File Number S360-36
Order No. GC24-5036-5

DOS

IBM Systems Reference Library

IBM System/360
Disk Operating System:
System Control and System Service Programs

This reference publication describes the IBM System/360 Disk Operating System. The system is a set of control programs and processing programs for IBM System/360. Using IBM 2311 Disk Storage or IBM 2314 Direct Access Storage for on-line program residence, the IBM System/360 Disk Operating System:

- Provides stacked-job programming capability.
- Provides multiprogramming and telecommunications capability.
- Controls all input/output.
- Provides for continuous operation of all programs run in its environment.

Detailed information is given on these major topics.

- System Control Program
  - System Organization
  - Supervisor Functions
  - Job Control Program
- System Service Programs
  - Linkage Editor
  - Librarian
  - Problem Determination

Prerequisite for understanding this publication is a basic knowledge of System/360 machine concepts.

For titles and abstracts of other associated publications, see the IBM System/360 Bibliography, GA22-6822.

Sixth Edition (October 1970)

This edition applies to Release 24 of IBM System/360 Disk Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest System/360 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This edition, GC24-5036-5, is a major revision of, and obsoletes, GC24-5036-4 and Technical Newsletters GN24-5385, GN24-5387, and GN24-5396.

Summary of Amendments

This edition documents Problem Determination Aids (PDAID), Weak External Reference (WXTRN), and the use of the American National Standard Code for Information Interchange (ASCII) mode. A new section, Problem Determination, provides several aids for the user to follow as error recovery procedures. A glossary of new terms is included.

This edition also provides documentation to support the IBM 1270/1275 Optical Reader/Sorters. The IBM 1270/1275 devices will not be available in the United States of America.

Changes to the text, and small changes to illustrations are indicated by a vertical line to the left of the change. Changed or added illustrations are denoted by the symbol • to the left of the caption.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be may be addressed to IBM Corporation, Programming Publications, Department 157, P. O. Box 6, Endicott, New York 13760.

Preface

The first three sections following the Introduction, entitled Supervisor, Job Control, and Initial Program Loader (IPL), describe the control program for the Disk Operating System. These sections are of interest to anyone using the system, including system analysts, programmers, and machine operators. The functions of the Supervisor are discussed, and the detailed Job Control statement formats are given. The macro instructions used to communicate with the Supervisor are discussed fully in the Supervisor and I/O Macros publication listed in this Preface.

The next two sections entitled Linkage Editor and Librarian are of particular interest to persons responsible for maintaining the resident system. These sections describe the Linkage Editor and Librarian programs fully.

The last section, Problem Determination, provides several procedures to follow as error recovery aids.

The appendixes on the diagnostic messages have been deleted. For all messages and their causes, refer to the Operator Communications and Messages manual listed in this Preface.

This edition contains a glossary of new terms.

The publications most closely related to this one are:

IBM System/360 Principles of Operation, GA22-6821.

IBM System/360 Disk Operating System, Data Management Concepts, GC24-3427.

IBM System/360 Disk Operating System, Supervisor and Input/Output Macros, GC24-5037.

IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, GC24-3414.

IBM System/360 Disk and Tape Operating Systems, Concepts and Facilities, GC24-5030.

IBM System/360 Disk Operating System, User's Guide: Control Statement Techniques, GC20-1685.

IBM System/360 Disk Operating System: System Programmer's Guide, GC24-5073.

IBM System/360 Disk Operating System, Operator Communications and Messages, GC24-5074.

The telecommunications publications most closely related to this manual are:

IBM System/360 Disk Operating System, Basic Telecommunications Access Method, GC30-5001.

IBM System/360 Disk Operating System, QTAM Message Control Program, GC30-5004.

IBM System/360 Disk Operating System, Queued Telecommunication Access Method, Message Processing Program Services, GC30-5003.

Contents

INTRODUCTION 9
Disk Operating System Components 9
Control Program 9
System Service Programs 9
Processing Programs 10
Multiprogramming 10
System Configuration 11
Machine Requirements 11
Minimum System Requirements for Multiprogramming 13
Organization of a DOS System Pack . . 13
Control Statement Conventions 13

SUPERVISOR 15
Main Storage Organization 15
I/O Units Control Tables 16
Communications Region 17
Supervisor Functions 19
Storage Protection 19
Interrupt Handling 19
Channel Scheduler 23
Device Error Recovery 24
System Availability Aids 25
DOS Volume Statistics 27
Operator Communication 30
Communication to the Operator 30
Communication from the Operator . . . 31
Single Program Initiation 34
System Operation Without a 1052 . . . 36
System Loader 36
Checkpoint/Restart 37
Normal and Abnormal End-of-Job Handling 38

JOB CONTROL 39
Job Control Functions 39
Prepare Programs for Execution 39
Symbolic Input/Output Assignment . . . 39
Set up Communications Region 42
Edit and Store Label Information . . . 42
Restarting Programs from Checkpoint . 44
Job Control Statements 44
General Control Statement Format . . . 44
Sequence of Control Statements 45
Descriptions and Formats of Commands and Statements 46
Usage of Statements and Commands 46
System I/O Operations 71
Control Statement Effect on I/O Units 73
Work Files Used by System Components . 74
Job Control Statement Example 75

INITIAL PROGRAM LOADER (IPL) 77

LINKAGE EDITOR 80
Stages of Program Development 80
Structure of a Program 80
Types of Linkage Editor Runs 82
Linkage Editor Control Statements . . . 84
Sources of Input 84
General Control Statement Format 84
Control Statement Placement 84
PHASE Statement 85
INCLUDE Statement 88
ENTRY Statement 90
ACTION Statement 91
Phase Entry Point 95
Self-Relocating Programs 95
FGP Programs 95
Linkage Editor Input Restrictions . . 96
Linkage Editor Job Setup 96
Example of Linkage Editor Input and Output 97

LIBRARIAN 99
Core Image Library 99
Relocatable Library 99
Source Statement Library 99
Disk Storage Space Required for Libraries and Directories 100
Librarian Functions 103
Maintenance Functions 113
Service Functions 114
Copy Function 114
General Control Statement Format . . .114
Librarian Functions: Core Image Library 115
Core Image Library: Maintenance Functions 115
Core Image Library: Service Functions 117
Librarian Functions: Relocatable Library 120
Relocatable Library: Maintenance Functions 120
Relocatable Library: Service Functions 123
Librarian Functions: Source Statement Library 125
Source Statement Library: Maintenance Functions 125
Source Statement Library: Service Functions 133
Librarian Functions: Directories . . .136
Reallocation Function 137
Copy Function 138
Library Merge Function 140
Punch Service Function: Special Considerations 144
Condense Maintenance Function: Special Considerations 144
Private Libraries 145
Creation of Private Libraries 146
Maintenance and Service of Private Libraries 146

PROBLEM DETERMINATION 147
Problem Determination Serviceability Aids (PDAID) 147
Environmental Recording, Editing, and Printing Program (EREP) 153
Error Statistics by Tape Volume Utility Programs 154
Stand-Alone Dump (DUMPGEN) 158

APPENDIX A: STANDARD DASD FILE LABELS, FORMAT 1 160

APPENDIX B: STANDARD TAPE FILE LABELS .163

APPENDIX C: OPERATOR-TO-SYSTEM COMMANDS 165

APPENDIX D: JOB CONTROL STATEMENTS . . .173

APPENDIX E: FORMAT OF LANGUAGE TRANSLATOR OUTPUT CARDS AND THE USER REPLACE CARD 177

Format of the ESD Card 177
Format of the TXT Card 177
Format of the RLD Card 177
Format of the END Card 178
Format of the REP (User Replace) Card .178

GLOSSARY OF NEW TERMS 179

INDEX 181

Figures

Figure 1. Organization of a DOS System Pack 14
Figure 2. Main Storage Organization . . 16
Figure 3. Communications Region in Supervisor 17
Figure 4. Flow of Control Between Supervisor and Problem Program During an Interrupt 19
Figure 5. Typical Sequence of Events After a Timer Interrupt Due to Use of SETIME 21
Figure 6. Example of Recorder File Creation 27
Figure 7. Format of Mode 1 Printout of ESTV Error Statistics 29
Figure 8. Sequence of LUBs in Device Table 41
Figure 9. Format of PUB Entry 41
Figure 10. Example of Symbolic Device Assignment 42
Figure 11. Example of LISTIO SYS Output 57
Figure 12. Commands and Statements by Function (Part 1 of 2) 70
Figure 13. Job Control Statement Example 76
Figure 14. Device Code Entries for Device Type Parameter in ADD Statement 79
Figure 15. Linkage Editor Input and Output 81
Figure 16. Three Types of Linkage Editor Operations 83
Figure 17. Placement of PHASE and INCLUDE Statements 85
Figure 18. Map of Main Storage 94
Figure 19. Example of Linkage Editor Input and Output 98
Figure 20. Maintenance Functions (Part 1 of 3) 104
Figure 21. Service Functions (Part 1 of 3) 107
Figure 22. Copy Function (Part 1 of 4) 110
Figure 23. Example of an UPDATE Job Stream 132
Figure 24. Example of an UPDATE Activity Log 132
Figure 25. Example of a Valid Job Setup for the Copy Function 140
Figure 26. Direction of Transfer for Merge Operations 142
Figure 27. Example of Job Set Up to Use the MERGE Function (Part 1 of 2) .143
Figure 28. PDAID Control Statements .149
Figure 29. Core Wrap Method for Fetch/Load, I/O and GSVC Traces 150
Figure 30. F/L Trace Entry in Core Wrap Mode 151
Figure 31. I/O Trace Entry for Core Wrap Mode 151
Figure 32. GSVC Trace Entry for Core Wrap Mode 151
Figure 33. Sample Output for F/L Trace 152
Figure 34. Sample Output for I/O Trace 152
Figure 35. Sample Output for GSVC Trace 153
Figure 36. Standard DASD File Labels, Format 1 (Part 1 of 3) 160
Figure 37. IBM Standard Tape File Label 163
Figure 38. ANSI (American National Standards Institute, Inc.) Standard Tape File Label 164
Figure 39. Job Control Commands (Part 1 of 3) 165
Figure 40. ATTN Commands 168
Figure 41. Single Program Initiation Commands (Part 1 of 4) 169
Figure 42. Job Control Statements (Part 1 of 4) 173

The IBM System/360 Disk Operating System provides operating system capabilities for 16K and larger System/360 configurations that include one or more IBM 2311 Disk Storage Drives or IBM 2314 Direct Access Storage Facilities. Systems above 16K that do not require the expanded functions provided in the larger operating system packages offered by IBM benefit from this 16K package. The system is disk resident, using IBM 2311 or IBM 2314 disk storage for online storage of all programs. Depending on the requirements of the particular application, the system can be expanded to include all processing programs that perform the various jobs of a particular installation, or it can be tailored to a minimum system to control a single program.

Disk Operating System Components

CONTROL PROGRAM

The control program is the framework of the Disk Operating System. It prepares and controls the execution of all other programs. The components of the control program are:

1. Supervisor. The Supervisor handles all input/output operations, interrupt conditions, and other functions for all problem programs. Part of the Supervisor resides in main storage at all times. Processing time is divided between the Supervisor and the program(s) being executed. This is true for the user's programs as well as other IBM-supplied components of the system. Certain functions of the Supervisor are provided by transient routines that remain in disk storage until needed and are then loaded into main storage for execution.

2. Job Control. Job Control runs between parts of a job and prepares the system for execution of all other programs in a batched-job environment. Job Control is loaded by the Supervisor from disk storage whenever needed. For foreground programs operating in other than batched-job environment, Job Control type functions are performed by the single program initiator (formerly the foreground initiator).

3. Initial Program Loader (IPL). The IPL routine loads the Supervisor into main storage when system operation is initiated. IPL also processes certain control statements. To load IPL from disk storage, simply select the address of the disk drive in the load-unit switches on the system console and press the load key.

The control program supervises all input/output functions. Required control program input/output units are:

1. System Residence (SYSRES): system residence unit

2. System Reader (SYSRDR): unit used for Job Control statements

3. System Input (SYSIPT): system input unit

4. System Punch (SYSPCH): system output unit

5. System List (SYSLST): system printer unit

6. System Communication (SYSLOG): medium for operator communication.

These control program input/output units are used by programs operating in either the background or batched foreground partitions.

SYSTEM SERVICE PROGRAMS

The system service programs generate the system, create and maintain the library sections, and edit programs into disk residence before execution. Minimum systems can be built that do not include the system service programs.

The system service programs are:

1. Linkage Editor. The Linkage Editor edits all programs into an area of the resident disk pack. These programs can then be permanently placed in the core image library of the system, requiring only control statements to call them for execution, or they can be stored temporarily in the core image library, executed, and then overlaid by new programs.

2. Librarian. This is a group of programs that maintains and reorganizes the disk library areas and provides printed and punched output from the libraries. Three libraries are used.

   a. Core Image Library. All programs in the system (IBM-supplied and user programs) are loaded from this library by the System Loader routine of the Supervisor.

   b. Relocatable Library. This library stores object modules that can be used for subsequent linkage with other program modules. A complete program of one or more modules can be placed in this library.

   c. Source Statement Library. This library stores IBM-supplied macro definitions and user-defined source statement routines (such as macro definitions) built to provide extended program-assembly capability.

PROCESSING PROGRAMS

All user programs are run within the Disk Operating System environment, using the functions of the control program. Minimum resident packs may consist of:

- Only the control program and one or more user programs, or
- The control program and the Linkage Editor, with user programs loaded and edited from cards or tape into a specified area in disk storage, and then into main storage for execution.

A full system may include user's programs and the following IBM-supplied programs:

- Language Translators: Assembler, COBOL, FORTRAN, RPG, and PL/I (D).
- Sort/Merge.
- Utilities.
- Autotest (2311 support only).
- Problem Determination.

Note: When the control program opens SYSLST assigned to an IBM 1403 printer with a Universal Character Set (UCS) feature, a mode is set to suppress data checks. If, however, the user wishes data checks to be allowed, the buffer must be loaded before execution of the problem program.

MULTIPROGRAMMING

For those systems with main storage equal to or in excess of 24K, Disk Operating System offers multiprogramming support. This support is referred to as fixed partitioned multiprogramming, because the number and size of the partitions are fixed, or defined, during system generation. The size of the partitions may be redefined by the console operator, subsequent to system generation, to meet the needs of a specific program to be executed.

Multitasking

Multitasking is a type of multiprogramming. With multitasking, it is possible to perform multiprogramming within any one or all of the partitions: background, foreground-one, and foreground-two. For multiprogramming users, multitasking extends the capabilities of the Disk Operating System to execute twelve programs rather than three. (For a complete discussion of multitasking, see the Supervisor and Input/Output Macros publication listed in the Preface.)

Background vs Foreground Programs

There are two types of problem programs in multiprogramming: background and foreground. Foreground programs may operate in either the batched-job mode or in the single-program mode. Background programs and batched-job foreground programs are initiated by Job Control from the batched-job input streams. Single-program foreground programs are initiated by the operator from the printer-keyboard. When one program is completed, the operator must explicitly initiate the next one.

Background and foreground programs initiate and terminate asynchronously from each other and are logically independent of each other.

The system is capable of concurrently operating one background program and one or two foreground programs. Priority for CPU processing is controlled by the Supervisor, with foreground programs having priority over background programs. All programs operate with interrupts enabled. When an interrupt occurs, the Supervisor gains control, processes the interrupt, and gives control to the highest priority program that is in a ready state. Control is taken away from a high priority program when that program encounters a condition that prevents continuation of processing until a specified event has occurred. Control is taken away from a lower priority program when an event on which a higher priority program was waiting has been completed. When all programs in the system are simultaneously waiting (that is, no program can process), the system is placed in the wait state enabled for interrupts. Interrupts are received and processed by the Supervisor. When an interrupt satisfies a program's wait condition, that program becomes active and competes with other programs for CPU processing time.

In addition to at least 24K positions of main storage, multiprogramming support requires the storage protection feature.

If the batched-job foreground option is selected when the system is generated, many types of programs can be run as foreground programs. (Specifying the option causes the generation of individual communication regions for each partition.) However, the Linkage Editor and the maintenance functions of the Librarian are restricted to the background partition. (Refer to the Concepts and Facilities publication listed in the Preface for the IBM-supplied programs that can be run in the foreground partition.)

Telecommunications

Disk Operating System includes telecommunications capability. Two access methods are available, Basic Telecommunications Access Method (BTAM) and Queued Telecommunications Access Method (QTAM). BTAM requires at least 24K positions of main storage, but QTAM requires a minimum main storage capacity of 64K.

A BTAM program can be run as either a foreground or a background program. Normally, it is run as a foreground-one program and thus has the highest priority of any program being executed at a particular time.

In a system operating under QTAM, the QTAM Message Control Program must be run in the foreground-one partition. Up to two QTAM Message Processing Programs may be run in either foreground or background partitions.

System Configuration

This section presents the:

1. Minimum system configuration required to operate the Disk Operating System.
2. Features in addition to the minimum (item 1) that can be supported.

The system control programs must always be present in order to execute any other programs.

MACHINE REQUIREMENTS

Minimum Features Required

16K bytes of main storage.

Standard instruction set. see Note 1.

One I/O channel (either multiplexor or selector). See Note 2.

One Card Reader (1442, 2501, 2520, or 2540). See Note 3.

One Card Punch (1442, 2520, or 2540). See Note 3.

One Printer (1403, 1404, or 1443). See Note 3.

One 1052 Printer-Keyboard. If used with a Model 65 or larger system, this should be attached to the multiplexor channel.

One 2311 Disk Storage Drive, or

One 2314 Direct Access Storage Facility.

Note 1: Language translators may require extended instruction sets.

Note 2: Telecommunications require a multiplexor channel and at least one selector channel. Telecommunication devices should not be on the same selector channel as SYSRES.

Optical Reader/Sorter and MICR processing requires at least two I/O channels. If MICR devices are attached to the multiplexor channel, no burst mode devices are supported on the multiplexor channel. MICRs should be attached as the highest priority devices on the multiplexor channel. Single addressing 1270s, 1275s, 1412s, or 1419s are supported on any selector channel, but device performance is maintained only if a selector channel is dedicated to a single MICR device. The Dual Address 1275/1419 is not attachable to selector channels.

MICR processing requires either the direct control feature or the external interrupt feature.

Note 3: One 7- or 9-track 2400-series magnetic tape unit may be substituted for this device. (If 7-track tape units are used, the data-convert feature is required, except when the tape unit is substituted for a printer.)

A disk extent can be substituted for this device if 24K bytes of main storage are available.

Additional Features Supported

Timer feature.

Simultaneous Read-while-Write Tape Control (2404 or 2804).

Any channel configuration up to one multiplexor channel and six selector channels.

Tape Switching Unit (2816).

Storage Protection feature (required for multiprogramming).

Universal Character Set (UCS) feature.

Selective Tape Listing Features (1403) for continuous paper tapes.

Dual Address Adapter (1419 or 1275) to allow more stacker selection processing. Once processing with the Dual Address Adapter is established, 1412s and 1419s or 1270s and 1275s cannot be mixed.

Additional main storage up to 16,777,216 bytes.

Problem programs can request I/O operations on the following devices:

1. 1442 Card Read Punch
2. 2501 Card Reader
3. 2520 Card Read Punch
4. 2540 Card Read Punch
5. 1403 Printer
6. 1404 Printer (for continuous forms only)
7. 1443 Printer
8. 1445 Printer
9. 1052 Printer-Keyboard (for operator communication).
10. 2671 Paper Tape Reader
11. 1017 Paper Tape Reader with 2826 Control Unit Model 1
12. 1018 Paper Tape Punch with 2826 Control Unit Model 1
13. 2311 Disk Storage Drive
14. 2314 Direct Access Storage Facility
15. 2321 Data Cell Drive
16. 2400-series Magnetic Tape Units
17. 2495 Tape Cartridge Reader
18. 1285 Optical Reader (see Note following Item 20)
19. 1287 Optical Reader (see Note following Item 20)
20. 1288 Optical Page Reader

    Note: A combined total of eight 1285 and/or 1287 Optical Readers and/or 1288 Optical Page Readers is supported by the system.
21. 1270 Optical Reader/Sorter[1] (maximum of six including MICR-type devices).
22. 1275 Optical Reader/Sorter[1] (maximum of six including MICR-type devices).
23. 1259 Magnetic Character Reader (maximum of one supported by the system)

---

[1]These devices are not available in the United States of America.

24. 1412 Magnetic Character Reader (maximum number supported depends upon the system configuration)

25. 1419 Magnetic Character Reader (maximum number supported depends upon the system configuration)

26. 1419P primary control unit address on 1275/1419 dual address adapter.

27. 1419S secondary control unit address on 1275/1419 dual address adapter.

28. 7770 and 7772 Audio Response Units

29. Teleprocessing devices specified in the BTAM and QTAM publications referenced in the Preface.

Minimum System Requirements for Multiprogramming

Multiprogramming, using only single program initiator facilities requires 24K bytes of main storage. Multiprogramming with batched-job foreground capability requires 32K bytes of main storage to support a single foreground partition in this mode, and 64K bytes to support both foreground partitions as batched-job processors. Since separate system input/output files are required for batched-job foreground processing, additional disk extents or additional input/output devices are required. Multiprogramming also requires a 1052 printer-keyboard.

ORGANIZATION OF A DOS SYSTEM PACK

The DOS disk resident system may be on a 2311 or a 2314 disk pack. Figure 1 shows the organization of the pack.

Control Statement Conventions

The conventions used in this publication to illustrate control statements are as follows.

1. Uppercase letters and punctuation marks (except as described in items 3 through 5) represent information that must be coded exactly as shown.

2. Lowercase letters and terms represent information that must be supplied by the programmer.

3. Information contained within brackets [ ] represents an option that can be included or omitted, depending on the requirements of the program.

4. Options contained within braces { } represent alternatives, one of which must be chosen.

5. An ellipsis (a series of three periods) indicates that a variable number of items may be included.

6. Underlined elements represent an assumed option in the event a parameter is omitted.

7. SYSmax represents the highest numbered programmer logical unit available for a partition. The largest number of programmer logical units available in the system is 222 (SYS000-SYS221) when MPS=BJF, and 244 (SYS000-SYS243) when MPS=YES or MPS=NO at system generation time. The value of SYSmax is determined by the distribution of the programmer logical units among the partitions.

Name	Starting Address, If Present
IPL Bootstrap Records	Record 1, track 0, cylinder 0
System Volume Label	Record 3, track 0, cylinder 0
User Volume Label	Record 4, track 0, cylinder 0
System Directory	Record 1, track 1, cylinder 0
IPL Loader Program	Record 5, track 1, cylinder 0
System Work Area	Record 1, track 2, cylinder 0
Transient Directory	Record 1, track 5, cylinder 0
OPEN Routine Directory	Record 1, track 6, cylinder 0
Library Routine Directory	Record 1, track 7, cylinder 0
Foreground Program Directory	Record 1, track 8, cylinder 0
Problem Program Phase Directory	Record 1, track 9, cylinder 0
Core Image Directory	Track 0 of Cylinder 1, on a 2311. Track 10 of Cylinder 0 on a 2314.
Core Image Library	Beginning of the first available track following the core image directory.
Relocatable Directory, Optional	Track 0 of the first available cylinder following the core image library.
Relocatable Library, Optional	Beginning of the first available track following the relocatable directory.
Source Statement Directory, Optional	Track 0 of the first available cylinder following the previous library.
Source Statement Library, Optional	Beginning of the first available track following the source statement directory.
Label Cylinder	First full cylinder after the last system library.
Track: Provides Storage For:	
0: Background User Labels	
1: Background PARSTD Labels	
2: Foreground 2 User Labels	
3: Foreground 2 PARSTD Labels	
4: Foreground 1 User Labels	
5: Foreground 1 PARSTD Labels	
6-9, or 6-19: Standard Labels (2311) / (2314)	
Volume Table of Contents (VTOC)	Location assigned by the user.

●Figure 1. Organization of a DOS System Pack

The Supervisor is the control program that operates with problem programs. Control is always given to the active program with the highest priority. Priority to programs in the system has been assigned as follows:

1. Supervisor (highest priority).
2. System operator communication routine.
3. Foreground-one program.
4. Foreground-two program.
5. Background program (lowest priority).

Part of the Supervisor resides in main storage at all times. Certain other routines are kept in the core image library in the resident disk pack and are called into transient areas when needed. The functions performed by the Supervisor are:

- Storage protection (required for multiprogramming)
- Interrupt handling
- Channel scheduling
- Device error recovery
- Collection of tape error statistics by volume
- Error Volume Analysis
- Error Logging and recovery
- Operator communication
- Program retrieval (fetch or load)
- End-of-job handling
- Timer services
- Checkpoint/Restart

All functions except certain interrupt handling (SVC, I/O, and machine check) are available to the problem program by issuing macro instructions. The programmer is not concerned with machine interrupt conditions, since these are handled automatically by the Supervisor.

The user can record machine check interrupt conditions for use as a customer engineering diagnostic aid by specifying the I/O error logging and machine check recording and recovery options at system generation time.

To process ASCII (American National Standard Code for Information Interchange) tape files, the ASCII parameter must be specified in the SUPVR macro at system generation time. The supervisor contains translate tables used to convert ASCII to EBCDIC (on input) and EBCDIC to ASCII (on output). All ASCII tape files are processed in the EBCDIC mode.

The Supervisor also contains a communications region for holding information useful to problem programs and to the Supervisor itself.

The Supervisor is generated from a set of source statements by way of an Assembler run.

Main Storage Organization

The Supervisor occupies the low area of main storage. The transient routines are called into the transient area (overlaying the previous routine in the area) and executed when needed. The area occupied by the background program begins just past the transient area. The background program area must be a minimum of 10K bytes. (IBM-supplied programs such as the Linkage Editor require at least 10K bytes to perform their functions. Certain language translators may require a background area larger than 10K bytes.) Following the background program area is the foreground-two program area. This area must be defined in increments of 2K. (Storage protection requires that main storage be divided into blocks of 2K bytes.) Following the foreground-two program area is the foreground-one program area. As with the foreground-two area, the foreground-one area must be defined in increments of 2K. The minimum size of a foreground area is 0K (zero K); the maximum is 510K. Each foreground area operating in a batched-job foreground mode requires a minimum of 10K bytes. A foreground area operating in single program initiation mode requires a minimum of 2K bytes. The main storage map in Figure 2 shows the relationship between the Supervisor and the problem program areas.

Each foreground area contains a save area (for storing the program name, the old

program status word, and registers), a label area for storing file-label information, and the area for executing the problem program. The save area and the label area are in the low part of the foreground program area.

In a batch-only system, the transient area can have a storage protection key of 0 or 1, dependent upon 2K boundaries. In a multiprogramming environment, the last 500 bytes can be 0 or 1.

If the user specifies the physical transient area (PTA) overlap feature at System Generation time, storage protection of the physical transient area is necessary to prevent destruction of information before the system is finished with it. Since the physical transient area is not considered to be part of the Supervisor for the purpose of storage protection, the user should ensure that it is protected by defining his supervisor end (SEND) address large enough to include the physical transient area.

I/O UNITS CONTROL TABLES

The principal components of the I/O Units Control Tables are the Logical Unit Blocks (LUB), Physical Unit Blocks (PUB), Job Information Blocks (JIB), Tape Error Blocks by unit (TEB), and Tape Error Blocks by Volume (TEBV). These tables, defined when the system is generated, are required for channel scheduling, input/output unit assignment and control, file protection and maintenance of miscellaneous information about jobs, such as multiple I/O assignments and tape error statistics.

Each LUB is two bytes and represents one logical (symbolic) I/O unit. Each LUB references an entry in the PUB. LUBs corresponding to logical units are ordered according to the logical units they represent. For information concerning the ordering of LUBs, see Logical Unit Block (LUB) and Physical Unit Block (PUB) in the Job Control section. The LUBs are grouped into two classes: system logical units and programmer logical units. The number of programmer logical units is a system generation parameter with a maximum of 222. (The former maximum is reduced because system logical units are defined for foreground programs.)

Each PUB is eight bytes and represents one physical I/O unit. Contained in each PUB is such information as the channel and unit numbers of the device, the characteristics of the device, references to the channel queue, and indicators used

	Permanent Storage Locations Used by CPU
	Communications Region
Supervisor Storage Protection Key: 0	EXCP Routine I/O Interruption Routine } Channel Scheduler Start I/O Routine Storage Protection (required for multiprogramming) Supervisor Call Routine Program Check Routine Machine Check Routine External Interruption Routine Timer Services (optional) System Loader (Program FETCH and LOAD) Resident Error Processing Routines Program Information Block (PIB) I/O Units Control Tables (LUB/PUB/JIB/TEB/TEBV)
Transient Areas Storage Protection Key: 0	Open Close Dump Operator Communications Checkpoint End of Job Error Processing Routines Attention Routine
Background Program Area Storage Protection Key: 1 Minimum Size: 10K	Job Control Linkage Editor Librarian Installation Processing Programs
Foreground-two Program Area Storage Protection Key: 2 Minimum Size: 2K for Single Programs 10K for Batched-job programs	Job Control or Single Program Initiator Installation Processing Programs
Foreground-one Program Area Storage Protection Key: 3 Minimum Size: 2K for Single Programs 10K for Batched-job programs	Job Control or Single Program Initiator Installation Processing Programs

Figure 2. Main Storage Organization

by the Channel Scheduler, Supervisor, and Job Control. The PUBs are ordered by the number of the channel to which the various devices are attached. The number of PUBs is a system generation parameter with a maximum of 255.

Each JIB is four bytes and contains LUB or extent information. The number of JIBs is specified by the user in the JIB= parameter of the IOTAB macro at system generation time. There is a minimum of 5, and a maximum of 255 JIBs.

The JIB contains one of the following:

- LUB entry of the standard assignment when a temporary LUB assignment is made.
- PUB pointer for an alternate LUB assignment.
- Extent information when DASD file protection is selected as a supervisor generation option.

Each TEB is six bytes and contains error statistics for one magnetic tape unit. The number of TEBs is specified by the user in the TEB= parameter of the FOPT macro at system generation time.

Each TEBV is 18 bytes and contains error statistics for one magnetic tape volume. The number of TEBVs is specified by the user in the TEBV= parameter of the FOPT macro at system generation time. When both TEB and TEBV are specified, both must specify the same number.

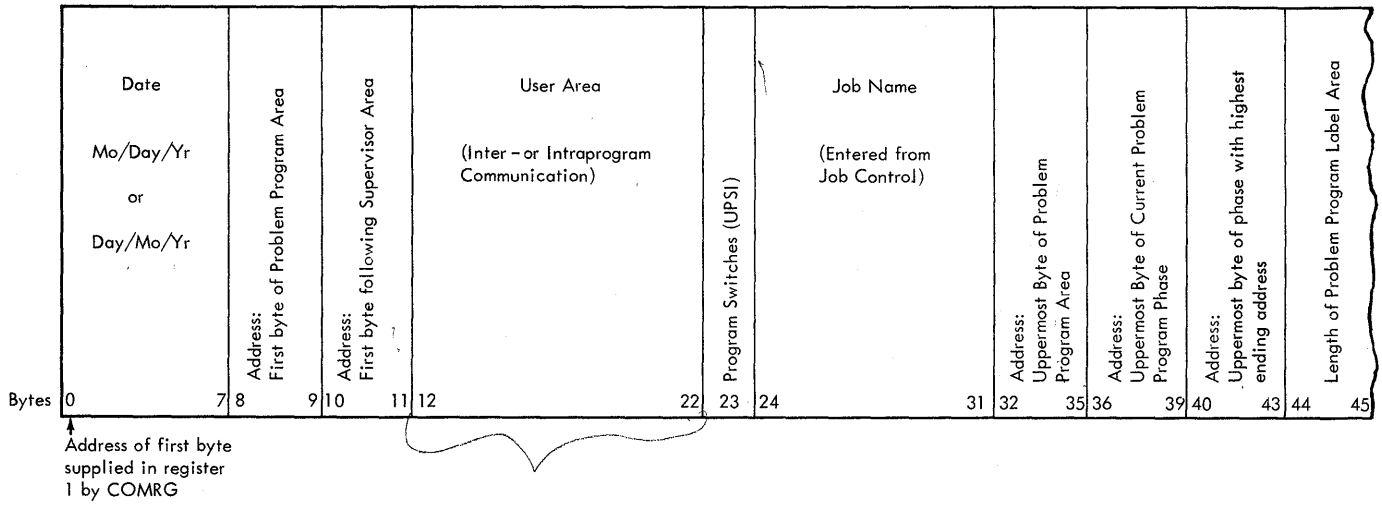
COMMUNICATIONS REGION

The communications region is a storage area within the Supervisor region for use by the Supervisor and problem programs. The MVCOM and COMRG macro instructions are available to allow access to the information contained in this region. Fields in the communications region are addressed relative to the first byte of the region.

If a batched job foreground environment is specified at system generation time, individual communications regions are defined for each of the three partitions. This facility allows each partition to modify its respective communications region by using the MVCOM macro instruction or to access the region by using the COMRG macro instruction.

Figure 3 shows the portion of the communications region that contains information of interest to the user. For a complete layout of the communications region, see the System Programmer's Guide, listed in the Preface.

8 bytes — Calendar date. Supplied from the system date whenever a JOB statement is encountered. In one of two forms: mm/dd/yy or dd/mm/yy where mm is month, dd is day, and yy is year. The calendar date for a partition can be temporarily overridden by a DATE statement.



● Figure 3. Communications Region in Supervisor

2 bytes — Address of first byte of problem program area (PPBEG, see Note). The problem program area follows the second part of the supervisor, composed of the Logical Transient Area, the Physical Transient Area, the CE Area (if any) and the BG Register Save Area. This is the same for all partitions.

2 bytes — Address of first byte following the supervisor area (EOSSP, see Note.) If storage protect is specified, the address is that of the first byte with a storage protect key of 1. If storage protect is not specified, the address is that of the first byte of the problem program area. This is the same for all partitions.

Note: These fields are normally reserved for control program use. A discussion of the relationship of PPBEG, EOSSP, and the end of supervisor macro instruction SEND is given in the System Generation and Maintenance manual listed in the Preface.

11 bytes — User area (for interprogram or intraprogram communications). All 11 bytes set to binary zero when the control statement JOB is encountered.

1 byte — UPSI (user program switch indicators). Set to binary zero when the control statement JOB is encountered. Initialized by UPSI job control statement.

8 bytes — Job name as found in the JOB control statement.

4 bytes — Address of the uppermost byte of the program area. The corresponding value is contained in general register 2 when the first phase of a background or foreground program is fetched.

4 bytes — Address of the uppermost byte of a phase placed in the problem-program area by the last FETCH or LOAD.

4 bytes — Highest ending main storage address of the phase among all phases having the same first four characters as the operand on the EXEC statement. For the background partition only, job control builds a phase directory of these phases. The address value may be incorrect if the program loads any of these phases above its link-edited origin address. If the EXEC statement has no operand, job control places in this location the ending address of the program just link-edited.

2 bytes — Length of batched job or background program label area.

Communications Region Macro Instructions

Macro instructions allow the problem program operating in batched job mode to access the communications region. A brief discussion of these macro instructions follows. Details can be found in the Supervisor and I/O Macros publication listed in the Preface.

COMRG — (Get address of communications region.) Allows the problem program to address information stored in the communications region (obtain date, test switches, etc). The address of the first byte of the region is placed in general register 1.

MVCOM — (Move to communications region.) Allows the problem program to modify the content of the user area and UPSI (bytes 12 through 23) of the communications region. The operand field of the MVCOM macro instruction contains three operands. The first specifies the first communications region byte to be modified. The second specifies the number of bytes to be inserted. The last specifies the address (or a register containing the address) of the bytes to be inserted.

Supervisor Functions

STORAGE PROTECTION

A storage protection key of 0 is set for the Supervisor and for all or part of the transient areas. A key of 1 is set for the background program area. A key of 2 is set for the foreground-two program area. A key of 3 is set for the foreground-one program area.

INTERRUPT HANDLING

An interrupt can be caused by either a program instruction or a machine condition. The Supervisor automatically handles all interrupts so that the programmer need not be directly concerned with them. If, however, the user wishes to record machine check interrupt conditions, he must specify this option at system generation time. In most cases after an interrupt is handled, control returns to the point of interrupt as if no break had occurred in the instruction sequence.

There are five kinds of interrupts:

1. Supervisor call
2. External
3. Program check
4. Machine check
5. Input/output.

Figure 4 shows the flow of control between the Supervisor and a problem program during an interrupt. Control is in the problem program initially. An interrupt occurs, transferring control to the Supervisor. The status of the program is saved in the program old PSW. Depending on the type and reason for the interrupt, control is given to an appropriate handling routine. Upon completion of the routine, the program can be restored to its original condition (via the old PSW). Control is normally given back to the problem program at the point where it was interrupted. The user may have control of program check and external interrupts.

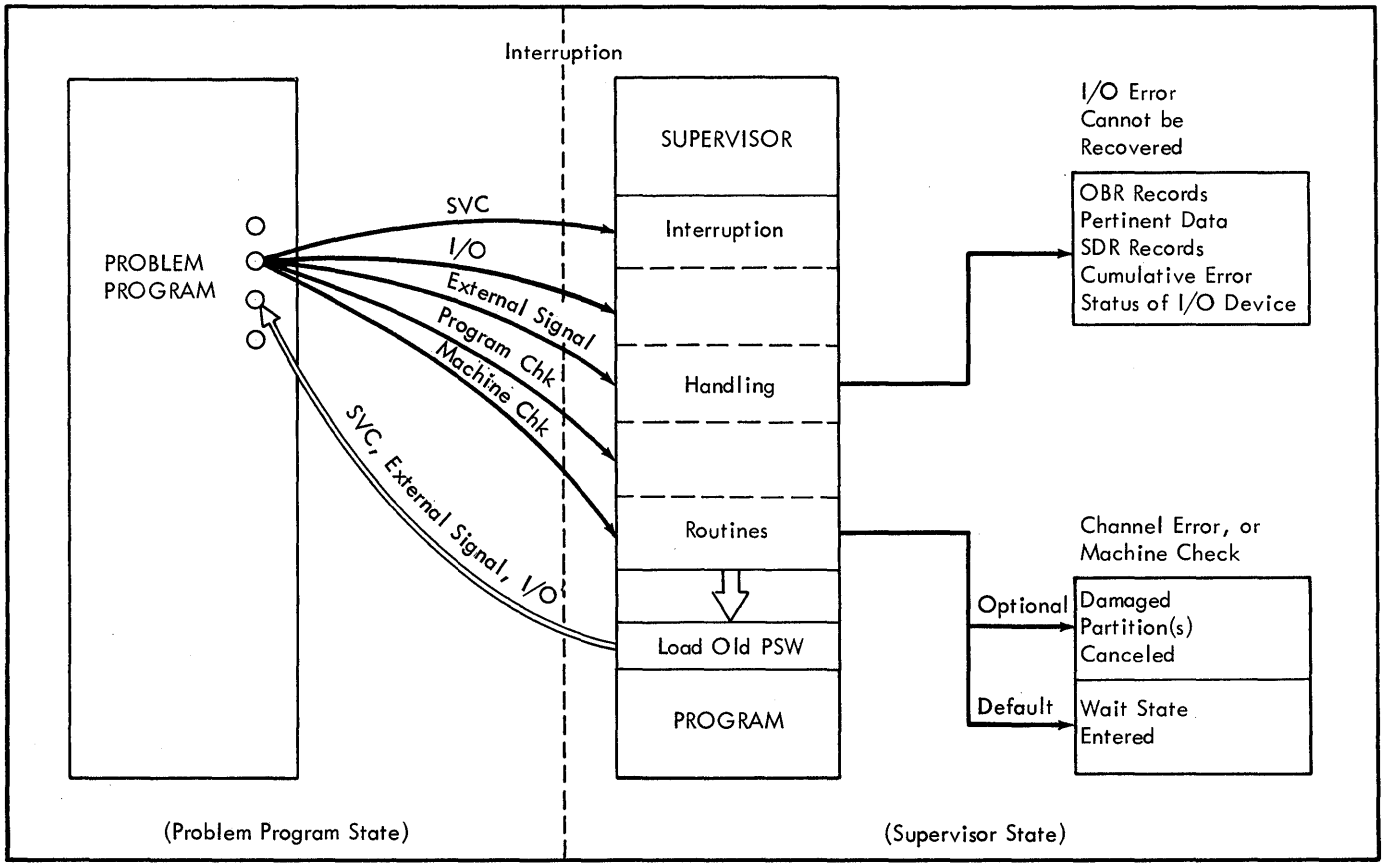


Figure 4. Flow of Control Between Supervisor and Problem Program During an Interrupt

Supervisor Call

The supervisor call interrupt is caused when the SVC instruction is executed. Certain macros use the SVC (supervisor call) instruction to provide communication between the problem program and the Supervisor. The SVC in each macro has a specific interrupt code that indicates to the Supervisor which routine is to be executed. The macro instructions that allow problem programs to have access to control functions, some of which are transient, via an SVC instruction are:

CANCEL To cancel all remaining steps of a job.

CHKPT To cause checkpoints to be taken in a problem program. (Checkpoints may not be taken in batched-job foreground programs if a teleprocessing device is in operation at the time of the CHKPT request. Programs to process ASCII files may not be checkpointed.)

CLOSE To close an input/output file.

COMRG To allow the problem program to address information stored in the communications region (obtain date, set switches, etc).

DUMP To get a printout of main storage and terminate the problem program.

EOJ To indicate the problem program is completed.

EXCP (Execute Channel Program) To request an I/O operation to be performed by physical IOCS.

EXIT To return to the user's main program after the user's handling an external or program check interrupt.

FETCH To load a program from the core image library into main storage for execution.

GETIME To obtain the time of day at any time during program execution.

LBRET To return from the problem program to an OPEN, CLOSE, or end-of-volume routine.

LOAD To load into main storage from the core image library a phase that is not to be executed immediately.

MVCOM To modify the content of the user area in the communications region.

OPEN To open an input/output file for processing.

PDUMP To get a printout of main storage between specified limits.

SETIME To request the Supervisor to interrupt the execution of a program after a specified period of time has elapsed.

STXIT To establish a linkage from the Supervisor to a user routine (program check, operator communication, or interval timer interrupt) or to cancel the use of such a routine.

WAIT To indicate a problem program is in a not-ready state, waiting on a specified event.

Each macro instruction generates a supervisor call interrupt with a specific parameter. The interrupt routine analyzes the parameter and gives control to another routine for the actual handling of the interrupt.

External Interrupt

An external interrupt can be caused by the timer feature, or by the operator pressing the console interrupt key, or by an external signal.

If the proper exits were provided in problem programs operating in the foreground partitions, the 1052 request key can be used to cause an interrupt, through access to the problem program communication routines.

If an interrupt-key or timer interrupt occurs, control is immediately given back to the interrupted program unless the user has provided an address of his own routine through a STXIT macro instruction. When this is the case, control transfers to the address specified.

The timer feature enables the control program to provide three functions:

1. Maintain the time of day which the user can reference at any point within the execution of the problem program.
2. Time-stamp the beginning and end of a job. This information can be used for accounting information and is printed

on the devices assigned to SYSLOG and SYSLST.

3. Enable the user to set the timer for a specified interval of time and either wait on it or to get control at a prespecified address after the time interval has elapsed. The interval timer can be used with only one class of program (background, foreground-one, or foreground-two) at a time.

If the presence of the timer feature was not specified when the system was generated, all timer interrupts are ignored and cause control to return immediately to the interrupted program.

Five macro instructions are provided for use with external interrupts. These macro instructions are TECB, GETIME, SETIME, STXIT, and EXIT. Figure 5 shows a typical sequence of events following an external interrupt.

TECB — Timer event control block. This macro instruction generates a control block for communicating interval timer status to the problem program. TECB can be used in conjunction with the WAIT and SETIME macro instructions.

GETIME — Get time of day. This macro instruction can be used at any point in the execution of a problem program to get the time of day. The value returned to the problem program can be in one of three forms, depending upon the requirements of the user. The time can be in hours, minutes, and seconds, or it can be a binary integer (in seconds), or it can be in units of 1/300 seconds. This macro instruction is useful only if the timer feature is installed and if its presence is indicated when the system is generated.

SETIME — Set interval timer. This macro instruction can be used to request the Supervisor to

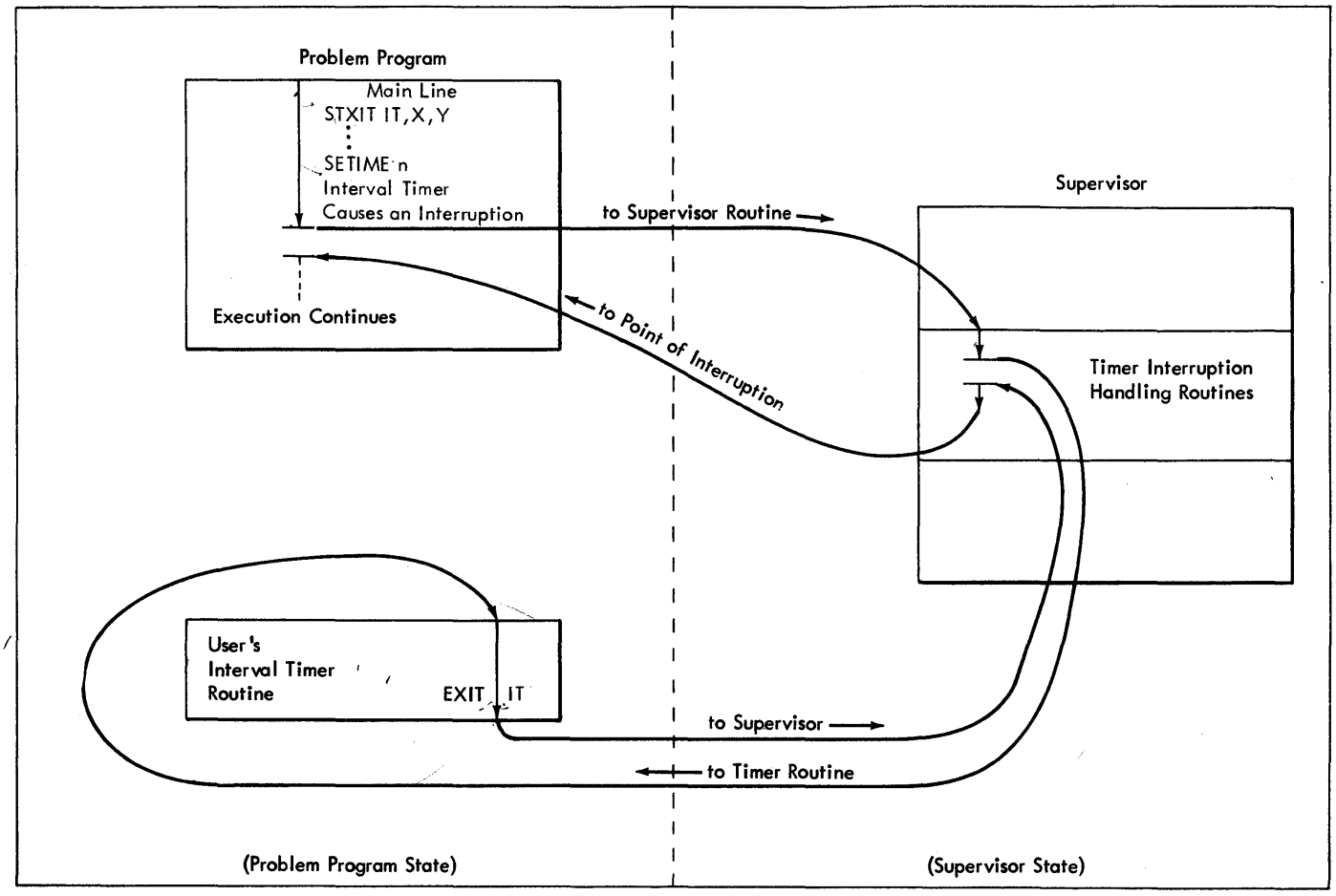


Figure 5. Typical Sequence of Events After a Timer Interrupt Due to Use of SETIME

interrupt the execution of a problem program after a specified time limit has elapsed or to allow the problem program to wait for the time interval to elapse. This macro instruction is useful only if the timer feature is installed and if its presence is indicated when the system is generated.

STXIT — Set linkage to user interrupt routine. This macro instruction can be used to establish a linkage from the Supervisor to a user routine. It can also be used to cancel the linkage to such a routine. This macro instruction can be used for timer, operator communications, and program-check interrupts.

EXIT — Set exit. This macro instruction is used with the STXIT macro instruction to return from the user's routine to the point of interrupt.

A complete description of these macro instructions is supplied in the Supervisor and I/O Macros publication listed in the Preface.

Program Check

When a program check occurs, each program can select one of these options:

1. Abort. The job being executed is terminated and a message output on SYSLOG and SYSLST describes the cause of the termination of a program.
2. Dump and Abort. This is requested by a system generation parameter, or by use of the DUMP option in the OPTION control statement. In addition to a message, all registers and main storage are printed on SYSLST. The job is then terminated.
3. Transfer to User Routine. If the address of a subroutine is supplied by the user by way of the STXIT macro instruction, the program-check interrupt routine branches to that subroutine when an appropriate interrupt occurs. The user routine can determine the cause of the interrupt. Return to the point of interrupt is possible by the EXIT macro instruction. This facility is a system generation option.
4. Program Mask in PSW. The program mask bits of the PSW (Program Status Word) are initially set to zero. If the user wishes to enable a program interrupt, he can change this configuration by the Set Program Mask (SPM) instruction. The program mask bits are reset to zero after each job step and after each FETCH of a problem program phase.

Machine Check

A machine-check interrupt results from a machine malfunction. The machine-check interrupt places the system in the wait state with a unique message code (0S) in bytes 0 and 1 of main storage. The SEREP program, supplied to IBM customer engineers, can then be run. The system can be restarted only through an IPL procedure.

When the machine check recording and recovery (MCRR) function is specified, these conditions, which formerly prevented the system from continuing to process, can be overcome. MCRR records and analyzes pertinent data, and cancels the damaged partition(s).

A machine-check interrupt also results from an I/O channel failure. The message code 1S is in bytes 0 and 1 of main storage.

Input/Output Interrupts

An input/output interrupt can be caused by:

1. I/O completion (Channel End). This is the end of transfer of data into or out of main storage or completion of a control operation.
2. Device available (Device End). A device that was busy or not ready is now available for use.
3. Control unit available (Control Unit End). A control unit that was busy is now available for use.
4. I/O attention. This results from pressing the request key on the 1052.

When one of these conditions is detected, control transfers to the Channel Scheduler. (For nonteleprocessing devices, a program-controlled interrupt (PCI) is ignored by the Channel Scheduler.)

CHANNEL SCHEDULER

The Channel Scheduler functions are:

1. Schedule I/O requests on each channel (queuing).
2. Start input/output operations.
3. Handle I/O interrupts--normal completion of data transfer, error detection, end-of-file detection, attention (1052).
4. Perform error detection and correction.
5. Detect end-of-job or end-of-job-step control statements on SYSRDR and SYSIPT.
6. Monitor DASD channel programs for file protection and address continuity for disk system input/output.
7. Optionally provide seek separation for DASD channel programs. Any seek that causes disk arm movement is separated from its channel program and executed separately. Channel time during the execution of that seek is available for scheduling other I/O activity on that channel. In a multiprogramming environment, this feature has particular significance when the different partitions have a mix of I/O requests for a single channel, and DASD devices (such as the 2314 or multiple 2311s) are on the channel.

I/O devices in the System/360 are attached to channels rather than directly to the CPU. A channel provides a path for data transfer between the CPU and the I/O device and allows I/O operations to be overlapped with the CPU processing the I/O operations on other channels. That is, instructions can be executed simultaneously with data movement in one or more input/output channels. For instance, at a given point in time, one channel may be reading data from a Direct Access Storage Device (DASD), another channel may be writing data on a printer, and a previously read record may be being processed. This is referred to as read/write/compute overlap.

The two types of channel in this system are: selector channels and the multiplexor channel. The selector channels allow I/O operations for devices on these channels to overlap with CPU processing and I/O operations on other channels. On the multiplexor channel, tape and DASD I/O operations cannot overlap with other I/O operations on the same channel. On System/360 Models 30 and 40, tape and DASD devices operating on the multiplexor channel must not overlap with processing. Card, printer and other low-speed (byte interleave mode) I/O devices on the multiplexor channel can overlap with each other, with CPU processing, and with I/O operations on other channels. Thus, greater throughput can be achieved if high-speed devices (tape and DASD) are attached to selector channels, and low-speed devices (card and printer) are attached to the multiplexor channel.

Overlapping I/O operations with CPU processing is inherent in the design of the machine and the Channel Scheduler. However, achieving maximum overlapping also partially depends on the problem program. For instance, if overlap is desired in tape or DASD operations, the problem program should provide for two I/O areas (or buffers). This allows data to be read into, or written from, one I/O area while records are being processed in the other area. Certain devices, however, have buffers built into the device (1403) and require only one I/O area in main storage to achieve overlap. The use of multiple I/O areas and separate work areas is discussed more fully in the Data Management publication listed in the Preface.

All requests for I/O operations are handled by the Channel Scheduler. When a request is received and the affected channel and device are not busy, the requested operation starts and control passes back to the problem program. If the channel or device is busy, the request is placed at the end of a list (or queue) of I/O requests, and the operation is performed as soon as all previous requests have been handled. (Separate queues are maintained for each device.)

The Channel Scheduler also handles all I/O interrupts. If the interrupt indicates the normal end of an I/O operation (channel end and no errors), the Channel Scheduler posts completion, and removes the request from the queue. It then examines the queues for the affected channel or subchannel. If the queues are empty, control returns to the problem program at the point of interrupt. If instead a request is pending, the Channel Scheduler starts the I/O operation and then returns to the problem program. Requests for devices for which device-end interrupts are outstanding cannot be serviced until the device end is received. These requests are bypassed when the Channel Scheduler is selecting an I/O operation to be started. As an example, for a 1403 Model N1, channel end is received as soon as the buffer is completely loaded (about 2ms), but device end is not received until completion of the print operation (55ms).

The Channel Scheduler detects the following specific status conditions:

1. Wrong Length Record (WLR)

2. Unit Exception

3. Channel and device errors.

Wrong-length record and unit exception are treated as normal conditions. They are posted to the user's CCB along with the other Channel Status Word information (residual count, status bytes, and CCW address). The physical IOCS user is responsible for checking and handling these conditions.

If an error is detected, the Channel Scheduler passes control to the appropriate device error recovery routine, which takes appropriate action: retry, operator intervention, notify problem program, or terminate job.

For certain devices (1052), the operator can initiate an I/O operation. To do so, he presses the request key on the device. When the Channel Scheduler detects an attention status condition, it passes control to a message processing routine.

A problem program can perform I/O operations in two ways:

1. The problem program can issue physical I/O macro instructions directly.

2. The problem program can use logical IOCS, which in turn issues the physical I/O macro instructions.

Physical I/O Macro Instructions

The physical I/O macro instructions are:

EXCP (Execute Channel Program.) This macro instruction communicates directly with the Channel Scheduler to request that an I/O operation be started. When the EXCP macro instruction is used, the problem program must supply the appropriate channel program consisting of channel command words (CCWs).

WAIT This macro instruction suspends program operation until an I/O operation (referenced in the WAIT macro instruction) is complete. The problem program must use this macro instruction at the point where processing cannot proceed until the I/O operation is complete. For instance, a problem program may issue the EXCP macro instruction to read a DASD block. At the point where the program needs the block for processing, a WAIT macro instruction must be issued. The instructions generated from this macro test a program switch to determine if the operation has been completed, and give control to the Supervisor if it has not been completed. The Supervisor places the program in the wait state until the operation is completed, and gives control to a ready lower priority program, if one exists. The completion of the operation causes an I/O interrupt to the Channel Scheduler. The program is taken out of the wait state, the switch is set to show the completion, and control returns to the problem program.

CCB (Command Control Block.) This declarative macro instruction generates a command control block for a channel program to be executed. The command control block contains information required by the Channel Scheduler to execute the EXCP and WAIT macro instructions. The block is used to pass information between the problem program and the Channel Scheduler, such as symbolic I/O unit address, channel program address, status of the operation, action to be taken in the event of an error, etc.

A complete description of these macro instructions is supplied in the Supervisor and I/O Macros publication listed in the Preface.

DEVICE ERROR RECOVERY

Each I/O device or class of I/O devices has a unique device error recovery routine. The appropriate routine is entered from the Channel Scheduler upon detection of an error. All these routines have one function in common. That is, an attempt is made to recover from the error. This may be by programming (rereading tape) or by operator action (2540 not ready).

If recovery is not possible, the following choices are available, where applicable.

1. An error can be ignored.

2. The job can be terminated.

3. The problem program can take action (an exit to a user routine is allowed).

4. The record in error can be bypassed.

Depending on the type of error, the type of device, and whether logical IOCS is used, some or all of the options are available. Choices 3 and 4 are available only through logical IOCS. In the absence of any other options, only choice 2 is available. Tape error statistics by unit can be printed on the printer-keyboard, and tape error statistics by volume can be printed on the printer-keyboard or collected on disk (for later transfer to tape or printer), if such statistic recording was specified when the system was generated. Another option is provided at system generation time to notify the operator by a message on the printer-keyboard when a specified number of temporary read or temporary write errors has been exceeded on a volume.

SYSTEM AVAILABILITY AIDS

The I/O error logging (OBR/SDR), machine check recording and recovery (MCRR), and IBM 2715 Transmission Control Error Recording features increase system availability. These features support all devices supported by the system. If these features are desired, the user specifies the options at system generation time. For teleprocessing devices, the user must also specify the OBR/SDR option in his BTAM or QTAM program. (See the DOS QTAM Message Control Program and the DOS Basic Telecommunications Access Method publications listed in the Preface.) The features are generated as part of the Supervisor. The user need only allocate a file once for the collection of error records. No further intervention is required. The error records can be displayed by the Environmental Recording, Editing, and Printing (EREP) program and used as a diagnostic aid. (See the Problem Determination section.) Certain conditions that formerly prevented the system from continuing to process can now be overcome by using these features. These features require a system with a minimum of 24K bytes of storage.

I/O Error Logging

I/O error logging is an additional option to the Supervisor that can be specified at system generation time in the SUPVR macro. The error logging option includes three features:

1. Outboard Recorder (OBR). When an I/O error occurs that cannot be retried or is not corrected after a standard number of retries, OBR records pertinent data and stores it in a recorder file on the logical unit SYSREC (the operand used in ASSGN and EXTENT statements). The file name of the recorder file is IJSYSRC (the operand used in the DLBL statement). The information contained in the OBR record includes channel unit address, device type and characteristics, date and time of day, job name or program name, channel status word, logical unit, first and failing CCWs, and volume ID (tape only). OBR records are not created for conditions that are considered to be operator or programmer errors such as intervention required, command reject, or invalid seek.

2. Statistical Data Recorder (SDR). SDR records the cumulative error status of an I/O device. The recorder file normally contains one SDR record for each I/O device. However, when specifying the I/O error logging option, the user may increase the number of SDR records by specifying this at system generation time. The SDR feature also contains limited error recording capabilities for unsupported devices. The information contained in the SDR record includes channel unit address, and device type and characteristics. Each SDR record also contains 16 two-byte counters, each representing an error condition to be counted for the device. SDR retains 16 half-byte counters in main storage that correspond to the error counters in the SDR record on the recorder file.

3. IBM 2715 Transmission Control Error Recorder. IBM 2715 error records are recorded in the OBR portion of the recorder file.

When an I/O error occurs, the counters in main storage are updated. Whenever any one of the 16 counters in main storage is filled, the contents of all 16 counters are added to the counters in the SDR record on the recorder file for that device. When the Record On Demand (ROD) command is given, all counters for all nonteleprocessing devices on the

recorder file are updated. When an OBR record is written onto the recorder file, the SDR record on the recorder file for the device in error is also updated. When the SDR record is updated, the main storage counters for that device are reset.

Machine Check Recording and Recovery (MCRR)

If the MCRR option is specified at system generation time in the SUPVR macro, MCRR records pertinent data after a machine check or a channel inboard error (channel control check, interface control check, or channel data check) has occurred. MCRR analyzes this data and cancels the damaged partition or partitions. No attempt is made to retry on any error involving this function.

After a channel failure occurs, all devices that are active on the channel are considered to be damaged. Any partition that has an I/O interrupt outstanding on these devices is canceled. In some cases, it can be determined that the channel failure caused damage to a particular device. In that case only that partition having an I/O interrupt outstanding on the damaged device is canceled.

A CPU machine check causes the interrupted partition to be canceled, unless the partition was the Supervisor. In this case the system enters an uninterruptable wait state with a unique message code (0S) in bytes 0 and 1 of main storage.

The MCRR function is CPU dependent and specific definitions of what constitutes channel failures or device failures differ for each model. MCRR is only available for System/360 Model 30, Model 40, and Model 50.

The MCRR record formats differ for machine check interruptions and for channel inboard errors. Information in the machine check record includes record entry type, CPU model number, date and time, job name, machine check old PSW, and content of general registers. Information in the channel inboard error record includes record entry type, CPU model number, date and time, job name, failing CCW, and channel status word.

Recorder File

A recorder file must be created using the file definition statements of the system, when either the I/O error logging or the MCRR function is specified. The file definition statements must be preceded by the // OPTION STDLABEL statement to ensure that they are retained on the standard label section of the label cylinder on SYSRES.

To create a recorder file: Assign SYSREC after IPL but before the first job. SYSREC must be assigned to a disk device that is always on line, such as the system resident disk. Add the necessary file definition statements to the standard label deck and build the standard label portion of the label cylinder. Do not include a JOB card, until all information applicable to SYSREC is supplied. Instruct the system to create the recorder file (SET RF=CREATE). This operand in the SET command is accepted only by Job Control, so that the SET command must be given twice (once for date at IPL, and once for recorder file after IPL). The file must be defined as an extent on an IBM 2311 or 2314 disk device. Split cylinder cannot be used. The recorder file requires a minimum of two tracks.

Once the file is created, no further operator intervention is required. On subsequent IPLs the system opens the recorder file and continues updating it.

Whenever the system is shut down, the operator must issue the Record on Demand (ROD) command to ensure that statistical data in core storage is recorded on the Recorder File. The command ROD has no operand. BTAM and QTAM use their own separate methods of updating all disk counters during closedown or cancel.

Recording on the Recording File is suppressed when the EREP program is executed.

Figure 6 shows an example of recorder file creation. The recorder file begins at cylinder 170 and is 43 tracks long. The recorder file is created when the // JOB NAME card is processed.

```
ADD
 .
 .     If not in system generation
 .
SET    DATE=04/03/68
ASSGN
 .
 .     If not in system generation
 .
ASSGN  SYSREC,X'190'
SET    RF=CREATE
// OPTION STDLABEL                    }   Submit with the
// DLBL IJSYSRC,'DOS RECORDER FILE'   }   rest of the
// EXTENT SYSREC,,,,1700,43           }   STDLABEL statements
// JOB NAME
 .
 .     Continue with the normal job stream
 .
```

Figure 6. Example of Recorder File Creation

Each track on the recorder file can contain the following number of error records.

Type of Error		2311	2314
SDR		29	38
OBR		25	40
MCRR	Channel Inboard		
	Model 30	25	40
	Model 40	6	10
	Model 50	6	10
	CPU		
	Model 30	8	13
	Model 40	5	8
	Model 50	5	8
2715		25	40

DOS VOLUME STATISTICS

A major factor affecting the quality of an operating system is the condition of the volumes stored on a magnetic medium, such as tape or disk. Such media are subject to contamination from dust, foreign materials, fingerprints, and particles of oxide coating.

Because of these environmental factors, it is desirable to record the number of read and write errors occurring on each tape volume. By monitoring the error rate, it is possible to judge the condition of a volume and to take remedial action against environmental contaminants.

Read and write errors per volume for IBM 2400 series tape units can be monitored by a facility called DOS Volume Statistics. This facility has two options: Error Statistics by Tape Volume (ESTV), and Error Volume Analysis (EVA).

Error Statistics by Tape Volume provides the user with a set of tape volume error data, which includes the time of day the errors occurred, the unit on which the volume was mounted, tape density, and other statistics necessary to evaluate the data.

The user has the option of specifying, at system generation time, whether to record the data on the direct access storage device (DASD) or on the console typewriter (SYSLOG). If DASD is chosen, the data can be retrieved by executing ESTV Dump File Program (see Problem Determination).

Error Volume Analysis produces a message to the operator (at the console typewriter) when a certain number of temporary read or temporary write errors occurs on the tape volume currently in use.

The user can specify either or both of these options when the system is generated.

Error Statistics by Tape Volume (ESTV)

ESTV collects data on tape errors by volume for any tape volumes used by the system. Although DOS itself does not require it, the ESTV program requires that each user program contain an OPEN(R) statement if he wishes to collect volume statistics.

Specifying ESTV at system generation time causes the system to collect the following set of records for each tape volume whenever the volume is in use.

Volume serial number of standard labeled volumes (blank for nonstandard and unlabeled volumes).

Date this set of records was collected.

Time of day this volume was closed (time the record was collected).

Address of the unit on which the volume was mounted and the channel to which the unit was attached.

Number of temporary read errors that occurred while the volume was open.

Number of temporary write errors that occurred while the volume was open.

Number of permanent read errors that occurred while the volume was open.

Number of permanent write errors that occurred while the volume was open.

Number of noise blocks encountered (records less than 12 bytes on a read operation, or less than 18 bytes on a write operation).

Number of erase gaps (three and one-half-inch lengths of erased tape) encountered.

Number of cleaner actions (passing the record in error back and forth under a cleaner blade) taken while trying to correct read errors.

Number of START I/Os issued to the tape (does not include SIOs issued for or during error recovery).

Bit density of the volume (in bits per inch for 7-track tape, and the designation 8/1600 for 9-track tape).

Block length of each record if the volume has fixed-length blocked records. When the type of record is undefined or variable length, or when the program terminates abnormally (ABEND), a 0 appears in the space allocated for block length. A 0 also appears when physical IOCS is being used.

Note 1: The temporary error counter is incremented whenever a data check error is detected. If the error is permanent, the permanent error counter is incremented. However, the temporary error counter is not decremented by permanent errors, and therefore contains the sum of true temporary errors and of permanent errors.

Note 2: The cleaner action counter is not incremented during read-opposite recovery.

ESTV OUTPUT MODES: Two modes of operation for ESTV are available, Mode 1 and Mode 2. They provide two different standard output formats. The user selects the desired mode (at the time the system is generated) in the FOPT system generation macro. The mode selected determines the method in which the collected statistics will be written.

Mode 1: Mode 1 formats the ESTV records and records them on a system direct access storage device in a data set named ESTVFLE. ESTVFLE may later be dumped to a tape or printed on the printer attached to the system by the system control program ESTVUT (see ESTV Dump File Program (ESTVUT) under Problem Determination).

Figure 7 shows the format of the Mode 1 printout of the ESTV error statistics when dumped to the printer by ESTVUT (shown as two lines instead of one because of space restrictions):

VOLUME SERIAL	DATE	TIME OF DAY	CHANNEL /UNIT	TEMP READ	TEMP WRITE	PERM READ
xxxxxx	yr/day	hr.mn.sc	cuu	nnn	nnn	nnn

PERM WRITE	NOISE BLOCKS	ERASE GAPS	CLEANER ACTIONS	SIOs USAGE	TAPE DENSITY	BLOCK LENGTH
nnn	nnn	nnn	nnn	nnnnn	nnn	nnnnn

Figure 7. Format of Mode 1 Printout of ESTV Error Statistics

The volume serial is the volume accessed when the error is recorded if standard labeled tapes are used. Otherwise, this field is blank.

The date is given in the form yr/day, where:

yr is a two-digit number representing the year (for example, 70 represents the year 1970).

day is a three-digit number representing the sequential day (Julian day) of the year (for example, 032 represents February 1).

The time of day is given in the form hr.mn.sc, where:

hr is a two-digit number representing the hour of the day on a 24-hour clock (for example, 18 represents 6 p.m.).

mn is a two-digit number representing the number of minutes after the hour.

sc designates the number of seconds after the minute.

The channel/unit designation is given in the form cuu.

The next four fields give the number of errors (type indicated by heading) that occurred while the volume was open.

The next field is a count of the noise blocks (blocks of 12 bytes or less for read operations, or 18 bytes or less for write operations) encountered while the volume was open.

The number of erase gaps while trying to write on the tape, and the number of cleaner actions taken, are the next two fields in the record.

A count of the START I/Os encountered while the volume was in use is the next field.

The tape density, in bits per inch for 7-track tape, and designated 8/1600 for 9-track tape, is in the next-to-last field.

Block length (the last column) is the length of the blocks when records of fixed-length are accessed on the volume. For volumes not having fixed-length records, or when the program terminates abnormally, a 0 is put into that record field. A 0 is also put into that field when physical IOCS is being used.

The headings correspond to the items that are collected into the ESTV record.

Mode 2: Mode 2 prints the ESTV data collected at the console typewriter (SYSLOG) each time a particular volume is ended by CLOSE, EOJ, EOV, or ABEND. When Mode 2 is the selected method of writing ESTV records, a subset of the Mode 1 ESTV record is printed as follows:

```
4E00I xxxxxx cuu TW=nnn TR=nnn NB=nnn
 PW=nnn PR=nnn SIO=nnnnn
```

where:

xxxxxx=serial number of standard label volume (blank when nonstandard or unlabeled volume is being used).

cuu=channel/unit address

TW=nnn Number of temporary write errors.

TR=nnn Number of temporary read errors.

NB=nnn Number of noise blocks (records less than 12 bytes in length on a read operation or less than 18 bytes in length on a write operation).

PW=nnn Number of permanent write errors.

PR=nnn Number of permanent read errors.

SIO=nnnnn Number of START I/Os.

After the message is printed, the error counters for that volume are reset to zero.

Note: The temporary error counter is incremented whenever a data check error is detected. If the error is permanent, the permanent error counter is incremented. However, the temporary error counter is not decremented by permanent errors, and therefore contains the sum of true temporary errors and of permanent errors.

Error Volume Analysis (EVA)

Specifying EVA at system generation time sends a message to the system operator when a predetermined number of temporary read errors or temporary write errors is exceeded on a currently accessed tape volume. The user must specify (in the FOPT system generation macro) the number of errors to be reached before the message is sent to the console, if EVA is to be in effect.

EVA can be used with both labeled and unlabeled tape volumes.

The message EVA sends to the console contains the number of temporary read errors, temporary write errors, and START I/Os, the physical unit, and if standard labeled tape is used, the volume serial number or identification. The message format follows:

```
|4E10I xxxxxx cuu TR=nnn TW=nnn SIO=nnnnn |
```

where:

xxxxxx=serial number of standard label volume (blank when nonstandard or unlabeled volume is being used).

cuu=channel/unit address (physical unit)

TR=nnn Number of temporary read errors.

TW=nnn Number of temporary write errors.

SIO=nnnnn Number of START I/Os.

Either the TR=nnn or TW=nnn field contains one more than the predetermined error threshold specified in the FOPT macro. Reaching the error count when notification is sent to the system operator does not cause any interruption in the execution of the problem program. When using an unlabeled or nonstandard labeled tape, the system operator should note the volume identification of the volume in use when the message is received so he can monitor it.

OPERATOR COMMUNICATION

Communication with the operator is through use of full-text messages issued via the IBM 1052 Printer-Keyboard. Two-way communication is possible: from the system to the operator and from the operator to the system.

The Supervisor permits:

1. Full-text messages to the operator. These messages are either information only or indications of required operator action.
2. Operator-initiated instructions to the control program.
3. Communication between the operator and the problem program.

COMMUNICATION TO THE OPERATOR

The control program communicates with the operator by issuing messages on SYSLOG, normally assigned to the IBM 1052 Printer-Keyboard. If no communication with the system is required, an I indicator is included in the message. If an operator action or reply is required, an action indicator A or D is included in the message. The program issuing the message waits until the operator keys in a response.

Each system-to-operator message consists of a 2-character program identifier (if multiprogramming support is provided), a four-character message code, a one-character operator action indicator, at least one blank, and the message itself. The first character of the message code is 0 for the Supervisor; 1 for Job Control; 2 for the Linkage Editor; 3 for the Librarian and EREP; 4 for logical IOCS, PDAID, DUMPGEN, and ESTV; 5 for PL/I (D); 6 for RPG; 7 for Sort/Merge; 8 for the Utilities; 9 for Autotest; A for the Assembler; B for FORTRAN; and C for COBOL. The second, third, and fourth characters are the message number. The action indicator specifies the type of operator action required. The message contains all information pertaining to the operator's decision and/or actions.

The program identifiers used in multiprogramming are as follows.

Identifier	Program
BG	Background program
F1	Foreground-one program
F2	Foreground-two program
AR	Attention routine
SP	Supervisor

When a Supervisor routine such as OPEN or device error recovery is operating on behalf of a program, any messages it issues contain the identifier of that program.

The action indicators are as follows.

Indicator	Meaning
A Action:	The operator must perform a specific manual action before continuing. An example of this is the mounting of a magnetic tape, or the readying of an I/O device.
D Decision:	The operator must choose between alternate courses of action.
I Information:	The message does not require communication with the system. For example, this type of message can be used to indicate the termination of a problem program.
W Wait:	Used when a condition (such as an error on SYSRES) occurs that makes it impossible to continue processing. This indicator is not printed on the printer-keyboard. Instead, a message number is placed in byte 0 of main storage. The indicator W is placed in byte 1 of main storage. The Wait state is entered, and all interruptions are disabled. The only way that the system can be restarted is through the IPL procedure.
S SEREP:	Used when a hardware condition occurs that makes it impossible to continue processing. This indicator is not printed on the printer-keyboard, but may be displayed on the console. A 0 in byte 0 of main storage indicates a machine-check interruption; 1 indicates an I/O channel failure. The indicator S is stored in byte 1 of main storage. A special diagnostic storage display program (SEREP: System Environmental Recording, Editing, and Printing Program) supplied by customer engineers should be used when an S condition occurs. Re-IPLing is necessary after running SEREP.

Following is an example in multiprogramming format of a system-to-operator message.

```
BG 1C10A PLEASE ASSIGN SYSRDR
```

The characters BG indicate the background program. The character 1 indicates that Job Control issued the message. The characters C10 are the message number. The character A indicates action is required on the part of the operator. (The operator must type on the 1052 the assignment for SYSRDR.) PLEASE ASSIGN SYSRDR is the content of the message.

COMMUNICATION FROM THE OPERATOR

The operator may enter information to the system via the 1052 Printer-Keyboard in any of the following instances.

1. The operator has pressed the request key so that commands can be issued.
2. The system has requested operator response.
3. The programmer or operator has requested operator response with a PAUSE statement.

Once a command is processed, the printer-keyboard is unlocked to permit the issuing of further messages. Operator-to-system Job Control commands are recognized on SYSRDR as well as on SYSLOG.

Each operator-to-system command consists of two parts. The first part is an operation code of from one to eight alphabetic characters describing the action to be taken. Separated from the operation code by at least one blank are any necessary parameters. The parameters are separated by commas. The command ends with an end-of-block (Ⓑ= Alter Code 5).

In order that processing continue, an end-of-communications command consisting of only Ⓑ must be given by the operator following the last command. See Ⓑ -- End-of-Communication Command.

There are three types of operator-to-system commands.

1. Job Control commands.
2. Message (ATTN) commands.
3. Single-program initiation commands.

Job Control commands may be issued only between job steps of the batched-job program. A suspension of processing between job steps can be achieved by the programmer using the PAUSE or STOP control statement, or by the operator using the PAUSE command. When Job Control is ready to process these commands, the message READY FOR COMMUNICATIONS is printed on the printer-keyboard. Job Control commands are accepted for a foreground partition following the ATTN command BATCH.

ATTN commands may be issued at any time. Pressing the request key on the printer-keyboard causes the message READY FOR COMMUNICATIONS to be printed on the printer-keyboard. ATTN commands may then be issued. If a partition has an outstanding intervention required condition, the following message is issued:

```
0P60D INTER REQ {BG,F1,F2} cuu
```

Single-program initiation commands may be issued following the ATTN command START. The START command gives control to the single-program initiation routines.

- For a brief description of the Job Control and ATTN commands, see Job Control Commands and ATTN Commands
- For commands for initiating a single foreground program, see Single Program Initiation.
- For a complete description, including formats, refer to Descriptions and Formats of Commands and Statements.
- For a listing of all commands, see Appendix C.

Job Control Commands

The Job Control commands are:

ALLOC — Allocate Main Storage Command. Permits the operator to allocate main storage among foreground and background programs.

ASSGN — Assign Logical Name Command. Assigns a logical I/O unit to a physical device.

CANCEL — Cancel Job Command. Cancels the execution of the current job in the partition in which the command is given.

CLOSE — Close Output Unit Command. Closes either a system or programmer output logical unit assigned to a magnetic tape, or a system logical unit assigned to a 2311 or 2314.

DVCDN — Device Down Command. Informs the system that a device is no longer physically available for system operations.

DVCUP — Device Up Command. Informs the system that a device is available for system operations after the device has been down.

HOLD — Hold Foreground Unit Assignments Command. Causes all I/O assignments for the single program foreground area(s) specified to stay in effect from one job to the next.

LISTIO — List I/O Assignment Command. Causes the system to print a listing of I/O assignments.

LOG — Log Command. Causes the system to log columns 1-72 of all Job Control statements on SYSLOG until a NOLOG command is sensed.

MAP — Map Main Storage Command. Causes the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment.

MTC — Magnetic Tape Control Command. Controls IBM 2400 series magnetic tape operations.

NOLOG — Suppress Logging Command. Causes the system to suppress the logging of all Job Control statements except JOB, PAUSE, ALLOC, MAP, HOLD, RELSE, UNA, DVCDN, DVCUP, *, and /&, until a LOG command is sensed.

PAUSE — Pause Command. Causes Job Control processing to pause (in the partition which the command specifies) at the end of the next batched-job job step, or at the

end of the current batched-job job.

RELSE — Release Foreground Unit Assignments at EOJ Command. Causes all I/O assignments for the foreground area(s) specified that are operating in single-program mode to be set to unassigned at the end of any job that is initiated for that area.

RESET — Reset I/O Assignments Command. Resets certain I/O assignments to the standard assignments.

ROD — Record on Demand Command. Causes all SDR counters for all nonteleprocessing devices on the recorder file on SYSREC to be updated from the SDR counters in main storage.

SET — Set Value Command. Initializes the date, clock, and UPSI configuration; specifies the number of lines to be printed on SYSLST; and specifies the remaining disk capacity when SYSLST or SYSPCH is assigned to disk, and defines to the system the status of the recorder file on SYSREC used by the I/O error logging (OBR/SDR) and machine check recording and recovery (MCRR) features.

STOP — Stop Batched-Job Processing Command. Can be used in any batched-job partition to indicate that there are no more batched-jobs in that partition to execute.

UCS — Load Universial Character Set Buffer Command. Causes the 240-character Universal Character Set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit.

UNA — Immediately Unassign Foreground Unit Assignments Command. Immediately causes all I/O assignments for the single program foreground area(s) specified to be set to unassigned.

UNBATCH — Terminate Batched-Job Processing. Terminates batched-job foreground processing and releases the partition.

Ⓑ — End-of-Communication Command. Must be issued whenever the operator is finished communicating with the system.

ATTN Commands

The ATTN commands are:

ALLOC — Allocate Main Storage Command. Permits the operator to allocate main storage among foreground and background programs.

BATCH — Start, or Continue Batched-Job Operation. Causes batched-job operation to start in F2 or F1, or to continue in BG, F2, or F1.

CANCEL — Cancel Job Command. Cancels the execution of the current job in the specified partition.

LOG — Log Command. Causes the system to log columns 1-72 of all single program initiation commands on SYSLOG until a NOLOG command is sensed.

MAP — Map Main Storage Command. Causes the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment.

MSG — Transfer Control Command. Gives control to a foreground program operator communications routine previously activated by a STXIT instruction.

NOLOG — Suppress Logging Command. Causes the system to suppress the logging of all single program initiation commands until a LOG command is sensed.

PAUSE — Pause Command. Causes Job Control processing to pause at the end of the current program job step in the specified partition or, optionally, at end-of-job of the current program.

START — Start Processing Command. Initiates a foreground program or resumes batched-job processing in any partition.

TIMER — Interval Timer Command. Gives interval timer support to the partition specified.

Ⓑ — End-of-Communication Command. Must be issued whenever the operator is finished communicating with the system.

SINGLE PROGRAM INITIATION

Single program foreground programs are initiated by the operator from the printer-keyboard assigned to SYSLOG. The operator may initiate a single program whenever an allocated foreground area does not contain a program.

The operator initiates a single program by pressing the request key on the printer-keyboard. Control is given to the message (ATTN) routine, which reads commands from the operator via the printer-keyboard. The START command (discussed in ATTN Commands) indicates a single program is to be initiated. The ATTN routine determines if the area specified in the START command is allocated and does not contain a program. If so, it transfers control to the single program initiation routine; otherwise, the operator is notified that he has given an invalid command.

The single program initiator reads subsequent commands required to initiate the program. These commands are used primarily to specify I/O assignments and label information. When an I/O assignment is attempted, the following verification is made:

1. The symbolic unit is a valid logical unit.

2. The symbolic unit is contained within the number specified for the area when the system was generated.

3. If the symbolic unit is to be assigned to a non-DASD, the device is neither in use by the other foreground program (if applicable), nor is it assigned to a background job either as a standard, temporary, or alternate unit.

Each set of label information is incorporated into a label information block and written on the system residence pack for later retrieval and processing by IOCS. A main storage area for label information is required under the same conditions as for background jobs, and may be calculated and reserved by the initiator as specified in the LBLTYP command for self-relocating foreground programs. For non-self-relocating foreground programs, the label information area is determined at link edit time by the LBLTYP statement, described in the Job Control section of this manual.

When the EXEC command is encountered, the single program initiator directs the Supervisor to provide loading information for the program to be executed. If the program has not been cataloged to the core image library of the system, the operator is notified so that he may correct the EXEC command (for example, the name may have been misspelled), or cancel the initiation of the program.

Following receipt of the loading information from the Supervisor, the initiator checks to determine if a self-relocating program is to be loaded. (This is determined by the load address being zero.) The single program initiator directs the program to be loaded following the label information area in the foreground area. Its entry point is given by the Supervisor when it is loaded. A non-self-relocating program is loaded using the information obtained when the program was cataloged. Diagnostics for such conditions as the program being outside the limits of the foreground area, are issued by the Supervisor when the program is loaded.

When control is given to the user's foreground program, register 2 contains the address of the uppermost byte of storage available to the program. This value may be used to calculate the total storage available to the program. A foreground program may dynamically determine the storage available to it by storing the contents of this register for later reference.

A single program is terminated under its own control by issuing an EOJ, DUMP, or CANCEL macro instruction; or through operator action, program error, or certain input/output failures. When a single program is terminated, the following action is taken:

1. All I/O operations which the program has requested are completed. If telecommunication device I/O requests are outstanding, they are terminated by Halt I/O.

2. Tape error statistics by unit (if specified when the system was generated) are typed on the printer-keyboard for tapes used by the program.

3. Tape error statistics by volume (if specified when the system was generated), for tape volumes used by the program, may be printed on the printer keyboard or collected on disk.

4. DASD extents in use by the program for purposes of DASD file protection are dequeued. (DASD file protection is an option that may be selected when the system is generated.)

5. All I/O assignments made for the program are canceled, unless held across jobs by the HOLD command. This cancellation allows devices to be made available to subsequent programs.

6. The operator is notified that the program is completed and of the cause of termination, if abnormal. The main storage used by the program remains allocated for the appropriate foreground program area.

7. The program is detached from the system's task selection mechanism.

Following the completion of a single program, the operator may initiate another program for the specific area.

For more information on the SPI see the Concepts and Facilities publication listed in the Preface.

Single Program Initiation Commands

Single program initiation commands are submitted following a START command to the ATTN routine. They can be submitted through the printer-keyboard (SYSLOG), or through a card reader (following a READ command). Two slashes (//) in columns 1 and 2 are optional for EXTENT, DLBL, EXEC, LBLTYP, and TLBL commands, and are accepted by the single program initiator for these commands.

The single program initiation commands follow.

ASSGN — Assign Logical Name Command. Assigns a logical I/O unit to a physical device.

CANCEL — Cancel Initiation Command. Cancels the initiation of a foreground program in the single program mode.

DLBL — DASD Label Information Command. Contains file label information for DASD label checking and creation.

EXEC — Execute Program Command. Specifies the single program foreground program to be executed.

EXTENT — DASD Extent Information Command. Defines each area, or extent, of a DASD file.

HOLD — Hold Foreground Unit Assignments Command. Causes all I/O assignments for the single program foreground area(s) specified to stay in effect from one job to the next.

LBLTYP — Reserve Storage for Label Information Command. Defines the amount of main storage to be reserved for processing of tape and nonsequential disk file labels in the problem program area of main storage.

LISTIO — List I/O Assignment Command. Causes the system to print a listing of I/O assignments.

LOG — Log Command. Causes the system to log columns 1-72 of all single program initiation commands on SYSLOG until a NOLOG command is sensed.

MAP — Map Main Storage Command. Causes the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment.

MSG — Transfer Control Command. Gives control to a foreground program operator communications routine previously activated by a STXIT instruction.

NOLOG — Suppress Logging Command. Causes the system to suppress the logging of all single program initiation commands until a LOG command is sensed.

PAUSE — Pause Command. Causes Job Control processing to pause at the end of the current batched-job job step, or at the end of the current batched-job job.

READ — Specify Reader Command. Specifies a card reader from which further single program initiation commands are to be read.

RELSE — Release Foreground Unit Assignments at EOJ Command. Causes all I/O assignments for the single program foreground area(s) specified to be set to unassigned at the end of any job that is initiated for that area.

TIMER — Interval Timer Command. Gives interval timer support to the partition specified.

TLBL — Tape Label Information Command. Contains file label information

for tape label checking and writing.

UCS — Load Universal Character Set Buffer Command. Causes the 240-character Universal Character Set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit.

UNA — Immediately Unassign Foreground Unit Assignments Command. Immediately causes all I/O assignments for the foreground area(s) specified to be set to unassigned.

Ⓑ — End-of-Communication Command. Causes input reading to switch back to the device specified in the READ command.

Programming support will continue for the following initiation commands provided in previous versions of the system. Two slashes (//) in columns 1 and 2 are optional for these commands and are accepted by the single program initiator for them.

DLAB — DASD Label Information Command. Contains file label information for DASD label checking and creation.

TPLAB — Tape Label Information Command. Contains file label information for tape label checking and writing.

VOL — Volume Information Command. Used when a set of label information for a magnetic tape file or a DASD file is specified. (It is not required with the current DLBL, EXTENT, or TLBL commands.)

XTENT — DASD Extent Information Command. Defines each area, or extent, of a DASD file. It is used in conjunction with the VOL, DLAB commands.

SYSTEM OPERATION WITHOUT A 1052

Multiprogramming requires a printer-keyboard. A batched-job environment, however, may be made operational when a 1052 is not available on the system, by assigning SYSLOG to a printer. Messages to the operator are printed on SYSLOG, after which an assumed operator response, where applicable, is taken. In most cases, the assumed response results in the termination of the job. There is no communication from the operator, except for I/O device error routines which require operator-stored response in low main storage. In such cases, the message is printed on the printer assigned to SYSLOG and the device error routines wait until the operator stores his response and presses the console interrupt key. PAUSE statements in the Job Control input stream are ignored.

In addition to the requirement that SYSLOG be assigned to a printer, SYSRDR and SYSIPT must each be assigned to a card reader (which may be the same card reader), SYSPCH must be assigned to a card punch, and SYSLST must be assigned to a printer. If SYSLOG and SYSLST are assigned to the same printer, system-to-operator messages may be embedded within user output.

When no 1052 is available, total throughput in the individual installation suffers, due to the frequent cancellation of jobs resulting from errors, such as incorrect job setup, I/O assignments, etc. In many instances, such errors could be corrected by the operator via the 1052.

SYSTEM LOADER

The System Loader is a permanently core-resident routine in the Supervisor. It loads all programs run in the Disk Operating System environment, with the exception of the core-resident Supervisor itself. Programs are loaded into main storage from the core image library.

FETCH Macro Instruction

This macro instruction has the format:

Name	Operation	Operand
[name]	FETCH	{phasename / (1)} [{,entryname / ,(0)}]

The FETCH macro instruction loads the phase specified in the first parameter. The phase name can be 1-8 characters long. Control is passed to the address specified by the second operand. If the second operand is not specified, control is passed

to the entry point determined at linkage-edit time.

The parameters may be specified in either symbolic or register notation. When register notation is used for phasename, the register must be preloaded with the address of an eight-byte field that contains the phasename as alphameric characters. If necessary, the phasename should be padded with blanks.

If ordinary register notation is used for entryname, the absolute address of the entry point of the phase should not be preloaded into register 1. If, instead, a symbolic name is used for entryname, the macro expansion results in a V-type address constant. The entryname does not have to be identified by an EXTRN statement.

LOAD Macro Instruction

This macro instruction has the format:

Name	Operation	Operand
[name]	LOAD	{phasename / (1)} [{,loadaddr / ,(0)}]

The LOAD macro instruction is used when a phase is to be loaded into main storage, but not executed immediately. It can be used to load tables and reference material. The LOAD macro instruction loads the phase specified in the first parameter and returns control to the calling phase. The phasename can be 1-8 characters long. The user should code his LOAD in a place where it cannot be overlaid by the new phase.

After execution of the macro, the entry point address of the called phase is returned in register 1 to the programmer. This entry point address is determined at linkage-edit time.

If an optional address parameter is provided, the load-point address specified to the linkage editor is overridden, and the phase is loaded at the address specified. The address used must be outside the Supervisor area. When an overriding address is given, the entry point address is relocated and returned in register 1. None of the other addresses in the phase are relocated.

The parameters may be specified in either symbolic or register notation. When register notation is used for phasename, the register must be preloaded with the address of an eight-byte field that contains the phasename. If necessary, the phasename should be left-justified and padded with blanks. If ordinary register notation is used for loadaddr, the parameter should not be preloaded into register 1.

CHECKPOINT/RESTART

When a background program or a batched job foreground program is expected to run for an extended period of time, provision may be made for taking checkpoint records periodically during the run. The records contain the status of the job and system at the time the records are written. Thus, they restart at some midway point rather than at the beginning of the entire job, if processing must be terminated for any reason before the normal end of job. Any programmer logical unit (SYS000-SYSmax) assigned to tape, 2311, or 2314, can be used for recording checkpoints if the proper file definitions have been made and the correct label statements have been submitted.

For example, some malfunction such as a power failure may occur and cause such an interruption. If checkpoint records are written periodically, operation can be restarted using a set of checkpoint records written before the interruption. Therefore, the records contain everything needed to reinitialize the system when processing is restarted.

The Disk Operating System includes routines to take checkpoint records and to restart a job at a given checkpoint. The checkpoint and restart routines are included in the core image library when the system is generated. The CHKPT routine is executed in the transient area. The checkpoint routine is called in response to a CHKPT macro instruction in the problem program. The restart routine is called by Job Control when it reads a RSTRT control statement. When a program is restarted, the user must reset any STXIT macro instructions that are desired. Checkpoint/restart does not save or restore floating-point registers.

Only background programs or batched-job foreground programs may be checkpointed. Programs processing ASCII may not be checkpointed. Checkpoint records are written on a 2311 or 2314 DASD or on magnetic tape. Each checkpoint is uniquely identified. When restarting, the RSTRT control statement specifies which checkpoint is to be loaded. If multireel files are being used, the operator must be

aware of which reels were being processed when the checkpoint was taken.

Multitasking users should only issue the CHKPT macro in the main task with no subtasks attached. In addition, no tracks on any DASD should be in the hold state. A multitasking abnormal termination routine should not contain a CHKPT macro. Checkpoints should be taken when a program is running successfully, not when it is canceling.

Checkpointed programs must be restarted in the same partition in which they were checkpointed.

Checkpoint records written by Version 1 of DOS are not acceptable to subsequent versions of the system. However, if the checkpoint records are embedded on magnetic tape, they will be bypassed by the current versions of DOS.

It is possible to increase partition allocation between the time the checkpoint is taken and the time the program is restarted if the starting address of the partition remains unchanged.

A detailed explanation of the CHKPT macro instruction is found in the Supervisor and I/O Macros publication listed in the Preface.

The restart facility is described in the Job Control section of this publication.

NORMAL AND ABNORMAL END-OF-JOB HANDLING

When a background program or batched-job foreground program reaches the normal end of a job step, issuing the EOJ macro instruction causes the Supervisor to fetch Job Control to begin processing the control statements for the next job or job step.

A special routine of the Supervisor can provide a printout of main storage in the event of some abnormal end-of-job-step situation. This routine is fetched into the transient area if DUMP is specified as a standard option at system generation time or when DUMP is specified in the OPTION control statement. The dump routine prints the contents of the registers, the supervisor area, and the partition that called it. For background programs, the printout is on SYSLST. SYSLST can be assigned to a printer, a magnetic tape unit, or a disk unit.

For foreground programs, the printout is also on SYSLST. However, foreground programs operating in single program mode require that SYSLST be assigned to a printer. SYSLST can be assigned to a printer, a disk unit, or to a magnetic tape unit for batched-job foreground programs. The dump routine provides, in the event of an abnormal termination of the job, a printout of the Supervisor area, the appropriate foreground area, and the program registers. If SYSLST is not assigned to a printer for single foreground programs, or to a printer, a disk unit, or a non-file-protected magnetic tape unit for batched-job foreground programs; any printout specified by DUMP or PDUMP in the problem program is suppressed.

When a magnetic tape unit is used, the storage print routine does not reposition the tape before writing. When the routine is completed, a tapemark is written and the tape is repositioned before the tapemark. When an end-of-reel condition is detected, the system automatically provides end-of-reel procedures, closing the current volume and opening a new volume. The procedure is identical to that outlined in the section CLOSE System Tape Output Files. Records written on SYSLST are 121 bytes in length, the first byte being an ASA control character.

The Job Control program provides job-to-job transition for background programs and for batched-job foreground programs (if the option has been specified at system generation time) within the Disk Operating System. It is also called into main storage to prepare each job step to be run. (One or more programs can be executed within a single job. Each such execution is called a job step.) It performs its functions between job steps and is not present while a problem program is being executed. Job Control is called by:

- The Initial Program Loader, to process the first job after an IPL procedure.
- The Supervisor, at normal end of a job step, or at an abnormal end of job.

A macro instruction, EOJ, is provided to call Job Control at normal end of a job step.

Single program foreground programs are initiated by the operator from the printer-keyboard. Therefore, each execution of a single program is a separate job.

Job Control Functions

Job Control performs various functions on the basis of information provided in job control statements. These functions are:

1. Prepare programs for execution.
2. Assign device addresses to symbolic names.
3. Set up fields in the communications region.
4. Edit and store volume and file label information.
5. Prepare for restarting of checkpointed programs.

Job Control clears the current partition area in main storage (except the first 150 bytes) to binary zero between job steps.

The single program initiation routine performs for single programs functions similar to those performed by Job Control for batched-job programs.

PREPARE PROGRAMS FOR EXECUTION

All background programs run in the system are loaded from the core image library in the resident disk pack. If a program has been previously cataloged (see Linkage Editor), Job Control constructs a phase directory in the resident disk pack and directs the system loader to load that program for execution. If the program is stored temporarily in the core image library, Job Control constructs a phase directory of that program from entries in the core image directory and then transfers to the system loader to load it for execution.

The phase directory for a cataloged program includes an entry for each program phase whose name has the same first four characters as the name in the EXEC control statement.

A foreground program directory contains an entry for each foreground phase whose name starts with FGP.

All foreground programs run in the system are loaded from the core image library. Thus, they will be link edited and cataloged to the library by the Linkage Editor.

SYMBOLIC INPUT/OUTPUT ASSIGNMENT

Job Control is responsible for assigning physical I/O units. Programs do not reference I/O devices by their actual physical addresses, but rather by symbolic names. The ability to reference an I/O device by a symbolic name rather than a physical address provides advantages to both programmers and machine operators. The symbolic name of a device is chosen by the programmer from a fixed set of symbolic names. He can write a program that is dependent only on the device type and not on the actual device address. At execution time, the operator or programmer determines the actual physical device to be assigned to a given symbolic name. He communicates this to Job Control by a control statement (ASSGN). Job Control associates the physical device with the symbolic name by which it is referenced.

A fixed set of symbolic names is used to reference I/O devices. No other names can be used. They are:

SYSRDR	Card reader, magnetic tape unit, or disk extent used for Job Control statements.
SYSIPT	Card reader, magnetic tape unit, or disk extent used as the input unit for programs.
SYSPCH	Card punch, magnetic tape unit, or disk extent used as the main unit for punched output.
SYSLST	Printer, magnetic tape unit, or disk extent used as the main unit for printed output.
SYSLOG	Printer-keyboard used for operator messages and to log Job Control statements. Can also be assigned to a printer if not in multiprogramming environment.
SYSLNK	Disk extent used as input to the Linkage Editor.
SYSRES	System residence area on a disk drive.
SYSRLB	Disk extent used for a private relocatable library.
SYSSLB	Disk extent used for a private source statement library.
SYSREC	Disk extent used to store error records collected by the I/O error logging (OBR/SDR) and machine check recording and recovery (MCRR) functions.
SYS000-SYSmax	All other units in the system.

Whenever a system logical unit (SYSRDR, SYSIPT, SYSLST, SYSPCH) is assigned to an extent of disk storage, the assignment must be permanent. Temporary assignments (via the // ASSGN statement or the ASSGN command with the TEMP option) are not permitted. A logical unit for a system output file cannot be assigned to a device that is assigned to another logical unit.

The first ten of these symbolic names, termed system logical units, are used by the system control program and system service programs. Of these ten units, user batched-job programs may also use SYSIPT and SYSRDR for input, SYSLST and SYSPCH for appropriate output, and SYSLOG for operator communication. Normally, SYSRDR and SYSIPT both refer to the same device. Any additional devices in the system, termed programmer logical units, are referred to by names ranging consecutively from SYS000 to SYSmax, with SYS000 to SYS009 being the minimum provided in any system. Programmer logical units are defined at system generation time for each class of program (background, foreground-one, and foreground-two) to be run in the system. For example, in a multiprogramming environment, a unique SYS000 is defined for each class of program, a unique SYS001 is defined for each class of program, etc. The combined number of programmer logical units defined for the system may not exceed 222 (SYS000-SYS221) when MPS=BJF, and 244 (SYS000-SYS243) when MPS=YES or MPS=NO.

For the convenience of the user, two additional names are defined for batched-job program assignments. These names are valid parameters to Job Control only via the ASSGN, CLOSE, VOL, EXTENT, and XTENT statements described in Descriptions and Formats of Commands and Statements. Reference within a program (such as in the CCB or a DTF) or in a LISTIO or MTC command must name the particular logical unit to be used (SYSLST or SYSPCH, SYSRDR or SYSIPT). The additional names are:

SYSIN	Name that can be used when SYSRDR and SYSIPT are assigned to the same card reader, magnetic tape unit, or disk extent. In the first two cases, it may be either a temporary or a permanent assignment. This name must be used when SYSRDR and SYSIPT are assigned to the same disk extent, and may be only a permanent assignment.
SYSOUT	Name that must be used when SYSPCH and SYSLST are assigned to the same magnetic tape unit. It may be only a permanent assignment. Separate file operation is reestablished by submitting a permanent assignment for either SYSLST or SYSPCH to a unit not currently in use by the combined file. A CLOSE command may be used to perform this function.

A tape or disk extent to be used as SYSIN can be prepared by using the IBM-supplied utility macros. Likewise, the IBM-supplied tape to printer/punch utility macros can be assembled and used to convert

SYSOUT output into printed and punched card output. Examples of these two functions are shown in Disk Operating System and Tape Operating System Utility Macros Specifications, GC24-5042.

With the exception of SYSLNK, SYSRLB, and SYSSLB, batched-job foreground programs can reference any system logical unit. If a foreground program is in the single initiated program mode of operation, only SYSIPT, SYSRDR, SYSPCH, and/or SYSLST can be referred to. When used, these units must be assigned to unit record devices.

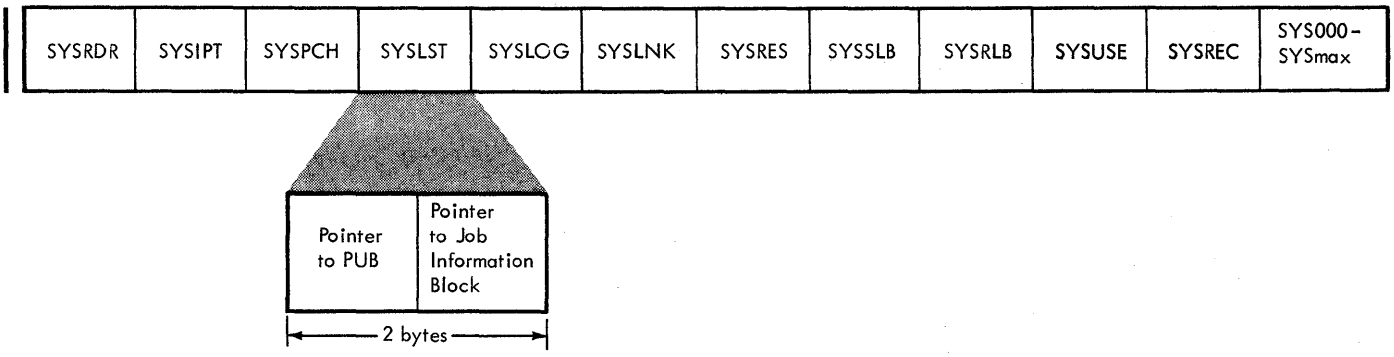
Logical Unit Block (LUB) and Physical Unit Block (PUB)

At system generation time when a Supervisor is assembled, a device table is set up with an entry for each of the symbolic names that will be used in the system. Each entry is called a logical unit block (LUB). Figure 8 shows the format of the device table for the background partition. Similar tables are set up for each of the foreground partitions as required. The sequence of the tables is background, foreground-two, foreground-one. The length of the table depends on the number of devices specified at system generation time. The system LUBs and the first ten programmer LUBs are always present.

A physical unit block (PUB) can be associated with each LUB. Figure 9 shows the format of a PUB. The PUBs are ordered by priority within the channel to which the various devices are attached.

Normally, each symbolic name is assigned a physical device address at the time the Supervisor is assembled. In some cases, a single device may be assigned to two or more symbolic names. An installation can make specific assignments at system generation time and establish these as conventions to be followed by all programmers. By following the conventions, most background jobs can be submitted for execution with no ASSGN control statements. Figure 10, for example, shows a typical system configuration. The following conventions might be established for the installation's own background programs and for IBM-supplied programs.

1. Control statement input is read from SYSRDR. This device is normally



●Figure 8. Sequence of LUBs in Device Table

Channel	Unit	Error Retry Counter or Pointer to Channel Queue	Pointer to Tape Error Block (both TEB and TEBV)	Device Type	Device Options (Tape Set Mode, etc.)	Channel Scheduler Flags	Job Control Flags

Figure 9. Format of PUB Entry

assigned to the same physical unit as SYSIPT. Most of the system programs (language translators, etc) and user programs normally are read from SYSIPT. Thus, when SYSRDR and SYSIPT refer to the same device, SYSIN can be used.

2. Card output is punched on SYSPCH.

3. Printed output is on SYSLST.

4. A 1052 is assigned to SYSLOG.

5. The two disk drives are addressed as SYSRES, SYSLNK, SYS001, SYS002, and SYS003. SYSLNK is used as input to the Linkage Editor. SYS001, SYS002, and SYS003 are used by the language translators as work files. Language translators also can output on SYSLNK for subsequent input to the linkage editor.

6. The two tape units are addressed as SYS004 and SYS005.

The initial device assignments present after each IPL procedure are those made when the system is generated plus any changes introduced at IPL time.

Once the Supervisor is loaded into main storage, reassignments made by the operator on the 1052 become permanent modifications to the existing system assignments unless the operator specifies temporary assignment. Reassignments made by the programmer are reinitialized to the original assignments at the completion of a job.

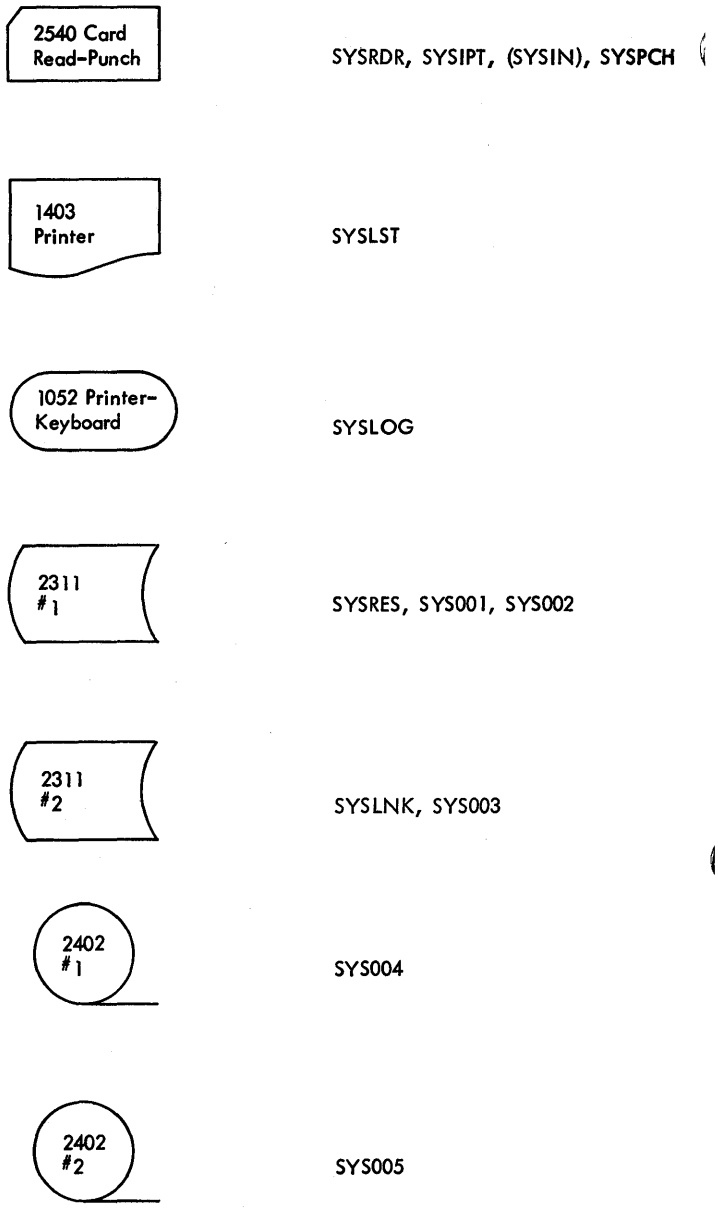


Figure 10. Example of Symbolic Device Assignment

SET UP COMMUNICATIONS REGION

Job Control takes the following information from control statements and places it in the communications region:

1. Job Name. Taken from the JOB statement. This field can be used by the problem program for accounting purposes.

2. Job Date. Taken from the DATE statement processed at the beginning of a job. It can be used by the problem program to date output reports. If the DATE statement is not used, the system uses the date supplied by the operator at IPL time via the SET command.

3. User Program Switch Indicators. Taken from the UPSI statement. The bit pattern in this byte can be used as switch indicators to specify program options.

EDIT AND STORE LABEL INFORMATION

All volume and file label processing is done during problem program execution. However, label information to be checked against is read from label statements by Job Control or SPI and stored in the resident pack for subsequent processing. The label area occupies one cylinder and is allocated in the following manner:

Track	Content
0	Background program temporary (USRLABEL) label information
1	Background program standard (PARSTD) label information
2	Foreground-two temporary (USRLABEL) label information
3	Foreground-two standard (PARSTD) label information
4	Foreground-one temporary (USRLABEL) label information
5	Foreground-one standard (PARSTD) label information
6-n	Standard (STDLABEL) label information for any partition. n is 9 for 2311; 19 for 2314.

Each partition has certain tracks allotted for temporary (USRLABEL) label information (tracks 0, 2, and 4) and for standard (PARSTD) label information (tracks 1, 3, and 5). Tracks 6-9 (or 6-19) are for standard (STDLABEL) label information for any partition.

The fraction of the total area available on each track that is used by each type of file identification statement is about as follows:

	2311	2314
For each tape label	1/20	1/30
For each sequential DASD extent	1/18	1/27
For nonsequential DASD per file	1/20	1/29
+ per extent	1	1
	20 x 7	29 x 11

Label blocks composed of information from DLBL and EXTENT, or TLBL statements are written on tracks 0, 2, or 4 (depending on the partition in which they are submitted) when:

- The DLBL and EXTENT, or TLBL statements are preceded by // OPTION USRLABEL, or
- The DLBL and EXTENT, or TLBL statements are not preceded by // OPTION PARSTD or by // OPTION STDLABEL.

Standard label information submitted with OPTION STDLABEL or PARSTD is carried from job to job until overwritten.

A complete description of the Job Control statements DLBL, EXTENT, OPTION, and TLBL is contained in the section Descriptions and Formats of Commands and Statements.

Each set of temporary label information submitted within a job or job step is written in the appropriate temporary label information area. This information is not carried from job to job. Unless overwritten by a succeeding job step, any label information submitted at the beginning of a job can be used by a subsequent job step. For example, if a job consists of three job steps, label information submitted at the beginning of the first job step can be used by the second and third job steps of the job. However, label information submitted at the beginning of the second job step would destroy the label information written at the beginning of the first job step.

Label blocks composed of information from DLBL and EXTENT, or TLBL statements are written on tracks 1, 3, or 5 (depending on the partition in which they are submitted) when:

- The DLBL and EXTENT, or TLBL statements are preceded by // OPTION PARSTD.

The facility of establishing standard label information for each partition (PARSTD) provides for different files for each partition with the same filename (DTF name) for all partitions. For example, system input/output requires separate files for each partition, but the file names must be IJSYSIN, IJSYSPH, and IJSYSLS for all partitions. Another use of the partition standard facility is to free the STDLABEL area for passing label information to partitions operating in the single program initiation mode.

Track 1 of the label cylinder can be used to establish extent information for the system logical unit SYSLNK, and the system component work files SYS001, SYS002, and SYS003. This eliminates the necessity of submitting DLBL and EXTENT statements each time a compilation or linkage editing function is performed.

Each time DLBL and EXTENT, or TLBL statements are submitted following // OPTION PARSTD, tracks 1, 3, and 5 (depending on the partition in which they are submitted) of the label information cylinder are overwritten. This facility is available to the user for any tape or DASD files.

Label blocks composed of information from DLBL and EXTENT, or TLBL statements are written on tracks 6-9 (or 6-19) when:

- The DLBL and EXTENT, or TLBL statements are preceded by // OPTION STDLABEL in a background job.

Each time DLBL and EXTENT, or TLBL statements are submitted following // OPTION STDLABEL, tracks 6-9 (or 6-19) of the label information cylinder are overwritten. This facility is available to the user for any tape or DASD files.

All DASD and tape file identification statements can be submitted under both OPTION STDLABEL and OPTION PARSTD.

The logical IOCS OPEN routines search the temporary user label area (USRLABEL), followed by the partition standard label area (PARSTD), followed by the standard label area (STDLABEL), in that order.

Self-relocating programs can use tape and nonsequential disk standard labels if a LBLTYP command or statement is submitted or if a DS statement at the beginning of the program reserves an equivalent number of bytes. Neither the LBLTYP command nor statement is required for sequential disk standard label processing.

The formats of the label information statements are discussed in the section Descriptions and Formats of Commands and Statements. See the Data Management Concepts publication listed in the Preface for a complete discussion of volume and file labels.

RESTARTING PROGRAMS FROM CHECKPOINT

Job Control prepares the system for restarting from a checkpoint by loading the restart program which repositions tape drives, reinitializes the communications region, and stores the information from the RSTRT statement. The restart program handles the actual restarting of the problem program.

Job Control Statements

GENERAL CONTROL STATEMENT FORMAT

Certain rules must be followed when filling out control statements. Job Control statements conform to these rules.

1. Name. Two slashes (//) identify the statement as a control statement. They must be in columns 1 and 2. At least one blank immediately follows the second slash. Exception: The end-of-job statement contains /& in columns 1 and 2, the end-of-data-file statement contains /* in columns 1 and 2, and the comment statement contains * in column 1 and blank in column 2.
2. Operation. This describes the type of control statement (the operation to be performed). It can be up to eight characters long. At least one blank follows its last character.
3. Operand. This may be blank or may contain one or more entries separated by commas. The last term must be followed by a blank, unless its last character is in column 71. Any blank within the operand fields, except for fields contained within apostrophes, is considered an end-of-operand indication. No further processing of that card occurs.

All control statements are essentially free form. Information starts in column 1 and cannot extend past column 71. Continuation cards are not recognized by Job Control. For the exception to this rule, see the descriptions of the DLAB and TPLAB statements.

Job Control normally reads from the device identified by the symbolic name SYSRDR. However, Job Control statements can also be entered through SYSLOG. A brief description of the Job Control statements follows. For a complete description, including formats, refer to the section Descriptions and Formats of Commands and Statements. A listing of all statements is shown in Appendix D.

ASSGN Statement: Used at execution time to assign a specific device address to the symbolic unit name used.

CLOSE Statement: Closes either a system or a programmer logical unit assigned to tape.

DATE Statement: Contains a date that is put in the communication region.

DLBL Statement: Contains file label information for DASD label checking and creation.

EXEC Statement: Indicates the end of job-control statements for a job step, and that execution of a program is to begin.

EXTENT Statement: Defines each area, or extent, of a DASD file.

JOB Statement: Indicates the beginning of control information for a job.

LBLTYP Statement: Defines the amount of main storage to be reserved at link-edit time for processing of tape and nonsequential DASD file labels in the problem program area of main storage.

LISTIO Statement: Used to get a listing of I/O assignments.

MTC Statement: Controls operations on logical units assigned to IBM 2400 series magnetic tapes.

OPTION Statement: Specifies one or more of the Job Control options.

PAUSE Statement: Causes pause immediately after processing this statement.

RESET Statement: Resets I/O assignments to the standard assignments.

RSTRT Statement: Restarts a checkpointed program.

TLBL Statement: Contains file label information for tape label checking and writing.

UPSI Statement (User Program Switch Indicators): Allows the user to set program switches that can be tested much the same as sense switches or lights used on other machines.

/* Statement: Indicates the end of a data file or the end of a job step.

/& Statement: Indicates the end of a job.

* Statement: Job Control comments statement.

Programming support continues for the following Job Control statements provided in previous versions of the system.

DLAB Statement: Contains file label information for DASD label checking and creation.

TPLAB Statement: Contains file label information for tape label checking and writing.

VOL Statement: Used when a set of label information for a magnetic tape file or a DASD file is specified. It is not required with the current DLBL, EXTENT, or TLBL statements.

XTENT Statement: Defines each area, or extent, of a DASD file. It is used in conjunction with the VOL, DLAB statements.

Any statement other than these is recognized as an error. A message is issued so that the programmer or operator can correct the statement in error. Some of the errors recognized are:

- Invalid symbolic unit name.
- No space reserved in LUB table for a symbolic unit.
- Invalid device type.
- Invalid length of field.
- Invalid character.
- Missing /& statement.
- A volume (VOL) statement does not precede a label (DLAB or TPLAB) statement.
- An EXTENT statement does not immediately follow its associated DASD label (DLBL) statement.

Whenever an invalid statement is indicated, the statement must be reissued to be effective. For example, if an OPTION LINK is encountered without a SYSLNK assignment, the OPTION statement must be reentered after assigning SYSLNK.

SEQUENCE OF CONTROL STATEMENTS

The Job Control statements for a specific job always begin with a JOB statement and end with a /& (end of job) statement. A specific job consists of one or more job steps. Each job step is initiated by an EXEC statement. Preceding the EXEC statement are any job control statements necessary to prepare for the execution of the specific job step. The only limitation on the sequence of statements preceding the EXEC statement is that discussed for the label information statements. The following statements can precede the EXEC statement for a job step.

```
ASSGN
CLOSE
RESET
DATE
UPSI
LBLTYP
DLBL
EXTENT
TLBL
LISTIO
MTC
OPTION
PAUSE
*
```

The label statements must be in the order:

DLBL
EXTENT (one for each area of file in volume)

and must precede the EXEC statement to which they apply. If the TLBL statement is used, it must precede the EXEC statement to which it applies.

The LBLTYP statement is used at Linkage Editor time and must precede the // EXEC LNKEDT statement with the exception of self-relocating batched job programs, for which it is instead submitted immediately preceding the // EXEC statement for the program.

DESCRIPTIONS AND FORMATS OF COMMANDS AND STATEMENTS

This section contains descriptions and formats of the Job Control commands and statements, single-program initiation commands, and ATTN routine commands. These commands and statements are arranged in alphabetic order. Each command or statement includes an indication, on the line immediately under the format, of the programs or routines that accept it. These programs or routines are identified as follows:

Job Control - JC
Single-Program Initiator - SPI
Attention routines - AR

Figure 12 contains the commands and statements grouped by function and also indicates the programs or routines that accept them.

Job Control statements, with the exception of /*, /&, and *, contain two slashes (//) in columns 1 and 2 to identify them. Job Control commands, ATTN commands, and single-program initiation commands are not preceded by the two slashes. The two slashes are optional for the DLBL, EXTENT, TLBL, VOL, XTENT, DLAB, and TPLAB single-program initiation commands and are accepted by the single-program initiator for these commands only. The two slashes are followed by at least one blank space, followed by the operation code. The operation code is followed by at least one blank space, followed by the required parameters of the operand. These parameters are separated from each other by commas. The last parameter must be followed by a blank, unless its last character is in column 71.

Job Control commands, ATTN commands, and single-program initiation commands contain the operation code, at least one blank space, and then the required parameters. The parameters are separated by commas. The operation code usually begins in column 1 of the command, but this is not an absolute necessity.

Usage of Statements and Commands

Job Control statements, normally entered by the programmer, are used for batched-job programs only. They are usually coded as part of the input job stream and are entered through SYSRDR.

Commands (Job Control, Attention, and Single Program Initiation) are normally entered by the operator.

- Job Control commands are for batch processing in a multiprogramming environment. They are issued between jobs or job steps and are entered through SYSRDR or SYSLOG.
- Attention commands can be issued at any time by pressing the request key on the 1052 printer-keyboard. Some of these commands can be issued only in a multiprogramming environment.
- Single-Program Initiation commands may only be issued in a multiprogramming environment following the Attention command START [F1 or F2].

ALLOC -- Allocate Main Storage

The ALLOC command permits the operator to allocate main storage among foreground and background programs. When used in the attention routine, ALLOC cannot reduce the background area at any time. The number of bytes to be allocated for one or both foreground areas is specified in 2K (2048 bytes) increments. If only one foreground area is referenced, it is assumed that the amount of storage allocated to the other is not to change.

ALLOC { F1=nK[,F2=nK] }
      { F2=nK[,F1=nK] }

Accepted by JC and AR.

The value n must be an even integer.

The following considerations apply to storage allocation among foreground and background programs:

1. The areas are always contiguous. No gaps are permitted between allocated areas.

2. The maximum size of a foreground area is 510K. This restriction does not apply to background programs.

3. To delete a foreground area from the system, an ALLOC command must be given specifying an area of 0K (zero K).

4. If storage allocation was specified when the system was generated, the IPL routine determines the size of main storage and allocates the specified foreground areas downward from high main storage.

Storage is not allocated when:

1. The allocation would cause a decrease in the storage allocated to an active foreground program.

2. The allocation would result in the relocation of an active foreground program.

3a. A job control allocation would reduce the background area [or foreground area(s) while operating in the batched foreground model to less than 10K bytes.

b. An ATTN allocation would reduce the background area, which is always considered active when allocating storage from the ATTN routine.

ASSGN -- Assign Logical Name

The ASSGN command or statement assigns a logical I/O unit to a physical device. All device assignments made for single-program foreground programs are canceled either by another ASSGN to the same unit or at the end of each program, unless held across jobs by the HOLD command. Except for DASD devices, a program cannot be assigned a device assigned to a program in another partition.

```
ASSGN SYSxxx,address [{,X'ss'}]
                     [{,ALT  }]
```

Accepted by SPI.

```
// ASSGN SYSxxx,address [{,X'ss'}]
                        [{,ALT  }]
```

Accepted by JC.

```
ASSGN SYSxxx,address [{,X'ss'}] [,TEMP]
                     [{,ALT  }]
```

Accepted by JC.

The job control statement (// ASSGN) is temporary. It remains in effect only until the next change in assignment or until the end of job whichever occurs first. The job control command (ASSGN) is permanent. It remains in effect or restorable until the next permanent assignment, the DVCDN command, or re-IPL of the system, whichever occurs first. A CLOSE to a system logical unit on disk (2311 or 2314) also removes a permanent assignment. See also the TEMP override of a permanent ASSGN.

At the completion of a job, a temporary assignment is automatically restored to the permanent assignment for the logical unit.

The entries in the operand field represent the following.

SYSxxx The symbolic unit name. It can be one of the following.

For Job Control:

SYSRDR
SYSIPT
SYSIN
SYSPCH
SYSLST
SYSOUT
SYSLNK
SYSLOG
SYSSLB
SYSRLB
SYSREC
SYS000-SYSmax

When SYSOUT is assigned, the magnetic tape device must not be the permanent assignment of either SYSLST or SYSPCH. Before assigning a tape drive to a system output unit (SYSOUT, SYSLST, SYSPCH), all previous assignments of this tape drive to any system input units and to any programmer units (input or output) must be unassigned. SYSLNK is restricted to background partitions. It is not possible to change the assignment of SYSLOG while a foreground partition is active.

For single-program initiation:

SYSRDR
SYSIPT
SYSLST
SYSPCH
SYS000-SYSmax

Note: The system units must be assigned to unit record devices.

address Can be expressed as X'cuu', UA, or IGN

X'cuu' Indicates the channel and unit number (in hexadecimal).

c = 0 for multiplexor channel, 1-6 for selector channels 1-6

uu = 00 to FE (0 to 254) in hexadecimal

UA Indicates the logical unit is to be unassigned. Any operation attempted on this device causes the cancelation of the job.

IGN Indicates the logical unit is to be unassigned and that all program references to the logical device are to be ignored. For files processed by logical IOCS, however, the OPEN to the file is ignored and the IGNORE indicator is set on in the DTF table. (See OPEN(R) Macro in the Supervisor and Input/Output Macros publication referred to in the Preface.) All I/O commands issued to the file are ignored. The IGN option is not valid for SYSRDR, SYSIPT, and SYSIN.

X'ss' Device specifications (used to specify mode settings for 7-track and 9-track tapes). If X'ss' is not specified at system generation time or at IPL time, the system assumes X'90' for 7-track tapes and X'C0' for 9-track tapes. C0 is the normal reset mode for a 9-track tape unit and specifies the maximum byte density for that device. C8 is an alternate mode setting for 9-track dual density tapes only.

The standard mode is entered in the PUB table at system generation or at IPL time. If the mode setting (different from, or the same as, the standard mode) is specified in a temporary ASSGN statement, it becomes the current mode setting and is entered as such in the PUB table. When the current job ends, the standard mode is restored in the PUB table. The mode specification in a permanent ASSGN becomes the standard mode. If the X'ss' parameter is not specified for a job, the mode is the same as the standard mode.

The specifications are:

ss	Bytes per Inch	Parity	Translate Feature	Convert Feature
10	200	odd	off	on
20	200	even	off	off
28	200	even	on	off
30	200	odd	off	off
38	200	odd	on	off
50	556	odd	off	on
60	556	even	off	off
68	556	even	on	off
70	556	odd	off	off
78	556	odd	on	off
90	800	odd	off	on
A0	800	even	off	off
A8	800	even	on	off
B0	800	odd	off	off
B8	800	odd	on	off
C0	800	single density 9-track		
C0	1600	single density 9-track		
C0	1600	dual density 9-track		
C8	800	dual density 9-track		

Note: The first 15 entries in this table are valid only for 7-track tape. The last four entries are valid only for 9-track tape.

In order to read a 7-track tape backwards, the user must first create the tape file with the data convert feature off.

Also, under certain conditions, the user must be responsible for setting the mode of the tape to be processed. When using PIOCS with dual density tape units, a mode set must be issued if a mode is desired other than the one in which the tape was previously written. The user should position the tape at LOAD POINT and issue a SET MODE command, followed by a WRITE command.

ALT — Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The specifications for the alternate unit are the same as those of the original unit. The characteristics of the alternate unit must be the same as those of the original unit. The original assignment and an alternate assignment must both be permanent or temporary assignments. Multiple alternates can be assigned to a symbolic unit. When SYSIPT is assigned to a magnetic tape device, the file may not be multivolume.

TEMP — Indicates the assignment for the logical unit will be destroyed either by another ASSGN to the same unit or by the next JOB statement. Unless this option is taken, the assignment made is carried from job to job.

BATCH -- Start or Continue Batched-Job Operation

The BATCH command causes batched-job operation to start in F2 or F1, or to continue in BG, F1, or F2. The BATCH command is invalid unless the BJF option is specified at system generation time. If the specified partition is available, Job Control reads the operator's next command from SYSLOG. When the operator desires to give control to another command input device, he makes an assignment to SYSRDR or SYSIN, followed by the end-of-communication (Ⓑ) indication.

If the specified partition has been temporarily halted by a STOP command, it is made active. If the partition is in operation, it continues, and message

```
1P1nD AREA NOT AVAILABLE
```

is issued to the operator. In either instance, attention routine communication with the operator terminates following the BATCH command.

BATCH {BG / F2 / F1}

Accepted by AR.

If the operand is omitted, BG is assumed.

CANCEL -- Cancel Job Command

The CANCEL command, when used as a Job Control command or as a SPI command, cancels the execution of the current job in the partition in which the command is given. When given in a single-program foreground program, it resets all previous single-program initiation commands and returns control to the Supervisor.

When used as an ATTN command, it cancels the execution of the current program in the specified partition.

CANCEL blank

Accepted by JC and SPI.

CANCEL {BG / F1 / F2}

Accepted by AR.

BG — Indicates the background job is to be canceled.

F1 — Indicates the foreground-one job is to be canceled.

F2 — Indicates the foreground-two job is to be canceled.

If one of these operands is specified, the ATTN routines accept additional commands by issuing another read to SYSLOG. However, if the operand field is blank, the ATTN routines assume the background job is to be canceled.

CLOSE -- Close Output Unit

The CLOSE command is used to close either a system or programmer output logical unit assigned to a magnetic tape, or a system logical unit assigned to a 2311 or a 2314. The CLOSE statement is used to close either a system or programmer logical unit assigned to tape. It applies only to temporarily assigned logical units. The logical unit can optionally be reassigned to another device, unassigned, or, in the case of a magnetic tape file, switched to an alternate unit. When SYSxxx is a system logical unit (SYSLST, SYSPCH, etc), one of the optional parameters must be specified. When closing a programmer logical unit (SYS000-SYSmax), no optional parameter need be specified. When none is specified, the programmer logical unit is closed and the assignment remains unchanged. Closing a magnetic tape unit consists of writing a tapemark, an EOV trailer record, two tapemarks, and rewinding and unloading the tape. The trailer record contains no block count, and later access by logical IOCS may result in a 4131D message, which can be ignored. For a complete description of opening and closing system input/output units, see System I/O Operations.

```
CLOSE SYSxxx[{,X'cuu'[,X'ss']}]
             [{,UA          }]
             [{,IGN         }]
             [{,ALT         }]
```

Accepted by JC.

```
// CLOSE SYSxxx[{,X'cuu'[,X'ss']}]
               [{,UA          }]
               [{,IGN         }]
               [{,ALT         }]
```

Accepted by JC.

SYSxxx — For the CLOSE command only: For 2311 or 2314: SYSIN, SYSRDR, SYSIPT, SYSPCH, or SYSLST

For both the statement and the command: For magnetic tape: SYSPCH, SYSLST, SYSOUT, or SYS000-SYSmax

X'cuu' — Specifies that after the logical unit is closed, it will be assigned to the channel and unit specified. c is the channel number (0-6) and uu is the unit number 00-FE (0-254) in hexadecimal. In the case of a system logical unit, the new unit will be opened if it is either a disk or a magnetic tape at load point.

X'ss' — Device specification for mode settings on 7-track and 9-track tape. The specifications are shown in ASSGN -- Assign Logical Name. If X'ss' is not specified, the mode settings remain unchanged.

UA — Specifies that the logical unit is to be closed and unassigned.

IGN — Specifies that the logical unit is to be closed and unassigned with the ignore option. This operand is invalid for SYSRDR, SYSIPT, or SYSIN.

ALT — Specifies that the logical unit is to be closed and an alternate unit is to be opened and used. This operand is valid only for system output logical units (SYSPCH, SYSLST, or SYSOUT) currently assigned to a magnetic tape unit.

DATE Statement

This statement contains a date that is put in the communication region. It is in one of the following formats:

```
// DATE mm/dd/yy
```

```
// DATE dd/mm/yy
```

Accepted by JC.

mm = Month (01 to 12)

dd = Day (01 to 31)

yy = Year (00 to 99)

When the DATE statement is used, it applies only to the current job being executed. Job Control does not check the operand except for a length of eight characters. If no DATE statement is used, Job Control supplies the date given in the last SET command.

DLAB -- DASD Label Information

The DASD-label statement or command (completed in a continuation statement or command) contains file label information for DASD-label checking and creation. This statement or command must immediately follow a volume (VOL) statement or command.

```
// DLAB 'label fields 1-3',          C
      xxxx,yyddd,yyddd,'systemcode'[,type]
```

Accepted by JC.

```
[//] DLAB 'label fields 1-3'         C
      xxxx,yyddd,yyddd,'systemcode'[,type]
```

Accepted by SPI.

'label fields 1-3'
: The first three fields of the Format 1 DASD file label are contained just as they appear in the label. This is a 51-byte character string, contained within apostrophes and followed by a comma. The entire 51-byte field must be contained in the first of the two statements. Column 72 must contain a continuation character. The columns between the comma and the continuation character must be blank. The Format 1 label is shown in Appendix A. Fields 1-3 are:

  File Name. 44-byte alphameric including file ID and, if used, generation number and version number of generation.

  Format Identifier. 1-byte, EBCDIC 1.

  File Serial Number. 6-byte alphameric, must be the same as the volume serial number in the volume label of the first or only pack of the file.

C
: Continuation punch in column 72.

xxxx
: Volume Sequence Number. This 4-digit EBCDIC number is the EBCDIC equivalent of the 2-byte binary volume sequence number in field 4 of the Format 1 label. This number must begin in column 16 of the continuation statement. Columns 1-15 are blank.

yyddd,yyddd
: The File Creation Date, followed by the File Expiration Date. These two 5-digit numbers are the EBCDIC equivalent of the 3-byte discontinuous binary dates in fields 5 and 6 of the Format 1 label. yy is the year (00-99), and ddd is the day of the year (001-366).

'systemcode'
: System Code is a 13-character string, within apostrophes. For an output file, it is written in field 8 of the Format 1 label. It is ignored when used for an input file. This field is not used by the Disk Operating System label processing routines, but is essential in order for the files to be processable by the Operating System. It is recommended that this field be left blank.

type
: This is a two- or three-character field indicating the type of file, as follows:

  SD for Sequential Disk or for DTFPH with MOUNTED=SINGLE

  DA for Direct Access or for DTFPH with MOUNTED=ALL

  ISC for Indexed Sequential using Load Create

  ISE for Indexed Sequential using Load Extension, Add, or Retrieve

  If this operand is omitted, SD is assumed.

DLBL -- DASD Label Information

This command or statement replaces the VOL and DLAB command or statement combination used in previous versions of the system. It contains file label information for DASD label checking and creation. (Programming support for the previous VOL, DLAB, and XTENT combinations will be continued.)

```
// DLBL filename,
['file-ID'],[date],[codes]
```

Accepted by JC.

```
[//] DLBL filename,['file-ID'],
[date],[codes]
```

Accepted by SPI.

filename

This can be from one to seven alphameric characters, the first of which must be alphabetic. This filename is identical to the symbolic name of the program DTF that identifies the file.

'file-ID'

The name associated with the file on the volume. This can be from one to 44 bytes of alphameric data, contained within apostrophes, including file-ID and if used, generation number and version number of generation. If fewer than 44 characters are used, the field is left-justified and padded with blanks. If this operand is omitted, filename is used.

date

This can be from one to six characters indicating either the retention period of the file in the format d through dddd (0-9999), or the absolute expiration date of the file in the format yy/ddd (75/032). If 00/000 is specified, the expiration date is treated as an omitted operand. ddd can also be specified as one to three digits. If this operand is omitted, a 7 day retention period is assumed. If this operand is present on an input file, it is ignored.

codes

This is a two- or three-character field indicating the type of file label, as follows:

SD for Sequential Disk or for DTFPH with MOUNTED=SINGLE

DA for Direct Access or for DTFPH with MOUNTED=ALL

ISC for Indexed Sequential using Load Create

ISE for Indexed Sequential using Load Extension, Add, or Retrieve

If this operand is omitted, SD is assumed.

For output files, the current date is used as the creation date and DOS/360 VER 3 is used as the system code.

No continuation statements or commands are supported for DLBL.

A comma must be inserted for each missing operand if any of the operands following filename are used.

DVCDN -- Device Down

The DVCDN command informs the system that a device is no longer physically available for system operations. If a temporary assignment was made to the device specified in the command, the symbolic unit(s) for the device is unassigned when the command is accepted. If a standard assignment was made to the device, the symbolic unit(s) for the device is unassigned as a standard device.

If the unit is a DASD device, attempt to close any system I/O units currently assigned to it. The DVCDN command unassigns these units without closing them. If a DVCDN command is issued with a system I/O unit assigned to the DASD device, closing the file or reassigning it to a DASD device will be impossible. If an alternate assignment was made for the device specified, the alternate is removed. This command utilizes the logical transient area, and blocks out operator communication functions until it is completed.

```
DVCDN X'cuu'

  Accepted by JC.
```

The entry X'cuu' is expressed in hexadecimal form, where c is the channel number (0-6) and uu is the unit number, 00-FE (0-254) in hexadecimal.

DVCUP -- Device Up: The DVCUP command informs the system that a device is available for system operations after the device has been down. An ASSGN command must be used to reassign this device.

```
DVCUP X'cuu'

  Accepted by JC.
```

The entry X'cuu' is expressed in hexadecimal form, where c is the channel

number (0-6) and uu is the unit number, 00-FE (0-254) in hexadecimal.

EXEC -- Execute Program: The EXEC command or statement indicates the end of control information for a job step and that execution of a program is to start. It must be the last command or statement processed before a job step is executed.

The EXEC command specifies the single-program foreground program to be executed. The program must be cataloged in the core image library of the system. The EXEC command terminates the single-program initiation routines, and causes the specified program to be loaded into main storage.

```
[//] EXEC progname
```

Accepted by SPI.

```
// EXEC [progname]
```

Accepted by JC.

progname Represents the name of the program in the core image library to be executed. The program name corresponds to the phase name of the first (or only) phase of the program in the library. The program name can be one to eight alphameric characters.

If the program to be executed has just been processed by the Linkage Editor, the operand of the EXEC statement is blank. (This format is not valid for foreground programs.)

When control is given to a single program or to a fetched phase, general register 2 contains the address of the uppermost byte of storage available to the program.

EXTENT -- DASD Extent Information: The EXTENT command or statement defines each area, or extent, of a DASD file. One or more EXTENT commands or statements must follow each DLBL command or statement except for single volume input files for Sequential Disk on either a 2311 or a 2314, for which the DEVADDR parameter has been specified in the DTF table.

This command or statement replaces the XTENT command or statement used in previous versions of the system. (Programming support for XTENT continues.)

```
[//] EXTENT [symbolic-unit],
           [serial-number], [type],
           [sequence-number],
           [relative-track],
           [number-of-tracks],
           [split-cylinder-track],
           [B=bins]
```

Accepted by SPI.

```
// EXTENT [symbolic-unit],
          [serial-number], [type],
          [sequence-number],
          [relative-track],
          [number-of-tracks],
          [split-cylinder-track],
          [B=bins]
```

Accepted by JC.

symbolic unit
A six-character field indicating the symbolic unit (SYSxxx) of the volume for which this extent is effective. If this operand is omitted, the symbolic unit of the preceding EXTENT, if any, is used. If this operand is omitted on the first or only EXTENT command or statement, the symbolic unit specified in the DTF is assumed. A symbolic unit included in the extent information for SD or DA files, however, overrides the DTF DEVADDR=SYSnnn specification. (This operand is not required for an IJSYSxx filename, where xx is IN, PH, LS, LN, RS, SL, or RL, or for a file defined with the DTF DEVADDR=SYSnnn.) If SYSRDR or SYSIPT is assigned, this operand must be included.

In multivolume SD and DA files, each different symbolic unit must be assigned to a separate physical device. For DA files, the extent statements must be in ascending order.

serial number
From one to six characters indicating the volume serial number of the volume for which this extent is effective. If fewer than six characters are used, the field is right-justified and padded with zeros.

If this operand is omitted, the volume serial number of the preceding EXTENT is used. Therefore, when a multivolume file is being processed, the volume serial number of the first volume

is assumed for the entire file, unless the user specifies this field for the first extent of each following volume. If no serial number was provided in the EXTENT command or statement, the serial number is not checked and it is the user's responsibility if files are destroyed because the wrong volume was mounted.

type — One character indicating the type of the extent, as follows:

1 - data area (no split cylinder)

2 - independent overflow area (for indexed sequential file)

4 - index area (for indexed sequential file)

8 - data area (split cylinder, for SD files only)

If this operand is omitted, type 1 is assumed.

sequence number
One to three characters containing a decimal number from 0 to 255 indicating the sequence number of this extent within a multiextent file. Extent sequence 0 is used for the master index of an indexed sequential file. If the master index is not used, the first extent of an indexed sequential file has the sequence number 1. The extent sequence number for all other types of files begins with 0. If this operand is omitted for the first extent of ISFMS files, the extent is not accepted. For SD or DA files, this operand is not required.

relative track
One to five characters indicating the sequential number of the track, relative to zero, where the data extent is to begin. If this field is omitted on an ISFMS file, the extent is not accepted. This field is not required for SD input files (the extents from the file labels are used). This field must be specified for DA input files.

Formulas for converting actual to relative track (RT) and relative track to actual for the DASD devices follow.

Actual to Relative

2311 — 10 x cylinder number + track number = RT

2314 — 20 x cylinder number + track number = RT

2321 — 1000 x subcell number + 100 x strip number + 20 x cylinder number + track number = RT

Relative to Actual

2311 — $\frac{RT}{10}$ = quotient is cylinder, remainder is track

2314 — $\frac{RT}{20}$ = quotient is cylinder, remainder is track

2321 — $\frac{RT}{1000}$ = quotient is subcell, remainder1

$\frac{remainder1}{100}$ = quotient is strip, remainder2

$\frac{remainder2}{20}$ = quotient is cylinder, remainder is track

Example: Track 5, cylinder 150 on a 2311 = 1505 in relative track.

number of tracks
One to five characters indicating the number of tracks to be allotted to the file. For SD input, this field may be omitted. For the indexed sequential file, the number of tracks for prime data must be a multiple of 10 for 2311, and 20 for 2314. The number of tracks for a split cylinder file must be a multiple of the number of cylinders specified for the file and the number of tracks per cylinder specified.

split cylinder track
One or two characters, from 0-19, indicating the upper track number for the split cylinder in SD files.

bins — One or two characters identifying the 2321 bin for which the extent was created, or on which the extent is currently located. If the field is one character, the creating bin is assumed to be zero. There is no need to specify a creating bin for SD or ISFMS files. If this operand is omitted, bin zero is assumed for both bins. If the operand is included and positional operands

are omitted, only one comma is required preceding the key-word operand. (One comma for each omitted positional operand is acceptable, but not necessary.)

HOLD -- Hold Foreground Unit Assignments

The HOLD command causes all I/O assignments for the single program foreground area(s) specified to stay in effect from one job to the next.

```
HOLD {F1[,F2]}
     {F2[,F1]}
```

Accepted by SPI and JC.

If only one foreground area is referenced, it is assumed that the I/O assignments for the other are not to be held.

Unless this command is used, all I/O assignments are unassigned upon termination of a foreground program operating in single-program initiation mode.

Any assignments made in initiating a job to run in an area whose assignments are to be held, override the previous assignment to the logical unit specified.

If DASD file protection has been specified as a supervisor generation option, use of the HOLD command may result in an expansion of the Job Information Block (JIB) table to a point where it is impossible to initiate jobs in the partitions involved. The DASD file protect function uses the JIB table to store information concerning the DASD extents (used by the OPEN macro) along with other information for the job. When the HOLD command is used, assignments and JIB information are held across jobs. When the JIB table becomes loaded with extent information, an attempt to initiate additional jobs in the partition results in the error message indicating that no more JIBs are available. It is possible to circumvent this situation by limiting or avoiding use of the HOLD command for DASD devices (used by the foreground partitions) when the DASD file protect option has been specified.

JOB Statement

This statement indicates the beginning of control information for a job. The JOB statement is in the following format.

```
// JOB jobname
```

Accepted by JC.

jobname — The name of the job. Must be one to eight alphameric characters. When a job is restarted, the jobname must be identical to that used when the checkpoint was taken. Any user comments can appear on the JOB statement following the jobname (through column 72). If the timer feature is present, the time of day appears in columns 73-80 when the JOB statement is printed on SYSLST. The time of day is printed in columns 1-8 on the next line of SYSLOG.

LBLTYP -- Reserve Storage for Label Information

The LBLTYP statement defines the amount of main storage to be reserved at link edit time or at execution time (for self-relocating programs) for processing of tape and nonsequential disk file labels in the problem area of main storage. It applies to both background and foreground programs. It is to be submitted preceding the // EXEC LNKEDT statement, with the exception of self-relocating programs, for which it is instead submitted preceding the // EXEC statement for the program.

The LBLTYP command defines storage to be reserved by SPI for self-relocating programs. The operator submits the LBLTYP command preceding the EXEC command.

```
[//] LBLTYP {TAPE[(nn)]}
            {NSD(nn)   }
```

Accepted by SPI.

```
// LBLTYP {TAPE[(nn)]}
           {NSD(nn)   }
```

Accepted by JC.

TAPE[(nn)] — Used only if tape files requiring label information are to be processed, and no nonsequential DASD files are to

be processed. nn is optional, and is present only for future expansion (it is ignored by Job Control).

NSD(nn) Used if any nonsequential DASD files are to be processed regardless of other type files to be used. nn specifies the largest number of extents to be used for a single file.

The amount of storage that must be reserved for label information is:

1. For standard tape labels (any number): 80 bytes.

2. For sequential DASD and DTFPH MOUNTED=SINGLE: 0 bytes.

3. For DTFIS, DTFDA, and DTFPH MOUNTED=ALL: 84 bytes plus 20 bytes per extent.

The area reserved is that required by the file with the largest requirement. This area is used during OPEN.

LISTIO -- List I/O Assignment

The LISTIO command or statement causes the system to print a listing of I/O assignments. The listing appears on a SYSLOG (command) or SYSLST (statement).

```
        ( SYS    )
        ( PROG   )
        ( F1     )
        ( F2     )
LISTIO  ( ALL    )
        ( SYSxxx )
        ( UNITS  )
        ( DOWN   )
        ( UA     )
        ( X'cuu' )
```

Accepted by JC.

```
        ( BG  )
        ( F1  )
LISTIO  ( F2  )
        ( UA  )
        ( ALL )
```

Accepted by SPI.

```
          ( SYS    )
          ( PROG   )
          ( F1     )
          ( F2     )
// LISTIO ( ALL    )
          ( SYSxxx )
          ( UNITS  )
          ( DOWN   )
          ( UA     )
          ( X'cuu' )
```

Accepted by JC.

SYS Lists the physical units assigned to all system logical units.

PROG Lists the physical units assigned to all background programmer logical units.

F1 Lists the physical units assigned to all foreground-one logical units.

F2 Lists the physical units assigned to all foreground-two logical units.

ALL Lists the physical units assigned to all logical units.

SYSxxx Lists the physical units assigned to the logical unit specified. The assignment is given for the partition from which the command is given.

UNITS Lists the logical units assigned to all physical units.

DOWN Lists all physical units specified as inoperative.

UA Lists all physical units not currently assigned to a logical unit.

X'cuu' Lists the logical units assigned to the physical unit specified.

Physical units are listed with current device specification for magnetic tape units. Logical units are listed with ownership (background, foreground-one, or foreground-two), when applicable. If a unit has a standard assignment in one mode and a temporary assignment in another mode, the CMNT column identifies the type of assignment for each indicated mode. An example of a listing produced by the LISTIO SYS command is shown in Figure 11. All channel and unit numbers are represented in hexadecimal.

```
BG  LISTIO SYS
BG
BG                ***Background***
BG
BG    I/O UNIT  CMNT    CHNL    UNIT    MODE
BG
BG     SYSRDR              0      0C
BG     SYSIPT              0      0C
BG     SYSPCH              0      0D
BG     SYSLST              0      0E
BG     SYSLOG              0      1F
BG     SYSLNK              1      90
BG     SYSRES              1      91
BG     SYSSLB              1      92
BG     SYSRLB              1      92
BG     SYSREC              1      92
```

●Figure 11. Example of LISTIO SYS Output

LOG -- Log Command

The LOG job control command causes the system to log, on SYSLOG, columns 1-72 of all Job Control commands and statements occurring in the batched-job partition in which the LOG is issued. When issued as a single program initiation command, it causes logging of all single program initiation commands. The AR LOG affects all the partitions. The LOG function is effective until a NOLOG command for the partition involved is sensed.

LOG blank

Accepted by JC, AR, and SPI.

The operand field is ignored by the system.

MAP -- Map Main Storage

The MAP command causes the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment. It indicates which program(s) is being executed and which has access to the interval timer.

MAP blank

Accepted by JC, AR, and SPI.

The map of main storage produced is in the following format.

```
SP              upper limit
BG      size    upper limit   name
F2      size    upper limit   name
F1  T   size    upper limit   name
```

Field 1 Field 2 Field 3 Field 4

The fields indicate the following:

Field 1 (area identification)

SP - Supervisor

BG - Background area

F2 - Foreground-two area

F1 - Foreground-one area

T - Shows which program has interval timer support

Field 2 (size of area allocated)

The number of bytes allocated to the area in main storage. The size is printed in multiples of 2K, where 2K is equal to 2048 bytes. For the background area, this represents the number of full 2K blocks. For example: if the area is 11.2K, the map indicates 10K.

Field 3 (area upper limit of main storage)

The highest storage address allocated to the corresponding area is printed in decimal.

Field 4 (user name)

BG - Background job name

F2 - Foreground-two program name

F1 - Foreground-one program name

When NO NAME is specified for BG, or when the name field is blank for F2 or F1, no active program is being executed in the area. However, in any batched-job partition NO NAME is specified when no job card is entered, but the program is active.

MSG -- Transfer Control

The MSG command transfers control to a single foreground program operator communications routine previously activated by a STXIT instruction.

```
MSG {F1}
    {F2}
```

Accepted by AR and SPI.

F1 Used to request a foreground-one program STXIT routine.

F2 Used to request a foreground-two program STXIT routine.

If the specified program has established no operator communication linkage, a message is printed on the printer-keyboard informing the operator of this condition.

MTC -- Magnetic Tape Control

The MTC command or statement controls IBM 2400 series magnetic tape operations. The first operand specifies the operation to be performed.

```
MTC opcode,{X'cuu' }[,nn]
           {SYSxxx}
```

Accepted by JC.

```
// MTC opcode,SYSxxx[,nn]
```

Accepted by JC.

The first operand can be:

Opcode	Meaning
BSF	Backspace one file so tape is positioned for reading the tapemark preceding the file backspaced.
BSR	Backspace record.
ERG	Erase gap (write blank tape).
FSF	The tape is positioned beyond the tapemark following the file spaced over.
FSR	Forward space record.
RUN	Rewind and unload tape.
REW	Rewind tape.
WTM	Write tapemark.

The second operand, SYSxxx, represents any assigned logical unit.

X'cuu' is the channel and unit in hexadecimal where c is the channel number (0-6) and uu is the unit number, 00-FE (0-254).

The optional third entry, nn, is a decimal number (01-99) representing the number of times the specified operation is to be performed.

NOLOG -- Suppress Logging

The NOLOG command terminates the listing, on SYSLOG, of Job Control commands and statements (except JOB, PAUSE, STOP, ALLOC, MAP, HOLD, RELSE, UNA, DVCDN, DVCUP, *, and /&) that occur in the batched-job partition in which the NOLOG is issued. When issued as a single program initiation command, it terminates logging of single program initiation commands. The NOLOG function is effective until a LOG command for the partition involved is sensed.

```
NOLOG blank
```

Accepted by JC, AR, and SPI.

The operand field is ignored by the system.

OPTION Statement

This statement specifies one or more of the Job Control options. The format of the OPTION statement is:

```
// OPTION option1[,option2,...]
```

Accepted by JC.

The options that can appear in the operand field follow. Selected options can be in any order. Options are reset to the standards established at system generation time upon encountering a JOB or a /& statement.

LOG Causes the listing of columns 1-80 of all control statements on SYSLST. Control statements are not listed until a LOG option is encountered. Once a LOG option

statement is read, logging continues from job-step to job-step until a NOLOG option is encountered or until either the JOB or /& control statement is encountered.

NOLOG Suppresses the listing of all control statements on SYSLST until a LOG option is encountered.

DUMP Causes a dump of the registers and main storage to be output on SYSLST, if assigned, in the case of an abnormal program end (such as program check).

NODUMP Suppresses the DUMP option.

LINK Indicates that the object module is to be linkage-edited. When the LINK option is used, the output of the language translators is written on SYSLNK. The LINK option must always precede an EXEC LNKEDT statement in the input stream. (CATAL also causes the LINK option to be set.) LINK is not accepted by JOB Control operating in a foreground partition.

NOLINK Suppresses the LINK option. The language translators can also suppress the LINK option if the problem program contains an error that would preclude the successful execution of the problem program.

DECK Causes language translators to output object modules on SYSPCH. If LINK is specified, the DECK option is ignored, except by the PL/I (D) compiler, the FORTRAN IV compiler, the Assembler variant requiring 14K bytes of main storage, and the F Assembler.

NODECK Suppresses the DECK option.

LIST Causes language translators to write the source module listing on SYSLST. Also, causes the Assembler to write the hexadecimal object module listing and causes the Assembler and FORTRAN to write a summary of all errors in the source program. All are written on SYSLST.

NOLIST Suppresses the LIST option.

LISTX Causes the COBOL compiler to output a PROCEDURE DIVISION MAP on SYSLST. Causes the PL/I (D) compiler to output the object module on SYSLST.

NOLISTX Suppresses the LISTX option.

SYM Causes the Assembler to output the symbol table on SYSPCH, the PL/I (D) compiler to output the symbol table on SYSLST, the COBOL compiler to output a DATA DIVISION map on SYSLST.

NOSYM Suppresses the SYM option.

XREF Causes the Assembler to write the symbolic cross-reference list on SYSLST.

NOXREF Suppresses the XREF option.

ERRS Causes the FORTRAN, COBOL, and PL/I (D) compilers to summarize all errors in the source program on SYSLST.

NOERRS Suppresses the ERRS option.

CATAL Causes the cataloging of a phase or program in the core image library at the completion of a Linkage Editor run. CATAL also causes the LINK option to be set. CATAL is not accepted by Job Control operating in a batched-job foreground environment.

STDLABEL Causes all DASD or tape labels submitted after this point to be written at the beginning of the standard label track. Reset to USRLABEL option at end-of-job or end-of-job step. All file definition statements submitted after this option are available to any program in any area until another set of standard file definition statements is submitted. STDLABEL is not accepted by Job Control operating in a batched-job foreground environment. All file definition statements following OPTION STDLABEL are included in the standard file definition set until one of the following occurs:

1. End-of-job step.
2. End-of-job.
3. OPTION USRLABEL is specified.
4. OPTION PARSTD is specified.

OPTION STDLABEL followed by a /& clears the standard label track.

USRLABEL Causes all DASD or tape labels submitted after this point to be written at the beginning of the user label track.

PARSTD Causes all DASD or tape labels submitted after this point to be

written at the beginning of the partition standard label track. Reset to USRLABEL option at end-of-job or end of job step. All file definition statements submitted after this option are available to any program in the current partition until another set of partition standard file definition statements is submitted. All file definition statements submitted after OPTION PARSTD are included in the standard file definition set until one of the following occurs:

1. End-of-job step.
2. End-of-job.
3. OPTION USRLABEL is specified.
4. OPTION STDLABEL is specified.

OPTION PARSTD followed by a /& clears the partition standard label track.

For a given filename, the sequence of search for label information during an OPEN is the USRLABEL area, followed by the PARSTD area, followed by the STDLABEL area.

48C — Specifies the 48-character set on SYSIPT (for PL/I (D)).

60C — Specifies the 60-character set on SYSIPT (for PL/I (D)).

The options specified in the OPTION statement remain in effect until a contrary option is encountered or until a JOB control statement is read. In the latter case, the options are reset to the standard that was established when the system was originally generated.

Any assignment for SYSLNK after the occurrence of the OPTION statement cancels the LINK and CATAL options. These two options are also canceled after each occurrence of an EXEC statement with a blank operand.

Note: The LOG and NOLOG control statements defined for Basic Programming Support and the Basic Operating System are recognized by Job Control as equivalent to the LOG and NOLOG options.

PAUSE

The PAUSE Job Control command causes a pause at the end of the current job step. The PAUSE Job Control statement causes a pause immediately after processing this statement. At the time, the printer-keyboard is unlocked for message input. The end-of-communication indication (Ⓑ) causes processing to continue. The PAUSE statement or command is always printed on SYSLOG. If no 1052 is available, the PAUSE statement or command is ignored.

The PAUSE SPI command causes Job Control processing to pause at the end of the current batched-job job step or at the end of the current batched-job job.

```
// PAUSE [any user comment]
```

Accepted by JC.

```
PAUSE [any user comment]
```

Accepted by JC and SPI.

The PAUSE ATTN command normally causes Job Control processing to pause at the end of the current job step in the partition specified. Use of the optional operand causes Job Control processing to pause at end-of-job in the partition specified.

```
PAUSE {BG}
      {F2}[,EOJ]
      {F1}
```

Accepted by AR.

If the first operand is omitted, BG is assumed. The EOJ operand must be preceded by a BG, F1, or F2 operand.

READ -- Specify Reader

The READ command specifies a card reader from which further single-program initiation commands are to be read. The device specified must not be assigned to any other program.

```
READ X'cuu'
```

Accepted by SPI.

The entry X'cuu' is expressed in hexadecimal form, where c is the channel number (0-6) and uu is the unit number, 00-FE (0-254) in hexadecimal.

RELSE -- Release Foreground Unit Assignments at EOJ

The RELSE command causes all I/O assignments for the single-program foreground area(s) specified to be set to unassigned at the end of any job that is initiated for that area.

```
RELSE {F1[,F2]}
      {F2[,F1]}
```

Accepted by JC and SPI.

If only one foreground area is referenced, the I/O assignments for the other are unaffected.

RELSE command terminates a previous HOLD. Subsequent assignments are not held.

To immediately unassign a foreground area currently inactive, both the RELSE and UNA commands must be used.

RESET -- Reset I/O Assignments

The RESET command or statement resets certain I/O assignments to the standard assignments in partition in which submitted. The standard assignments are those specified when the system is generated, plus any modifications made by the operator via an ASSGN command without the TEMP option.

When the physical device affected by RESET is a magnetic tape drive, the current mode set in the PUB table is set to the standard mode set for the device. The standard mode set is established at IPL time and is modified by a permanent ASSGN with an X'ss' parameter.

```
RESET {SYS   }
      {PROG  }
      {ALL   }
      {SYSxxx}
```

Accepted by JC.

```
// RESET{SYS   }
        {PROG  }
        {ALL   }
        {SYSxxx}
```

Accepted by JC.

SYS	Resets all system logical units to their standard assignments.
PROG	Resets all programmer logical units to their standard assignments.
ALL	Resets all logical units to their standard assignments.
SYSxxx	Resets the logical unit specified to its standard assignment. SYSIN or SYSOUT cannot be specified.

ROD -- Record on Demand

The ROD command updates all SDR counters for all nonteleprocessing devices on the recorder file on SYSREC from the SDR counters in main storage. The command must not be issued until all partitions in an MPS environment have completed. The ROD command has no operand.

```
ROD  blank
```

Accepted by JC.

RSTRT -- Restart Checkpointed Program

A restart facility is available for checkpointed programs. A programmer can use the CHKPT macro instruction in his program to cause checkpoint records to be written. The maximum checkpoint that can be taken is decimal 9999. This allows

enough information to be stored so that program execution can be restarted at a specified point. The checkpointed information includes the registers, tape-positioning information, a dump of main storage, and a restart address.

The restart facility allows the programmer to continue execution of an interrupted job at a point other than the beginning. The procedure is to submit a group of job control statements including a restart (RSTRT) statement.

```
// RSTRT SYSxxx,nnnn[,filename]
```

Accepted by JC.

SYSxxx Symbolic unit name of the device on which the checkpoint records are stored. This unit must have been previously assigned.

nnnn Identification of the checkpoint record to be used for restarting. This serial number is four characters. It corresponds to the checkpoint identification used when the checkpoint was taken. The serial number is supplied by the checkpoint routine.

filename Symbolic name of the 2311 or 2314 disk checkpoint file to be used for restarting. It must be identical to the filename of the DTFPH to describe the disk checkpoint file and the fifth parameter of the CHKPT macro instruction. This operand applies only when specifying a 2311 or 2314 disk as the checkpoint file.

See the Supervisor and I/O Macros publication listed in the Preface for further details on the CHKPT macro instruction.

When a checkpoint is taken, the completed checkpoint is noted on SYSLOG. Restarting can be done from any checkpoint record, not just the last. The jobname specified in the JOB statement must be identical to the jobname used when the checkpoint was taken. The proper I/O device assignments must precede the RSTRT control statement.

Assignment of input/output devices to symbolic unit names may vary from the initial assignment. Assignments are made for restarting jobs in the same manner as assignments are made for normal jobs.

SET -- Set Value

The SET command initializes the date, clock, UPSI configuration, specifies the number of lines to be printed on SYSLST, specifies the remaining disk capacity when SYSLST or SYSPCH is assigned to disk, and defines to the system the status of the recorder file on SYSREC used by the I/O error logging (OBR/SDR) and machine check recording and recovery (MCRR) features. The SET card should precede the JOB card in job control sequence.

```
SET [DATE=n1][,CLOCK=n2][,UPSI=n3]
    [,LINECT=n4][,RCLST=n5][,RCPCH=n6]
    [,RF=n7]
```

Accepted by JC.

DATE=n1 Sets the system date permanently to the specified value. The system date in the communications region of each partition is reset to reflect the new value. This subsequently resets the JOB date when a new job is run. n1 has one of the following formats:

```
mm/dd/yy
dd/mm/yy
```

mm specifies the month; dd specifies the day; yy specifies the year. The format to be used is the format that was selected when the system was generated.

CLOCK=n2 Sets the system clock to the specified value. n2 has the following format:

```
hh/mm/ss
```

hh specifies hours (00-23); mm specifies minutes (00-59); ss specifies seconds (00-59).

UPSI=n3 Sets the bit configuration of the UPSI byte in the communications region. n3 consists of one to eight digits, either 0, 1, or X. Positions containing 0 are set to 0; positions containing 1 are set to 1; positions containing X are unchanged. Unspecified rightmost positions are assumed to be X.

LINECT=n4 Sets the standard number of lines to be printed on each page of SYSLST. n4 is an integer between 30 and 99.

RCLST=n5 n5 is a decimal number indicating the minimum number of records

remaining to be written on SYSLST when assigned to disk before a warning is issued to the operator that the capacity of the extent is near. It may be any decimal number from 100 through 65535.

Note: This warning is given only between jobs, and if the extent limits are to be exceeded before the next end-of-job step, this next job is terminated.

If no value is given, the system sets RCLST equal to the value specified in the SYSFIL parameter when the system was generated. If no value was specified, the system sets RCLST equal to 1000.

RCPCH=n6 n6 is a decimal number indicating the minimum number of records remaining to be written on SYSPCH when assigned to disk before a warning is issued to the operator that the capacity of the extent is near. It may be any decimal number from 100 through 65535.

Note: This warning is given only between jobs, and if the extent limits are to be exceeded before the next end-of-job step, this next job is terminated.

If no value is given, the system sets RCPCH equal to the value specified in the SYSFIL parameter when the system was generated. If no value was specified, the system sets RCPCH equal to 1000.

RF={YES / NO / CREATE}

Defines to the system the status of the recorder file (IJSYSRC) on SYSREC used by the I/O error logging (OBR/SDR) and machine check recording and recovery (MCRR) features.

YES Indicates that an active recorder file exists on the system and can be opened as an input file. The open takes place when the first JOB card is encountered.

NO indicates that no recording is to be done on the recorder file. OBR, SDR and MCRR features are suppressed, and the system enters the wait state with a unique message code (0S) in bytes 0 and 1 of main storage when a machine check occurs.

CREATE Instructs the system to create a recorder file when the first JOB card is encountered.

START -- Start Background or Foreground Processing

The START command can be used to initiate a single-program foreground program or to resume batched-job processing in any partition.

START {BG / F1 / F2}

Accepted by AR.

BG Causes Job Control to read the next control statement from SYSLOG. The START BG command is effective only if a STOP command was issued previously.

F1 or F2 Specifies either that a single foreground program is to be initiated in the partition, or that a batched-job foreground program that has been stopped by a STOP command is to be restarted.

In the first instance, the single program initiation routines are given control. Commands that may be issued following the START command are indicated in the section Single Program Initiation. If the specified foreground area is either being used by a program or has no area allocated to it, a message is printed on the printer-keyboard informing the operator of the condition.

STOP -- Stop Batched-Job Processing

The STOP command can be used in a multiprogramming environment to indicate that there are no more batched jobs to be

executed in the partition in which the command is given.

STOP blank

Accepted by JC.

This command removes the batched job from the system's task selection mechanism. Job Control remains in the partition and is available without reinitialization. If no program is being executed in another partition, the system is placed in the wait state. Either the START or the BATCH commands may be used to resume processing in the specified partition.

Note: In a multiprogramming environment it may be advisable to use a STOP command instead of a PAUSE command. The PAUSE command issues a read to SYSLOG, tying up the 1052 until the operator responds.

TIMER -- Interval Timer

The TIMER command gives interval timer support to the partition specified.

```
TIMER {BG}
      {F1}
      {F2}
```

Accepted by AR and SPI.

If interval timer support is already allocated to the partition specified, the command is ignored. (This may be as a result of the timer option specified when the system was generated, or a previous TIMER command.) If the interval timer was allocated to a different program and that program has an existing STXIT or SETIME linkage established, a message is printed on the printer-keyboard. If the command is accepted, the timer is set to the maximum interval. A subsequent STXIT or SETIME instruction issued by the program previously having access to the timer causes the cancelation of that program. Once established, timer support remains with an area from program to program until changed by a TIMER command.

TLBL -- Tape Label Information

This command or statement replaces the VOL and TPLAB command or statement combination used in previous versions of the system. It contains file label information for tape label checking and writing. (Programming support for the previous VOL and TPLAB combinations continues.) The TLBL command or statement may be used with both EBCDIC and ASCII files. For ASCII file processing, the fourth and fifth operands are called set identifier and file section number, respectively.

```
// TLBL filename,['file-ID'],[date],
        [{file-serial-number (Note 1)}],
        [{set-identifier (Note 2)    }]
        [{volume-sequence-number    }],
        [{  (Note 1)                }]
        [{file-section-number (Note 2)}]
        [file-sequence-number],
        [generation-number],
        [version-number]
```

Accepted by JC.

```
[//] TLBL filename,['file-ID'],[date],
        [{file-serial-number (Note 1)}],
        [{set-identifier (Note 2)    }]
        [{volume-sequence-number    }],
        [{  (Note 1)                }]
        [{file-section-number (Note 2)}]
        [file-sequence-number],
        [generation-number],
        [version-number]
```

Accepted by SPI.

Note 1: For EBCDIC files.
Note 2: For ASCII files.

filename
: This can be from one to seven alphameric characters, the first of which must be alphabetic. This filename is identical to the symbolic name of the program DTF that identifies the file.

'file-ID'
: One to 17 alphameric characters, contained within apostropes, indicating the name associated with the file on the volume. This operand may contain embedded blanks. On output files, if this operand is omitted, "filename" is used. On input files, if the operand is omitted, no checking will be done.

date
: Four to six numeric characters, in the format yy/ddd (75/032), indicating the expiration date of the file for output, or the

creation date for input. (The day of the year may have from one to three characters.) For output files, a one- to four-character retention period (d through dddd, 0-9999) may be specified. If this operand is omitted, a 0 day retention period is assumed for output files. For input files, no checking is done if this operand is omitted or if a retention period is specified. For output files, the current date is used as the creation date.

file serial number or set identifier
: One to six alphameric characters indicating the volume serial number of the first (or only) reel of the file. All six characters must be specified for ASCII files. For EBCDIC files, if fewer than six characters are specified, the field is right justified and padded with zeros. If this operand is omitted on output, the volume serial number of the first (or only) reel of the file is used. If the operand is omitted on input, no checking is done.

volume sequence number or file section number
: One to four numeric characters in ascending order for each volume of a multiple volume file. This number is incremented automatically by OPEN/CLOSE routines as required. If this operand is omitted on output, BCD 0001 is used. If omitted on input, no checking is done.

file sequence number
: One to four numeric characters in ascending order for each file of a multiple file volume. This number is incremented automatically by OPEN/CLOSE routines as required. If omitted on output, BCD 0001 is used. If omitted on input, no checking is done.

generation number
: One to four numeric characters that modify the file ID. If omitted on output, BCD 0001 is used. If omitted on input, no checking is done.

version number
: One or two numeric characters that modify the generation number. If omitted on output, BCD 01 is used for EBCDIC files, and BCD 00 for ASCII files. If omitted on input, no checking is done.

Additional fields of the standard tape file label are filled with default options for output files, with DOS/TOS/360 used as the system code.

TPLAB -- Tape Label Information

This command or statement, which can be used for both EBCDIC and ASCII files, contains file label information for tape label checking and writing. It must immediately follow a volume (VOL) command or statement. The TPLAB command or statement contains an image of a portion of the standard tape file label. The format and content of this label are presented in Appendix B. Label fields 3-10 are always included just as they appear in the label. These are the only fields used for label checking. The additional fields (11-13) can be included, if desired. If specified for an output file, they are written in the corresponding fields of the output label. They are ignored when used for an input file. These fields are never used by the IBM System/360 Disk Operating System label-processing routines.

```
// TPLAB {'label fields 3-10'}
         {'label fields 3-13'}
```

Accepted by JC.

```
[//] TPLAB {'label fields 3-10'}
          {'label fields 3-13'}
```

Accepted by SPI.

'label fields 3-10'
: This is a 49-byte character string, included within apostrophes (8-5 punch), identical to positions 5-53 of the tape file label. These fields can be included in one line.

'label fields 3-13'
: This is a 69-byte character string, included within apostrophes (8-5 punch), identical to positions 5-73 of the tape file label. These fields are too long to be included on a single line. The character string must extend into column 71, a continuation character (any character) is present in column 72, and the character string is completed on the next line. The continuation line starts in column 16.

UCS -- Load Universal Character Set Buffer

The UCS command causes the 240-character Universal Character Set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit. The 240 EBCDIC characters correspond to the 240 print positions on 1403 chains and trains. A character sent to the printer for printing is matched against the characters in the UCS buffer. When a match occurs, the corresponding chain/train character is printed in the print-line position that the output character occupied. Thus, the user, through the UCS buffer and the many chains/trains available, can adapt his 1403 Printer to many variable printing applications.

The logical unit must be assigned to an IBM 1403 Printer with the UCS feature. It is the user's responsibility to assemble, linkage edit, and catalog his UCS buffer phases into the core image library, and to mount the new chain or train before the UCS command is executed. The UCS command is not logged on SYSLST. (For more information on the UCS command, see the DOS User's Guide: Control Statement Techniques publication listed in the Preface.)

```
UCS SYSxxx,phasename[,FOLD][,BLOCK]
     [,NULMSG]
```

Accepted by JC and SPI.

SYSxxx The name of the logical unit assigned to a 1403 UCS printer to be loaded.

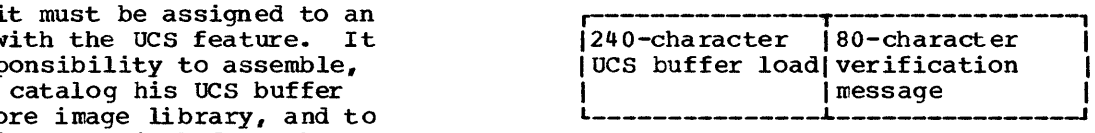
phasename
The symbolic name of the core image library phase containing the 240 EBCDIC characters to be loaded, followed by an 80-character verification message. Each phase may have any valid phasename.

FOLD Signifies that the buffer is to be loaded with the folding operation code in the CCW to permit printing either uppercase or lowercase bit configurations.

BLOCK Signifies that the 2821 latch is to be set to inhibit data checks generated by the 1403 UCS printer because of print line character mismatches with the UCS buffer.

NULMSG Signifies that the 80-character verification message is not to be printed on the 1403 after the buffer is loaded. If this parameter is not specified after the UCS buffer has been loaded, the program skips to channel 1, issues a print of the last 80 characters in the phase specified by the first parameter, and again skips to channel 1. This is to identify the phase, if the phasename is incorporated in the verification message. If the user's chain/train can be identified by a unique character, this message can also be used to verify that the mounted chain or train is compatible with the UCS buffer contents, by including this unique character in the verification message.

The UCS phase format is:



UNA -- Immediately Unassign Foreground Unit Assignments Command

The UNA command immediately causes all I/O assignments for the single program foreground area(s) specified to be set to unassigned.

```
UNA {F1[,F2]}
    {F2[,F1]}
```

Accepted by JC and SPI.

The foreground area specified must be currently inactive. This command is used to free physical units currently assigned to a foreground area under the HOLD command. A previous HOLD for the area remains in effect, and any future assignments in the area are held. To immediately unassign logical units in the area and prevent future assignments from being held, both the UNA and RELSE commands must be used.

UNBATCH -- Terminate Batched-Job Processing

The UNBATCH command causes batched-job foreground processing to be terminated and the partition to be released. UNBATCH is accepted only when no job is in process in the partition and only from SYSLOG. The operator can gain command of the 1052

printer-keyboard following a PAUSE or STOP command or a PAUSE statement. All tape or disk system I/O files must have been closed. Following the UNBATCH command, the attention routine accepts BATCH or START commands for the affected partition. UNBATCH permits storage allocation for the partition to be reduced, and also performs the functions that are automatically provided when a program called by the single program initiator goes to end-of-job. This command is valid only for foreground partitions, and when the BJF option is specified.

UNBATCH blank

Accepted by JC.

UPSI -- Set User Program Switch Indicators

This statement allows the user to set program switches that can be tested much the same as sense switches or lights used on other machines. The UPSI statement has the following format.

// UPSI nnnnnnnn

Accepted by JC.

The operand consists of one to eight characters of 0, 1, or X. Positions containing 0 are set to 0. Positions containing 1 are set to 1. Positions containing X are unchanged. Unspecified rightmost positions are assumed to be X.

Job Control clears the UPSI byte to zeros before reading control statements for each job. When Job Control reads the UPSI statement, it sets or ignores the bits of the UPSI byte in the communication region. Left to right in the UPSI statement, the digits correspond to bits 0 through 7 in the UPSI byte. Any combination of the eight bits can be tested by problem programs at execution time.

VOL -- Volume Information

The VOL command or statement is used when standard labels for a DASD or tape file are checked unless the DLBL or TLBL commands or statements are used. A VOL command or statement must be used for each file on a multifile volume (when the DLAB or TPLAB commands or statements are used). The VOL, TPLAB or VOL, DLAB, XTENT commands or statements must appear in that order and must immediately precede the EXEC command or statement to which they apply.

// VOL SYSxxx,filename

Accepted by JC.

[//] VOL SYSxxx,filename

Accepted by SPI.

SYSxxx Symbolic unit name (present for compatibility with the Tape Operating System). The symbolic unit name is taken from the XTENT command or statement.

filename
This can be from one to seven alphameric characters, the first of which must be alphabetic. This filename is identical to the symbolic name of the program DTF that identifies the file.

XTENT -- DASD Extent Information

The XTENT command or statement defines each area, or extent, of a DASD file. One or more XTENT commands or statements must follow each DLAB command or statement.

// XTENT type,sequence,lower,upper,
'serial no.',SYSxxx[,B_2]

Accepted by JC.

[//] XTENT type,sequence,lower,upper,
'serial no.',SYSxxx[,B_2]

Accepted by SPI.

type Extent Type. 1 or 3 columns, containing:

1 = data area (no split cylinder)

2 = overflow area (for indexed sequential file)

4 = index area (for indexed sequential file)

128 = data area (split cylinder). If type 128 is specified, the lower head is assumed to be $H_1H_2H_2$ in lower, and the upper head $H_1H_2H_2$ in upper.

sequence
Extent Sequence Number. 1-3 columns, containing a decimal number from 0 to 255, indicating the sequence number of this extent within a multiextent file. Extent sequence 0 is used for the master index of an indexed sequential file. If the master index is not used, the first extent of an indexed sequential file has sequence number 1. The extent sequence for all other types of files begins with 0.

lower Lower Limit of Extent. 9 columns, containing the lowest address of the extent in the form $B_1C_1C_1C_2C_2C_2H_1H_2H_2$, where:

B_1 = initially assigned cell number.

0 for 2311 and 2314
0 to 9 for 2321

C_1C_1 = Subcell number.

00 for 2311 and 2314
00 to 19 for 2321

$C_2C_2C_2$ = cylinder number.

000 to 199 for 2311 and 2314

or

strip number:

000 to 009 for 2321

H_1 = head block position.

0 for 2311 and 2314
0 to 4 for 2321

H_2H_2 = head number.

00 to 09 for 2311
00 to 19 for 2321 or 2314

Although a part of the address (such as B_1 or $C_2C_2C_2$) can be zero, a lower extent of all zeros is invalid.

Note: The last 4 strips of subcell 19 are reserved for alternate tracks for 2321.

upper Upper Limit of Extent. 9 columns containing the highest address of the extent, in the same form as the lower limit.

'serial no.'
Volume Serial Number. This is a 6-byte alphameric character string, contained within apostrophes. The number is the same as in the volume label (volume serial number) and the Format 1 label (file serial number).

SYSxxx This is the symbolic address of the DASD drive.

B_2 Currently assigned cell number.

0 for 2311 and 2314
0-9 for 2321

This field is optional. If missing, Job Control assigns $B_2=B_1$.

Ⓑ -- End-of-Communication

The end-of-communication command must be issued whenever the operator is finished communicating with the system. It causes the communication routine to return control to the mainline job.

Note: CANCEL (without an operand), BATCH, and START commands automatically terminate ATTN communication.

When single program initiation commands are entering through a card reader as the result of a READ command, and an invalid command is encountered, an error message prints on the printer-keyboard. Further single program initiation commands can then be read from the printer-keyboard. The end-of-communication command causes input reading to switch back to the device specified in the READ command.

Ⓑ blank

Accepted by JC, AR, and SPI.

Ⓑ is the end-of-block character, alter code 5.

/* -- End of Data File

This statement must be the last statement of each input data file on SYSRDR and SYSIPT.

```
/* ignored
```

Accepted by JC.

Columns 1 and 2 contain a slash (/) and an asterisk (*). Column 3 must be blank. /* causes the channel scheduler to post the end-of-file indicator in the user's CCB. Logical IOCS also recognizes /* when a card reader is assigned to the symbolic units SYS000-SYSmax.

/& -- End of Job

This statement must be the last statement of each job.

```
/& ignored
```

Accepted by JC.

Columns 1 and 2 contain a slash and an ampersand (12-punch). Column 3 must be blank. Upon occurrence of /&, the channel scheduler posts an end-of-file indicator in the user's CCB. If the user attempts to read past the /& on SYSRDR or SYSIPT, the job is terminated. Any comments can begin in column 14 and are printed at end-of-job.

The end-of-job statement is printed on SYSLOG and SYSLST in the following format:

Columns 1-3 contain EOJ, columns 5-12 the job name, columns 14-72 blanks or any user comments. If the timer feature is present, print positions 73-98 of SYSLST contain the time of day and the job duration in the following format:

```
hh.mm.ss,DURATION hh.mm.ss
```

It is printed in the same format, occupying 26 positions, on the line following the end-of-job statement on SYSLOG.

End-of-job information is not printed on SYSLST if // OPTION NOLOG has been specified. (The NOLOG statement itself is logged on SYSLST).

* -- Comments

This statement can be used as a Job Control comments statement.

```
* any user comments
```

Accepted by JC.

Column 1 contains an asterisk. Column 2 is blank. The remainder of the statement (through column 72) contains any user comments. The content of the comment statement is printed on SYSLOG. If followed by a PAUSE statement, the statement can be used to request operator action.

Type of Command or Statement	Name	Accepted by		
		JC	SPI	AR
Job Identification	JOB	X		
	/&	X	N5	
File Definition	DLAB	X	X	
	DLBL	X	X	
	EXTENT	X	X	
	TLBL	X	X	
	TPLAB	X	X	
	VOL	X	X	
	XTENT	X	X	
	/*	X	N5	
Pass Information to Operator	*	X		
Pass Information to Program	DATE	X		
	LBLTYP	X	X	
	OPTION	N1		
	UPSI	X		
Job Stream Control	BATCH			N2
	CANCEL	X	X	X
	PAUSE	X	X	X
	READ		X	
	START			N3
	STOP	X		
	UNBATCH	N4		
Setting System Parameters	ALLOC	X		X
	SET	X		
	TIMER		X	X
Operator Communications	LOG	X	X	X
	MSG		N3	N3
	NOLOG	X	X	X
	(B)	X	X	X

●Figure 12. Commands and Statements by Function (Part 1 of 2)

Type of Command or Statement	Name	Accepted by		
		JC	SPI	AR
Control of I/O System	ASSGN	X	X	
	CLOSE	X		
	DVCDN	X		
	DVCUP	X		
	HOLD	N3	X	
	LISTIO	X	X	
	MAP	X	X	X
	MTC	X		
	RELSE	N3	X	
	RESET	X		
	ROD	X		
	UCS	X	X	
	UNA	N3	X	
Execution of Program	EXEC	X	X	
	RSTRT	X		

Notes:

N1 - OPTION LINK, OPTION STDLABEL, and OPTION CATAL are available only in the background partition.
N2 - Valid only if the batched-job foreground option has been specified during system generation.
N3 - Valid only for MPS.
N4 - Valid only in a batched-job foreground partition.
N5 - The Single Program Initiator does not recognize these statements. The Supervisor posts an end-of-file in the user's CCB when they are read on SYSIPT and cancels any program that reads past /&.

•Figure 12. Commands and Statements by Function (Part 2 of 2)

SYSTEM I/O OPERATIONS

This section describes the:

- Opening and closing of magnetic tape and disk devices when assigned to system logical units.
- Opening and closing of magnetic tape and disk devices when assigned to three programmer logical units (SYS001 - SYS003).

The IBM-supplied utility macros can be used to prepare magnetic tapes or disk extents to be used as SYSRDR, SYSIPT, and/or SYSIN. They may also be used to convert SYSPCH and SYSLST output on disk or tape, and SYSOUT output on tape into printed and/or punched card output.

SYSRDR records must be 80 characters in length, SYSLST records are 121 characters, and SYSPCH records 81 characters in length. Job Control accepts either 80- or 81-character records from SYSIPT, unless SYSIPT is assigned to a 2314. (Only 80-character records are acceptable from SYSIPT assigned to a 2314.) Thus, object modules produced when SYSPCH was assigned to a magnetic tape or to a 2311 disk extent can be read by Job Control as Linkage Editor input when the tape or disk extent is later assigned to SYSIPT. The first character of the SYSLST and SYSPCH records is assumed to be an ASA carriage control or stacker selection character. SYSIPT, SYSRDR, SYSPCH, and SYSLST records assigned to DASD have no keys, and record lengths are the same as stated. When SYSIPT is assigned to a magnetic tape device, the file cannot be multivolume.

OPEN System Tape Files

When the system logical unit SYSRDR, SYSIPT, SYSPCH, or SYSLST is assigned, Job Control checks to determine whether the device assignment is to a magnetic tape device positioned at load point. If so, Job Control performs an OPEN that:

1. Accepts a leading tapemark for input or output.
2. Accepts a completely unlabeled tape (other than blank tape) for input. Nonstandard labels are not recognized.
3. Accepts any label set starting with VOL1 (including user labels) for input.
4. Accepts a valid IBM-standard label set (including user labels) having a past expiration date for output.
5. Rejects all other volumes and gives the operator the option of either accepting the invalid label (IGNORE) or mounting an acceptable volume and replying NEWTAP.

All tapes mounted on single density 9-track drives (800 bpi or 1600 bpi) must be written in the same density that the drives can read or write.

CLOSE System Tape Output Files

When SYSPCH, SYSLST, or the combined output file, SYSOUT, is assigned to magnetic tape, the system processes any end-of-volume condition that may occur. The system provides the following functions:

1. Closes the affected file(s) and rewinds and unloads the reel. A tapemark, an EOV trailer record, and two tapemarks are written before the rewind.
2. Provides automatic volume switching; or if alternates are not specified, provides the ability to change tape reels.
3. Prints a message on SYSLOG informing the operator of an end-of-volume condition on either SYSLST, SYSPCH, or SYSOUT.
4. Opens the new or alternate tape volume.

If an alternate unit was not assigned, or if all assigned alternates were currently active as other system files, a message prints on SYSLOG requesting the operator to mount a new reel. The system automatically continues when the device is readied.

If a tape reel (alternate drive or the same drive) cannot be opened, a message is printed on SYSLOG. The operator can either mount a new reel or use the current reel.

System Disk Input and Output Files

In systems with at least 24K positions of main storage, the system logical units SYSRDR, SYSIPT, SYSIN, SYSLST and/or SYSPCH can be assigned to an extent of 2311 or 2314 disk storage.

If both SYSRDR and SYSIPT are to be assigned to disk, they must be assigned to the same extent and be referred to as SYSIN. SYSLST and SYSPCH must be assigned to separate extents. Thus, SYSOUT cannot be used to refer to a combined SYSLST/SYSPCH on 2311 or 2314 disk.

The assignment of system logical units to extents of disk storage must be permanent. The operator ASSGN command must be used instead of the programmer statement (// ASSGN). Temporary assignments (via the // ASSGN statement) to other device types are permitted. Thus, a job not in the input job stream on disk could be run by causing a pause at the end of the current job, temporarily assigning SYSRDR to a card reader or a magnetic tape unit, and running the job. At completion, the assignment for SYSRDR reverts to the disk assignment.

The system generation parameter SYSFIL is required to allow assignment of system logical units to a disk. It provides for warning the operator when SYSPCH and SYSLST files on disk reach a certain (specified or assumed) capacity. (See SET command in the Job Control Commands section.)

> Note: This warning is given only between jobs, and if the extent limits are to be exceeded before the next end-of-job step, this next job is terminated.

Assigning System Files to Disk

System input and output files are assigned to disk by providing a set of DLBL and EXTENT statements and then submitting a permanent ASSGN Command. The set of DLBL and EXTENT statements preceding the ASSGN command must contain only one EXTENT statement.

The filename in the DLBL statement (which will be associated with the SYSxxx entry from the accompanying EXTENT statement) must be one of the following:

IJSYSIN for SYSRDR, SYSIPT, or the combined SYSRDR/SYSIPT file SYSIN

IJSYSPH for SYSPCH

IJSYSLS for SYSLST

IJSYSRC for SYSREC

Note: A combined SYSPCH/SYSLST file (SYSOUT) may not be assigned to disk.

In the DLBL statement, the codes operand must specify SD (or blank, which means SD) to indicate sequential DASD file type.

In the EXTENT statement, the type operand may be 1 (data area, no split cylinder) or 8 (data area, split cylinder). There is no unique requirement for the remaining operands of the EXTENT statement.

The ASSGN command must be one of the following:

1. ASSGN SYSIN,X'cuu' (for a combined SYSRDR/SYSIPT file).
2. ASSGN SYSRDR,X'cuu' (for SYSRDR only).
3. ASSGN SYSIPT,X'cuu' (for SYSIPT only).
4. ASSGN SYSPCH,X'cuu' (for SYSPCH).
5. ASSGN SYSLST,X'cuu' (for SYSLST).
6. ASSGN SYSREC,X'cuu' (for SYSREC).

Note: All must be permanent assignments.

OPEN System Disk Files

Upon encountering a system input or output assignment to 2311 or 2314 Job Control performs these functions:

1. Rejects the assignment if it is not permanent.
2. Rejects the assignment if a previous assignment to 2311 or 2314 for the same logical unit still exists (has not been closed). Also, because SYSRDR and SYSIPT must be a single combined file if both are on disk, one cannot be assigned to disk if an assignment to disk for the other (or both) already exists.
3. OPENs the file. If input, the labels are checked. If output, DASD labels are written. Also, information is placed into the Supervisor for the problem program OPEN, and for monitoring of file operations by physical IOCS.
4. If the OPEN is unsuccessful, Job Control unassigns the unit and requests further operator commands.

CLOSE System Disk Files

System logical units assigned to disk must be closed by the operator. The operator CLOSE command must be used to specify a system input or output file which has been previously assigned to a 2311 or 2314. The optional second parameter (X'cuu') of the CLOSE command can be used (instead of an ASSGN command) to unassign the system logical unit or to assign it to a physical device. The system notifies the operator that a CLOSE is required when the limit of the file has been exhausted. If a program attempts to read or write beyond the limits of the file, the program will be terminated and the file must be closed.

The CLOSE function:

1. Writes a file mark if the file is an output file.
2. Resets the I/O control table in the Supervisor to indicate that the file no longer exists.
3. Reassigns the logical unit to the value of the second operand of the CLOSE command.

CONTROL STATEMENT EFFECT ON I/O UNITS

Certain control statements in the Job Control input stream affect the use of system I/O units by Job Control. These statements are:

// JOB
This statement must be the first statement of the job. When the JOB statement is encountered, the content of the statement is printed on SYSLOG and SYSLST. The first JOB card causes the SYSREC file to be opened if the I/O error logging and the machine check recording and recovery options have been specified. All I/O assignments are reset to the standards established when the system was generated, plus any

modifications that may have been made by the operator at IPL time and between jobs and job steps. If SYSLST is assigned to a printer, it is first skipped to a new page. If SYSLST is assigned to a magnetic tape or disk a carriage eject character is prefixed to the card image, which is then written on tape.

/&

When Job Control encounters /& on SYSRDR during normal operation, the standard assignment for SYSIPT becomes effective and SYSIPT is checked for an end-of-file condition. If the standard assignments for SYSRDR and SYSIPT are not to the same device, SYSIPT is advanced to the next /& statement. The end-of-job statement is written on SYSLOG and the last line of SYSLST. (The end-of-job information is not printed on SYSLST if // NOLOG or // OPTION NOLOG has been specified.)

In the event of an abnormal termination, Job Control advances SYSRDR and SYSIPT to the next /&, and proceeds, only if a // JOB statement was provided.

Beware of omitting /&, because protection of one job from errors in the preceding job cannot then be guaranteed. The // JOB statement is required in all cases.

Job Control has no responsibility for the arrangement of output on any file, except that connected with control statements. Such items as page ejection and line count must be managed by the individual processing program. The first line printed on SYSLST must be preceded by a line skip or a page eject. Otherwise, an overprint may result.

WORK FILES USED BY SYSTEM COMPONENTS

The system logical unit SYSLNK and the four programmer logical units SYS001, SYS002, SYS003, and SYS004 are used as work files by the various system components (Linkage Editor, Librarian, Language Translators, etc). SYSLNK is always assigned to a single area (extent) in 2311 or 2314 disk storage. SYS001, SYS002, SYS003, and SYS004 can be assigned to either disk or tape.

Disk Work Files

The programmer must provide ASSGN, DLBL, and EXTENT information for each file (SYSLNK and SYS001-SYS004) to be used by the system components. The only time the ASSGNs are not necessary is if, at IPL time, the logical units are permanently assigned to specific physical devices, and all jobs use those particular assigns. The information is used for the OPEN and CLOSE routines when called by the system components. SYSLNK is opened and closed by Job Control and the various system components when OPTION LINK or CATAL is specified. SYS001, SYS002, SYS003, and SYS004 are opened and closed by each component that uses these files as work files. The filenames for SYSLNK and SYS001-SYS004 on the DLBL statement are IJSYSLN, IJSYS01, IJSYS02, IJSYS03, and IJSYS04 respectively.

For convenience, these label information statements may be stored on the standard label information track so that they need not be submitted with each job. See the section Job Control: Edit and Store Label Information.

DASD file protection does not fully protect work files with expired labels in a multiprogramming environment, as follows: OPEN treats files with expired labels the same, whether or not they are in use. Thus, a foreground program may open and write in an area on disk that overlaps on a background program's work file (with expired labels), even if the background program is still using the file. Protection can be achieved by creating an unexpired label for work files, by the use of an expiration date such as 99/365. These work files can be used by subsequent jobs, because:

- The CLOSE used by the assembler and compilers erases the references to them from their respective VTOCs.
- They are not maintained from job to job by the system so that no check for equal file ID is made when a work file of this type is created.

The following additional precautions may be taken by the user to provide additional protection for his DASD work files.

1. Use of a unique volume serial number on all DASD volumes.
2. Identification of temporary files and of partition usage in the file ID label field.

3. Use of unique extent limits for work files in each partition.

Tape Work Files

The work files SYS001, SYS002, SYS003, and SYS004 are opened and closed by each system component that uses them. The programmer does not provide label information for these tape work files.

JOB CONTROL STATEMENT EXAMPLE

Figure 13 is an example of job control statement input (SYSRDR=SYSIPT) required to perform a series of background program job steps in an installation using unlabeled magnetic tape. In the discussion that follows, each point corresponds to the number at the left of the two slashes in the job control statements. The PHASE, INCLUDE, and ENTRY statements are Linkage Editor control statements. These statements are described in detail in the section Linkage Editor. They are included in this discussion to present a more meaningful example.

1. JOB statement for the series of job steps to be performed.

2. ASSGN statements required for the job steps. It is assumed that the label information for SYSLNK has been stored and that the assignments differ from those specified when the system was generated. The new assignments are carried through for the entire job and are reassigned at the end of the job to the standards established at system generation time.

3. OPTION statement specifying that the output of the Basic FORTRAN compilation and Assembler assembly will be written on SYSLNK for subsequent linkage editing and that the dump option will be exercised in the event of an abnormal end of job.

4. EXEC statement for a Basic FORTRAN compilation, followed by the Basic FORTRAN source deck and the end-of-data-file indicator (/*).

5. EXEC statement for an assembly, followed by the source deck and the end-of-data-file indicator.

6. EXEC statement for the Linkage Editor. The Linkage Editor edits the combined Basic FORTRAN and Assembler object programs on SYSLNK and writes the edited program temporarily in the core image library.

7. EXEC statement for the linkage-edited object program in the core image library. The input data for the execution is followed by the end-of-data-file indicator.

8. PAUSE statement that requests operator action.

9. OPTION statement specifying that the no-dump option be exercised. The link option must be included to enable a new linkage-edit. (An EXEC statement with a blank operand suppresses the LINK option.)

10. INCLUDE statements for modules in the relocatable library that are to be included with the object deck on SYSIPT. A blank operand indicates that the program to be included follows on SYSIPT. The resulting program is edited and written in the core image library.

11. EXEC statement for the program to be executed. The data for the execution is followed by the end-of-data-file indicator.

12. PAUSE statement that requests operator action.

13. End-of-job indicator. All symbolic unit assignments are reset to the standards established when the system was generated.

14. JOB statement for the next job.

```
 1 // JOB          EXAMPLE1

 2 ASSGN          SYSLNK,X'191'
   ASSGN          SYS001,X'180'
   ASSGN          SYS002,X'181'
   ASSGN          SYS003,X'182'

 3 // OPTION      LINK,DUMP

 4 // EXEC        FORTRAN
   (FORTRAN Source Deck)
   /*

 5 // EXEC        ASSEMBLY
   (Assembler Source Deck)
   /*

 6 // EXEC        LNKEDT

 7 // EXEC
   (Data for User Object Program)
   /*

 8 /&

 9 // PAUSE       SAVE SYS001, MOUNT SCRATCH TAPE

10 // JOB          EXAMPLE2

11 // OPTION      NODUMP, LINK

12    PHASE       PHNAM,ORIGIN
      INCLUDE     SORT
      INCLUDE     SINE
      INCLUDE
   (Object Deck to be Included)
   /*
      ENTRY
   // EXEC        LNKEDT

13 // EXEC
   (Data for User Object Program)
   /*

14 // PAUSE       SAVE SYS002

15 /&

16 // JOB          NEXT
```

●**Figure 13. Job Control Statement Example**

Initial Program Loader (IPL)

Operation of the Disk Operating System is initiated through an initial program load (IPL) procedure from the resident disk pack. The operator places the resident disk pack on a drive, selects the address of that drive in the load unit switches, and presses the load key. This causes the first record on track 0 to be read into main storage bytes 0-23. The information read in consists of an IPL PSW and two CCWs, which in turn cause the reading and loading of the IPL.

Operating in the supervisor state, IPL reads the Supervisor nucleus into low main storage. If a read error is sensed while reading the Supervisor nucleus, the wait state is entered and an error code is set in the first word of main storage. The IPL procedure must then be restarted.

After successfully reading in the Supervisor nucleus, IPL performs these operations.

1. Sets the LUB table entry for SYSRES to point to the PUB entry of the channel and unit number of the resident drive.

2. Places the processing unit in the wait state with all interruptions enabled. When the wait state is entered, the operator decides whether a 1052 or a card reader will be used to communicate with the system. If a 1052, the request key is pressed; if a card reader that is not assigned as SYSRDR at system generation time, it is brought to the ready state; if a card reader that is assigned as SYSRDR at system generation time, the interrupt key on the console is pressed.

3. Changes the PUB configuration, if indicated, by adding or deleting a device. When a device is deleted, all references to it are removed. A device may be added only if additional space was made available in the PUB table. This is specified as a system generation parameter. If a tape is to be added, there must also be enough space for an associated Tape Error Block (TEB) if TEBs are specified as a system generation parameter.

To add a device to the PUB table, a control statement, read by the communication device (1052 or card reader), in the following format is required.

Operation	Operand
ADD	X'cuu'[(k)],devicetype[,X'ss']

where:

X'cuu' = channel and unit numbers.

k = S if the device is to be switchable (the device is physically attached to two adjacent channels). The designated channel is the lower of the two channels. If the device is not switchable, k = 0-255, indicating the priority of the device, with 0 indicating the highest priority. If k is not given, the assumed priority is 255.

devicetype = actual device (2400T9, 1443, etc). See device codes in Figure 14.

X'ss' = device specifications (see ASSGN Statement). If absent, the following values are assigned:
X'C0' for 9-track tapes
X'90' for 7-track tapes
X'00' for nontapes
X'00', X'01', X'02', and X'03' are invalid as X'ss' for magnetic tape.

This parameter specifies SADxxx (Set ADdress) requirements for IBM 2702 lines:
X'00' for SAD0
X'01' for SAD1
X'02' for SAD2
X'03' for SAD3
This information is not accepted on the ASSGN statements.

X'ss' is required for 1270, 1275, 1412, 1419, and 1419P device types. It specifies the external interrupt bit in the old PSW, which is used by this device to indicate "read complete". The specifications are:

X'01' PSW bit 31
X'02' PSW bit 30
X'04' PSW bit 29
X'08' PSW bit 28
X'10' PSW bit 27
X'20' PSW bit 26

The X'ss' parameter specifies whether or not the error correction feature is present on an IBM 1018 Paper Tape Punch with 2826 Control Unit Model 1. These specifications are:

X'00' no error correction feature
X'01' error correction feature

To delete a device from the PUB table, a control statement, read by the communication device (1052 or card reader), in the following format is required.

Operation	Operand
DEL	X'cuu'

where cuu is the channel and unit numbers of the device to be deleted.

Note: ADD and DEL statements are issued only at IPL time.

The only communication required at IPL time is the date and, if the timer is present, the time of day. It must follow any ADD or DEL statements. It is entered via the communication device (1052 or card reader) and is in the following format.

Operation	Operand
SET	DATE=value1[,CLOCK=value2]

value1 Has one of the following formats.

mm/dd/yy
dd/mm/yy

mm specifies the month; dd specifies the day; yy specifies the year. The format to be used is the format that was selected when the system was generated.

value2 Has the fcllowing format.

hh/mm/ss

hh specifies hours; mm specifies minutes; ss specifies seconds. The CLOCK parameter is required only if timer support was indicated at system generation time.

After completing these operations, IPL loads the Job Control program into the background, which begins processing the control statements for the first job. Control statements are present on the device assigned to SYSRDR.

If the operator wishes to change any symbolic unit assignments for the background, ASSGN statements or commands are entered via the communication device (1052 or card reader). The ASSGN statements or commands are as described in Descriptions and Formats of Commands and Statements.

Card Code	Actual Device	Device Type
2400T9	Nine Track Magnetic Tapes (2400 Series)	Tapes
2400T7	Seven Track Magnetic Tapes (2400 Series)	
2495TC	2495 Tape Cartridge Reader	Tape Cartridge Reader
1442N1	1442N1 Card Read Punch	Card Readers - Punches
2520B1	2520B1 Card Read Punch	
2501	2501 Card Reader	Card Readers
2540R	2540 Card Reader	
2540P	2540 Card Punch	Card Punches
2520B2	2520B2 Card Punch	
1442N2	1442N2 Card Punch	
2520B3	2520B3 Card Punch	
1403	1403 Printer	Printers
1403U	1403 Printer with UCS Feature	
1404	1404 Printer	
1443	1443 Printer	
1445	1445 Printer	
1050A	1052 Printer - Keyboard	1050 Control Unit
UNSP	Unsupported Device	Unsupported. No Burst Mode on Multiplexor Channel
UNSPB	Unsupported Device	Unsupported with Burst Mode on Multiplexor Channel
2260 (Local)	2260 or 2265 Display Station	Display Unit
2260 (Local)	A 1053 attached to a 2848. The mode operand must be entered as X'01'	Printer
2311	2311 Disk Storage Drive	DASD
2314	2314 Direct Access Storage Facility	DASD
2321	2321 Data Cell Drive	DASD

Card Code	Actual Device	Device Type
2701	2701 Data Adapter Unit	Teleprocessing Lines
2702	2702 Transmission Control Unit	
2703	2703 Transmission Control Unit	
2703	System/360 Model 25 with the Integrated Communications Attachment and its Synchronous Data Adapter	
2671	2671 Paper Tape Reader	Paper Tape Readers
1017	1017 Paper Tape Reader with 2826 Control Unit, Model 1	
1017TP	1017 Paper Tape Reader with 2826 Control Unit, Model 2	
1018	1018 Paper Tape Punch with 2826 Control Unit, Model 1	Paper Tape Punches
1018TP	1018 Paper Tape Punch with 2826 Control Unit, Model 2	
7770	7770 Audio Response Unit	Audio Response Unit
7772	7772 Audio Response Unit	
1285	1285 Optical Reader	Optical Readers
1287	1287 Optical Reader	
1288	1288 Optical Page Reader	
1412	1412 Magnetic Character Reader	MICR - Magnetic Ink Character Recognition Devices and Optical Reader/Sorters
1419	Single Address Adapter 1419 Magnetic Character Reader, 1259 Magnetic Character Reader or 1270/1275 Optical Reader/Sorter	
1419P	Primary Address, Dual Address Adapter 1419 Magnetic Character Reader or 1275 Optical Reader/Sorter	
1419S	Primary Address, Dual Address Adapter 1419 Magnetic Character Reader or 1275 Optical Reader/Sorter	

Figure 14. Device Code Entries for Device Type Parameter in ADD Statement

Linkage Editor

All programs executed in the Disk Operating System environment must be edited by the Linkage Editor. The Linkage Editor reads the relocatable output of the language translators and edits it into executable, nonrelocatable programs in the core image library. The linkage editor performs on one program at a time; that is, it cannot linkage edit a series of programs concurrently. Once a program is edited, it can be executed immediately, or it can be cataloged as a permanent entry in the core image library. When a program has been cataloged in the core image library, the Linkage Editor is no longer required for that program. The program is run as a distinct job step and is loaded directly from the resident pack by the System Loader.

The extent of the editing function performed depends on the structure of the input program. The simplest case is that of a single-module program. The Linkage Editor has only to edit the program, creating a single phase entry in the core image format. This corresponds to the first diagram in Figure 15.

In more complex situations, the operation may involve linking together and relocating multiple-control sections from separate assemblies to produce a number of separate phases (see the last diagram in Figure 15). The Linkage Editor resolves all linkages (symbolic reference) between segments of the program and relocates the phases to load at specified main-storage locations.

To facilitate writing and testing large programs, assembled program modules cataloged in the relocatable library can be combined with other modules from SYSIPT (card, tape, or disk).

> Note: A supervisor cannot be cataloged into the core image library when multiprogramming is in progress.

Stages of Program Development

The term program could be confused with several things. The programmer codes sets of source statements that may be a complete program or part of a program. These source statements are then compiled or assembled into a relocatable machine-language program which, in turn, must be edited into an executable program, and may be combined with other programs. Consequently, it is convenient to refer to each stage of program development by a particular name.

A set of source statements that is processed by a language translator (Assembler, COBOL, FORTRAN, RPG, or PL/I (D)), is referred to as a source module.

The output of a language translator is referred to as an object module. All object modules must be further processed by the Linkage Editor before they can be executed in the system. Each element in the relocatable library is called a module. These relocatable modules each consist of a single object module.

The output of the Linkage Editor consists of one or more program phases in the core image library. A phase is in executable, nonrelocatable, core image form. Each separate phase is loaded by the System Loader in response to a FETCH or LOAD macro.

Structure of a Program

Source Module

A source module is input to a language translator and consists of definitions for one or more control sections. When the source module is translated, the output (object module) consists of one or more defined control sections. Each control section is a block of code assigned to contiguous main-storage locations. The input for building a phase (a section of a program loaded as a single overlay) must consist of one or more complete control sections.

Object Module

An object module is the output of a complete language translator run. It consists of the dictionaries and text of one or more control sections. The dictionaries contain the information necessary for the Linkage Editor to resolve cross references between different object modules. The text consists of the actual

instructions and data fields of the object module. The program cards produced by the language translators (as distinct from the Linkage Editor control statements discussed in the Linkage Editor section) have an identifier field in columns 1-4 that indicates the content of the card. The following card types, except REP cards, are produced by the language translators:

Identification	Contents or Meaning
ESD	External Symbol Dictionary
TXT	Text
RLD	Relocation List Dictionary
REP	Replacement to Text made by the programmer
END	End of a Module

REP cards, when required, appear after the TXT cards or after the RLD cards and before the END card. They are produced by the programmer. The format of each of these card types is shown in Appendix E.

ESD (External Symbol Dictionary): The external symbol dictionary contains all the symbol and storage assignments for an object module. For example, it contains all symbols defined in this module that are referred to by some other module. It can contain all symbols referred to by this module that are defined in some other module.

The six classifications of the ESD record recognized by the Linkage Editor are:

1. SD (Section Definition): Specifies the symbolic name, the assembled origin, and the length of the control section. SD is generated by a named START or named CSECT in a source module.

2. LD/LR (Label Definition/Label Reference): Defines a label that may be used by any other separately assembled module as an entry point, a data label, etc. The LD specifies the assembled address of the external label and a pointer to the section definition item (in this module) that describes the control section containing the external label. LD is generated by ENTRY in a source module. The LD is termed LR when the entry is matched to an ER entry.

3. ER (External Reference): Refers to control sections and external labels defined in other separately assembled modules. ER is generated by EXTRN or a V-type address constant in a source module.

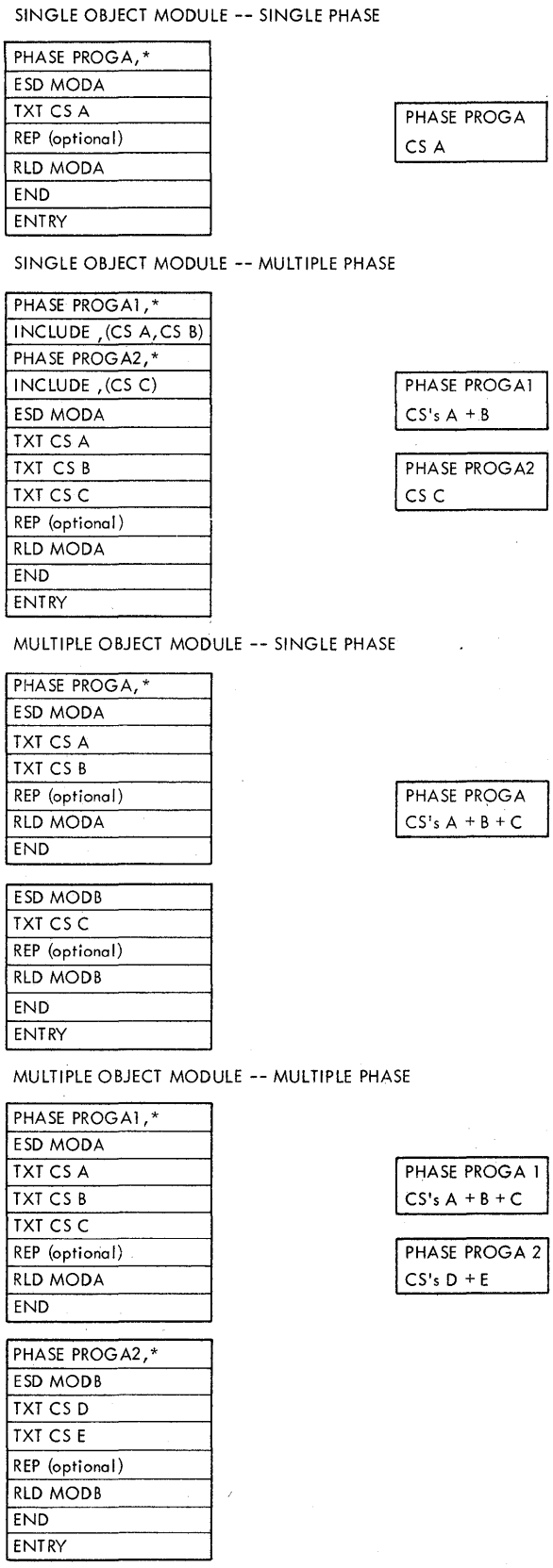


Figure 15. Linkage Editor Input and Output

4. WX is generated by WXTRN (Weak External Reference). WXTRN has a function similar to EXTRN. The difference is that WXTRN suppresses AUTOLINK for the symbols identified by WXTRN. When a WXTRN is encountered, it is converted to a regular EXTRN, subtype NOAUTO.

5. PC (Private Code): Refers to control sections that are unnamed in an assembly. Private code is a special type of section definition containing the assembled origin and the control section length. The name field is filled with blank characters. PC is generated by an unnamed START or unnamed CSECT in a source module.

6. CM (COMMON): Indicates the amount of main storage that must be allocated by this module for a COMMON area to be shared between phases. CM is generated by COM in a source module.

TXT (Text): The program that is eventually loaded into storage for execution is contained within the TXT cards. The text card contains the assembled origin of the instructions or data included in the card, and also the count of the number of bytes contained in the card. This card includes a reference to the control section in which this information occurs that allows the relocation factor involved to be applied. TXT information will be modified as required by RLD information.

RLD (Relocation List Dictionary): The relocation list dictionary cards identify portions of the TXT that must be modified due to relocation. They provide information necessary to perform the relocation. RLD cards are generated by relocatable address constants in a source module.

END (End of Module): The END card indicates end of module to the Linkage Editor. The END card may supply a transfer address. It may also contain the control section length, which was not previously specified in the ESD section definition or private code.

REP (User Replace Card): The REP card allows replacement (substitution) of new text for portions of assembled text. Each REP card contains the assembled address of the first byte to be replaced, the identification of the control section to which it refers, and may contain from 2 to 22 bytes of text. The text is substituted, byte for byte, for the original text, beginning at the address specified. The address, the control section reference, and the new text must be stated in hexadecimal. The REP card must be placed within the module that it modifies.

Program Phase

A program phase, the output of the Linkage Editor, is that section of a program that is loaded as a single overlay with a single FETCH or LOAD by the System Loader. Programs may consist of many phases, the first fetched by Job Control, and each of the rest by a preceding program phase. Successive phases of a multiphase program are often called overlays.

The input for building a single phase consists of the text from one or more complete control sections. When building a phase, the Linkage Editor constructs composite ESD and composite PHASE data, known as the control dictionary, and a composite RLD from each of the modules that make up the phase. These composite dictionaries are used to resolve all linkages between different control sections as if they had been assembled as one module. Each control section within the phase is relocated as necessary, and the entire phase is assigned a contiguous area of main storage. All relocatable address constants are modified to contain the relocated value of their symbols. The Linkage Editor always ensures that each phase or control section begins on a double-word boundary.

Each phase is constructed by building the text in the core image format. Thus, a phase may consist of one or more blocks of contiguous core image locations. Backward origin within a phase may cause slower operation of the Linkage Editor.

Types of Linkage Editor Runs

The Linkage Editor is run as a distinct job step. Because of this fact, it is meaningful to classify it as one of the system service programs. The Linkage Editor function is performed as a job step in three kinds of operations (Figure 16).

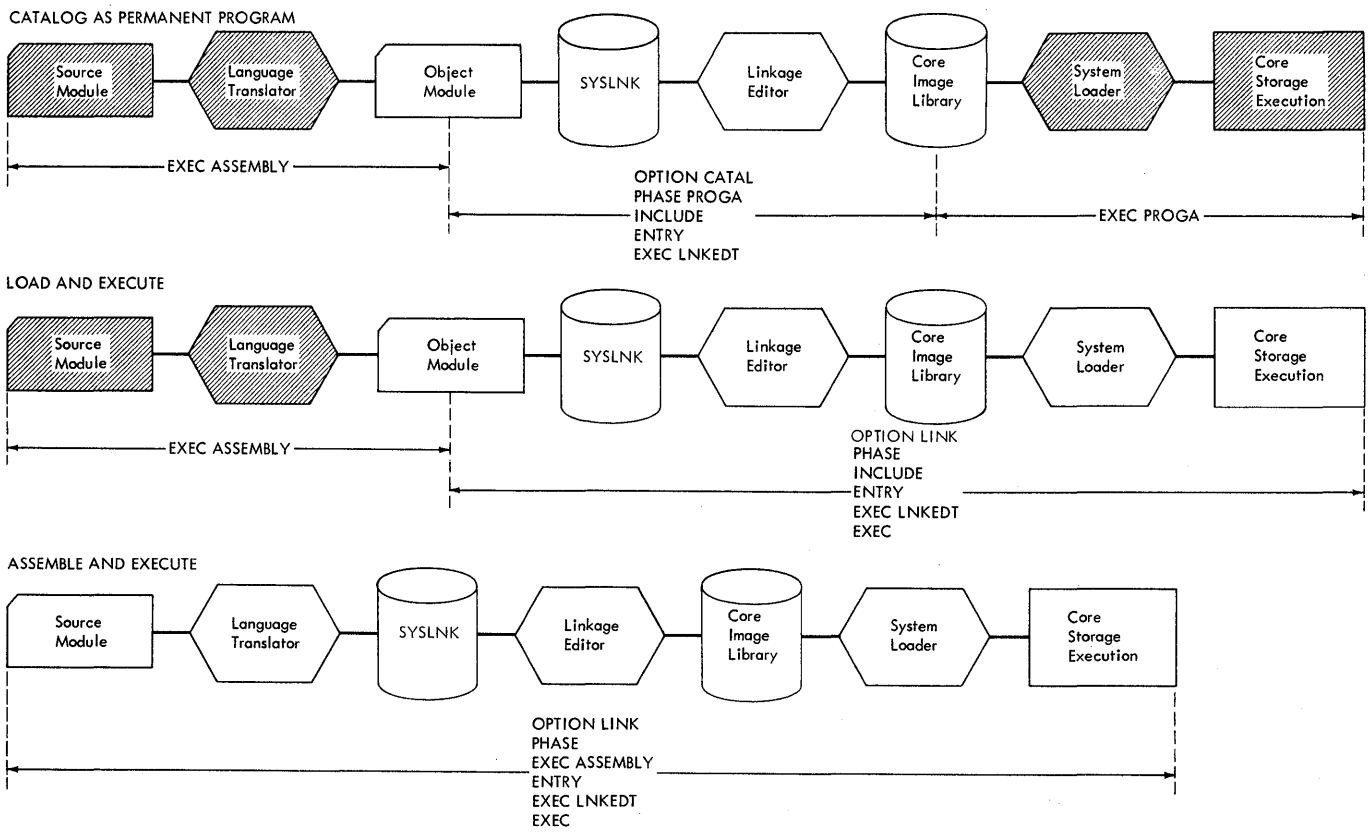


Figure 16. Three Types of Linkage Editor Operations

1. Catalog Programs in Core Image Library. The Linkage Editor function is performed immediately preceding the operation that catalogs programs into the core image library. By specifying the CATAL option, the Linkage Editor not only edits the programs, but also catalogs them permanently in the core image library. Figure 16 (first diagram) shows the sequence of events when this operation is performed. The input for the LNKEDT function could include modules from the system or private (if assigned) relocatable library instead of, or in addition to, those modules from the card reader, tape unit, or disk extent assigned to SYSIPT. This is accomplished by including the name of the module to be included in an INCLUDE statement.

2. Load-and-Execute. Figure 16 (second diagram) shows the sequence of events when this operation is performed. Specifying OPTION LINK causes Job Control to open SYSLNK and allows Job Control to place the object module(s) and Linkage Editor control statements on SYSLNK. Just as with the catalog operation, the input can consist of object modules from the system or private (if assigned) relocatable library instead of, or in addition to, those modules from the card reader, tape unit, or disk extent assigned to SYSIPT. This is accomplished by including the name of the module to be included in the operand of an INCLUDE statement. After the object modules have been edited and placed in the core image library, the program is executed. The blank operand in the EXEC control statement indicates that the program just linkage-edited and temporarily stored in the core image library is to be executed.

3. Assemble- or Compile-and-Execute. Source modules can be assembled or compiled and then executed in a single sequence of job steps. In order to do this, the language translator is directed to output the object module directly in SYSLNK. This is done by using the LINK option in the OPTION control statement. Upon completion of this output operation, the Linkage Editor function is performed. The program is linkage edited and temporarily stored in the core image library. Figure 16 (third diagram) shows the sequence of events when this operation is performed.

Linkage Editor Control Statements

In addition to the program cards previously listed, object modules used as input for the Linkage Editor include Linkage Editor control statements. There are four kinds of these control statements.

PHASE — Indicates the beginning of a phase. It gives the name of the phase and the main-storage address where it is to be loaded. It may also cancel the Automatic Library Lookup (AUTOLINK) feature.

INCLUDE — Signals that an object module is to be included. A blank operand indicates to Job Control that the module is on SYSIPT. INCLUDE statements with blank operands are recognized only on SYSRDR. Each series of relocatable modules on SYSIPT must be terminated by a /* control statement. The first optional entry in the operand indicates that a module by that name is to be included from the relocatable library. The second optional operand indicates that only selected control sections are to be included from a module.

ENTRY — Provides an optional transfer address for the first phase.

ACTION — Specifies options to be taken.

The first (or only) object module input for the Linkage Editor should include a PHASE control statement before the first ESD item. If no PHASE statement is used, or if the PHASE statement is in error, the Linkage Editor will construct a dummy statement. This will allow the user to test the program; however, the program with the dummy PHASE statement cannot be cataloged in the core image library. The last (or only) object module may optionally be followed by an ENTRY control statement. The rules governing placement of INCLUDE and other PHASE control statements are discussed under Control Statement Placement.

Sources of Input

Input for the Linkage Editor always begins from the area in disk assigned to SYSLNK. Object module input to the Linkage Editor can be:

1. Output from the language translator programs immediately after a compilation or assembly (SYSLNK).
2. From the card reader, tape unit, or disk extent assigned to SYSIPT.
3. From SYSIPT and from the relocatable library.
4. From the relocatable library.

In the first case, the LINK option of the OPTION control statement indicates to Job Control and to language translators that the output resulting from an assembly or compilation will be written directly on SYSLNK.

In the second case, an INCLUDE statement with a blank operand indicates to Job Control that the input on SYSIPT up to the /* statement is to be copied on SYSLNK.

In the third case, an INCLUDE statement with a blank operand indicates to Job Control the presence of input on SYSIPT. An INCLUDE statement with an entry in the operand indicates that a module in the relocatable library is to be included in the program phase by the Linkage Editor.

In the fourth case, an INCLUDE statement with an entry in the operand indicates that a module in the relocatable library is to be included in the program phase. Modules in the relocatable library can also contain INCLUDE statements.

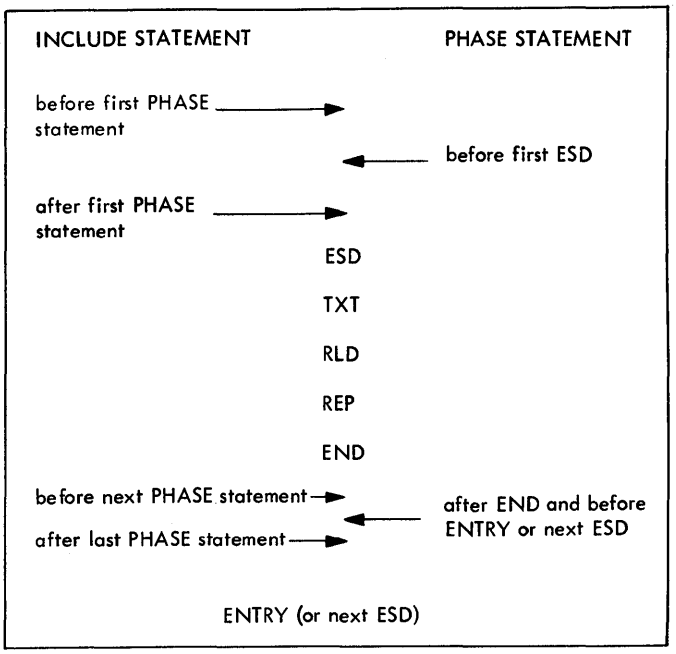
General Control Statement Format

The Linkage Editor control statements are similar in format to statements processed by the Assembler. The operation field must be preceded by one or more blanks. The operation field must begin to the right of column 1 and must be separated from the operand field by at least one blank position. The operand field is terminated by the first blank position. It cannot extend past column 71.

Control Statement Placement

When preparing multiple-object modules in a single Linkage Editor run, the single ENTRY statement should follow the last object module. The ACTION statement(s) must be the first record(s) encountered in the input stream; otherwise, they are ignored.

PHASE and INCLUDE statements may be present on SYSRDR, SYSIPT, or in the relocatable library. Figure 17 shows the possible placement of the PHASE and INCLUDE statements.



Note: INCLUDE statements within modules in the relocatable library must precede the ESD statement for the module.

Figure 17. Placement of PHASE and INCLUDE Statements

PHASE STATEMENT

The PHASE statement must precede the first object module of each phase processed by the Linkage Editor. Under no circumstances can a PHASE statement occur within a control section. There can be several control sections within a phase. When several PHASE statements appear before an object module, each of the statements must be followed by at least one INCLUDE statement. Any object module not preceded by a PHASE statement will be included in the current phase.

This statement provides the Linkage Editor with a phase name and an origin point for the phase. The phase name is used to catalog the phase in the core image library. This name is used in a FETCH or LOAD macro to retrieve the phase for execution. The PHASE statement is in the following format.

Name	Operation	Operand
blank	PHASE	name,origin[,NOAUTO]

The name field is blank. The operation field contains PHASE. Entries in the operand field must be separated by commas. The entries in the operand field represent the following.

name Symbolic name of the phase. One to eight alphameric characters are used as the phase name. Multiphase programs should have phase names of five to eight alphameric characters. The first four characters of each phase within a multiphase background program should be the same, but should not be identical to the first four characters of other phases in the core image library. Under this condition, or if a dollar sign ($) is used as the first character of a phase name, bytes 40-43 of the communication region will not contain an accurate indication of the uppermost byte used when loading problem program phases. An asterisk cannot be used as the first character of a phase name.

origin Specifies the load address of the phase. If COMMON is used, the length of the largest COMMON is added to every phase origin, even if the origin is given as an absolute value. The load address can be in one of six forms:

1. symbol[(phase)][±relocation]
2. *[±relocation]
3. S[+relocation]
4. ROOT
5. +displacement
6. F+address

The elements that make up the six forms that specify the origin signify the following.

1. symbol: May be a previously defined phase name, a previously defined control section, or a previously defined external label (the operand of an ENTRY source statement).

   (phase): If symbol is a previously defined control section or a previously defined external label that appears in more than one phase, phase (in parentheses) directs the Linkage Editor to the phase in which the origin name is to be found. The

phase name must have been defined previously.

relocation: Indicates the origin of the phase currently being processed will be set relative to the symbol by a relocation term consisting of a + or a - immediately followed by a hexadecimal number X'hhhhhh' of one to six digits, a decimal number ddddddd of one to eight digits, or nK, where K=1024.

2. *: Indicates the Linkage Editor phase location counter. It will cause the Linkage Editor to assign the next main-storage location (with forced double-word alignment) as an origin for the next phase. In the case of the first PHASE statement, it signifies the first double-word address after the end of the Supervisor, the label block area (if any), and the area assigned to the COMMON pool (if any). Refer also to ACTION Statement, F1 and F2 operands.

   relocation: Indicates relocation of the phase as described in item 1.

3. S: Indicates the origin is to be made at the end of the Supervisor, the label block area (if any), and the area assigned to the COMMON pool (if any). Refer also to ACTION Statement, F1 and F2 operands.

   relocation: Indicates relocation of the phase as described in item 1, although negative relocation is invalid for S-type.

4. ROOT: Causes the Linkage Editor to assume the phase that follows as a root phase that will always be resident in main storage while the program is being executed. The main-storage address assigned to this phase is the first double-word address after the end of the Supervisor, the label block area (if any), and the area assigned to the COMMON pool (if any). Only the first PHASE statement is permitted to specify ROOT. Any qualitative information (phase or relocation) is ignored when ROOT is specified. If a control section appears in the root phase, other occurrences of the same control section are ignored and all references are resolved to the control section in the root. Control sections are not duplicated within the same phase. If any subsequent phase overlays any part of the ROOT phase, a warning diagnostic is displayed on SYSLST if ACTION MAP is specified. Refer also to ACTION Statement, F1 and F2 operands.

5. +displacement: Allows the origin point (loading address) to be set at a specified location. The origin point is an absolute address, relative to zero. displacement is a hexadecimal number X'hhhhhh' of one to six digits, a decimal number ddddddd of one to eight digits, or nK, where K=1024. A displacement of zero (+0) would be used to denote a self relocating program. If COMMON is used, the COMMON start address is resolved to the end of supervisor address.

6. F: Indicates a foreground program is being linkage edited and an area is to be reserved at the beginning of the foreground area for the program name, a register save area, and label information. F should never be used for self relocating programs. If COMMON is used, the COMMON start address is resolved to the first double-word boundary after the reserved area at the beginning of the area specified by the F + displacement in the PHASE card.

   +address: The positive absolute main storage address of the foreground area in which the linkage-edited program is to be executed. It may be specified by a hexadecimal number X'hhhhhh' of four to six digits, or a decimal number ddddddd of five to eight digits, or in the form nnnnK, where n is two to four digits and K=1024. For example, an address may be specified as +32K or +X'8000' or +32768. The origin of the phase is on the first double-word boundary after the sum of address, the adjustment for the save area requirements, the label area and the length of the COMMON area if applicable.

NOAUTO Indicates that the Automatic Library Look-up (AUTOLINK) feature is suppressed for both the private relocatable library and the system relocatable library. AUTOLINK collects each unresolved external reference from the phase. It then searches the private relocatable library (if SYSRLB has been assigned) and then the system relocatable library for a cataloged object module with the same name as each unresolved external reference. When a match is found, the module in the private relocatable library, or in the system relocatable library is edited into the phase. The AUTOLINK retrieved module must have an entry point matching the external reference in order to resolve its address. Unresolved external references are processed sequentially in alphameric order. Object-module cross references with labels identical to library object-module entry-point labels are erroneous. The use of NOAUTO as the last operand in a PHASE statement causes the AUTOLINK process to be suppressed for that phase only. (Also see ACTION Statement.)

Some examples of PHASE statements follow.

```
PHASE PHNAME,*+504
```

This causes loading to start 504 bytes past the end of the previous phase.

```
PHASE PHNAME3,PHNAME2
```

This causes loading to start in the same point where the loading of the phase by the name PHNAME2 started.

```
PHASE PHNAME,ROOT
```

This causes loading to start after the end of the Supervisor, the label block area (if any) and the COMMON pool (if any) in the problem program area. When the PHASE statement contains a ROOT origin, this PHASE statement must be the first PHASE statement read by the Linkage Editor. Otherwise, it is treated as a symbol.

```
PHASE PHNAME,CSECT1(PHNAME2)
```

This causes loading to start at the point where CSECT1 was loaded. CSECT1, the named control section, must have appeared in the phase named PHNAME2.

```
PHASE PHNAME,F+X'6000'
```

This causes loading to start at 24K plus the length of the save area and label area.

```
PHASE PHNAME,F+32K
```

This causes loading to start at 32K plus the length of the save area and label area.

```
PHASE PHNAME1,F+30K
PHASE PHNAME2,*
PHASE PHNAME3,PHNAME2
```

The first phase (PHNAME1) of the preceding series is loaded starting at 30K plus the length of the save area and label area. The second phase (PHNAME2) of the series is loaded at the end of PHNAME1. The third phase (PHNAME3) is loaded at the same address as was PHNAME2, i.e., at the end of PHNAME1.

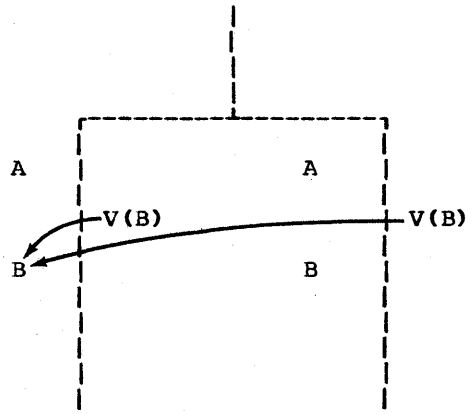
Note: In each of the preceding examples, if the origin address supplied is not on a double-word boundary, the Linkage Editor automatically increments to the next double-word boundary.

The Linkage Editor allows the inclusion of the same control section within each of several phases. If a control section appears in a ROOT phase, it will not appear in any other phase. A duplicate control section within the same phase will be ignored.

As external references occur in a phase, they are resolved preferentially with the entry point within the ROOT phase (if any), or the last previous occurrence of this entry point. For example, the coding

```
A   START
      .
      .
      .
    DC V(B)
      .
      .
      .
B   CSECT
      .
      .
      .
    END
```

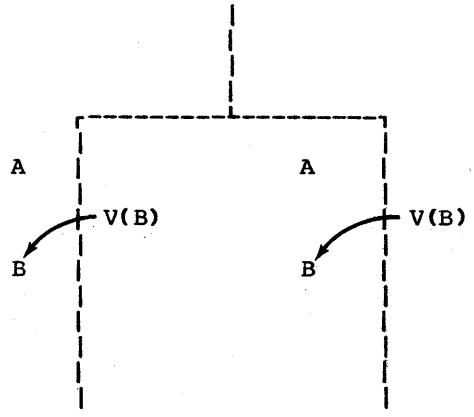
when used as a module in two phases produces



Whereas the coding

```
A   START
      .
B   CSECT
      .
A   CSECT
      .
      .
      .
    DC V(B)
      .
      .
      .
B   CSECT
      .
      .
      .
    END
```

when used as a module in two phases produces



This method of coding redefines the sequence of ESD information to allow valid cross reference by the Linkage Editor.

When linkage editing in the AUTOLINK mode, this is also true except for the case of privileged external references. Privileged external references are, by definition, those external references whose labels begin with the letters IJ. In this case, if the resolution is not possible within the current phase or ROOT phase, the AUTOLINK function is performed on this external reference at the end of the phase. The other previously defined phases are not examined for possible resolution. If NOAUTO is specified, the IJ prefix is not privileged. If the IJ module is not defined in the relocatable library, the address constant referencing this label will not be resolved.

INCLUDE STATEMENT

This statement is used to indicate that an object module is to be included for editing by the Linkage Editor. It has two optional operands. When both operands are used, they must be in the prescribed order. When the first operand is omitted and the second operand is used, a comma must precede the second operand. The first operand indicates that the input is in the relocatable library. The second operand indicates that the input is in submodular structure. The names appearing in the namelist (second operand) are the names of selected control sections from which a phase is to be constructed.

If both operands are omitted, the object module to be included is assumed to be on SYSIPT. Job Control copies it onto SYSLNK.

If the first operand is present, the object module is assumed to be in either the private or the system relocatable library. The Linkage Editor first searches the private relocatable library (if SYSRLB has been assigned) and then the system relocatable library for the module. The module name must be the same as that used when the module was cataloged in the library. Including modules from the relocatable libraries permits the programmer to include standard subroutines in his program at linkage-edit time.

If the first operand is omitted and the second operand is present, the object module to be included is assumed to be in the input stream (SYSLNK). The Linkage Editor reads the object module and extracts the control section(s) indicated by the second operand of the INCLUDE.

Note: If this option is elected, the module must be preceded by an INCLUDE statement with a blank operand in order for Job Control to place the module on SYSLNK.

If both operands are present, the object module is read from the relocatable library and the indicated control section(s) are extracted.

The placement of the INCLUDE statement determines the position of the module in the program phase. An included module (in the relocatable library) can be preceded by one or more additional INCLUDE statements.

The format of the INCLUDE statement is:

Name	Operation	Operand
blank	INCLUDE	[modulename][,(namelist)]

modulename — Symbolic name of the module, as used when cataloged in the relocatable library. It consists of one to eight alphameric characters.

(namelist) — Causes the Linkage Editor to construct a phase from only the control sections specified. The namelist is in the following format.

(csname1,csname2,...)

Entries within the parentheses are the names of the control sections that will be used to constitute a phase. When the namelist option is used and only selected control sections are included in a phase, a submodular phase is created. The counterpart of a submodular phase is a normal phase. A normal phase contains all control sections of one or more object modules. It is possible to include within the same phase an object module(s) without the namelist option and an object module(s) specifying the namelist option. The total number of control sections in a namelist cannot exceed five; however, any number of INCLUDE statements can be used.

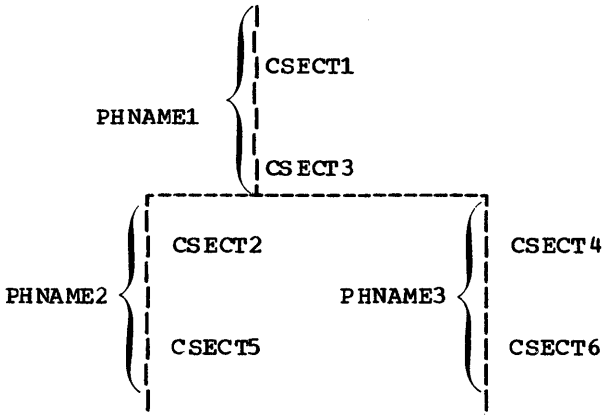
Modules in the relocatable library can be nested by using INCLUDE statements up to a depth of five (level of six). Modules included by INCLUDE statements read from SYSRDR are referred to as being in the first level. Modules included by statements in the first level are at the second level. This is illustrated in Figure 16. Modules included by statements in the second level are at the third level, and so on up to six levels.

Submodular Structure

When several control sections are compiled together in one object module, it is sometimes desirable to break them up into several phases at linkage-edit time. This is done by using a PHASE statement followed by an INCLUDE statement with the namelist option. For example, the sequence

```
PHASE     PHNAME1,*
INCLUDE   ,(CSECT1,CSECT3)
PHASE     PHNAME2,*
INCLUDE   ,(CSECT2,CSECT5)
PHASE     PHNAME3,PHNAME2
INCLUDE   ,(CSECT4,CSECT6)
```

causes the Linkage Editor to structure the next module composed of CSECT1-CSECT6 in three overlays as shown:



The absence of the first operand in the INCLUDE statement indicates that the control sections are to be incorporated from the next succeeding module in the input stream.

The preceding sequence of PHASE and INCLUDE statements may be read by Job Control onto SYSLNK in one of two ways:

- If the PHASE and INCLUDE statements are on SYSRDR, an INCLUDE statement with a blank operand must follow the sequence to read the module (on SYSIPT) containing CSECT1-CSECT6 onto SYSLNK.

- If the PHASE and INCLUDE statements are on SYSIPT (immediately preceding the module), an INCLUDE statement with a blank operand on SYSRDR directs Job Control to read everything onto SYSLNK from SYSIPT down to the /* statement.

PHASE and INCLUDE statements can also be in the relocatable library. This implies that submodular phases can be constructed from modules in the relocatable library. If PHASE and INCLUDE statements come from the relocatable library (via an INCLUDE MODNAME), then the control sections for that module are in the relocatable library. In this structure, the required control sections (in the relocatable library) immediately follow the last INCLUDE statement. For example, the sequence

```
PHASE     PHNAME1,*
INCLUDE   MODNAME1,(CSECT1,CSECT3)
PHASE     PHNAME2,*
INCLUDE   MODNAME1,(CSECT2,CSECT5)
PHASE     PHNAME3,PHNAME2
INCLUDE   MODNAME1,(CSECT4,CSECT6)
```

causes the Linkage Editor to structure the next module (cataloged in the relocatable library under MODNAME1) composed of CSECT1-CSECT6 into the same three overlays as shown in the preceding example.

If MODNAME1 contains an INCLUDE statement, the Linkage Editor interprets this to mean that the module to be included should also be searched for the control sections requested in the namelist. For example, in the relocatable library if MODNAME1 contains

```
INCLUDE  MODNAME2
CSECT3
CSECT5
CSECT6
```

and in the relocatable library MODNAME2 contains

```
CSECT1
CSECT2
CSECT4
```

upon encountering an

```
INCLUDE  MODNAME1,(CSECT1,CSECT3)
```

statement, the Linkage Editor goes to MODNAME1 and finds INCLUDE MODNAME2. Linkage Editor then goes to MODNAME2 and extracts CSECT1 and returns to MODNAME1 and extracts CSECT3.

A nonsubmodular INCLUDE statement may be placed before or after a submodular INCLUDE statement. This results in the addition of the included module into the phase at the point the INCLUDE statement is encountered. For example, if MOD1 contains CSECT4 and CSECT5, the sequence

```
PHASE     PHNAME1,*
INCLUDE   ,(CSECT1,CSECT3)
INCLUDE   MOD1
```

results in the following structure:

```
PHNAME1   CSECT1
          CSECT3
          CSECT4
          CSECT5
```

while the sequence

```
PHASE     PHNAME1,*
INCLUDE   MOD1
INCLUDE   ,(CSECT1,CSECT3)
```

results in the following structure:

```
PHNAME1   CSECT4
          CSECT5
          CSECT1
          CSECT3
```

Note: Both of the following statements produce the same result.

```
INCLUDE  ,(CSECT1,CSECT3)

INCLUDE  ,(CSECT3,CSECT1)
```

That is, CSECT1 and CSECT3 are in storage in that sequence. This is because the Linkage Editor extracts control sections in the order in which they appear in the input stream, not as they are ordered in the namelist. In order to have CSECT3 physically located ahead of CSECT1 in storage, two INCLUDEs must be used:

```
INCLUDE  ,(CSECT3)

INCLUDE  ,(CSECT1)
```

As no diagnostic is given if a control section, specified in the namelist, is not present in the indicated module, the user can inspect the MAP supplied by the Linkage Editor to determine if the proper control sections are in the correct phases.

ENTRY STATEMENT

Every program, as input for the Linkage Editor, is terminated by an ENTRY statement. Its format is:

Name	Operation	Operand
blank	ENTRY	[entrypoint]

entrypoint — Symbolic name of an entry point. It must be the name of a CSECT or a label definition (source ENTRY) defined in the first phase. This address is used as the transfer address to the first phase in the program. If the operand field is blank, the Linkage Editor will use as a transfer address the first significant address provided in an END record encountered during the generation of the first phase. If no such operand is found on the END card, the transfer address will be the load address of the first phase.

It is necessary to supply the ENTRY statement only if the user wishes to provide a specific entry point. Job Control writes an ENTRY statement with a blank operand on SYSLNK when EXEC LNKEDT is read to ensure that an ENTRY statement will be present to halt linkage editing.

ACTION STATEMENT

This statement is used to indicate Linkage Editor options. When used, the statement must be the first Linkage Editor record(s) in the input stream. Its format is:

Name	Operation	Operand
blank	ACTION	CLEAR MAP NOMAP NOAUTO CANCEL F1 F2

CLEAR — Indicates that the unused portion of the core image library will be set to binary zero before the beginning of the Linkage Editor function. CLEAR is a time consuming function. It should be used only if it is necessary to fill areas defined by DS statements with zeros.

MAP — Indicates that SYSLST is available for diagnostic messages. In addition, a main storage map is output on SYSLST. The map contains every entry within each CSECT and every CSECT within each phase.

NOMAP — Indicates that SYSLST is not available when performing the linkage-edit function. Mapping of main storage is not performed and all Linkage Editor error diagnostics are listed on the printer-keyboard (SYSLOG).

NOAUTO — Indicates the AUTOLINK function is to be suppressed during the linkage editing of the entire program. AUTOLINK will be suppressed for both the private and the system relocatable libraries.

> Note: When a WX is encountered, it is treated in the same manner as an EXTRN, NOAUTO.

CANCEL — Causes an automatic cancellation of the job if any of the errors 2100I through 2170I occur. If this option is not specified, the job continues.

F1
F2 — Causes the end-of-Supervisor address used in Linkage Editor calculations to be set to the beginning of the partition specified, plus the length of the label area and of the save area. The end of Supervisor address in the communication region is not changed.

Use of the F1 and F2 operands facilitates linkage editing of programs to be run in the foreground areas. Programs that have a phase origin of S (or *, for the first phase of a program) can be origined to the proper foreground area by the use of these operands. One of these operands is required when blank COMMON is used in a foreground partition or when the ROOT specification is used as the phase origin for a program to be run in the foreground areas (see the section PHASE Statement).

Use of the ACTION F1 (or F2) statement in a multiprogramming environment requires that the partition be allocated. If these operands are used in a non-multiprogramming environment, they are ignored.

An example of the use of the ACTION F1 statement follows

Assume a 64K machine with:

```
 8K Supervisor
24K Background area
16K Foreground 2 area
16K Foreground 1 area.
```

The statement PHASE PHASE1,S causes PHASE1 to be origined at 8K (the end of the Supervisor area).

The sequence

```
ACTION F1
PHASE PHASE1,S
```

causes PHASE1 to be origined at 48K (the beginning of the Foreground 1 area) plus the length of the foreground save area.

An ACTION statement flagged as invalid (as the result of an invalid operand, etc) causes all subsequent ACTION statements submitted during the job to be ignored.

The ACTION statement is not required. If the MAP option is specified, SYSLST must be assigned. If the statement is not used and SYSLST is assigned, MAP is assumed and a map of main storage and any error diagnostics are output on SYSLST. If the statement is not used and SYSLST is not assigned, NOMAP is assumed.

The following information is contained in the map of main storage.

1. The name of each phase, the lowest and highest main storage locations of each phase, and the hexadecimal disk address where the phase begins in the core image library.
2. An indication if the phase is a ROOT phase, or if a phase overlays the ROOT phase in any way (designated by OVEROOT).
3. The length of COMMON, if appropriate.
4. The names of all CSECTs belonging to a phase, the address where each CSECT is loaded, and the relocation factor of each CSECT.
5. All defined entry points within a CSECT. If an entry point is unreferenced, it is flagged with an asterisk (*).
6. The names of all external references that are unresolved.
7. The transfer (execute) address of each phase.
8. Warning messages are printed if: the ROOT phase has been overlaid; a possible invalid entry point duplication occurred; the ENTRY or END statement contained an invalid (undefined) transfer label; at least one control section had a length of zero; the assembled origin on an RLD statement was outside the limits of the phase; or, an address constant could not be resolved. These messages may or may not indicate actual programming errors. The user will not be aware of these warnings if NOMAP is operational.

The difference between specifying NOAUTO in a PHASE statement and specifying ACTION NOAUTO. The NOAUTO operand in a PHASE statement indicates to the Linkage Editor that AUTOLINK is to be suppressed for that phase only. If an entire program requires NOAUTO, then specifying ACTION NOAUTO cancels AUTOLINK during the linkage editing of the entire program, thereby eliminating the necessity of specifying NOAUTO in each PHASE statement.

Figure 18 shows a map of main storage and a diagnostic listing produced on SYSLST as a result of SYSLST being assigned and ACTION NOMAP not specified.

For the line numbers referred to in the following discussion, see the "Disk Linkage Editor Diagnostic of Input" portion of Figure 18.

1. Line 1 (ACTION TAKEN). MAP and CLEAR have been specified on separate ACTION cards. Had NOAUTO been specified, it would also appear on this line.
2. Lines 4, 7, 10, and 13. Error 2141I (duplicated ESID number) is printed four times because the submodular structure of the phase demanded four passes over the same module. As the Linkage Editor processes in its own input area, the record printed may not have identical information to the original input record. Lines 10 and 13 differ in content from lines 4 and 7 for this reason.
3. Lines 11 and 12. Line 11 is printed when the statement is read by the Linkage Editor. Line 12, error 2131I indicating that the requested module is not in the relocatable library, is printed after the error is detected.
4. Line 16. This is an example of an error detected in a TXT statement. Error 2144I indicates the ESID number F0F1 is invalid. (It should be binary 01.)

5. Line 17. Indicates the AUTOLINK feature was used for relocatable library module inclusion in the phase named above it.

6. Line 19. An example of a valid REP statement.

7. Lines 20 and 21. An example of an invalid REP statement. Line 20 is printed when the statement is read by the Linkage Editor. Line 21, error 2102I indicating an invalid operand in the statement, is printed after the error is detected.

When a module is included from the relocatable library, it is not possible to guarantee that the sequence identification printed in columns 8-15 is that of the record printed. This occurs because the MAINT librarian program reblocks the content of the cards to a more compressed format.

For the line numbers referred to in the following discussion, see the "Map" portion of Figure 18.

1. Line 2 (COMMON). The entry under REL-FR contains the length instead of the relocation factor in the case of ESD-type COMMON.

2. Lines 5 and 9 (referring to UNREFERENCED SYMBOLS). These ENTRY labels (POINT2 and POINT3) are not referenced as an external symbol, that is, by no corresponding EXTRN statement.

3. Lines 17 and 18. These labels indicate EXTRN references that cannot be matched with a corresponding entry point. In such a case, * ENTRY ESD-types may be the corresponding, but misspelled, point. In the submodular structure, CSECTs not specified in any namelist appear as EXTRNs. The labels can also indicate unreferenced EXTRN's.

4. Lines 3, 6, 7, 11, and 15. All phase origins (entries under LOCORE) are incremented by the length of COMMON.

5. Line 19. Warning message. When this message appears, OVEROOT is printed to the left of the name of the phase (PHASE3) that overlays the ROOT phase.

6. Line 20. Warning message. An entry label appeared at least twice in the input stream. At the time of the second (or more) occurrence, it was not possible to validate it as being a true duplication of the previous occurrence. The most common reason for this message is in submodular structure with (source) ENTRY labels defined before the CSECT in which the entry point appears.

7. Line 21. An overriding transfer label in the ENTRY statement was not defined within the first phase, or a transfer label was not defined in an END statement in its module.

8. Line 22. Warning message. The COBOL, FORTRAN, RPG, and PL/I (D) compilers do not supply all of the information required by the Linkage Editor in the ESD records. Specifically, the control section length is provided in the END record. If a control section defined in the ESD information has a length of zero, it normally indicates the length will appear in the END record. It is possible for the user to generate zero-length control sections through the Assembler. Such a condition produces this message. This is not an invalid condition if it is not the last control section that is of zero length. If the last control section is of zero length, the length is implied to be in the END record and is an error condition if not present.

9. Line 23. These address constants correspond to the EXTRNs shown in lines 17 and 18.

10. Line 24. Address constants had load addresses outside the limits of the phase in which they occurred. This normally occurs if the control section length is incorrectly defined in the input.

```
JOB  EXAMPLE              DISK LINKAGE EDITOR DIAGNOSTIC OF INPUT

ACTION TAKEN    MAP  CLEAR
LIST           PHASE PHASE1,ROOT,NOAUTO
LIST         INCLUDE  ,(NAMEONE)

21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  2 FF0130 0000CA NAMTHREE 0 000200 0000A8

LIST          PHASE   PHASE2,*,NOAUTO
LIST            INCLUDE     ,(NAMEFOUR)

21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  2 FF0130 0000CA NAMTHREE 0 000200 0000A8

LIST           PHASE PHASE3,PHASE1+73,NOAUTO
LIST           INCLUDE  ,(NAMETWO,NAMTHREE)

21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000007 NAMETWO  0 000130 001898 NAMTHREE 0 000200 0000A8

LIST        INCLUDE  RELMOD

21311       INCLUDE  RELMOD

21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  0 000130 001928 NAMTHREE 0 000200 0000A8

LIST          PHASE   PHASE4,+16500
LIST     INCLUDE   RELMODUL

21441  REL 0015 TXT 00425C 0038 F0F1 1A361A56 46D0E254 4130EF1E D500EF1E E5FA477C E2869201 EF1E0630 9509EF1D 4770E286

LIST    AUTOLINK   AUTOMOD2
LIST           PHASE PHASE5,+X'25BA',NOAUTO
LIST    REP  0C4018 0034130,C03A,47F0,C30E       PATCH ASSEMBLY ERRORS
LIST    REP  0C40CC 003D20E,

21021          REP 0040C0 C3F0 0003 D2CEF0C5 68404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040

LIST    ENTRY             INVALID TRANSFER LABEL
```

```
           PHASE  XFR-AD  LOCORE  HICORE  DSK-AD   ESD TYPE  LABEL     LOADED  REL-FR

COMMON                                              COM                001800  0000C8

ROOT    PHASE1    0018C8  0018C8  0019F7  13 2 2   CSECT     NAMEONE   0018C8  0C18C8
                                                      ENTRY   POINT1    0018CC
                                                    * ENTRY   POINT2    001930

        PHASE2    0019F8  0019F8  001A87  13 3 1   CSECT     NAMEFOUR  0019F8  0C1750

OVERCOT PHASE3    0019E8  001918  001B1F  13 3 2   CSECT     NAMETWO   001918  0017E8

                                                    CSECT     NAMTHREE  0019E8  0C17E8
                                                    * ENTRY   POINT3    001A2C

                                                    CSECT     NAMEFOUR  001A90  0C17E8

        PHASE4    0043A0  00414C  0059A3  13 4 1   CSECT     AUTOMOD1  004140  003A98
                                                      ENTRY   AUTOENT   0042D0

                                                    CSECT               0043A8  0C3EF8

                                                    CSECT     AUTOMOD2  0043C0  0003C0

        PHASE5    002688  002688  002767  13 6 1   CSECT               0024F8 -0C1B08

                                                    CSECT     NAME5     0024F8 -0C1B08

* UNREFERENCED SYMBOLS                              EXTRN     PONT2
                                                    EXTRN     POINT4

ROOT STRUCTURE OVERLAID BY SUCCEEDING PHASE

POSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT

INVALID TRANSFER LABEL ON END OR ENTRY STATEMENT IGNORED

CONTROL SECTIONS OF ZERO LENGTH IN INPUT

002 UNRESOLVED ADDRESS CONSTANTS

003 ADDRESS CONSTANTS OUTSIDE LIMITS OF PHASE
```

Figure 18. Map of Main Storage

PHASE ENTRY POINT

The Linkage Editor stores with each phase the absolute entry point of that phase. These entry points are as follows.

1. For the first phase, the entry point specified in the ENTRY statement is used. If no entry point is specified, the first significant entry address taken from an END statement encountered in the construction of the first phase is used.

2. For submodular phases, the entry point specified in the END record is used if the CSECT that contains this entry point was specified in the namelist. If an external label is the entry point defined by the END record, it must be previously defined and will be used for all phases constructed from this module. If no entry point is specified, the beginning address of each phase is used.

3. For all phases, the entry address (if any) specified in the first END record encountered in the construction of the phase is used.

SELF-RELOCATING PROGRAMS

A system with multiprogramming has the capability of executing self-relocating programs. A self-relocating program is one that can be executed at any location in main storage. A program that is self-relocating must initialize its address constants, including Channel Command Words (CCWs), at execution time.

The IBM-supplied logical IOCS access methods are self-relocating. All self-relocating programs that use logical IOCS must use the OPENR macro to obtain DTF table address relocation. The CLOSER macro must be used when previously opened files are being deactivated.

Although self-relocating programs are somewhat more difficult to program, the advantages may compensate for the extra programming effort, particularly in a system having two foreground areas. Some advantages are:

1. The operator need not be aware of the origin address established by the Linkage Editor to partition main storage correctly.
2. The program may be executed in any of the three areas.
3. Three copies of the same program may be executed simultaneously.

Self-relocating programs must be assigned an origin of location zero when they are linkage edited. The program initiator recognizes that a program being loaded is self-relocating by virtue of the zero address. The program is entered for execution at the entry point specified at linkage edit time by relocating this entry point by the address at which the program is loaded. For example, a program to be loaded at X'4000' with an entry point of X'50' is entered at storage location X'4050'.

> Note: A COMMON area cannot be used in a self-relocating program.

Because self-relocating programs must be assigned to origin at location zero when linkage edited, multiphase programs should use the LOAD macro instruction rather than the FETCH macro instruction, with the register specifying the load address. Control is given to the loaded phase by branching to the contents of register 1 (ENTRY point). The user should code his loading mechanism in a place where it is not overlaid by a new phase.

> Note: A self-relocating program with a zero address linkage edit origin cannot be initiated by the EXEC statement in a batched-job foreground (no MPS) system.

FGP PROGRAMS

For more rapid retrieval of multiphase or frequently used foreground programs, each phase can be given the prefix FGP as the first three characters of the program name. Phases with this prefix are cataloged into a separate FGP subdirectory on the resident pack. (The capacity of the directory is 144 phases.) When one of these phases is fetched, this subdirectory is searched before the core image directory.

The link-edit-and-go mode of operation cannot be used if the FGP program is already cataloged on the system because the FGP subdirectory is searched before the core image directory. A new phase, therefore, cannot be accessed until the catalog function is completed because the FGP subdirectory is not updated until end-of-job is reached.

LINKAGE EDITOR INPUT RESTRICTIONS

In a background area of 10K, the Linkage Editor can process at least 30 phases in a program, 221 unique ESD items in a program, and 27 ESID items in a module.

A unique ESD item is defined as being an occurrence in the control dictionary. All symbols that appear in the MAP are unique occurrences. A symbol that occurs several times in the input stream is normally incorporated into a unique ESD item. However, if the same symbol occurs in different phases (e.g., control sections), each resolved occurrence of the symbol within a different phase is a unique ESD item.

The number of ESID items is the number of ESD items within a module not counting the LD (source ENTRY) type.

An RLD item is related to the address constants within a phase. An address constant can have one or more RLD items, each item corresponding to a symbol in the relocatable expression of an address constant. For example, A(X + Y) is an address constant of two relocatable expressions that will have two RLD items.

A combination of the preceding values can be accommodated by the Linkage Editor in accordance with the following formulas.

This formula can be used for determining the maximum number of PHASE, ESD, and ESID items that can be processed in the control dictionary of a 16K system.

$$16(x + y) + 3z \leq 4000$$

where x = total number of PHASE statements, not to exceed 120
y = total number of unique ESD items
z = total number of ESID items per module.

A background area greater than 10K but less than 14K does not change this formula. However, if the background program area is greater than 14K, the Linkage Editor uses additional storage to increase the dictionary area. The value 4000 in the formula for a background area greater than 14K increases linearly to a value of 33,500 at a background area of 40K.

In a background area greater than 10K, the additional main storage is used to produce faster linkage-edit times and accommodate a greater number of combinations of the preceding values in accordance with the above formulas.

The maximum number of phases that can be processed by the Linkage Editor at any one time is 120. The maximum number of bytes per phase is 440,640.

LINKAGE EDITOR JOB SETUP

When performing a linkage edit function, the following system and programmer logical units are used. SYSRDR and SYSIPT may contain input for the Linkage Editor. This input is written onto SYSLNK by Job Control.

Unit	Function
SYSRDR	Control statement input (via Job Control)
SYSIPT	Module input
SYSLST	Programmer messages and listings
SYSLOG	Operator messages
SYSLNK	Input to the Linkage Editor
SYS001	Workfile

In normal operations, all preceding logical units must be assigned. In a unique circumstance (when all modules to be linkage edited are in the relocatable library), SYSIPT would not need to be assigned.

A linkage edit job is set up in the following manner.

Control Statement	Remarks
// JOB	Required only if this is the first job step of a job.
// ASSGN	Required only if device assignments are to differ from the system standard assignments. Units that can be assigned are SYSRDR, SYSIPT, SYSLST, SYSLNK, and SYS001.
// OPTION	OPTION statement must follow the ASSGN statement (if any) for SYSLNK.
ACTION	Optional ACTION statement (with appropriate operand) must precede the first Linkage Editor control statement.
PHASE INCLUDE	As many PHASE and INCLUDE statements as are required are used to construct phases from the modules input to the Linkage Editor.

ENTRY	Optional statement to provide a transfer address for the first phase.
// LBLTYP	LBLTYP statement (if required) to define the amount of main storage to be reserved at linkage-edit time for processing of tape or nonsequential DASD file labels in the problem program area of main storage.
// EXEC LNKEDT	EXEC statement to call the Linkage Editor from the core image library. Job Control creates an ENTRY statement on SYSLNK to ensure its presence to halt linkage editing.
/&	End-of-job statement.

When linkage editing multiple object modules into one program phase, make sure that the linkage editor selects the intended entry point. Either specify or place the main control section first in the linkage editor input, or use a linkage editor ENTRY statement with the name of the main control section as the entry-point operand.

Example of Linkage Editor Input and Output

The program shown in Figure 19 illustrates the rules governing input for the Linkage Editor and shows the output obtained. Though this example is somewhat more complex than the normal program, by following the flow of the input, one can find practically every situation that may arise.

The leftmost block shows control statements being read by Job Control from SYSRDR. The next block is read by Job Control from SYSIPT and contains an object module (module 1) and a source module to be assembled. The next block shows the output from Job Control on SYSLNK, which is the input to the Linkage Editor. The next two blocks represent two levels in the relocatable library. The rightmost block shows the output phases as they appear in the temporary portion of the core image library after the execution of the Linkage Editor function. A detailed sequence of events follows.

Linkage Editor control statements are read by Job Control from SYSRDR and are copied on SYSLNK until an INCLUDE statement with a blank first operand is read. This statement is not copied on SYSLNK. Instead, Job Control copies the module on SYSIPT onto SYSLNK until a /* statement is read. Job Control then reads from SYSRDR. An assembly is executed and its output is written directly on SYSLNK. (It is assumed that LINK was specified in an OPTION statement preceding the Linkage Editor control statements.) Job Control then writes the ENTRY statement with a transfer label for CS A on SYSLNK and issues a fetch for the Linkage Editor.

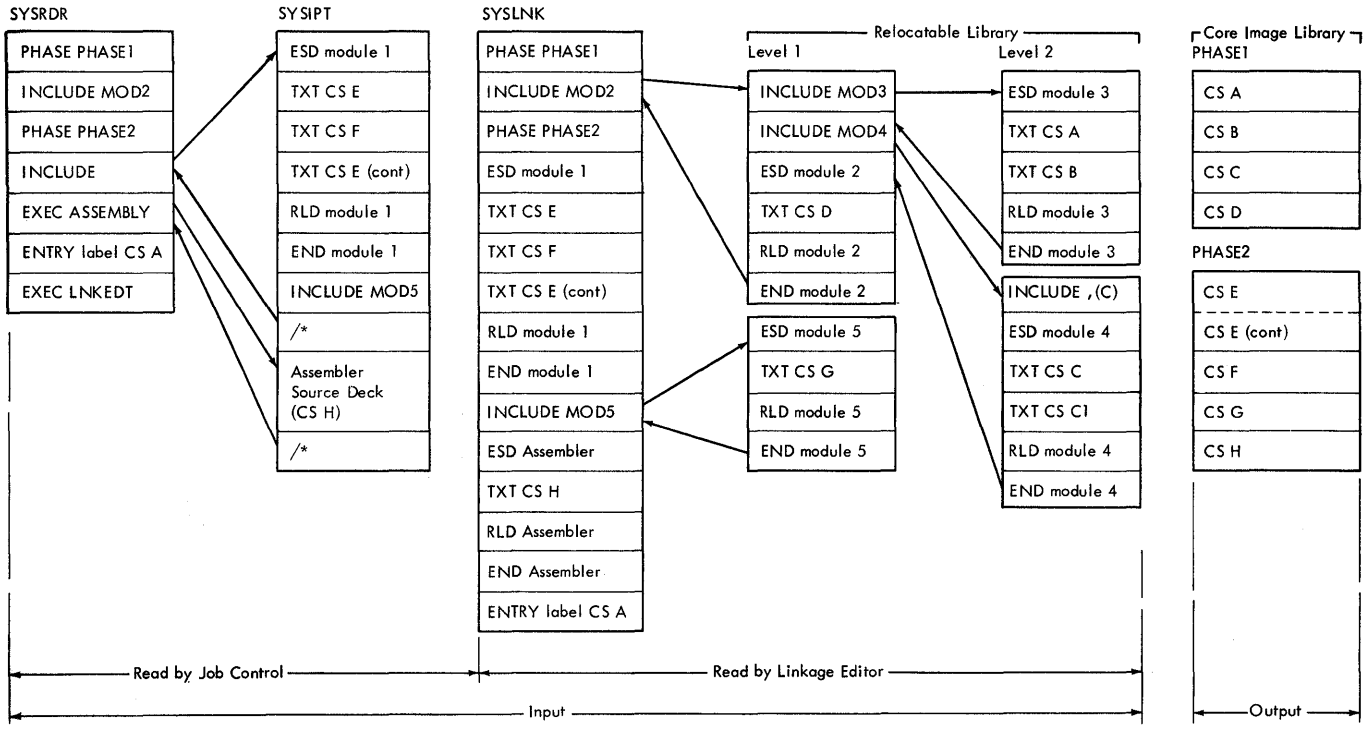


Figure 19. Example of Linkage Editor Input and Output

The Linkage Editor reads from SYSLNK and starts to create a program. An INCLUDE statement with a nonblank first entry in the operand field signals the Linkage Editor to access the relocatable library. This is the first level of an INCLUDE. In the first level of the relocatable library, the Linkage Editor reads an INCLUDE (for the second level) and performs this inclusion. As no INCLUDE is present in the second level, control is returned to the calling input level. This process is repeated for the next INCLUDE. Note that the namelist specifies only CS C is wanted.

After the inclusion of the module at the first level, control is returned to SYSLNK where a new phase is encountered. The control sections are read from SYSLNK and added to PHASE2 until the next INCLUDE is read. At this time, the Linkage Editor again accesses the relocatable library, performs the inclusion of MOD5 into PHASE2, and continues reading input from SYSLNK. Processing continues until the ENTRY statement is reached.

The split control section (CS E) is assigned a contiguous area of main storage.

Librarian

This section describes the set of programs that maintain, service, and copy the libraries of the Disk Operating System. This set of programs is collectively referred to as the Librarian.

The system residence (SYSRES) can contain three separate and distinct libraries:

1. Core image library
2. Relocatable library
3. Source statement library.

The core image library is required for each disk-resident system. The other two libraries, the relocatable library and the source statement library, are not required for operating a system.

CORE IMAGE LIBRARY

The core image library contains any number of programs. Each program is made up of one or more separate phases. Hence, each phase may either be a single program, or an overlay of a multiphase program. The size and number of programs in the core image library are the factors that determine the amount of space that need be allocated to the library.

All programs in the core image library are edited to run with the resident Supervisor. Each program phase is assigned a fixed location in main storage. The programs in the core image library include system programs, other IBM programs such as Assembler, RPG, COBOL, FORTRAN, PL/I (D), and sort programs, and user programs.

Associated with the core image library is a core image directory. The directory contains a unique descriptive entry for each phase in the core image library. Each entry includes the name of the phase, the starting disk address of the phase in the core image library, the number of records necessary to contain the phase, the number of bytes in the last record, the starting address in main storage where the phase will be loaded, and its entry point. The entries in the core image directory are used to locate and retrieve phases from the core image library.

Phases in the core image library and entries in the core image directory are in the order in which they were cataloged.

RELOCATABLE LIBRARY

The relocatable library contains any number of modules. Each module is a complete object deck in relocatable format. The size and number of modules in the relocatable library are the factors that determine the amount of space that need be allocated to the library.

The purpose of the relocatable library is to allow the user to maintain frequently used routines in residence and combine them with other modules without requiring recompilation. The routines from the relocatable library are edited in the core image library by the Linkage Editor.

Associated with the relocatable library is a relocatable directory. The directory contains a unique descriptive entry for each module in the relocatable library. Each entry includes the name of the module, the starting disk address of the module in the relocatable library, the number of records required to contain the module, the change level of the module, and the number of bytes in the last record. The entries in the relocatable directory are used to locate and retrieve modules in the relocatable library.

Modules in the relocatable library and entries in the relocatable directory are in the order in which they were cataloged.

Private relocatable libraries and system relocatable libraries are supported. (See the section Private Libraries.)

SOURCE STATEMENT LIBRARY

The source statement library contains any number of books. Each book in the source statement library is made up of a sequence of source language statements. The size and number of books in the source statement library are the factors that determine the amount of space that need be allocated to the library.

The purpose of the source statement library is to provide an extension of the functions of a macro library. If a source program contains a macro instruction, the macro definition in the source statement library corresponding to the macro instruction is generated in the source program. If the source program contains a COPY statement, the specific language translator will compile a book from the source statement library into the source program.

Each book in the source statement library is classified as belonging to a specific sublibrary. Sublibraries are currently defined for two programming languages (Assembler and COBOL) in the system. Classifying books by a sublibrary prefix allows a program written in COBOL to have the same name as a program written in Assembler.

Card images are stored in compressed form within the library. In the compressed format, all blanks are eliminated. When a book is retrieved, the card images are expanded to their original 80-character format.

Associated with the source statement library is a source statement directory. The directory contains a unique descriptive entry for each book in the source statement library. Each entry includes the name of the book, the starting disk address of the book in the source statement library, the number of records required to contain the book, the change level of the book, and an indication of the requirement for change level verification before updates. The entries in the source statement directory are used to locate and retrieve books in the source statement library.

Books in the source statement library and entries in the source statement directory are in the order in which they were cataloged.

Private source statement libraries and system source statement libraries are supported. (See the section Private Libraries.)

DISK STORAGE SPACE REQUIRED FOR LIBRARIES AND DIRECTORIES

The relative location of each of the library and directory areas is fixed. The amount of space allocated to each is determined by the user. Each library area consists of one or more complete disk cylinders. Each directory area consists of one or more complete disk tracks. Each directory occupies the first track(s) of the first cylinder allocated to its respective library.

On the 2311, the core image directory starts on track 0 of cylinder 1 because the system material requires 10 full tracks of cylinder 0. The core image library starts on the track following the last track of the core image directory and fills the number of cylinders specified.

On the 2314, the core image directory starts on track 10 of cylinder 0 because the system material takes up the first 10 tracks of cylinder 0. The core image library starts on the track following the last track of the core image directory, completing cylinder 0 if any available tracks remain, and then fills the number of cylinders specified for the library.

The difference between the 2311 and the 2314 is that on the 2314, the last 10 tracks of cylinder 0 are added to the core image library. If 3 cylinders are specified for the core image library on the 2314, the library will be on the last 10 tracks of cylinder 0 and the following 3 complete cylinders.

Beginning with the core image directory and library, the sequence of areas is:

1. Core image directory and library (required)
2. Relocatable directory and library (optional)
3. Source statement directory and library (optional).

If the relocatable library is not used, the source statement library immediately follows the core image library. If neither the relocatable library nor the source statement library is used, the label control card area (volume information area) immediately follows the core image library.

In each of the formulas contained in this section, the formula on the left applies to the 2311. The formula on the right applies to the 2314. If the formula applies equally to the 2311 and the 2314, it is given only once.

Core Image Directory Size

Each track allocated to the core image directory can contain entries for 144 phases on the 2311 (or 270 phases on the 2314) with the exception of the last track, which can contain only 143 (or 269) entries. Thus, the number of tracks (T) required for the core image directory is:

$$T_D = \frac{P+1}{144} \qquad T_D = \frac{P+1}{270}$$

where P = total number of phases in the core image library. The value of T_D is rounded to the next higher integer if a remainder results.

Core Image Library Size

Each track allocated to the core image library contains two fixed-length blocks on the 2311 (or 4 fixed-length blocks on the 2314). Each block contains a maximum of 1728 (or 1688) bytes of instructions or data. The core image library contains exactly the same information as is loaded into main storage for execution. Each phase is written beginning in a new block. The number of tracks required for the core image library can be calculated as follows.

1. Determine the number of blocks (B_n) required for a phase:

   $$B_n = \frac{L}{1728} \qquad B_n = \frac{L}{1688}$$

   where L = total number of bytes in the phase. The value B_n is rounded to the next higher integer.

2. Determine the total number of blocks (B_t) required for all phases in the core image library:

   $$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$

3. Determine the number of tracks (T_L) required to hold all phases in the core image library:

   $$T_L = \frac{B_t}{2} \qquad T_L = \frac{B_t}{4}$$

4. Determine the number of cylinders (C) required to hold the core image library and core image directory:

   $$C = \frac{T_D+T_L}{10} \qquad C = \frac{T_D+T_L}{20}$$

   The value C is rounded to the next higher integer if a remainder results.

Relocatable Directory Size

Each track allocated to the relocatable directory can contain entries for 180 modules on the 2311 (or 340 modules on the 2314) with the exception of the first track, which can contain only 175 (or 335) entries, and the last track, which can contain only 179 (or 339) entries. Thus the number of tracks (T_D) required for the relocatable directory is:

$$T_D = \frac{m+6}{180} \qquad T_D = \frac{m+6}{340}$$

where m = total number of modules in the relocatable library. The value T_D is rounded to the next higher integer if a remainder results.

Relocatable Library Size

Each track allocated to the relocatable library contains 9 fixed-length blocks for the 2311 (or 16 fixed length blocks for the 2314). Each block is 322 bytes long. A number of factors affect the packing of information in these blocks. The factors include the following variables:

1. The number of separate control sections.
2. The use of DS (define storage) statements, which reserve storage that may or may not be utilized for data constants defined in the program.
3. Alteration of the location counter during assembly (use of ORG statements).

The following calculations approximate fairly accurately the library area required for typical programs.

1. Determine the number of blocks (B_c) required for all cards or statements except the actual program text. Assume a separate block for each card of the following types:

   a. PHASE

   b. INCLUDE

   c. REP

   d. END

   e. SYM

   f. ENTRY

   Let B_c = total number of cards of the above types.

2. Determine the number of blocks (B_e) required for ESD and RLD cards. Assume a separate block for every two ESD or RLD cards.

3. Determine the number of blocks (B_i) required for the actual instructions or data in the TXT cards. Assume an average of 200 bytes of text in each block. (A maximum per block, for contiguously assigned text, is 264 bytes per block.) Thus,

$$B_i = \frac{\text{total bytes of text in TXT cards}}{200}$$

4. Determine the total number of blocks (B_n) required for a module in the relocatable library:

$$B_n = B_c + B_i + B_e$$

5. Determine the total number of blocks (B_t) required to hold all of the modules in the library:

$$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$

6. Determine the number of tracks (T_L) required for the relocatable library:

$$T_L = \frac{Bt}{9} \qquad T_L = \frac{Bt}{16}$$

The value T_L is rounded to the next higher integer if a remainder results.

7. Determine the number of cylinders (C) required to hold the relocatable library and relocatable directory:

$$C = \frac{T_D+T_L}{10} \qquad C = \frac{T_D+T_L}{20}$$

The value C is rounded to the next higher integer if a remainder results.

Source Statement Directory Size

Each track allocated to the source statement directory can contain entries for 160 books on the 2311 (or 270 books on the 2314) with the exception of the first track which can contain only 155 (or 265) entries, and the last track, which can contain only 159 (or 269) entries. The number of tracks (T_D) required for the source statement directory will be:

$$T_D = \frac{B+6}{160} \qquad T_L = \frac{B+6}{270}$$

where B = total number of books in the source statement library. The value T_D is rounded to the next higher integer if a remainder results.

Source Statement Library Size

Each track allocated to the source statement library contains sixteen fixed-length blocks for the 2311 (or 27 fixed-length blocks for the 2314). Each block contains a maximum of 160 bytes of source statement information. The source statements coded by the user are compressed before writing them out in the source statement library. This compression is performed by eliminating all blanks in each source statement. Several count bytes indicating the number of blanks eliminated are added to each statement before writing it in the source statement library. The number of tracks required for the source statement library can be calculated as follows.

1. Determine the number of statements (N) used to define a book.

2. Determine the average compressed statement length (L_s) in the book. The compressed statement length approximately equals:

$$L_s = (L_1 + 1) + (L_2 + 1) + \ldots + (L_n + 1) + 3$$

where each L_n = number of bytes in each word of the source statement.

3. Determine the number of blocks (B_n) needed to hold the book:

$$B_n = \frac{N(L_s)}{160}$$

The value B_n is rounded to the next higher integer if a remainder results.

4. Determine the total number of blocks (B_t) required to hold all of the books in the library:

$$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$

5. Determine the number of tracks (T_L) required to hold all of the books in the source statement library:

$$T_L = \frac{Bt}{16} \qquad T_L = \frac{Bt}{27}$$

The value T_L is rounded to the next higher integer if a remainder results.

6. Determine the number of cylinders (C) required to hold the source statement library and source statement directory:

$$C = \frac{T_D+T_L}{10} \qquad C = \frac{T_D+T_L}{20}$$

The value C is rounded to the next higher integer if a remainder results.

Librarian Functions

The Librarian programs perform three major functions:

1. Maintenance
2. Service
3. Copy.

Maintenance functions add, delete, or rename components of the three libraries, condense directories and libraries, and reallocate directory and library extents. The MAINT program is the maintenance program for all three libraries of the system.

Service functions translate information from a particular library to printed (displayed) or punched output. Information in a library directory can also be displayed. The CSERV program is the service program for the core image library. The SSERV program is the service program for the source statement library. The RSERV program is the service program for the relocatable library. The DSERV program is the service program for the directories.

If private libraries are used, the maintenance and service functions apply only to the assigned private library.

The copy function is used to either completely or selectively:

1. Copy the disk on which the system resides.
2. Create private libraries.
3. Merge phases, modules, or books from one core image, relocatable, or source statement library to another core image, relocatable, or source statement library respectively (system or private).

The CORGZ program is the program for the copy function.

Librarian functions are performed through use of control statements. The control statements are:

1. A JOB control statement.
2. A number of ASSGN control statements that may be required to change the assignment of actual input/output devices.
3. An EXEC control statement requesting a particular librarian program.
4. Librarian specification statements describing various functions to be performed.
5. A /* control statement.
6. A /& control statement.

The JOB, ASSGN, /*, and /& control statements are the same as those described in the Job Control section. The operand field of the EXEC control statement is described in this section. The other statements pertain to the Librarian and are described in this section.

Librarian functions can be performed separately, or in certain combinations as described in the following sections. All Job Control statement information is read from the device assigned (in the ASSGN statements) to SYSRDR. Librarian control statement information and input data is read from the device assigned to SYSLNK or SYSIPT. For the librarian functions, SYSIPT and SYSRDR are usually assigned to the same physical input/output device, SYSIN.

Note: If SYSIPT is assigned to disk, the device type must be the same as SYSRES.

Figure 20 is a table of all maintenance functions. Figure 21 is a table of all service functions. Figure 22 is a table of the copy function. In all these figures, input is shown on SYSIN.

Function	Unit	Element	Control Statements Required

Catalog	Core Image Library	Phase	// JOB jobname
			// OPTION CATAL
			(Linkage Editor control statements)
			// EXEC LNKEDT
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			CATALR modulename[,v.m]
			(module to be cataloged)
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			CATALS sublib.bookname[,v.m[,C]]
			(book to be cataloged)
			/*
			/&

Compile/ Assemble and Catalog	Relocatable Library	Module	Using a tape file for SYSPCH/SYSIPT with temporary assignments
			
			// JOB jobname
			// ASSGN SYSPCH,X'cuu'
			CATALR modulename[,v.m]
			PHASE name,origin[,NOAUTO]
			// EXEC COBOL
			
			source deck
			
			/*
			// CLOSE SYSPCH,X'cuu'
			// PAUSE -- RELOAD TAPE SYSPCH WAS ASSIGNED TO
			// ASSGN SYSIPT,X'cuu'
			// EXEC MAINT
			/&
			
			To compile/assemble more than one program and catalog all of them into the relocatable library, use the same setup except that the compilation/assembly control cards (CATALR through /* inclusive) for each program follow the /* of the preceding program.
			// CLOSE through /& follow the /* of the last program.
			
			Using a DASD file for SYSPCH/SYSIPT. (Must always be a permanent assignment.)
			
			// JOB jobname
			// DLBL balance of information required for SYSPCH file
			// EXTENT balance of information required for SYSPCH file
			ASSGN SYSPCH,X'cuu'
			CATALR modulename[,v.m]
			PHASE name,origin[,NOAUTO]
			// EXEC COBOL
			
			source deck
			
			/*
			// PAUSE -- CLOSE AND REASSIGN SYSPCH FROM LOG
			// DLBL balance of information required for SYSIPT file
			// EXTENT balance of information required for SYSIPT file
			ASSGN SYSIPT,X'cuu'
			// EXEC MAINT
			/&
			
			The 'file-ID' in the DLBL statements must be the same in both sets.
			
			To compile/assemble more than one program and catalog all of them into the relocatable library, use the same setup except that the compilation/assembly control cards (CATALR through /* inclusive) for each program follow the /* of the preceding program.
			// PAUSE through /& follow the /* of the last program.

Figure 20. Maintenance Functions (Part 1 of 3)

Function	Unit	Element	Control Statements Required

Delete	Core Image Library	Phase	// JOB jobname
			// EXEC MAINT
			DELETC phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC MAINT
			DELETC prog1.ALL[,prog2.ALL,...]
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			DELETR module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC MAINT
			DELETR prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC MAINT
			DELETR ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			DELETS sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC MAINT
			DELETS sublib.ALL
			/*
			/&

		Library	// JOB jobname
			// EXEC MAINT
			DELETS ALL
			/*
			/&

Rename	Core Image Library	Phase	// JOB jobname
			// EXEC MAINT
			RENAMC oldname,newname[,oldname,newname,...]
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			RENAMR oldname,newname[,oldname,newname,...]
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			RENAMS sublib.oldname,sublib.newname[,sublib.oldname,sublib.newname,...]
			/*
			/&

Figure 20. Maintenance Functions (Part 2 of 3)

Function	Unit	Element	Control Statements Required

Update	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			UPDATE sublib.bookname,[s.bookl],[v.m],[nn]
			) ADD,) DEL, or) REP statements as required
			with source statements to be added
			) END [v.m[,C]]
			/*
			/&

Condense	Core Image Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS CL
			/*
			/&

	Relocatable Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS RL
			/*
			/&

	Source Statement Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS SL
			/*
			/&

	Libraries	All	// JOB jobname
			// EXEC MAINT
			CONDS CL,RL,SL
			/*
			/&

Set Parameter for Automatic Condense	Libraries	Any or All	// JOB jobname
			// EXEC MAINT
			CONDL lib=nnnnn[,lib=nnnnn[,lib=nnnnn]]
			/*
			/&
			
			Notes: Values to be substituted for lib:
			CL -- Core image library
			RL -- Relocatable library
			SL -- Source statement library
			Values to be substituted for nnnnn:
			One to five decimal digits, with a maximum value of 65536.

Reallo-cation	System	Library	// JOB jobname
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYSRES,balance of extent information
			// EXEC MAINT
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			/*
			/&
			
			Notes: Values to be substituted for id:
			CL -- Core image library
			RL -- Relocatable library
			SL -- Source statement library
			Values to be substituted for cylin and tracks:
			Any integer

Figure 20. Maintenance Functions (Part 3 of 3)

Function	Unit	Element	Control Statements Required

Display	Core Image Library	Phase	// JOB jobname
			// EXEC CSERV
			DSPLY phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC CSERV
			DSPLY prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC CSERV
			DSPLY ALL
			/*
			/&

		Directory	// JOB jobname
			// EXEC DSERV
			DSPLY CD
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			DSPLY module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			DSPLY prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			DSPLY ALL
			/*
			/&

		Directory	// JOB jobname
			// EXEC DSERV
			DSPLY RD
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			DSPLY sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			DSPLY sublib1.ALL[,sublib2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC SSERV
			DSPLY ALL
			/*
			/&

		Directory	// JOB jobname
			// EXEC DSERV
			DSPLY SD
			/*
			/&

Figure 21. Service Functions (Part 1 of 3)

Function	Unit	Element	Control Statements Required

Display	Transient Directory	Directory	// JOB jobname
			// EXEC DSERV
			DSPLY TD
			/*
			/&

	System Directory	Directory	// JOB jobname
			// EXEC DSERV
			/*
			/&

	Directories	All	// JOB jobname
			// EXEC DSERV
			DSPLY ALL
			/*
			/&

Punch	Core Image Library	Phase	// JOB jobname
			// EXEC CSERV
			PUNCH phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC CSERV
			PUNCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC CSERV
			PUNCH ALL
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			PUNCH module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			PUNCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			PUNCH ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			PUNCH sublib.book1[,sublib.book2,...][,CMPRSD]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			PUNCH sublib1.ALL[,sublib2.ALL,...][,CMPRSD]
			/*
			/&

		Library	// JOB jobname
			// EXEC SSERV
			PUNCH ALL[,CMPRSD]
			/*
			/&

Figure 21. Service Functions (Part 2 of 3)

Function	Unit	Element	Control Statements Required

Display and Punch	Core Image Library	Phase	// JOB jobname
			// EXEC CSERV
			DSPCH phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC CSERV
			DSPCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC CSERV
			DSPCH ALL
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			DSPCH module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			DSPCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			DSPCH ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			DSPCH sublib.book1[,sublib.book2,...][,CMPRSD]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			DSPCH sublib1.ALL[,sublib2.ALL,...][,CMPRSD]
			/*
			/&

		Library	// JOB jobname
			// EXEC SSERV
			DSPCH ALL[,CMPRSD]
			/*
			/&

Figure 21. Service Functions (Part 3 of 3)

Function	Unit	Element	Control Statements Required

Copy	Core Image Library	Phase	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC ALL
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR ALL
			/*
			/&

Figure 22. Copy Function (Part 1 of 4)

Function	Unit	Element	Control Statements Required

Copy	Source Statement Library	Book	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS sublib1.ALL[,sublib2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS ALL
			/*
			/&

	Libraries	All	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPY ALL
			/*
			/&

	Definition of a Private Relocatable Library		// JOB jobname
			// ASSGN SYSRLB,X'cuu'
			// DLBL IJSYSRL,'user identification of private library',date,code
			// EXTENT SYSRLB,balance of extent information
			// EXEC CORGZ
			NEWVOL RL=cylin(tracks)
			/*
			/&
			
			The private library can be updated with either a MAINT or a copy 'MERGE' function.

	Definition of a Private Source Statement Library		// JOB jobname
			// ASSGN SYSSLB,X'cuu'
			// DLBL IJSYSSL,'user identification of private library',date,code
			// EXTENT SYSSLB,balance of extent information
			// EXEC CORGZ
			NEWVOL SL=cylin(tracks)
			/*
			/&
			
			The private library can be updated with either a MAINT or a copy 'MERGE' function.

Figure 22. Copy Function (Part 2 of 4)

Function	Unit	Element	Control Statements Required

Copy	Definition and Creation of a Private Relocatable Library		// JOB jobname
			// ASSGN SYSRLB,X'cuu'
			// DLBL IJSYSRL,'user identification of private library',date,code
			// EXTENT SYSRLB,balance of extent information
			// EXEC CORGZ
			NEWVOL RL=cylin(tracks)
			COPYR operands
			/*
			/&
			
			The private library can be updated with either a MAINT or a copy 'MERGE' function.

	Definition and Creation of a Private Source Statement Library		// JOB jobname
			// ASSGN SYSSLB,X'cuu'
			// DLBL IJSYSSL,'user identification of private library',date,code
			// EXTENT SYSSLB, balance of extent information
			// EXEC CORGZ
			NEWVOL SL=cylin(tracks)
			COPYS operands
			/*
			/&
			
			The private library can be updated with either a MAINT or a copy 'MERGE' function.

	Merge System Residence To New System Residence		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRS,'NEW SYSTEM RESIDENCE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			MERGE RES,NRS
			COPY statements (COPYC, COPYR, COPYS, COPY)
			as required
			/*
			/&

	Merge New System Residence to System Residence		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRS,'NEW SYSTEM RESIDENCE',date,code
			// EXTENT SYS002,balance of extent information
			// EXEC CORGZ
			MERGE NRS,RES
			COPY statements (COPYC, COPYR, COPYS, COPY)
			as required
			/*
			/&

	Merge System Residence to Private Libraries		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRL,'PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYSRLB,balance of extent information
			// DLBL ISYSSL,'PRIVATE SOURCE STATEMENT LIBRARY',date,code
			// EXTENT SYSSLB,balance of extent information
			// EXEC CORGZ
			MERGE RES,PRV
			COPY statements (COPYR, COPYS) as required
			/*
			/&

	Merge New System Residence to Private Libraries		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRS,'NEW SYSTEM RESIDENCE',date,code
			// EXTENT SYS002,balance of extent information
			// DLBL IJSYSRL,'PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYSRLB,balance of extent information
			// DLBL IJSYSSL,'PRIVATE SOURCE STATEMENT LIBRARY',date,code
			// EXTENT SYSSLB,balance of extent information
			// EXEC CORGZ
			MERGE NRS,PRV
			COPY statements (COPYR, COPYS) as required
			/*
			/&

Figure 22. Copy Function (Part 3 of 4)

Function	Unit	Element	Control Statements Required

Copy	Merge Private Libraries to System Residence		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSPR,'PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYS001,balance of extent information
			// DLBL IJSYSPS,'PRIVATE SOURCE STATEMENT LIBRARY',date,code
			// EXTENT SYS000,balance of extent information
			// EXEC CORGZ
			MERGE PRV,RES
			COPY statements (COPYR, COPYS) as required
			/*
			/&

	Merge Private Libraries to New System Residence		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRS,'NEW SYSTEM RESIDENCE',date,code
			// EXTENT SYS002,balance of extent information
			// DLBL IJSYSPR,'PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYS001,balance of extent information
			// DLBL IJSYSPS,'PRIVATE SOURCE STATEMENT LIBRARY',date,code
			// EXTENT SYS000,balance of extent information
			// EXEC CORGZ
			MERGE PRV,NRS
			COPY statements (COPYR, COPYS) as required
			/*
			/&

	Merge Private Libraries to Private Libraries		// JOB jobname
			// ASSGN (statements as required)
			// DLBL IJSYSRL,'PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYSRLB,balance of extent information
			// DLBL IJSYSPR,'SECOND PRIVATE RELOCATABLE LIBRARY',date,code
			// EXTENT SYS001,balance of extent information
			// DLBL IJSYSSL,'PRIVATE SOURCE STATEMENT LIBRARY',date,code
			// EXTENT SYSSLB,balance of extent information
			// DLBL IJSYSPS,'SECOND PRIVATE SOURCE STATEMENT LIBRARY',
			date,code
			// EXTENT SYS000,balance of extent information
			// EXEC CORGZ
			MERGE PRV,PRV
			COPY statements (COPYR, COPYS) as required
			/*
			/&
			
			To define the private library in the same job step, precede MERGE with NEWVOL statement.

●Figure 22. Copy Function (Part 4 of 4)

MAINTENANCE FUNCTIONS

The set of maintenance functions contains six subsets:

1. Catalog
2. Delete
3. Rename
4. Condense
5. Reallocate
6. Update

The catalog function adds a phase to the core image library, adds a module to the relocatable library, or adds a book to the source statement library. If control statements and books or modules to be cataloged are read from the same device, the control statement must precede its associated book or module.

Programs to be cataloged in the core image library must first be edited by the Linkage Editor. Input for the Linkage Editor can be from SYSIPT, from the relocatable library, or directly from the language translator if the CATAL option is specified in the OPTION statement. See the Linkage Editor section for a description of Linkage Editor functions that are performed prior to the catalog function for the core image library.

The delete function deletes an entry from a directory that corresponds to a phase, module, or book in a library. The phase, module, or book in the appropriate library is not removed; however, as far as the system is concerned, the phase, module, or book no longer exists. In addition, entire programs can be deleted from the core image library and the relocatable library.

The rename function renames an existing phase, module, or book in the appropriate library and directory.

The condense function eliminates vacancies between elements in a library. The condense function is used whenever a number of vacancies have accumulated within a library, resulting from deletions.

The reallocation function redefines the sizes of the libraries and the library directories. The reallocation function can be used to increase, decrease, eliminate, or add specific areas of the disk-resident system. Each library reallocated is automatically condensed. Any number of areas can be reallocated within a single run.

The update function updates statements within a book of a source statement library. One or more source statements may be added to, deleted from, or replaced in a book in the library without the necessity of replacing the entire book. The function also provides the following additional facilities:

1. Resequencing statements within a book in the source statement library.

2. Changing the change level (v.m) of the book.

3. Adding or removing the change level verification requirement.

4. Copying a book with optional retention of the old book with a new name (for backup purposes).

When the /& statement is processed at the completion of a maintenance function, the system directory is displayed on SYSLST.

> Note: In order to re-create the transient directory, a /& statement must conclude all maintenance functions affecting the core image library.

Condense and/or reallocation functions cannot be performed while a foreground program is being executed. This is because the library directories do not accurately reflect the content of the corresponding library at various times during these functions. The control program ignores any request for the ATTN routine when a condense and/or reallocation function is being performed.

SERVICE FUNCTIONS

The set of service functions contains three subsets:

1. Display
2. Punch
3. Display and punch.

The disk-resident system can display and/or punch phases in the core image library, modules in the relocatable library, and books in the source statement library. In addition, all system directories can be displayed.

Whenever a requested service function provides punched-card output, the output is on the device assigned to SYSPCH. Whenever a requested service function provides printed output, the output is on the device assigned to SYSLST.

COPY FUNCTION

The copy function copies the disk resident system selectively or completely. A complete copy can be used to obtain backup if the original system is inadvertently destroyed. A selective copy can be used to reduce a complete system to a system that is designed to perform a specific purpose. The copy function is also used to define private libraries and to copy modules and books into them from the SYSRES relocatable and source statement libraries. The MERGE control statement allows selective or complete transfer between libraries, private or resident, without generating punched output and recataloging. The libraries can be created in the same job step as MERGE or in a previous job. The result is less complicated job streams, less operator intervention, and shorter job run time. System generation and maintenance is simplified since library routines can be copied directly from one version of the system to a subsequent version.

GENERAL CONTROL STATEMENT FORMAT

The librarian control statements are similar in format to statements processed by the Assembler. The operation field must be preceded by one or more blanks. The operation field must begin to the right of column 1 and must be separated from the operand field by at least one blank position. The operand field is terminated by the first blank position. It cannot

extend past column 71. Continuation statements are not recognized.

Librarian Functions: Core Image Library

This section describes the maintenance functions that relate to the core image library. The copy function for the core image library is discussed in the section entitled Copy Functions.

CORE IMAGE LIBRARY: MAINTENANCE FUNCTIONS

To request a maintenance function, other than the catalog function, for the core image library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of the maintenance functions (delete, rename, condense, set condense limit, or reallocate) can be requested within a single run. Any number of programs within the core image library can be acted upon in this run. Further, one or more of the maintenance functions (catalog, delete, rename, condense, set condense limit, or reallocate) for either of the other two libraries (relocatable or source statement), in addition to the update function for the source statement library, can be requested within this run; the same MAINT program maintains all three of the libraries.

If a maintenance operation concerns a phase contained in the transient directory (consisting of phases with a $$ prefix), the library routine directory (consisting of phases with a $ prefix), or the FGP directory (consisting of phases with a FGP prefix), these subdirectories are not updated until the /& statement is read.

Catalog

The catalog function adds a phase to the core image library. The CATAL option in the Job Control OPTION statement indicates that the Linkage Editor function will catalog the phase as a permanent phase in the core image library.

Each phase that is cataloged in the core image library derives its name from the PHASE control statement.

If a phase in the core image library is to be replaced by a new phase having the same name, only the catalog function need be used because the delete function is implied.

Up to 120 phases can be cataloged in the core image library within a single run.

For the catalog function for the core image library, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLNK must be assigned to a disk unit. SYS001 must be assigned to a tape or disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the catalog function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYSLNK, SYS001, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The OPTION control statement, with CATAL specified in the operand field, followed by
4. The appropriate Linkage Editor control statements (PHASE, INCLUDE, and ENTRY), followed by
5. The LBLTYP control statement, if needed, followed by
6. The EXEC LNKEDT control statement, followed by
7. Any job control statement.

Delete

The delete function removes references to specific phases or programs of the core image library. Any number of phases or programs can be deleted during a single run. The phases or programs are not physically deleted from the library; rather, the entry in the core image directory describing the phase or program is deleted. Programs and phases can be physically removed from the library (after the delete function has removed the entry from the directory) by performing a condense function. See the subsection entitled Condense.

The DELETC control statement in one of the following formats is used to delete phases or programs from the core image library.

```
DELETC phasename1[,phasename2,...]

DELETC prog1.ALL[,prog2.ALL,...]
```

In the first format, the entry in the operation field is DELETC. phasename in the operand field represents the name(s) of the phase(s) to be deleted. The name of the phase may be a maximum of eight characters. Entries in the operand field must be separated by commas.

In the second format, prog refers to the first four characters of the program name. (All phases within a program have the same first four characters. Therefore, the first four characters of each program within the library should be unique.) The four characters are followed by a period and ALL.

Any number of DELETC control statements can be used for the core image library within a single run.

For the delete function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the delete function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The DELETC control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Rename

The rename function changes the name of a phase in the core image library to another name.

Use the RENAMC control statement to achieve the rename function. If, on the rename function, the new name is already in the directory or an old name is not in the directory, an error message is issued. On a valid pair of operands, the new name simply replaces the old name in the directory. In either case, a check is then made for more operands on the card. As soon as the statement is processed, the system recognizes only the new phase name. The RENAMC statement is in the following format.

```
RENAMC oldname,newname[,oldname,newname,..]
```

The operation field contains RENAMC. The operand field entries, oldname and newname, represent the old phase name and the new phase name. The two entries in the operand field must be separated by a comma. The names in the operand field may be a maximum of eight characters.

Any number of RENAMC control statements can be used for the core image library within a single run.

For the rename function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The RENAMC control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Condense

The condense function eliminates vacancies, resulting from delete or catalog functions, between programs in the core image library. The condense function is used whenever a number of vacancies have accumulated within the library.

The CONDS control statement, in the following format, is used to condense the core image library.

```
CONDS CL
```

The operation field contains CONDS. The operand field contains CL. The relocatable library and/or the source statement library can also be condensed in this run. If this is desired, the entry RL (for the relocatable library) and SL (for the source statement library) can appear in the operand field. Multiple entries in the operand field are separated by commas.

For the condense function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The CONDS control statement, followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

CORE IMAGE LIBRARY: SERVICE FUNCTIONS

To request a service function for the core image library, use the following EXEC control statement.

```
// EXEC CSERV
```

One or more of the three service functions can be requested within a single run. Punched output is sequenced in columns 77 through 80. The first card punched for each phase is sequenced zero. Any number of phases within the core image library can be acted upon in this run.

Display

The display function produces a printout of a phase in the core image library. Any number of phases can be displayed within a single run. The printed output consists of a header and the phase.

The printed header contains the phase name and the length of the phase in number of bytes.

The printed output of the phase contains a three-byte hexadecimal load address of the first byte in the line, followed by 48 bytes of text displayed in hexadecimal.

The DSPLY control statement in one of the following formats is used to display phases in the core image library.

```
DSPLY phase1[,phase2,...]

DSPLY prog1.ALL[,prog2.ALL,...]

DSPLY ALL
```

The first format is used if only specific phases are to be displayed. The entry in the operation field is DSPLY. phase in the operand field represents the name of the phase to be displayed. If more than one phase is to be displayed, the phase names are separated by commas. Phase names must be from one to eight characters long.

The second format is used when an entire program is to be displayed. The entry in the operation field is DSPLY. In the operand field, prog refers to the first four characters of the phase names making up a program. (All phases of a multiphase program should have the same first four characters.) The four characters are followed by a period and ALL.

The third format is used if the entire core image library is to be displayed. The entry in the operation field is DSPLY. The entry in the operand field is ALL.

It is possible to use all three types of operands together in a single control statement.

For the display function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC CSERV control statement, followed by
4. The DSPLY control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Punch

The punch function converts a phase in the core image library into a punched-card output deck.

Any number of phases in the core image library can be punched within a single run unless SYSPCH is assigned to a disk or tape unit (for subsequent use as SYSIPT to a link edit job step). In this case a maximum of 120 phases can be punched. The punched-card output is acceptable as input to the Linkage Editor for recataloging to the core image library. Phases punched from the core image library are not relocatable. Thus, when cataloged to the core image library, they origin in main storage at the same address as when they were originally linkage edited.

The following cards are contained in the phase decks punched from the core image library.

1. PHASE card: contains the phase name and the beginning load address.
2. ESD card: SD type, contains the phase name, the length of the phase, and the beginning load address.
3. TXT cards: contain the loading address of the first byte in the card, the number of bytes of text punched in the card (usually 56, except for the last card), the identification number of the control section (always 0001) containing the text, and the actual text.
4. END card: contains the transfer address, and signifies the end of the phase.

Because phases are not relocatable, no RLD cards are punched.

To facilitate recataloging of the phase(s), a /* card will be the last card of any punched output of a CSERV job step. If SYSIPT and SYSPCH are assigned to the same device (1442 or 2520), the end-of-file indicator (/*) from SYSIPT is selected to stacker 2.

The PUNCH control statement in one of the following formats is used to convert phases in the core image library to punched-card output.

```
PUNCH phase1[,phase2,...]

PUNCH prog1.ALL[,prog2.ALL,...]

PUNCH ALL
```

The first format is used if only specific phases are to be punched. The entry in the operation field is PUNCH. The entry in the operand field, phase, represents the name of the phase to be punched. If more than one phase is to be punched, the phase names are separated by commas. Phase names must be from one to eight characters long.

The second format is used when an entire program is to be punched. The entry in the operation field is PUNCH. In the operand field, prog refers to the first four characters of the phase names making up a program. (All phases of a multiphase program should have the same first four characters.) The four characters are followed by a period and ALL.

The third format is used if the entire core image library is to be punched. The entry in the operation field is PUNCH. The entry in the operand field is ALL.

When SYSPCH is assigned to a tape or disk unit, each card image is preceded by a stacker-select character.

For the punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit.

SYSLST must be assigned to a printer, a tape unit, or a disk unit and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSPCH, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC CSERV control statement, followed by
4. The PUNCH control statement(s), followed by
5. The /* control statement, followed by
6. Control statements for a succeeding job step, or
7. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSIN and also to SYSPCH, enough blank cards for punching the module must follow each PUNCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Display and Punch

The display-and-punch function combines the separate operations of the display function and the punch function. The output of the display-and-punch function is identical to that described in the two preceding subsections. Any number of phases in the core image library can be displayed and punched within a single run.

The DSPCH control statement is used to convert phases in the core image library to printed and punched-card output. The DSPCH control statement is in one of the following formats.

```
DSPCH phase1[,phase2,...]

DSPCH prog1.ALL[,prog2.ALL,...]

DSPCH ALL
```

The first format is used if only specific phases are to be displayed and punched. The entry in the operation field is DSPCH. The entry in the operand field, phase, represents the name of the phase to be displayed and punched. If more than one phase is to be displayed and purched, the phase names are separated by commas. Phase names must be from one to eight characters long.

The second format is used when an entire program is to be displayed and punched. The entry in the operation field is DSPCH. In the operand field, prog refers to the first four characters of the names of the phases making up the program. (All phases of a multiphase program should have the same first four characters.) The four characters are followed by a period and ALL.

The third format is used if the entire core image library is to be displayed and punched. The entry in the operation field is DSPCH. The entry in the operand field is ALL.

When SYSPCH is assigned to a tape or disk unit, each card image is preceded by a stacker-select character.

For the display and punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display-and-punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, SYSPCH, and SYSLOG. The ASSGN statements are followed by
3. The EXEC CSERV control statement, followed by
4. The DSPCH control statement(s), followed by
5. The /* control statement, followed by
6. Control statements for a succeeding job step, or
7. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSIN and also to

SYSPCH, enough blank cards for punching the module must follow each DSPCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Librarian Functions: Relocatable Library

This section describes the maintenance and service functions that relate to the relocatable library. The copy function for the relocatable library is discussed in the Copy Functions section.

RELOCATABLE LIBRARY: MAINTENANCE FUNCTIONS

To request a maintenance function for the relocatable library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of the maintenance functions (catalog, delete, rename, condense, set condense limit, or reallocate) can be requested within a single run. Any number of modules within the relocatable library can be acted upon in this run. Further, one or more of the maintenance functions for either of the other two libraries (core image or source statement) can be requested within this run, for the same MAINT program maintains all three libraries. The maintenance function for private libraries is discussed in the section Private Libraries.

Catalog

The catalog function adds a module to the relocatable library. Input for the catalog function is from the device assigned to SYSIPT. A module in the relocatable library is the output of a complete language translator run.

A module added to the relocatable library is removed by using the delete function.

The catalog function implies a delete function. Thus, if a module exists in the relocatable library with the same name as a module to be cataloged, the module in the relocatable library is deleted.

The CATALR control statement is required to add a module to the relocatable library. The CATALR control statement is read from the device assigned to SYSIPT and is in the following format.

```
CATALR modulename[,v.m]
```

The operation field contains CATALR. The entry in the operand field, modulename, is the name by which the module will be known to the control system. The modulename is one to eight characters, the first of which must not be an asterisk.

The optional entry in the operand field, v.m, specifies the change level at which the module is to be cataloged. v may be any decimal number from 0-127. m may be any decimal number from 0-255. If this operand is omitted, a change level of 0.0 is assumed.

A change level can be assigned only when a module is cataloged. The change level is displayed and punched by the service functions.

The statements composing the input for a module are described in the Linkage Editor section. The statements are:

1. PHASE
2. INCLUDE control statement (if appropriate)
3. ESD
4. TXT
5. RLD
6. REP
7. END
8. ENTRY

These statements are read from the device assigned to SYSIPT. All input is diagnosed by the Linkage Editor. The CATALR statement is recognized but ignored by the Linkage Editor. The END statement indicates end of module.

The ENTRY statement can only be used in a module that contains only Linkage Editor control statements and an END statement. The ENTRY statement must be the last control statement in the module, preceding the END statement.

Normally, modules in the relocatable library are output from a language translator. However, the user can construct an artificial module of Linkage Editor control statements, referred to as a calling module. The following example illustrates a valid calling module:

```
PHASE PHNAM1,ROOT
INCLUDE MODULE1
PHASE PHNAM2,*
INCLUDE MODULE2
PHASE PHNAM3,PHNAM2
      .
      .
      .
ENTRY CSECTNME
END
```

Operands in INCLUDE statements refer to modules in the relocatable library. If, for example, the preceding calling module is cataloged by the name BIGPROG, all modules referred to in BIGPROG can be linkage edited by using the following control statements:

```
// OPTION CATAL
   INCLUDE BIGPROG
// EXEC LNKEDT
```

A calling module may consist only of INCLUDE statements. In this case, the PHASE statements would precede the included modules.

A ninth statement, SYM, can be in the Linkage Editor input. When recognized, however, it is bypassed by the Linkage Editor. (The SYM statement identifies the symbol table output by the Assembler as a result of specifying SYM in the OPTION statement. The symbol table may be used for Autotest processing.)

For the catalog function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer keyboard. If SYSIN is assigned to a tape unit, the MAINT program assumes that the tape is positioned to the first input record. The tape is not rewound at the end of job.

Control statement input for the catalog function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The CATALR control statement(s), followed by
5. The module to be cataloged, followed by
6. The /* control statement if other job steps are to follow, or
7. The /& control statement, which is the last control statement of the job.

Any number of modules can be cataloged in a single run. Each module must immediately follow its respective CATALR control statement.

An additional capability of the system permits a user to assemble or compile a program and to catalog it to the system relocatable, or private relocatable, library in one continuous run. The user inserts a CATALR statement in his job control input stream preceding the assembler/compiler execute statement. The CATALR statement will be written on the SYSPCH file (on tape or DASD) ahead of the assembler/compiler output. The user then reassigns the SYSPCH file as SYSIPT and executes the MAINT program to perform the catalog function. The output of the assembly/compilation (on tape or DASD) may be cataloged immediately or it may be cataloged at some later time. It can also be held after cataloging as backup of the assembly/compilation.

Note: This facility is not available for RPG and PL/I (D) compilations or for IBM 2314 applications.

Delete

The delete function deletes references to specific modules in the relocatable library. Any number of modules may be deleted during a single run. The modules are not physically deleted from the library; rather, the entry in the relocatable directory describing the module is deleted. Modules can be physically removed from the library (after the delete function has removed the entry from the directory) by performing a condense function. See the subsection entitled Condense.

The DELETR control statement in one of the following formats is used to delete a module from the relocatable library.

```
DELETR modname[,modname,...]

DELETR prog1.ALL[,prog2.ALL,...]

DELETR ALL
```

The first format is used when a specific module is to be deleted. The entry in the

operation field is DELETR. The entry in the operand field, modname, is the name of the module to be deleted. If more than one module is to be deleted, the module names are separated by a comma. modname is one to eight characters, the first of which must not be an asterisk.

The second format is used when an entire program is to be deleted. The entry in the operation field is DELETR. In the operand field, prog refers to the first three characters of the modules used to build the program. (All IBM-supplied modules in the relocatable library making up a program have the same first three characters, such as IJQ for the Assembler and IJS for COBOL.) The three characters are followed by a period and ALL.

The third format is used if the entire library is to be deleted. The entry in the operation field is DELETR. The entry in the operand field is ALL. When this function is performed, the system status record is reset to show that all library blocks are now available to the system. Therefore, it is unnecessary to perform a condense function after a DELETR ALL has been performed.

Any number of DELETR control statements can be used for the relocatable library within a single run.

For the delete function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Note: Use the first format to delete the DIMOD. The second format causes the CPMOD to be deleted. This results in job termination.

Control statement input for the delete function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The DELETR control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Rename

The rename function changes the name of a module in the relocatable library to another name.

Use the RENAMR control statement to achieve the rename function. If, on the rename function, the new name is already in the directory or an old name is not in the directory, an error message is issued. On a valid pair of operands, the new name simply replaces the old name in the directory. In either case, a check is then made for more operands on the card. As soon as the statement is processed, the system recognizes only the new module name. The RENAMR statement is in the following format:

```
RENAMR oldname,newname[,oldname,newname,..]
```

The operation field contains RENAMR. The entries in the operand field, oldname and newname, represent the old module-name and the new module-name, respectively, and are separated by a comma. oldname and newname are one to eight characters, the first of which must not be an asterisk.

Any number of RENAMR control statements can be used for the relocatable library within a single run.

For the rename function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The RENAMR control statement(s), followed by
5. The /* control statement if other job steps are to follow, or

6. The /& control statement, which is the last control statement of the job.

Condense

The condense function eliminates vacancies, resulting from delete or catalog functions, between modules in the relocatable library. The condense function is used whenever a number of vacancies have accumulated within the library.

The CONDS control statement, in the following format, is used to condense the relocatable library.

```
CONDS RL
```

The operation field contains CONDS. The operand field contains RL.

For the condense function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The CONDS control statement, followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

RELOCATABLE LIBRARY: SERVICE FUNCTIONS

To request a service function for the relocatable library, use the following EXEC control statement.

```
// EXEC RSERV
```

One or more of the three service functions can be requested within a single run. Any number of modules within the relocatable library can be acted upon in this run. Punched output is sequenced in columns 77 through 80. The first card punched for each module is sequenced zero. The service function for private libraries is discussed in the section Private Libraries.

Display

The display function produces a printout of a module in the relocatable library. Any number of modules can be displayed within a single run. The printed output consists of a header and the module.

Contained in the printed header is the module name and the number of records needed to contain the module.

The printed output of the module is represented by hexadecimal characters and EBCDIC, depending on the type of record and the information contained within the record. The fields of the printed output correspond to the card columns of the output cards shown in Appendix E.

The DSPLY control statement in one of the following formats is used to display modules in the relocatable library.

```
DSPLY module1[,module2,...]

DSPLY prog1.ALL[,prog2.ALL,...]

DSPLY ALL
```

The first format is used if only specific modules are to be displayed. The entry in the operation field is DSPLY. module in the operand field represents the name of the module to be displayed. If more than one module is to be displayed, the module names are separated by commas. Module names must be from one to eight characters long.

The second format is used when an entire program is to be displayed. The entry in the operation field is DSPLY. In the operand field, prog refers to the first three characters of the modules used to build the program. (All IBM-supplied modules in the relocatable library making up a program have the same first three characters, such as IJQ for the Assembler and IJS for COBOL.) The three characters are followed by a period and ALL.

The third format is used if the entire relocatable library is to be displayed.

The entry in the operation field is DSPLY. The entry in the operand field is ALL.

For the display function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC RSERV control statement, followed by
4. The DSPLY control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Punch

The punch function converts a module in the relocatable library into a punched-card output deck.

Any number of modules in the relocatable library can be punched within a single run. The punched-card output is acceptable to every function that uses relocatable modules as input. Each module punched is preceded by a CATALR statement. The last card punched is a /* statement.

The PUNCH control statement in one of the following formats is used to convert modules in the relocatable library to punched-card output.

```
PUNCH module1[,module2,...]

PUNCH prog1.ALL[,prog2.ALL,...]

PUNCH ALL
```

The first format is used if only specific modules are to be punched. The entry in the operation field is PUNCH. The entry in the operand field, module, represents the name of the module to be punched. If more than one module is to be punched, the module names are separated by commas. Module names must be from one to eight characters long.

The second format is used when an entire program is to be punched. The entry in the operation field is PUNCH. In the operand field, prog refers to the first three characters of the modules used to build the program. (All IBM-supplied modules in the relocatable library making up a program have the same first three characters, such as IJQ for the Assembler and IJS for COBOL.) The three characters are followed by a period and ALL.

The third format is used if the entire relocatable library is to be punched. The entry in the operation field is PUNCH. The entry in the operand field is ALL.

When SYSPCH is assigned to a tape or disk unit, each card image is preceded by a stacker-select character.

For the punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSPCH, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC RSERV control statement, followed by
4. The PUNCH control statement(s), followed by
5. The /* control statement, if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSIN and also to SYSPCH, enough blank cards for punching the module must follow each PUNCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Display and Punch

The display-and-punch function combines the separate operations of the display function and the punch function. The output of the display-and-punch function is identical to that described in the two preceding subsections. Any number of modules in the relocatable library may be displayed and punched within a single run. The last card punched is a /* statement.

The DSPCH control statement is used to convert modules in the relocatable library to printed and punched-card output. The DSPCH control statement is in one of the following formats.

```
DSPCH module1[,module2,...]

DSPCH prog1.ALL[,prog2.ALL,...]

DSPCH ALL
```

The first format is used if only specific modules are to be displayed and punched. The entry in the operation field is DSPCH. The entry in the operand field, module, represents the name of the module to be displayed and punched. If more than one module is to be displayed and punched, the module names are separated by commas. Module names must be from one to eight characters long.

The second format is used when an entire program is to be displayed and punched. The entry in the operation field is DSPCH. In the operand field, prog refers to the first three characters of the modules used to build the program. (All IBM-supplied modules in the relocatable library making up a program have the same first three characters, such as IJQ for the Assembler and IJS for COBOL.) The three characters are followed by a period and ALL.

The third format is used if the entire relocatable library is to be displayed and punched. The entry in the operation field is DSPCH. The entry in the operand field is ALL.

When SYSPCH is assigned to a tape or disk unit, each card image is preceded by a stacker-select character.

For the display and punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display-and-punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, SYSPCH, and SYSLOG. The ASSGN statements are followed by
3. The EXEC RSERV control statement, followed by
4. The DSPCH control statement(s), followed by
5. The /* control statement, if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSIN and also to SYSPCH, enough blank cards for punching the module must follow each DSPCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Librarian Functions: Source Statement Library

This section describes the maintenance and service functions that relate to the source statement library. The copy function for the source statement library is discussed in the Copy Function section.

SOURCE STATEMENT LIBRARY: MAINTENANCE FUNCTIONS

To request a maintenance function for the source statement library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of the maintenance functions (catalog, delete, rename, condense, update, set condense limit, or reallocate) can be requested within a single run. Any number of books within the source statement library can be acted upon in this run. Further, one or more of the maintenance functions for either of the other two libraries (core image or relocatable) can be requested within this run; the same MAINT program maintains all three

libraries. The maintenance function for private libraries is discussed in the section Private Libraries.

Catalog

The catalog function adds a book to a sublibrary of the source statement library. Card input for the catalog function is from the device assigned to SYSIPT. Books to be cataloged in the source statement library can be in any order. Any number of books can be added within a single run.

A book added to a sublibrary of the source statement library is removed by using the delete function.

The catalog function implies a delete function. Thus, if a book exists in a sublibrary with the same name as a book to be cataloged, the module in the sublibrary is deleted.

The CATALS control statement is required to add a book to a sublibrary of the source statement library. It is read from the device assigned to SYSIPT and is in the following format.

```
CATALS sublib.bookname[,v.m[,C]]
```

The operation field contains CATALS. The qualifier sublib in the operand field represents the sublibrary to which the book is to be cataloged and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the book.

bookname in the operand field represents the name of the book to be cataloged. The bookname is one to eight alphameric characters, the first of which must be alphabetic.

The first optional entry in the operand field, v.m, specifies the change level at which the book is to be cataloged. v may be any decimal number from 0-127. m may be any decimal number from 0-255. If this operand is omitted, a change level of 0.0 is assumed. The change level is displayed and punched by the service functions.

The second optional entry in the operand field, C, indicates that change level verification is required before updates are accepted for this book, providing the v.m operand is present on the update card (see UPDATE librarian control statement). This requirement is reflected in the DSERV output by a C appearing in the column headed LEV CHK (level check).

Books that are to be cataloged in a sublibrary of the source statement library must be preceded and followed by special statements indicating the beginning and the end of a book.

Macro definitions that are to be cataloged in the Assembler sublibrary are preceded by the MACRO statement and are followed by the MEND statement. MACRO is the standard macro definition header statement; MEND is the standard macro definition trailer statement.

When books to be retrieved by the Assembler COPY statement are to be cataloged to the Assembler sublibrary, the Assembler END statement should not be included in the book. (Assembler does not recognize END statements from the source statement library.)

Books other than macro definitions that are to be cataloged in the source statement library are preceded and followed by a BKEND statement. A BKEND statement must precede each book, and a BKEND statement must follow each book. If desired, the BKEND statement may precede and follow a macro definition (in addition to the MACRO and MEND statements). This is desirable when the options provided in the BKEND statement are required. The statement is in the following format.

```
BKEND [sub.book],[SEQNCE],[count],[CMPRSD]
```

The entry in the operation field is BKEND. All operand entries are optional. When used, the entries must be in the prescribed order, and need appear only in the BKEND statement preceding the book to be cataloged. The first entry in the operand field, sub.book, is identical to the operand of the CATALS control statement. If the second operand, SEQNCE, is specified, columns 76 to 80 of the card images making up the book are checked for ascending sequence numbers. The count operand specifies the number of card images in the book. When used, the card input is counted, beginning with the preceding BKEND statement and including the following BKEND statement. If an error is detected in either the sequence checking or the card count, an error message is printed. The error can be corrected, and the book can be recataloged. The CMPRSD operand indicates

that the book to be cataloged in the library is in the compressed format, output as a result of specifying CMPRSD when performing a PUNCH or DSPCH service function.

For the catalog function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSIPT must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement and card image input, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSIPT, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The CATALS control statement(s), followed by
5. The BKEND statement, and/or the macro header statement if the book is a macro definition, followed by
6. The book to be cataloged, followed by
7. The BKEND statement, and/or the MEND trailer statement if the book is a macro definition. If a BKEND statement precedes a book, one must also follow it; followed by
8. The /* control statement if other job steps are to follow, or
9. The /& control statement, which is the last control statement of the job.

Any number of books may be cataloged in a single run. Each book must immediately follow its respective CATALS control statement.

Delete

The delete function removes references to specific books in a sublibrary of the source statement library. The function can also delete an entire sublibrary or an entire library. Any number of books can be deleted during a single run. The books are not physically deleted from the sublibrary; rather, the entry in the source statement directory describing the book is deleted. Books can be physically removed from the library (after the delete function has removed the entry from the directory) by performing a condense function. See the subsection entitled Condense.

The DELETS control statement is used to delete books from the source statement library. The control statement is in one of the following formats.

```
DELETS sublib.book1[,sublib.book2,...]

DELETS sublib.ALL

DELETS ALL
```

The first format is used if only specific books are to be deleted. The entry in the operation field is DELETS. The qualifier sublib in the operand field represents the sublibrary containing the book to be deleted and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the book.

book in the operand field represents the name of the book in the sublibrary to be deleted. If more than one book is to be deleted, the entries must be separated by commas. If books to be deleted are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified.) The name of the book can be of any length; however, a maximum of the first eight characters is used to locate and delete the book. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be deleted. The entry in the operation field is DELETS. The first entry in the operand field is the name of the sublibrary to be deleted. The qualifier sublib represents the sublibrary containing the book to be deleted and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary.

The second entry in the operand field is ALL. The two entries must be separated by a period.

The third format is used if the entire source statement library is to be deleted. The entry in the operation field is DELETS. The entry in the operand field is ALL. When this function is performed, the system status record is reset to show that all library blocks are now available to the system. Therefore, it is unnecessary to perform a condense after a DELETS ALL has been performed.

For the delete function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the delete function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The DELETS control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Rename

The rename function changes the name of a book in the source statement library to another name.

Use the RENAMS control statement to achieve the rename function. If, on the rename function, the new name is already in the directory or an old name is not in the directory, an error message is issued. On a valid pair of operands, the new name simply replaces the old name in the directory. In either case, a check is then made for more operands on the card. As soon as the statement is processed, the system recognizes only the new book name. The RENAMS statement is in the following format.

```
RENAMS sublib.oldname,sublib.newname
    [,sublib.oldname,sublib.newname,...]
```

The operation field contains RENAMS. The qualifier sublib in the operand field represents the sublibrary containing the book to be renamed and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary.

oldname and newname represent the old book-name and the new book-name. If sublibrary is specified for the newname, it must be the same as for the oldname. Otherwise, an error message is issued. If omitted, the name is assumed to be the same as for the oldname. The entries in the operand field must be separated by commas. The names in the operand field can be of any length; however, only a maximum of the first eight characters is used by the system to locate and rename the book.

The DOS Assemblers flag any reference to a macro in the source statement library as an UNDEFINED OPERATION CODE if the cataloged name of the macro is not identical to the operation code in the macro prototype statement. The Assemblers locate macros in the source statement library by the cataloged name, but thereafter use the operation code of the macro prototype statement for identification.

Any number of RENAMS control statements can be used for the source statement library within a single run.

For the rename function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The RENAMS control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Update

The update function updates properly identified statements within a book of a source statement library. Statements are identified in the identification field, columns 73-80, which is fixed in format as:

Columns 73-76	Program identification (constant throughout the book).
Columns 77-80	Sequence number of the statement within the book

One or more source statements may be added to, deleted from, or replaced in a book in the library without the necessity of replacing the entire book. The function also provides these facilities:

1. Resequencing statements within a book in the source statement library.
2. Changing the change level (v.m) of the book.
3. Adding or removing the change level requirement.
4. Copying a book with optional retention of the old book with a new name (for backup purposes).

The UPDATE control statement has the following format:

```
UPDATE sublib.bookname,[s.book1],[v.m],[nn]
```

The operation field contains UPDATE.

sublib in the operand field represents the sublibrary that contains the book to be updated. It may be any of the characters A-Z, 0-9, $, @, or #.

bookname represents the book that is to be updated in the sublibrary.

s.book1 in the operand field provides a temporary update option. The old book is renamed s.book1 and the updated book is named sublib.bookname. s indicates the sublibrary that contains the old, renamed book. It may be one of the characters A-Z, 0-9, $, @, or #. If this operand is not specified, the old book is deleted.

v.m represents the change level of the book to be updated. v may be any decimal number from 0-127. m may be any decimal number from 0-255. This operand must be present if change level verification is to be done, and it must correspond to the change level in the book's directory entry. If the change level is verified, the change level in the book's directory entry is increased by 1 for verification of the next update. If m is at its maximum value and an update is processed, m is reset to 0 and the value of v is increased by 1.

If both v and m are at their maximum values and an update is processed, both v and m are reset to zeros. If the directory entry specifies that change level verification is not required before updating, the change level operand in the statement is ignored. Use of the optional entry C in the CATALS control statement at the time the book is cataloged to the library determines if change level verification is required before updating.

nn in the operand field represents the resequencing status required for the update. nn may be a one- or two-character decimal number from 1-10, or it may be the word NO. If nn is a decimal number, it represents the increment that will be used in resequencing the statements in the book. If nn is NC, the statements will not be resequenced. If nn is not specified, the statements will be resequenced with an increment of 1. When a book is resequenced, the sequence number of the first statement is 0000.

The UPDATE control statement is followed by ADD, DEL (delete), and/or REP (replace) control statements as required. Each ADD or REP statement is followed by source statements that are to be added to the book. The update section is terminated by an END statement. The ADD, DEL, REP, and END statements are identified as update control statements by a right parenthesis

in the first position (column 1 in card format). The second position is blank. This is a variation from the general librarian control statement format, but it clearly identifies these control statements as part of the update function.

ADD Statement: The ADD statement is used for the addition of source statements to a book. The format is:

```
) ADD seq-no
```

ADD indicates that source statements following this statement are to be added to the book.

seq-no represents the sequence number of the statement in the book after which the new statements are to be added. It may be any decimal number from one to four characters in length.

DEL Statement: The DEL statement causes the deletion of source statements from the book. The format is:

```
) DEL first-seq-no[,last-seq-no]
```

DEL indicates that statements are to be deleted from the book.

first-seq-no and last-seq-no represent the sequence numbers of the first and last statements of a section to be deleted. Each number may be a decimal number from one to four characters in length. If last-seq-no is not specified, the statement represented by first-seq-no is the only statement deleted.

REP Statement: The REP statement is used when replacement of source statements in a book is required. The format is:

```
) REP first-seq-no[,last-seq-no]
```

REP indicates that source statements following this statement are to replace existing statements in a book.

first-seq-no and last-seq-no represent the sequence numbers of the first and last statements of a section to be replaced. Each number may be a decimal number from one to four characters in length. Any number of new statements can be added to a book when a section is replaced. (The number of statements added need not equal the number of statements being replaced.)

Sequence number 9999 is the highest number acceptable for a statement to be updated. If the book is so large that statement sequence numbers have wrapped around (progressed from 9998, 9999, to 0000, 0001), it will not be possible to update statements 0000 and 0001.

END Statement: This statement indicates the end of an update to a book. The format is:

```
) END [v.m[,C]]
```

END indicates the end of updates to a book.

The v.m operand provides another means of explicitly setting the change level of a book in the library. (The other way is through the use of the v.m operand in the CATALS statement.) v may be any decimal number from 0-127. m may be any decimal number from 0-255.

C indicates that change level verification is required before any subsequent updates to this book.

If v.m is specified and C is omitted, the book does not require change level verification before a subsequent update. This feature removes a previously specified verification requirement for a particular book.

If both optional operands are omitted, the change level in the book's directory entry is increased, as the result of the update, and the verification requirement remains unchanged.

UPDATE Function Invalid Operand Defaults

UPDATE Statement:

1. If the first or second operand is invalid, the statement is flagged, the book is not updated, and the remaining control statements are checked for validity.
2. If change level verification is required and the wrong change level is specified, the statement is flagged, the book is not updated, and the remaining control statements are checked for validity.
3. If the resequencing operand is invalid, resequencing is done in increments of 1.

ADD, DEL, or REP Statements:

1. If there is an invalid operation or operand in an ADD, DEL, or REP statement, the statement is flagged, the book is not updated, and the remaining control statements are checked for validity. All options of the UPDATE and END statements are ignored.

2. The second operand must be greater than the first operand in a DEL or REP statement. If not, the statement is considered invalid, it is flagged, the book is not updated, and the remaining control statements are checked for validity. If a second operand is present on an ADD statement, it is flagged as an invalid operand and ignored. The book is updated and normal processing continues. All options of the UPDATE and END statements are ignored.

3. All updating to a book between an UPDATE statement and an END statement must be in ascending sequential order of statement sequence numbers. The first operand of a DEL or REP statement must be greater than the last operand of the preceding control statement. The operand of an ADD statement must be equal to or greater than the last operand of the preceding control statement. Consecutive ADD statements must not have the same operand. If these conditions are not met, the default is the same as for items 1 and 2.

END Statement: If the first operand of the END statement is invalid, the statement is flagged, both operands are ignored, and the book is updated as though no operands were present. If the second operand is invalid, the statement is flagged, the operand is ignored, and the book is updated as though the second operand were not specified.

Out-of-Sequence Updates: If the source statements to be added to a book are not in sequence, or do not contain sequence numbers, the book is updated and a message indicating the error appears following the END statement. If the resequencing option has been specified in the UPDATE statement, the book is sequenced by the specified value and subsequent updating is possible. If the resequencing option is not specified, the book is resequenced in increments of 1 and subsequent updating will be possible. If the resequencing option NO is specified, the book will not be in sequence and subsequent updating may not be possible.

For the update function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the update function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The UPDATE control statement, followed by
5.) ADD,) DEL, or) REP statements with appropriate source statements, followed by
6.) END statement, followed by
7. The /* control statement, if other job steps are to follow, or
8. The /& control statement, which is the last control statement of the job.

UPDATE Function - Activity Log: The UPDATE function provides a log, on SYSLST, of all update activity to books in the source statement library. The log indicates the operation performed and all operands of the update control statements. It also shows the contents of the source statements affected, including the old program identification and sequence number, the new program identification and sequence number, and the update activity involved. Figures 23 and 24 show examples of an UPDATE job stream and an UPDATE activity log respectively.

```
// JOB UPDATE
// EXEC MAINT
UPDATE A.TESTCASE
) REP 0032
OPTNFND   CLC     0(1,REGG),BLANKS
) REP 0039,0044
OPTNCHK   CLC     0(5,REGG),=C'PUNCH'
          BE      PUNCH
          CLC     0(5,REGG),=C'DSPCH'
) ADD 0052
PUNCH     MVI     SWITCH,C'X'
          B       CHKOPND
) DEL 0134,0135
) END
/&
```

Figure 23. Example of an UPDATE Job Stream

```
UPDATE  A.TESTCASE

  ) REP 0032
     OPTNFND  CLI   0(REGG),C' '                          A4530032
     OPTNFND  CLC   0(1,REGG),BLANKS                      A4530032     A4530027     REPLACEMENT

  ) REP 0039,0044
     OPTNCHK  CLC   0(5,REGG),PUNCHOP                     A4530039
              BNE   CHKDSP                                A4530040
              MVI   SWITCH,C'X'                           A4530041
              B     CHKOPND                               A4530042
     CHKDSP   CLC   0(5,REGG),DSPCHOP                     A4530043
              BNE   MSGLOG                                A4530044
     OPTNCHK  CLC   0(5,REGG),=C'PUNCH '                  A4530039     A4530034     REPLACEMENT
              BE    PUNCH                                 A4530040     A4530035     REPLACEMENT
              CLC   0(5,REGG),=C'DSPCH '                  A4530041     A4530036     REPLACEMENT

  ) ADD 0052
              B     MSGLOG                                A4530052     A4530044     RESEQUENCED
     PUNCH    MVI   SWITCH,C'X'                           A4530053     A4530045     ADD
              B     CHKOPND                               A4530054     A4530046     ADD

  ) DEL 0134,0135
     DSPCHOP  DC    C'DSPCH '                             A4530134                  DELETE
     PUNCHOP  DC    C'PUNCH '                             A4530135                  DELETE

  ) END
```

Figure 24. Example of an UPDATE Activity Log

If no resequencing has been specified, only the old ID and sequence numbers of the statements involved are indicated.

Condense

The condense function eliminates vacancies, resulting from delete or catalog functions, between books in the source statement library. The condense function is used whenever a number of vacancies have accumulated within the library.

The CONDS control statement, in the following format, is used to condense the source statement library.

CONDS SL

The operation field contains CONDS. The operand field contains SL.

For the condense function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by

2. The ASSGN statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by

3. The EXEC MAINT control statement, followed by

4. The CONDS control statement, followed by

5. The /* control statement if other job steps are to follow, or

6. The /& control statement, which is the last control statement of the job.

SOURCE STATEMENT LIBRARY: SERVICE FUNCTIONS

To request a service function for the source statement library, use the following EXEC control statement.

```
// EXEC SSERV
```

One or more of the three service functions can be requested within a single run. Any number of books within the source statement library can be acted upon in this run. Punched output is sequenced in columns 77 through 80. The first card punched for each module is sequenced zero. The service function for private libraries is discussed in the section Private Libraries.

Display

The display function produces a printout of a book in the source statement library. Any number of books can be displayed within a single run.

Books are displayed in the card image format. Each book is preceded and followed by a BKEND statement.

The DSPLY control statement in one of the following formats is used to display books in the source statement library.

```
DSPLY sublib.book1[,sublib.book2,...]

DSPLY sublib1.ALL[,sublib2.ALL,...]

DSPLY ALL
```

The first format is used if only specific books are to be displayed. The entry in the operation field is DSPLY. The qualifier sublib in the operand field represents the sublibrary containing the book to be displayed and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

book in the operand field represents the name of the book in the sublibrary to be displayed. If more than one book is to be displayed, the entries must be separated by commas. If books to be displayed are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters in length. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be displayed. The entry in the operation field is DSPLY. The first entry in the operand field is the name of the sublibrary to be displayed. The qualifier sublib can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary.

The second entry in the operand field is ALL. The two entries must be separated by a period.

The third format is used if the entire source statement library is to be displayed. The entry in the operation field is DSPLY. The entry in the operand field is ALL.

For the display function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC SSERV control statement, followed by
4. The DSPLY control statement(s), followed by
5. The /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Punch

The punch function converts a book in the source statement library into a punched-card output deck. The resulting punched-card deck consists of card images of the book in the library. If the optional operand, CMPRSD, is specified, the card images are punched in the compressed form in which they are stored in the library. Each book is preceded by CATALS and BKEND statements and followed by a BKEND statement. The last book punched is followed by a BKEND statement and a /* statement.

Any number of books in the source statement library can be punched within a single run.

The PUNCH control statement in one of the following formats is used to convert books in the source statement library to punched-card output.

```
PUNCH sub.book1[,sub.book2,...][,CMPRSD]

PUNCH sub1.ALL[,sub2.ALL,...][,CMPRSD]

PUNCH ALL[,CMPRSD]
```

The first format is used if only specific books are to be punched. The entry in the operation field is PUNCH. The qualifier sub in the operand field represents the sublibrary containing the book to be punched and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

book in the operand field represents the name of the book in the sublibrary to be punched. The entry CMPRSD is used if the books are to be punched in the compressed form in which they are stored in the library. When this option is elected, the cards are punched in the first seventy-one columns. If more than one book is to be punched, the entries must be separated by commas. If books to be punched are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters long. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be punched. The entry in the operation field is PUNCH. The first entry in the operand field is the name of the sublibrary to be punched. The qualifier sub can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary.

The second entry in the operand field is ALL. The entry CMPRSD is used if the books are to be punched in the compressed format. A /* statement will always be punched at the end of the output. When SYSPCH is assigned to a tape unit or disk unit, each card image is preceded by a stacker-select character.

The third format is used if the entire source statement library is to be punched. The entry in the operation field is PUNCH. The entry in the operand field is ALL. The entry CMPRSD is used if the books are to be punched in the compressed format. A /* statement is always punched at the end of the output.

For the punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSPCH, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC SSERV control statement, followed by
4. The PUNCH control statement(s), followed by
5. The /* control statement, if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSRDR and also to SYSPCH, enough blank cards for punching the book must follow each PUNCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Display and Punch

The display-and-punch function combines the separate operations of the display function and the punch function. The output of the display-and-punch function is identical to that described in the two preceding subsections. Any number of books in the source statement library can be displayed and punched within a single run.

The DSPCH control statement in one of the following formats is used to convert books in the source statement library to printed and punched-card output.

```
DSPCH sub.book1[,sub.book2,...][,CMPRSD]

DSPCH sub1.ALL[sub2.ALL,...][,CMPRSD]

DSPCH ALL[,CMPRSD]
```

The first format is used if only specific books are to be displayed and punched. The entry in the operation field is DSPCH. The qualifier sub in the operand field represents the sublibrary containing the book to be displayed and punched and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

book in the operand field represents the name of the book in the sublibrary to be displayed and punched. The entry CMPRSD is used if the books are to be punched in the compressed format, but printed in the original card image format. If more than one book is to be displayed and punched, the entries must be separated by commas. If books to be displayed and punched are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters long. Continuation statements are not recognized. A /* statement will be punched at the end of the output.

The second format is used if an entire sublibrary is to be displayed and punched. The entry in the operation field is DSPCH. The first entry in the operand field is the name of the sublibrary to be displayed and punched. The qualifier sub can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary.

The second entry in the operand field is ALL. The entry CMPRSD is used if the books are to be punched in the compressed format. When SYSPCH is assigned to a tape or disk unit, each card image is preceded by a stacker-select character.

The third format is used if the entire source statement library is to be displayed and punched. The entry in the operation field is DSPCH. The entry in the operand field is ALL. The entry CMPRSD is used if the books are to be punched in the compressed format.

For the display and punch function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSPCH must be assigned to a

card punch, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display-and-punch function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, SYSPCH, and SYSLOG. The ASSGN statements are followed by
3. The EXEC SSERV control statement, followed by
4. The DSPCH control statement(s), followed by
5. The /* control statement, if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Whenever an IBM 1442 or 2520 Card Read Punch is assigned to SYSIN and also to SYSPCH, enough blank cards for punching the book must follow each DSPCH control statement. This prevents erroneously punching the cards of a following job step. Extra cards are automatically bypassed.

Librarian Functions: Directories

This section describes the display service function that relates to the five directories. The copy function for these directories are discussed in the section Copy Function.

To request the display service function for a directory (core image directory, relocatable directory, source statement directory, transient directory, or system directory), use the following EXEC control statement.

```
// EXEC DSERV
```

The display function prints the status of the directories defined for the system. Any number of directories can be displayed within a single run. No directory is displayed more than once in the same job step. The system directory is unconditionally displayed.

Each printed line contains one entry in the directory. All fields in the directory are headed by the title DEC (decimal) or HEX (hexadecimal).

Each printed directory is preceded by a header that contains the name of the directory in EBCDIC characters.

The DSPLY control statement in the following format is used to display specific directories or all directories.

```
DSPLY dir1[,dir2,...]

DSPLY ALL
```

The first format is used if only specific directories are to be displayed. The entry in the operation field is DSPLY. The entry in the operand field, dir, represents the name of the directory to be displayed. It can be:

1. TD for the transient directory. The transient directory shows the routines processed in the transient area and some frequently used library routines.
2. CD for the core image directory.
3. RD for the relocatable directory.
4. SD for the source statement directory.

If more than one directory is to be displayed, the symbols for the directories must be separated by a comma and can be in any order.

The second format is used if all five directories are to be displayed. The entry in the operation field is DSPLY. The entry in the operand field is ALL. In the absence of DSPLY, only a system directory is displayed. No more than one display of each directory is provided during each execution of DSERV.

For the display function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and

SYSLOG. The ASSGN statements are followed by

3. The EXEC DSERV control statement, followed by

4. The DSPLY control statement, followed by

5. The /* control statement if other job steps are to follow, or

6. The /& control statement, which is the last control statement of the job.

Reallocation Function

The reallocation function redefines the number of tracks and cylinders allotted to a 2311 or 2314 disk resident system.

Any number of libraries or directories can be reallocated within a single run. The reallocation function can be used to increase, decrease, eliminate, or add specified areas in the disk resident system. Each area that is reallocated is automatically condensed. The EXEC control statement required to perform a reallocation function is in the following format. Note that any other maintenance function (catalog, delete, or rename) for the three libraries may be performed in this run.

```
// EXEC MAINT
```

Associated with the EXEC statement for the reallocation function is the ALLOC control statement. The ALLOC control statement is in the following format.

```
ALLOC id=cylin(track)[,id=cylin(track),...]
```

The operation field contains ALLOC. The entry, id, in the operand field refers to the specific library and directory being reallocated and can be one of the following entries.

CL for the core image library and directory
RL for the relocatable library and directory
SL for the source statement library and directory.

The entry, cylin, in the operand field refers to the number of cylinders that contain the specified library. The entry, track, is enclosed within parentheses and refers to the number of tracks that contain the specified library directory. The tracks allocated to the directory are contained in the cylinders allocated to the library. The keyword operands are separated by a comma if more than one operand is present. For maximum efficiency, all requested operands should be entered on one statement.

When reallocation is being performed, only the areas being changed in the resident disk pack need be specified. To delete the relocatable library and/or the source statement library from SYSRES, allocations of RL=0(0) and/or SL=0(0) can be used. The respective library must first be cleared by deleting all entries. Consider this example:

```
ALLOC CL=16(1),SL=4(1)
```

In this example, the core image library would contain 16 cylinders and the core image directory would be in the first track of the first cylinder allocated to the library. The source statement library would contain 4 cylinders and the source statement directory would be in the first track of the first cylinder allocated to the library.

When increasing the size of a library, enough space must be left for the libraries that follow so that the ending cylinder address of the last library is not greater than 198. If enough space is unavailable for the following libraries, one or more of these libraries must be reduced in size to compensate for the increase.

As an example, the SYSRES library space was originally allocated as

```
CL=90(5),RL=40(2),SL=60(3)
```

An attempt to reallocate only the core image library to 120 cylinders would fail, because cylinder 198 would be exceeded. To avoid this, the combined sizes of the relocatable and source statement libraries can be reduced by 22 cylinders so as not to exceed the limit. Thus, the ALLOC statement could read:

```
ALLOC CL=120(7),RL=30(2),SL=48(3)
```

For the reallocation function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

If message 4n44A comes up due to a change in the label set when reallocating SYSRES, type DELETE on SYSLOG. The job will continue using new extents.

When executing the reallocation function of MAINT, a file must be defined for IJSYSRS on SYSRES via DLBL and EXTENT

control statements. The label information needed for this file is the same as for the IJSYSRS file of the copy disk function (see Copy Function), except that the symbolic unit is SYSRES instead of SYS002.

The lower extent for this file must be cylinder 0, track 1, and the upper extent must include the label cylinder. The label cylinder is one cylinder reserved for label information, and is located on the cylinder immediately following the last library of the system. The total allocation must include cylinder 0, all defined system libraries, and the label cylinder.

Control statement input for the reallocation function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. DLBL and EXTENT statements for the disk residence, followed by
4. The EXEC MAINT control statement, followed by
5. The ALLOC control statement, followed by
6. The /* control statement if other job steps are to follow, or
7. The /& control statement, which is the last control statement of the job.

Copy Function

The copy function performs the following operations, individually or in combination:

1. Defines and/or creates a new system pack.
2. Defines and/or creates private libraries.
3. Transfers phases, modules, or books between any assigned libraries.

The number of tracks and cylinders allocated to libraries and/or directories of the new system pack can be defined. The device number of the disk pack on which the disk resident system is to be copied is assigned to SYS002. It cannot have the same physical unit address as SYSRES but must be the same type of DASD. The SYS002 extent is completely overwritten with the new system.

To request a copy function, the following Job Control statement is required:

```
// EXEC CORGZ
```

Associated with the EXEC statement are three independent copy control statements and four COPY statements. The three independent copy control statements are:

1. ALLOC statement (see the section Reallocation Function). Defines any or all libraries on a new system resident pack. If this statement, or any individual library allocations are missing, the respective allocations of SYSRES are used. To avoid using SYSRES allocations, use zero allocations. For example, if the relocatable library is not to be defined, use ALLOC RI=0(0).

   When new limits for any of the three libraries are being established, re-IPL in order to have the correct new address of the label cylinder in the communications region in the supervisor.

2. NEWVOL statement (see the section Creation of Private Libraries). Defines private libraries.
3. MERGE statement (see the section Library Merge Function). Transfers data between libraries that were defined, or defined and created previously.

   Note: The lack of a NEWVOL or MERGE statement indicates the creation of a new system residence file.

The COPYC control statement is used to specify the phases or programs in the core image library that are to be copied. It is in one of the following formats.

```
COPYC phase1[,phase2,...]

COPYC prog1.ALL[,prog2.ALL,...]

COPYC ALL
```

The first format is used when specific phases are to be copied. The entry in the operation field is COPYC. The entry, phase, in the operand field represents the name(s) of the phase(s) to be copied. Entries in the operand field must be separated by commas.

The second format is used when specific programs are to be copied. The entry in the operation field is COPYC. The entry, prog.ALL, in the operand field represents the name of the program to be copied. prog is the first four characters of the program name. (All phases within a program have the same first four characters.) prog is followed by a period and ALL. Entries in the operand field must be separated by commas.

The third format is used to copy the complete core image library. The entry in the operation field is COPYC. The entry in the operand field is ALL.

The COPYR control statement is used to specify the modules in the relocatable library that are to be copied. It is in one of the following formats.

```
COPYR module1[,module2,...]

COPYR prog1.ALL[,prog2.ALL,...]

COPYR ALL
```

The first format is used when specific modules are to be copied. The entry in the operand field is COPYR. The entry, module, in the operand field represents the name(s) of the module(s) to be copied. Entries in the operand field must be separated by commas.

The second format is used when specific programs are to be copied. The entry in the operation field is COPYR. The entry, prog, in the operand field represents the name of the program to be copied. prog is the first three characters of the program name. (All modules within an IBM-supplied program have the same first three characters, such as IJB for the Supervisor and IJK for PL/I (D).) prog is followed by a period and ALL. Entries in the operand field must be separated by commas.

The third format is used to copy the complete relocatable library. The entry in the operation field is COPYR. The entry in the operand field is ALL.

The COPYS control statement is used to specify the books in the source statement library that are to be copied. It is in one of the following formats.

```
COPYS sublib.book1[,sublib.book2,...]

COPYS sublib1.ALL[,sublib2.ALL,...]

COPYS ALL
```

The first format is used when specific books are to be copied. The entry in the operation field is COPYS. The qualifier sublib in the operand field represents the name of the sub-library containing the book and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the book.

book represents the name(s) of the book(s) to be copied.

The second format is used when an entire sublibrary is to be copied. The entry in the operation field is COPYS. The entry, sublib, in the operand field represents the name of the sublibrary to be copied and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary

Any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand will be flagged as invalid and no processing will be done on the sublibrary. The qualifier sublib is followed by a period and ALL.

The third format is used to copy the complete source statement library. The entry in the operation field is COPYS. The entry in the operand field is ALL.

The COPY control statement copies the complete system, but can only be used when SYSRES contains all three libraries. It is in the following format:

```
COPY ALL
```

The entry in the operation field is COPY. The entry in the operand field is ALL.

Any number of elements of a particular library can be specified in one control statement. Continuation statements are not valid. All entries in the operand field must be separated by commas. Each library that is to be selectively copied requires a separate group of control statements.

The following functions are performed automatically by the CORGZ program, except when MERGE or NEWVOL statements are used

```
// JOB    COPY
// ASSGN  SYS002,X'191'
// DLBL   IJSYSRS,'DOS SYSTEM RESIDENCE FILE',99/365,SD
// EXTENT SYS002,111111,1,000,00001,1219
// EXEC   CORGZ
   ALLOC  CL=60(10),RL=30(10),SL=30(10)
   COPY   ALL
/*
/&
```

Figure 25. Example of a Valid Job Setup for the Copy Function

(see Library Merge Function and Private Libraries).

- All programs essential to a minimum system will be copied in a system copy. These programs are all logical and physical transients, IPL, Supervisor, Job Control, and Linkage Editor.

- The partition and system standard labels are copied from the SYSRES label cylinder to the label cylinder on SYS002.

When executing the copy disk (CORGZ) function, a file must be defined for IJSYSRS on SYS002 via DLBL and EXTENT control statements. The filename on the DLBL statement must be IJSYSRS. The file identification portion of the DLBL statement can be as shown in the example of the copy function.

The lower extent for this file must be cylinder zero, track one, and the upper extent must include the label cylinder. The label cylinder is one cylinder reserved for label information, and is located on the cylinder immediately following the last library of the system. The total allocation must include cylinder 0, all defined system libraries, and the label cylinder.

Figure 25 shows an example of a valid job setup for the copy function.

For the copy function, SYSIN must be assigned to a card reader, a tape unit, or a disk unit. SYS002 must be assigned to a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer keyboard.

Control statement input for the copy function, read from the properly assigned device (usually SYSIN), is:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYS002, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. DLBL and EXTENT statements for the disk pack on which the system residence is to be copied, followed by
4. The EXEC CORGZ control statement, followed by
5. The ALLOC control statement, if required, followed by
6. The COPY control statement(s), followed by
7. The /* control statement if other jobs are to follow, or
8. The /& control statement, which is the last control statement of the job.

Library Merge Function

The MERGE function copies the contents of one core image, relocatable, or source statement library to another core image, relocatable, or source statement library respectively. If the phase, module, or book being copied already exists in the library being updated, the resident phase, module, or book becomes inaccessible because its directory entry is deleted. Any phase, module, or book being copied is added to the library at the next available entry point. Its directory entry is added at the directory's next available entry point.

The control statement indicates the type of library (resident or private) involved and the direction in which the data will move. It has the following format:

```
MERGE from,to
```

The operation field contains MERGE. The operand field entry from represents the file from which data will be copied. from can be one of these:

RES	System residence file on SYSRES.
NRS	Modified, or duplicate system residence file on SYS002 (New ReSidence).
PRV	Private relocatable library on SYS001 and/or a private source statement library on SYS000.

The operand field entry to indicates the file to which library data will be transferred. to can be one of these:

RES	System residence file on SYSRES.
NRS	Modified, or duplicate system residence file on SYS002 (New ReSidence).
PRV	Private relocatable library on SYSRLB and/or private source statement library on SYSSLB.

The MERGE statement is followed by appropriate copy statements (COPYC, COPYR, COPYS) that indicate the phases, modules, or books to be transferred.

All copy statements following a MERGE statement apply to that function until another MERGE, NEWVOL, or ALLOC statement is encountered.

Note 1: If the COPYC ALL statement is used, the Supervisor and transient phases are also transferred to the receiving system resident file. The Supervisor and transients previously contained on the receiving SYSRES disk pack are deleted. No indication of this deletion is given, and it is the user's responsibility to ensure that the receiving system is able to continue operating.

Note 2: The lack of a NEWVOL or MERGE statement indicates the creation of a new system residence file.

Transfer of Library Data between Libraries

Library data can be transferred between libraries as shown in the sections Core Image Library, Relocatable Library, and Source Statement Library.

CORE IMAGE LIBRARY: Selected phases, or the entire library, can be transferred (in either direction) between the core image library of the system residence file on SYSRES and the core image library of another system residence file on SYS002. The user must ascertain that the receiving system has the ability to execute the phase(s) being copied.

RELOCATABLE LIBRARY: Selected modules, or the entire library, can be transferred (in either direction) between:

1. The relocatable library of the system residence file on SYSRES and the relocatable library of another system residence file on SYS002.
2. A private relocatable library and the relocatable library of the system residence file on SYSRES.
3. A private relocatable library and the relocatable library of a system residence file on SYS002.
4. Two private relocatable libraries.

SOURCE STATEMENT LIBRARY: Selected books, or the entire library, can be transferred (in either direction) between:

1. The source statement library of the system residence file on SYSRES and the source statement library of another system residence file on SYS002.
2. A private source statement library and the source statement library of the system residence file on SYSRES.
3. A private source statement library and the source statement library of a system residence file on SYS002.
4. Two private source statement libraries.

MERGE Considerations

File definitions (through DLBL and EXTENT statements) must be made before the MERGE control statement is used. When defining files, remember:

1. When merging to, or from, a modified or duplicate system residence file, the modified or duplicate file name must be IJSYSRS, the logical unit must be SYS002, and the file ID must be identical to the ID supplied when the file was created.

2. When merging to a private relocatable library file, the file name must be IJSYSRL, the logical unit must be SYSRLB, and the file ID must be identical to the ID supplied when the file was created.

3. When merging from a private relocatable library file, the file name must be IJSYSPR, the logical unit must be SYS001, and the file ID must be identical to the ID supplied when the file was created.

4. When merging to a private source statement library file, the file name must be IJSYSSL, the logical unit must be SYSSLB, and the file ID must be identical to the ID supplied when the file was created.

5. When merging from a private source statement library file, the file name must be IJSYSPS, the logical unit must be SYS000 and the file ID must be identical to the ID supplied when the file was created.

Figure 26 shows the file name, logical unit, and direction of transfer for each of the MERGE operations. Any combination of the indicated operations can be performed in one job step.

Diagnostic messages for erroneous assignments, file definitions, etc, are provided on SYSLST. Figure 27 is an example of a job set up to use the MERGE function.

		1	2	3	4	5
File Name / Logical Unit	IJSYSRS / SYSRES	IJSYSRS / SYS002	IJSYSRL / SYSRLB	IJSYSPR / SYS001	IJSYSSL / SYSSLB	IJSYSPS / SYS000
Merge RES to NRS	from	to				
Merge NRS to RES	to	from				
Merge RES to PRV	from		to		to	
Merge NRS to PRV		from	to		to	
Merge PRV to RES	to			from		from
Merge PRV to NRS		to		from		from
Merge PRV to PRV			to	from	to	from

●Figure 26. Direction of Transfer for Merge Operations

```
        Assume two disk drives with addresses of 190 and 191.

        // JOB EXAMPLE
        // ASSGN SYSRLB,X'191'
        // ASSGN SYSSLB,X'191'
        // ASSGN SYS000,X'190'
        // ASSGN SYS001,X'190'
        // ASSGN SYS002,X'191'

Note 1  // DLBL IJSYSRL,'PRIVATE RL',99/365
        // EXTENT SYSRLB,111111,1,0,1500,100

Note 2  // DLBL IJSYSSL,'PRIVATE SL',99/365
        // EXTENT SYSSLB,111111,1,0,1600,100

Note 3  // DLBL IJSYSPR,'PRIVATE RL TEST',99/365
        // EXTENT SYS001,111111,1,0,1300,100

Note 4  // DLBL IJSYSPS,'PRIVATE SL TEST',99/365
        // EXTENT SYS000,111111,1,0,1400,100

Note 5  // DLBL IJSYSRS,'SYSTEM RESIDENCE',99/365
        // EXTENT SYS002,111111,1,0,1,170

        // EXEC CORGZ

        NEWVOL RL=10(2),SL=10(2)
Note 6  COPYR ALL
        COPYS ALL

        MERGE PRV,PRV
Note 7  COPYR ALL
        COPYS ALL

        MERGE NRS,PRV
Note 8  COPYR ALL
        COPYS ALL

        /*
        /&
```

•Figure 27. Example of Job Set Up to Use the MERGE Function (Part 1 of 2)

Note 1: File definition statements for a private relocatable library file which will be created and updated.

Note 2: File definition statements for a private source statement library file which will be created and updated.

Note 3: File definition statements for a private relocatable library file from which modules will be copied.

Note 4: File definition statements for a private source statement library file from which books will be copied.

Note 5: File definition statements for a modified, or duplicate system residence file from which modules and books will be copied.

Note 6: Creates private relocatable and source statement libraries on SYSRLB and SYSSLB, and copies the relocatable and source statement libraries from the system residence file on SYSRES into them.

Note 7: Merges all modules and books from private relocatable and source statement libraries on SYS001 and SYS000 into the appropriate private libraries created on SYSRLB and SYSSLB.

Note 8: Merges all modules and books from the relocatable and source statement libraries of a modified, or duplicate system residence file on SYS002 into private libraries created on SYSRLB and SYSSLB.

Figure 27. Example of Job Set Up to Use the MERGE Function (Part 2 of 2)

Punch Service Function: Special Considerations

All librarian control statements are read from SYSIPT, rather than from SYSRDR as in previous versions of the system. This eliminates the problem of switching job streams from SYSRDR to SYSIPT and vice versa. Thus, the special librarian control statements IPTCTRL and RDRCTRL are no longer required. Existing job streams that include these statements must be changed.

The special librarian control statements are handled as follows.

IPTCTRL is detected by Job Control and is flagged as an invalid entry.

RDRCTRL is treated as /* (end of data file) and control given to Job Control.

Records on SYSIPT for the MAINT program can be either 80 or 81 characters in length, unless SYSIPT is assigned to an IBM 2314. (Records on a 2314 disk extent must be 80 characters in length.) Thus, punched output from RSERV and SSERV, when SYSPCH is assigned to a magnetic tape unit or an IBM 2311 disk extent, can be used as input to MAINT.

Condense Maintenance Function: Special Considerations

The condense maintenance function can be performed automatically at the end of a catalog or delete maintenance function under the control of an installation specified parameter. The parameter is stored in the system directory. It indicates that when the number of blocks available in the corresponding library is less than the number specified by the parameter, the condense function is performed for that library. The system interrogates the parameter at the completion of each maintenance function for the library. If a condense function is to be performed, a message is printed on the printer-keyboard (SYSLOG) to inform the operator that the library is to be condensed. If multiprogramming is in progress and the core image library should be condensed, the automatic condense function will be suppressed. (Multiprogramming has no effect on a

condense function for either the relocatable or source statement library.)

The CONDL control statement (as opposed to the CONDS control statement for a user-specified condense function) informs the MAINT librarian program that a parameter to specify an automatic condense is to be set. The CONDL control statement is in the following format.

```
CONDL lib=nnnnn[,lib=nnnnn[,lib=nnnnn]]
```

The entry in the operation field is CONDL. In the operand field, the entry lib is CL for the core image library, RL for the relocatable library, and SL for the source statement library. The entry nnnnn represents the number of blocks specified for the specific library and is from one to five decimal digits. The maximum value of nnnnn is 65536. Each track of the core image library contains 2 blocks on the 2311 (or 4 blocks on the 2314), each track of the relocatable library contains 9 blocks on the 2311 (or 16 blocks on the 2314), and each track of the source statement library contains 16 blocks on the 2311 (or 27 blocks on the 2314).

If 0 (zero) is specified for nnnnn, an automatic condense will not be performed for the specific library. If the number of blocks specified exceeds the number of blocks allocated for the library, a condense is performed each time deleted blocks appear in the library at the end of a maintenance function. When the system is copied onto another pack, the condense limit on SYSRES is also copied.

The automatic condense of the core image library is bypassed when a new supervisor is cataloged.

The condense limits will be displayed with the system status on a DSERV and at the end of a maintenance job.

The control statement input to establish a value for an automatic condense is the same as that for a user-specified condense. See any of the subsections entitled Condense for a description of the control statement input.

Private Libraries

Private libraries are desirable in the system to permit some libraries to be located on a disk pack other than the one used as SYSRES. The copy function defines and/or creates private libraries using the NEWVOL and COPY statements. The MERGE statement transfers modules and books to and from private libraries.

Private libraries are supported for the relocatable library and for the source statement library on both 2311 and 2314 disk storage. However, the following restrictions apply:

1. The private library must be on the same type of DASD as SYSRES.

2. Reference is made tc a private relocatable library only if SYSRLB is assigned. If SYSRLB is assigned, the system relocatable library cannot be changed.

3. Reference is made to a private source statement library only if SYSSLB is assigned. If SYSSLB is assigned, the system source statement library cannot be changed.

4. Private libraries cannct be reallocated.

5. Private source statement libraries are not supported under the DOS D Assembler, 10K variant.

An unlimited number of private libraries is possible. However, each must be distinguished by a unique file identification in the DLBL statement for the library. No more than one private relocatable and one private source statement library may be assigned at one time.

The only system service programs that can access the system libraries when SYSRLB and SYSSLB are assigned are the Linkage Editor and the two Librarian functions, copy and maintenance reallccate.

The following language components support private source statement libraries:

DOS 14K D Assembler
DOS F Assembler
COBOL-D.

These language components first search the private source statement library (SYSSLB), if assigned, until the desired book is found. If not found, or if no private source statement library is assigned, the system source statement library is searched until either the desired book is found or the end of the system source statement library is reached.

The DOS 10K D Assembler does not support private source statement libraries and searches only the system source statement library for the desired book.

CREATION OF PRIVATE LIBRARIES

The COPY program is used for the creation of private libraries. SYSRLB must be assigned if a private relocatable library is required. SYSSLB must be assigned if a private source statement library is required.

When creating a private library using CORGZ, a file must be defined for IJSYSRL on SYSRLB and/or IJSYSSL on SYSSLB. IJSYSRL and IJSYSSL are the respective file names used in the creation of private relocatable and source statement libraries.

Creation of a private library is requested by the NEWVOL librarian control statement. Its format follows.

```
NEWVOL id=cylin(track)[,id=cylin(track)]
```

id — Indicates the specific library and directory to be created and can be:

- RL for a private relocatable library and directory
- SL for a private source statement library and directory.

cylin — Indicates the number of cylinders that will contain the specified library.

track — Indicates the number of tracks that will contain the specified library directory. The tracks allocated to the directory are contained within the cylinders allocated to the library. The difference between the number of cylinders for the library and the number of tracks for the library directory has to be at least 5 tracks.

The COPY program also provides the ability to copy all or part of the system relocatable and/or system source statement library into its respective private library. If this facility is to be used, it must be employed in the same job step in which the private library is created. This is done by inserting COPYR and/or COPYS statements immediately behind the NEWVOL statement(s) in the job stream.

To define and create a private library from an existing private library, the MERGE statement must be used between the NEWVOL and the COPYR and/or COPYS statements. Below is an example of the sequence of steps.

```
// EXEC CORGZ
   NEWVOL RL=10(2),SL=10(1)
   MERGE PRV,PRV
   COPYR ALL
   COPYS ALL
/*
/&
```

The following precautions should be observed.

1. When a NEWVOL statement and a COPYR and/or a COPYS statement are both present, the NEWVOL statement must precede the COPYR or COPYS statement.
2. The creation of a new system residence file and creation of private libraries cannot both be done in the same job step.
3. For each job, label cards for the private libraries containing the same information as at creation time must be submitted.

MAINTENANCE AND SERVICE OF PRIVATE LIBRARIES

The following maintenance functions may be performed on a private library.

1. Catalog
2. Delete
3. Rename
4. Condense
5. Condense limit
6. Update

All private library maintenance and service functions are the same as system library functions. SYSRLB and SYSSLB must be assigned for the private relocatable library and for the private source statement library, respectively.

Private libraries must have been previously created, and must be unassigned in order to perform maintenance and service on system libraries.

Diagnostics are printed only on SYSLST, with the exception of disaster errors from the condense and allocation functions. Disaster messages also appear on SYSLOG.

Problem Determination

Problem determination is a process or procedure for determining the cause of an error. Some DOS facilities (such as I/O Error Logging, MCRR, and the DUMP option of job control) are problem determination tools. DOS Problem Determination recommends a specific procedure to follow when an error condition occurs. These recommendations are in the Operator Communications and Messages publication listed in the Preface.

Some programs recommended for error analysis are:

- Problem Determination Serviceability Aids (PDAID).
- Environment Record Editing and Printing Program (EREP).
- Error Statistics by Tape Volume Utility Programs (ESTVUT, ESTVFMT).

Another facility for problem determination is the DUMPGEN program. This program allows the user to generate a stand-alone program, tailored to his requirements, for displaying the contents of main storage when processing under DOS cannot continue.

Problem Determination Serviceability Aids (PDAID)

PDAIDs provide the option to trace one of the following specified events occurring during the operation of a program:

- Fetching or loading of other programs (F/L Trace).
- Input/Output activity (I/O Trace).
- Supervisor calls (SVCs); that is, communications between the control program and the problem program (GSVC trace).

Tracing consists of recording pertinent data when the event occurs. This data can be used for error analysis.

Tracing can be limited to recording the events of the problem program, the supervisor, or both.

Fetch/Load Trace

The Fetch/Load trace function records information about phases and transients as they are called from the core image library under the control of DOS. When a fetch or a load is issued, causing a SVC 1, 2, 3, or 4, the data recorded is:

- Location of the SVC.
- Program interrupt key.
- SVC number.
- Phase or transient name.
- Load address of the phase.
- Entry address of the phase.

At times, SVCs 5, 6, 11, and 14 branch directly into the supervisor fetch or load routine. The fetch or load (SVCs 1-4) is recorded. However, the calling address and the SVC values for SVCs 5, 6, 11, and 14 are not indicated in the actual fetch or load trace record.

GSVC Trace

The generalized SVC trace function records SVC interrupts as they occur. All SVCs, or a selected group of SVCs, can be traced. The data recorded by the SVC trace function is:

- SVC old PSW.
- Task identification.
- Last three bytes of register 0.
- Contents of register 1.

When PTO=YES in the FOPT macro at system generation time, the SVCs issued when the physical transient area is busy are not traced.

Input/Output Trace

The I/O trace function records I/O device activity. The data recorded by the I/O trace function can be for all I/O devices,

or for a selected group. When an I/O interrupt occurs the data recorded is:

- I/O old PSW.
- CSW.

When the CSW is stored in response to a SIO instruction issued by the DOS supervisor, the data recorded is:

- Device address.
- CCB address.
- CSW.

Initiation of PDAID

PDAIDs are executed by using standard DOS Job Control language from either SYSLOG or SYSIPT. The statement

```
// EXEC PDAID
```

causes the main phase, PDAID, to be loaded at the address of the initiating partition. Control is given to PDAID for further specifications to indicate the type of trace to be performed. PDAID issues the following message to the operator on SYSLOG:

```
4P10D PDAID=
```

The operator must respond to this message with one of the following:

FT Specifies a fetch/load trace is to be performed. (See Note.)

GT Specifies a GSVC trace is to be performed. (See Note.)

IT Specifies an I/O trace is to be performed. (See Note.)

XX Terminates the PDAID presently running.

Ⓑ Indicates PDAID control statements are to be entered through SYSIPT.

Note: When FT, GT, or IT are specified, additional PDAID control statements must be given by the operator through SYSLOG. Figure 28 shows the PDAID control statements in the sequence they must be used.

- The Ⓑ response is valid for SYSLOG and cannot be used as a SYSIPT operand.
- Multiple operands or operator responses to PDAID control statements for traces with a variable number of functions (such as ignoring SVCs) are not allowed. Repeat each parameter with each variable. Repeat each message until either the maximum number of variables is reached or a Ⓑ response is given.
- GO terminates the PDAID control statements. With SYSLOG input, GO is a valid response (Figure 28). With SYSIPT input, GO must be the last parameter for the PDAID control statements and must have no operand associated with it.

SYSLOG /SYSIPT Message/ Parameter	SYSLOG /SYSIPT Response/ Operand	Meaning	Default
PDAID=	{ FT, GT, IT, XX, Ⓑ }	FT Fetch/Load Trace GT GSVC Trace IT I/O Trace XX Terminate present PDAID function. Ⓑ Additional PDAID control input through SYSIPT.	None, the function continues.
OUTPUT DEVICE=	{ cuu, X'cuu', Ⓑ, GC }	Specify the hexadecimal channel and unit number of either a magnetic tape unit or a printer for the output device of the PDAID.	Core-Wrap mode.
TRACE PARTITION= (Valid for Fetch/ Load and GSVC Trace.)	[{ SP, BG, F2, F1, Ⓑ, GO }]	SP Supervisor BG Background F2 Foreground 2 F1 Foreground 1	Trace all partitions and the supervisor.
IGNORE DEVICE=	[{ cuu, X'cuu', Ⓑ, GO }]	Specify the hexadecimal channel and unit number of the device to be ignored by the I/O trace. A maximum of 3 may be specified.	Trace all devices.
TRACE DEVICE=	[{ cuu, X'cuu', Ⓑ, GO }]	Specify the hexadecimal channel and unit number of the device to be traced by the I/O trace. A maximum of 3 may be specified.	Trace all devices.
IGNORE SVC=	[{ nn, Ⓑ, GO }]	Specify the hexadecimal SVC number to be ignored by the GSVC trace. A maximum of 6 may be specified.	Trace all SVCs.
TRACE SVC=	[{ nn, Ⓑ, GO }]	Specify the hexadecimal SVC number to be traced by the GSVC trace. A maximum of 6 may be specified.	Trace all SVCs.
GO (Valid SYSIPT parameter)	GO (Valid SYSLOG response)	GO must be the last parameter for SYSIPT input.	None

●Figure 28. PDAID Control Statements

System Consideration for PDAIDs

The following must be performed before the execution of PDAIDs.

- During system generation, specify CE=800 in the FOPT macro of the installation tailored supervisor.

  Note: Up to 10,240 bytes can be specified to increase the size of the save area for the core-wrap mode.

- Catalog the main phase (PDAID) in either absolute or self-relocating form to the core image library before execution.

If data provided by PDAIDs is recorded on magnetic tape, use the PDLIST program to make the data readable. Thus, catalog either the absolute or self-relocating version of PDLIST to the core image library before execution.

CATALOGING PDAID AND PDLIST: All the PDAID programs are self-relocating for initialization in any partition (4K or greater) of a multiprogramming system. The main phase,PDAID, and PDLIST are in the relocatable library so they can be cataloged for the system in which they are used. All other phases are in the core image library.

The following job stream catalogs PDAID to the core image library for operation in a batched-job system:

```
// JOB
// OPTION CATAL
   PHASE PDAID,S
   INCLUDE IJBPDAID
/*
// EXEC LNKEDT
/&
```

The following job stream catalogs PDLIST to the core image library for operation in a batched-job system:

```
// JOB
// OPTION CATAL
   PHASE PDLIST,S
   INCLUDE IJBPDLST
/*
// EXEC LNKEDT
/&
```

The following job stream catalogs PDAID to the core image library for operation in a multiprogramming system:

```
// JOB
// OPTION CATAL
   PHASE PDAID,+0
   INCLUDE IJBPDLST
/*
// EXEC LNKEDT
/&
```

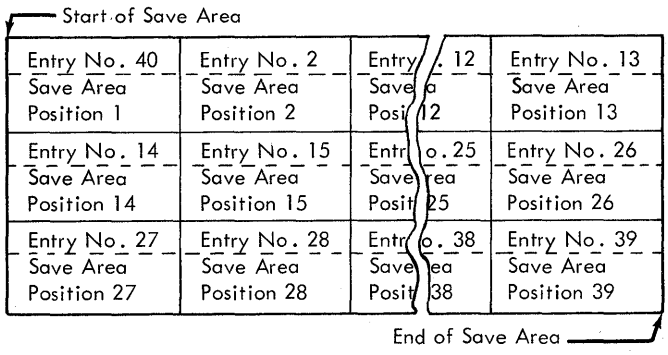
The following job stream catalogs PDLIST to the core image library for operation in a multiprogramming system:

```
// JOB
// OPTION CATAL
   PHASE PDLIST,+0
   INCLUDE IJBPDLST
/*
// EXEC LNKEDT
/&
```

CORE WRAP MODE: The core wrap mode is the default if no output device is selected. This process records the events in the area of main storage reserved by the CE parameter of the FOPT macro. If the core wrap mode is selected, the user must display the contents of main storage by either a dump program, or manually from the operating panel of the CPU.

When the core wrap mode is selected and CE=800, a maximum of 39 entries can be recorded for the Fetch/Load and I/O traces, and 32 entries for the GSVC Trace.

Figure 29 shows the method for recording Fetch/Load, I/O, and SVC events. When the maximum number of entries for the function is reached, new entries are entered by overlaying the original entries starting at the first position of save area.



●Figure 29. Core Wrap Method for Fetch/Load, I/O and GSVC Traces

Core Wrap Mode Data Location: To find the location of the data saved by core wrap mode, locate the CEAREA. The address of the CEAREA is +136 bytes (X'88') from the address of the communications region extension (BGXTNSN) and is a four byte address.

With the F/L, I/O, and GSVC traces, three pointers locate the trace data:

SLOT1 — Address of the beginning of the save area.

NEXT — Address where the next entry should be placed. NEXT contains unchecked new data, which is either the last entry of the save area or the most recent data of an event not being traced. When the most recent data is the case, ignore the entry.

WRAPADR — Address of the end of the save area.

The location of the pointers for the fetch/load trace is CEAREA + 176 (X'B0'), the location of the I/O trace pointers is CEAREA + 208 (X'C0'), and the location of the GSVC trace pointers is CEAREA + 228 (X'E4'). Figures 30, 31, and 32 show the formats for the entries of the save area.

When an F/L event occurs, the phase name, supervisor call, address of the supervisor call, program interrupt key, load address, and entry address are stored for each fetch or load, as follows:

Phase Name	SVC	Calling Address	PIK	Load Address	Entry Address
xxxxxxxxxxxxxxxx	xx	xxxxxx	xx	xxxxxx	xxxxxxxx

●Figure 30. F/L Trace Entry in Core Wrap Mode

When an I/O interrupt occurs, the I/O old PSW and the CSW are stored, as follows:

I/O Old PSW	CSW
xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx

When the CSW stored condition occurs after SIO, the device address, the CCB address, and the CSW are stored, as follows:

Channel Unit Address	CCB Address	CSW
00000xxx	xxxxxxxx	xxxxxxxxxxxxxxxx

The CSW stored condition can be detected by checking the PSW portion of an entry in the core-dump and determine if the system mask byte contains an X'00'. If the system mask byte is not X'00', the I/O event was an interruption.

●Figure 31. I/O Trace Entry for Core Wrap Mode

When a SVC interrupt occurs, the SVC old PSW, task ID, register 0 and register 1 are stored as follows:

SVC old PSW	Task ID	Last 3 Bytes of Reg 0	Register 1
xxxxxxxx	x	xxx	xxxx

PIK value

●Figure 32. GSVC Trace Entry for Core Wrap Mode

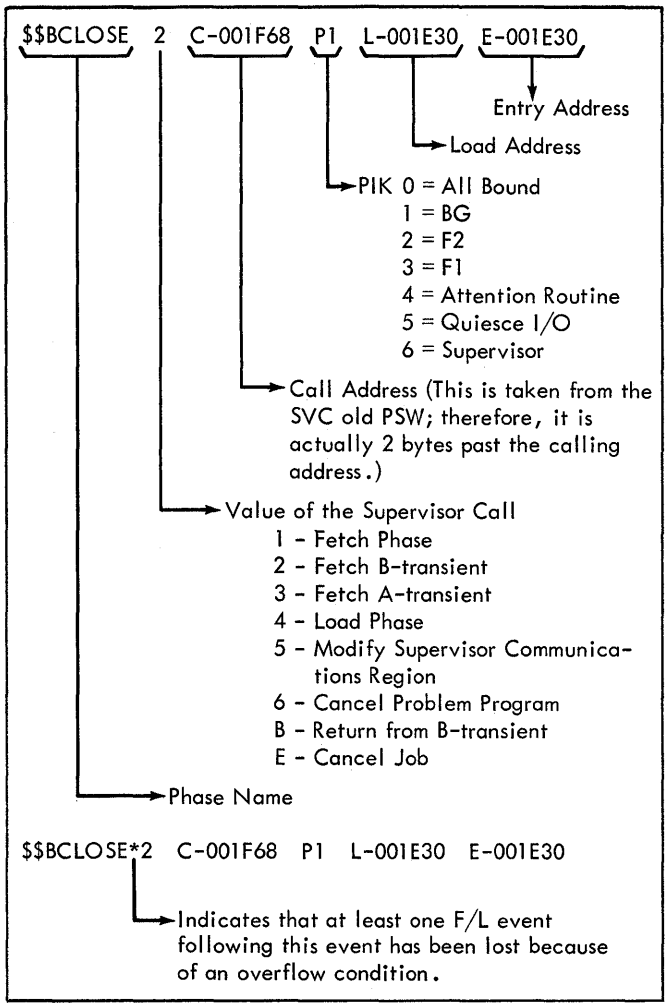
PDLIST is initiated by the command:

```
// EXEC PDLIST
```

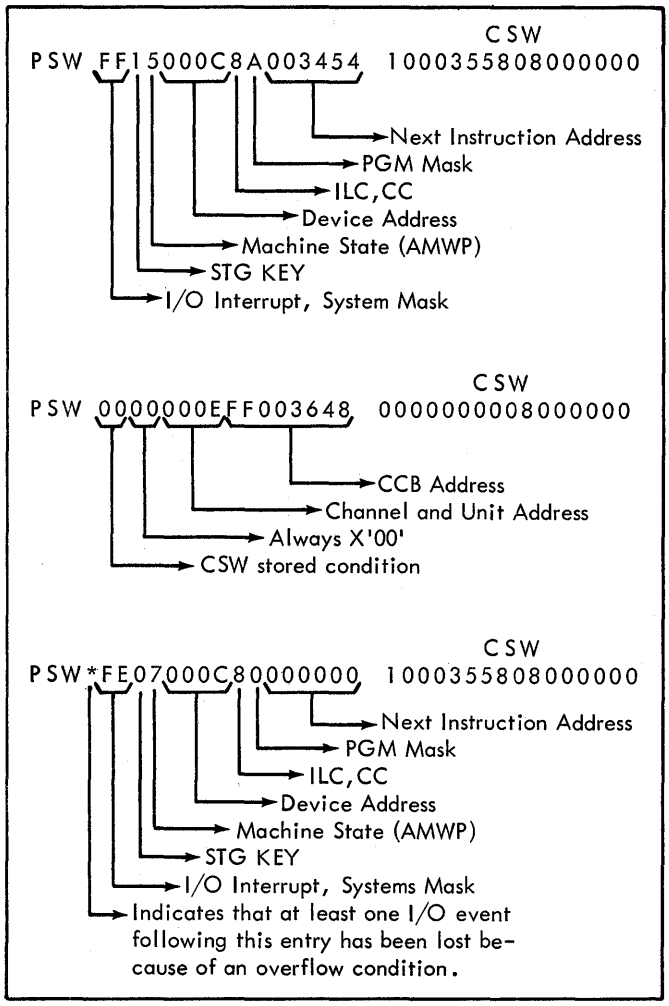
PDLIST then prints the contents of the tape reel mounted on SYS005 (which can be the output of more than one PDAID function) or SYSLST. See Figures 33, 34, and 35 for samples of the printed record for each PDAID.

Printing PDAID Data

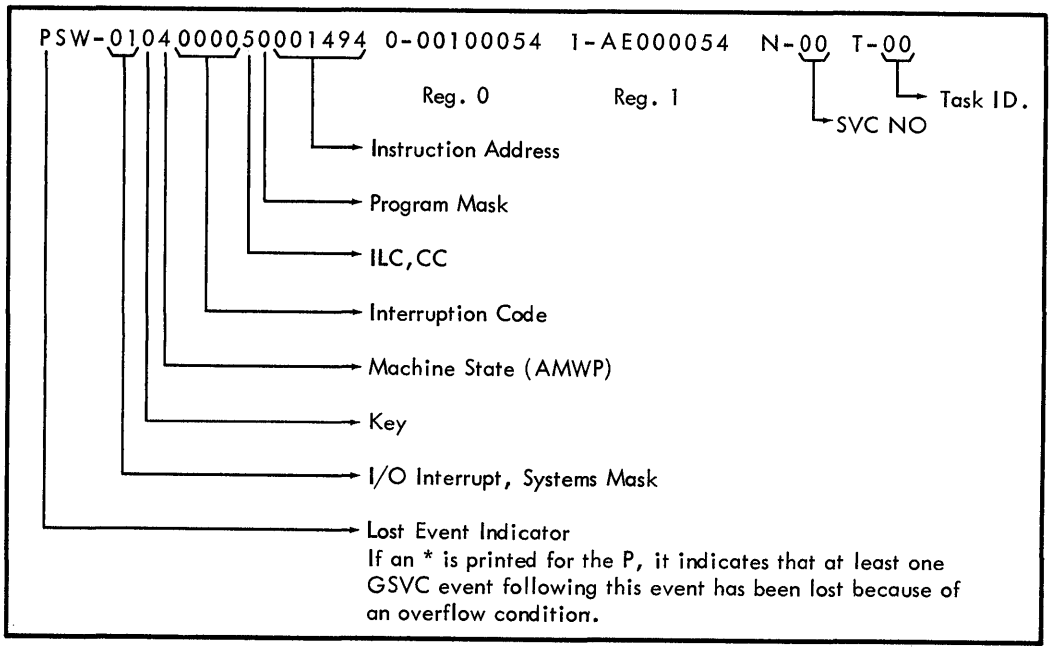
When the PDAID function uses a printer for its output device, or the PDLIST program prints the output of a tape unit, the data printed out is identical. However, if a tape unit is selected for the output device of a PDAID function, then use the PDLIST program to make the data readable.



●Figure 33. Sample Output for F/L Trace



●Figure 34. Sample Output for I/O Trace



●Figure 35. Sample Output for GSVC Trace

PDAID Usage

The following job stream is an example of how to execute PDAIDs from SYSIPT. For each job step, a different trace is performed.

```
// JOB
// EXEC PDAID
   PDAID=IT
   OUTPUT DEVICE=00E
   IGNORE DEVICE=X'180'
   GO
/*
// EXEC USRPROG1
/*
// EXEC PDAID
   PDAID=XX,GO
/*
// EXEC PDAID
   PDAID=FT,OUTPUT DEVICE=180,TRACE
   PARTITION=SP,GO
/*
// EXEC USRPROG2
/*
// EXEC PDAID
   PDAID=XX,GO
/*
// EXEC PDAID
   PDAID=GT,OUTPUT DEVICE=X'180',GO
/*
// EXEC USRPROG3
/*
// EXEC PDAID
   PDAID=XX,GO
/*
// ASSGN SYS005,X'180'
// MTC REW,SYS005
// EXEC PDLIST
/*
/&
```

Environmental Recording, Editing, and Printing Program (EREP)

The EREP program edits and prints data that has been stored in the recorder file by the I/O error logging and MCRR functions.

The EREP program is run as a 10K problem program using standard job control language. However, if IBM 2715 errors are encountered, only errors occurring on the first 60 area station/device combinations are summarized. If more than 60 combinations are present, errors are summarized by area station only, and only for a maximum of 60 area stations. If EREP is run in a 12K or larger partition, the limit becomes 100 area station/device combinations. Whenever the user needs the environment data, EREP can be executed from the printer-keyboard by typing a control statement in the following format:

```
// EXEC EREP
```

EREP does not require nor use any control statements or commands during its execution. The only devices used by EREP are SYSREC (input) and SYSLST (output). The user can link-edit EREP to run in any partition.

EREP displays SDR records first (in the order of their appearance on disk) followed by OBR records (grouped by device address). Then channel inboard records are displayed, followed by CPU machine check records. Each of these groups is displayed on a first-in first-out basis. The record counters are reset to zero after each SDR record is processed. OBR and MCRR records are reset by EREP only if CLEAR is specified in the I/O error logging or MCRR operands at system generation time. None of these records is reset if an error occurs while EREP is processing any OBR or MCRR record. Information in the recorder file is eliminated by rebuilding the recorder file. While the EREP program is operating, recording to the recorder file is suppressed.

EREP displays IBM 2715 error records from the SYSREC file in the order of their appearance within each type, such as: disk adapter error, 2790 adapter error, MPX adapter error, etc. If area station errors are recorded, they are written to SYSLST both in order of occurrence and in summary (by area station/device).

If the EREP program is canceled for any reason:

- SDR records printed are reset to zero. Those not printed are not reset. Execute EREP again to print them.
- If the cancel occurs before all OBR, MCRR, and IBM 2715 transmission control error records are printed, execute EREP again to print them.

Error Statistics by Tape Volume Utility Programs

ESTV Dump File Program (ESTVUT)

When DOS is generated, the user has the option of requesting the collection of error statistics by tape volume (ESTV). This ESTV data can be printed on the system console, or stored on a direct access storage device. If it is stored, a dump file program must be used to process the data from the disk. The ESTV Dump File Program (ESTVUT) is for this purpose.

ESTVUT PROCESSING FUNCTIONS: ESTVUT gives the user five options to process the error statistics collected and stored on disk by the ESTV program. The system operator specifies the processing method at the start of execution of the ESTVUT program. To do this, he responds to messages sent to him by the program. The five options are:

1. ESTVUT dumps the data from the disk file to a printer and clears the disk file.
2. ESTVUT dumps the data from the disk file to a printer and leaves the disk file as it was. More records can be added to the disk and a later dump taken of the entire file, including the added records.
3. ESTVUT dumps the disk file to a magnetic tape and clears the disk file. This option includes dumping any statistics from a previous tape (obtained by this processing method) and then dumping the new data from the disk file to the new tape. This collects all error statistics on one tape.
4. ESTVUT dumps the collective tape file resulting from option 3 back to the original tape. This allows the user to keep his error volume statistics on a particular tape volume rather than on a new tape each time the file is dumped.
5. ESTVUT dumps the tape file that results from either option 3 or 4 to a printer.

CONTROL CARDS NECESSARY TO RUN ESTVUT: ESTVUT can be executed from either a card reader or from the console typewriter. An operator must be at the console, however, to answer system inquiries as described in Running ESTVUT. An example of the job control statements (either on cards or typed on the console) required for executing any of the ESTVUT options is:

```
// JOB ESTVDUMP
// ASSGN SYS004,X'191'
// ASSGN SYS005,X'183'
// ASSGN SYS006,X'184'
// ASSGN SYSLST,X'00E'
// TLBL TAPEIN
// TLBL TAPEOUT,'VOLUME STATISTICS',70/365
// LBLTYP TAPE
// EXEC ESTVUT
/*
/&
```

These job control statements permit execution of any of the Dump File program options. When a disk-to-printer option is used, the LBLTYP and TLBL statements and the ASSGN statements for tape drives are not required. When a tape-to-printer option is used, the DLBL, EXTENT, and disk

ASSGN statements are not required. However, the entire set of job control statements can be entered for any execution. Only the information required by the particular option chosen is used.

SYMBOLIC UNIT ASSIGNMENTS: The addresses of the specific input/output devices used by ESTVUT must be assigned to symbolic unit names (SYSxxx). Do this with ASSGN statements at the time the job is executed, or with permanent assignments that are made before this run. For example, SYSLST may have been permanently assigned to the printer. If this is the case, an ASSGN for SYSLST does not have to be included in the job control for ESTVUT.

Tape Assignments: The xxx of the symbolic unit names for tape files must be 005 for input tape and 006 for output tape.

Disk Assignment: For the disk file, the xxx of the symbolic unit name must ordinarily be 004, although a different logical unit may be used if the EXTENT information is in agreement. The logical unit must be assigned to the same device to which SYSREC is assigned.

LABEL INFORMATION: Before ESTVUT can be executed, label information for the disk file must be included as standard label information. Label information for any tape files to be processed can be included either as standard label information or with the job as temporary label information.

Disk File: The first operand of the disk file DLBL statement must be ESTVFLE.

The second operand, the file identifications, is optional.

Tape Files: The first operand of an input tape TLBL must be TAPEIN. TAPEOUT is the required operand for output tape file TLBLs.

The second operand is selected by the user.

For additional information about the operands that can be included, and their purposes, see Job Control Statements.

A LBLTYP for tape is required only if the program is executed in the background partition, uses tapes, and has been cataloged as self-relocating (+0 on the PHASE card). This statement reserves space for processing standard label information.

RUNNING ESTVUT: When the ESTVUT program begins execution, a message is sent to the console to notify the operator that the program has begun. The text of the message is:

```
4R01I  * ESTV DUMP UTILITY *
```

This is an information message only and requires no response.

Immediately following this notification, another message is printed at the console. It requests the operator to specify the form of input being used for the current run of the program. The text of the message is:

```
4R02A  INPUT=
```

The operator must respond to this message by keying in the designation of the input file. This must be 2311, 2314, or TAPE. In the event that a mistake is made in replying to this message, the system prints another message to the operator reminding him of the only valid responses to the query for input file designation. The text of this message is:

```
4R03I  INCORRECT INPUT - OPTIONS ARE
         2311, 2314, TAPE
```

The system then prints the request for input form again. The operator types the appropriate reply.

After the input form is accepted by the program, the operator is asked to specify the output method. The text of this message is:

```
4R04A  OUTPUT=
```

The operator has a choice of three responses: PRC, PRNC, and TAPE. If the error statistics are to be sent to a printer and the file is to be cleared, the response must be PRC. If the printer is to be the output device and the file is to remain as it is, the reply must be PRNC. The operator types TAPE when the output is to go to tape. In the event of an error in entering the output method, the operator is sent another message specifying the only valid responses. The text of this message is:

```
4R05I  INCORRECT OUTPUT - OPTIONS
         ARE PRC, PRNC, TAPE
```

The system then prints the request for output method again. The operator types the appropriate reply.

> Note: When dumping the disk to a printer, both PRC and PRNC are valid responses to the request for output method. The option to clear is not applicable when the tape is dumped to a printer. The tape remains as it was before execution of the program. Every time the disk file is dumped to tape the file is cleared.

When the program completes its run, a message is printed on the console to indicate to the operator that the program has completed execution. When the disk file is the input device and the output method specifies clearing the file to zeros, the text of the message is:

```
4R06I  FILE DUMPED AND CLEARED
```

In all other cases, including the tape-to-printer option, the text of the message is:

```
4R07I  FILE DUMPED
```

DUMPING FROM DISK TO TAPE: The option that dumps from disk to tape allows the user to keep one file on tape for volume statistics and to update that file each time the disk file is dumped. Two standard label tapes are used for dumping from disk to tape. One of the tapes is an input tape that can have volume statistics on it from a previous ESTVUT execution. The second is an output tape to which is dumped any previous statistics on the input tape and then the statistics from the disk file.

If this is the first time the disk file is dumped to tape, statistics are dumped to the output tape. A labeled tape that does not contain previous volume statistics must be mounted as the input tape. This tape is necessary, though ESTV does not dump statistics to or from this tape on this run through the program. The volume statistics are written from the disk to the output tape.

The user who wishes to keep all of his error data on one tape then makes this tape (which has just collected the statistics from disk, or any tape containing previously collected error statistics) the input tape. Any tape not previously used to collect error statistics is mounted as the output tape. The standard header label information for the input tape reel is printed on the console so the operator can verify that the correct tape is mounted as the input tape. The program first copies the input tape to the output tape, and then the disk file is dumped to the output tape.

At this point, the output tape contains the statistics from the input tape plus the statistics from the disk file. The following message, asking the operator if the output tape is to be dumped back to the input tape, is now printed at the console.

```
4R09D  DUMP OUTPUT TAPE BACK TO
         INPUT TAPE?(YES,NO)
```

If the operator replies NO, the job ends and the complete volume statistics are on a new tape. If the operator replies YES, the output tape is dumped back to the input tape, which can now be used as an updated master tape. The other tape can be used as a backup tape, or as an intermediate work tape.

The operator may choose to dump the output tape to another tape at some other time. To do this, he executes ESTVUT and replies TAPE for both input and output devices. In this case, the input tape (TAPEIN) is dumped to the output tape (TAPEOUT).

CONTENTS AND FORMAT OF PRINTED OUTPUT: When the operator specifies a printer as the output device, the collected error statistics are formatted and printed as indicated in ESTV Output Modes: Mode 1. Each page of output contains 50 lines of data. The last page of data printed has one of three messages printed below the last line of data. The message printed depends on which option of the ESTVUT program is executed. The alternative messages are:

```
ESTV DISK FILE DUMPED BUT NOT
        CLEARED TO ZEROS

ESTV DISK FILE DUMPED AND
        CLEARED TO ZEROS

ESTV TAPE FILE DUMPED
```

CATALOGING ESTVUT: ESTVUT is made up of three modules that must be cataloged into the core image library. They are the main routine (ESTVUT), the printer routine (ESTVPR), and the magnetic tape routine (ESTVMT). The module names that must be used in the INCLUDE cards for these routines are as follows.

ESTVUT - Use IJBESTUT

ESTVPR - Use IJBESTPR

ESTVMT - Use IJBESTMT

An example of the job control required to catalog the modules into the core image library follows.

```
// JOB CATALOG
// OPTION CATAL
   PHASE ESTVUT,+0
   INCLUDE IJBESTUT
   PHASE ESTVPR,+0
   INCLUDE IJBESTPR
   PHASE ESTVMT,+0
   INCLUDE IJBESTMT
// EXEC LNKEDT
/&
```

The PHASE cards must have +0 displacement as the origin point for the program to be self-relocating. Only the modules required for the desired option need be cataloged.

IJBESTUT is required for all options.

IJBESTPR is required if output is to the printer.

IJBESTMT is required if output is to a tape.

Batched-Job System Cataloging: Batched-job systems cannot recognize self-relocating programs. To make the ESTVUT modules usable in these systems, the PHASE cards of the job control statements must have an S in place of the +0. An example of the job control statements required for batch-only systems follows.

```
// JOB CATALOG
// OPTION CATAL
   PHASE ESTVUT,S
   INCLUDE IJBESTUT
   PHASE ESTVPR,S
   INCLUDE IJBESTPR
   PHASE ESTVMT,S
   INCLUDE IJBESTMT
// LBLTYP TAPE
// EXEC LNKEDT
/&
```

ESTV Format Data Set Program (ESTVFMT)

A system control program is available which must be the first program executed after the first initial program load (IPL) after the system is generated, if error statistics by tape volume are to be collected by the system on a disk. It must also be executed whenever new label information is entered into the system for the file. This is required in order to update information in the volume table of contents (VTOC). This program, ESTVFMT, opens the ESTV data set (ESTVFLE) on the disk file, enabling it to collect this system output, by putting the label information in the disk's volume table of contents. The data set must be on SYSREC.

The following steps regarding the ESTVFMT program must be performed to prepare the disk file to receive the collected statistics:

1. Specify TEBV=DASD at system generation time, and include ESTVFLE definition statements in the standard label section of the label cylinder on SYSRES. Ordinarily, SYSREC and SYS004 must be assigned to the same device. However, the user is not restricted to SYS004, but may select another logical unit if the EXTENT information is in agreement, and if SYSREC and the logical unit assignments are to the same device. The label information for ESTVFLE must be included as standard label information to make the file accessible to the ESTV programs. This is done with the following statements.

   ```
   // OPTION STDLABEL

   // DLBL ESTVFLE,'ERROR STAT BY TAPE
        VOLUME',99/365,SD

   // EXTENT
        SYS004,nnnnnn,1,1,nnnn,nnnn
   ```

2. Catalog ESTVFMT into the core image library with the name LJBESTFM. This is done with the following statements.

   ```
   // JOB CATALOG
   // OPTION CATAL
      INCLUDE IJBESTFM
   // EXEC LNKEDT
   /&
   ```

3. Execute ESTVFMT immediately after the first IPL after system generation, or whenever new label information is entered for the file. The operator does this at the console keyboard by typing:

   ```
   // EXEC ESTVFMT
   ```

When ESTVFMT completes its functions, a message is printed at the keyboard. The text of this information only message is:

```
4R00I  ESTV FILE INITLZD, ENTRY IN
       VTOC
```

No response is required to this message.

Stand-Alone Dump (DUMPGEN)

DUMPGEN produces a stand-alone dump program tailored to system requirements. This stand-alone dump can display the contents of main storage from a minimum of 8K bytes to a maximum of 16384K bytes.

When the stand-alone dump is executed, the original contents of bytes 0-23 and the 504 bytes required by stand-alone are destroyed. If information in either of these areas is needed, manually display it on the operational panel of the CPU and record it.

Executing DUMPGEN

DUMPGEN is in the relocatable library and must be cataloged to the core image library before it can be executed. The following job stream catalogs DUMPGEN to the core image library:

```
// JOB
// OPTION CATAL
   INCLUDE IJBDMPGN
// EXEC LNKEDT
/*
/&
```

DUMPGEN must be executed in the background partition by the command:

```
// EXEC DUMPGEN
```

DUMPGEN then reads its control statements from SYSIPT. Code the two types of control statements, ASSGN or OPTN, as follows:

1. The first 9 columns must be blank.
2. The operation (ASSGN or OPTN) must start in column 10.
3. The operand(s) must start in column 16.

ASSGN Statement: ASSGN defines the output device for the stand-alone dump. The format of the ASSGN statement is:

Operation	Operands
ASSGN	SYSLST,X'cuu'

SYSLST Is the only valid logical unit assignment for stand-alone dump.

x'cuu' Must define the address of the SYSLST printer. If the ASSGN statement is omitted, the X'00E' is assumed.

OPTN Statement: OPTN defines the upper limit of main storage to be displayed, the type of printer control, number of card decks, and the load address for the stand-alone dump. The format of the OPTN statement is:

Operation	Operand
OPTN	CORE=nnnnnK NOINTR INTR DECKS=nnnnnnnn LOADADR= D'nnnnnnnn' X'xxxxxx'

CORE Defines the area of main storage to be displayed by the stand-alone dump. nnnnnK can be any number from 8K to 16384K in increments of 2K. An odd specification (for example, 9K) is rounded high to the next even number (in the example, 10K is assumed).

NOINTR Produces a stand-alone dump program which, when loaded, prints out the contents of main storage on the SYSLST printer.

INTR Produces a stand-alone dump program which, when loaded, enters the WAIT state. Either press the INTERRUPT button on the CPU operating panel to print the output on X'00E', or first press the STOP button and then the START button of the printer desired for the output device.

DECKS Specifies the number of card decks desired for the stand-alone dump program. nnnnnnnn may be any

decimal number from 1 to 99,999,999. A blank card separates each deck produced. If DECKS is omitted, then 1 deck is produced.

LOADADR Specifies the load address of stand-alone dump. Any valid main storage address of the CPU for which the program is intended can be entered from 128 to 16,766,712. However, if the load address is not aligned to a double word, then the specified address is rounded high to the next double word. The specified address is checked for validity.

The control statements may be specified in any order or amount. However, the following rules apply:

1. The last statement of a duplicate operation that is processed overrides all previous statements of the same operation with similar operands (that is, if NOINTR is preceded by INTR, then the NOINTR functions are provided in the stand-alone dump program that is generated).

   NOTE: CORE and LOADADR are considered similar functions. Thus, the one that is processed last determines the amount of main storage to be displayed and the load address of the Stand-Alone Dump.

2. Decimal operands can contain leading zeros.

3. A dump program using the OPTN LOADADR statement displays the entire main storage, while the OPTN CORE statement displays the entire main storage up to the beginning of the dump program.

4. Only one operation per control statement is allowed.

5. One or more blanks must follow the operand if comments are desired.

6. DUMPGEN requires either the OPTN CORE or LOADADR statements to get the address where the dump is to be loaded. All other statements may be omitted, and, if so, DUMPGEN produces one stand-alone dump card deck with the NOINTR option and a printer assignment of X'00E'.

DUMPGEN Messages

Messages are issued when an error is made in the control statements for DUMPGEN. All errors can be corrected through SYSLOG. The messages are discussed in the Operator Communications and Messages publication listed in the Preface.

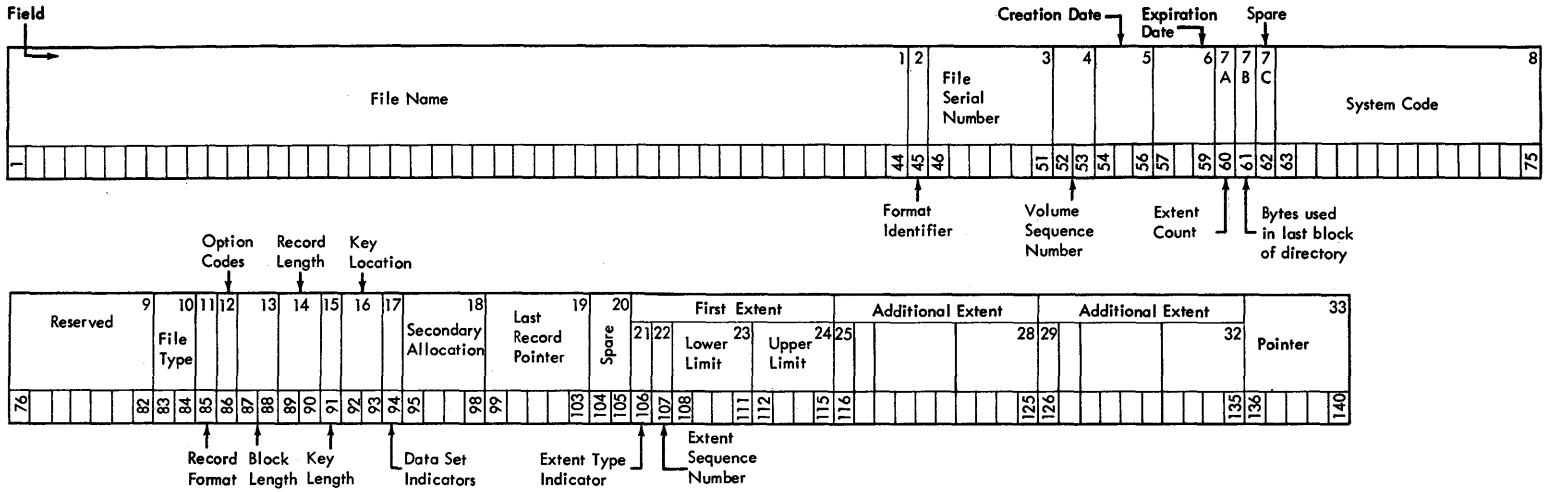


Figure 36. Standard DASD File Labels, Format 1 (Part 1 of 3)

Format 1: This format is common to all data files on Direct Access Storage Devices.

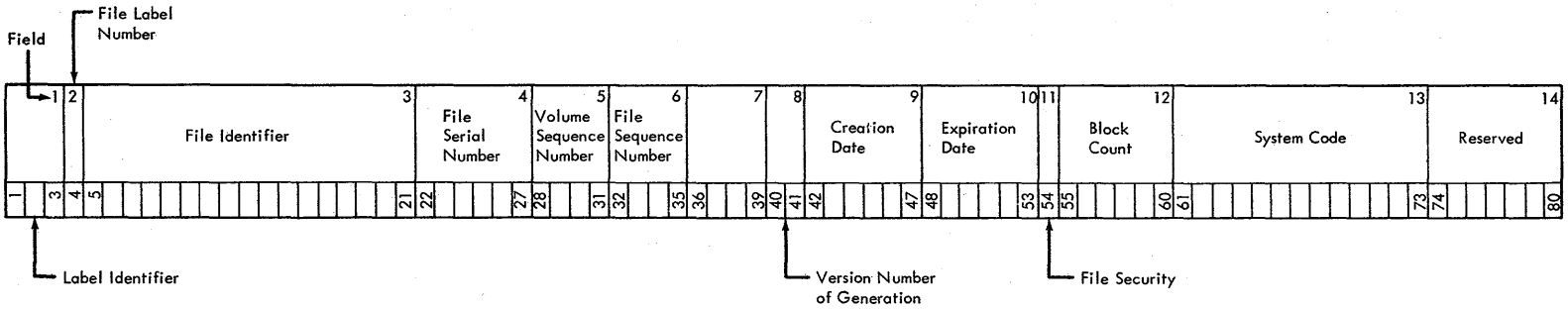
FIELD	NAME AND LENGTH	DESCRIPTION
1.	FILE NAME 44 bytes, alphameric EBCDIC	This field serves as the key portion of the file label. Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections: 1. File ID is an alphameric name assigned by the user and identifies the file. Can be 1-35 bytes if generation and version numbers are used, or 1-44 bytes if they are not used. 2. Generation Number. If used, this field is separated from File ID by a period. It has the format Gnnnn, where G identifies the field as the generation number and nnnn (in decimal) identifies the generation of the file. 3. Version Number of Generation. If used, this section immediately follows the generation number and has the format Vnn, where V identifies the field as the version of generation number and nn (in decimal) identifies the version of generation of the file.
		Note: The Disk Operating System compares the entire field against the file name given in the DLAB card. The generation and version numbers are treated differently by Operating System/360.
		The remaining fields comprise the DATA portion of the file label:
2.	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	1 = Format 1
3.	FILE SERIAL NUMBER 6 bytes, alphameric EBCDIC	Uniquely identifies a file/volume relationship. It is identical to the Volume Serial Number of the first or only volume of a multivolume file.
4.	VOLUME SEQUENCE NUMBER 2 bytes, binary	Indicates the order of a volume relative to the first volume on which the data file resides.
5.	CREATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file was created. It is of the form YDD, where Y signifies the year (0-99) and DD the day of the year (1-366).

FIELD	NAME AND LENGTH	DESCRIPTION
6.	EXPIRATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file may be deleted. The form of this field is identical to that of Field 5.
7A.	EXTENT COUNT 1 byte	Contains a count of the number of extents for this file on this volume. If user labels are used, the count does not include the user label track. This field is maintained by the Disk Operating System programs.
7B.	BYTES USED IN LAST BLOCK OF DIRECTORY 1 byte, binary	Used by O/S.
7C.	SPARE 1 byte	Reserved
8.	SYSTEM CODE 13 bytes	Uniquely identifies the programming system. The character codes that can be used in this field are limited to 0-9, A-Z, or blanks.
9.	RESERVED 7 bytes	Reserved
10.	FILE TYPE 2 bytes	The contents of this field uniquely identify the type of data file: Hex 4000 = Consecutive organization Hex 2000 = Direct-access organization Hex 8000 = Indexed-sequential organization Hex 0200 = Library organization Hex 0000 = Organization not defined in the file label.
11.	RECORD FORMAT 1 byte	Used by O/S.
12.	OPTION CODES 1 byte	Bits within this field indicate various options used in building the file. Bit 0 = 0 1 = Reserved 2 = Master index present (ISFMS) 3 = Independent overflow present (ISFMS) 4 = Cylinder overflow present (ISFMS) 5 = Reserved 6 Used by O/S. 7 Used by O/S.
13.	BLOCK LENGTH 2 bytes, binary	Indicates the block length for fixed length records or maximum block size for variable length blocks.
14.	RECORD LENGTH 2 bytes, binary	Indicates the record length for fixed length records or the maximum record length for variable length records.
15.	KEY LENGTH 1 byte, binary	Indicates the length of the key portion of the data records in the file.
16.	KEY LOCATION 2 bytes, binary	Indicates the high order position of the data record.
17.	DATA SET INDICATORS 1 byte	Bits within this field are used to indicate the following: Bit 0 If on, indicates that this is the last volume on which this file normally resides. 1-7: 0 for DOS Used by O/S.
18.	SECONDARY ALLOCATION 4 bytes, binary	Used by O/S.
19.	LAST RECORD POINTER 5 bytes, discontinuous binary	Used by O/S.

Figure 36. Standard DASD File Labels, Format 1 (Part 2 of 3)

Figure 36. Standard DASD File Labels, Format 1 (Part 3 of 3)

FIELD	NAME AND LENGTH	DESCRIPTION
20.	SPARE 2 bytes	Reserved
21.	EXTENT TYPE INDICATOR 1 byte	Indicates the type of extent with which the following fields are associated: HEX CODE 00 Next three fields do not indicate any extent. 01 Prime data area (Indexed Sequential); or Consecutive area, etc., (i.e., the extent containing the user's data records.) 02 Overflow area of an Indexed Sequential file. 04 Cylinder index or master index area of an Indexed Sequential file. 40 User label track area. 80 Shared cylinder indicator.
22.	EXTENT SEQUENCE NUMBER 1 byte, binary	Indicates the extent sequence in a multi-extent file.
23.	LOWER LIMIT 4 bytes, discontinuous binary	The cylinder and the track address specifying the starting point (lower limit) of this extent component. This field has the format CCHH.
24.	UPPER LIMIT 4 bytes	The cylinder and the track address specifying the ending point (upper limit) of this extent component. This field has the format CCHH.
25-28.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
29-32.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
33.	POINTER TO NEXT FILE LABEL WITHIN THIS LABEL SET 5 bytes, discontinuous binary	The address (format CCHHR) of a continuation label if needed to further describe the file. If field 10 indicates Indexed Sequential organization, this field points to a Format 2 file label within this label set. Otherwise, it points to a Format 3 file label, and then only if the file contains more than three extent segments. This field contains all binary zeros if no additional file label is pointed to.

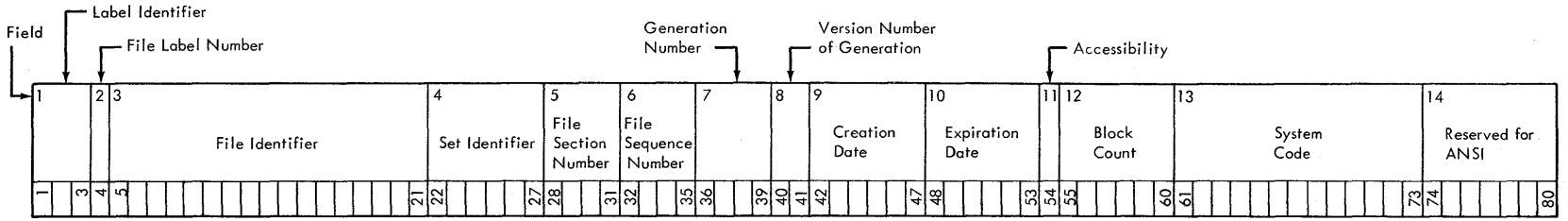


●Figure 37. IBM Standard Tape File Label

The IBM standard tape file label format and contents are as follows:

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, EBCDIC	identifies the type of label HDR = Header -- beginning of a data file EOF = End of File -- end of a set of data EOV = End of Volume -- end of the physical reel
2.	FILE LABEL NUMBER 1 byte, EBCDIC	always a 1
3.	FILE IDENTIFIER 17 bytes, EBCDIC	uniquely identifies the entire file, may contain only printable characters.
4.	FILE SERIAL NUMBER 6 bytes, EBCDIC	uniquely identifies a file/volume relationship. This field is identical to the Volume Serial Number in the volume label of the first or only volume of a multi-volume file or a multi-file set. This field will normally be numeric (000001 to 999999) but may contain any six alphameric characters.
5.	VOLUME SEQUENCE NUMBER 4 bytes	indicates the order of a volume in a given file or multi-file set. This number must be numeric (0000-9999). Multiple volumes of an output file will be numbered in consecutive sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	assigns numeric sequence to a file within a multi-file set.
7.	GENERATION NUMBER 4 bytes	numerically identifies the various editions of the file
8.	VERSION NUMBER OF GENERATION 2 bytes	idicates the version of a generation of a file
9.	CREATION DATE 6 bytes	indicates the year and the day of the year that the file was created: Position / Code / Meaning 1 / blank / none 2-3 / 00-99 / Year 4-6 / 001-366 / Day of Year (e.g., January 31, 1965 would be entered as 65031).
10.	EXPIRATION DATE 6 bytes	indicates the year and the day of the year when the file may become a scratch tape. The format of this field is identical to Field 9. On a multi-file reel, processed sequentially, all files are considered to expire on the same day.
11.	FILE SECURITY 1 byte	indicates security status of the file. 0 = no security protection 1 = security protection. Additional identification of the file is required before it can be processed.
12.	BLOCK COUNT 6 bytes	indicates the number of data blocks written on the file from the last header label to the first trailer label, exclusive of tape marks. Count does not include checkpoint records. This field is used in trailer labels.
13.	SYSTEM CODE 13 bytes	uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved. Should be recorded as blanks.

●Figure 38. ANSI (American National Standards Institute, Inc.) Standard Tape File Label



The ANSI standard tape file label format and contents are as follows:

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, ASCII	Identifies the type of label. HDR = Header -- beginning of a data file. EOF = End of File -- end of a set of data. EOV = End of Volume -- end of the physical reel.
2.	FILE LABEL NUMBER 1 byte, ASCII	Indicates the sequence of this label within a label group (HDR, EOF, EOV). DOS supports File Label 1 only and ignores subsequent numbers..
3.	FILE IDENTIFIER 17 bytes, ASCII	Identifies the entire file. May be any character except a quote (').
4.	SET IDENTIFIER 6 bytes, ASCII	Identifies the volume/file relationship. Generally, this field is identical to the volume serial number from the VOL label or the first or only volume of the logical file.
5.	FILE SECTION NUMBER 4 bytes	Indicates the order of a volume in a given file or multifile set. (The first file must be numbered 0001.)
6.	FILE SEQUENCE NUMBER 4 bytes	Assigns numeric sequence to a file within a multifile set. (The first file must be numbered 0001.)
7.	GENERATION NUMBER 4 bytes	Numerically identifies this edition of the file. (Must be numerical or blank.)
8.	VERSION NUMBER OF GENERATION 2 bytes	Indicates this version of the generation in field 7. (Must be numerical or blank.)
9.	CREATION DATE 6 bytes	Indicates the year and the day of the year that this file was created (bbyyddd) where b = blank yy = year (00-99) ddd = day (001-366)
10.	EXPIRATION DATE 6 bytes	Indicates the year and the day of the year that this file may become a scratch tape. Same format as above (Field 9).
11.	ACCESSIBILITY 1 byte	Indicates the accessibility protection of the file. Space = no accessibility protection. Nonspace = accessibility protection.
12.	BLOCK COUNT 6 bytes	Indicates the number of data blocks (physical records) written for this logical file.
13.	SYSTEM CODE 13 bytes	Name of programming system.
14.	RESERVED 7 bytes	Reserved for future use a required by ANSI (American National Standards Institute, Inc.). Should be recorded as spaces.

Appendix C: Operator-to-System Commands

<table>
<tr><th>Operation</th><th>Operand</th><th>Remarks</th></tr>
<tr><td>ALLOC</td><td>{F1=nK[,F2=nK]}
{F2=nK[,F1=nK]}</td><td>Allocates foreground program areas
Value of n is an even number</td></tr>
<tr><td>ASSGN</td><td>SYSxxx,address [{,X'ss'} [,TEMP]
{,ALT }]</td><td>SYSxxx: can be SYSRDR
SYSIPT
SYSIN
SYSLST
SYSPCH
SYSOUT
SYSLOG
SYSLNK
SYSRLB
SYSSLB
SYSREC
SYS000-SYSmax
address: can be X'cuu', UA, or IGN

X'cuu': c=0-6
uu=00-FE(0-254) in hex

UA: unassign

IGN: unassign and ignore

X'ss': used for magnetic tape only

<table>
<tr><th>ss</th><th>Bytes per Inch</th><th>Parity</th><th>Trans-late Feature</th><th>Convert Feature</th></tr>
<tr><td>10</td><td>200</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>20</td><td>200</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>28</td><td>200</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>30</td><td>200</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>38</td><td>200</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>50</td><td>556</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>60</td><td>556</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>68</td><td>556</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>70</td><td>556</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>78</td><td>556</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>90</td><td>800</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>A0</td><td>800</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>A8</td><td>800</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>B0</td><td>800</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>B8</td><td>800</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>C0</td><td>800</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">dual density 9 track tape</td></tr>
<tr><td>C8</td><td>800</td><td colspan="3">dual density 9 track tape</td></tr>
</table>

ALT: specifies alternate unit

TEMP: assignment for logical unit will be destroyed by next JOB statement</td></tr>
<tr><td>CANCEL</td><td>blank</td><td>Cancels execution of current job</td></tr>
<tr><td>CLOSE</td><td>SYSxxx [{,X'cuu'[,X'ss']}
{,UA }
{,IGN }
{,ALT }]</td><td>SYSxxx: for 2311 or 2314 - SYSIN
SYSRDR
SYSIPT
SYSPCH
SYSLST

for magnetic tape -
SYSPCH
SYSLST
SYSOUT
SYS000-SYSmax

X'cuu', X'ss', UA, IGN, ALT: values as described in ASSGN command</td></tr>
</table>

Figure 39. Job Control Commands (Part 1 of 3)

Operation	Operand	Remarks
DVCDN	X'cuu'	X'cuu': c = 0-6 uu = 00-FE (0-254) in hex
DVCUP	X'cuu'	X'cuu': c = 0-6 uu = 00-FE (0-254) in hex
HOLD	{F1[,F2]} {F2[,F1]}	Causes assignments for foreground logical units to be held across jobs.
LISTIO	{SYS PROG F1 F2 ALL SYSxxx UNITS DOWN UA X'cuu'}	Causes listing of specified I/O assignments.
LOG	blank	Causes logging of job control statements and single program initiation commands on SYSLOG.
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG.
MTC	opcode, {X'cuu' SYSxxx} [,nn]	opcode: BSF, BSR, ERG, FSF, FSR, RUN, REW, or WTM X'cuu': c = 0-6 uu = 00-FE (0-254) in hex SYSxxx: any logical unit nn: decimal number (01-99)
NOLOG	blank	Suppresses logging of job control statements and single program initiation commands on SYSLOG.
PAUSE	[any user comment]	Causes pause at end of current job step.
RELSE	{F1[,F2]} {F2[,F1]}	Causes single program logical units to be unassigned at EOJ.
RESET	{SYS PROG ALL SYSxxx}	Resets specified I/O device assignments
ROD	blank	Causes all SDR counters for all nonteleprocessing devices on the recorder file on SYSREC to be updated from the SDR counters in main storage.

Figure 39. Job Control Commands (Part 2 of 3)

Operation	Operand	Remarks
SET	[DATE=value1][,CLOCK=value2] [,UPSI=value3][,LINECT=value4] [,RCLST=value5][,RCPCH=value6] [,RF=value7]	value1: in one of the following formats mm/dd/yy or dd/mm/yy mm: month (01-12) dd: day (01-31) yy: year (00-99) value2: in the following format hh/mm/ss hh: hours (00-23) mm: minutes (00-59) ss: seconds (00-59) value3: 0, 1, or X value4: standard number of lines for output on each page of SYSLST value5: decimal number indicating minimum number of SYSLST disk records remaining to be written before operator warning value6: decimal number indicating minimum number of SYSPCH disk records remaining to be written before operator warning value7: defines to the system the status of the recorder file (IJSYSRC) on SYSREC used by the I/O error logging (OBR/SDR) and machine check recording and recovery (MCRR) features. RF = {YES - File Exists NO - File does not Exist CREATE - Create a File}
STOP	blank	Stops batched-job program processing
UCS	SYSxxx,phasename[,FOLD] [,BLOCK][,NULMSG]	Causes the 240-character universal character set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit. SYSxxx must be assigned to a 1403 Printer with the UCS feature.
UNA	{F1[,F2]} {F2[,F1]}	Causes immediate unassignment of foreground logical units.
UNBATCH	blank	Terminates batched-job foreground processing
Ⓑ	blank	Ⓑ is alter code 5

•Figure 39. Job Control Commands (Part 3 of 3)

Operation	Operand	Remarks
ALLOC	{F1=nK[,F2=nK] F2=nK[,F1=nK]}	Allocates foreground program areas Value of n is an even number
BATCH	{BG F1 F2}	Start, or continue batched-job operation
CANCEL	{BG F1 F2}	Cancels execution of current job in specified area
LOG	blank	Causes logging of job control statements and single program initiation commands on SYSLOG
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG
MSG	{F1 F2}	Transfers control to foreground program message routine
NOLOG	blank	Suppresses logging of job control statements and single program initiation commands on SYSLOG
PAUSE	{BG F2 F1}[,EOJ]	Causes pause at end of current job step or at end of job
START	{BG F1 F2}	Initiates a single program foreground program or resumes batched-job operations
TIMER	{BG F1 F2}	Causes interval timer support to be given to the partition specified
Ⓑ	blank	Ⓑ is alter code 5

●**Figure 40. ATTN Commands**

<table>
<tr><th>Operation</th><th>Operand</th><th>Remarks</th></tr>
<tr><td>ASSGN</td><td>SYSxxx,address [{,X'ss' / ,ALT}]</td><td>SYSxxx: can be SYSRDR, SYSIPT, SYSLST, SYSPCH, SYS001-SYSmax

address: can be X'cuu' or IGN

X'cuu': c = 0-6
uu = 00-FE (0-254) in hex

UA: unassign

IGN: unassign and ignore

X'ss': used for magnetic tape only

<table>
<tr><th>ss</th><th>Bytes per Inch</th><th>Parity</th><th>Translate Feature</th><th>Convert Feature</th></tr>
<tr><td>10</td><td>200</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>20</td><td>200</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>28</td><td>200</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>30</td><td>200</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>38</td><td>200</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>50</td><td>556</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>60</td><td>556</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>68</td><td>556</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>70</td><td>556</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>78</td><td>556</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>90</td><td>800</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>A0</td><td>800</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>A8</td><td>800</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>B0</td><td>800</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>B8</td><td>800</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>C0</td><td>800</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">dual density 9 track tape</td></tr>
<tr><td>C8</td><td>800</td><td colspan="3">dual density 9 track tape</td></tr>
</table>

ALT: specifies alternate unit</td></tr>
<tr><td>CANCEL</td><td>blank</td><td>Cancels initiation of current program</td></tr>
<tr><td>DLAB</td><td>'label fields 1-3',
xxxx,yyddd,yyddd,'system code'
[,type]</td><td>'label fields 1-3': first three fields of Format 1 DASD file label. Is a 51-byte character string, contained within apostrophes and followed by a comma. Entire 51-byte field must be contained in the first of the two commands. A continuation character is in column 72. Field 1 is the file name (44-byte alphameric); field 2 is the format identifier (1-byte numeric); field 3 is the file serial number (6-byte alphameric).

xxxx: volume sequence number (4-digit numeric). Must begin in column 16 of the continuation command. Columns 1-15 are blank.

yyddd,yyddd: file creation date followed by file expiration date. Each is 5-digit numeric.

'system code': not required. When used, a 13-character string, within apostrophes.

type: SD, DA, ISC, or ISE. If omitted, SD is assumed.</td></tr>
</table>

Figure 41. Single Program Initiation Commands (Part 1 of 4)

Operation	Operand	Remarks
DLBL	filename,['file-ID'],[date], [codes]	filename: one to seven alphameric characters, the first of which must be alphabetic 'file-ID': one to forty-four alphameric characters date: one to six characters (yy/ddd) codes: two or three alphabetic characters
EXEC	progname	progname: one to eight alphameric characters, the first of which must be alphabetic
EXTENT	[symbolic unit],[serial number], [type],[sequence number], [relative track],[number of tracks],[split cylinder track], [B=bins]	symbolic unit: six alphameric characters serial number: one to six alphameric characters type: one numeric character sequence number: one to three numeric characters relative track: one to five numeric characters number of tracks: one to five numeric characters split cylinder track: one or two numeric characters bins: one or two numeric characters
HOLD	{F1[,F2]} {F2[,F1]}	Causes assignments for foreground logical units to be held across jobs.
LBLTYP	{TAPE[(nn)]} {NSD(nn)}	TAPE: Used when tape files requiring label information are to be processed and no nonsequential disk files are to be processed. (nn): Optional and is present only for future expansion. (It is ignored by Job Control.) NSD: Nonsequential disk files are to be processed. (nn): Largest number of extents per single file.
LISTIO	{BG} {F1} {F2} {UA} {ALL}	Causes listing of specified I/O assignments on SYSLOG.
LOG	blank	Causes logging of job control statements and single program initiation commands on SYSLOG.
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG.
MSG	{F1} {F2}	Transfers control to single program message routine.
NOLOG	blank	Suppresses logging of job control statements and single program initiation commands on SYSLOG.
PAUSE	[any operator comments]	Causes pause at end of current batched-job job step or at end of current batched-job job.
READ	X'cuu'	X'cuu': c = 0-6 uu = 00-FE (0-254) in hex Note: Device must be a card reader.
RELSE	{F1[,F2]} {F2[,F1]}	Causes single program logical units to be unassigned at EOJ.
TIMER	{BG} {F1} {F2}	Causes interval timer support to be given to the specified partition.

Figure 41. Single Program Initiation Commands (Part 2 of 4)

Operation	Operand	Remarks
TLBL	filename,['file-ID'],[date], [file serial number],[volume sequence number],[file sequence number],[generation number], [version number] Note: for ASCII file processing, the fourth and fifth operands are called set identifier and file section number, respectively.	filename: one to seven alphameric characters, the first of which must be alphabetic 'file-ID': one to seventeen alphameric characters date: four to six characters (yy/ddd) {[file serial number (EBCDIC): one to six alphameric characters] [set identifier (ASCII): six alphameric characters]} {[volume sequence number (EBCDIC)] [file section number (ASCII)]} one to four numeric characters file sequence number: one to four numeric characters generation number: one to four numeric characters version number: one or two numeric characters
TPLAB	'label fields 3-10'	'label fields 3-10': indicated fields of the standard tape file label for either EBCDIC or ASCII files. A 49-byte character string, contained within apostrophes.
TPLAB	'label fields 3-10 label fields 11-13'	'label fields 3-10: same as above label fields 11-13': 20-character direct continuation of the same character string begun with fields 3-10 (no blanks, apostrophes, or commas separating). A continuation character must be present in column 72.
UCS	SYSxxx,phasename[,FOLD] [,BLOCK] [,NULMSG]	Causes the 240-character universal character set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit. SYSxxx must be assigned to a 1403 Printer with the UCS feature.
UNA	{F1[,F2] F2[,F1]}	Causes immediate unassignment of foreground logical units.
VOL	SYSnnn,filename	SYSnnn: can be SYS000, SYS001,... filename: one to seven alphameric characters, the first of which must be alphabetic

●Figure 41. Single Program Initiation Commands (Part 3 of 4)

Operation	Operand	Remarks
XTENT	type,sequence,lower,upper, 'serial no.',SYSxxx[,B_2]	type: 1 for data area (no split cylinder) 2 for overflow area (for indexed sequential file) 4 for index area (for indexed sequential file) 128 for data area (split cylinder) sequence: sequence number of extent within multi-extent file. Can be 0-255. lower: lower limit of extent in the form $B_1C_1C_1C_2C_2C_2H_1H_2H_2$ where: B_1 = 0 for 2311 or 2314; 0-9 for 2321 C_1C_1 = 00 for 2311 or 2314; 00-19 for 2321 $C_2C_2C_2$ = 000-199 for 2311 or 2314; 000-009 for 2321 H_1 = 0 for 2311 or 2314; 0-4 for 2321 H_2H_2 = 00-09 for 2311; 00-19 for 2321 or 2314 Note that the last 4 strips of subcell 19 are reserved for alternate tracks for 2321. upper: upper limit of extent in the same form as for lower limit. 'serial no.': 6-alphameric-character volume serial number contained within apostrophes SYSxxx: can be SYS000-SYSmax B_2: 0 for 2311 or 2314; 0-9 for 2321
Ⓑ	blank	Ⓑ is alter code 5

●Figure 41. Single Program Initiation Commands (Part 4 of 4)

Appendix D: Job Control Statements

<table>
<tr><th>Name</th><th>Operation</th><th>Operand</th><th>Remarks</th></tr>
<tr><td>//</td><td>ASSGN</td><td>SYSxxx,address [{,X'ss' / ,ALT}]</td><td>SYSxxx: can be SYSRDR
SYSIPT
SYSIN
SYSPCH
SYSLST
SYSOUT
SYSLOG
SYSLNK
SYSRLB
SYSSLB
SYS000-SYSmax

address: can be X'cuu', UA, or IGN
X'cuu': c = 0-6
uu = 00-FE (0-254) in hex
UA: unassign
IGN: unassign and ignore
X'ss': used for magnetic tape only

<table>
<tr><th>ss</th><th>Bytes per Inch</th><th>Parity</th><th>Translate Feature</th><th>Convert Feature</th></tr>
<tr><td>10</td><td>200</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>20</td><td>200</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>28</td><td>200</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>30</td><td>200</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>38</td><td>200</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>50</td><td>556</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>60</td><td>556</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>68</td><td>556</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>70</td><td>556</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>78</td><td>556</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>90</td><td>800</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>A0</td><td>800</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>A8</td><td>800</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>B0</td><td>800</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>B8</td><td>800</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>C0</td><td>800</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">dual density 9 track tape</td></tr>
<tr><td>C8</td><td>800</td><td colspan="3">dual density 9 track tape</td></tr>
</table>

ALT: specifies alternate unit</td></tr>
<tr><td>//</td><td>CLOSE</td><td>SYSxxx [{,X'cuu'[,X'ss'] / ,UA / ,IGN / ,ALT}]</td><td>SYSxxx: for magnetic tape -
SYSPCH
SYSLST
SYSOUT
SYS000-SYSmax

X'cuu', X'ss', UA, IGN, ALT: values as described in ASSGN command</td></tr>
<tr><td>//</td><td>DATE</td><td>mm/dd/yy
or
dd/mm/yy</td><td>mm: month (01-12)
dd: day (01-31)
yy: year (00-99)</td></tr>
</table>

Figure 42. Job Control Statements (Part 1 of 4)

Name	Operation	Operand	Remarks
//	DLAB	'label fields 1-3' C xxxx,yyddd,yyddd, 'system code'[,type]	'label fields 1-3': first three fields of Format 1 DASD file label. Is a 51-byte character string, contained within apostrophes and followed by a comma. Entire 51-byte field must be contained in the first of the two statements. Field 1 is the file name (44-byte alphameric); field 2 is the format identifier (1-byte numeric); field 3 is the file serial number (6-byte alphameric). C: any nonblank character in column 72 xxxx: volume sequence number (4-digit numeric). Must begin in column 16 of the continuation statement. Columns 1-15 are blank. yyddd,yyddd: file creation date followed by file expiration date. Each is 5-digit numeric. 'system code': not required. When used, a 13-character string, within apostrophes. type: SD, DA, ISC, or ISE. If omitted, SD is assumed.
//	DLBL	filename,['file-ID'],[date], [codes]	filename: one to seven alphameric characters, the first of which must be alphabetic 'file-ID': one to forty-four alphameric characters date: one to six characters (yy/ddd) codes: two or three alphabetic characters
//	EXEC	[progname]	progname: one to eight alphameric characters. Used only if the program is in the core image library.
//	EXTENT	[symbolic unit],[serial number],[type],[sequence number],[relative track],[number of tracks],[split cylinder track],[B=bins]	symbolic unit: six alphameric characters serial number: one to six alphameric characters type: one numeric character sequence number: one to three numeric characters relative track: one to five numeric characters number of tracks: one to five numeric characters split cylinder track: one or two numeric characters bins: one or two numeric characters
//	JOB	jobname	jobname: one to eight alphameric characters.
//	LBLTYP	{TAPE[(nn)]} {NSD (nn) }	TAPE: used when tape files requiring label information are to be processed and no nonsequential disk files are to be processed. (nn): optional and is present only for future expansion (it is ignored by Job Control) NSD: nonsequential disk files are to be processed (nn): largest number of extents per single file

●Figure 42. Job Control Statements (Part 2 of 4)

Name	Operation	Operand	Remarks
//	LISTIO	(SYS PROG F1 F2 ALL SYSxxx UNITS DOWN UA X'cuu')	Causes listing of I/O assignments on SYSLST
//	MTC	opcode,SYSxxx[,nn]	opcode: BSF, BSR, ERG, FSF, FSR, REW, RUN, or WTM SYSxxx: any logical unit nn: decimal number (01-99)
//	OPTION	option1[,option2,...]	option: can be any of the following LOG Log control statements on SYSLST NOLOG Suppress LOG option DUMP Dump registers and main storage on SYSLST in the case of abnormal program end NODUMP Suppress DUMP option LINK Write output of language translator on SYSLNK for linkage editing NOLINK Suppress LINK option DECK Output object module on SYSPCH NODECK Suppress DECK option LIST Output listing of source module on SYSLST NOLIST Suppress LIST option LISTX Output listing of object module on SYSLST NOLISTX Suppress LISTX option SYM Punch symbol deck on SYSPCH NOSYM Suppress SYM option XREF Output symbolic cross-reference list on SYSLST NOXREF Suppress XREF option ERRS Output listing of all errors in source program on SYSLST NOERRS Suppress ERRS option CATAL Catalog program or phase in core image library after completion of Linkage Editor run STDLABEL Causes all DASD or tape labels to be written on the standard label track USRLABEL Causes all DASD or tape labels to be written on the user label track PARSTD Causes all DASD or tape labels to be written on the partition standard label track 48C 48-character set 60C 60-character set
//	PAUSE	[comments]	Causes pause immediately after processing this statement. PAUSE statement is always printed on 1052 (SYSLOG). If no 1052 is available, the statement is ignored.
//	RESET	(SYS PROG ALL SYSxxx)	Resets I/O device assignments
//	RSTRT	SYSxxx,nnnn[,filename]	SYSxxx: symbolic unit name of the device on which the checkpoint records are stored. Can be SYS000-SYSmax. nnnn: four character identification of the checkpoint record to be used for restarting filename: symbolic name of the DASD file to be used for restarting

•Figure 42. Job Control Statements (Part 3 of 4)

Name	Operation	Operand	Remarks
//	TLBL	filename,['file-ID'],[date],[file serial number],[volume sequence number],[file sequence number],[generation number],[version number] Note: For ASCII file processing the fourth and fifth operands are called set identifier and file section number, respectively.	filename: one to seven alphameric characters, the first of which must be alphabetic 'file-ID': one to seventeen alphameric characters date: one to six characters (yy/ddd) {[file serial number (EBCDIC): one to six alphameric characters] [set identifier (ASCII): six alphameric characters]} {[volume sequence number (EBCDIC)] [file section number (ASCII)]} one to four numeric characters file sequence number: one to four numeric characters generation number: one to four numeric characters version number: one to two numeric characters
//	TPLAB	'label fields 3-10'	'label fields 3-10': Indicated fields of the standard tape file label for either EBCDIC or ASCII files. A 49-byte character string, contained within apostrophes.
//	TPLAB	'label fields 3-10 C label fields 11-13'	'label fields 3-10: same as above C: Any nonblank character in column 72 label fields 11-13': 20-character direct continuation of the same character string begun with fields 3-10 (no blanks, apostrophes, or commas separating)
//	UPSI	nnnnnnnn	n: 0, 1, or X
//	VOL	SYSxxx,filename	SYSxxx: can be SYS000-SYSmax filename: one to seven alphameric characters, the first of which must be alphabetic
//	XTENT	type,sequence,lower,upper,'serial no.',SYSxxx[,B_2]	type: 1 for data area (no split cylinder) 2 for overflow area (for indexed sequential file) 4 for index area (for indexed sequential file) 128 for data area (split cylinder) sequence: sequence number of extent within multiextent file. Can be 0 to 255. lower: lower limit of extent in the form $B_1C_1C_1C_2C_2C_2H_1H_2H_2$ where: B_1 = 0 for 2311 or 2314; 0-9 for 2321 C_1C_1 = 00 for 2311 or 2314; 00-19 for 2321 $C_2C_2C_2$ = 000-199 for 2311 or 2314; 000-009 for 2321 H_1 = 0 for 2311 or 2314; 0-4 for 2321 H_2H_2 = 00-09 for 2311; 00-19 for 2321 or 2314 Note that the last 4 strips of subcell 19 are reserved for alternate tracks for 2321. upper: upper limit of extent in the same form as for lower limit. 'serial no.': 6-alphameric-character volume serial number contained within apostrophes. SYSxxx: can be SYS000-SYSmax B_2: 0 for 2311 or 2314; 0-9 for 2321
/*	ignored	ignored	Columns 1 and 2 are the only columns checked.
/&	ignored	ignored	Columns 1 and 2 are the only columns checked.
*		comments	Column 2 must be blank.

●Figure 42. Job Control Statements (Part 4 of 4)

Appendix E: Format of Language Translator Output Cards and the User Replace Card

Format of the ESD Card

Card Columns	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2 - 4	ESD -- External Symbol Dictionary card.
11 - 12	Number of bytes of information contained in this card.
15 - 16	External symbol identification number (ESID) of the first SD, PC, CM or ER on this card. Relates the SD, PC, CM or ER to a particular control section.
17 - 72	Variable information. 8 positions - Name 1 position - Type code hex '00', '01', '02', '04', '05', or '0A' to indicate SD, LD, ER, PC, CM, or WX, respectively. 3 positions - Assembled origin 1 position - Blank 3 positions - Length, if an SD-type, CM-type, or a PC-type. If an LD-type, this field contains the external symbol identification number (ESID) of the SD containing the label.
73 - 80	May be used by the programmer for identification.

Format of the TXT Card

Card Columns	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2 - 4	TXT -- Text card.
6 - 8	Assembled origin (address of first byte to be loaded from this card).
11 - 12	Number of bytes of text to be loaded.
15 - 16	External symbol identification number (ESID) of the control section (SD or PC) containing the text.
17 - 72	Up to 56 bytes of text -- data or instructions to be loaded.
73 - 80	May be used for program identification.

Format of the RLD Card

Card Columns	
1	Multiple punch (12-2-9). Identifies this as a loader card.
2 - 4	RLD -- Relocation List Dictionary card.
11 - 12	Number of bytes of information contained in this card.
17 - 72	Variable information (multiple items). a. Two positions - (relocation identifier) pointer to the ESID number of the ESD item on which the relocation factor of the contents of the address constant is dependent. b. Two positions - (position identifier) pointer to the ESID number of the ESD item on which the position of the address constant is dependent. c. One position - flag indicating type of constant, as follows:
Bits	
0-2	ignored
3	0 - a nonbranch type load constant 1 - a branch type load constant
4-5	00 - load constant length = 1 byte 01 - load constant length = 2 bytes

10 - load constant length = 3 bytes

11 - load constant length = 4 bytes

6 0 - relocation factor is to be added

1 - relocation factor is to be subtracted

7 0 - Next load constant has different R and P identifiers; therefore, both R and P must be present.

1 - Next load constant has the same R and P identifiers; therefore they are both omitted.

Five significant bits of this byte are expanded in the RSERV printout.

d. Three positions - assembled origin of load constant.

73 - 80 May be used for program identification.

Format of the END Card

Card Columns

1 Multiple punch (12-2-9). Identifies this as a loader card.

2 - 4 END

6 - 8 Assembled origin of the label supplied to the Assembler in the END card (optional).

15 - 16 ESID number of the control section to which this END card refers (only if 6-8 present).

17 - 22 Symbolic label supplied to the Assembler if this label was not defined within the assembly.

29 - 32 Control section length (if not specified in last SD or PC).

73 - 80 Not used.

Format of the REP (User Replace) Card

Card Columns

1 Multiple punch (12-2-9). Identifies this as a loader card.

2 - 4 REP -- Replace text card.

5 - 6 Blank.

7 - 12 Assembled address of the first byte to be replaced (hexadecimal). Must be right justified with leading zeros if needed to fill the field.

13 Blank.

14 - 16 External symbol identification number (ESID) of the control section (SD) containing the text (hexadecimal). Must be right justified with leading zeros if needed to fill the field.

17 - 70 From 1 to 11 4-digit hexadecimal fields separated by commas, each replacing two bytes. A blank indicates the end of information in this card.

71 - 72 Blank.

73 - 80 May be used for program identification.

Glossary of New Terms

For a more complete list of data processing terms, refer to IBM Data Processing Techniques, A Data Processing Glossary, GC20-1699.

ANSI (American National Standards Institute, Inc.) Label Format: The tape file format used when the label is written in the ASCII mode.

ASCII (American National Standard Code for Information Interchange): A 128-character, 7-bit code. The high-order bit in the System/360 8-bit environment is zero.

core wrap mode: The method of operation that records the events of a trace in the area of main storage reserved by the CE parameter of the FOPT macro. It is the default process when no output device for the trace has been specified. The contents can be displayed by either a dump program or manually from the console.

DOS Volume Statistics: A facility that monitors and records the number of temporary read and write errors on currently accessed tape volumes. This facility has two options, Error Statistics by Tape Volume (ESTV) and Error Volume Analysis (EVA).

EREP (Environmental Recording, Editing, and Printing): A program that edits and prints data that has been stored in the recorder file by the I/O error logging and Machine Check Recording and Recovery (MCRR) functions.

ESTV (Error Statistics by Tape Volume): One of the two options of the DOS Volume Statistics. With ESTV support, the system collects data on tape errors by volume for any tape volumes used by the system.

EVA (Error Volume Analysis): One of the two options of the DOS Volume Statistics. With this option, the system issues a message to the operator when a number of temporary read or write errors (specified by the user at system generation time) has been exceeded on a currently accessed tape volume.

file section number: For ASCII files, the number indicates the order of a volume in a given file or multifile set. The first file must be numbered 0001. The ASCII file section number is equivalent to the EBCDIC volume sequence number.

F/L Trace (Fetch/Load Trace): A program that records information about phases and transients as they are called from the core image library. It records the number and location of the SVC, the program interrupt key, and the name, load, and entry addresses of the phase or transient.

GSVC Trace (Generalized Supervisor Calls Trace): A program that records SVC interrupts as they occur. All or a selected group of SVCs can be traced. The SVC old PSW, the task identification, the last three bytes of register 0, and the contents of register 1 are recorded.

I/O Trace (Input/Output Trace): A program that records I/O device activity for all or a selected group of I/O devices. The data recorded is the I/O old PSW and the CSW.

PDAID (Problem Determination Aids): Programs that trace a specified event when it occurs during the operation of a program. The traces provided are I/O Trace, F/L Trace, and GSVC Trace.

problem determination: A procedure or process (provided by IBM) that the user can follow after an error message to determine the cause of that error. (See PDAID)

stand-alone dump: A program that displays the contents of main storage from a minimum of 8K bytes to a maximum of 16384K bytes. It helps to determine the cause of an error.

trace:
1. To record a series of events as they occur.
2. The record of a series of events.

WXTRN (Weak External Reference): A reference to control sections and external labels defined in other separately assembled modules.

WX: One of the classifications of the ESD record recognized by the linkage editor. It is generated by WXTRN.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Disk Operating System Master Index, GC24-5063. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

(*) - comments statement 45,69
(/&) - end-of-job statement 45,69,74
(/*) - end-of-data file statement 45,69, 84

A (Action) 31
abnormal end-of-job handling 38
ABORT 22
action indicators, error messages 31
ACTION statement 84-96
    CANCEL 91
    CLEAR 91
    F1 91
    F2 91
    MAP 91
    NOAUTO 91
    NOMAP 91
activity log, update function 131
ADD statement (librarian) 130
ADD
    at IPL 77
    librarian 130
ALLOC
    allocate main storage command 32,33, 46
    control statement 137,138
Am. Nat. Std. Code for Info. Interchange (ASCII) 15
American National Standards Institute, Inc. (ANSI) 164
ANSI (American National Standards Institute, Inc.) 164
appendixes
    A. standard DASD file labels, format 1 160-162
    B. standard tape file labels 163-164
    C. operator to system commands 165-172
    D. job control statements 173-176
    E. language translator, user replace card 177
ASCII (Am. Nat. Std. Code for Info. Interchange) 15
assemble-and-execute operation 83
ASSGN statement
    job control 96,44-47
    problem determination 158
ASSGN
    assign logical name command 32,35,47
    statement 44,47
assigning system files to disk 72
ATTN commands 33,168
ATTN routine commands
    ALLOC 46
    ASSGN 47
    Ⓑ, end-of-communication 68
    BATCH 49
    CANCEL 49
    LOG 57
    MAP 57
    MSG 58
    NOLOG 58
    START 63
    TIMER 64
AUTOLINK (automatic library look-up) 87
automatic condense limit 146
automatic library look-up (AUTOLINK) 87
availability
    facilities 25
    system aids 25

Ⓑ - end-of-communication command 33,36, 68,148
background partition 10
    minimum size 15
background programs 10
background vs foreground programs 10
BATCH command 33,49
batched-job
    foreground 10
    system cataloging of ESTVUT 157
BKEND statement 126
book 133-139
bookname 126
BTAM 11

calling module 121
CANCEL 20
    cancel initiation command 35,49
    cancel job command 49,32-35
catalog
    batched-job only, ESTV 157
    core image library 115
    PDAID 149
    PDLIST 149
    programs in core image library 83
    relocatable library 120
    source statement library 126
cataloging ESTVUT 157
CATALR 120
CATALS 126
CCB (command control block) 23
CCW (channel command word) 24,95
channel command word (CCW) 24,95
channel status word (CSW) 24
channel
    command word (CCW) 24,95
    failure 26
    multiplexor 23
    requirements, MICR 12

channel (CONT.)
requirements, optical reader/sorter 12
requirements, telecommunications 11
scheduler 23
selector 23
checkpoint/restart 37
checkpoint, restarting programs from 44
CHKPT 19
CLOSE 20
close output unit command 32,50
close
system disk files 73
system tape output files 72
CLOSER macro 95
CM-common 82,85
CMPRSD operand (librarian) 135
code
device 79
private 82
COM 82
command
ALLOC (allocate main storage) 46, 32-33
ASSGN (assign logical name) 47,32-35
Ⓑ, end of communication 68,33-35
BATCH (batched-job) 33,49
CANCEL (cancel job) 49,32-35
CLOSE (close output unit) 32,50
DLAB 36,51
DLBL 51
DLBL (DASD label information) 35
DVCDN (device down) 32,52
DVCUP (device up) 32,52
EXEC (execute program) 35,53
EXTENT (DASD-extent information) 35, 53
HOLD (foreground unit assignments) 32,35,55
LBLTYP (storage for label information) 35,55
LISTIO (list I/O assignments) 32,35
LOG 57,32-35
MAP (map main storage) 57,32-35
MSG (transfer control) 58,33-35
MTC (magnetic tape control) 32,58
NOLOG (suppress logging) 58,32-35
PAUSE 60,32-35
READ (specify reader) 35,60
RELSE (release foreground unit assign) 33,35,61
RESET (reset I/O assignments) 33,61
ROD (record on demand) 33,61
SET (set value) 33,62
START (foreground and batched-job) 33,63
STOP (stop batched-job processing) 33,63
TIMER (interval timer) 64,33-35
TLBL 64
TLBL (tape label information) 35
TPLAB 36,65
UCS (load universal character set) 66,33-35
command (CONT.)
UNA (unassign foreground units) 66, 33-35
UNBATCH (terminate batched-job) 33, 66
VOL 36,67
XTENT 36,67
commands
ATTN 33,168
foreground initiation 169,33-35
job control 165,32-46
operator to system 31,165-172
single program initiation 169,33-35
common (CM) 82-86
COMMON area allocation 82-87
communication to the operator 30
communications region 15,17
foreground batched-job environmt 17
macros 19
modification 18
setting up 42
compile-and-execute operation 83
COMRG 18
condense limit, automatic 145
condense
core image library 117
relocatable library 123
source statement library 132
special considerations 144
CONDL statement 145
CONDS statement 117-132
considerations
condense 144
PDAID 149
punch 144
control programs 9,15
IPL 9,77
job control 9,39
supervisor 9,15
control statement format
job control 44
librarian 114
control statement
conventions 13
effect on I/O units 73
placement, linkage editor 84
control
cards necessary to run ESTVUT 155
dictionary 82
functions, macros for access 20
PDAID, statements 148
section 82
sequence of statements 45
statement format, librarian 114
supervisor and problem program 20
conventions, control statement 13
copy
control statements 138
core image library 138
functions 138,103-114,138-140
private libraries 146
relocatable library 139
source statement library 139
COPYC 138

COPYC (CONT.)
ALL statement 139
COPYR 139
COPYS 139
core image directory 99
size 100
core image library 10,115
catalog 115
condense 117
copy 138
delete 115
display 117
display and punch 119
maintenance 115
maintenance functions 115
punch 118
reallocation 137
rename 116
service 117
size 101
transfer of data 141
core wrap mode 150
data location 150
CORGZ 138
CPU machine check 26
creation of
file 52
private libraries 146
recorder file 26
CSECT-control section 82-95
CSERV program 103,117
CSW (channel status word) 24
current mode 48

D (Decision) 31
DASD file
identification statements 43
labels, format 1 160
protection 17,22
data
core wrap mode, location 150
PDAID printing 151
date statement 44,50

DEL statement
at IPL 78
librarian 130
delete
core image library 115
relocatable library 122
source statement library 127
DELETR 122
DELETS 127
description and format of job control statements 46
device assignment 39
device error recovery 24
device
code entries in ADD 79
specifications 48
symbolic assignment 42
unit record 41
devices supported for I/O operations 12
directories
foreground 39,95
formula for size 100
librarian functions 136
phase 18,39
directory
core image 99
relocatable 99
source statement 100
disk assignment required for ESTVUT 155
disk files
closing system 73
opening system 73
Disk Operating System components 9
disk space required for libraries and directories 100
disk storage
directories 100
libraries 100
disk work files 74
display and punch
core image library 119
relocatable library 125
source statement library 135
display program (EREP) 153
display
core image library 117
relocatable library 123
source statement library 133
DLAB
DASD label information command 36,51
statement 45,51
DLBL
command 35,51
statement 44,51
DOS system pack 13
DOS volume statistics 27
DSERV program 103,136
DUMP 20,38
dump and abort 22
DUMPGEN (stand-alone dump) 158
dumping from disk to tape, ESTVUT 156

DVCDN - device down command 32,52
DVCUP - device up command 32,52

EBCDIC 15
edit and store label information 42
END statement (librarian) 131
end-of-communication command Ⓑ 33,35
END
card 118,120
end of module 82,178
librarian 130
ENTRY card 120
ENTRY statement 84-96
environmental recording, editing, printing prog (EREP) 153
EOJ 20
ER - external reference 81
EREP (environmental recording, editing, printing prog) 153

EREP record display 154
error logging, I/O 25
error recovery 24
    device 24
error statistics by tape volume (ESTV) 27
error volume analysis (EVA) 27
ESD (external symbol dictionary) 81,177
ESD card 118,120
ESTV (error statistics by tape volume) 27
ESTV (error statistics by tape volume) 154
ESTV dump file program
    ESTVUT 27
ESTV format data set program (ESTVFMT) 157
ESTV output modes 28
ESTVFLE (ESTV data set) 28
ESTVFMT, ESTV format data set program 157
ESTVMT, ESTV magnetic tape routine 157
ESTVPR, ESTV printer routine 157
ESTVUT
    cataloging 156
    control cards 155
    dumping from disk to tape 156
    ESTV dump file program 27,154
    label information required 155
    main routine 157
    messages required 155,156
    processing functions 154
    responses to messages 155,156
    running the program 154,156
    symbolic unit assignments 155
EVA (error volume analysis) 27-30
EVA message 27,30
EXCP 20,24
EXEC
    execute program command 35,53
    statement 44,53
EXIT 20,22
EXTENT
    command 35,53
    statement 44,53
external interrupt 19
external reference (ER) 81
external symbol dictionary (ESD) 81,177
EXTRN 81

F/L (fetch/load) trace 147
factor, relocation 82,177
feature
    convert 48
    translate 48
features supported 11
    additional 12
    availability aids 25
    I/O 11
    minimum 11
FETCH macro instruction 20,36
fetch/load (F/L) trace 147
FGP programs 95
file identification
    DASD statements 43
    description 43
file identification (CONT.)
    job control 44
    linkage editor 84,96
file labels
    standard DASD 160
    standard tape 163,164
file-ID 52
file
    codes 52
    creation date 52
    DASD identification 42
    date 52
    File-ID 52
    filename 51
    IJBESTMT 157
    IJBESTPR 157
    IJBESTUT 157
    IJSYSPR 142
    IJSYSPS 142
    IJSYSRC 25
    IJSYSRL 142
    IJSYSRS 142
    IJSYSSL 142
    new residence 142
    protection 17,22
    protection, DASD 74
    recorder 26
    sequence number 54
    serial number 53
    standard DASD labels 160-162
    standard tape labels 163-164
    system code 52
    type 54
filename 52
fixed partitioned multiprogramming 10
FOPT macro 28,30
foreground initiation commands 169,33-35
foreground partition 10
    minimum size 15
foreground programs 10
    single program initiation 35
format
    CATALR 120
    CATALS 126
    CONDS 117-133
    DELETC 116
    DELETR 121
    DELETS 127
    DSPCH 119,125
    DSPLY 117-123,133-135
    general control statement 114
    job control statements 44
    language translator output cards 177
    LUB entry 41
    PUB entry 41
    PUNCH 134,118-124
    RENAMC 116
    RENAMR 122
    RENAMS 128
    UPDATE 129
    user REPLACE (rep) card 178
function
    catalog 121-125
    commands and statements 70,71

function (CONT.)
condense 117
condense, special considerations 144
copy 138,103-114
delete 121-127
display 117-133
display and punch 135,119-124
librarian 103
maintenance 103-113
maintenance, core image 115
maintenance, relocatable 120
maintenance, source statement 126
merge 140
punch 118-124
punch, special considerations 144
reallocation 137
rename 116-128
service 103-114
service, core image 117
service, relocatable 123
service, source statement 133
stand-alone dump 158
update 129
update activity log 131
functions
channel scheduler 23
job control 39
supervisor 19

generalized supervisor calls (GSVC) trace 147
GETIME 20
glossary of new terms 179
GSVC (generalized supervisor calls) trace 147

HOLD - hold foreground unit assignments command 32,55

I (Information) 31
I/O (input/output) trace 147
I/O error logging 25
I/O units control tables 16
I/O units
SYSIPT 9
SYSLOG 9
SYSLST 9
SYSPCH 9
SYSRDR 9
SYSRES 9
I/O
control tables 16
devices 12
error logging (OBR/SDR) 25
interrupt 22
linkage editor 81
IBM 2715 records 154
identification, source statement library statement 126
IJBESTMT, module name for ESTVMT 157
IJBESTPR, module name for ESTVPR 157
IJBESTUT, module name for ESTVUT 157
IJSYSPR 142
IJSYSPS 142
IJSYSRC 25
IJSYSRL 142
IJSYSRS 138,142
IJSYSSL 142
INCLUDE card 120
INCLUDE statement 84-96
indicator
Action (A) 31
Decision (D) 31
Information (I) 31
SEREP (S) 31
Wait (W) 31
initial program loader (IPL) 9,77
initiating single-program foreground programs 10
input to catalog 121
input/output (I/O) trace 147
input/output error logging (OBR/SDR) 25
input/output interrupt 19,22
input/output units 9,12
input/output units control tables 16
interrupt
external 19
handling 19
input/output 19,22
machine check 19,22
program check 19,22
supervisor call 19
supervisor processing of 11
IOCS
logical 25,95
physical 24
IPL (initial program loader) 9,77
IPL statement
ADD 77
DEL 78
SET 78

JC (job control) 39
JIB (job information block) 17
job control (JC) 39
job control commands
ALLOC 46
ASSGN 47
Ⓑ, end of communication 68
CLOSE 50
DLAB 51
DLBL 51
DVCDN 52
DVCUP 52
EXEC 53
EXTENT 53
HOLD 55
LBLTYP 55
LISTIO 56
LOG 57
MAP 57
MTC 58
NOLOG 58

job control commands (CONT.)
PAUSE 60
RELSE 61
RESET 61
ROD 61
SET 62
STOP 64
TLBL 64
TPLAB 65
UCS 66
UNA 66
UNBATCH 67
VOL 67
XTENT 67
job control statement
*-comments 45,69
/& 45,69
/* 45,69
ASSGN 96,44-47
CLOSE 44,50
DATE 44,50
DLAB 45,51
DLBL 44,51
EXEC 97,44-52
EXTENT 44,53
JOB 96,44-55
LBLTYP 97,45-55
LISTIO 45,56
MTC 45,58
OPTION 96,45-58
PAUSE 45,60
RESET 45,61
RSTRT 45,61
TLBL 45,64
TPLAB 45,65
UPSI 45,67
VOL 45,67
XTENT 45,67
job control 9,39
commands 32,165
commands, description and format 46
functions 39
statement example 75,76
statements 44,173
statements description and format 46
job date 42
job information block (JIB) 17
job name 42
JOB statement 55,74

label area in foreground partition 15
label cylinder 42
label definition (LD) 81
label information required for ESTVUT 155
label reference (LR) 81
label
information, edit and store 42
processing 43
labels
standard DASD file 160-162
standard tape file 163-164
language translator, format of output cards 177
language translator 10
output card formats 177
LBLTYP command 35,55
LBLTYP statement 45,97
LBRET 20
LD-label definition 81
LD/LR-label definition and reference 81
librarian functions 103
core image library 115
directories 136
relocatable library 120
source statement library 125
librarian 10,99
control statement format 114
reallocation 137
libraries
formulas for size 100
private 145
library 10
core image 99
relocatable 99
source statement 99
link edit job, example of setup 96
linkage editor statement
/* 84
ACTION 84-96
ENTRY 84
INCLUDE 84-96
PHASE 84-96
linkage editor 9,80
control statement format and placement 84
core requirements 15
in autolink 87
input and output 80
input and output, example of 97
input restrictions 96
job setup 96
map 94
operations 83
runs, types of 82
LISTIO
list I/O assignment command 56,32-35
statement 45,56
LOAD 20
LOAD macro instruction 37
load-and-execute operation 83
location, core wrap mode 150
LOG - log command 57,32-35
logical unit block (LUB) 16,41
logical units
programmer 40,13-16
system 16,40
LR-label reference 82
LUB (logical unit block) 16,41
LUB entry, format 41

machine check interrupt 19,22
machine check recording and ...very (MCRR) 25,26
machine requirements 11
MACRO statement 126
macro
CANCEL 20

macro (CONT.)
CCB 24
CHKPT 20
CLOSE 20
CLOSER 95
communications region 18
COMRG 18,20
DUMP 20,38
EOJ 20
EXCP 20
EXIT 20,22
external interrupts 21
FETCH 20,36
FOPT 28,30
GETIME 20
LBRET 20
LOAD 20,37
logical IOCS 25,95
MACRO 126
MEND 126
MVCOM 20
OPEN 20
OPENR 95
PDUMP 20,38
physical IOCS 24
SETIME 20
STXIT 20,22
supervisor 20
TECB 21
WAIT 20
macros for use with external interrupt 21
main storage
allocation 32
availability aids 25
background 15
batched-job 15
common 82
disk operating system 9
foreground 15
linkage editor 15
map 94
multiprogramming 11,13
multitasking 11
organization 15,16
SPI 15
telecommunications 11
MAINT program 144,103-115
maintenance functions 104-113
catalog 113
condense 113
core image library 115
delete 113
reallocate 113
relocatable library 120
rename 113
source statement library 125
update 113
maintenance of private libraries 146
MAP - map main storage command 57,32-35
MAP example 94
MCRR (machime check recording and recovery) 26
MCRR (machine check recording and recovery) 25
MCRR records 154
MEND statement 126
merge considerations 141
merge function 140
direction of transfer 142
sample job setup 143
MERGE statement 114,138,141
merge
functions 141-144
message
ATTN commands 33,168
DUMPGEN 159
ESTV 155
EVA 27,29
messages provided by ESTVUT 155,156
MICR processing requirements 12
minimum system configuration 11
minimum system requirements for multiprogramming 13
minimum
features required 11
multiprogramming system requirements 13
system configuration 11
mode 1 ESTV printout format 28
mode 2 ESTV printout format 29
mode
ASCII 15
assemble-or compile-and-execute 83
batched-job 10
catalog 83
core wrap 150
current 48
EBCDIC 15
ESTV output 28
F/L trace entry 151
GSVC trace entry 151
I/O trace entry 151
load and execute 83
single program 10
standard 48
module names for ESTV modules 157
MSG - transfer control command 58,33-35
MTC
magnetic tape control command 32,58
statement 45,58
multiplexor channel 23
multiprogramming 10
background vs foreground 10
minimum system requirements 13,36
multitasking 10
storage protection 15
telecommunications 11
multitasking 10
MVCOM 18,20

newname 128
NEWVOL statement 138,146
NOAUTO-no automatic library look-up 87
NOLOG - suppress logging command 58, 32-35
normal and abnormal end-of-job handling 38

NRS (new system residence file) 141

object module 80
OBR (outboard recorder) 25
OBR records 154
old PSW 19
oldname 127,128
OPEN 20
open
    system disk files 73
    system tape files 72
OPENR macro 95
operand
    book 126-128,133-139
    CMPRSD 135
    CORE, OPTN 158
    DECK, OPTN 158
    INTR, OPTN 158
    LOADADR, OPTN 158
    NOINTR, OPTN 158
    NRS 141,142
    PRV 141,142
    RES 141,142
    s.book1 129
    sublib 126-129,133-139
    v.m 129
operator communication 31
operator to system commands 32,165-172
optical reader/sorter requirements 12
OPTION statement 43-58
    CATAL 59,115-121
    DECK 59
    DUMP 59
    ERRS 59
    LINK 59
    LIST 59
    LISTX 59
    LOG 58
    NODECK 59
    NODUMP 59
    NOERRS 59
    NOLINK 59
    NOLIST 59
    NOLISTX 59
    NOLOG 59
    NOSYM 59
    NOXREF 59
    PARSTD 59
    STDLABEL 59
    SYM 59
    USRLABEL 59
    XREF 59
    48C 60
    60C 60
OPTN statement 158
OPTN
    CORE 158
    DECK 158
    INTR 158
    LOADADR 159
    NOINTR 158
organization of DOS system pack 13
organization
    DOS system pack 13,14
organization (CONT.)
    main storage 15,16
outboard recorder (OBR) 25
output modes, ESTV 28
overlap, read/write/compute 23
overlays, phases of multiphase program 82

PARSTD 43,59
partition
    background 10
    background vs foreground 10
    foreground 10
PAUSE
    command 60,32-35
    statement 45,60
PC-private code 82
PDAID (problem determination serviceability aids) 147-149
PDAID 147-149
    F/L trace 147
    GSVC trace 147
    I/O trace 147
    initiation 148
    PDLIST 149
    printing of data 151
    system considerations 149
    usage 153
PDLIST 149
PDUMP 20,38
permanent error counter (ESTV) 27
PHASE card 118-120
phase directory 18,39
phase entry point 95
PHASE statement 84-96
    displacement 86
    foreground 86
    name 85
    origin 85
    relocation 86
    ROOT 86
physical I/O macro instructions 24
physical transient area overlap 16
physical unit block (PUB) 16,41
prepare programs for execution 39
printing PDAID data 151
printout format
    ESTV mode 1 28
    ESTV mode 2 29
priority for CPU processing 11
private code (PC) 82
private libraries
    definition and creation 145
    maintenance 146
    use and restrictions 145
problem determination serviceability aids (PDAID) 147-149
problem determination
    DUMPGEN 158
    EREP 153
    ESTVFMT 157
    ESTVUT 154
    F/L trace 147
    GSVC trace 147

problem determination (CONT.)
I/O trace 147
PDAID 147-153
PDLIST 149
serviceability aids (PDAID) 147
processing functions, ESTVUT 28
processing programs 10
program check interrupt 19
program mask in PSW 22
program
Assembler 10
Autotest 10
background vs foreground 10
batched-job 10
BTAM 11
catalog 115-126
CATALR 120
CATALS 126
COBOL 10
condense 116-132
CONDS 117-132
control 9
CORGZ 138
CSERV 103,117-119
definition 80
delete 115-127
DELETR 121
DELETS 127
display 117-123,133-136
display and punch 119-135
DSERV 103,136
DSPCH 119-135
DSPLY 117-136
DUMPGEN 158
EREP 153
ESTV 28,154
FGP 95
fixed partitioned 10
FORTRAN 10
identifiers used in multiprogramming 31
IPL 9,77
job control 9,39
librarian 10,99
linkage editor 9,80
MAINT 137,103-125
multiprogramming 10
PDAID 147
phase 80-95
PL/I (D) 10
problem determination 10,147
processing 10
punch 134,118-124
QTAM 11
QTAM message control 11
RENAMC 116
rename 116-128
RENAMR 122
RENAMS 128
RPG 10
RSERV 103,123
SEREP 31
single program foreground 10
single program initiator 10
program (CONT.)
Sort/Merge 10
SSERV 103
supervisor 9,15
system service 9
update 129
user 10
Utilities 10
programmer logical units 40,13-16
PRV (private relocatable library) 141
PSW 19
PTA overlap feature 16
PUB (physical unit block) 16,41
PUB entry, format 41
punch
core image library 118
relocatable 124
relocatable library 124
source statement library 134
special considerations 144

QTAM 11
queue 23

READ command 35,60
read/write/compute overlap 23
reallocation function (librarian) 137
record on demand (ROD) command 61,25-33
recorder file 25
records collected by ESTV 28
records
IBM 2715 154
MCRR 154
OBR 154
SDR 154
relocatable directory 99
size 101
relocatable library 10,99
catalog 120
condense 123
copy 139
delete 121
display 123
display and punch 125
maintenance 120
maintenance functions 120
punch 124
reallocation 137
rename 122
service functions 123
size 101
transfer of data 141
relocation factor 82,177
RELSE - release foreground unit assignments at EOJ 61,32-35
RENAMC 116
rename
core image library 116
relocatable 122
relocatable library 122
source statement library 128
RENAMR 122

RENAMS 129
REP statement (librarian) 130
REP, user replace card 82,177
requirements
channel, MICR 12
machine 11
main storage for multiprogramming 13
MICR processing 12
minimum features 11
minimum system 36
multiprogramming 14
optical reader/sorter 12
telecommunications 11
RES (system residence file) 141
resequencing (single statement library statements) 128
RESET
reset I/O assignments command 33
statement 45,61
responses to ESTVUT messages 155,156
restart 37
restarting programs from checkpoint 44
restrictions, linkage editor input 96
RLD (relocation list dictionary) 177
RLD card 120
RLD-relocation list dictionary 82
ROD (record on demand) command 32,61, 25-33
ROOT 86
routines
supervisor 15
transient 15
RSERV program 103,123
RSTRT statement 45,61
running ESTVUT 155
runs, linkage editor 82

S (SEREP) 31
s.book1 operand 129
sample output for F/L trace 152
sample output for GSVC trace 153
sample output for I/O trace 152
save area in foreground partition 15
SD-section definition 81
section definition (SD) 81
selector channel 23
self-relocating programs 95

sequence of job control statements 45
SEREP program 31
service functions 114,103-109
core image library 117
relocatable library 123
source statement library 133
serviceability aids
EREP 153
error statistics (ESTV) 154
I/O error logging (OBR/SDR) 25
machine check recording recovery 26
problem determination (PDAID) 147
serviceability facilities 25
SET - set value command 32,33,62
set up communications region 42
SET, at IPL 78
SETIME 20,21
single program initiation 10
single program initiation (SPI) commands 33
single program initiation commands 35, 169
single program initiator 9
source module 80
source statement directory 100
size 102
source statement library 10,99
catalog 125
copy 139
delete 127
display 133
display and punch 135
maintenance functions 125
punch 134
reallocation 137
rename 128
service functions 133
size 102
transfer of data 141
update 129
sources of linkage editor input 84
SPI commands 35
ASSGN 47
Ⓑ, end-of-communication 68
DLAB 50
DLBL 51
EXEC 53
EXTENT 53
HOLD 55
LBLTYP 55
LISTIO 56
LOG 57
MAP 57
MSG 58
NOLOG 58
PAUSE 60
READ 60
RELSE 61
TIMER 64
TLBL 64
TPLAB 65
UCS 66
UNA 67
VOL 67
XTENT 67
SRD records 154
SSERV program 103,133
stages of program development 80
stand-alone dump (DUMPGEN) 158
standard DASD file labels, format 1 160
standard mode 48
standard tape file labels 163-164
standard tape labels
ANSI 164
IBM 163
START command 33,63
statement identification (source statement library) 129

statement
*-comments 45,69
/& (job control) 45,69
/& (linkage editor) 97
/* (job control) 45,69
ACTION 84-96
ADD 77,130
ALLOC 137
ASSGN 44,158
ASSGN (job control) 96,44-47
ASSGN (problem determination) 158
Ⓑ, end-of-communication 148
BKEND 126
CLOSE 44,50
CONDL 145
CONDS 117,123,132
conventions 14
COPYC 138
COPYR 139
COPYS 139
CSERV 103,117
DATE 44,50
DEL 78,130
DELETC 116
DELETR 121
DLAB 45,51
DLBL 44,51
DSERV 103
DSPCH 135,119-125
DSPLY 117-136,133-136
END 130
ENTRY 84-96
EREP 153
EXEC 44,97,44-52
EXTENT 44,53
general, librarian 114
INCLUDE 84-96
input to catalog 121
JOB 96,44-55
job control 44,75,173
LBLTYP 97,45-55
LISTIO 45,56
MACRO 126
MEND 126
MERGE 114,138
MTC 45,58
NEWVOL 138,146
OPTION 58,96
OPTN 158
PAUSE 45,60
PDAID 149,153
PDLIST 150
PHASE 85,84-96
PUNCH 118,124,134
RENAMC 116
RENAMR 121
RESET 45,61
RSERV 103
RSTRT 45,61
sequence 45
SET 78
SSERV 103
TLBL 45,64
TPLAB 45,65
statement (CONT.)
trailer 126
UPDATE 129
UPSI 45,67
VOL 45,67
XTENT 45,67

statistical data recorder (SDR) 25
status
channel and device errors 24
unit exception 24
wrong length record (WLR) 24
STDLABEL 43,59
STOP - stop background processing command 33,63
storage protection 15,16
storage protection
key 18
multiprogramming 15
transient area 16
structure of a program 80
STXIT 20,22
sublib 133-139
sublib (sublibrary) 126-129
sublibrary 100,133-139
sublibrary (sublib) 126-129
submodular structure 89
supervisor call (SVC) 20
supervisor call interrupt 19
supervisor 9,15
communications region 15
functions 15,18
generation 15
interrupt 20
SVC (supervisor call) 20
SVC interrupt 20
symbolic device assignment 39-42
symbolic unit assignments, ESTVUT 155
SYSFIL 73
SYSIN 40,73
SYSIPT 40
SYSIPT 73
definition 9
SYSLNK 40,75
SYSLOG 40
definition 9
SYSLST 40,73
definition 9
SYSMAX (program logical unit) 40
SYSMAX (programmer logical unit) 13
SYSOUT 40,73
SYSPCH 73
SYSPCH 40
definition 9
SYSRDR 40,73
definition 9
SYSREC 25,40,73
SYSRES 99
SYSRES 40
definition 9
SYSRLB 40,146
SYSSLB 40,146
system availability aids 25

system configuration 11
system considerations for PDAID 149
system logical units 16
system service programs 9
librarian 10,99
linkage editor 9,80
problem determination 10,147
system
cataloging batched-job only, ESTV 157
disk input and output files 72
I/O operations 71
loader 36
logical units 40
operation without an IBM 1052 36
PDAID, considerations 150
to operator messages 31
SYS000 142
SYS001 75,142
SYS002 75,141,142
SYS003 75

tables
ASCII 15
I/O units control 16
tape assignments required for ESTVUT 155
tape error block by unit (TEB) 17
tape error block by volume (TEBV) 17
tape error data (ESTV) 27
tape file, ASCII 15
tape files, opening system 72
tape modes 48
tape output files, closing system 72
tape work files 75
TEB (tape error block by unit) 17
TEBV (tape error block by volume) 17
TECB 21
telecommunications 11
temporary error counter (ESTV) 27
TIMER - interval timer command 33-35
timer feature 20
TIMER-INTERVAL, timer command 64
TLBL
command 35,64
statement 45
TPLAB
statement 45,65
tape label information command 36,65
trace
F/L 147,151
GSVC 147,151
I/O 147,151
trailer statement 126
transfer of library data 141
transfer to user routine 22
transient routines 15
translator, language 10
TXT (text) card 177,118-120
explanation 82
types of linkage editor runs 82

UCS - load universal character set buffer command 33,35,61
UCS, Universal Character Set 33
UNA - unassign foreground unit assignments command 33,36,66
UNBATCH command 33,66
uninterruptable wait state 31
universal character set (UCS) 10,33
UPDATE function
activity log 131
invalid operand defaults 130
UPDATE
activity log 131
ADD 130
control statement 129
DEL 130
function defaults 130
out of sequence 131
REP 130
source statement library 129
statement 130
UPSI byte, communications region 17,18
UPSI 62
statement 45,67
USC - load universal character set buffer command 36
user replace (REP) card 82,177
USRLABEL 43,59

v.m operand 129
VOL
statement 45,67
volume information command 36,67
volume statistics, DOS 27

W (Wait) 31
WAIT 20,24
weak external reference
WX 82
WXTRN 82
work files 74
disk, file protection of 74
WX-weak external reference 82
WXTRN-weak external reference 82

XTENT
DASD extent information command 36, 67
statement 45

DOS/360 System Control and System Service Programs (S360-36) Printed in U. S. A. GC24-5036-5

IBM®

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

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

Reader's Comments Form

IBM System/360
Disk Operating System:
System Control and System Service Programs GC24-5036-5

Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Please give specific page and line references with your comments when appropriate. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM branch offices.

	Yes	*No*
● Does the publication meet your needs?	☐	☐
● Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

● What is your occupation? _____

● How do you use this publication:

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Your comments:

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. *If you would like a reply, please supply your name and address on the reverse side of this form.*

Staple

Your comments, please . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____

Department _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM

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

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

Cut Along Line

DOS/360 System Control and System Service Programs (S360-36) Printed in U. S. A. GC24-5036-5