCF8 0 7CE6 CO127 CC MS6 ISM01310
OCF9 0 7CE6 CO127 CC MS7 ISM01320
OCF10 0 7CE6 CO127 CC MS8 ISM01330
OCF11 0 7CE6 CO127 CC MS9 ISM01340
OCF12 0 7CE6 CO127 DC MS10 ISM01350
OCF13 0 CC11 CO136 MS5 DC 17 ISM01360
OCF14 0 CC1B CC137 CMES 'TOTAL NUMBER OF SECTORS = ' ISM01370
CCF4 0 CC05 CO138 MS5P CMES 'E ISM01380
OCF7 0 CC11 CO139 MS6 CC 17 ISM01390
OCF8 0 CC18 CC140 DCMES 'RNUMBER OF INDEX SECTORS = ' ISM01400
OCF9 0 CC18 CC140 MS6P CMES 'E ISM01410
OCF10 0 CC18 CC140 MS7 DC 19 ISM01420
OCF11 0 CC06 CC144 MS7P CMES 'RNUMBER OF PRIME DATA SECTORS = ' ISM01430
OCF12 0 CC1B CC146 CMES 'E ISM01440
OCF13 0 CC1B CC146 MS8 DC 20 ISM01450
OCF14 0 CC22 CC148 MS8P CMES 'RNUMBER OF PRIME DATA CYLINDERS = ' ISM01460
OCF15 0 CC06 CC147 MS8P CMES 'E ISM01470
OCF16 0 CC13 CC148 MS9 DC 15 ISM01480
OCF17 0 CC2O CC149 DCMES 'RNUMBER OF RECORDS PER SECTOR = ' ISM01490
OCF18 0 CC06 CC150 MS9P CMES 'E ISM01500
OCF19 0 CC13 CC151 MS10 DC 15 ISM01510
OCF20 0 CC20 CC152 DCMES 'RNUMBER OF INDEXES PER SECTOR = ' ISM01520
OCF21 0 CC06 CC153 MS1CP CMES 'E ISM01530
OCF22 0 CC1B CC154 CMES 'E ISM01540
```

### CROSS-REFERENCE

#### SYMBOL VALUE REL DEFN REFERENCES

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>VALUE</th>
<th>REL</th>
<th>DEFN</th>
<th>REFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>COCF</td>
<td>1</td>
<td>C0115</td>
<td>CC1C8, M</td>
</tr>
<tr>
<td>BTAB1</td>
<td>COCF</td>
<td>1</td>
<td>C0116</td>
<td>C0019, R</td>
</tr>
<tr>
<td>BTAB2</td>
<td>COCF</td>
<td>1</td>
<td>C0117</td>
<td>C0120, R</td>
</tr>
<tr>
<td>CAL</td>
<td>CO91</td>
<td>1</td>
<td>C0057</td>
<td>CC047, B</td>
</tr>
<tr>
<td>CVA</td>
<td>CO71</td>
<td>1</td>
<td>C0034</td>
<td>C0018, B</td>
</tr>
<tr>
<td>IEPS</td>
<td>COB2</td>
<td>1</td>
<td>C0024</td>
<td>C0032, R</td>
</tr>
<tr>
<td>IEPS2</td>
<td>COB1</td>
<td>1</td>
<td>C0049</td>
<td>CC060, M</td>
</tr>
<tr>
<td>IA</td>
<td>CC08</td>
<td>1</td>
<td>C0008</td>
<td>CC046, B</td>
</tr>
<tr>
<td>IC</td>
<td>CO80</td>
<td>1</td>
<td>C0048</td>
<td>C0017, M</td>
</tr>
<tr>
<td>LENGI</td>
<td>CC90</td>
<td>1</td>
<td>C0056</td>
<td>C0059, R</td>
</tr>
<tr>
<td>LENGR</td>
<td>COBF</td>
<td>1</td>
<td>C0055</td>
<td>C0061, M</td>
</tr>
<tr>
<td>MESS</td>
<td>CO0C</td>
<td>1</td>
<td>C0112</td>
<td>C0009, M</td>
</tr>
<tr>
<td>MESSP</td>
<td>CO88</td>
<td>1</td>
<td>C0121</td>
<td>C0199, M</td>
</tr>
<tr>
<td>MESS1</td>
<td>CO3C</td>
<td>1</td>
<td>C0026</td>
<td>C0023, R</td>
</tr>
<tr>
<td>MESS2</td>
<td>CO41</td>
<td>1</td>
<td>C0028</td>
<td>C0022, R</td>
</tr>
<tr>
<td>MESS3</td>
<td>CO4F</td>
<td>1</td>
<td>C0030</td>
<td>C0021, R</td>
</tr>
<tr>
<td>MESS6</td>
<td>CO61</td>
<td>1</td>
<td>C0032</td>
<td>C002C, R</td>
</tr>
<tr>
<td>MESS10</td>
<td>C0151</td>
<td>1</td>
<td>C0151</td>
<td>C0135, R</td>
</tr>
<tr>
<td>MESS11</td>
<td>C0156</td>
<td>1</td>
<td>C0153</td>
<td>C016C, R</td>
</tr>
<tr>
<td>MESS12</td>
<td>C0157</td>
<td>1</td>
<td>C0136</td>
<td>C0130, R</td>
</tr>
<tr>
<td>MESS13</td>
<td>COF4</td>
<td>1</td>
<td>C0138</td>
<td>C0101, R</td>
</tr>
<tr>
<td>MESS14</td>
<td>COF7</td>
<td>1</td>
<td>C0139</td>
<td>C0131, R</td>
</tr>
<tr>
<td>MESS15</td>
<td>CO71</td>
<td>1</td>
<td>C0141</td>
<td>C0102, R</td>
</tr>
<tr>
<td>MESS16</td>
<td>C0105</td>
<td>1</td>
<td>C0143</td>
<td>C0132, R</td>
</tr>
<tr>
<td>MESS17</td>
<td>C0108</td>
<td>1</td>
<td>C0144</td>
<td>C0143, R</td>
</tr>
<tr>
<td>MESS18</td>
<td>C0119</td>
<td>1</td>
<td>C0145</td>
<td>C0133, R</td>
</tr>
<tr>
<td>MESS19</td>
<td>C011C</td>
<td>1</td>
<td>C0140</td>
<td>C0104, R</td>
</tr>
<tr>
<td>MESS20</td>
<td>C012E</td>
<td>1</td>
<td>C0147</td>
<td>C0140, R</td>
</tr>
<tr>
<td>MESS21</td>
<td>C0138</td>
<td>1</td>
<td>C0148</td>
<td>C0140, R</td>
</tr>
<tr>
<td>MESS22</td>
<td>C0149</td>
<td>1</td>
<td>C0150</td>
<td>C0150, R</td>
</tr>
<tr>
<td>MESS23</td>
<td>C0151</td>
<td>1</td>
<td>C0152</td>
<td>C0152, R</td>
</tr>
<tr>
<td>MESS24</td>
<td>C0153</td>
<td>1</td>
<td>C0154</td>
<td>C0154, R</td>
</tr>
<tr>
<td>MESS25</td>
<td>C0155</td>
<td>1</td>
<td>C0156</td>
<td>C0156, R</td>
</tr>
<tr>
<td>MESS26</td>
<td>C0157</td>
<td>1</td>
<td>C0157</td>
<td>C0157, R</td>
</tr>
<tr>
<td>MESS27</td>
<td>C0158</td>
<td>1</td>
<td>C0158</td>
<td>C0158, R</td>
</tr>
<tr>
<td>MESS28</td>
<td>C0159</td>
<td>1</td>
<td>C0159</td>
<td>C0159, R</td>
</tr>
<tr>
<td>MESS29</td>
<td>C0160</td>
<td>1</td>
<td>C0161</td>
<td>C0161, R</td>
</tr>
<tr>
<td>MESS30</td>
<td>C0162</td>
<td>1</td>
<td>C0163</td>
<td>C0163, R</td>
</tr>
<tr>
<td>MESS31</td>
<td>C0164</td>
<td>1</td>
<td>C0164</td>
<td>C0164, R</td>
</tr>
<tr>
<td>MESS32</td>
<td>C0165</td>
<td>1</td>
<td>C0165</td>
<td>C0165, R</td>
</tr>
<tr>
<td>MESS33</td>
<td>C0166</td>
<td>1</td>
<td>C0166</td>
<td>C0166, R</td>
</tr>
<tr>
<td>MESS34</td>
<td>C0167</td>
<td>1</td>
<td>C0167</td>
<td>C0167, R</td>
</tr>
<tr>
<td>MESS35</td>
<td>C0168</td>
<td>1</td>
<td>C0168</td>
<td>C0168, R</td>
</tr>
<tr>
<td>MESS36</td>
<td>C0169</td>
<td>1</td>
<td>C0169</td>
<td>C0169, R</td>
</tr>
<tr>
<td>MESS37</td>
<td>C0170</td>
<td>1</td>
<td>C0170</td>
<td>C0170, R</td>
</tr>
<tr>
<td>MESS38</td>
<td>C0171</td>
<td>1</td>
<td>C0171</td>
<td>C0171, R</td>
</tr>
<tr>
<td>MESS39</td>
<td>C0172</td>
<td>1</td>
<td>C0172</td>
<td>C0172, R</td>
</tr>
<tr>
<td>MESS40</td>
<td>C0173</td>
<td>1</td>
<td>C0173</td>
<td>C0173, R</td>
</tr>
<tr>
<td>MESS41</td>
<td>C0174</td>
<td>1</td>
<td>C0174</td>
<td>C0174, R</td>
</tr>
<tr>
<td>MESS42</td>
<td>C0175</td>
<td>1</td>
<td>C0175</td>
<td>C0175, R</td>
</tr>
<tr>
<td>MESS43</td>
<td>C0176</td>
<td>1</td>
<td>C0176</td>
<td>C0176, R</td>
</tr>
<tr>
<td>MESS44</td>
<td>C0177</td>
<td>1</td>
<td>C0177</td>
<td>C0177, R</td>
</tr>
<tr>
<td>MESS45</td>
<td>C0178</td>
<td>1</td>
<td>C0178</td>
<td>C0178, R</td>
</tr>
<tr>
<td>MESS46</td>
<td>C0179</td>
<td>1</td>
<td>C0179</td>
<td>C0179, R</td>
</tr>
<tr>
<td>MESS47</td>
<td>C0180</td>
<td>1</td>
<td>C0180</td>
<td>C0180, R</td>
</tr>
<tr>
<td>MESS48</td>
<td>C0181</td>
<td>1</td>
<td>C0181</td>
<td>C0181, R</td>
</tr>
<tr>
<td>MESS49</td>
<td>C0182</td>
<td>1</td>
<td>C0182</td>
<td>C0182, R</td>
</tr>
</tbody>
</table>

H-20
ISAM Sample Program
Calculating file parameters

CAC COB1 1 CC068 CCC70,R CC078,R CC084,R
CLT COB7 1 C0051 C0114,R
CLT1 COB6 1 CG05C CC040,R CC011,R
CVRSC COB8 1 C0053 C0083,R
RCRPS COB8 1 C009e C0067,R CC069,R CC072,R
RCRDO 00E 1 C0054 C0068,R
RCRT COB4 1 C0107 CC087,R CC0127,R
SEVER COB3 1 CC090 CC074,R
START COB7 1 CC001 C0154,R
TCTSC COB7 1 CC094 C0085,R
ThC COB2 1 C0089 C0062,R
VTAB1 COB5 1 C0052 CC044,R
VTAB2 COB6 1 CC093 C0109,R
WAIT C074 1 C0037 C0039,R
WAIT1 COOE 1 C0013 C0015,R
WAIT3 COC9 1 C0123 C00125,R
WAIT4 COC3 1 CC004 CC006,R
Werk COB5 1 C0092 C0063,R CC066,R

CCC OVERFLOWS SECTORS SPECIFIED
CC0 OVERFLOW SECTORS REQUIRED
052 SYMBOLS DEFINED

NC ERROR(S) AND NO WARNING(S) FLAGGED IN ABOVE ASSEMBLY
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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Type of program</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mainline (absolute)</td>
</tr>
<tr>
<td>2</td>
<td>Mainline (relocatable)</td>
</tr>
<tr>
<td>3</td>
<td>Subprogram, not an ISS, referenced by an LBF statement</td>
</tr>
<tr>
<td>4</td>
<td>Subprogram, not an ISS, referenced by a CALL statement</td>
</tr>
<tr>
<td>5</td>
<td>Interrupt service subroutine (ISS), referenced by an LBF statement</td>
</tr>
<tr>
<td>6</td>
<td>Interrupt service subroutine (ISS), referenced by a CALL statement</td>
</tr>
<tr>
<td>7</td>
<td>Interrupt level subroutine (ILS)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Subtype</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3, 4</td>
<td>In-core subprograms</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Disk FORTRAN I/O subroutines</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Arithmetic subroutines</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Nondisk FORTRAN I/O and &quot;Z&quot; conversion subroutines</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>&quot;Z&quot; device subroutines</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Function subprogram</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>Dummy ILS02, ILS04</td>
</tr>
</tbody>
</table>

Monitor system formats are described in the following text.
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
- 100: LIBF
- 110: 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zero</td>
</tr>
<tr>
<td>2</td>
<td>Checksum, if the source was cards; otherwise, a zero.</td>
</tr>
</tbody>
</table>
| 3    | Subtype (bits 0 through 3)  
Program type (bits 4 through 7)  
Precision bits: 

<table>
<thead>
<tr>
<th>Integer Precision</th>
<th>3RD HEX DIGIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>unspecified</td>
<td>0</td>
</tr>
<tr>
<td>matches real</td>
<td>8</td>
</tr>
<tr>
<td>one word</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Real Precision</th>
<th>4TH HEX DIGIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>unspecified</td>
<td>0</td>
</tr>
<tr>
<td>standard</td>
<td>1</td>
</tr>
<tr>
<td>extended</td>
<td>2</td>
</tr>
</tbody>
</table>

| 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) |

1. All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.
2. 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 and 14</td>
<td>Name of entry point 2 (in name code)</td>
</tr>
<tr>
<td>15</td>
<td>Relative address of entry point 2</td>
</tr>
<tr>
<td>16 and 17</td>
<td>Name of entry point 3 (in name code)</td>
</tr>
<tr>
<td>18</td>
<td>Relative address of entry point 3</td>
</tr>
<tr>
<td>19 through 51</td>
<td>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.</td>
</tr>
</tbody>
</table>

For program types 5 and 6, the program header, in addition to words 1 through 12, contains the following information:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>ISS number plus 50</td>
</tr>
<tr>
<td>14</td>
<td>ISS number</td>
</tr>
<tr>
<td>15</td>
<td>Number of interrupt levels required</td>
</tr>
<tr>
<td>16</td>
<td>Interrupt level number associated with the primary interrupt</td>
</tr>
<tr>
<td>17</td>
<td>Interrupt level number associated with the secondary interrupt</td>
</tr>
</tbody>
</table>

1 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.

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.
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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Relative address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>@XEOA</td>
<td>0</td>
<td>Execution address of the core load</td>
</tr>
<tr>
<td>@CMON</td>
<td>1</td>
<td>Length of COMMON (in words)</td>
</tr>
<tr>
<td>@DREQ</td>
<td>2</td>
<td>Disk I/O subroutine indicator — 0FFF for DISKZ, 0000 for DISK1, 0001 for DISK</td>
</tr>
<tr>
<td>@FILE</td>
<td>3</td>
<td>Number of files defined</td>
</tr>
<tr>
<td>@HWCT</td>
<td>4</td>
<td>Length of the core image header (in words)</td>
</tr>
<tr>
<td>@LSCT</td>
<td>5</td>
<td>Sector count of files in system WS</td>
</tr>
<tr>
<td>@LDAD</td>
<td>6</td>
<td>Loading address of the core load</td>
</tr>
<tr>
<td>@XCTL</td>
<td>7</td>
<td>Exit control address for DISK1/N</td>
</tr>
<tr>
<td>@TVWC</td>
<td>8</td>
<td>Length of the transfer vector (in words)</td>
</tr>
<tr>
<td>@WCNT</td>
<td>9</td>
<td>Length, in words, of the core load, core image header, and the transfer vector</td>
</tr>
<tr>
<td>@XR3X</td>
<td>10</td>
<td>Setting for the index register 3 during execution of the core load</td>
</tr>
<tr>
<td>@ITVX</td>
<td>11</td>
<td>Contents of word 8 during execution</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Contents of word 9 during execution</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Contents of word 10 during execution</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Contents of word 11 during execution</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Contents of word 12 during execution</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Contents of word 13 during execution</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>18 through 20</td>
<td></td>
<td>Interrupt entry to 1231 ISS</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Interrupt entry to 1403 ISS</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Interrupt entry to 2501 ISS</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Interrupt entry to 1442 ISS</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Interrupt entry to keyboard/console printer ISS</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Interrupt entry to 1134/1055 ISS</td>
</tr>
<tr>
<td>@OVSW</td>
<td>26</td>
<td>Sector count of LOCALs/SOCALs</td>
</tr>
<tr>
<td>@CORE</td>
<td>27</td>
<td>Core size of system on which core load built</td>
</tr>
<tr>
<td>28 and 29</td>
<td></td>
<td>Define file table checksum work area</td>
</tr>
<tr>
<td>@HEND</td>
<td>29</td>
<td>Last word of core image header</td>
</tr>
</tbody>
</table>
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.

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.

The mainline header card is the first card of every type 1 or 2 program in CDS format. This card contains:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
</tbody>
</table>
| 3    | Type code (first 8 bits):  
|      | 0000 0001 absolute  
|      | 0000 0010 relocatable  
|      | Precision bits:  
|      | Integer Precision |
|      | unspecified  
|      | matches real  
|      | one word  
|      | Real Precision |
|      | unspecified  
|      | standard  
|      | extended  
| 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 |

1. All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified, unless an EPR or SPR card is used.
2. Two words for standard precision, three for extended.
The subprogram header card is the first card of every type 3 or 4 program in card system format. This card contains:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Precision bits:</td>
</tr>
<tr>
<td></td>
<td>Integer Precision 3RD HEX DIGIT</td>
</tr>
<tr>
<td></td>
<td>unspecified 0</td>
</tr>
<tr>
<td></td>
<td>matches real 8</td>
</tr>
<tr>
<td></td>
<td>one word 9</td>
</tr>
<tr>
<td></td>
<td>Real Precision 4TH HEX DIGIT</td>
</tr>
<tr>
<td></td>
<td>unspecified 0</td>
</tr>
<tr>
<td></td>
<td>standard 1</td>
</tr>
<tr>
<td></td>
<td>extended 2</td>
</tr>
<tr>
<td>4 and 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Number of entry points times three</td>
</tr>
<tr>
<td>7 through 9</td>
<td>Reserved</td>
</tr>
<tr>
<td>10 and 11</td>
<td>Name of entry point 1 (in name code)</td>
</tr>
<tr>
<td>12</td>
<td>Relative address of entry point 1</td>
</tr>
<tr>
<td>13 through 51</td>
<td>Names and relative addresses of entry point 2 through 14, as required</td>
</tr>
<tr>
<td>52 through 54</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

1. All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.
2. Two words for standard precision, three for extended.
The ISS header card is the first card of every type 5 or 6 program in CDS format, and contains:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td>Type code (first 8 bits):&lt;br&gt;0000 0101 to be called by an LIBF statement only&lt;br&gt;0000 0110 to be called by a CALL statement only&lt;br&gt;Precision bits:&lt;br&gt;Integer Precision&lt;br&gt;unspecified 1&lt;br&gt;matches real 2&lt;br&gt;one word 3&lt;br&gt;Real Precision&lt;br&gt;unspecified 1&lt;br&gt;standard 2&lt;br&gt;extended 3</td>
</tr>
<tr>
<td>4 and 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Number of interrupt levels required plus 6</td>
</tr>
<tr>
<td>7 through 9</td>
<td>Reserved</td>
</tr>
<tr>
<td>10 and 11</td>
<td>Subroutine name (in name code)</td>
</tr>
<tr>
<td>12</td>
<td>Relative entry point address</td>
</tr>
<tr>
<td>13 and 14</td>
<td>Reserved for parameters used by the 1130 Card/Paper Tape System</td>
</tr>
<tr>
<td>15</td>
<td>Number of interrupt levels required</td>
</tr>
<tr>
<td>16</td>
<td>Interrupt level number associated with the primary interrupt</td>
</tr>
<tr>
<td>17</td>
<td>Interrupt level associated with the secondary interrupt level</td>
</tr>
<tr>
<td>18 through 29</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>One</td>
</tr>
<tr>
<td>31 through 54</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

1. All FORTRAN programs specify precision of both real and integers. Real precision in assembler is unspecified unless an EPR or SPR card is used.
2. Two words for standard precision, three for extended.
3. 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td>Type code (first 8 bits): 0000 0111, Reserved (last 8 bits)</td>
</tr>
<tr>
<td>4 and 5</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>0000 0000 0000 0100</td>
</tr>
<tr>
<td>7 through 9</td>
<td>Reserved</td>
</tr>
<tr>
<td>10 through 12</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>Interrupt level number</td>
</tr>
<tr>
<td>14 through 54</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td>Type code (first 8 bits): 0000 1010, Count of data words, excluding indicator data words, in these cards (last 8 bits)</td>
</tr>
<tr>
<td>4 through 9</td>
<td>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)</td>
</tr>
<tr>
<td>10 through 54</td>
<td>Data words 1 through 45</td>
</tr>
</tbody>
</table>
The end-of-program card is the last card of all programs in CDS format, and contains:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Effective length of the program. This number is always even and is assigned by the assembler, FORTRAN compiler, or RPG compiler.</td>
</tr>
<tr>
<td>2</td>
<td>Checksum</td>
</tr>
<tr>
<td>3</td>
<td>Type code (first 8 bits):&lt;br&gt;0000 1111&lt;br&gt;Last 8 bits:&lt;br&gt;0000 0000</td>
</tr>
<tr>
<td>4</td>
<td>Execution address (mainline program only)</td>
</tr>
<tr>
<td>5 through 54</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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.
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:

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 CDI program:

### DSF Program

<table>
<thead>
<tr>
<th>*ECPM</th>
<th>UA</th>
<th>PR</th>
<th>SAMPL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADDR</strong></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>2</strong></em></td>
</tr>
<tr>
<td>000C</td>
<td>C00C</td>
<td>0000</td>
<td>0100</td>
</tr>
<tr>
<td>010C</td>
<td>717C</td>
<td>C00C</td>
<td>0237</td>
</tr>
<tr>
<td>020C</td>
<td>720C</td>
<td>C00C</td>
<td>0337</td>
</tr>
<tr>
<td>030C</td>
<td>024H</td>
<td>C00C</td>
<td>0437</td>
</tr>
<tr>
<td>040C</td>
<td>032H</td>
<td>13C</td>
<td>850C</td>
</tr>
<tr>
<td>050C</td>
<td>021H</td>
<td>3C</td>
<td>13C</td>
</tr>
<tr>
<td>070C</td>
<td>024H</td>
<td>024H</td>
<td>024H</td>
</tr>
<tr>
<td>080C</td>
<td>032H</td>
<td>032H</td>
<td>032H</td>
</tr>
<tr>
<td>090C</td>
<td>021H</td>
<td>021H</td>
<td>021H</td>
</tr>
<tr>
<td>0B0C</td>
<td>024H</td>
<td>024H</td>
<td>024H</td>
</tr>
<tr>
<td>0C0C</td>
<td>032H</td>
<td>032H</td>
<td>032H</td>
</tr>
<tr>
<td>0D0C</td>
<td>021H</td>
<td>021H</td>
<td>021H</td>
</tr>
<tr>
<td>0F0C</td>
<td>024H</td>
<td>024H</td>
<td>024H</td>
</tr>
</tbody>
</table>

### Core Image Program

**(note that the actual starting address**

### Disk block on sector. For data files, this position will always be 0 (data files must start on sector boundary).

### Formats I-13
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.
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:

```
Name code words in hexadecimal    D505 4140
Equivalent binary words
Indicator bits

<table>
<thead>
<tr>
<th>Input characters</th>
<th>EBCDIC hex</th>
<th>EBCDIC binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>D5</td>
<td>1101 0101</td>
</tr>
<tr>
<td>A</td>
<td>C1</td>
<td>1100 0001</td>
</tr>
<tr>
<td>M</td>
<td>D4</td>
<td>1101 0100</td>
</tr>
<tr>
<td>E</td>
<td>C5</td>
<td>1100 0101</td>
</tr>
<tr>
<td>@</td>
<td>40</td>
<td>0100 0000</td>
</tr>
</tbody>
</table>
```
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:

\[ m \cdot w \cdot t \ (P) \]

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 15-18.2 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F0F1</td>
<td>F4F9</td>
<td>F7F4</td>
<td>F0F0</td>
</tr>
</tbody>
</table>

Example 2 (truncation): The integer field of Example 1 is converted using the field specification 15-16.2 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F4F9</td>
<td>F7F4</td>
<td>F0F0</td>
</tr>
</tbody>
</table>

Example 3 (packed format): The integer field of example 1 is converted using the field specification 15-18.2(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).

<table>
<thead>
<tr>
<th>Record word</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>0014</td>
<td>9740</td>
<td>0F40</td>
</tr>
</tbody>
</table>

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 15-18.2 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F0F1</td>
<td>F4F9</td>
<td>F7F4</td>
<td>F0D0</td>
</tr>
</tbody>
</table>

**J-FIELD TYPE**

This field type describes 2-word integer conversion; input is a 2-word integer. The specification is:

\[ m-J^w.t (P) \]

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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>0021</td>
<td>4748</td>
<td>3647</td>
<td>0F40</td>
</tr>
</tbody>
</table>

**R-FIELD TYPE**

This field type describes FORTRAN real-variable conversion. The specification is:

\[ m-R^w.t (P) \]

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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>0053</td>
<td>125D</td>
</tr>
</tbody>
</table>

**Example 2:** The real field of Example 1 is converted using the field specification 25-R7.5 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F0F0</td>
<td>F5F3</td>
<td>F1F2</td>
<td>D540</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>8000</td>
<td>000F</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>9999</td>
<td>999F</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F0F0</td>
<td>F0F0</td>
<td>F0F0</td>
<td>F0F0</td>
<td>F040</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Record word</th>
<th>n</th>
<th>n+1</th>
<th>n+2</th>
<th>n+3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>E5C5</td>
<td>E3C9</td>
<td>E2C9</td>
<td>D7D6</td>
</tr>
<tr>
<td></td>
<td>VE</td>
<td>TI</td>
<td>SI</td>
<td>PO</td>
</tr>
</tbody>
</table>

This field is converted using the field specification 21-B8.2 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>D7D6</td>
<td>E2C9</td>
<td>E3C9</td>
<td>E5C5</td>
</tr>
<tr>
<td></td>
<td>PO</td>
<td>SI</td>
<td>TI</td>
<td>VE</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Record word</th>
<th>210</th>
<th>211</th>
<th>212</th>
<th>213</th>
<th>214</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>4BC3</td>
<td>4B40</td>
<td>4040</td>
<td>D6D5</td>
<td>6BC4</td>
<td>4040</td>
</tr>
<tr>
<td></td>
<td>.C</td>
<td>.ON</td>
<td>.D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Record word</th>
<th>216</th>
<th>217</th>
<th>218</th>
<th>219</th>
<th>220</th>
<th>221</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>C9D5</td>
<td>C7E3</td>
<td>4040</td>
<td>E6C1</td>
<td>E2C8</td>
<td>4040</td>
</tr>
<tr>
<td></td>
<td>IN</td>
<td>GT</td>
<td>WA</td>
<td>SH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This field type describes CSP D1 conversion. The specification is

\[ \text{D-FIELD TYPE} \]

\[ \text{m-D}I_{12}.j=1_{3}.K(P) \]

where

- \( m \) is the column of the RPG record in which the converted field begins (1 through 640).
- \( D \) identifies the field type.
- \( I_1 \) 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.
- \( I_2 \) 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, \( I_1 = I_2 \) 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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>72</th>
<th>73</th>
<th>74</th>
<th>75</th>
<th>76</th>
<th>77</th>
<th>78</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>0008</td>
<td>0008</td>
<td>0006</td>
<td>0004</td>
<td>0009</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

This field is converted using the field specification 35-C15.4 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>E6C1</td>
<td>E2C8</td>
<td>C9D5</td>
<td>C7E3</td>
<td>D6D5</td>
<td>6BC4</td>
</tr>
<tr>
<td></td>
<td>WA</td>
<td>SH</td>
<td>IN</td>
<td>GT</td>
<td>ON</td>
<td>, D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Record word</th>
<th>24</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>4BC3</td>
<td>4B40</td>
</tr>
</tbody>
</table>

This field is converted using the field specification 25-D7.2=6.3 to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>F9F4</td>
<td>F6F8</td>
<td>F8F0</td>
</tr>
</tbody>
</table>
This field describes CSP D4 conversion. The specification is:

\[ m=\text{E}1, j=1, k(P) \]

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:

<table>
<thead>
<tr>
<th>Record word</th>
<th>101</th>
<th>102</th>
<th>103</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>FFF7</td>
<td>68FF</td>
<td>0094</td>
</tr>
</tbody>
</table>

This field is converted using the field specification 25-E7.2=7.2 (P) to the following RPG field.

<table>
<thead>
<tr>
<th>Record word</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>0094</td>
<td>688D</td>
</tr>
</tbody>
</table>

This field type describes CSP A3 conversion, and requires a 40 character translation table. The specification is:

\[ m=\text{F}w \]

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.

<table>
<thead>
<tr>
<th>Record word</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>E6C8</td>
<td>E840</td>
</tr>
<tr>
<td>WH</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**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 L. Disk Storage Unit Conversion Factors

<table>
<thead>
<tr>
<th>No. of</th>
<th>Word</th>
<th>Disk block</th>
<th>Sector</th>
<th>Track</th>
<th>Cylinder</th>
<th>Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>16</td>
<td>320</td>
<td>5,112</td>
<td>20,480</td>
<td>40,960</td>
<td>8,192,000</td>
</tr>
<tr>
<td>Data words</td>
<td>20</td>
<td>320</td>
<td>1,280</td>
<td>2,560</td>
<td>512,000</td>
<td></td>
</tr>
<tr>
<td>Disk blocks</td>
<td>16</td>
<td>64</td>
<td>128</td>
<td>25,600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sectors</td>
<td></td>
<td>4</td>
<td>8</td>
<td>1,600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracks</td>
<td></td>
<td>2</td>
<td></td>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylinders</td>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 These follow the first actual word of each sector, which is used for the address.
### Appendix M. Character Code Set

<table>
<thead>
<tr>
<th>Ref no.</th>
<th>EBCDIC Binary</th>
<th>Hex</th>
<th>Graphics and control names</th>
<th>PTTC/8 1132 Printer hex</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0000 0000 00</td>
<td>00 12 11 0 9 8 7-1</td>
<td>8030</td>
<td>NUL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0001 01</td>
<td>01 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>PF Punch Off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0110 002</td>
<td>02 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>Horiz.Tab</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 0111 003</td>
<td>03 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>Lower Case</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 0110 004</td>
<td>04 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>DELETE Delete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 0110 005</td>
<td>05 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>DEL Delete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 0110 006</td>
<td>06 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>CONSOLE Console</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 0111 007</td>
<td>07 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>L-lowercase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 1000 008</td>
<td>08 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 1001 009</td>
<td>09 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1010 010</td>
<td>10 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 1010 011</td>
<td>11 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1010 012</td>
<td>12 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 1110 013</td>
<td>13 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 1110 014</td>
<td>14 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 1111 015</td>
<td>15 12 11 0 9 8 7-1</td>
<td>8010</td>
<td>U-upperc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: Typewriter output

1. Tabulate
2. Carrier return
3. Shift to black
4. Shift to red

* Recognized by all conversion subroutines

Codes that are not asterisked are recognized by the SPEE subroutine.
The ZIPCO subroutine also recognizes these codes in conjunction with
the appropriate code tables, notably EBHOL and HLEBC.
<table>
<thead>
<tr>
<th>Ref no.</th>
<th>EBCDIC</th>
<th>IBM card code</th>
<th>Graphics and control names</th>
<th>1132 Printer EBCDIC subset hex</th>
<th>PTTC/B hex U-upper case L-lower case</th>
<th>Console printer hex</th>
<th>1403 Printer hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>64*</td>
<td>0100</td>
<td>0000</td>
<td>40</td>
<td>blank</td>
<td>40 **</td>
<td>10 (U/L)</td>
<td>7F</td>
</tr>
<tr>
<td>65</td>
<td>0001</td>
<td>41</td>
<td>12</td>
<td>0 9 1</td>
<td>B010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>0010</td>
<td>42</td>
<td>12</td>
<td>0 9 2</td>
<td>A810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>0011</td>
<td>43</td>
<td>12</td>
<td>0 9 3</td>
<td>A410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>0100</td>
<td>44</td>
<td>12</td>
<td>0 9 4</td>
<td>A210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>0101</td>
<td>45</td>
<td>12</td>
<td>0 9 5</td>
<td>A110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>0110</td>
<td>46</td>
<td>12</td>
<td>0 9 6</td>
<td>A990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>0111</td>
<td>47</td>
<td>12</td>
<td>0 9 7</td>
<td>A050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>1000</td>
<td>48</td>
<td>12</td>
<td>0 9 8</td>
<td>A030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>1001</td>
<td>49</td>
<td>12</td>
<td>8 1</td>
<td>9200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74*</td>
<td>1010</td>
<td>4A</td>
<td>12</td>
<td>8 2</td>
<td>8800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75*</td>
<td>1011</td>
<td>4B</td>
<td>12</td>
<td>8 3</td>
<td>8420</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76*</td>
<td>1100</td>
<td>4C</td>
<td>12</td>
<td>8 4</td>
<td>8220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77*</td>
<td>1101</td>
<td>4D</td>
<td>12</td>
<td>8 5</td>
<td>8120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>78*</td>
<td>1110</td>
<td>4E</td>
<td>12</td>
<td>8 6</td>
<td>80A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79*</td>
<td>1111</td>
<td>4F</td>
<td>12</td>
<td>8 7</td>
<td>8060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80*</td>
<td>1011</td>
<td>50</td>
<td>12</td>
<td>8000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>0001</td>
<td>51</td>
<td>12</td>
<td>11 9 1</td>
<td>D010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>0010</td>
<td>52</td>
<td>12</td>
<td>11 9 2</td>
<td>C810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>0011</td>
<td>53</td>
<td>12</td>
<td>11 9 3</td>
<td>C410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>0100</td>
<td>54</td>
<td>12</td>
<td>11 9 4</td>
<td>C210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>0101</td>
<td>55</td>
<td>12</td>
<td>11 9 5</td>
<td>C110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>0110</td>
<td>56</td>
<td>12</td>
<td>11 9 6</td>
<td>C090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>0111</td>
<td>57</td>
<td>12</td>
<td>11 9 7</td>
<td>C050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>1000</td>
<td>58</td>
<td>12</td>
<td>11 9 8</td>
<td>C030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>1001</td>
<td>59</td>
<td>12</td>
<td>11 9 9</td>
<td>5020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90*</td>
<td>1010</td>
<td>5A</td>
<td>12</td>
<td>11 8 2</td>
<td>4820</td>
<td></td>
<td></td>
</tr>
<tr>
<td>91*</td>
<td>1011</td>
<td>5B</td>
<td>12</td>
<td>11 8 3</td>
<td>4420</td>
<td></td>
<td></td>
</tr>
<tr>
<td>92*</td>
<td>1100</td>
<td>5C</td>
<td>12</td>
<td>11 8 4</td>
<td>4220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93*</td>
<td>1101</td>
<td>5D</td>
<td>12</td>
<td>11 8 5</td>
<td>4120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94*</td>
<td>1110</td>
<td>5E</td>
<td>12</td>
<td>11 8 6</td>
<td>40A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95*</td>
<td>1111</td>
<td>5F</td>
<td>12</td>
<td>11 8 7</td>
<td>4060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96*</td>
<td>0110</td>
<td>60</td>
<td>11</td>
<td>11</td>
<td>4000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>97*</td>
<td>0001</td>
<td>61</td>
<td>11</td>
<td>0 1</td>
<td>3000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>0010</td>
<td>62</td>
<td>11</td>
<td>0 9 2</td>
<td>6810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>0011</td>
<td>63</td>
<td>11</td>
<td>0 9 3</td>
<td>6410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0100</td>
<td>64</td>
<td>11</td>
<td>0 9 4</td>
<td>6210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>0101</td>
<td>65</td>
<td>11</td>
<td>0 9 5</td>
<td>6110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0110</td>
<td>66</td>
<td>11</td>
<td>0 9 6</td>
<td>6090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>0111</td>
<td>67</td>
<td>11</td>
<td>0 9 7</td>
<td>6050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>1000</td>
<td>68</td>
<td>11</td>
<td>0 9 8</td>
<td>6030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>1001</td>
<td>69</td>
<td>11</td>
<td>0 8 1</td>
<td>6020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>1010</td>
<td>6A</td>
<td>12</td>
<td>11</td>
<td>6010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>107*</td>
<td>1011</td>
<td>6B</td>
<td>12</td>
<td>0 8 3</td>
<td>2400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>108*</td>
<td>1100</td>
<td>6C</td>
<td>12</td>
<td>0 8 4</td>
<td>2220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>109*</td>
<td>1101</td>
<td>6D</td>
<td>12</td>
<td>0 8 5</td>
<td>2120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110*</td>
<td>1110</td>
<td>6E</td>
<td>12</td>
<td>0 8 6</td>
<td>20A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111*</td>
<td>1111</td>
<td>6F</td>
<td>12</td>
<td>0 8 7</td>
<td>2060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>0111</td>
<td>70</td>
<td>12</td>
<td>11</td>
<td>E000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>0001</td>
<td>71</td>
<td>12</td>
<td>11 9 1</td>
<td>F010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>0010</td>
<td>72</td>
<td>12</td>
<td>11 9 2</td>
<td>E810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>0011</td>
<td>73</td>
<td>12</td>
<td>11 9 3</td>
<td>E410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>0100</td>
<td>74</td>
<td>12</td>
<td>11 9 4</td>
<td>E210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>0101</td>
<td>75</td>
<td>12</td>
<td>11 9 5</td>
<td>E110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>0110</td>
<td>76</td>
<td>12</td>
<td>11 9 6</td>
<td>E090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>0111</td>
<td>77</td>
<td>12</td>
<td>11 9 7</td>
<td>E050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>1000</td>
<td>78</td>
<td>12</td>
<td>11 9 8</td>
<td>E030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>1001</td>
<td>79</td>
<td>12</td>
<td>11 9 9</td>
<td>E020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>122*</td>
<td>1010</td>
<td>7A</td>
<td>12</td>
<td>11 9 10</td>
<td>E010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123*</td>
<td>1011</td>
<td>7B</td>
<td>12</td>
<td>11 9 11</td>
<td>E000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>124*</td>
<td>1100</td>
<td>7C</td>
<td>12</td>
<td>11 9 12</td>
<td>E020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>125*</td>
<td>1101</td>
<td>7D</td>
<td>12</td>
<td>11 9 13</td>
<td>E030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>126*</td>
<td>1110</td>
<td>7E</td>
<td>12</td>
<td>11 9 14</td>
<td>E040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127*</td>
<td>1111</td>
<td>7F</td>
<td>12</td>
<td>11 9 15</td>
<td>E050</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** Any code other than those defined for 1132 is interpreted by the PRNT1 subroutine as a blank.**
<table>
<thead>
<tr>
<th>Ref no.</th>
<th>EBCDIC</th>
<th>IBM card code</th>
<th>Graphics and control names</th>
<th>1132 Printer hex</th>
<th>Console printer hex</th>
<th>1403 Printer hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1000</td>
<td>0000</td>
<td>B0 12 0 8 1 8020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>0001</td>
<td>81</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>0010</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>131</td>
<td>0011</td>
<td>83</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>0100</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>133</td>
<td>0101</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>0110</td>
<td>86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135</td>
<td>0111</td>
<td>87</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>1000</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>1001</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>1010</td>
<td>8A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>139</td>
<td>1011</td>
<td>8B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>1100</td>
<td>8C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>1101</td>
<td>8D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>1110</td>
<td>8E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>143</td>
<td>1111</td>
<td>8F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>1001</td>
<td>0000</td>
<td>90 12 11 8 1 9020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>0001</td>
<td>91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>0010</td>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>0011</td>
<td>93</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>0100</td>
<td>94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>0101</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>0110</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>0111</td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>152</td>
<td>1000</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>1001</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>154</td>
<td>1010</td>
<td>9A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>1011</td>
<td>9B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>1100</td>
<td>9C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>1101</td>
<td>9D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>1110</td>
<td>9E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>1111</td>
<td>9F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>1010</td>
<td>0000</td>
<td>A0 11 0 8 1 7020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>0001</td>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>162</td>
<td>0010</td>
<td>A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>0011</td>
<td>A3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>164</td>
<td>0100</td>
<td>A4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>0101</td>
<td>A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>0110</td>
<td>A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>0111</td>
<td>A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>1000</td>
<td>A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>1001</td>
<td>A9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>1010</td>
<td>AA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>171</td>
<td>1011</td>
<td>AB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>1100</td>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>173</td>
<td>1101</td>
<td>AD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>1110</td>
<td>AE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>1111</td>
<td>AF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>1011</td>
<td>0000</td>
<td>B0 12 11 0 8 1 F020</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>0001</td>
<td>B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>0010</td>
<td>B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>0011</td>
<td>B3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>0100</td>
<td>B4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>0101</td>
<td>B5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>182</td>
<td>0110</td>
<td>B6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>183</td>
<td>0111</td>
<td>B7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>1000</td>
<td>B8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>1001</td>
<td>B9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>1010</td>
<td>BA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>1011</td>
<td>BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>188</td>
<td>1100</td>
<td>BC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>1101</td>
<td>BD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>1110</td>
<td>BE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>1111</td>
<td>BF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ref no.</td>
<td>EBCDIC ref no.</td>
<td>Binary</td>
<td>Hex</td>
<td>Rows</td>
<td>12 11 0 9 8 7-1</td>
<td>Hex</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>--------</td>
<td>-----</td>
<td>------</td>
<td>-----------------</td>
<td>-----</td>
</tr>
<tr>
<td>192</td>
<td>1100 0000</td>
<td>C0</td>
<td>12 0 1</td>
<td>9000</td>
<td>A</td>
<td>C1</td>
</tr>
<tr>
<td>193*</td>
<td>0001 C1</td>
<td>4567</td>
<td>9800</td>
<td>C2</td>
<td>8800</td>
<td>C2</td>
</tr>
<tr>
<td>194*</td>
<td>0010 C2</td>
<td>12 0 2</td>
<td>8400</td>
<td>C3</td>
<td>73 (U)</td>
<td>C3</td>
</tr>
<tr>
<td>195*</td>
<td>0011 C3</td>
<td>12 0 3</td>
<td>8200</td>
<td>C4</td>
<td>64 (U)</td>
<td>C4</td>
</tr>
<tr>
<td>196*</td>
<td>0100 C4</td>
<td>12 0 4</td>
<td>8100</td>
<td>C5</td>
<td>75 (U)</td>
<td>C5</td>
</tr>
<tr>
<td>197*</td>
<td>0101 C5</td>
<td>12 0 5</td>
<td>8080</td>
<td>C6</td>
<td>76 (U)</td>
<td>C6</td>
</tr>
<tr>
<td>198*</td>
<td>0110 C6</td>
<td>12 0 6</td>
<td>8040</td>
<td>C7</td>
<td>67 (U)</td>
<td>C7</td>
</tr>
<tr>
<td>199*</td>
<td>0111 C7</td>
<td>12 0 7</td>
<td>8020</td>
<td>D1</td>
<td>68 (U)</td>
<td>C8</td>
</tr>
<tr>
<td>200*</td>
<td>1000 C8</td>
<td>12 0 8</td>
<td>7900</td>
<td>C9</td>
<td>79 (U)</td>
<td>C9</td>
</tr>
<tr>
<td>201*</td>
<td>1010 C9</td>
<td>12 0 9</td>
<td>7800</td>
<td>D2</td>
<td>51 (U)</td>
<td>D1</td>
</tr>
<tr>
<td>202</td>
<td>1010 CA</td>
<td>0 9 8 2</td>
<td>A830</td>
<td>D3</td>
<td>52 (U)</td>
<td>D2</td>
</tr>
<tr>
<td>203</td>
<td>1011 CB</td>
<td>0 9 8 3</td>
<td>A430</td>
<td>D4</td>
<td>43 (U)</td>
<td>D3</td>
</tr>
<tr>
<td>204</td>
<td>1100 CC</td>
<td>0 9 8 4</td>
<td>A230</td>
<td>D5</td>
<td>54 (U)</td>
<td>D4</td>
</tr>
<tr>
<td>205</td>
<td>1101 CD</td>
<td>0 9 8 5</td>
<td>A130</td>
<td>D6</td>
<td>55 (U)</td>
<td>D5</td>
</tr>
<tr>
<td>206</td>
<td>1110 CE</td>
<td>0 9 8 6</td>
<td>A080</td>
<td>D7</td>
<td>56 (U)</td>
<td>D6</td>
</tr>
<tr>
<td>207</td>
<td>1111 CF</td>
<td>0 9 8 7</td>
<td>A070</td>
<td>D8</td>
<td>57 (U)</td>
<td>D7</td>
</tr>
</tbody>
</table>

...
$PST3$ address, indicates postoperative error 3-2
$PST2$ address, indicates postoperative error 3-2
$PST1$ address, indicates postoperative error 3-2
$PRLT$ address, indicates preoperative error 3-2
$EXIT$ entry (skeleton supervisor) use by CIL 3-13
$DUMP$ entry (skeleton supervisor) use by CIL 3-13
MSS data files, initializing for use with FORTRAN unformatted
I/O 6-27
$SDUMP$ entry (skeleton supervisor) use by CIL 3-13
$SEXIT$ entry (skeleton supervisor) use by CIL 3-13
SLINK entry (skeleton supervisor) use by CIL 3-14
$SPRT$ address, indicates preoperative error 3-2
$SPST1$ address, indicates postoperative error 3-2
$SPST2$ address, indicates postoperative error 3-2
$SPST3$ address, indicates postoperative error 3-2
$SPST4$ 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
* DUMPFLET (see also DUMPFLET DUP control record) 5-29
* DUMPFLET E (see also DUMPFLET E DUP control record) 5-26
* DUMPFLET E D (see also DUMPFLET E DUP control record) 5-29
* DUMPFILE (see also DUMPFILE DUP control record) 5-28
* DUMPFILE D (see also DUMPFILE D 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
*ICOS (see also ICOS 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 assembler control record) 5-13
*MALIB (see also MALIB 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
*TIME (see also TIME 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
// CPRINT (see also CPRINT 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
appendices
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
ero errors 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) 1-6
The format in which absolute and relocatable programs are punched into cards. In this format, columns 75 through 80 contain the card ID and sequence number.
card input at the RJE work station 10-2
Card Punch, reading the 1442 Model 5 7-5
card read errors, 1442 B-6
Card Read Punch, reading the 1442 Model 6 and 7 7-5
card reader, entering jobs from the 7-12
Card Reader, reading the 2501 7-6
card sequencing and ID field 1-6
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 ID (CIDN), in sector @IDAD 2-5
cartridge copy code (COPY), 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 1-11
CDP data file card format 1-11
CDS data card format 1-9
CDS program
end-of-program card 1-10
format 1-6
ILS header card 1-9
ISS header card 1-8
mainline header card 1-6
subprogram header card 1-7
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 5-1
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 procedure 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 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 5-8
components of disk storage 2-2
computing
file label 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
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 (continued)

** 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
*DUMPFILE, 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 analyzer monitor 3-3
control record analyzer supervisor 3-3
control records 5-1

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

core image loader 3-13

processing of low COMMON 3-14

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, its required subroutines, and its interrupt, CALL and LIBF transfer vectors. This term should not be confused with core image program.

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.

core load construction

assignment of core load origin 3-9

core load, including the setting for index register 3.

conversion during core load build 3-9

contents of 1-5

core image header storage area, restriction on use by FORTRAN

subroutines 3-9

core image loader 3-13

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, 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

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 origin locations assigned by the core load builder 3-9

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 program disk that provides for expansion of the Monitor 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
A collection of data. Also, an area in either the user area or the fixed area in which data is stored.

DATA file

*DEFINE DUP control record 5-45
*DEFINE FIXED AREA examples 5-46
*DEFINE COMMON assembler control record 5-63
*DEFINE FIXED AREA format 5-46
*DEFINE VOID format 5-46
*DEFINE DUP control record
changing the size of the fixed area 5-45
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, *DEFINE COMMON assembler control record 5-63
DELETE DUP control record
deleting information from the UA or FX 5-44
events 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
DEEND 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

DCIP disk maintenance program 4-20
B-field type J-4
code 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
event 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
D-field data format 4-24
X-field type J-7

DCIP program, loading the 9-11

DCIP 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 1-4
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
defining the length of COMMON, *DEFINE COMMON assembler control record 5-63
DELETE DUP control record
deleting information from the UA or FX 5-44
events 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

DEEND 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

DCIP disk maintenance program 4-20
B-field type J-4
code 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
event 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
D-field data format 4-24
X-field type J-7

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 1-4
decimal and hexadecimal disk addresses, appendix K K-1
defective cylinder table (DCYL), in sector @IDAD 2-5
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  
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  
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 1-2  
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 1-2  
disk core image format (DCI) 1-5  
The format in which core image programs are stored on  
the disk prior to execution.  
disk data format (DDF) 1-4  
The format in which a data file is stored in either the  
user area or the fixed area.  
disk system format (DSF) 1-2  
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  
ADDRS 4-8  
COPY 4-7  
DFCNV 4-20  
DISC 4-6  
DLCSB 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, reading the 2310 7-3  
disk storage unit conversion factors, appendix L L-1  
disk system format program 1-2  
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  
DISKL 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 1-2  
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 1-2  
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 1-2  
header 1-3  
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 5-24  
DUMPFILE DUMP control record  
additional field information 5-29  
examples 5-30  
format 5-29  
printing the contents of FLET 5-29

Index X-7
DUMPLET 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 5-34
  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 1-10
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
MODSF A-62
MUP, description A-13
RJE A-28
SGJP A-26
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
  file descriptor DFCNV 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. Pucking 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 1 1-1
    card 1-6
    CDC program card 1-11
    CDD data file card 1-11
    CDS data card 1-9
    CDS program 1-6
    data 1-15
    DCI program 1-5
    DDF data file 1-4
    disk 1-2
    DSF program 1-2
    NCF 1-15
    paper tape 1-12
    PRD print 1-13
    print 1-13
  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
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 TYP$EZ, WR$TY, TYP$0 or WR$TY0 6-47
rules for coding an 6-45
interrupt branch table 6-42
interrupt level subroutine (ILS) 6-42
interrupt level subroutine of DCIP 9-8

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 BS$1 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 BS$1 to core location 500 occurs. Similarly, interrupts on levels 1 through 3 cause BS$1s 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 SIOCT) that is incremented by one upon the initialization of every I/O operation and decremented by one upon completion of the I/O operations.

ISS header card, CDS program 1-8
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
nameO 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 *DUMPFLET 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-1
entry format D-1
IDUMY 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

list of
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 in core 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 1-6
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 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

Index X-13
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) 1-15
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
ameZ 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 NOCAL 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 150 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.
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

patch control records

paper tape formats 1-12

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 8-9

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 1-13

preload operating procedure, card system 8-25

preoperative error trap, skeleton supervisor 3-2

preoperative error waits, ISS subroutine 8-2

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 1-13

print formats 1-13

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

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 LET, *DUMPLET 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 1-3

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 1-1

programmers

tips for assembler language 6-35

assembler language 6-35

programming tips and techniques 6-1

data file processing 6-23

RPG object program considerations 6-52

Index X-15
programming tips and techniques (continued)
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 cartridge samples H-13
processing on two disk drives a file that extends over two cartridge samples H-16
RDREC, system library utility subroutine 4-4

PTUTL
error wait codes B-9
example 9-51
materials needed to use the 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, 2301 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
reading 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 multidiive 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 multidiive 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 * 3-3
reloading 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: GSBE, GBE, or GBCl. 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 GBCl. 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 image of the resident monitor minus the disk I/O subroutine. The resident image resides on the 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
skeloton supervisor 3-2
supervisor 3-2
resident monitor including table of equivalences, appendix G 6-1
REST KB (restore keyboard) key 7-13
restart procedures, RJE 10-11
restore keyboard key (REST KB) 7-13

X-16
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 3-4
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 1-10
system program 8-4
sector header format, LET/FLET D-1
selecting a core location 7-14

Index X-17
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
ILSO2 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
STOREDUP control record
additional field information 5-32
examples 5-32
format 5-31
storing information in WS or UA 5-30
STOREDCI control record
additional field information 5-40
examples 5-41
format 5-39
storing a program in DCI format 5-38
storing programs, size discrepancies in 6-7
STOREDATA 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
STOREDATAI control record
examples 5-38
format 5-37
storing information in WS, UA, or FX 5-37
STOREDATE control record
additional field information 5-36
examples 5-36
format 5-35
storing information in WS, UA, or FX 5-34
STOREMOD 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
*STOREDATE 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 1-1
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 WRTYO 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
X-18
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 1-1
  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 in formation 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 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 operator 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 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 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

Index  X-19
term control record, IBM-supplied system loader 8-2
TEND monitor control record reassigns card or paper tape reader
SYSTEM SYMBOL TABLE assembler control record
tips for FORTRAN programmers 6-48
Tips for assembler language programmers
temporary mode, restrictions on DUP caused by temporary
system loader
error messages A-22
sector break cards used by 1-10
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 multidiive 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 SOCAIs 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
transferring 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
ansfer vector (TV)
A combination 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 sim-
ulate 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
- *STOREDATA1 5-37
- *STOREDATAF 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, LST/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
13"6 Disk Pack 2-1.1
141 1 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
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

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.)
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.

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


This Technical Newsletter, a part of Version 2, Modification 12, of the IBM 1130 Disk Monitor System provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are:

<table>
<thead>
<tr>
<th>Page Range</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front cover, ii</td>
<td>5-19.1, 5-19.2 (added)</td>
</tr>
<tr>
<td>4-11 through 4-14</td>
<td>5-20</td>
</tr>
<tr>
<td>4-23, 4-24</td>
<td>5-39 through 5-42</td>
</tr>
<tr>
<td>5-7, 5-8</td>
<td>5-63 through 5-66</td>
</tr>
<tr>
<td>5-11 through 5-14</td>
<td>6-1, 6-2</td>
</tr>
<tr>
<td>5-17 through 5-19</td>
<td>6-13</td>
</tr>
<tr>
<td></td>
<td>6-13, 6-13.2 (added)</td>
</tr>
<tr>
<td></td>
<td>6-14</td>
</tr>
<tr>
<td></td>
<td>6-14, 6-22</td>
</tr>
<tr>
<td></td>
<td>6-45, 6-46</td>
</tr>
<tr>
<td></td>
<td>6-45, 6-46</td>
</tr>
<tr>
<td></td>
<td>A-5, A-6</td>
</tr>
<tr>
<td></td>
<td>A-15 through A-18</td>
</tr>
<tr>
<td></td>
<td>A-65, A-66</td>
</tr>
<tr>
<td></td>
<td>G-1 through G-18</td>
</tr>
<tr>
<td>9-21</td>
<td>9-22</td>
</tr>
<tr>
<td>9-33, 9-34</td>
<td>9-33, 9-34</td>
</tr>
<tr>
<td>4-11 through 4-14</td>
<td>5-20</td>
</tr>
<tr>
<td>4-23, 4-24</td>
<td>5-39 through 5-42</td>
</tr>
<tr>
<td>5-7, 5-8</td>
<td>5-63 through 5-66</td>
</tr>
<tr>
<td>5-11 through 5-14</td>
<td>6-1, 6-2</td>
</tr>
<tr>
<td>5-17 through 5-19</td>
<td>6-13</td>
</tr>
<tr>
<td></td>
<td>6-13, 6-13.2 (added)</td>
</tr>
<tr>
<td></td>
<td>6-14</td>
</tr>
<tr>
<td></td>
<td>6-14, 6-22</td>
</tr>
<tr>
<td></td>
<td>6-45, 6-46</td>
</tr>
<tr>
<td></td>
<td>6-45, 6-46</td>
</tr>
<tr>
<td></td>
<td>A-5, A-6</td>
</tr>
<tr>
<td></td>
<td>A-15 through A-18</td>
</tr>
<tr>
<td></td>
<td>A-65, A-66</td>
</tr>
<tr>
<td></td>
<td>G-1 through G-18</td>
</tr>
<tr>
<td>9-21</td>
<td>9-22</td>
</tr>
<tr>
<td>9-33, 9-34</td>
<td>9-33, 9-34</td>
</tr>
</tbody>
</table>

A technical change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary Of Amendments

Minor technical changes have been made.

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