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

DOS

IBM

Systems Reference Library

DOS System Control and Service

This reference publication describes the IBM Disk Operating System (DOS). DOS is a set of control programs and processing programs for IBM System/360 and System/370. Using a DASD device for on-line program residence, DOS provides:

- Batched-job programming capability.
- Multiprogramming and telecommunications capability.
- Control of all input/output.
- Continued operation of all programs run in its environment.
- Problem determination data.

Detailed information is given on these major topics:

- System Organization
- System Control Programs
  Supervisor
  Job Control
  IPL (Initial Program Load)
- System Service Programs
  Linkage Editor
  Librarian
- Special Service Programs
  System Buffer Load
- Problem Determination Programs

Prerequisite for understanding this publication is a basic knowledge of System/360 or System/370 machine concepts.

For titles and abstracts of associated publications, see the IBM System/360 and System/370 Bibliography, GA22-6822.

DOS Release 26

Eighth Edition (October, 1971)

This publication was formerly titled IBM System/360 Disk Operating System: System Control and Service Programs. Although titles of some DOS publications (including this one) have been simplified, the change does not affect the contents of the publication.

This edition applies to Release 26 of the IBM 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 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This edition, GC24-5036-7, is a major revision of, and obsoletes GC24-5036-6.

Summary of Amendments

This edition reflects information about the PDAIDs transient dump program, and the three new attention commands ALTER, DSPLY, and DUMP. Maintenance changes and technical corrections are also included.

Changes and additions to the text and illustrations are indicated by a vertical line to the left of the change.

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

Comments concerning the contents of this publication may be addressed to IBM Laboratory, Publications Department, P.O. Box 24, Uithoorn, Netherlands. Comments become the property of IBM.

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 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 next section, entitled System Buffer Load (SYSBUFLD), is of particular interest to DOS users who have an IBM 3211 Printer attached to their system. This section describes the purpose of SYSBUFLD and how to use it.

The last section, Problem Determination, discusses programs to use for error recovery analysis.

Appendixes A and B contain the standard DASD (Format 1) and Tape File Labels, respectively. Appendix C contains a summary of job control statements and commands. Appendix D summarizes the linkage editor and its input cards. Appendix E summarizes multiprogramming.

> Note: In case of difference between the conventions given in this manual for control program functions and those appearing in IBM-supplied DOS component publications (such as: guides for language translators, sorts, utilities, etc, and specifications manuals), observe the specific restrictions of the component.

This edition contains a glossary of terms.

Note: The term EOB replaces B which was used in previous editions of this publication. EOB corresponds to Alternate Code 5 on the IBM 1052 Printer-Keyboard (SYSLOG for IBM System/360). The term END corresponds to the END key on the IBM 3210 and 3215 Console Printer-Keyboards (SYSLOG for IBM System/370). EOB and END are used as message terminators on their respective systems. Thus, the term EOB/END is used in this publication in reference to SYSLOG for both IBM System/360 and IBM System/370, respectively.

Alternate Code 0 is used to cancel an IBM 1052 Printer-Keyboard entry. The CANCEL key performs the same function for the IBM 3210 and 3215 Console Printer-Keyboards. Thus, CANCEL is used in this publication for both IBM System/360 and IBM System/370.

The following devices that are mentioned in these publications are not available in the United States of America:

- IBM 1270 Optical Reader/Sorter
- IBM 1275 Optical Reader/Sorter

The publications most closely related to this one are listed below.

Note: Although titles of some DOS publications have been simplified, the change does not affect the contents of the publications.

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

IBM System/370 Principles of Operation, GA22-7000.

Concepts and Facilities for DOS and TOS, GC24-5030.

DOS Data Management Concepts, GC24-3427.

DOS Supervisor and I/O Macros, GC24-5037.

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

DOS System Programmer's Guide, GC24-5073.

DOS Messages, GC24-5074.

DOS DASD Labels, GC24-5072.

Tape Labels for BPS, BOS, TOS, and DOS, GC24-5070.

DOS OLTEP, GC24-5086.

DOS System Generation, GC24-5033.

IBM System/360 Disk Operating System and Tape Operating System, Utility Macro Specifications, GC24-5042.

IBM System/360 Disk Operating System, American National Standard COBOL Programmer's Guide, GC28-6398.

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, QTAM Message Processing Program Services, GC30-5003.

Contents

Figures

The IBM Disk Operating System provides operating system capabilities for 16K and larger IBM System/360 and IBM System/370 configurations that include one or more IBM 2311 Disk Storage Drives, IBM 2314 Direct Access Storage Facilities, or IBM 2319 Disk Storage. 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, IBM 2314, or IBM 2319 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 your 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 a system work area. These programs can then be permanently placed in the system or a private core image library, 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.

Private libraries can be created, maintained, and serviced by the Librarian programs. Private libraries are similar in format to system libraries.

PROCESSING PROGRAMS

All problem 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 problem programs, or

- The control program and the Linkage Editor, with problem 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, data checks are to be allowed, the buffer must be loaded before execution of the problem program.

SPECIAL SERVICE PROGRAMS

DOS users with a specific I/O device may find a special service program useful for controlling the device. A special service program runs within the Disk Operating System environment, using the functions of the control program.

- System Buffer Load (SYSBUFLD) is a self-relocating special service control processing program for Disk Operating System users with an IBM 3211 Printer. SYSBUFLD can be executed as a job or job step to load the Forms Control Buffer (FCB) and the Universal Character Set Buffer (UCSB) of the 3211 printer. SYSBUFLD requires 2K of main storage for its execution. For detailed information, see the System Buffer Load (SYSBUFLD) section.

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 I/O 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 SYSLOG. 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.

Multiprogramming with foreground programs operating in the single-program mode is available on systems with minimum of 24K positions of main storage. Multiprogramming with a batched-job foreground capability requires a system with a minimum of 32K positions of main storage. In addition, multiprogramming support requires the storage protection feature.

Batched-Job Multiprogramming

Two system generation options, batched-job foreground processing (MPS=BJF) and private core image library (PCIL=YES), allow for full utilization of the system's multiprogramming capabilities. These options allow the linkage editor to execute in the foreground partition. Thus, foreground processing can be similar to that of background.

In a disk system supporting both options, several choices are available to you as to the partition in which programs are to be link edited and in which they are to execute. A program may be link-edited in:

1. The background to execute in the background partition, and placed in the system or a private core image library.

2. The background to execute in a foreground partition, and placed in the system or a private core image library.

3. A foreground partition to execute in the background, and placed in a private core image library which will be assigned to the background at execute time.

4. A foreground partition to execute in that foreground partition, and placed in a private core image library assigned to that partition.

Most IBM-supplied programs can be linkage edited to run in any desired partition or partitions, providing enough main storage is available to execute the linkage editor. A program which is to execute in more than one partition can be link edited for each desired partition and each version placed in a different core image library. Problem programs that are not self-relocating can be handled in the same manner.

Self-relocating programs can be placed in any library desired and can be executed in any partition. Compile, link-edit-and-go jobs (except for RPG and Basic FORTRAN) may be executed in any partition. A private core image library must be assigned to compile, link edit and go in a foreground partition.

In addition to the Linkage Editor, certain Librarian functions are self-relocating. Refer to Appendix E for the system control and service facilities applicable to each partition. Concepts and

Facilities, listed in the Preface, contains additional information about this subject.

SYSTEM-USED AREAS: Each foreground partition contains a save area for program name, old program status word, and registers. The background save area is located in the supervisor. All partitions contain a label area for label processing if the LBLTYP statement is used. These save areas (except for the background save area) are at the low end of the appropriate partition.

Save area length is 88 bytes or 120 bytes if the floating point feature (FP=YES) was specified in the CONFG macro.

Label area length is determined by the system according to the LBLTYP card specification:

- TAPE (standard tape labels) = 80 bytes.
- NSD (nn) (nonsequential disk) = 84 bytes + 20 bytes per extent statement.
- Omitted = 0.

Save Area Consideration: If you have a background job that uses nearly all the partition and you plan to run that same job in the foreground, you may need a foreground partition larger than background. For example, assume that you specified FP=YES, your background partition is 14K (14,336 bytes), and your background job, including the label area, is 14,290 bytes. Because of the save area, the job requires an additional 120 bytes in foreground and exceeds 14K. The fcreground partition, then, has to be at least 16K to run the job unchanged.

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 secticn 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, 1443, or 3211). See Note 3.

One 1052 Printer-Keyboard for IBM System/360 and one 3210 or 3215 Console Printer-Keyboard for IBM System/370. If a 1052 is 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, or

One 2319 Disk Storage.

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 2400/3420-series magnetic tape unit may be substituted for this device, except when the tape unit is substituted for a card reader during IPL. (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 these devices:

CARD READERS AND PUNCHES:

- 1442 Card Read Punch
- 2501 Card Reader
- 2520 Card Read Punch
- 2540 Card Read Punch

PRINTERS:

- 1403 Printer
- 1404 Printer (for continuous forms only)
- 1443 Printer
- 1445 Printer
- 3211 Printer

PRINTER-KEYBOARDS:

- 1052 Printer-Keyboard or 3210 or 3215 Console Printer-Keyboard (for operator communication)

PAPER TAPE UNITS:

- 1017 Paper Tape Reader with 2826 Control Unit Model 1
- 1018 Paper Tape Punch with 2826 Control Unit Model 1
- 2671 Paper Tape Reader

DASD:

- 2311 Disk Storage Drive
- 2314 Direct Access Storage Facility
- 2319 Disk Storage
- 2321 Data Cell Drive

MAGNETIC TAPE

- 2400-series Magnetic Tape Units
- 2495 Tape Cartridge Reader
- 3420-series Magnetic Tape Units

OPTICAL READERS AND SORTERS:

- 1285 Optical Reader (see Note 1)
- 1287 Optical Reader (see Note 1)
- 1288 Optical Page Reader (see Note 1)
- 1270 Optical Reader/Sorter (see Note 2)
- 1275 Optical Reader/Sorter (see Note 2)

MAGNETIC CHARACTER READERS:

- 1255 Magnetic Character Reader (see Note 2)
- 1259 Magnetic Character Reader (see Note 2)
- 1412 Magnetic Character Reader (see Note 2)
- 1419 Magnetic Character Reader (see Note 2)
- 1419P primary control unit address on 1275/1419 dual address adapter
- 1419S secondary control unit address on 1275/1419 dual address adapter
- 7770 and 7772 Audio Response Units

TELEPROCESSING DEVICES:

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

NOTES:

1. A combined total of eight 1285 and/or 1287 Optical Readers and/or 1288 Optical Page Readers is supported by the system.
2. The maximum number supported depends upon the system configuration.

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 either a 1052 Printer-Keyboard or a 3210 or 3215 Console Printer-Keyboard.

ORGANIZATION OF A DOS SYSTEM PACK

The DOS disk resident system may be on a 2311, a 2314, or a 2319 disk pack. Figure 1 shows the organization of the pack.

Control Statement Conventions

The conventions used in this publication tc 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.

NO.	COMPONENT		STARTING DISK ADDRESS				NUMBER OF TRACKS (Allocation)	R = REQUIRED O = OPTIONAL
			BB	CC	HH	R		
1	IPL Bootstrap Record 1 ($A$IPL1)		00	00	00	1	1	R
	IPL Bootstrap Record 2 ($A$IPLA)		00	00	00	2		R
	Volume Label		00	00	00	3		R
	User Volume Label		00	00	00	4		O
2	System Directory	Record 1	00	00	01	1	1	R
		Record 2	00	00	01	2		R
		Record 3	00	00	01	3		R
		Record 4	00	00	01	4		R
	IPL Retrieval Program ($$A$IPL2)		00	00	01	5		R
3	System Work Area (Librarian Area)		00	00	02	1	3	R
4	Transient Directory ($$A and $$B Transients)		00	00	05	1	1	R
5	Open Directory ($$B0)		00	00	06	1	1	R
6	Library Routine Directory ($ Phasenames)		00	00	07	1	1	R
7	Foreground Program Directory (FGP)		00	00	08	1	1	R
8	Phase Directory (For Problem Program Phases)		00	00	09	1	1	R
9	Core Image Library Directory		00	01 for 2311 00 for 2314/ 2319	00 for 2311 10 for 2314/ 2319	1	*	R
10	Core Image Library		00	End of CI Directory X	Y + 1	1	*	R
11	Relocatable Library Directory		00	End of CI Library Z + 1	00	1	*	O
12	Relocatable Library		00	End of RL Directory X	Y +1	1	*	O
13	Source Statement Library Directory		00	End of RL Library Z + 1	00	1	*	O
14	Source Statement Library		00	End of SS Directory X	Y + 1	1	*	O
15	Volume Area File Definition Storage Area			End of SS Library Z + 1	00	1	2311:10 2314/2319:20	R
16	User Area			End of Volume Area Z +2	00	1	*	O

*Allocation Dependent On User Requirements
X=Ending CC of the Preceding Directory
Y=Ending HH of the Preceding Directory
Z=Ending CC of the Preceding Library

Figure 1. Organization of a DOS System Pack

Supervisor

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 pricrity).

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 or a private core image library 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 lcad)
- End-of-job handling
- Timer services
- Checkpoint/Restart
- Job Accounting Interface

All functions except certain interrupt handling (SVC, I/O, and machine check) and job accounting interface are available to the problem program by issuing macro instructions. The programmer is nct concerned with machine interrupt conditions, since these are handled automatically by the Supervisor.

The DOS system can record machine check interrupt conditions for use as a customer engineering diagnostic aid if you specify 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 cutput). 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 intc 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 an 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 problem 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 physical transient area overlap feature specified 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, ensure that it is protected by defining the supervisor end (SEND) address large enough to include the physical transient area. Contention for the physical transient area is reduced and overall system performance is improved if Independent Directory Read-In Area (IDRA) is specified at system generation time. This option generates an independent directory read-in area, allowing fetching of phases while the physical transient area is busy. The IDRA is used for all supervisor calls which require reading of directories.

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, represents one logical (symbolic) I/O unit, and references an entry in the PUB. LUBs corresponding to logical units are ordered according to the logical units they represent. The LUBs are grouped into two classes: system logical units and programmer logical units. For information concerning the ordering of LUBs, see Logical Unit Block (LUB) and Physical Unit Block (PUB) in the Job Control section. The number of LUBs must not exceed 222 when MPS=BJF or 244 when MPS=NO or YES is specified in the SUPVR macro at system generation time.

Supervisor Storage Protection Key: 0	Permanent Storage Locations Used by CPU Communications Region EXCP Routine I/O Interrupt Routine } Channel Start I/O Routine } Scheduler Storage Protection (required for multiprogramming) Supervisor Call Routine Program Check Routine Machine Check Routine External Interruption Routine Timer Services (optional) Job Accounting Interface (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 Linkage Editor 1 Librarian Maintenance Function 2 Librarian Core Image and Directory Service 3 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 Linkage Editor 1 Librarian Maintenance Function 2 Librarian Core Image and Directory Service 3 Installation Processing Programs

1. MPS=BJF must be specified and a private core image library must be assigned.
2. MPS=BJF must be specified and the maintenance function must be performed on an assigned private core image library.
3. MPS=BJF must be specified.

Figure 2. Main Storage Organization

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 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 and the priority within a channel. The number of PUBs is specified at system generation time with a minimum of 5 and a maximum of 255.

Each JIB is four bytes and contains LUB or extent information. The number of JIBs is specified at system generation time with a minimum of 5, and a maximum of 255.

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 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 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 that may be of interest to you. For a complete layout of the communications region, see the System Programmer's Guide, listed in the Preface.

Bytes	Length	Description
0-7	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.
8,9	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.
10,11	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 manual listed in the Preface.

Figure 3. Communications Region in Supervisor (Part 1 of 2)

Bytes	Length	Description
12-22	11 bytes	User (problem program) area (for interprogram or intraprogram communications). All 11 bytes set to binary zero when the control statement JOB is encountered.
23	1 byte	UPSI (user program switch indicators). Set to binary zero when the control statement JOB is encountered. Initialized by UPSI job control statement.
24-31	8 bytes	Job name as found in the JOB control statement.
32-35	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.
36-39	4 bytes	Address of the uppermost byte of a phase placed in the problem-program area by the last FETCH or LOAD.
40-43	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 background programs only, job control builds a problem program 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.
44,45	2 bytes	Length of batched job or background program label area.

Figure 3. Communications Region in Supervisor (Part 2 of 2)

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.

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, you wish to record machine check interrupt conditions, you 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.

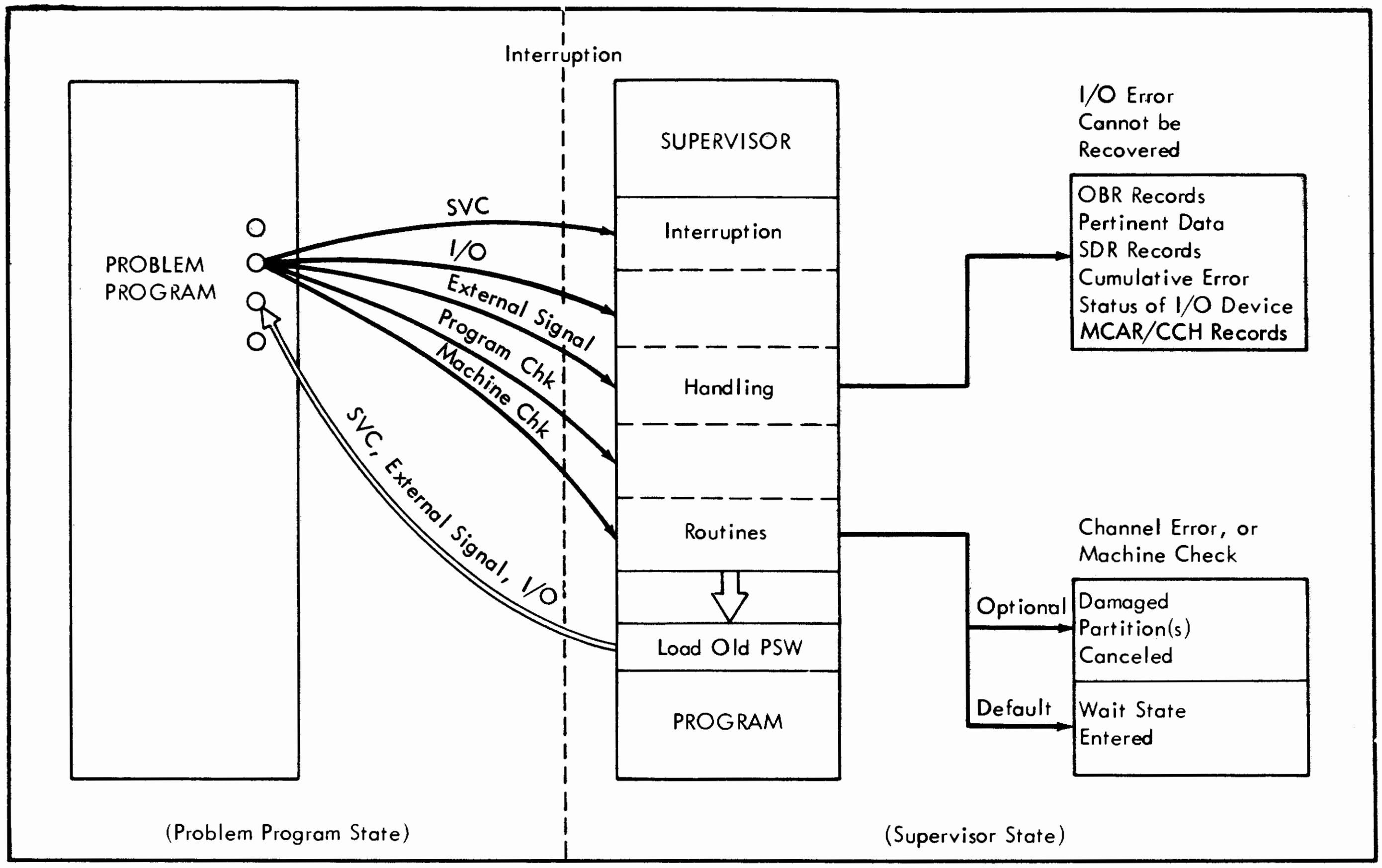


Figure 4. Flow of Control Between Supervisor and Problem Program During an Interrupt

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 problem program may have control of the processing of program check and external interrupts.

SUPERVISOR CALL INTERRUPT

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 a 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 a 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. The SVC and the macro instruction that caused it are listed in the Problem Determination section.

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 SYSLOG 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 you have provided an address of your 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 you 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

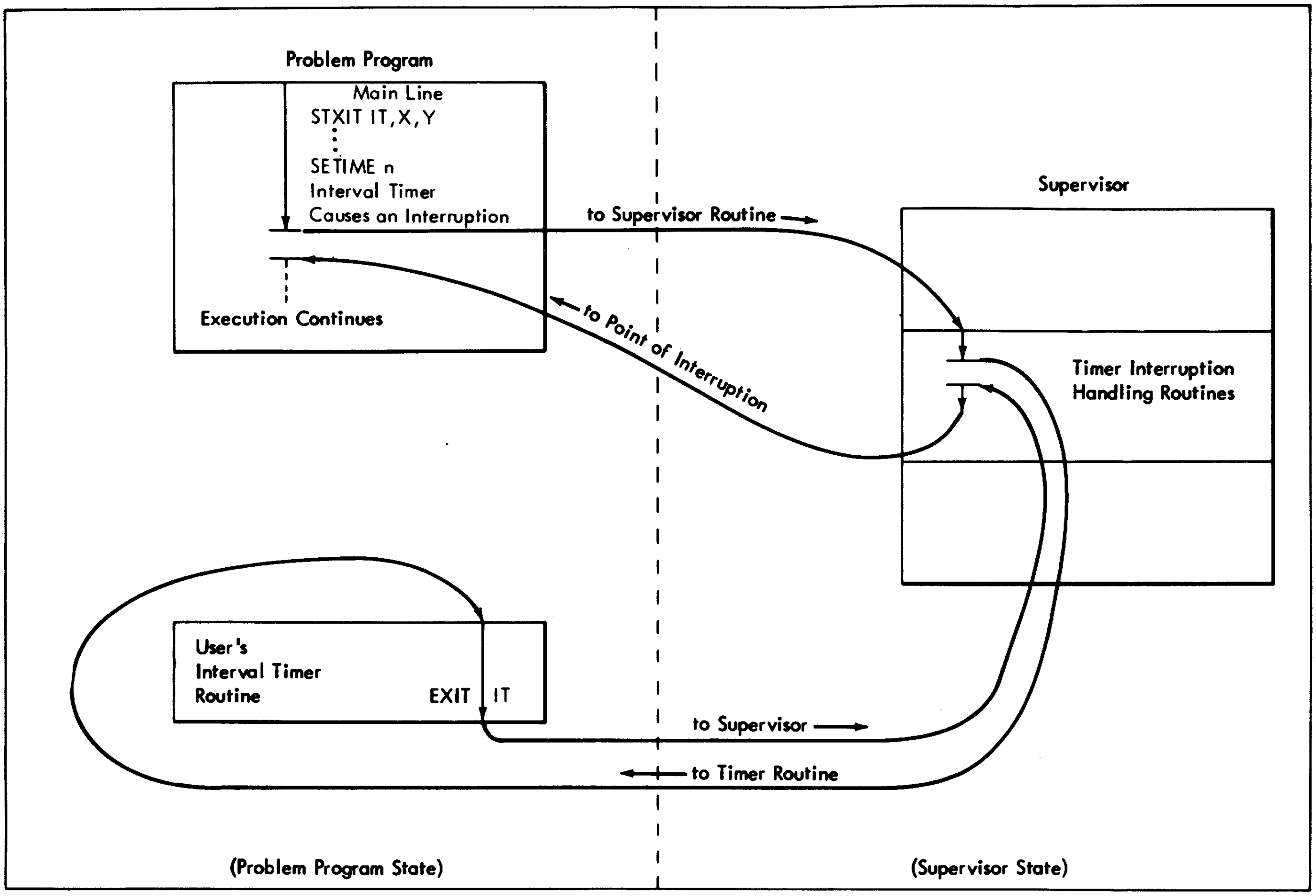


Figure 5. Typical Sequence of Events After a Timer Interrupt Due to Use of SETIME

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 your requirements. 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 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 INTERRUPT

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 you via the STXIT macro instruction, the program-check interrupt routine branches to that subroutine when an appropriate interrupt occurs. The subroutine 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 you wish to enable a program interrupt, 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 INTERRUPT

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 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 for IBM System/360 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).

The machine check analysis and recording (MCAR) function, when included for IBM System/370, gathers information about the system hardware that aids in diagnosing failures, and tries to overcome the conditions that prevent the system from continuing to process.

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 INTERRUPT

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 SYSLOG.

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 detecticn, attention (SYSLOG).

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 IBM System/360 Models 22, 30, and 40, and IBM System/370, tape and DASD devices operating cn 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 corpletion 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, operatcr intervention, notify problem program, or terminate job.

The operator can initiate an I/C operation through SYSLOG. 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 frorr 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 ICCS. In the absence of any other options, only choice 2 is available. Tape error statistics by unit can be printed on SYSLCG, and tape error statistics by volume can be printed on SYSLOG 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 tire to notify the operator by a message on SYSLOG 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 and IBM 2715 Transmission Control Error Recording) and either the machine check recording and recovery (MCRR) or the recovery management support (RMS) features increase system availability. These features support all devices supported by the systerr. If these features are desired, you must specify the options at system generation tire. For teleprocessing devices, you must also specify the OBR/SDR ιption in the 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. You 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 forrerly 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 these features:

- 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.

- 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 number of SDR records may be increased by specifying this at systerr 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.

- 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. The ROD command should be given when a 2314/2319 plug is switched from one module to another.

- I/O error logging for System/370 users includes RDE (Reliability Data Extractor).

If ERRLOG=RDE is specified during system generation, RDE gathers hardware reliability data that IBM personnel use to evaluate hardware performance. Two types of records are written on SYSREC by RDE.

1. An IPL record. This specifies the reason for IPLing.
2. An EOD (End-of-Day) record. This is initiated by the ROD command. The ROD command should be issued before the system is shut down.

EREP uses these records to identify RDE data.

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.

RECOVERY MANAGEMENT SUPPORT (RMS)

The Recovery Management Support for the IBM System/370 consists of two functions: Machine Check Analysis and Recording (MCAR) and the Channel Check Handler (CCH). These functions of IBM Disk Operating System gather reliability information about the system hardware. This information can reduce system downtime caused by hardware failures. The MCAR and CCH programs contain error recovery procedures that reduce the number of conditions that put the system in a hard wait state. The channel error recovery function is performed by the Channel Check Handler Error Recovery Procedures (CCH/ERP).

Because the MCAR/CCH programs gather information about the system hardware and produce environmental records to aid in diagnosing failures, system reliability, availability, and serviceability are increased.

If the machine check or channel check hardware does not function properly, the results of MCAR/CCH are unpredictable.

If a System/370 CPU is not specified in the CONFG macro during system generation, the system enters a hard wait on all machine checks and channel checks. CCH support returns control to the problem program under some channel check conditions if the accept unrecoverable I/O error bit in the CCB is turned on.

MCAR/CCH is only functional after IPL is completed. Thus, any MCI or channel detected errors occurring before IPL is completed results in immediate system termination without an attempt to record externally or issue an operator message. In such a case a unique message code is in bytes 0 and 1 of main storage. When a System/370 CPU is specified at system generation time OLTEP hooks, OBR/SDR, and MCAR/CCH functions are automatically included in the supervisor. The minimum size supervisor supported for the MCAR/CCH feature is 14K.

SYSRES portability is an RMS function that allows the System/370 DOS system that you generate to be CPU independent. For portability, specify MODEL=155 or MODEL=145 and PORT=155 in the CONFG macro during system generation. If MODEL=145 is specified and PORT=155 is not, then the extended logout area of the Model 155 is not generated. Although the Model 145 DOS system can operate on a Model 155, any records written on SYSREC do not contain the data from the extended logout area of the Model 155.

The RMS function records data on the recorder file. This file is identified as the system logical unit SYSREC. The filename operand (in the DLBL statement) is IJSYSRC, and must be defined as a disk extent on either the system resident disk or any IBM 2311, 2314, or 2319 DASD device with a minimum of ten tracks.

MCAR/CCH supports the following IBM I/O devices:

- 1403 Printer
- 1443 Printer Model N1
- 3211 Printer
- 3210 Console Printer-Keyboard Model 1
- 3215 Console Printer-Keyboard
- 2311 Disk Storage Drive
- 2314 Direct Access Storage Facility
- 2319 Disk Storage
- 2321 Data Cell Drive
- 2400-Series Magnetic Tape Units
- 3420-Series Magnetic Tape Units
- 1442 Card Read/Punch
- 2501 Card Reader
- 2520 Card Read Punch
- 2540 Card Read Punch

Resident MCAR/CCH components follow:

Machine Check Handler: The machine check handler determines the severity of the error, informs the operator with a message on SYSLOG and performs one of the following actions:

- Continues processing.
- Cancels the affected partition.
- Puts the system in hard wait.

Channel Check Handler: The channel check handler determines the severity of the error, prints any appropriate message on SYSLOG, and performs one of the following actions:

- Provides information for the ERP.
- Cancels the affected partition.
- Puts the system in hard wait.

Error Handling Monitor: This functional component monitors I/O requests from the recovery transient area (RTA). The RTA is a 1000-byte resident area where analysis and recovery procedures for machine checks and channel checks are performed. All MCAR/CCH non-resident modules use this area.

To control the RTA, the error handling monitor:

- Fetches R-transients into the RTA.
- Schedules I/O requests from the RTA.
- Provides an exit interface from RTA transients.

Error Handling Communication Area: Contains the RTA tables and information bytes for the MCAR/CCH functions.

Supervisor Interface: Provides linkage between MCAR/CCH and the supervisor.

DASD CCH/ERP: Retries the applicable DASD device.

Machine Check Analysis and Recording (MCAR)

Whenever possible, recovery from a hardware malfunction is attempted by RMS (CPU retry, and Error Correction Code, ECC). If recovery from hardware failure is unsuccessful, processor activity is terminated, and MCAR attempts to isolate the failure to a partition(s) in order to cancel the job (or task) selectively and, if possible, continue system operation. The MCAR facility (software) isolates partitions and selectively cancels jobs.

If a soft MCI occurs, system operation can continue. When a soft MCI occurs, MCAR externally records pertinent error information on SYSREC, and returns control to the interrupted code. Processing then continues in recording or quiet mode.

Partition reduction results if MCAR determines that the MCI is caused by an inoperable position of main storage. MCAR locates the failure and determines in which half of the partition it occurred. If failure occurred in the upper half of any partition or in the lower half of the background partition, the partition's ending address is lowered to the highest 2K boundary below the address of the failure. If the failure occurred in the lower half of a foreground partition, the partition's address is raised to the lowest 2K boundary above the starting address of the failure. If partition reduction causes the size of a foreground partition to be less than 2K or the background partition to be less than 10K, the partition is not usable. When the partition size is successfully reduced, appropriate supervisor tables are updated to reflect the change and the operator is notified.

> Notes: Programs that exceed the size of the reduced partition cannot be executed in the reduced partition. If a foreground partition's starting address is changed, then programs to be executed in this partition must be self-relocating or linkage edited at the new starting address.

If system operation cannot continue, external recording is attempted, after which the system enters a hard wait state.

In the case of a hard machine check in the problem program area, the problem program does not regain control, but the system does not enter a hard wait state. Whenever an MCI occurs, communications with the operator and requests for his decisions take place.

MCAR MODES OF OPERATION: An Error Frequency Limit (EFL) algorithm prevents SYSREC from filling up too quickly if a large number of intermittent failures (soft MCI) occur. The initial IBM-supplied EFL threshold values are eight soft HIR MCI, and either eight (Model 155) or sixteen (Model 145) soft ECC MCI within an eight-hour period. These values are set at system generation time, and can be changed by the MODE command. MCAR supports EFL for two hardware facilities:

- Hardware Instruction Retry (HIR)

- Error Correction Code (ECC)

Both HIR and ECC are recovery facilities of System/370.

EFL Threshold Values: At Initial Program Load (IPL) time, the EFL threshold values are effected so that the EFL algorithm controls the number of soft MCI recorded. These values are:

- The number of soft MCI.

- A specific time period.

When these EFL values are reached, a change in mode of operation occurs. Until the EFL threshold values are reached, the system operates in recording mode. This is the normal mode of operation in which an MCI occurs for all machine check conditions. After the EFL threshold values are reached, ECC (and/or HIR) is placed in quiet mode. In quiet mode, no soft MCI occurs, and the number of corrected errors will be unknown.

Hardware Instruction Retry (HIR) Modes: The two HIR modes are:

- Recording. A soft MCI occurs for every hardware instruction correction

- Quiet. No soft MCI occurs for hardware instruction corrections

If HIR is in quiet mode, ECC is also.

Error Correction Code (ECC) Modes: The ECC modes are:

- Recording. A soft MCI occurs for every main/control storage correction

- Quiet. No soft MCI occurs for main/control storage corrections

- Threshold (Model 145 only). A soft MCI occurs after a predetermined number of unrecorded control storage errors have occured within a given time period. Threshold mode is a hardware function and is not affected by EFL threshold values.

When ECC is in quiet mode HIR can still be in recording mode. System Reset places HIR in quiet mode. When IPL is completed:

- For the Model 145, recording mode is entered for HIR, quiet mode is entered for main storage ECC, and threshold mode is entered for control storage ECC.

- For the Model 155, full recording mode is entered for both HIR and ECC.

At this time, the DOS system has the IBM-supplied EFL threshold values in effect. These values can be changed by the MODE command.

Channel Check Handler (CCH)

When a channel check occurs, the CCH resident program (the Channel Check Severity Detect Routine) analyzes the severity of the malfunction and assigns it a disposition.(On the Model 155, preceding each channel check is a soft MCI.) Every channel check is submitted to a severity analysis and assigned a disposition. Thus, the CCH resident program does not attempt to recover from a channel malfunction.

If channel error information is logged and an interrupt is generated, CCH examines the error information to determine if the system is to be terminated. If system termination is not required, the following occur:

- Error information is gathered and saved in the Error Recovery Procedure Interface Bytes (ERPIB) for use by the appropriate CCH/ERP in the RTA.
- Error information is recorded cn the recorder file.

If the error cannot be isolated to a device, CCH cancels all problem programs using the malfunctioning channel. If the accept unrecoverable error bit is cn in the CCB, the error is posted and control is returned to the problem program.

If the error cannot be retried, CCH cancels all problem programs using the malfunctioning channel.

Channel Check Handler Error Recovery Procedures (CCH/ERP)

When a channel error occurs, and CCH has determined that recovery is possible, the proper CCH/ERP is loaded into the RTA, and activated. The ERP examines the ERPIB supplied by CCH and determines the severity of the error. Whenever possible, the failing channel command is retried. If the failure cannot be retried, or if the retry fails, a message is issued by the ERP Message Writer to the operator cn SYSLOG and the problem program is canceled. If the accept unrecoverable error bit is on in the CCB, the error is posted, and control returns to the problem program. A message is issued if the error recovery was successful, except for SYSLOG. Certain retry conditions require manual operator intervention to enable proper retry.

A request to run the program EREP or SEREP is made for any of the following conditions:

- When the first record on the last track in the OBR/MCAR section of the recorder file is reached, run EREP to avoid the risk of losing statistics.
- When an unrecoverable I/O error on the recorder file occurs while accessing the record indicated, the record is ignored, and processing continues. When this error persists, run EREP to retrieve the information from the file, and the file should be recreated at a different location.
- When the SYSREC becomes full, no further recording occurs until the file IJSYSRC is purged. To avoid the risk of losing statistics, run EREP. No recycling of the file occurs.
- For system termination type situations (for example, a machine check was unrecoverable, the channel caused system reset, or two channels are damaged) encountered by MCAR/CCH, recording is attempted. Depending on the success of recording, the execution of EREP or SEREP is requested. An attempt is made to write an operator awareness message. If the attempt is unsuccessful, the message code is in low core.
- If the recorder file is at least 90% full at IPL time, the operator is requested to run EREP to prevent the loss of pertinent error data.

Note: After running SEREP, the system must be re-IPLed.

RECORDER FILE (SYSREC)

A recorder file must be created using the file definition statements of the system, when either the I/O error logging or the MCRR functions or when a System/370 CPU is specified during system generation. The recorder file is not CPU or SYSRES dependent. Thus, it can contain records from more than one DOS system as long as the supervisor PUB tables are identical. 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, 2314, or 2319 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.

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

Type of Error		2311	2314/ 2319
OBR		25	40
2715		25	40
SDR		29	38
MCRR or MCAR/ CCH	Channel Inboard		
	Model 30, 145, or 155	25	40
	Model 40 or 50	6	10
	CPU		
	Model 30	8	13
	Model 40 or 50	5	8
	Model 145 or 155	1	2

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.

```
ADD
 .
 .     If not in system generation
 .
SET    DATE=mm/dd/yy
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

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 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.

You have the option of specifying, at system generation time, whether to record the data on the direct access storage device (DASD) or on 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 SYSLOG) when a certain number of temporary read or temporary write errors occurs on the tape volume currently in use.

You can specify either or both cf 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 problem program contain an OPEN(R) statement 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, 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.

Note 3: On a multi-file volume each file is processed independently, as if each file were a new volume. Therefore, if a volume serial number is encountered on the first file of the volume, it is recorded for the first file only, and any other files on the same volume are recorded as unlabeled volumes.

```
┌──────────────────────────────────────────────────────────┐
|VOLUME              TIME      CHANNEL  TEMP   TEMP    PERM |
|SERIAL  DATE        OF DAY    /UNIT    READ   WRITE   READ |
|xxxxxx  yr/day      hr.mn.sc  cuu      nnn    nnn     nnn  |
├──────────────────────────────────────────────────────────┤
|PERM    NOISE       ERASE     CLEANER  SIOs   TAPE    BLOCK |
|WRITE   BLOCKS      GAPS      ACTIONS  USAGE  DENSITY LENGTH|
|nnn     nnn         nnn       nnn      nnnnn  nnn     nnnnn |
└──────────────────────────────────────────────────────────┘
```

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

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 file 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):

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 cccurred 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 SYSLOG each time a particular volume is ended by CLOSE, EOJ, EOV, or CANCEL. When Mode 2 is the selected method of writing ESTV records, a subset of the Mode 1 ESTV record is printed as fcllows:

```
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 SYSLOG 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.

On-Line Test Executive Program (OLTEP)

The On-Line Test Executive Program (OLTEP), together with the On-Line Tests (OLTs), make up the On-Line Test System, which is designed to test I/O devices with minimum interference to other programs running on the system. OLTEP functions as an interface between the system and the test programs, and provides communications with the operator via SYSLOG during the running of the tests.

For its execution, OLTEP must be specified during system generation and the appropriate OLTEP phases for either a batched-only or a multiprogramming system must be cataloged to a core image library. Because the minimum supervisor size to execute OLTEP is 8K, it requires a 24K or larger system.

RETAIN/370 is an OLTEP function that allows the OLTEP programs to be executed on your System/370 from a remote location. RETAIN/370 is a problem determination tool used by IBM. To use this function, RETAIN must be specified during system generation, and the IBM 2955 Retain Communications Device must be included in the PUB table. The RETAIN/370 function is available on the System/370 Models 145 and 155 in the United States of America only.

For a more detailed description of OLTEP and the operating procedures, see the DOS OLTEP manual listed in the Preface.

Operator Communication

Communication with the operator is through use of full-text messages issued via a SYSLOG (either IBM 1052 Printer-Keyboard for IBM System/360 or a 3210 or 3215 Console Printer-Keyboard for System/370). 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. 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 communications 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 SYSLOG. 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 SYSLOG, but may be displayed on theconsole. 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 enter via SYSLOG 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 SYSLOG 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, SYSLOG 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 EOB/END.

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

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

1. Job Control commands (JCC).

2. Message (ATTN) commands (AR).

3. Single-program initiation commands (SPI).

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 SYSLOG. 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 SYSLOG causes the message READY FOR COMMUNICATIONS to be printed on SYSLOG. 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 (JCC)

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, 2314, or 2319.

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.

EOB or END End-of-Communication Command. Must be issued whenever the operator is finished communicating with the system.

HOLD Hold Foreground Unit Assignments Command. Causes all I/C 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 2400/3420 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; and also initiates writing the EOD record for System/370 RDE users.

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 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 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.

ATTN Commands (AR)

The ATTN commands are:

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

ALTER — Alter Main Storage Command. Permits the operator to alter one to sixteen bytes of main storage at a time.

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.

DSPLY — Display Main Storage Command. Permits the operator to display sixteen bytes of main storage.

DUMP — Dump Main Storage Command. Permits the operator to dump logical (for example, a partition) or physical areas of main storage on SYSLST.

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

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.

MODE — Mode Command. Changes the mode of operation between recording and quiet, changes the EFL threshold values, and gives status information. MODE, which is only valid for System/370, provides the following MCAR/CCH information:

1. Current error count.
2. Error count threshold.
3. Current elapsed time.
4. Time threshold.

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.

Single Program Initiation (SPI)

Single program foreground programs are initiated by the operator from 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 SYSLOG. Control is given to the message (ATTN) routine, which reads commands from the operator via SYSLOG. 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 a 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 program in foreground, 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 SYSLOG 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 SYSLOG 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.

SPI COMMANDS

Single program initiation commands are submitted following a START command to the ATTN routine. They can be submitted through 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.

EOB or END End-of-Communication Command. Causes input reading to switch back to the device specified in the READ command.

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.

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	<u>DASD Label Information Command</u>. Contains file label information for DASD label checking and creation.
TPLAB	<u>Tape Label Information Command</u>. Contains file label information for tape label checking and writing.
VOL	<u>Volume Information Command</u>. 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	<u>DASD Extent Information Command</u>. Defines each area, or extent, of a DASD file. It is used in conjunction with the VOL, DLAB commands.

System Operation without an IBM 1052 or an IBM 3210 or 3215

Multiprogramming requires a 1052 or a 3210 or 3215. A batched-job environment, however, may be made operational when the normal SYSLOG 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 the normal SYSLOG is not 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 SYSLOG.

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 a core image library.

When a phase is requested, the phasename is examined. If the phase name starts with $, the system core image library is searched first. If it is not located, the private core image library assigned to the partition is searched. If the phase is a transient (IBM-supplied transients begin with $$A, $$B, or $$R), only the system core image is searched. If it is not located, the system enters the wait state with a low core error message (refer to <u>DOS Messages</u> listed in the <u>Preface</u> for error messages). For all other phases, the private core image library assigned to the partition is searched first. If it is not located, the system core image library is searched.

> <u>Note</u>: IBM-supplied transients must be cataloged to the system core image library.

<u>FETCH Macro Instruction</u>

This macro instruction has the format:

Name	Operation	Operand
[name]	FETCH	$\left\{\begin{matrix}\text{phasename}\\ (1)\end{matrix}\right\}\left[\left\{\begin{matrix},\text{entryname}\\ ,(0)\end{matrix}\right\}\right]$

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	$\left\{\begin{matrix}\text{phasename}\\ \text{(1)}\end{matrix}\right\}\left[\left\{\begin{matrix}\text{,loadaddr}\\ \text{,(0)}\end{matrix}\right\}\right]$

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, 2314, or 2319 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 toa 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, you 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, 2314, or 2319 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, nc 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. Output tapes with embedded checkpoint records created under DOS can also be processed by OS (IBM System/360 Operating System).

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.

Job Accounting Interface

Job Accounting Interface is an additional option to the supervisor that can be specified at system generation time. Information is accumulated for each batch job partition, and is kept in the Job Accounting Table. To use this information, you must write a self-relocating output routine (no larger than 4K) and catalog it in the Core Image Library under the name $JOBACCT.

For programming considerations, see 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.

Job Control

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 SYSLOG. 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:

- Prepare programs for execution.
- Assign device addresses to symbolic names.
- Set up fields in the communications region.
- Edit and store volume and file label information.
- Provide the interface with a user-written output routine for job accounting.
- Prepare for restarting of checkpointed programs.
- Provide System/370 RDE Data.

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 or from a private core image library. If a program has been previously cataloged (see Librarian), Job Control constructs a problem program phase directory on the private core image library (if it is assigned) or on SYSRES, and directs the system loader to load that program for execution. The problem program phase directory is only built for background programs. Job Control, however, does not build a program phase directory, if the program phases are on the system core image library and a private core image library is assigned. If the program is stored temporarily, Job Control constructs a phase directory of that program by copying first track of SYSRES work area and then transfers to the system loader to load it for execution, in the BG, F1 or F2 partition.

Unless all phases of a program are cataloged to the same library, bytes 40-43 of the communications region do not reflect the correct ending address of the largest phase fetched or loaded. Attentive use of the problem phase directory reduces FETCH/LOAD time.

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 are loaded from the system core image library or a private core image library. If a foreground program is stored temporarily (compile, link and go), a problem program phase directory is created on the private core image library (if it is assigned) or on SYSRES.

SYMBOLIC INPUT/OUTPUT ASSIGNMENT

Job Control is responsible for assigning physical I/O units. Programs do nct 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 or console 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.
SYSCLB	Disk extent used for a private core image library.
SYSRLB	Disk extent used fcr 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 2715 Transmission Control Error) and either the machine check recording and recovery (MCRR) or the recovery management support (RMS) functions.
SYS000-SYSmax	All other units in the system.

Whenever a system logical unit (SYSRDR, SYSIPT, SYSLST, SYSPCH, SYSCLB) 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 eleven of these symbolic names, termed system logical units, are used by the system control program and system service programs. Of these eleven 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 your convenience, 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 tc 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.

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.

```
┌────────┐
|SYSRDR  |
├────────┤
|SYSIPT  |
├────────┤
|SYSPCH  |       ┌────────┬───────────────┐
├────────┤───────|Pointer |Pointer to     |
|SYSLST  |       |to PUB  |Job Information|
├────────┤───────|        |Block          |
|SYSLOG  |       ├────────┴───────────────┤
├────────┤       | -------2 bytes-------  |
|SYSLNK  |
├────────┤
|SYSRES  |
├────────┤
|SYSSLB  |       Note:  SYSCLB is not
├────────┤       shown in the LUB table
|SYSRLB  |       because it is in a
├────────┤       special location in
|SYSUSE  |       the supervisor.
├────────┤
|SYSREC  |
├────────┤
|SYS000- |
|SYSmax  |
└────────┘
```

Figure 8. Sequence of LUBs in Device Table

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 varicus devices are attached.

Channel
Unit
Pointer to Channel Queue
Error Retry Counter or 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

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 symbclic 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 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 or a 3210 or 3215 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 via SYSLOG 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.

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.

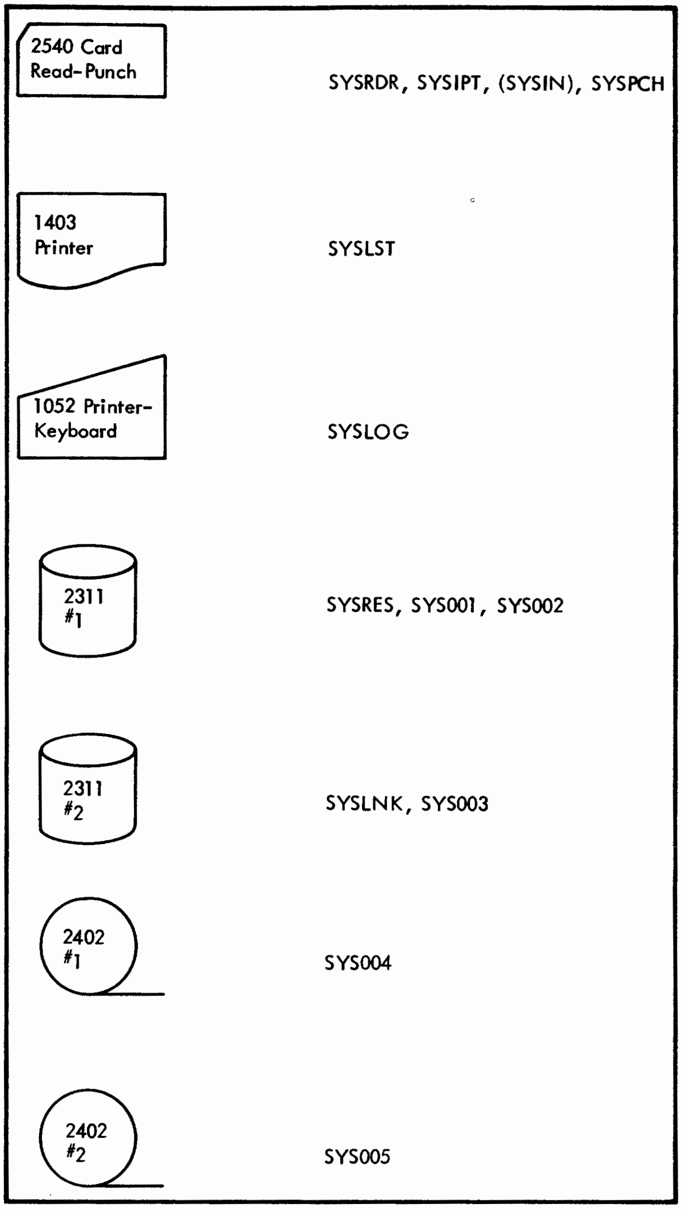


Figure 10. Example of Symbolic Device Assignment

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 or 2319.

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/ 2319
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	$\frac{1}{20 \times 7}$	$\frac{1}{29 \times 11}$

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 submited 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 tc 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, Formats, and Usage of Commands and Statements. See the Data Management Concepts publication listed in the Preface for a complete discussion of volume and file labels.

JOB ACCOUNTING INTERFACE

Job Accounting Interface, a function of the Disk Operating System (DOS) Supervisor, accumulates job information to:

1. Charge usage of the system.

2. Help supervise system operaticn.

3. Use the system more efficiently (especially the I/O devices on the system).

4. Plan new applications, additional devices, and new systems.

Information accumulated for each job step consists of:

- Job name
- User information
- Partition ID
- Cancel code
- Record type
- Date
- Job start time
- Phase name (from EXEC card)
- High core used (from communications region)
- CPU time
- Overhead time
- Stop time (at EOJ only)
- All bound time
- SIO count (optional)

Note: If the CPU is not equipped with a timer, time fields are zero.

Information that the Job Accounting Interface accumulates is kept in the Job Accounting Table (see Figure 11). To use this information, you must write a program that accesses and processes this table. This could involve either formatting and printing, or storing on a secondary device.

Two new parameters, JA and JALIOCS, are added to the FOPT macro. These generate support for the Job Accounting Interface functions. When the JA parameter is included in the FOPT macro, job accounting information is accumulated in the Job Accounting Table.

If the JALIOCS parameter is also included, two more areas can be reserved in the supervisor. A default of 16 bytes is reserved for a user save area. However from 0 to 1024 bytes can be specified. An alternate label area (necessary for LIOCS label processing under Job Accounting Interface) can also be generated.

How to Use the Job Accounting Interface

The two methods of using the information accumulated by the Job Accounting Interface functions of DOS are:

1. Using Logical IOCS (LIOCS) with no label processing, or Physical IOCS (PIOCS).

2. Using LIOCS where label processing is necessary.

Using LIOCS with No Label Processing, or PIOCS: Include only the JA parameter in the FOPT macro in the supervisor assembly. Provide your job accounting program (called at the end of each job step and at end of job). Your program could involve either formatting and printing or storing on a secondary storage device.

Using LIOCS Where Label Processing Is Necessary: Include both the JA and the JALIOCS parameters (with a large enough label area to accommodate the contents of the LBLTYP card) in the FOPT macro in the supervisor assembly.

GENERATING A SUPERVISOR: When only the JA parameter is included, you can write your program using either LIOCS with no label processing or PIOCS. If you use LIOCS with label processing, you must include both the JA and the JALIOCS parameters. JALIOCS reserves space in the supervisor for a user save area and if specified reserves an alternate label area in the supervisor. This label area is necessary when using LIOCS with label processing.

> Limitations: If the JA parameter is specified and the JALIOCS parameter is not specified, the alternate label area necessary for LIOCS label processing is not reserved. Your job accounting program must then use a device or method that does not require LIOCS label processing.

Job accounting information is accumulated whenever you include the JA parameter in the FOPT macro.

The JA parameter generates Job Accounting Interface support in your supervisor. Information accumulates for items 1 through 14 in Figure 11.

If desired, the JA parameter can also generate Job Accounting Interface support in your supervisor to count the Start I/Os (SIOs) issued to each I/O device. For the information accumulated, see items 15 and 16 in Figure 11. The values for SIO count specify the number of I/O devices for which an SIO count is desired in the background, foreground 2, and foreground 1 programs. Each value may be in the range 0-255. If any number is omitted, zero is assumed for that partition. Any other specification is invalid and defaults to JA=NO.

> Note: The numbers specified for SIO count are independent of the system generation option of devices attached to the system. If more I/O devices than you specify for SIO count are used by the partition, accounting information for the extra devices in that partition is not accumulated.

With JALIOCS you can specify the number of bytes needed for your save area (located in the supervisor). This save area is for your use only and is not accessed by the Job Accounting Interface program. You can use this area to save DTF information or for any other purpose. It can be any number in the range 0-1024. If it is omitted, or if JALIOCS is not specified, 16 bytes are still reserved in the supervisor.

If your Job Accounting Interface program is written in PIOCS, you might use this area to keep track of the current record and extent on the DASD device that you are using as external storage for your job accounting information.

If your program is written in LIOCS, you should save DTF information (status switches, extent information, pointers) in this area. If you can afford the extra bytes in the supervisor, you might want to save the entire DTF. Also, if you use more than one DTF, you must save information from each.

Also with JALIOCS, you can specify the number of bytes you will need for your label area. This label area is located in the supervisor. It can be any number between 0 and 224. This number is usually the amount of main storage that is reserved by the LBLTYP statement for your output program. If it is omitted, the label area is not reserved for your output program.

System Considerations: Job Accounting Interface runs in conjunction with DOS, and can be used on any CPU of at least 24K supported by DOS.

If JALIOCS is specified, JA must also be specified. If JA is not specified, or if JA=NO, JALIOCS is ignored.

When the JA parameter is included in the FOPT macro, timer support is included, even if timer support has not already been specified (TIMER=YES in the CONFG macro). If the CPU does not have a timer installed, time fields in the account table are zero.

The Job Accounting Table is not generated for foreground partitions unless the MPS=BJF parameter is specified in the SUPVR macro. Tables for foreground partitions cannot be accessed when the system is running in Single Program Initialization (SPI) mode.

PREPARING JOB CARDS: You can put 16 characters of user information after the job name on the job card. One blank column must follow the job name, followed by up to 16 characters to identify the job. The system functions place the information (unaltered) in the Job Accounting Table.

USING ACCUMULATED RECORD OUTPUT DATA: Information accumulated by Job Accounting Interface functions of DOS is kept in the Job Accounting Table.

To use the information in the Job Accounting Table, you must write a program that either formats and prints the table or stores the table on secondary storage. Your job accounting program must be self-relocating and cataloged in the Core Image Library under the name $JOBACCT. While this program can be up to 4,096 bytes, for more efficient loading, it should not exceed one Core Image Library block (1,728 bytes on a 2311, or 1,688 bytes on a 2314 or 2319). With the one-block length, only one LOAD is required to get the routine into main storage. Because $JOBACCT is called in at the end of each job step, it should not attempt very extensive functions if you want to hold timing degradation to a minimum.

> Note: If you do not provide a routine, a dummy routine ignores the Job Accounting Table.

As each job step ends, $JOBACCT is called into main storage. All registers are stored before your program is called, and several registers contain information that you can use:

- Register 11. Length of the Job Accounting Table.
- Register 12. Base register for $JOBACCT.
- Register 13. Address of a save area in the supervisor. This save area is either 16 bytes long or the length specified by s in the JALIOCS parameter.
- Register 14. Link register. This register provides an exit from $JOBACCT. After this program has executed, it must return via register 14 (BR 14).
- Register 15. Job Accounting Table address. Provides the location of the Job Accounting Table.

Register 14 must have the same contents at exit from $JOBACCT as it did upon entry to the routine. LIOCS, for example, uses registers 14, 15, 0, 1, and sometimes 13. You should save and restore these registers as needed.

$JOBACCT is called after each job step because items 4, 5, and 10 through 16 in the Job Accounting Table are cleared between job steps. If you do not save this information, it will be lost. EOJ information (stop time) is included in the table when the last job step has terminated. Deletion of JOB and/or /& cards result in incorrect start and stop time information.

Before $JOBACCT is called, the system checks to ensure that only one accounting program is executing at any one time.

ERROR CONDITIONS: If an error occurs that cancels $JOBACCT, job accounting information is no longer available and your program is not called again until the system is re-IPLed. Normal job processing continues with the next job. You can use the STXIT option to inform the operator that an error occurred in $JOBACCT rather than in the problem program. If you use this option, reset it by specifying

$$\text{STXIT} \begin{Bmatrix} \text{AB} \\ \text{IT} \\ \text{PC} \\ \text{OC} \end{Bmatrix}$$

before taking an exit from $JOBACCT.

Item	Displacement	Byte Length	Contents
1	0-7	8	Job Name. 8-byte character string taken from JOB card.
2	8-23	16	User Information. 16 characters of information taken from the JOB card.
3	24-25	2	Partition ID. BG, F2, or F1.
4	26	1	Cancel Ccde. (See Figure 12)
5	27	1	Type of Record. S = Job Step L = Last step of job
6	28-35	8	Date. mm/dd/yy or dd/mm/yy depending on supervisor option.

7	36-39	4	Job Start Time. 0hhmmssF where:
			h = hours
			m = minutes
			s = seconds
			F is the sign (in packed decimal format).

8	40-43	4	Stop Time. Zeros except in last record, which has job stop time in same format as start time.
9	44-47	4	Reserved.
10	48-55	8	Phase Name. 8-byte character string taken from the EXEC card.
11	56-59	4	High Core. Hex address of uppermost byte of any program fetched or loaded (taken from communications region).
12	60-63	4	CPU Time. 4 binary bytes given in 300ths of a second. Time is calculated from exit of the user-written routine called during job control to the next entry of the routine. Time used by the user-written job accounting routine is charged to overhead of the next record.

Figure 11. Job Accounting Table (Part 1 of 2)

Item	Displacement	Byte Length	Contents
13	64-67	4	Overhead Time. 4 binary bytes given in 300ths of a second. Includes time taken by functions that cannot be charged readily to one partition (attention routine, error recovery, etc). This time is divided by the number of active batch partitions and recorded in each accounting table.
14	68-71	4	All Bound Time charged to this partition. 4 binary bytes given in 300ths of a second. This is the time the system is in the wait state divided by the number of partitions running.
15	72-?	Varies	SIO Tables. 6 bytes for each device specified by the JA parameter. First 2 bytes are X'0cuu', next 4 are hex count of SIOs for step. Unused entries contain X'10' followed by 5 bytes of zeros. Stacker select commands for MICR are not counted. Error recovery SIOs are not charged to the Job Accounting Table. Devices are added to the table as they are used.
16		1	Overflow (normally X'20'). Set to X'30' if more devices are used than the number set by the JA parameter at supervisor assembly time.

Note: The difference between Start and Stop times does not necessarily equal the sum of CPU, All Bound, and Overhead times. All Bound and Overhead times vary depending upon the number of active partitions and the type of partition activity. CPU time is accurate for each partition, but it may not be reproducible (recreatible). That is, the same job being executed under different considerations (such as a varying number of active partitions, logical transient availability, etc) may show differences in CPU time.

Figure 11. Job Accounting Table (Part 2 of 2)

Cancel Code (hex)	Cause
10	Normal EOJ
17	Program Request. Same as 23 but causes dump because subtasks were attached when maintask issued CANCEL macro.
18	Elminates cancel message when maintask issues DUMP macro with subtasks attached.
19	I/O operator option.
1A	I/O error.
1B	Channel failure.
1C	CANCEL ALL macro issued.
1D	Maintask termination.
1E	Unknown ENQ requestor.
1F	CPU failure.
20	Program check.
21	Illegal SVC.
22	Phase not found.
23	Program request.
24	Operator intervention.
25	Invalid address or insufficient core allocation to partition.
26	SYSxxx not assigned. (unassigned LUB code)
27	Undefined logical unit. (invalid LUB code in CCB)
28	QTAM cancel in progress.
30	Read past /& on SYSRDR or SYSIPT.
31	I/O error queue overflow. (error queue overflow or no CHANQ entry available for ERP)
32	Invalid DASD address (disk).
33	No long seek (disk).
34	I/O error during fetch. (unrecoverable I/O error during fetch of non-$ phase)
35	Job control open failure.
40	Load $$BEOJ.
80	Cancel occurred in Logical Transient Area (LTA).
FF	Unrecognized cancel code, or, if the system is placed in the wait state and no further processing is done by the terminator, supervisor catalog failure.

In addition to the above, the same cancel codes are recognized with the 0-bit (X'80') on.

Figure 12. DOS Cancel Codes

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.

RDE DATA ENTRY FOR SYSTEM/370

During the processing of the first // JOB statement after IPL, RDE users must provide additional information about the system. Message 1I90D IPL REASON CODE = is issued on SYSLOG. you must respond to message 1I90D with a Reason Code followed by END. The Reason Codes are:

IPL Reason Code	Comments
CE	IBM CE/SE has control of the system and is not doing user work.
DF	Default
EN	Environmental problem (such as: power, overheating, etc) caused failure.
IE	IBM hardware or a IBM-supplied program error that did not require an IBM CE/SE.
IM	IBM hardware of IBM-supplied program error that required an IBM CE/SE.

ME Media. Hardware error caused by a faulty disk pack, reel of tape, cards, etc.

NM Normal IPL.

OP Operational problem. Operator error or procedural problem.

UN Unknown. Undetermined error.

UP A user (non-IBM-supplied) program caused the failure.

If a Reason Code is not entered (only the END key is pressed) or SYSLOG is down or SYSLOG is not assigned to a 3210 or 3215, then the default, DF, is assumed. However, if an invalid code is entered, message 1I89I is issued and message 1I90D is reissued until a valid response is made.

After the Reason Code is entered, message 1I91D SUB-SYSTEM ID = is issued. You must respond to message 1I91D with one of the ID codes allowed by END. The ID codes further identify the reason for performing IPL. The ID codes are:

ID Codes	Comments
00	Unknown. Must be used with Reason Codes DF, EN, NM, OP, UN and UP. 00 is the default.
10	Processor. CPU, channel (integrated), storage unit, etc failure.
20	DASD. A failure occurred in a DASD unit or its associated control unit (2311, 2314, 2841, etc).
30	Other. A device without an ID code (such as a paper tape unit) caused the failure.
40	Magnetic Tape. A failure occurred in a magnetic tape unit or its associated control unit (2401, 2803, 3400, etc).
50	A failure occurred in a card reader/punch, a printer or in its associated control unit (2540, 1403, 2821, etc).
60	MICR/OCR. A magnetic ink character reader (1412, 1419, etc) or an optical character reader (1285, 1287, etc) failure.
70	A teleprocessing failure occurred in a teleprocessing control unit (2701, 2702, etc).
80	Graphic. A video display unit (2260, etc) or its associated control unit failure.
90	An IBM-supplied Type 1 or Type 2 program (such as the DOS system or one of its components) failure.
91	An IBM Programming Product failure.

If the ID code is not entered (only the END key is pressed) or SYSLOG is down or SYSLOG is not assigned to a 3210 or 3215, then the default, 00, is assumed. However, if an invalid ID code is specified, message 1I89I is issued and message 1I91D is repeated until a valid response is made.

Notes:

1. Always use ID code 00 with Reason Codes DF, EN, NM, OP, UN, and UP.
2. ID codes 10, 20, 30, 40, 50, 60, 70, 80, 90, and 91 should be used with Reason Codes CE, IE, IM, and ME.

After the RDE IPL information is specified, normal processing continues. When you have concluded your processing for any period, before shutting down your DOS system, issue a ROD command to write the RDE EOD (End-of-Day) record on SYSREC and ensure that no environmental data is lost.

Job Control Statements (JCS)

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, DLBL, 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 C.

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 communications 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 linkage-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. Ignored by Job Control if SYSLST is not assigned.

MTC Statement: Controls operations on logical units assigned to IBM 2400/3420 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 DLBL and EXTENT statements for a private core image library are in the input stream (if the information is not contained on the label cylinder), they must precede the ASSGN SYSCLB statement. 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, Formats, and Usage of Commands and Statements

This section contains descriptions formats, and usages 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 heading, of the programs or routines that accept it. These programs or routines are identified as follows:

Job Control Statement - JCS
Job Control Command - JCC
Single-Program Initiator - SPI
Attention routines - AR

Figure 13 contains the commands and statements grouped by function and also indicates the programs or routines that accept them.

Job Control statements (JCS), with the exception of /*, /&, and *, contain two slashes (//) in columns 1 and 2 to identify them. Job Control commands (JCC), ATTN commands (AR), and single-program initiation commands (SPI) 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.

Job Control statements (JCS), 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 (JCC) 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 (AR) can be issued at any time by pressing the request key on SYSLOG. Some of these commands can be issued only in a multiprogramming environment.

- Single-Program Initiation commands (SPI) may only be issued in a multiprogramming environment following the Attention command START [F1 or F2].

Type of Command or Statement	Name	Accepted by			
		JCS	SPI	AR	JCC
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	X
	READ		X		
	START			N3	
	STOP				X
	UNBATCH				N4
Setting System Parameters	ALLOC			X	X
	SET				X
	TIMER		X	X	
Operator Communications	ALTER			X	
	DSPLY			X	
	DUMP			X	
	EOB or END		X	X	X
	LOG		X	X	X
	MSG		N3	N3	
	MODE			N6	
	NOLOG		X	X	X

Figure 13. Commands and Statements by Function (Part 1 of 2)

Type of Command or Statement	Name	Accepted by			
		JCS	SPI	AR	JCC
Control of I/O System	ASSGN	X	X		X
	CLOSE	X			X
	DVCDN				X
	DVCUP				X
	HOLD		X		N3
	LISTIO	X	X		X
	MAP		X	X	X
	MTC	X			X
	RELSE		X		N3
	RESET	X			X
	ROD				X
	UCS		X		X
	UNA		X		N3
Execution of Program	EXEC	X	X		
	RSTRT	X			

Notes

N1 If MPS=BJF and PCIL=YES are specified during system generation OPTICN STDLABEL is available only in the background partition.

N2 Valid only if the batched-jcb 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 cn SYSIPT and cancels any program that reads past /&.

N6 Valid only for IBM System/370.

Figure 13. Commands and Statements by Function (Part 2 of 2)

ALLOC

The ALLOC command (Allocate Main Storage) 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.

JCC and AR Format

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

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 mode] to less than 10K bytes.

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

ALTER

The ALTER command allows the operator to alter 1 to 16 bytes of main storage, starting at the specified hexadecimal address, through the ccnsole printer-keyboard. Two characters (0-9,a-f) must be entered to change each byte; these characters represent the hexadecimal equivalent of the information to be stored.

AR Format

```
  ALTER xxxxxx
```

xxxxxx   The six digit hexadecimal address, with leading zeros if necessary, to start main storage alteration.

ASSGN

The ASSGN command or statement (Assign Logical Name) assigns a logical I/C unit to a physical device. All device assignments made for single-program foreground programs are canceled either by another ASSGN tc the same unit or at the end of each prcgran, 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.

JCS Format

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

JCC Format

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

SPI Format

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

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, 2314, or 2319) 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 JCS and JCC:

SYSRDR
SYSIPT
SYSIN
SYSPCH
SYSLST
SYSOUT
SYSLNK
SYSLOG
SYSSLB
SYSRLB
SYSREC
SYSCLB (not valid for JCS)
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 permanently unassigned. It is not possible to change the assignment of SYSLOG while a foreground partition is active.

If SYSLNK is assigned to a foreground partition(s), SYSCLB must also be assigned to the same partition(s). Whenever the DLBL and EXTENT information for SYSCLB changes, SYSCLB must be reassigned.

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 For FORTRAN, RPG, and certain American National Standard COBOL problem programs, the IGN option unassigns the specified logical unit, and any subsequent logical IOCS command (OPEN, GET, PUT etc) issued for that unit to be ignored.

This allows you to disable a logical unit that is used in a program without removing the code for that unit. You can then execute the program as if the unit did not exist. This may be especially helpful when debugging a program.

For Assembler language problem programs, IGN indicates that the logical unit is to be ignored. With files processed by logical IOCS, the OPEN to the file is ignored, the DTF table is not initialized (e.g. IOREG, extent limits), and the IGNORE indicator is set on in the DTF table. It is your responsibility to check this indicator and bypass any I/O commands (GET, PUT, etc) for this file.

The IGN option is not valid for SYSRDR, SYSIPT, SYSIN, and SYSCLB, nor for PL/I (D) and COBOL programs.

Additional information about ignore is in the OPEN (R) section of the Supervisor and I/O Macros manual listed in the Preface. IGN restrictions for American National Standard COBOL users are given in the COBOL Programmer's Guide that is also listed in the Preface.

X'ss' Device specifications (used to specify mode settings for 7-track and 9-track tapes). If X'ss' is nct 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. For 800 BPI single density 9-track tape, a specification of C8 reduces the time required to OPEN an output file.

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, you 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. You 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 TEMP is only valid for JCC and it indicates the assignment for the logical unit is to 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. TEMP is not invalid for SYSCLB.

BATCH

The BATCH command (Start or Continue Batched-Job Operation) 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 an EOB or END.

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.

AR Format

$$\text{BATCH} \left\{\begin{array}{l}\underline{\text{BG}}\\ \text{F2}\\ \text{F1}\end{array}\right\}$$

If the operand is omitted, BG is assumed.

CANCEL

The CANCEL command (Cancel Job), 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.

JCC and SPI Format

```
CANCEL blank
```

AR Format

```
       (BG)
CANCEL {F1}
       (F2)
```

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

The CLOSE command (Close Output Unit) 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, 2314, or 2319. 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*.

JCS Format

```
                 [(,X'cuu'[,X'ss'])]
// CLOSE SYSxxx  [),UA             (]
                 [),IGN            (]
                 [(,ALT            )]
```

JCC Format

```
              [(,X'cuu'[,X'ss'])]
CLOSE SYSxxx  [),UA             (]
              [),IGN            (]
              [(,ALT            )]
```

SYSxxx *For the CLOSE command only*: For 2311, 2314, or 2319: 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 tc 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

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

JCS Formats

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

```
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, except for DASD output files which use the date from the SET command. 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

The DASD-label information 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. For a detailed discussion of DLAB, see the DOS DASD Labels manual that is listed in the Preface.

JCS Format

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

SPI Format

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

'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

The DASD-label information 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 theprevious VOL, DLAB, and XTENT combinations will be continued.) For a detail discussion of DLBL see the <u>DOS DASD Labels</u> manual listed in the <u>Preface</u>.

<u>JCS Format</u>

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

<u>SPI Format</u>

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

filename

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

'file-ID'

The unique 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, <u>filename</u> 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 (based on the date entered via the SET command) 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.

data security

A three-character field indicating that a data secured file is to be created or processed. DSF must be specified for a data secured <u>output</u> file. If it is omitted for an output file, an unsecured file is created.

This operand is not required for an <u>input</u> file, and it does not invoke data security if the file was not originally created as a data secured file.

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

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.

DSPLY

The DSPLY command allows the operator to display 16 bytes of main storage, starting at the specified hexadecimal address, on the console printer-keyboard. Two characters (0-9,a-f) are printed for each byte of information; these characters represent the hexadecimal equivalent of the current information in main storage.

AR Format

```
  DSPLY xxxxxx
```

xxxxxx The six digit hexadecimal address, with leading zeros if necessary, tc start the main storage display.

DUMP

The DUMP command allows the operatcr tc print logical or physical areas of main storage on SYSLST. The SYSLST used may be the one assigned to any partiticn, but it must be printer and it should not be used by the partition (interspersed partition and dump output will result).

AR Format

```
DUMP ⎛blank          ⎞ ⎧(BG)⎫ *
     ⎜S              ⎟ ⎨(F1)⎬
     ⎜BG             ⎟ ⎩(F2)⎭
     ⎜F1             ⎟
     ⎨F2             ⎬
     ⎜BGS            ⎟
     ⎜F1S            ⎟
     ⎜F2S            ⎟
     ⎜CEAREA         ⎟
     ⎝xxxxxx,xxxxxx  ⎠
```

*This opticnal operand indicates which partition has the SYSLST to be used for output. If omitted, the BG SYSLST is used.

blank   If the dump area operand is omitted, all of main storage, except the supervisor, and the associated registers are printed.

S   All of main storage and the associated registers are printed.

BG
F1
F2   The applicable partition and its associated registers are printed.

BGS
F1S
F2S   The applicable partition and its associated registers, and the supervisor are printed.

CEAREA   The CE table, CE area, and the Alternate Address Area, if present, are printed (see Core Wrap Mode in the PDAIDs section).

xxxxxx,xxxxxx
Main storage, with the associated registers, starting and ending at the specified hexadecimal addresses is printed. If the starting address is not on a fullword boundary, the address is rcunded down to the first fullword boundary; if the ending address is nct on a fullword boundary, the address is rounded up to the first fullword boundary. A minimum of one fullword is dumped, beginning at the start address.

DVCDN

The DVCDN command (Device Down) informs the system that a device is no lcnger physically available for system operations. If a tempcrary 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, issue a CLOSE command for any system logical units currently assigned to it before issuing DVCDN. 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 is 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.

<u>JCC Format</u>

```
DVCDN X'cuu'
```

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

DVCUP

The DVCUP command (Device Up) 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.

<u>JCC Format</u>

```
DVCUP X'cuu'
```

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

EOB or END

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.

> <u>Note</u>: CANCEL (without an operand), EATCH and START commands automatically terminate ATTN commands.

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 SYSLOG. Further single program initiation commands can then be read from SYSLOG. The end-of-communication command causes input reading to switch back to the device specified in the READ command.

<u>JCC, AR, and SPI Format</u>

```
EOB or END blank
```

EXEC

The EXEC command or statement (Execute Program) 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 a 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.

<u>JCS Format</u>

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

<u>SPI Format</u>

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

progname Represents the name of the program in a 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 (0-9, A-Z, ., #, $, and @) characters.

If the program to be executed has just been processed by the Linkage Editor, the operand of the EXEC statement is blank.

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

The EXTENT command or statement (DASD Extent Information) 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 input files for Sequential Disk on a 2311, 2314, or a 2319, provided the DEVADDR parameter has been specified in the DTF table. For a detail discussion of EXTENT, see the DOS DASD Labels manual listed in the Preface.

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

JCS Format

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

SPI Format

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

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. (To avoid right-justification and padding use // VOL, // DLAB, and // XTENT for DASD information.)

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 you specify 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 nct checked and it is your 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.

Note: For index sequential files, the extent information should be entered in the following order:

1. Master index sequence number 0 and type 4

2. Cylinder index sequence number 1 and type 4

3. Prime data area sequence number 2 - n and type 1

4. Independent overflow area sequence number n + 1 and type 2

Where n is the sequence number of the last prime data area extent.

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.

When using split cylinder files, this parameter designates the beginning of the split as well as the first track of the file.

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

<u>Actual to Relative</u>

2311	10 x cylinder number + track number = RT
2314/ 2319	20 x cylinder number + track number = RT
2321	1000 x subcell number + 100 x strip number + 20 x cylinder number + track number = RT

<u>Relative to Actual</u>

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

2314/ 2319 $\frac{RT}{20}$ = quotient is cylinder, remainder is track

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

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

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

<u>Example</u>: 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 or 2319. The number of tracks for a split cylinder file must be the product of the number of cylinders for the file and the specified number of tracks per cylinder for that file.

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

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

JCC and SPI Format

$$\text{HOLD} \begin{Bmatrix} \text{F1[,F2]} \\ \text{F2[,F1]} \end{Bmatrix}$$

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 a HOLD command is issued prior to an UNBATCH command, the device assignments stay in effect after the UNBATCH command is issued.

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

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

JCS Format

```
// JOB jobname [accounting information]
```

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.

accounting information
If the job accounting interface has been specified during system generation, 16 characters of user specified accounting information can be entered in the job statement. This information is moved to the Job Accounting Table (see Figure 11). If accounting information is specified, it must be separated from the job name by a blank. If the job accounting interface is not specified during system generation, accounting information is ignored.

LBLTYP

The LBLTYP statement (Reserve Storage for Label Information) defines the amount of main storage to be reserved at linkage-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 immediately preceding the // EXEC LNKEDT statement, with the exception of self-relocating programs, for which it is instead submitted immediately 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.

JCS Format

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

SPI Format

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

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

The LISTIO command or statement (List I/O Assignment) causes the system to print a listing of I/O assignments. The listing appears on a SYSLOG (command) or SYSLST (statement). If SYSLST is not assigned, the LISTIO statement is ignored.

JCS Format

```
             { SYS    }
             { PROG   }
             { F1     }
             { F2     }
   // LISTIO { ALL    }
             { SYSxxx }
             { UNITS  }
             { DOWN   }
             { UA     }
             { X'cuu' }
```

SPI Format

```
             { BG  }
             { F1  }
      LISTIO { F2  }
             { UA  }
             { ALL }
```

JCC Format

```
             { SYS    }
             { PROG   }
             { F1     }
             { F2     }
   LISTIO    { ALL    }
             { SYSxxx }
             { UNITS  }
             { DOWN   }
             { UA     }
             { X'cuu' }
```

SYS Lists the physical units assigned to all system logical units. (See Note)

PROG Lists the physical units assigned to all background programmer logical units. (See Note)

BG Lists the physical units assigned to all background logical units. BG is valid for SPI only.

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. (See Note)

UNITS Lists the logical units assigned to all physical units. (See Note)

DOWN Lists all physical units specified as inoperative. (See Note)

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

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

Note: SYS, PROG, SYSxxx, UNITS, and DOWN are invalid for SPI.

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 14. 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 14. Example of LISTIO SYS Output

LOG

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.

JCC, AR, and SPI Format

```
LOG blank
```

The operand field is ignored by the system.

MAP

The MAP command (Map Main Storage) 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.

JCC, AR, and SPI Format

```
MAP blank
```

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 NC NAME is specified when no job card is entered, but the program is active.

MODE

The MODE command provides these options for controlling soft MCI:

- The mode that the system is operating in (the status of the system) can be requested.
- The mode of operation can be changed from quiet to recording, or from recording to quiet.
- An EFL threshold value can be specified to override the IBM-supplied value.
- The MODE command can also be used to place the Model 145 Control Storage ECC in threshold mode.

The MODE command is a notational command. Operands of the MODE command can be entered in any order and must be continuous (that is, no blanks are allowed between or within operands). The STATUS operand cannot have any other operands before or after it.

The total length of the MCDE command must not exceed 30 characters.

<u>AR Format</u>

$$\text{MODE}\left\{\begin{array}{l}\text{R}\\ \text{STATUS}\\ \begin{array}{l}\text{HIR}\\ \text{ECC}\end{array}\left[,\left\{\begin{array}{l}\text{M}\\ \text{C}\end{array}\right\}\right]\left\{^{*}\left[,\left\{\begin{array}{l}\text{R}\\ \text{Q}\\ \text{TH}\end{array}\right\}\right]\text{[,E=eeee][,T=tttt]}\right\}^{*}\end{array}\right\}$$

*Note: When HIR or ECC is specified, at least one of the optional operands within these braces must be selected. TH is only valid for the Model 145 when ECC,C is specified with the MODE command.

The meanings of the operands are:

Operand	Meaning
STATUS	A report is printed on SYSLOG, showing: • Each particular facility (HIR, ECC). • System mode of operation. • Current error count. • Error count threshold. • Current elapsed time. • Time threshold. • Number of buffer pages deleted.

The status report formats are:

$$\text{HIR,}\left\{\begin{array}{l}\text{R}\\ \text{Q}\end{array}\right\}\text{,aaaa/eeee,bbbb/tttt}$$

For the Model 145

$$\text{ECC,}\left\{\begin{array}{l}\text{R}\\ \text{Q}\end{array}\right\}\text{,}\left\{\begin{array}{l}\text{M}\\ \text{C}\end{array}\right\}\text{,aaaa/eeee,bbbb/tttt}$$

Fcr the Model 155

$$\text{ECC,}\left\{\begin{array}{l}\text{R}\\ \text{Q}\end{array}\right\}\text{,aaaa/eeee,bkbb/tttt}$$

BUF DLT=XXX

where:

aaaa	=	Current error count.
eeee	=	Error count threshold.
bbbb	=	Current elapsed time.
tttt	=	Time threshold.
XXX	=	Total number of inoperable buffer pages deleted.

A buffer page is a 32-byte work area in control storage that is used by the Model 155 hardware program.

HIR Hardware Instruction Retry. This operand changes the mode of the HIR facility to R or Q, and/or modifies the error count threshold and/or time threshold.

> Note: When HIR is placed in quiet mode, ECC also goes into quiet mode.

ECC Error Correction Code. This operand changes the mode cf the ECC facility to R or Q, and/or modifies the error count threshold and/or time threshold. If ECC is specified for a Model 145, then M or C must also be specified. ECC can also place the Model 145 Control Storage in threshold mode.

Note: Use of the Error Correction Code (ECC) in full recording mode may cause severe system degradation. Thus, the

$$[ECC,\left\{\begin{matrix}M\\C\end{matrix}\right\},R]$$

operand combination of the MODE command should be used only by the customer engineer or at his request.

R When used:

- Alone: Places both HIR and ECC in recording.
- With HIR: Places HIR in recording mode.
- With ECC on the Model 155: If HIR is in recording mcde, it places ECC in recording mode.
- With ECC,M on the Model 145: If HIR is in recording mode: Places Model 145 main storage in recording mode.
- With ECC,C on the Model 145: If HIR is in recording mode: Places Model 145 control storage in recording mode.

Q When used:

- With HIR: Places both HIR and ECC in quiet mode.
- With ECC on the Model 155: Places ECC in quiet mcde.
- With ECC,M on the Model 145: Places Model 145 main storage in quiet mode.
- With ECC,C on the Model 145: Places Model 145 control storage in quiet mode.

M or C Main or Control storage: M or C is only valid for the Model 145. M or C must be specified when ECC is specified for the Model 145. M indicates main storage and C control storage.

TH Threshold Mode: TH is only valid fcr the Model 145 is ECC,C is specified. TH places the Model 145 control storage ECC in threshold mode.

E=eeee
T=tttt Values entered for E and T must be included in the following decimal ranges:

E - 8 (initial value) through 9999
T - 8 (initial value) through 9999

The IBM-supplied value is 8.

Ncte: Whenever HIR is in quiet mcde, ECC mode must not be changed.

MSG

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

AR and SPI Format

$$\text{MSG}\ \left\{\begin{matrix}\text{F1}\\\text{F2}\end{matrix}\right\}$$

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 SYSLOG informing the cperator cf this condition.

MTC

The MTC command or statement controls IBM 2400/3420 series magnetic tape operations. The first operand specifies the operation to be performed.

JCS Format

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

JCC Format

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

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). X'CUU' is not valid for JCS.

The optional third entry, nn, is a decimal number (01-99) representing the number of times the specified operation is to be performed. If nn is not specified, the operation is performed once.

NOLOG

The NOLOG command (Suppress Logging) 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.

JCC, AR, and SPI Format

```
NOLOG blank
```

The operand field is ignored by the system.

OPTION

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

JCS Format

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

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. (See Note 1.)

NOLOG — Suppresses the listing of all valid control statements on SYSLST until a LOG option is encountered. If SYSLST is assigned, invalid statements and commands are listed. (See Note 1.)

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 accepted by Job Control operating in a foreground partition if MPS=BJF and PCIL=YES is specified during system generation and a private core image library is assigned.

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), FORTRAN IV, and American National Standard COBOL compilers, and the Assembler D 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 and the American National Standard COBOL compiler 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 accepted by Job Control operating in a batched-job foreground environment if MPS=BJF and PCIL=YES is specified during system generation and a private core image library is assigned.

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. (See Note 2.)

USRLABEL Causes all DASD or tape labels submitted after this point to be written at the beginning of the user label track. (See Note 2.)

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. (See Note 2.)

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

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

SYSPARM= 'string' Specifies a value for the Assembler system variable symbol $SYSPARM. $SYSPARM gets the value of the string, which is enclosed by quotes. The string can contain 0-8 EBCDIC characters. One internal quote must be represented by two. (Job Control removes one of them when setting the value.) The surrounding quotes are not included and the length of $SYSPARM is determined by the resulting string.

This operand is invalid if SYSPARM support was not specified when the system was originally generated.

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 1: 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.

Note 2: Refer to the DASD and Tape Labels publications listed in the Preface for additional information about STDLABEL, USRLABEL, and PARSTD.

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, SYSLOG is unlocked for message input. EOB or END causes processing to continue. The PAUSE statement or command is always printed on SYSLOG. If a 1052 or a 3210 or 3215 is not 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.

JCS Format

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

JCC and SPI Format

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

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.

AR Format

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

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

READ

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.

SPI Format

```
  READ X'cuu'
```

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

The RELSE command (Release Foreground Unit Assignments at EOJ) 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.

JCC and SPI Format

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

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

The RESET command or statement (Reset I/O Assignments) 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.

JCS Format

```
           (SYS    )
  // RESET )PROG   (
           )ALL    (
           (SYSxxx )
```

JCC Format

```
        (SYS    )
  RESET )PROG   (
        )ALL    (
        (SYSxxx )
```

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

The ROD command (Record on Demand) 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 also initializes the writing of the RDE End-of-Day (EOD) record on SYSREC for System/370. The ROD command has no operand.

JCC Format

```
ROD  blank
```

RSTRT

A Restart Checkpointed Program 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.

JCS Format

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

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, 2314, or 2319 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, 2314, or 2319 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

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 2715 Transmission Control Errors) and either the machine check recording and recovery (MCRR) or the recovery management support (RMS) features. The SET card should precede the JOB card in job control sequence.

JCC Format

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

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 tc 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 cperator 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.

$\text{RF=}\left\{\begin{array}{l}\underline{\text{YES}}\\\text{NO}\\\text{CREATE}\end{array}\right\}$

Defines to the system the status of the recorder file (IJSYSRC) on SYSREC used by the I/O error logging (OBR/SDR and 2715 Transmission Control Error) and either the machine check recording and recovery (MCRR) or recovery management support (RMS) 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. NO is not valid if a System/370 CPU is specified during system generation.

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

START

The START command (Start Background or Foreground Processing) can be used to initiate a single-program foreground program or to resume batched-job processing in any partition.

AR Format

```
        (BG)
  START {F1}
        (F2)
```

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 SYSLOG informing the operator of the condition.

STOP

The STOP command (Stop Batched-Job Processing) 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.

JCC Format

```
  STOP blank
```

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 it up until the operator responds.

TIMER

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

AR and SPI Format

```
        (BG)
  TIMER {F1}
        (F2)
```

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

The Tape Label Information 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. For detail information about TLBL refer to the Tape Labels manual in the Preface.

JCS Format

```
  // 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],
```

SPI Format

```
[//] 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]
```

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

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

'file-ID'
One to seventeen alphameric characters, contained within apostropes, indicating the unique 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
Output Files: A one to four numeric character retention period in the format d through dddd (0-9999) can be specified. If omitted, a 0 retention period is assumed. The current system date is always used as the creation date for output files.

Input Files: A four to six numeric character creation date in the format yy/ddd (99/365) can be specified (ddd can be from 1-365). If omitted or a retention date is specified, no checking is done for input files.

file serial number (EBCDIC) or
set identifier (ASCII)
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 (EBCDIC) or
file section number (ASCII)
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/360bb used as the system code.

TPLAB

This Tape Label Information 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. The Tape Labels manual listed in the Preface contains additional information about TPLAB.

JCS Format

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

SPI Format

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

'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

The UCS command (Load Universal Character Set Buffer) 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, through the UCS buffer and the many chains/trains available, the 1403 Printer can be adapted to many variable printing applications.

The logical unit must be assigned to an IBM 1403 Printer with the UCS feature. It is the user 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 Appendix C.)

JCC and SPI Format

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

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 a 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:

240-character UCS buffer load	80-character verification message

UNA

The UNA command immediately causes all I/O assignments for the single program foreground area(s) specified tc be set to unassigned.

JCC and SPI Format

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

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 HCLD 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

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 SYSLOG following a PAUSE or STOP command or a PAUSE statement. All tape or disk system I/O files must have been closed. Follcwing the UNBATCH command, the attention routine accepts BATCH or START commands for the affected partition and if a HOLD ccmmand has been issued, device assignments for the affected partition stays in effect. 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.

JCC Format

```
UNBATCH blank
```

UPSI

The Set User Program Switch Indicators statement allows you 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.

JCS Format

```
// UPSI nnnnnnnn
```

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

The VOL command or statement (Volume Information) 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.

JCS Format

```
// VOL SYSxxx,filename
```

SPI Format

```
[//] VOL SYSxxx,filename
```

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 unique filename is identical to the symbolic name of the program DTF that identifies the file.

XTENT

The XTENT command or statement (DASD Extent Information) defines each area, or extent, of a DASD file. One or more XTENT commands or statements must follow each DLAB command or statement.

JCS Format

```
// XTENT type,sequence,lower,upper,
         'serial no.',SYSxxx[B2]
```

SPI Format

```
[//] XTENT type,sequence,lower,upper,
           'serial no.',SYSxxx[,B2]
```

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, 2314, and 2319
0 to 9 for 2321

C_1C_1 = Subcell number.

00 for 2311, 2314, and 2319
00 to 19 for 2321

$C_2C_2C_2$ = cylinder number.

000 to 199 for
2311, 2314, and 2319

or

strip number:

000 to 009 for
2321

H_1 = head block position.

0 for 2311, 2314, and 2319
0 to 4 for 2321

H_2H_2 = head number.

00 to 09 for 2311
00 to 19 for 2321, 2314, and 2319

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, 2314, and 2319
0-9 for 2321

This field is optional. If missing, Job Control assigns $B_2=B_1$.

/* - End-of-Data File

The End of Data File statement must be the last statement of each input data file on SYSRDR and SYSIPT.

JCS Format

```
/* ignored
```

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 End of Job statement must be the last statement of each job.

JCS Format

```
/& ignored [comments]
```

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. If a job updates a system directory, comments included on the /& statement are not printed.

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.

JCS Format

```
* any user comments
```

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.

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 or 2319. (Only 80-character records are acceptable from SYSIPT assigned to a 2314 or 2319.) 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 nc keys, and record lengths are the same as stated. When SYSIPT is assigned tc 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 nct reccgnized.
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, cr if all assigned alternates were currently active as other system files, a message prints on SYSLOG requesting the cperator tc 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, 2314, or 2319 disk storage.

If both SYSRDR and SYSIPT are to be assigned tc 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, 2314, or 2319 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, 2314, or 2319 Job Control performs these functions:

1. Rejects the assignment if it is not permanent.
2. Rejects the assignment if a previous assignment to 2311, 2314, or 2319 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, any existing files are deleted and 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, 2314, or 2319. 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 nc 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 ccntent 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 /& cn 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 NOLCG 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.

ASSIGNING 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, 2314, or 2319 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 15 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 in Example 1.

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. End-of-job indicator. All temporary symbolic unit assignments are reset to the standards established when the system was generated.

9. PAUSE statement that requests operator action.

10. JOB Statement for the series of job steps to be performed in Example 2.

11. OPTION statement specifying that the no-dump option be exercised. The link option must be included to enable a new linkage-edit.

12. 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.

13. EXEC statement for the program to be executed. The data for the execution is followed by the end-of-data-file indicator.

14. PAUSE statement that requests operator action.

15. End-of-job indicator. All temporary symbolic unit assignments are reset to

the standards established when the system was generated.

16. JOB statement for the next job.

```
   1    // JOB          EXAMPLE1

  ┌-    ASSGN         SYSLNK,X'191'
  |     ASSGN         SYS001,X'180'
  2     ASSGN         SYS002,X'181'
  └-    ASSGN         SYS003,X'182'

   3    // OPTION       LINK,DUMP

  ┌-    // EXEC         FORTRAN
  4     (FORTRAN Source Deck)
  └-    /*

  ┌-    // EXEC         ASSEMBLY
  5     (Assembler Source Deck)
  └-    /*

   6    // EXEC         LNKEDT

  ┌-    // EXEC
  7     (Data for User Object Program)
  └-    /*

   8    /&

   9    // PAUSE        SAVE SYS001, MOUNT SCRATCH TAPE

  10    // JOB          EXAMPLE2

  11    // OPTION       NODUMP,LINK

  ┌-       PHASE        PHNAM,ORIGIN
  |        INCLUDE      SORT
  |        INCLUDE      SINE
  |        INCLUDE
  12    (Object Deck to be Included)
  |     /*
  |        ENTRY
  └-    // EXEC         LNKEDT

  ┌-    // EXEC
  13    (Data for User Object Program)
  └-    /*

  14    // PAUSE        SAVE SYS002

  15    /&

  16    // JOB          NEXT
```

Figure 15. 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.

- Sets the LUB table entry for SYSRES to point to the PUB entry of the channel and unit number of the resident drive.

- Places the processing unit in the wait state with all interruptions enabled. When the wait state is entered, the operator decides whether SYSLOG or a card reader will be used to communicate with the system. If SYSLOG, 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.

- 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.

- Determines whether System/360 cr System/370. If System/370 and the RMS feature is included in the supervisor, IPL initializes the RMS. If a supervisor includes RMS and is initialized on System/360, IPL writes a message to the operator that no RMS is supported. In that situation, all machine check and channel check errors cause the system to enter a hard wait state.

- Scans the PUB table to determine if at least one IBM 3211 Printer is attached to the system. If a 3211 PUB entry is found, $$BUFLDR (the 3211 buffer load transient) is fetched to load all 3211 buffers. $$BUFLDR uses the buffer loads cataloged as $$BUCB and $$BFCB. The standard loads supplied with the system are:

  $$BUCB (UCSB). The A11 standard commercial character set, folding and data checks suppressed.

  $$BFCB (FCB). Forms control for a 66 line page (6 lines per inch) with 56 lines available for printing and channel 1 for line 1 and channel 12 for line 56.

To add a device to the PUB table, a control statement, read by the communication device (SYSLOG or SYSRDR), in the following format is required.

Operaticn	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 16.

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 nct 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 (SYSLOG or SYSRDR), 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 (SYSLOG or SYSRDR) 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 following 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 (except SYSRES) for the background, ASSGN statements or commands are entered via the communication device (SYSLOG or SYSRDR). The ASSGN statements or commands are as described in Descriptions and Formats of Commands and Statements.

After IPL has been completed, the SYSRES pack should not be replaced with another. Information about the SYSRES pack is part of the supervisor currently in main storage. If the SYSRES pack must be changed, then IPL must be performed for the new pack.

Card Code	Actual Device	Dev. Type X'nn'	Device Type
2400T9	9-track 2400 Series Magnetic Tape Units	50	Magnetic Tape Units
	9-track 3420 Magnetic Tape Units		
2400T7	7-track 2400 Series Magnetic Tape Units		
	7-track 3420 Magnetic Tape Units		
2495TC	2495 Tape Cartridge Reader	51	Tape Cartridge Reader
1442N1	1442N1 Card Read Punch	30	Card Readers - Punches
	2596 Card Read Punch		
2520B1	2520B1 Card Read Punch	31	
2501	2501 Card Reader	10	Card Readers
2540R	2540 Card Reader	11	
2540P	2540 Card Punch	21	Card Punches
2520B2	2520B2 Card Punch	20	
1442N2	1442N2 Card Punch	22	
2520B3	2520B3 Card Punch	20	
1403	1403 Printer	40	Printers
1403U	1403 Printer with UCS Feature	42	
3211	3211 Printer	43	
1404	1404 Printer	40	
1443	1443 Printer	41	
1445	1445 Printer	41	
1050A	1052, 3210, or 3215 Printer-Keyboard	00	
UNSP	Unsupported Device	FF	Unsupported. No burst mode on multiplexor channel
UNSPB	Unsupported Device	FF	Unsupported with burst mode on multiplexor channel
2311	2311 Disk Storage Drive	60	DASD
2314	2314 Direct Access Storage Facility	62	
	2319 Disk Storage		
2321	2321 Data Cell Drive	61	
1412**	1412 Magnetic Character Reader	75	MICR - Magnetic Ink Character Recognition Devices and Optical Reader/Sorters
1419**	1419 Magnetic Character Reader	72	
	1255 Magnetic Character Reader		
	1259 Magnetic Character Reader		
1419P **	1419 Dual Address Adapter Primary Control Unit	73	
1419S **	1419 Dual Address Adapter Secondary Control Unit	74	
2701 *	2701 Data Adapter Unit	D0	Teleprocessing lines A = SAD0 command when enabling the line B = SAD1 command when enabling the line C = SAD2 command when enabling the line D = SAD3 command when enabling the line
2702 A B C D		D1	
2703	2703 Transmission Control	D2	
2955	2955 Data Adapter Unit	D7	Data link for RETAIN/370
2671	2671 Paper Tape Reader	70	Paper Tape Reader
1285	1285 Optical Reader	76	Optical Readers
1287	1287 Optical Reader	77	
1288	1288 Optical Page Reader		
1017	1017 Paper Tape Reader with 2826 Control Unit Model 1	78	Paper Tape Reader
1018	1018 Paper Tape Punch with 2826 Control Unit Model 1	79	Paper Tape Punch
2260	2260 or 2265 Display Station	C0	Display Station
7770	7770 Audio Response Unit	D3	Audio Response Units
7772	7772 Audio Response Unit	D4	
1017TP	1017 Paper Tape Reader with 2826 Control Unit Model 2	D5	Paper Tape Reader
1018TP	1018 Paper Tape Punch with 2826 Control Unit Model 2	D6	Paper Tape Punch

Note: The codes used in the DVCGEN macros are the same codes used in IPL statements.

* For other teleprocessing devices, see IBM System/360, DOS BTAM and QTAM PLMs, GY30-5001 and GY30-5002.

** This device type code is also used for the 1270/1275 optical reader/sorters.

Figure 16. 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 either the core image library or a private 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 a core image library. When a program has been cataloged in a 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 a core image library by the System Loader.

The linkage editor can run in any partition within a minimum of 10K for execution of the program. If run in the background, the linkage editor can catalog or link phases into the system core image library on SYSRES or into a private core image library (SYSCLB). To catalog or link to the system core image library, the linkage editor must be run in the background, and SYSCLB must not be assigned. To catalog or link to a private core image library, the linkage editor can be run in any partition, and SYSCLB must be uniquely assigned to the partition in which the linkage editor is operating.

Execution of the linkage editor in a batched-job foreground partition requires that the batched-job foreground and private core image library options be specified at system generation time.

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 17.

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 17). 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 anywhere after the TXT cards they modify 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 D.

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.

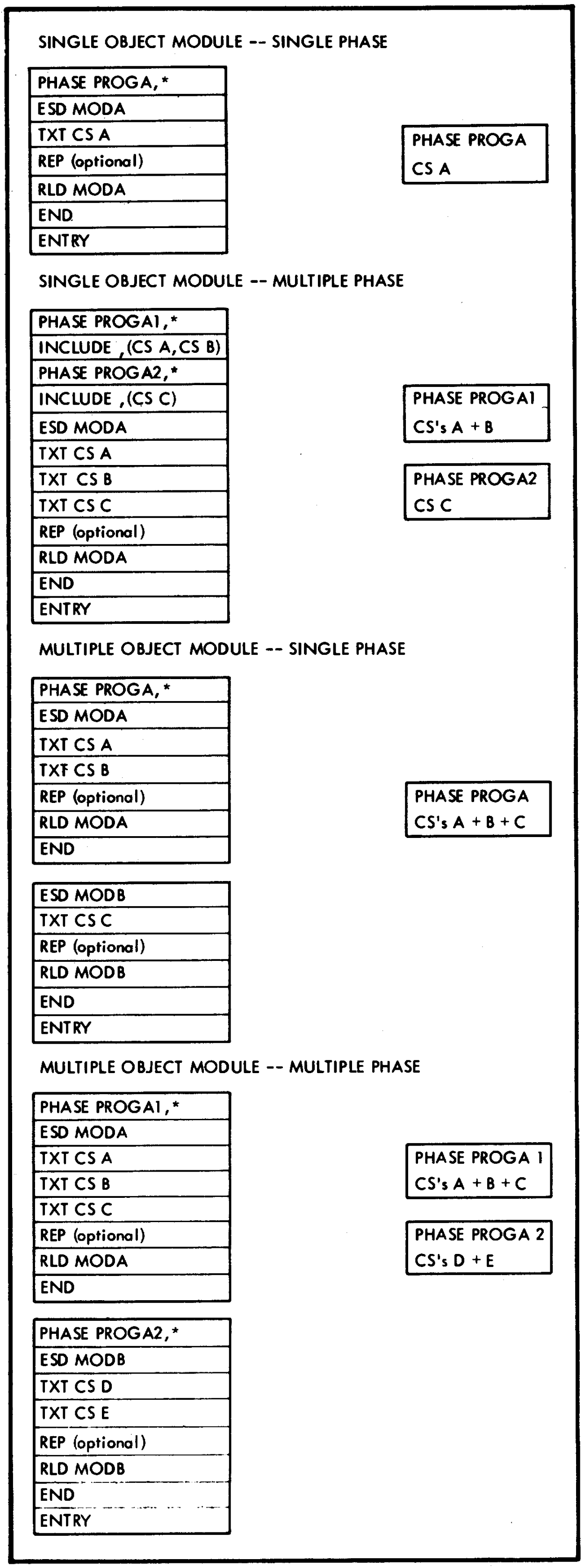


Figure 17. Linkage Editor Input and Output

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.

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. The linkage editor maps these external references as WXTRN, to distinguish them from EXTRN external references.

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.

<u>TXT (Text)</u>: 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.

<u>RLD (Relocation List Dictionary)</u>: 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.

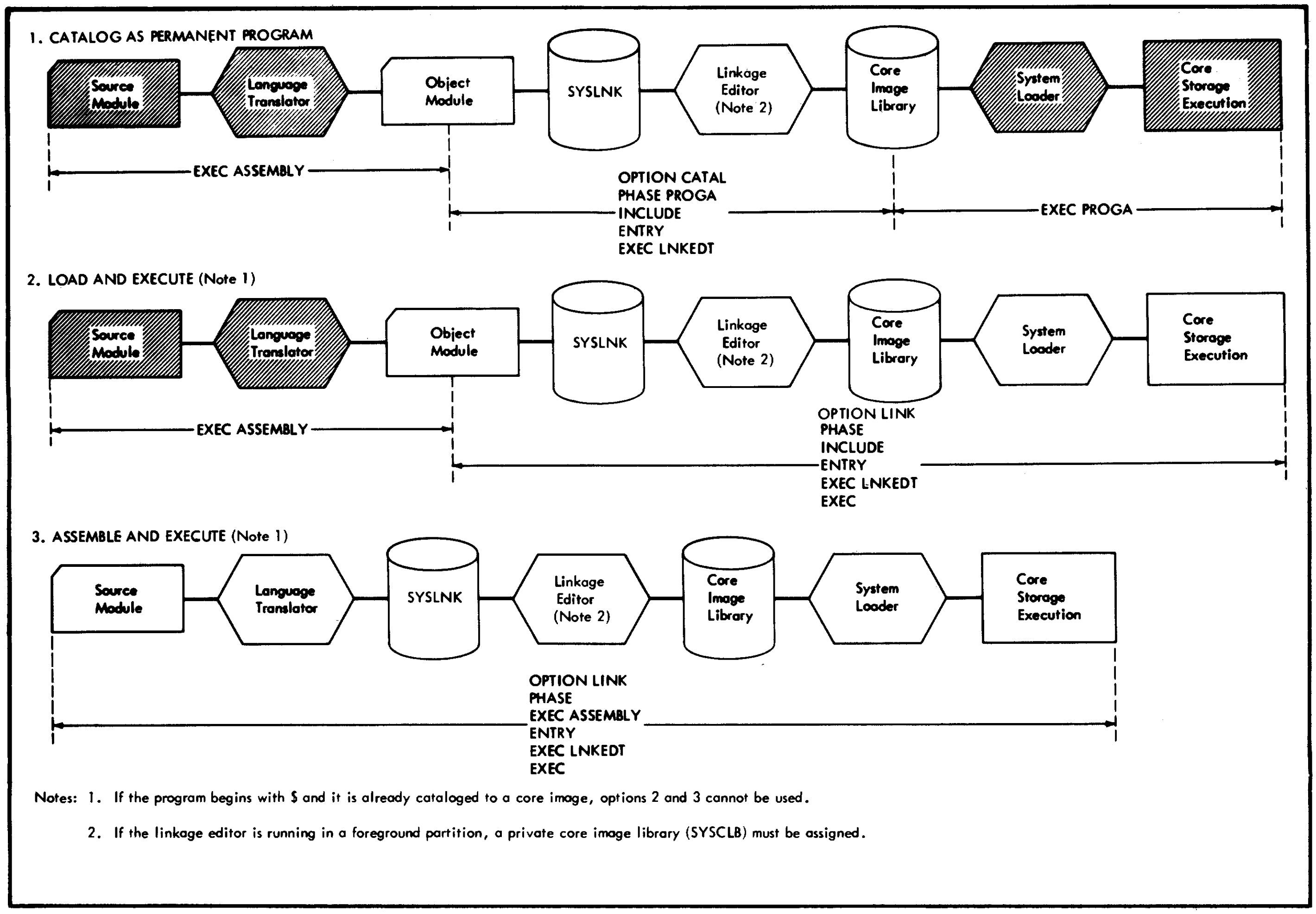
<u>END (End of Module)</u>: The END card indicates end of module to the Linkage Editor. The END card may supply a transfer address. It may also contain the control section length, which was not previously specified in the ESD section definition or private code.

<u>REP (User Replace Card)</u>: 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 may be placed anywhere within the module, as long as it appears after the TXT card it modifies and before the END card. If the REP card does not follow the TXT card it has to modify, the TXT remains unchanged.

<u>Program Phase</u>

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 <u>overlays</u>.

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 <u>control dictionary</u>, 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.



Notes: 1. If the program begins with $ and it is already cataloged to a core image, options 2 and 3 cannot be used.

2. If the linkage editor is running in a foreground partition, a private core image library (SYSCLB) must be assigned.

Figure 18. Three Types of Linkage Editor Operations

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 18).

1. Catalog Programs in Core Image Library. Figure 18 (first diagram) shows the sequence of events when this operation is performed. The Linkage Editor function is performed immediately preceding the operation that catalogs programs into a 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. The input for the LNKEDT function could include modules from a 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 18 (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 a 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 a 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 a core image library is to be executed. Do not use this operation for programs that begin with a $ and are already cataloged in a core image library.

3. Assemble- or Compile-and-Execute. Figure 18 (third diagram) shows the sequence of events when this operation is performed. 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 a core image library. Do not use this operation for programs that begin with a $ and are already cataloged in a core image library.

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 constructs a dummy statement. This allows testing of the program. However, the program with the dummy PHASE statement cannot be cataloged in a 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 is to 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 alsc contain INCLUDE statements.

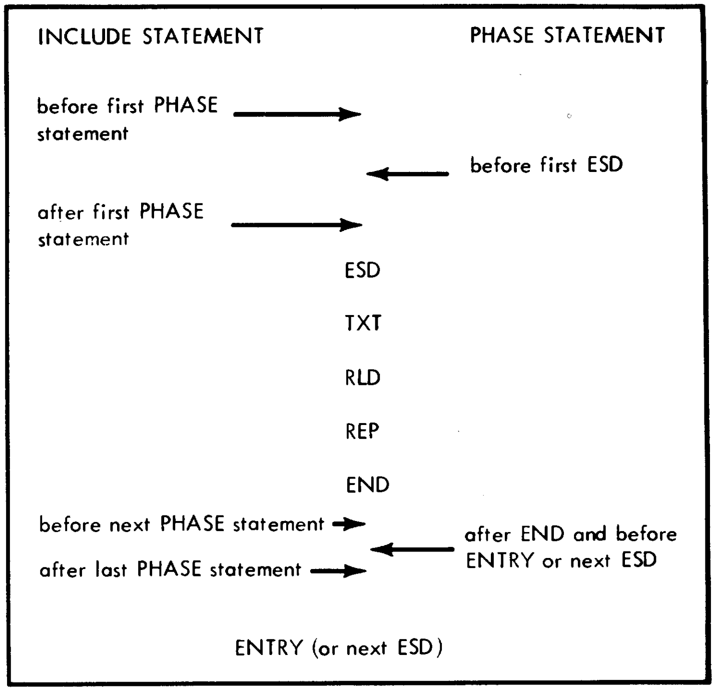
General Control Statement Format

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 19 shows the possible placement of the PHASE and INCLUDE statements.


INCLUDE STATEMENT
PHASE STATEMENT
before first PHASE statement
before first ESD
after first PHASE statement
ESD
TXT
RLD
REP
END
before next PHASE statement
after END and before ENTRY or next ESD
after last PHASE statement
ENTRY (or next ESD)

Note: INCLUDE statements within modules in the relocatable library must precede the ESD statement for the module.

Figure 19. 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 is 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 into a ccre 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 (0-9, A-z, ., #, $, and @) characters are used as the phase name. To reduce the time required to FETCH/LOAD programs, multiphase programs should have phase names of five to eight alphameric characters. The first four characters of each phase within a multiphase batched job program should be the same, but should not be identical to the first four characters of other phases in a core image library. Under this condition, or if a dollar sign ($) 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 COMMCN 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, control section, or 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 that contains the origin name. 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: X'hhhhhh' (one to six hexadecimal digits), dddddddd (one to eight decimal digits), or nK (K=1024).

2. *: Causes 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 in background, * indicates the origin is to be the first doubleword main storage address after the end of the supervisor, the label save area (if any), and the area assigned to the COMMON pool (if any). In foreground, * indicates the origin is to be the first doubleword main storage address after the beginning of the partition, the partition save area, the label save area (if any), and the area assigned to the COMMON pool (if any).

   relocation: Indicates relocation of the phase as described in item 1.

3. S: If S is specified, the origin is determined in the same manner as the first PHASE statement in item 2.

   relocation: Indicates relocation of the phase as described in item 1, although negative relocation is invalid for S-type.

4. ROOT: Tells the Linkage Editor that the phase that follows is a root phase. The root phase is always resident in main storage while the program is being executed. The main-storage address assigned to the root phase is determined in the same manner as the first PHASE statement in item 2. 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 (CSECT) 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 must be: X'hhhhhh' (one to six hexadecimal digits), dddddddd (one to eight decimal digits), or nK (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+address: This format allows the origin point of the program to be set at the start of a foreground partition when link editing in the background and the foreground partition is not allocated. If the foreground partition is allocated, ACTION F1 or F2 has the same effect as F+address.

   It 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: X'hhhhhh (four to six hexadecimal digits), dddddddd (five to eight decimal digits), or nnnnK (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 and system relocatable libraries. AUTOLINK collects each unresolved external reference from the phase. It then searches the private relocatable library (if 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 or 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
```

For background, this causes loading to start at the first doubleword main storage address after the end of the supervisor, the label save area (if any), and the area assigned to the COMMON pool (if any). For foreground, it is the first doubleword after the beginning of the partition, the partition save area, the label save area (if any), and the area assigned to the COMMON pool (if any). 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.

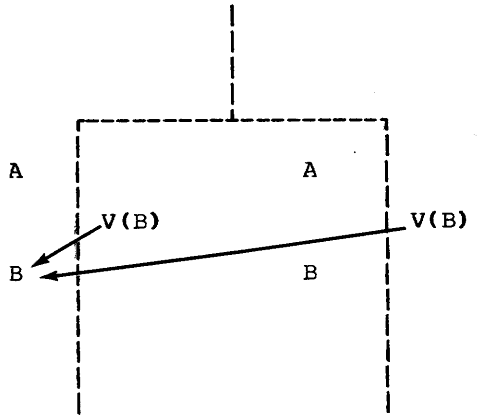
Note: In each of the preceding examples, if the origin address supplied is not on a double-word boundary, the Linkage Editor automatically increments to the next double-word boundary.

The Linkage Editor allows the inclusion of the same control section within each of several phases. If a control section (CSECT) appears in a RCOT phase, it does 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
```

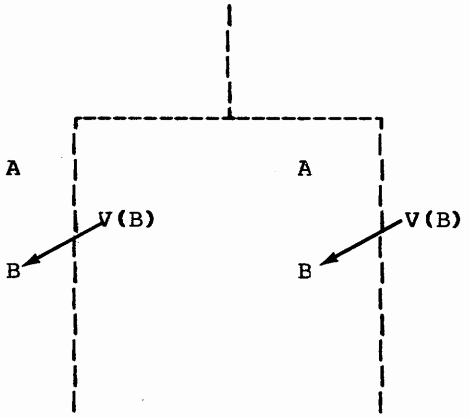
when used as a module in two phases has two different results. When the module is part of the ROOT phase, the external reference [DC V(B)] is resolved with the entry pcint within the ROOT phase (B CSECT). When the module is not part of the ROOT phase, the external reference is resolved with the last previous occurrence of this entry point. Since the reference is given before the C3ECT labeled B is defined in that phase, it is resolved with the CSECT labeled B in the ROOT phase.



Whereas the coding

```
A    START
       .
B    CSECT
       .
A    CSECT
       .
       .
       .
     DC V(B)
       .
       .
       .
B    CSECT
       .
       .
       .
     END
```

when used as a module in two phases has the same result. When the module is part of the ROOT phase, the external reference [DC V(B)] is resolved with the entry point within the ROOT phase (B CSECT). When the module is not part of the ROOT phase, the external reference is resolved with the last previous occurrence of this entry point. Since the CSECT labeled B is defined in this phase before the reference is given, the external reference is resolved in the same phase. No problem arises from defining duplicate CSECTs in the same phase, since these are ignored.



This method of coding redefines the sequence of ESD information to allow valid cross reference by the Linkage Editor. With the AUTOLINK mode, this is also true except for the case of privileged external references (external references whose labels begin with the letters IJ). For privileged external references, if the resolution is not possible within the current phase or ROOT phase, then the AUTOLINK function is performed on this external reference at the end of the phase, and 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 current phase, in the ROOT phase, or in the relocatable library, then 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 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.

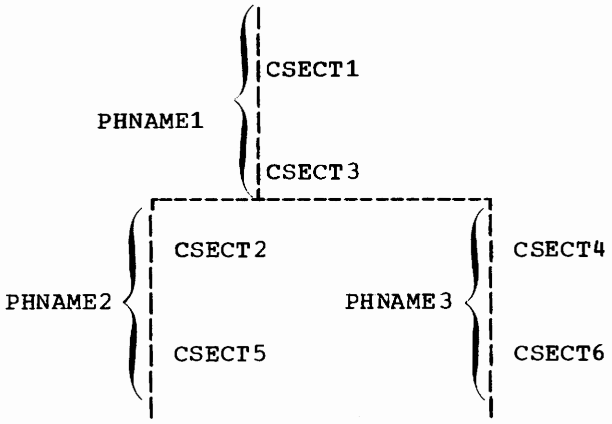
Modules in the relocatable library can be nested by using INCLUDE statements up to a depth of five (level of six). Modules included by INCLUDE statements read from SYSRDR are referred to as being in the first level. Modules included by statements in the first level are at the second level. This is illustrated in Figure 18. 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
 (Object module containing CSECT1 and
 CSECT3)
```

results in the following structure:

```
PHNAME1   CSECT1
          CSECT3
          CSECT4
          CSECT5
```

while the sequence

```
PHASE     PHNAME1,*
INCLUDE   MOD1
INCLUDE   ,(CSECT1,CSECT3)
 (Object module containing CSECT1 and
 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, you 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 uses 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 is the load address of the first phase.

It is necessary to supply the ENTRY statement cnly if a specific entry point is desired. 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. If multiple operands are required, they can be placed in separate ACTION statements or in one ACTION statement separated by commas. Its format is:

Name	Operation	Operand
blank	ACTION	{CLEAR,MAP,NOMAP, NOAUTO,CANCEL, BG,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 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.

BG
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.

The BG, F1 and F2 operands link edit a program to execute in a partition other than that in which the link edit function is taking place. Programs that have a phase origin of S (or * for the first phase of a program) can be origined to the specified partition by use of the operands.

Use of the ACTION BG statement is possible only in a system supporting the batched-job foreground and private core image library options when the linkage editor is executing in a foreground partition.

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. If none of these operands are present, the program is link edited to execute in the partition in which the link edit function is taking place, unless otherwise specified on the PHASE statement.

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.
```

If you are executing the linkage editor in background, 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.

Now, assume the linkage editor is executing in foreground 2. The statement PHASE PHASE1,S causes PHASE1 to be origined at 32K (the beginning of the foreground 2 area) plus the length of the foreground save area.

The sequence

```
ACTION BG
PHASE PHASE1,S
```

causes PHASE1 to be origined at 8K (the end of the supervisor area) plus the length of the background 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 igncred.

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;
- An address constant could not be resolved.

These messages may or may not indicate actual programming errors. If NOMAP is cperational, the warning messages are not printed.

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 20 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 20.

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 20.

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 SYMBCLS). 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.

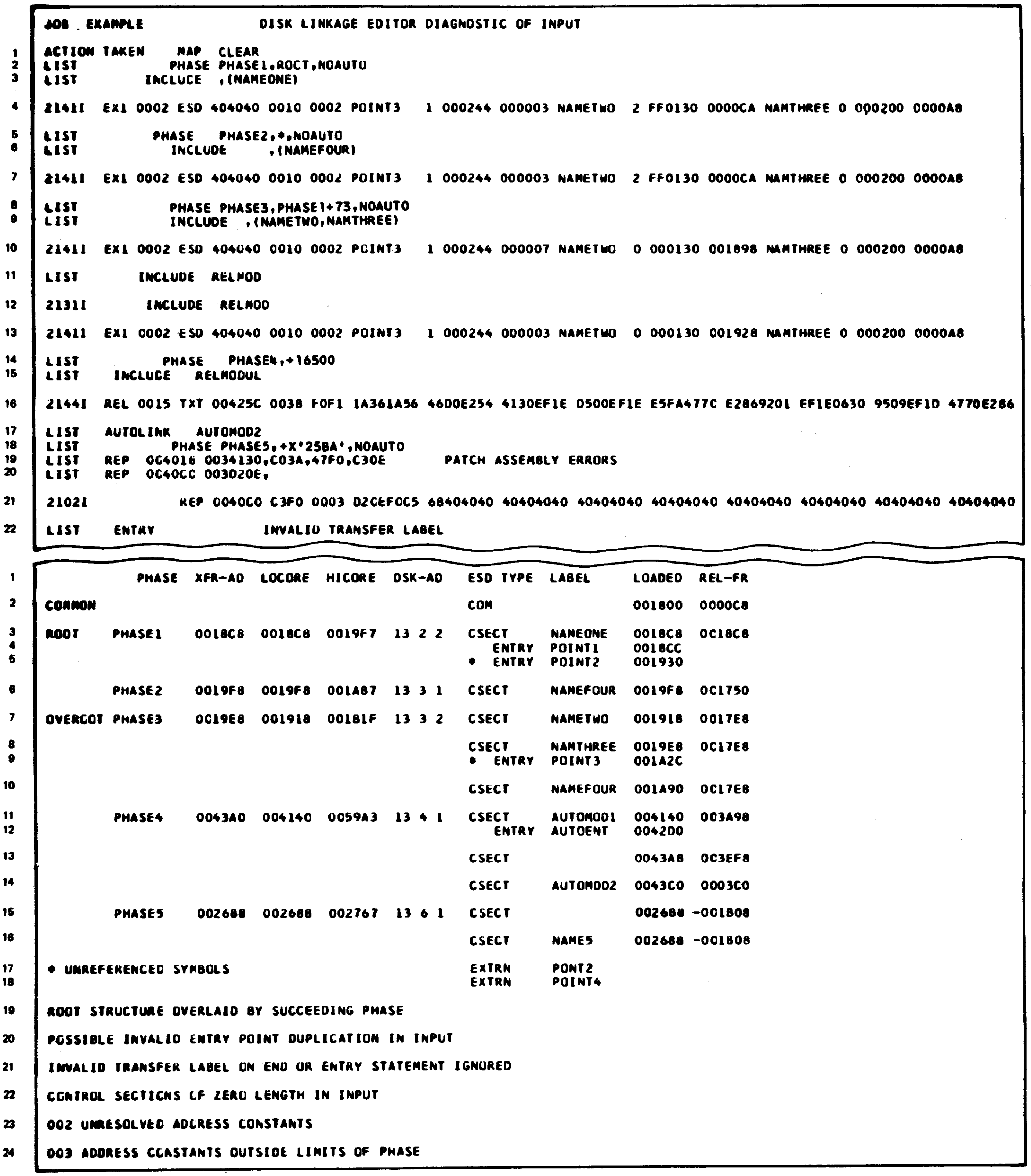
6. Line 20. Warning message. An entry label appeared at least twice in the input stream. At the time of the second (or more) occurrence, it was not possible to validate it as being a true duplication of the previous occurrence. The most common reason for this message is in submodular structure with (source) 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 is to appear in the END record. It is possible 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

 1   ACTION TAKEN     MAP  CLEAR
 2   LIST               PHASE PHASE1,ROCT,NOAUTO
 3   LIST          INCLUCE   ,(NAMEONE)

 4   21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  2 FF0130 0000CA NAMTHREE 0 000200 0000A8

 5   LIST            PHASE   PHASE2,*,NOAUTO
 6   LIST               INCLUDE     ,(NAMEFOUR)

 7   21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  2 FF0130 0000CA NAMTHREE 0 000200 0000A8

 8   LIST               PHASE PHASE3,PHASE1+73,NOAUTO
 9   LIST               INCLUDE  ,(NAMETWO,NAMTHREE)

10   21411  EX1 0002 ESD 404040 0010 0002 PCINT3   1 000244 000007 NAMETWO  0 000130 001898 NAMTHREE 0 000200 0000A8

11   LIST        INCLUDE  RELMOD

12   21311        INCLUDE  RELMOD

13   21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000003 NAMETWO  0 000130 001928 NAMTHREE 0 000200 0000A8

14   LIST              PHASE   PHASE4,+16500
15   LIST    INCLUCE   RELMODUL

16   21441  REL 0015 TXT 00425C 0038 F0F1 1A361A56 4600E254 4130EF1E D500EF1E E5FA477C E2869201 EF1E0630 9509EF1D 4770E286

17   LIST   AUTOLINK   AUTOMOD2
18   LIST               PHASE PHASE5,+X'25BA',NOAUTO
19   LIST   REP  0C4016 0034130,C03A,47F0,C30E       PATCH ASSEMBLY ERRORS
20   LIST   REP  0C40CC 003D20E,

21   21021             REP 0040C0 C3F0 0003 D2CEF0C5 68404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040

22   LIST    ENTRY               INVALID TRANSFER LABEL

 1             PHASE   XFR-AD  LOCORE  HICORE  DSK-AD   ESD TYPE  LABEL      LOADED  REL-FR

 2   COMMON                                             COM                  001800  0000C8

 3   ROOT     PHASE1   0018C8  0018C8  0019F7  13 2 2   CSECT     NAMEONE    0018C8  0C18C8
 4                                                        ENTRY   POINT1     0018CC
 5                                                      *  ENTRY  POINT2     001930

 6            PHASE2   0019F8  0019F8  001A87  13 3 1   CSECT     NAMEFOUR   0019F8  0C1750

 7   OVERCOT  PHASE3   0C19E8  001918  00181F  13 3 2   CSECT     NAMETWO    001918  0017E8

 8                                                      CSECT     NAMTHREE   0019E8  0C17E8
 9                                                      *  ENTRY  POINT3     001A2C

10                                                      CSECT     NAMEFOUR   001A90  0C17E8

11            PHASE4   0043A0  004140  0059A3  13 4 1   CSECT     AUTOMOD1   004140  003A98
12                                                        ENTRY   AUTOENT    0042D0

13                                                      CSECT                0043A8  0C3EF8

14                                                      CSECT     AUTOMOD2   0043C0  0003C0

15            PHASE5   002688  002688  002767  13 6 1   CSECT                002688 -001808

16                                                      CSECT     NAME5      002688 -001808

17   * UNREFERENCED SYMBOLS                             EXTRN     PONT2
18                                                      EXTRN     POINT4

19   ROOT STRUCTURE OVERLAID BY SUCCEEDING PHASE

20   PGSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT

21   INVALID TRANSFER LABEL ON END OR ENTRY STATEMENT IGNORED

22   CGNTROL SECTICNS CF ZERO LENGTH IN INPUT

23   002 UNRESOLVED ADCRESS CONSTANTS

24   003 ADDRESS CCNSTANTS OUTSIDE LIMITS OF PHASE
```

Figure 20. 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:

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 is 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). You should code the 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 non-multiprogramming (MPS=NO) system.

FOREGROUND PROGRAMS (FGP)

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. (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 partition 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 contains 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 partition 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 partition greater than 14K increases linearly to a value of 33,500 at a background area of 40K.

In a partition 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 for a 2311 and 430,440 for a 2314 or 2319.

If a program begins with a $ and it is already cataloged to a core image library, then the load-and-execute and assemble-and-execute operations (see Figure 18) cannot be used.

LINKAGE EDITOR JOB SETUP

The linkage editor can operate in any partition of at least 10K if the system supports the batched-job foreground and private core image library options. Without these options, the linkage editor can only operate in the background. When link editing in the foreground, a private core image library (SYSCLB) must be uniquely assigned to the partition. The program phase (output of the linkage editor) is put in the private core image library. In the background, the linkage editor can put a phase in either a private or the system core image library. If SYSCLB is uniquely assigned in the background, the linkage editor output is put in the private core image library. If SYSCLB is not assigned, the system core image library is used.

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.

If output from the linkage editor is to be placed in a private core image library, the following symbolic unit is also required:

SYSCLB The private core image library

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.
ASSGN SYSCLB	Required if output of linkage editor is to be placed in a private core image library.
// OPTION	OPTION statement must follow the ASSGN statement (if any) for SYSLNK.
ACTION	Optional ACTION statement (with appropriate operands) 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. If a linkage editor job does not end with a /&, the subdirectories are not updated.

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 21 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.

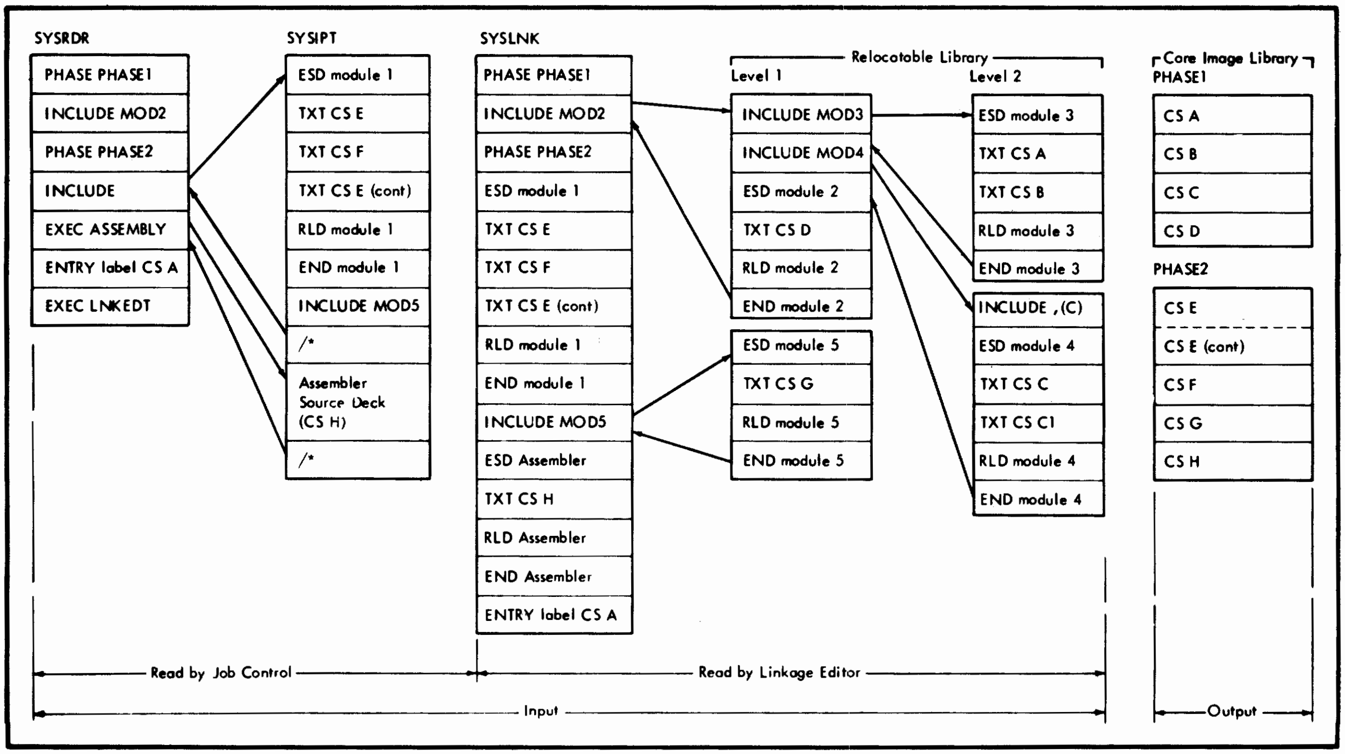


Figure 21. Example of Linkage Editor Input and Output

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 tc 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 system 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 is to 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.

Private core image libraries are discussed in the section Private Libraries.

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 you 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 a 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 are discussed in 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 compiles 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 are discussed in the section Private Libraries.

VERSION AND MODIFICATION LEVEL

Most IBM supplied programs have a 2-byte VM (version and modification level) number. The number may be in decimal or hexadecimal form in a core dump, depending on the input format. It is in decimal form in a DSERV printout of the source statement or relocatable library. For example, version 3 modification level 0 appears as 0300 or F3F0 in a core dump and as 3.0 in a DSERV printout. The VM for modules and macros is contained in their respective directory entry. The VM for phases and transients is contained within the phase or transient. The standard location of the VM for a phase is a displacement of 8 bytes from the load address (the beginning) of the phase and for a transient it is 12 bytes.

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 you. 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 a 2314 or 2319, 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 either the 2314 or 2319 is that on the latter, 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 2314/2319, the library is 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 cn the right applies to either the 2314 or the 2319. If the formula applies equally to the 2311, the 2314, and the 2319 it is given only once.

CORE IMAGE

Core Image Directory Size

Each track allocated to the core image directory can contain entries for 144 phases on the 2311 (or 270 phases cn the 2314/2319) with the exception of the last track, which can contain only 143 (or 269) entries. Thus, the number of tracks (TCD) required for the core image directory is:

$$TCD = \frac{P+1}{144} \qquad TCD = \frac{P+1}{270}$$

where P = total number of phases in the core image library. The value of TCD 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 2314/2319). 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 (Bn) required for a phase:

   $$Bn = \frac{L}{1728} \qquad Bn = \frac{L}{1688}$$

   where L = total number of bytes in the phase. The value Bn is rounded to the next higher integer.

2. Determine the total number of blocks (Bt) required for all phases in the core image library:

   $$Bt = B1 + B2 + B3 + ... + Bn$$

3. Determine the number of tracks (TCL) required to hold all phases in the core image library:

   $$TCL = \frac{Bt}{2} \qquad TCL = \frac{Bt}{4}$$

4. Determine the number of cylinders (CCL) required to hold the core image library and core image directory:

   $$CCL=\frac{TCD+TCL}{10} \qquad CCL=\frac{TCD+TCL}{20}$$

   The value CCL is rounded to the next higher integer if a remainder results.

RELOCATABLE

Relocatable Directory Size

Each track allocated to the relocatable directory can contain entries for 180 modules on the 2311 (or 340 modules on the 2314/2319) 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 (TRD) required for the relocatable directory is:

$$TRD = \frac{m+6}{180} \qquad TRD = \frac{m+6}{340}$$

where m = total number of modules in the relocatable library. The value TRD 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 (cr 16 fixed length blocks for the 2314/2319). 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 apprcximate fairly accurately the library area required for typical programs.

1. Determine the number of blocks (Bc) 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 Bc = total number of cards of the above types.

2. Determine the number of blocks (Be) required for ESD and RLD cards. Assume a separate block for every two ESD or RLD cards.

3. Determine the number of blocks (Bi) 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,

   $$Bi = \frac{\text{total bytes of text in TXT cards}}{200}$$

4. Determine the total number of blocks (Bn) requied for a module in the relocatable library:

   $$Bn = Bc + Bi + Be$$

5. Determine the total number of blocks (Bt) required to hold all of the modules in the library:

   $$Bt = B1 + B2 + B3 + \ldots + Bn$$

6. Determine the number of tracks (TRL) required for the relocatable library:

   $$TRL = \frac{Bt}{9} \qquad TRL = \frac{Bt}{16}$$

   The value TRL is rounded to the next higher integer if a remainder results.

7. Determine the number of cylinders (CRL) required to hold the relocatable library and relocatable directory:

   $$CRL=\frac{TRD+TRL}{10} \qquad CRL=\frac{TRD+TRL}{20}$$

   The value CRL is rounded to the next higher integer if a remainder results.

SOURCE STATEMENT

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 2314/2319) 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 (TSD) required for the source statement directory will be:

$$TSD = \frac{B+6}{160} \qquad TSD = \frac{B+6}{270}$$

where B = total number of books in the source statement library. The value TSD 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 2314/2319). 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 (Ls) in the book. The compressed statement length approximately equals:

$$Ls = (L1 + 1) + (L2 + 1) + \ldots + (Ln + 1) + 3$$

where each Ln = number of bytes word of the source statement.

3. Determine the number of blocks (Bn) needed to hold the book:

   $$Bn = \frac{N(Ls)}{160}$$

   The value Bn is rounded to the next higher integer if a remainder results.

4. Determine the total number of blocks (Bt) required to hold all of the books in the library:

   $$Bt = B1 + B2 + B3 + \ldots + Bn$$

5. Determine the number of tracks (TSL) required to hold all of the books in the source statement library:

   $$TSL = \frac{Bt}{16} \qquad TSL = \frac{Bt}{27}$$

   The value TSL is rounded to the next higher integer if a remainder results.

6. Determine the number of cylinders (CSL) required to hold the source statement library and source statement directory:

   $$CSL = \frac{TSD+TSL}{10} \qquad CSL = \frac{TSD+TSL}{20}$$

   The value CSL 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 and/or to assign a private library to be maintained or serviced.

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,

	MAINT	CORGZ	DSERV	CSERV	RSERV	SSERV
SYSRDR	R	R	R	R	R	R
SYSIPT	R	R	R	R	R	R
SYSLST	O		R	O	O	O
SYSLOG[1]	R	R	R	R	R	R
SYSRES	R	R	R	R	R	R
SYSCLB	O	O	O	O		
SYSRLB	O	O	O		O	
SYSSLB	O	O	O			O
SYS000		O				
SYS001		O				
SYS002		O				
SYS003		O				
SYSIN[2]	O	O	O	O	O	O
SYSPCH	O			O	O	O

R = Required
O = Optional dependent upon function specified

[1] SYSLOG must be assigned to a 1052 or a 3210 or 3215
[2] If SYSRDR and SYSIPT are assigned to the same device, SYSIN may be used.

Figure 22. Logical Units Required and Used by the Librarian Programs

SYSIPT and SYSRDR are usually assigned to the same device, SYSIN. Figure 22 lists the logical units required and used by the librarian programs.

Figure 28 is a table of all maintenance functions. Figure 29 is a table of all service functions. Figure 30 is a table of the copy function. In all these figures, input is shown on SYSIN.

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 module to a relocatable library, or adds a book to a 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 a 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 a Linkage Editor functions that are performed prior to the catalog function for a 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 a core image library and a relocatable library.

The rename function renames an existing phase, module, or book in the appropriate library and directory. When an item is renamed, its version and modification level are not changed.

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 and private directories (if assigned) are displayed on SYSLST.

Note: In order to re-create the transient directory, a /& statement must conclude all maintenance functions affecting the core image library.

The core image library cannot be condensed, nor can any of the three libraries be reallocated, 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.

When running MAINT in a foreground partition, a private core image library (SYSCLB) must be assigned. The only maintenance functions that are performed in F1 or F2 are ones applicable to that private core image library. (No maintenance is done on SYSRES, SYSRLB or SYSSLB when in the foreground.)

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 a core image library, modules in a relocatable library, and books in a source statement library. In addition, all system directories can be displayed with either an alphabetically sorted listing or a listing of the entries as they appear in the directory.

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.

The service function for a core image library and the directory service function can be executed in any partition with a minimum of 10K bytes.

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 phases, modules, and books into them from the system libraries or other libraries of the same type. 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. The copy function executes as a background program only.

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.

Core Image Libraries: Maintenance and Service Programs

This section describes the maintenance and service functions that relate to a core image library. The copy function for a core image library is discussed in the section entitled Copy Functions.

MAINT, CORE IMAGE LIBRARY

To request a maintenance function, other than the catalog function (see Linkage Editor) 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.

The maintenance function for a private core image library is discussed in the Private Libraries section.

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 a 1052 or a 3210 or 3215.

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; the version and modification levels are not changed. 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 a 1052 or a 3210 or 3215.

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 when 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 a 1052 or a 3210 or 3215.

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.

Note: If a private core image library is assigned to either foreground partition, condense is not performed.

CSERV, CORE IMAGE LIBRARY

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 1052 or a 3210 or 3215.

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 1052 or a 3210 or 3215.

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 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 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 a 1052 or a 3210 or 3215

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.

Relocatable Library: Maintenance and Service Programs

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.

MAINT, RELOCATABLE LIBRARY

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 is to 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, following the END statement.

Normally, modules in the relocatable library are output from a language translator. However, you 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 1052 or a 3210 or 3215. 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 allows for assembling or compiling a program and cataloging it to a relocatable library in one continuous run. This is done by inserting a CATALR statement in the job control input stream preceding the phase statement (if present) and the assembler/compiler execute statement. The CATALR statement is written on the SYSPCH file (on tape or DASD) ahead of the assembler/compiler output. Then reassign 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 2314 or 2319 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, and must always be used when deleting DIMODs. 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 a 1052 or a 3210 or 3215.

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; the version and modification levels are not changed. 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 a 1052 or a 3210 or 3215.

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 a 1052 or a 3210 or 3215.

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.

Note: If a private core image library is assigned to either foreground partition, condense is not performed.

RSERV, RELOCATABLE LIBRARY

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 D.

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. SYS-LOG LOG must be assigned to a 1052 or a 3210 or 3215.

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 a 1052 or a 3210 or 3215.

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 a 1052 or a 3210 or 3215.

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.

Source Statement Library: Maintenance and Service Programs

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.

MAINT, SOURCE STATEMENT LIBRARY

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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 (A - Z, #, $, and @).

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 1052 or a 3210 or 3215.

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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 a 1052 or a 3210 or 3215.

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; the version and modification levels are not changed. 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is done on the sublibrary.

oldname and newname represent the old book-name and the new book-name. If newname is 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 tc 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 a 1052 or a 3210 or 3215.

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 cf 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 NO, 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.

> Note: See the Catalog function in the section MAINT, Source Statement Library for additional information about sequence numbering books in a source statement library.

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]
```

This indicates that source statements following the REP statement are to replace existing source 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 is not possible to update

Note: See the Catalog function in the sections MAINT, Source Statement Library for additional information about sequence numbering books in a source statement library.

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 cf 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, bcth 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 is possible. If the resequencing option NO is specified, the updated book is not 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 a 1052 or a 3210 or 3215.

Control statement input for the update function, read from the properly assigned device (usually SYSIN), is:

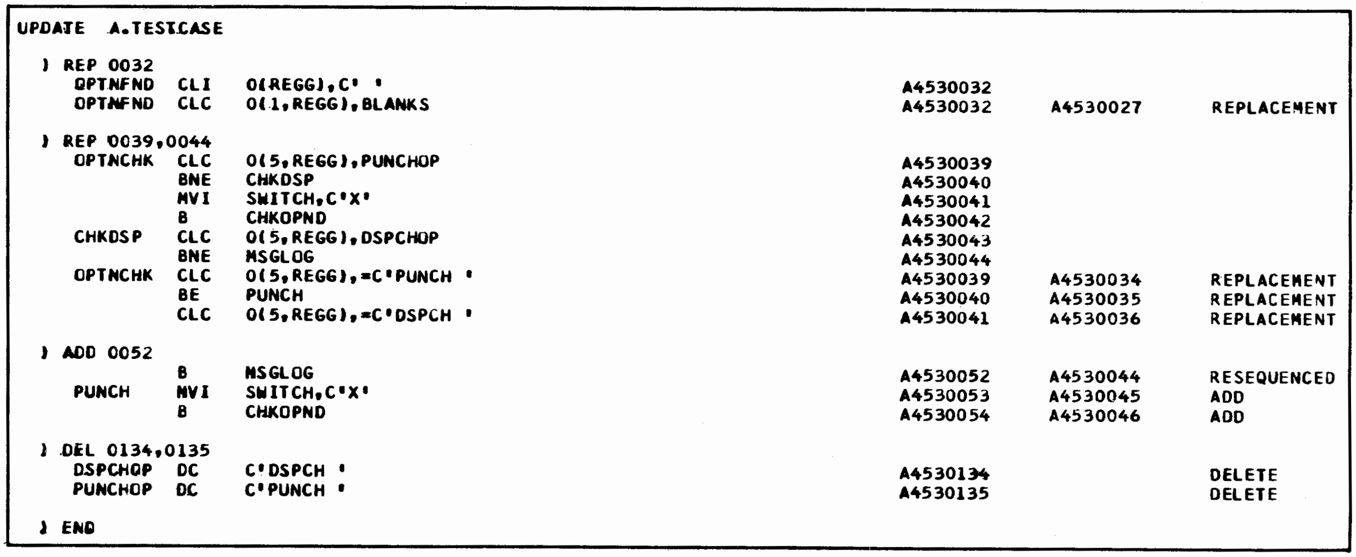
1. The JOB control statement, followed by
2. The ASSGN statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSIN, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The EXEC MAINT control statement, followed by
4. The UPDATE control statement, followed by
5.) ADD,) DEL, or) REP statements with appropriate source statements, followed by
6.) END statement, followed by either another UPDATE (if so, go to item 4), or
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.

<u>UPDATE Function - Activity Log</u>: 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.

If no resequencing has been specified, only the old ID and sequence numbers of the statements involved are indicated.

```
// 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   O(REGG),C' '                                 A4530032
    OPTNFND  CLC   O(1,REGG),BLANKS                             A4530032    A4530027     REPLACEMENT

  ) REP 0039,0044
    OPTNCHK  CLC   O(5,REGG),PUNCHOP                            A4530039
             BNE   CHKDSP                                       A4530040
             MVI   SWITCH,C'X'                                  A4530041
             B     CHKOPND                                      A4530042
    CHKDSP   CLC   O(5,REGG),DSPCHOP                            A4530043
             BNE   MSGLOG                                       A4530044
    OPTNCHK  CLC   O(5,REGG),=C'PUNCH '                         A4530039    A4530034     REPLACEMENT
             BE    PUNCH                                        A4530040    A4530035     REPLACEMENT
             CLC   O(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

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 a 1052 or a 3210 or 3215.

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.

Note: If a private core image library is assigned to either foreground partition, condense is not performed.

SSERV, SOURCE STATEMENT LIBRARY

To request a service function for the source statement library, use the follcwing 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 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 1052 or a 3210 or 3215.

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 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand flagged as invalid and no processing 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 is always 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 1052 or a 3210 or 3215.

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 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 is 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 1052 or a 3210 or 3215.

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.

DSERV, Directory Service

This section describes the display service function that relates to all of the directories. The copy function for these directories are discussed in the section Copy Function.

The DSERV program (directory service) can display the following directories:

- Transient

- Core image library or the directory entry of any phase or group of phases in the core image library with their respective version and modification level, if present.

- Relocatable library

- Source statement library

The system status report (system directory) is unconditionally printed before the printing of the specified directory.

The directories can be displayed in one of two formats.

- An alphamerically sorted listing of the directory entries.

- A listing of the entries in the order they appear in the directory.

The format of the display depends upon the operation specified in the DSERV control statement(s).

Multiple displays of the same directory, either sorted or unsorted, may be obtained in the same job step. To do this, use a separate control statement for each desired display.

OPERATIONAL CHARACTERISTICS

If any private library is assigned, then it is displayed in place of the corresponding system library. To display the system library, the corresponding private library must be unassigned.

The printed output of DSERV contains the system directory followed by the specified directory. Each printed directory is preceded by a header that contains the name of the directory in EBCDIC characters. All fields are headed by the title DEC (decimal) or HEX (hexadecimal). Each page contains up to 96 directory entries by printing 2 columns per page with 48 entries per column. The standard printer form with 56 lines per page is used.

DSERV can be executed in any partition with a minimum of 10,240 bytes of main storage (10K) with the statement:

```
        // EXEC  DSERV
```

However, if a sorted display is specified, then DSERV makes use of any additional main storage beyond the minimum of 10K that is available to the partition for sorting more of the directory in a single pass. The number of entries that are sorted in a single pass depends on the amount of storage available for DSERV and the directory that is specified (Figure 25). If the directory contains more entries than the maximum allowed for a single pass, the display is a group of alphamerical listings. Each group contains a portion of the directory equal to the maximum number of entries allowed for a single pass; except for the last group sorted, which contains the remainder of the directory.

	Maximum Number of Entries for Sorted DSERV	
Directory	for a 10K partition	per additional 2K
Core Image or Transient	384	102
Core Image with Version and Modification Level	240	93
Relocatable or Source Statement	480*	128

* 432 for F1 and F2

Figure 25. Maximum Number of Entries for Sorted DSERV in One Pass

For example, a sorted display of a core image library with 968 entries is printed in 3 groups. The first 2 groups contain 384 entries each and the last contains the remaining 200. The same directory sorted in 12K partition is also printed in 3 groups. However, the first 2 groups contain 480 entries each, and the last group contains only 8 entries. Thus, when calculating the maximum number of entries that can be sorted in a partition larger than 10K, only multiples of 96 are valid.

For DSERV, 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 1052, or a 3210 or 3215.

Control statement input for DSERV 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 statments are followed by

3. The EXEC DSERV control statement, followed by

4. The DSPLY or DSPLYS 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.

DSERV CONTROL STATEMENTS

The control statements for DSERV are:

Name	Operation	Operands
blank	{DSPLY DSPLYS}	directory[,directory] [(phasename[,nn])]

Name

Name field must be blank.

Operation

DSPLY Display the directory entries in the sequence that they appear in the directory.

DSPLYS Display the directory entries sorted alphamerically.

Operands

Directory: Can be one of the following:

TD The transient directory

CD The core image library directory

RD The relocatable library directory

SD The source statement library directory

ALL All the above (TD, CD, RD, and SD) are specified.

Note: If CD (without any other operands) or ALL is specified, the version and modification level of CD entries is not displayed.

Phasename: If CD is specified, then the version and modification level of any single phase or group of phases can be displayed by using phasename. Specify the phasename desired. If it contains less than eight characters, the specification should either be padded out with blanks (COBOL   ) or one blank should be included immediately after the last character of the phasename (COBOL). In the example, COBOL, if many phases contain the first five characters, COBOL, then to display the version and modification level of all of these phases specify (COBOL) without any blanks.

nn: Specify (in decimal) the displacement into the phase where the location of the version and modification level is for one of the following:

- A version and modification level in the nonstandard position.
- The version and modification level of the phase(s) specified is higher than that of the DSERV in use.

If nn is not specified, the program assumes 12 bytes for IBM-supplied transients and 8 bytes for all other entries.

Notes:

1. Only one phase or group of phases can be specified per CD control statement. If more than one is specified, then the last specification is the only one processed.
2. Continuation cards are not valid.
3. A blank operand or no control statement results in a system status report only.

Reallocation Function

The reallocation function redefines the number of tracks and cylinders allotted to a 2311, 2314, 2319 disk resident system. Private libraries cannot be reallocated. If a private library is assigned and any reallocation is attempted, a message is issued and the job is canceled.

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. All three libraries are 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 197. 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 23 cylinders so as not to exceed the limit. Thus, the ALLOC statement could read:

```
ALLOC CL=120(7),RL=30(2),SL=47(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 1052 or a 2310 or 3215.

If message 4n44A comes up due to a change in the label set when reallocating SYSRES, type DELETE on SYSLOG. The job continues 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. 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 RL=0(0). The format of the ALLOC statement is identical to the ALLOC statement described in the section Reallccation Function.

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 and the existing system libraries on the new SYSRES are destroyed.

The following functions are performed automatically by the CORGZ program, except when MERGE or NEWVOL statements are used (see Library Merge Function and Private Libraries).

- All programs essential to a minimum system are copied in a system copy (that is: defining and creating any new SYSRES pack with CORGZ). 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.

COPY, ALL SYSTEM LIBRARIES

The COPY control statement copies the complete system, but can only be used when SYSRES contains all three libraries. The information copied into the new pack includes cylinder 0 (tracks 0 and 1), the three libraries, the label cylinder, and the VTOC cylinder. It is in the following format:

```
COPY ALL
```

The entry in the operation field is COPY. The entry in the operand field is ALL.

COPY, CORE IMAGE LIBRARY

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 seccnd 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.

COPY, RELOCATABLE LIBRARY

The COPYR control statement is used to specify the modules in the relocatable library that are to be copied. It is in cne 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 cperand 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 seccnd 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.

COPY, SOURCE STATEMENT LIBRARY

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 an alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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 any alphameric character (0-9, A-Z, #, $, and @), representing source statement sublibraries.

The sublib qualifier is required. If omitted, the operand is flagged as invalid and no processing is 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.

COPY CONSIDERATIONS

Each library that is to be selectively copied requires a separate group of control statements. 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.

The following functions are performed automatically by the CORGZ program, except when MERGE or NEWVOL statements are used (see Library Merge Function and Private Libraries).

- All programs essential to a minimum system are copied in a system copy (that is: defining and creating any new SYSRES pack with CORGZ). 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 26 shows an example of a valid job setup for the copy function.

```
// 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 26. 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 1052 or a 3210 or 3215.

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.

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 and/or a private core image library on SYS003.

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 and/or a private core image library on SYSCLB.

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.

Notes:

1. If the COPYC ALL statement is used, the Supervisor and transient phases are also transferred to the receiving system resident file or private core image library. The Supervisor and transients previously contained on the receiving SYSRES or SYSCLB 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.

2. The lack of a NEWVOL or MERGE statement indicates the creation of a new system residence file.

3. If COPY ALL or COPYI $$A$IPL2 is specified, the IPL bootstrap phase at cylinder 0, track 1, record 5 of SYSRES is copied.

TRANSFER OF LIBRARY DATA BETWEEN LIBRARIES

Library data can be transferred between libraries as shown in the following 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:

1. The core image library of the system residence file on SYSRES and the core image library of another system residence file on SYS002.

2. A private core image library and the core image library of the system residence file on SYSRES.

3. A private core image library and the core image library of a system residence file on SYS002.

4. Two private core image libraries.

You 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.

6. When merging to a private core image library file, the file name must be IJSYSCL, the logical unit must be SYSCLB, and the file ID must be identical to the ID supplied when the file was created.

7. When merging from a private core image library file, the file name must be IJSYSPC, the logical unit must be SYS003 and the file ID must be identical to the ID supplied when the file was created.

Figure 27 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 28 is an example of a job set up to use the MERGE function.

		1	2	3	4	5	6	7
File Name	IJSYSRS	IJSYSRS	IJSYSRL	IJSYSPR	IJSYSSL	IJSYSPS	IJSYSCL	IJSYSPC
Logical Unit	SYSRES	SYS002	SYSRLB	SYS001	SYSSLB	SYS000	SYSCLB	SYS003
Merge RES to NRS	from	to						
Merge NRS to RES	to	from						
Merge RES to PRV	from		to		to		to	
Merge NRS to PRV		from	to		to		to	
Merge PRV to RES	to			from		from		from
Merge PRV to NRS		to		from		from		from
Merge PRV to PRV			to	from	to	from	to	from

Figure 27. 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 SYS003,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 IJSYSPC,'PRIVATE CIL',99/365
  └--     // EXTENT SYS003,111111,1,0,1700,100

Note 4    // DLBL IJSYSPR,'PRIVATE RL TEST',99/365
  └--     // EXTENT SYS001,111111,1,0,1300,100

Note 5    // DLBL IJSYSPS,'PRIVATE SL TEST',99/365
  └--     // EXTENT SYS000,111111,1,0,1400,100

Note 6    // DLBL IJSYSRS,'SYSTEM RESIDENCE',99/365
  └--     // EXTENT SYS002,111111,1,0,1,179

          // EXEC CORGZ
  ┌--        NEWVOL RL=10(2),SL=10(2),CL=10(15)
Note 7       COPYR ALL
  |          COPYS ALL
  └--        COPYC ALL
  ┌--        MERGE PRV,PRV
Note 8       COPYR ALL
  └--        COPYS ALL
  ┌--        MERGE NRS,PRV
Note 9       COPYR ALL
  └--        COPYS ALL
          /*
          // ASSGN SYS003,X'190'
          // ASSGN SYS002,X'191'

Note 10   // DLBL IJSYSCL,'PRIVATE CIL',99/365
  └--     // EXTENT SYSCLB,111111,1,0,1700,100

Note 11   ASSGN SYSCLB,X'191'

Note 12   // DLBL IJSYSPC,'PRIVATE CL TEST',99/365
  └--     // EXTENT SYS003,111111,1,0,1500,100

Note 13   // DLBL IJSYSRS,'SYSTEM RESIDENCE',99/365
  └--     // EXTENT SYS002,111111,1,0,1,179

          // EXEC CORGZ

Note 14      MERGE PRV,PRV
  └--        COPYC ALL

Note 15      MERGE NRS,PRV
  └--        COPYC ALL

          /&
```

Figure 28. Example of Job Set Up to Use the MERGE Function (Part 1 of 2)

Note 1: File definition statements for a private relccatable library file to be created and updated.

Note 2: File definition statements for a private source statement library file to be created and updated.

Note 3: File definition statements for a private care image library file to be created.

Note 4: File definition statements for a private relocatable library file from which modules to be copied.

Note 5: File definition statements for a private source statement library file from which books are to be copied.

Note 6: File definition statements for a modified, or duplicate system residence file from which modules and books are to be copied.

Note 7: Creates private core image, relocatable, and source statement libraries on SYS003, SYSRLB, and SYSSLB, and copies the core image, relocatable, and source statement libraries from the system residence file on SYSRES into them.

Note 8: 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 9: 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.

Note 10: File definition statements for the private core image library file just created and to be updated.

Note 11: In order to merge to the private core image library just created, assign it to SYSCLB. The from file must be assigned to SYS003 and is a previously created private core image library. See MERGE Considerations for additional information.

Note 12: File definition statements for a private core image library from which phases are to be copied.

Note 13: File definition statements for a modified, or duplicate system residence file from which the phases of the core image library are to be copied.

Note 14: Merges all phases from the private core image library on SYS003 into the newly created private core image library on SYSCLB.

Note 15: Merges all phases from the core image library of a modified, or duplicate system residence file on SYS002 into the newly created private core image library on SYSCLB.

Figure 28. Example of Job Set Up to Use the MERGE Function (Part 2 of 2)

Special Considerations for the Librarian Programs

PUNCH SERVICE FUNCTION

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 2314/2319. (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

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 SYSLOG to inform the operator that the library is to be condensed. If a private core image library is assigned to more than one partition, the condense is not performed. If multiprogramming is in progress, the auto-condense is ignored for any library except a private core image library. Also, if condense (CONDS) is specified when MAINT is executed and a private core image library is assigned to either foreground partition, the condense is not performed.

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 2314/2319), each track of the relocatable library contains 9 blocks on the 2311 (or 16 blocks on the 2314/2319), and each track of the source statement library contains 16 blocks on the 2311 (or 27 blocks on the 2314/2319).

If 0 (zero) is specified for nnnnn, an automatic condense is not 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 are 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 core image, relocatable, and source statement libraries on 2311, 2314, and 2319 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 to a private core image library only if SYSCLB is assigned. If SYSCLB is assigned, the system core image library cannot be changed.

3. Reference is made to a private relocatable library only if SYSRLB is assigned. If SYSRLB is assigned, the system relocatable library cannot be changed.

4. 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.

5. Private libraries cannot be reallocated.

6. 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 core image, one private relocatable, and one private source statement library may be assigned at one time to the same partition.

With multiprogramming, you can assign a different private core image, relocatable, and source statement library to each partition, but each partition can have only one private core image, relocatable, and source statement library assigned to it.

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 reallocate.

The following language components support private source statement libraries:

DOS 14K D Assembler
DOS F Assembler
COBOL-D.
American National Standard COBOL

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.

Note: If a problem program accesses a private relocatable and/or source statement library, and the logical unit name in the DTF is SYSRLB and/or SYSSLB respectively, then the format-1 labels are deleted from the VTOC at CLOSE time. Each time a problem program is to access these files (logical unit name in the DTF is SYSRLB or SYSSLB), it is necessary to provide new label information.

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. SYS003 must be assigned if a private core image library is required.

When creating a private library using CORGZ, a file must be defined for IJSYSPC on SYS003 and/or IJSYSRL on SYSRLB and/or IJSYSSL on SYSSLB. IJSYSPC, IJSYSRL, and IJSYSSL are the respective file names used in the creation of private core image, 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:

- CL for a private core image library and directory
- RL for a private relocatable library and directory
- SL for a private source statement library and directory.

cylin — Indicates the number of cylinders to contain the specified library.

track — Indicates the number of tracks to 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 organization of each private core image library is the same as system residence organization from cylinder 0, head 0 through the end of the system core image library. However, a private core image library may start on any cylinder boundary. In determining the space requirements to be entered on the NEWVOL statement, ten tracks for the various directories and system work areas must be included, in addition to the space requirements for the private core image library directory and private core image library. These ten tracks are included in the specification for the size of the directory as well as in the specification for the total size of the library. For example, on a 2311 system

```
NEWVOL CL=14(20)
```

would create a directory of ten tracks and a library of 12 cylinders, with the first ten tracks of the extent used for the various system directories and work areas.

In this example, if the relative track specification in the EXTENT statement is 00120, the system directories and work areas reside on cylinder 12, the private core image library directory on cylinder 13 and the library on cylinder 14-25.

If you desire to locate a private core image library in the same relative location as the system core image library (that is, system directories and work areas on cylinder 0, and the private core image library directory starting on cylinder 1, track 0) the relative track specification in the EXTENT statement must be 00001 because Job Control does not accept 00000.

The COPY program also provides the ability to copy all or part of the system core image and/or relocatable and/or 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 COPYC and/or 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 COPYC and/or COPYR and/or COPYS statements. Below is an example of the sequence of steps.

```
// EXEC CORGZ
   NEWVOL RL=10(2),SL=10(1),CL=15(11)
   MERGE PRV,PRV
   COPYR ALL
   COPYS ALL
   COPYC ALL
/*
/&
```

The following precautions should be observed.

1. When a NEWVOL statement and a COPYC and/or a COPYR and/or a COPYS statement are both present, the NEWVOL statement must precede the COPYC, 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 (MAINT) may be performed on a private library.

1. Catalog
2. Delete
3. Rename
4. Condense
5. Condense limit
6. Update

All the maintenance and service functions for private libraries are performed in the same manner as with system libraries. SYSCLB, SYSRLB, and SYSSLB must be assigned for the private core image, relocatable, and source statement libraries, respectively.

Private libraries must be unassigned in order to perform maintenance and service on system libraries.

During the condense of a private library, attention interrupts are accepted cn SYSLOG. Thus, condense can be canceled before it is completed. If the condense is canceled, the private core image library must be recreated.

Diagnostics are printed only on SYSLST, with the exception of disaster errors from the condense and allocation functions. Disaster messages also appear on SYSLOG.

Function	Unit	Element	Control Statements

Catalog	Core Image Library	Phase	// OPTION CATAL
			(Linkage Editor control statements and if in card form, the phase to be cataloged)
			/*
			// EXEC LNKEDT

	Relocatable Library	Module	// EXEC MAINT CATALR modulename[,v,m] (module to be cataloged)
	Source Statement Library	Book	// 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.
			// ASSGN SYSPCH,X'cuu'
			CATALR modulename[,v.m]
			PHASE name,origin[,NOAUTO]
			// EXEC COBOL
			(source deck)
			/*
			// MTC WTM,SYSPCH,2
			// MTC REW,SYSPCH
			// 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. The MTC statements through /& follow the /* of the last program. This facility is not available for RPG and PL/I (D) compilations or for IBM 2314 or IBM 2319 applications.
			
			Using a DASD file for SYSPCH/SYSIPT. (Must always be a permanent assignment.)
			// DLBL Balance of information
			// EXTENT required for SYSPCH file
			ASSGN SYSPCH,X'cuu'
			CATALR modulename[,v.m]
			PHASE name,origin[,NOAUTO]
			// EXEC COBOL
			(source deck)
			/*
			CLOSE SYSPCH,X'00D'
			// DLBL Balance of information
			// EXTENT 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. CLOSE through /& follow the /* of the last program.

Figure 29. Maintenance Functions, Example (Part 1 of 3)

Function	Unit	Element	Control Statements
Delete	Core Image Library	Phase	// EXEC MAINT DELETC phase1[,phase2,...]
		Program	// EXEC MAINT DELETC prog1.ALL[,prog2.ALL,...]
	Relocatable Library	Module	// EXEC MAINT DELETR module1[,module2,...]
		Program	// EXEC MAINT DELETR prog1.ALL[,prog2.ALL,...]
		Library	// EXEC MAINT DELETR ALL
	Source Statement Library	Book	// EXEC MAINT DELETS sublib.book1[,sublib.book2,...]
		Sub-library	// EXEC MAINT DELETS sublib.ALL
		Library	// EXEC MAINT DELETS ALL
Rename	Core Image Library	Phase	// EXEC MAINT RENAMC oldname,newname[,oldname,newname,...]
	Relocatable Library	Module	// EXEC MAINT RENAMR oldname,newname[,oldname,newname,...]
	Source Statement Library	Book	// EXEC MAINT RENAMS sublib.oldname,sublib.newname[,sublib. oldname,sublib.newname,...]
Update	Source Statement Library	Book	// EXEC MAINT UPDATE sublib.bookname,[s.book1],[v.m],[nn]) ADD,) DEL, or) REP statements as required with source statements to be added) END [v.m[,C]]

Figure 29. Maintenance Functions, Example (Part 2 of 3)

Function	Unit	Element	Control Statements
Condense	Core Image Library	Library	// EXEC MAINT CONDS CL

	Relocatable Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS RL

	Source Statement Library	Library	// EXEC MAINT CONDS SL
	Libraries	All	// EXEC MAINT CONDS CL,RL,SL
Set Parameter for Automatic Condense	Libraries	Any or All	// EXEC MAINT CONDL lib=nnnnn[,lib=nnnnn[,lib=nnnnn]] Notes: Values to be substitued for lib: CL -- Core image library RL -- Relocatable library SL -- Source statement library Values to be substitued for nnnnn: One to five decimal digits, with a maximum value of 65536.

Reallocation	System	Library	// 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 substitued for cylin and tracks:
			Any integer

Note: // JOB, /*, and /& must be included where needed.

Figure 29. Maintenance Function, Example (Part 3 of 3)

Display Unit	Element	Control Statements
Core Image Library	Phase	// EXEC CSERV DSPLY phase1[,phase2,...]
	Program	// EXEC CSERV DSPLY prog1,ALL[,prog2.ALL,...]
	Library	// EXEC CSERV DSPLY ALL
	Directory	// EXEC DSERV DSPLY CD or DSPLYS CD
	Phases(s) with Version and Modification Level	In the standard position: // EXEC DSERV DSPLY[S] CD(phasename) or CD(phasename)
	Phase(s) with Version and Modification Level	In the nonstandard position or higher than DSERV in use: // EXEC DSERV DSPLY[S] CD(phasename,nn) or CD(phasename ,nn)
Relocatable Library	Module	// EXEC RSERV DSPLY module1[,module2,...]
	Program	// EXEC RSERV DSPLY prog1.ALL[.prog2.ALL,...]
	Library	// EXEC RSERV DSPLY ALL
	Directory	// EXEC DSERV DSPLY RD or DSPLYS RD
Source Statement Library	Book	// EXEC SSERV DSPLY sublib.book1[.sublib.book2,...]
	Sublibrary	// EXEC SSERV DSPLY sublib1.ALL[,sublib2.ALL,...]
	Library	// EXEC SSERV DSPLY ALL
	Directory	// EXEC DSERV DSPLY SD or DSPLYS SD
Transient Directory	Directory	// EXEC DSERV DSPLY TD or DSPLYS TD
System Directory	Directory	// EXEC DSERV
Directories	All	// EXEC DSERV DSPLY ALL or DSPLYS ALL

Figure 30. Service Functions, Example (Part 1 of 3)

Punch Unit	Element	Control Statements
Core Image Library	Phase	// EXEC CSERV PUNCH phase1[,phase2,...]
	Program	// EXEC CSERV PUNCH prog1.ALL[,prog2.ALL,...]
	Library	// EXEC CSERV PUNCH ALL
Relocatable Library	Module	// EXEC RSERV PUNCH module1[,module2,...]
	Program	// EXEC RSERV PUNCH prog1.ALL[.prog2.ALL,...]
	Library	// EXEC RSERV PUNCH ALL
Source Statement Library	Book	// EXEC SSERV PUNCH sublib.book1[,sublib.book2,...][,CMPRSD]
	Sublibrary	// EXEC SSERV PUNCH sublib1.ALL[,sublib2.ALL,...][,CMPRSD]
	Library	// EXEC SSERV PUNCH ALL[,CMPRSD]

Figure 30. Service Functions, Example (Part 2 of 3)

Display and Punch Unit	Element	Control Statements
Core Image Library	Phase	// EXEC CSERV DSPCH phase1[,phase2,...]
	Program	// EXEC CSERV DSPCH prog1.ALL[,prog2.ALL,...]
	Library	// EXEC CSERV DSPCH ALL
Relocatable Library	Module	// EXEC RSERV DSPCH module1[,module2,...]
	Program	// EXEC RSERV DSPCH prog1.ALL[,prog2.ALL,...]
	Library	// EXEC RSERV DSPCH ALL
Source Statement Library	Book	// EXEC SSERV DSPCH sublib.book1[,sublib.book2,...][,CMPRSD]
	Sublibrary	// EXEC SSERV DSPCH sublib1.ALL[,sublib2.ALL,...][,CMPRSD]
	Library	// EXEC SSERV DSPCH ALL[,CMPRSD]

Note: // JOB, /*, and /& must be included where needed.

Figure 30. Service Functions, Example (Part 3 of 3)

Copy Unit	Element	Control Statements

Core Image Library	Phase	// 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	// 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	// 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	// 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	// 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	// 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 31. Copy Function, Example (Part 1 of 5)

Copy Unit	Element	Control Statements

Source Statement Library	Book	// 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,...]

	Sublibrary	// 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	// 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	// 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 Library (Note 2)	Core Image	// ASSGN SYS003,X'cuu'
		// DLBL IJSYSPC,'user identification of
		private library',date,code
		// EXTENT SYS003,balance of extent information
		// EXEC CORGZ
		NEWVOL CL=cylin(tracks)

	Relocatable	// 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)

	Source Statement	// 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)

Figure 31. Copy Function, Example (Part 2 of 5)

Copy Unit	Element	Control Statements

Definition and Creation of a Private Library (Note 2)	Core Image	// ASSGN SYS003,X'cuu'
		// DLBL IJSYSPC,'user identification of private library',date,code
		// EXTENT SYS003,balance of extent information
		// EXEC CORGZ
		NEWVOL CL=cylin(tracks)
		COPYC operands

	Relocatable	// 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

	Source Statement	// 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

Merge System Residence to New System Residence		// 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		// ASSGN (statements as required)
		// DLBL IJSYSRS,'NEW SYSTEM RESIDENCE',date,code
		// EXTENT SYS002,balance of extent information
		// EXEC CORGZ
		MERGE NRS,RES
		COPY statements (COPYI, COPYC, COPYR, COPYS, COPY) as required

Figure 31. Copy Function, Example (Part 3 of 5)

Copy Unit	Element	Control Statement

Merge System Residence to Private Libraries		// ASSGN (statements as required)
		// 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
		// DLBL IJSYSCL,'PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYSCLB,balance of extent information
		ASSGN SYSCLB,X'cuu'
		// EXEC CORGZ
		MERGE RES,PRV
		COPY statements (COPYI, COPYR, COPYS, COPYC) as required

Merge New System Residence to Private Libraries		// 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 information
		// DLBL IJSYSSL,'PRIVATE SOURCE STATEMENT
		LIBRARY',date,code
		// EXTENT SYSSLB,balance of extent information
		// DLBL IJSYSCL,'PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYSCLB,balance of extent information
		ASSGN SYSCLB,X'cuu'
		// EXEC CORGZ
		MERGE NRS,PRV
		COPY statements (COPYR, COPYS, COPYC) as required

Merge Private Libraries to System Residence		// 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
		// DLBL IJSVSPC,'PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYS003,balance of extent information
		// EXEC CORGZ
		MERGE PRV,RES
		COPY statements (COPYR, COPYS, COPYC) as required

Figure 31. Copy Function, Example (Part 4 of 5)

Copy Unit	Element	Control Statement

Merge Private Libraries to New System Residence		// 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
		// DLBL IJSVSPC,'PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYS003,balance of extent information
		// EXEC CORGZ
		MERGE PRV,NRS
		COPY statements (COPYR, COPYS, COPYC) as required

Merge Private Libraries to Private Libraries		// 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
		// DLBL IJSYSCL,'PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYSCLB,balance of extent information
		ASSGN SYSCLB,X'cuu'
		// DLBL IJSYSPC,'SECOND PRIVATE CORE IMAGE
		LIBRARY',date,code
		// EXTENT SYS003,balance of extent information
		// EXEC CORGZ
		MERGE PRV,PRV
		COPY statements (COPYR, COPYS, COPYC) as required
		
		To define the private library in the same job step, precede MERGE with NEWVOL statement.

Notes:

1. // JOB, /*, and /& must be included where needed.

2. The private library can be updated with either a MAINT or a copy MERGE function.

Figure 31. Copy Function, Example (Part 5 of 5)

System Buffer Load (SYSBUFLD)

SYSBUFLD is a special service program that loads the Universal Character Set Buffer (UCSB) and the Forms Control Buffer (FCB) for the 3211 printer with buffer load programs. For the UCSB , these programs must reside in the core image library with valid phase names. SYSBUFLD is self-relocating and requires 2K of main storage for its operation.

SYSBUFLD is executed in your job stream when it is necessary to change the contents of either the UCSB or the FCB. This program is distinct from Job Control and is initiated by the command:

```
// EXEC SYSBUFLD
```

SYSBUFLD then reads a control card from SYSIPT, which identifies the unit and the type of buffer load to be performed. The format of this card is:

Name	Operation	Operands
Not used: Must not be present	FCB UCB	SYSxxx[,phasename] [,FOLD][,NOCHK] [,NULMSG]

Operation

The operation defines the type of buffer load to be performed.

FCB operation causes either a phase in a core image library or an FCB card load (on SYSIPT following the SYSBUFLD control card) to be loaded into the FCB in the IBM 3811 Control Unit. The EBCDIC characters of the FCB load correspond to the lines of printing for any single form (the maximum is 180 lines per form). When a form-skip command is issued to the printer, forms movement is initiated. A character of the form-skip command is compared to the character in the FCB. When a match occurs, forms movement is terminated. Thus, the FCB can adapt the printer to many variable forms.

UCB operation causes the phase in a core image library to be loaded into the UCSB in the IBM 3811 Control Unit. The UCSB load corresponds to the print positions on 3211 printer trains. A character sent to the printer for printing is matched against the character in the UCSB. When a match occurs, the corresponding train character is printed in the printline position that the output character occupied. Thus, with the use of SYSBUFLD and the trains available, the 3211 printer can be adapted to many printing applications.

Phase (transient) $$BUCB supplied in the system core image library is for an A11 train. Additional UCSB loads that are supplied in the Relocatable library are:

Train type	Module name
A11	IJBTRA11
G11	IJBTRG11
H11	IJBTRH11
P11	IJBTRP11
T11	IJBTRT11

The additional UCSB loads must be cataloged to a core image library before their execution. They can be assigned any valid phasename.

The logical unit must be assigned to an IBM 3211 Printer. It is your responsibility to:

- Assemble, linkage edit, and catalog any FCB load phases into the core image library and
- Linkage edit and catalog any IBM-supplied UCSB load phases or assemble, linkage edit, and catalog any user-written UCSB load phases into the core image library, and
- Mount the new train before the UCSB is loaded.

SYSxxx

The name of the logical unit assigned to a 3211 printer to be loaded is SYSxxx. It must be SYSLST, SYSLOG, or a programmer logical unit.

Note: If SYSLOG is specified in the SYSBUFLD control statement, it must be assigned to a 3211.

phasename

The core image name of the phase containing the applicable buffer load. If FCB is specified and phasename is omitted, a FCB card load from SYSIPT is assumed.

FOLD

FOLD signifies that the UCSB buffer is to be loaded with the folding operation code to permit printing of uppercase for lowercase bit configurations. FOLD is optional and only valid for UCB.

NOCHK

NOCHK prevents the data checks that are generated by the 3211 printer because of printline character mismatches with the UCSB. NOCHK is optional and only valid for UCB.

NULMSG

NULMSG signifies that the 80-character verification message is not to be printed on SYSxxx after the buffer is loaded. If NULMSG is not specified after the FCB or UCB has been loaded, the program skips to channel 1, issues a print of the last 80 characters in the phase, and again skips to channel 1. This is repeated for each message.

This message could identify the phase that was loaded. During a UCSB load, the train of the 3211 could be identified by printing a unique character of the train in the message. This would ensure that the mounted train of the 3211 is compatible with the contents of the UCSB.

The UCSB phase format is:

432-character UCSB load	80-character field (see User Written UCSB Load Phase)	80-character verification message

The FCB phase format is:

180-character FCB load	80-character verification message

Note: Other than $$BFCB (loaded by IPL), no additional FCB loads are supplied by IBM.

The FCB card format is:

FCB load
(maximum of 180 bytes)

The verification message is not allowed. Figure 32 contains examples of FCB card loads.

```
       Cols 1 8  11       20       29    35        45                  65             80
       |      |  |        |        |     |         |                   |             |
       |      |  |        |        |     |         |                   |             |

Card 1           1                 2               3                   4
Card 2                                   5
Card 3        C           X
       * EXAMPLE OF FCB CARD LOAD 6 LPI, 180 LINES PER PAGE

Card 1 *         1                 2               3                   4
Card 2                                   5
Card 3        C           X
       * EXAMPLE OF FCB CARD LOAD 8 LPI, 180 LINES PER PAGE. (NOTE: * IN
       * COLUMN 1 OF CARD 1.)

Card 1 *C     1           X
       * EXAMPLE OF FCB CARD LOAD FOR PRINTING 8 LPI. CHANNEL 1 IS THE FIRST
       * LINE AFTER THE PAGE PERFERATION. IT IS NECESSARY TO KEEP THE FCB
       * SYNCHRONIZED WITH THE FORM; HOWEVER, IT IS NOT NECCESSARY TO
       * INITIALLY ALIGN THE FIRST LINE OF THE PAGE WITH THE FIRST POSITION
       * OF THE FCB. THIS EXAMPLE IS FOR A FORM WITH 20 LINES PER PAGE WITH
       * PRINTING ON 14 LINES.
```

Figure 32. FCB Card Load Examples for 3211 Printer

The FCB characters are:

Hex	Channel Indication	SYSIPT Card Code
00	Null (Space)	0
01	Channel 1	1
02	Channel 2	2
03	Channel 3	3
04	Channel 4	4
05	Channel 5	5
06	Channel 6	6
07	Channel 7	7
08	Channel 8	8
09	Channel 9	9
0A	Channel 10	A
0B	Channel 11	B
0C	Channel 12	C
1n	End of FCB	X
1n	8 Lines per inch	*

Bit-position 3 (the flag-bit) of any channel indication specifies the end of the buffer (for example, X'1C') and, if used in the first buffer position, printing is forced to 8 lines per inch (for example, X'11'). When the FCB is being loaded from SYSIPT, an X must be used to indicate the end of the buffer.

Figure 33 is an example of how to use SYSBUFLD. In the example, the first job step of each job loads the FCB and the UCB.

```
// JOB ONE
// EXEC SYSBUFLD    (load the FCB and
 FCB SYSLIST,PHASE1 UCSB for PROG001)
 UCB SYSLST,UCBPH001,FOLD,NOCHK,NULMSG
/*
// EXEC PROG001
/*
/&
// JOB TWO
// EXEC SYSBUFLD      (load the FCB and
 UCB SYSLST,UCBPH002  UCSB for PROG002)
 FCB SYSLST,PHASE2
/*
// EXEC PROG002
/*
/&
```

Figure 33. Example of System Buffer Load for the IBM 3211 Printer

CORE-IMAGE-LIBRARY PHASES FOR FCB

Figure 34 shows how PHASE001 might be defined, assembled, and cataloged to the core-image library for an FCB load.

```
// JOB FCBXMPLE
// OPTION CATAL,DECK
// EXEC ASSEMBLY
 PUNCH ' PHASE   PHASE001,+0'
 DC X'01' CHANNEL 1 & 1ST LINE OF PAGE
 DC XL64'00' 64 LINES OF PAGE
 DC X'1C' CHANNEL 12 & LAST LINE PAGE
 DC XL114'00' UNUSED BUFFER POSITIONS
 DC CL80'FCB PHASE001 LOADED' MESSAGE
 END
/*
// EXEC LNKEDT
/&
```

Figure 34. FCB Load Program

USER WRITTEN UCSB LOAD PHASE

If your 3211 uses a special train configuration, then you must assemble, linkage edit, and catalog the UCSB load phase for the special train into the core image library. The UCSB load phase format is:

Position	Comment
1-432	Train image. The hexadecimal equivalent of all characters on the train.
433-447	Zeros.
448-511	Associative field. This is used by the 3811 Control Unit to check for invalid hexadecimal characters sent to the 3211. For a complete description of this field and how to prepare it, see IBM 3211/3216/3811 Component Description and Operating Procedures, GA24-3543. Note: The -0 edition does not contain this information.
512	Zero.
513-592	Verification message (optional).

SYSBUFLD Messages

Appropriate messages are issued when an invalid parameter specification is made or when a required parameter is omitted. With the exception of an invalid phasename, all errors on the control card may be corrected through SYSLOG. An invalid phasename causes job cancelation.

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 DOS 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).

A facility for problem determination is the DOS Stand-Alone Dump Generation (DUMPGEN) program. This program allows you to generate a stand-alone program, tailored to your requirements, for displaying the contents of main storage when processing under DOS cannot continue.

The label cylinder display program (LSERV) can be used for problem determination. LSERV displays the TLBL and the DLBL and EXTENT information contained on the SYSRES label cylinder. Information about secured data files is not displayed.

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).
- QTAM input/output activity (QTAM 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.

PDAIDs also provide the possibility to:

- Dump transient areas (Transient dump).

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. See Figure 42 for a list of the DOS SVCs. 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 a SIO instruction is issued by the DOS supervisor, the data recorded is:

- Condition Code.
- Device address.
- CCB address.
- CSW.

QTAM Trace

The QTAM trace function records the input/output activity of QTAM. The QTAM events recorded are:

- SVCs 0 and 31
- Supervisor issued SIO
- I/O interrupt

The I/O old PSW and the CSW are recorded when an I/O interrupt occurs. The condition code, device, CCB address, and CSW are recorded when a supervisor issues a SIO. The SVC old PSW and the contents of registers 0 and 1 are recorded when a SVC 0 and 31 is issued.

Transient Dump

The transient dump program displays the 16 general registers, the first 144 bytes of main storage, the physical transient area, and the logical transient area.

Once this routine has been initiated, it is executed each time a program interrupt occurs. Once it is initiated and in control, the system only accepts external interrupts. All system processing is suspended, and the operator should ready the transient dump output device.

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

```
       4C10D 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.)

QT Specifies a QTAM trace is to be performed. (See Note.)

TD Specifies that transient dump is to be initiated. (See Note.)

XX Terminates the PDAID presently running.

EOB or END
Indicates PDAID control statements are to be entered through SYSIPT.

> Note: When FT, GT, IT, QT, or TD are specified, additional PDAID control statements must be given by the operator through SYSLOG. Figure 35 shows the PDAID control statements in the sequence they must be used.

- The EOB or END 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 an EOB or END response is given.
- GO terminates the PDAID control input, and the default is taken for any PDAID options that are not specified. With SYSLOG input, GO is a valid response (Figure 35). When used with SYSIPT input, it should be the last option and must have no operand associated with it.

SYSLOG Message / SYSIPT Parameter	SYSLOG Response / SYSIPT Operand	Meaning	Default
PDAID=	FT GT IT QT TD XX EOB/END	FT Fetch/Load Trace GT GSVC Trace IT I/O Trace QT QTAM Trace TD Transient dump XX Terminate present PDAID function. EOB/END Additional PDAID control input through SYSIPT.	None, the function continues.
OUTPUT DEVICE= (Note 1)	cuu X'cuu' EOB/END GO	Specify the hexadecimal channel and unit number of either a magnetic tape unit or a printer for the output device of the PDAID. A printer is invalid for QTAM Trace. A device must be specified for transient dump.	Core - Wrap mode; none for transient dump
AAA= (Note 3)	X'ℓℓℓℓℓℓ',X'hhhhhh' EOB/END GO	Specify the beginning and ending addresses of an alternate area for Core-Wrap Mode. If it is desired, a minimum of 512 bytes must be specified.	Core-Wrap Mode using CE Save Area.
TRACE PARTITION= (Valid only for Fetch/Load, GSVC and QTAM Trace)	SP BG F2 F1 EOB/END GO	SP Supervisor BG Background F2 Foreground 2 F1 Foreground 1 F1 or F2 is valid for a MPS supervisor only.	Trace all partitions and the supervisor.
IGNORE DEVICE= (Notes 2 and 3)	cuu X'cuu' EOB/END GO	Specify the hexadecimal channel and unit number of the device to be ignored by the I/O or QTAM trace. A maximum of 3 may be specified.	Trace all devices.
TRACE DEVICE= (Notes 2 and 3)	cuu X'cuu' EOB/END GO	Specify the hexadecimal channel and unit number of the device to be traced by the I/O or QTAM trace. A maximum of 3 may be specified.	Trace all devices.
IGNORE SVC= (Notes 2 and 3)	nn EOB/END GO	Specify the hexadecimal SVC number to be ignored by the GSVC trace. A maximum of 6 may be specified.	Trace all SVC's.
TRACE SVC= (Notes 2 and 3)	nn EOB/END GO	Specify the hexadecimal SVC number to be traced by the GSVC trace. A maximum of 6 may be specified.	Trace all SVC's.
GO (Valid SYSIPT parameter)	GO (Valid SYSLOG response)	GO terminates the PDAID control input and uses the default for those options that are not specified.	None.

Note 1. OUTPUT DEVICE and AAA are mutually exclusive. Neither can be shared with any other program.
Note 2. TRACE and IGNORE are mutually exclusive.
Note 3. Not valid for transient dump.

Figure 35. 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.

If an output device or AAA is specified, it cannot be shared with another program.

CATALOGING PDAID AND PDLIST: All the PDAID programs are self-relocating for initialization in any partition (6K or greater) of a multiprogramming system. The main phases, 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:

```
// JOB
// OPTION CATAL
   PHASE PDAID,S or +0 (Note)
   INCLUDE IJBPDAID
// EXEC LNKEDT
/&
```

The following job stream catalogs PDLIST to the core image library:

```
// JOB
// OPTION CATAL
   PHASE PDLIST,S or +0 (Note)
   INCLUDE IJBPDLST
// EXEC LNKEDT
/&
```

Note:

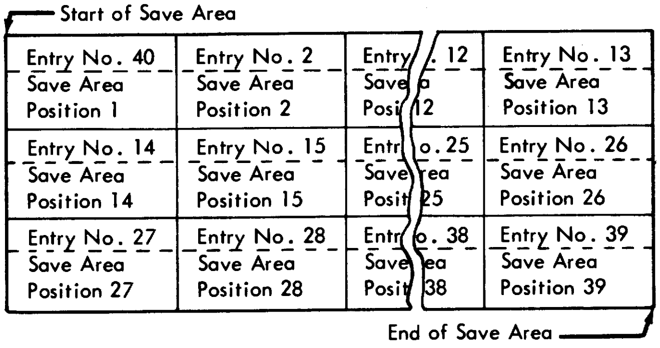
S is required for a batched-job system and

+0 is required for a multiprogramming system.

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, you must display the contents of main storage by either using a stand-alone dump program (such as that generated by DUMPGEN), or manually from the operating panel of the CPU.

When the core wrap mode is selected and CE=800, a maximum of 38 entries can be recorded for the I/O trace, 36 entries for the Fetch/Load trace, and 26 entries for the QTAM trace.

Figure 36 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.



Start of Save Area

Entry No. 40 Save Area Position 1	Entry No. 2 Save Area Position 2	Entry / . 12 Save a Posi 12	Entry No. 13 Save Area Position 13
Entry No. 14 Save Area Position 14	Entry No. 15 Save Area Position 15	Entr o. 25 Save rea Posit 25	Entry No. 26 Save Area Position 26
Entry No. 27 Save Area Position 27	Entry No. 28 Save Area Position 28	Entr o. 38 Save ea Posit 38	Entry No. 39 Save Area Position 39

End of Save Area

Figure 36. Core Wrap Method for Fetch/Load, I/O, GSVC, and QTAM Traces

If the core-wrap mode is selected, then an alternate area can be used for saving the trace data. The alternate area is assigned to the trace through the AAA= message/parameter. The alternate area must be a minimum of 512 consecutive bytes and may be assigned to any area of main storage. However, it must be free for exclusive use by the trace. Program checks and/or unpredictable program operation can result if the area is used by another program. When the alternate area is assigned in a multiprogramming system it may not be displayed in a system dump if it is in a partition other than the one being dumped. If the latter is the case, a stand-alone dump must be used to display the trace data.

Note: If the alternate area is assigned to an active partition, trace data is cleared by Job Control between job steps.

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 the four-byte address at the beginning of the communications region extension (BGXTNSN). The extension is 136 (X'88') bytes from the beginning of the communications region.

With the F/L, I/O, GSVC and QTAM 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 + 212 (X'D4'), the location of the GSVC trace pointers is CEAREA + 228 (X'E4'), and the location of the QTAM trace pointers is CEAREA+338(X'180'). Figure 37 shows the formats for the entries of the save area.

F/L Entry

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	xxxxxx

If a phase-not-found condition is detected, the load and entry addresses are set to X'FFFFFF'.

Figure 37. Trace Entries for Core Wrap Mode (Part 1 of 4)

I/O Entries

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 condition code, the device address, the CCB address, and the CSW are stored, as follows:

C C	Channel Unit Address	CCB Address	CSW
x	0000xxx	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 37. Trace Entries for Core Wrap Mode (Part 2 of 4)

GSVC Entries

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 37. Trace Entries for Core Wrap Mode (Part 3 of 4)

QTAM Entries

When an SVC interrupt occurs, the event type, SVC old PSW, register 0 and register 1 are saved as follows:

Type	SVC old PSW	Register 0	Register 1
V	XXXXXXXX	XXXX	XXXX

When a SIO interrupt occurs, the event type, condition code, CSW status, CAW address, CCW first executed are saved as follows:

Type	CC	Device	CCB	CSW	Not Used
S	X	XX	XXXX	XXXXXXXX	X

When an I/O interrupt occurs, the event type, I/O old PSW, and CSW are saved as follows:

Type	I/O old PSW	CSW
I	XXXXXXXX	XXXXXXXX

Figure 37. Trace Entries for Core Wrap Mode (Part 4 of 4)

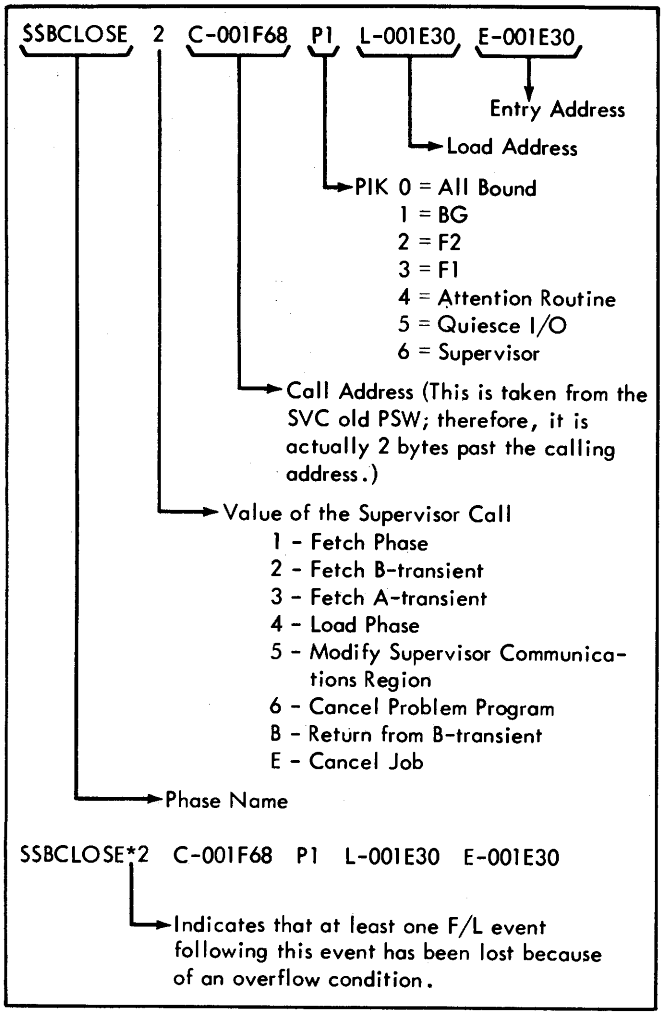
Printing PDAID Data

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.

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 X OR SYSLST. See Figures 38 through 41 for samples of the printed record for each PDAID.



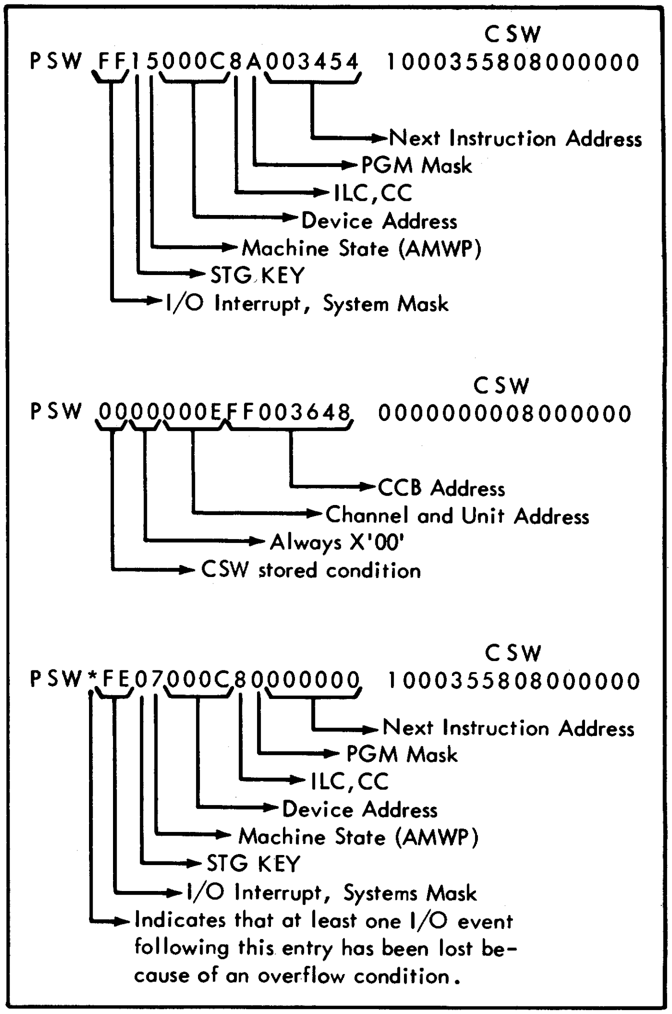
```
SSBCLOSE  2  C-001F68  P1  L-001E30  E-001E30
```

Entry Address
Load Address
PIK 0 = All Bound
1 = BG
2 = F2
3 = F1
4 = Attention Routine
5 = Quiesce I/O
6 = Supervisor
Call Address (This is taken from the SVC old PSW; therefore, it is actually 2 bytes past the calling address.)
Value of the Supervisor Call
1 - Fetch Phase
2 - Fetch B-transient
3 - Fetch A-transient
4 - Load Phase
5 - Modify Supervisor Communications Region
6 - Cancel Problem Program
B - Return from B-transient
E - Cancel Job
Phase Name

```
SSBCLOSE*2  C-001F68  P1  L-001E30  E-001E30
```

Indicates that at least one F/L event following this event has been lost because of an overflow condition.

Figure 38. Sample Output for F/L Trace



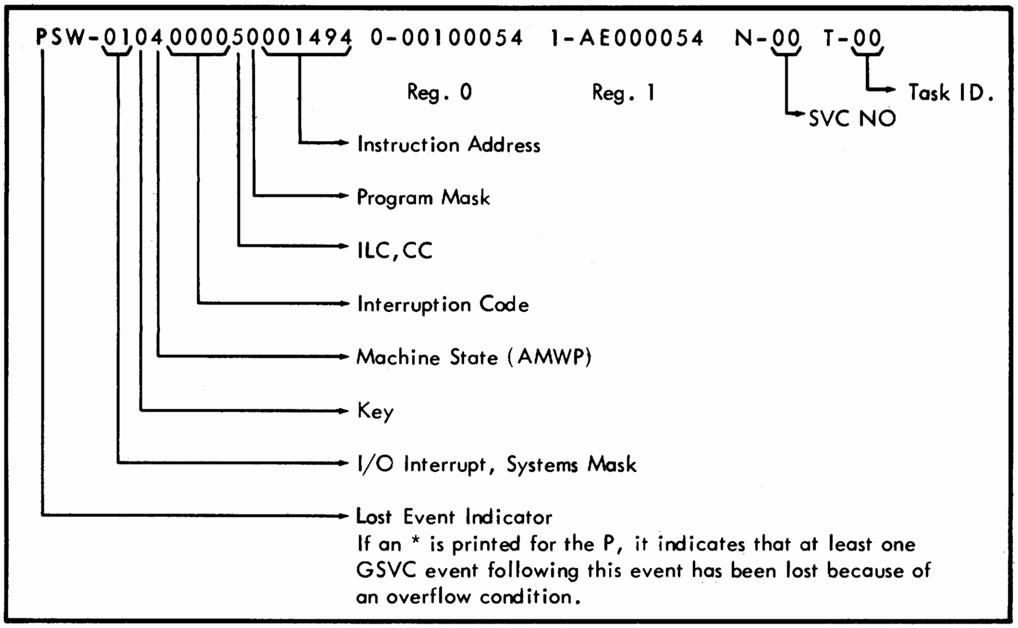
```
                                           CSW
PSW FF15000C8A003454  1000355808000000
```

```
                                          CSW
PSW 0000000EFF003648  0000000008000000
```

```
                                           CSW
PSW*FE07000C80000000  1000355808000000
```

Next Instruction Address
PGM Mask
ILC,CC
Device Address
Machine State (AMWP)
STG KEY
I/O Interrupt, Systems Mask
Indicates that at least one I/O event following this entry has been lost because of an overflow condition.

Figure 39. Sample Output for I/O Trace



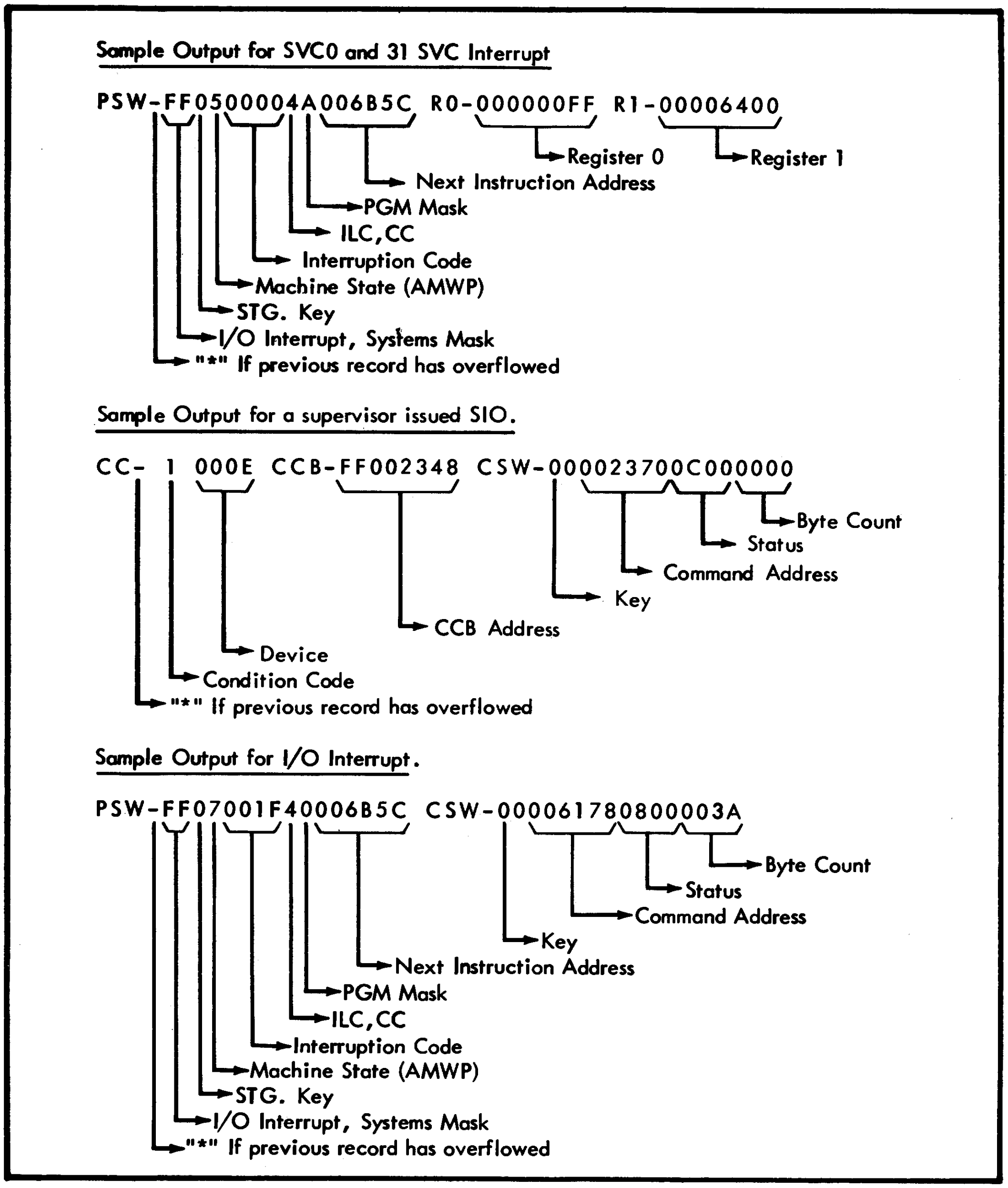
If an * is printed for the P, it indicates that at least one GSVC event following this event has been lost because of an overflow condition.

Figure 40. Sample Output for GSVC Trace

PDAID Usage

The following job stream is an example of how to execute PDAIDs from a card reader. 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
/*
/&
```



Sample Output for SVC0 and 31 SVC Interrupt

```
PSW-FF0500004A006B5C R0-000000FF R1-00006400
```

Sample Output for a supervisor issued SIO.

```
CC- 1 000E CCB-FF002348 CSW-000023700C000000
```

Sample Output for I/O Interrupt.

```
PSW-FF07001F40006B5C CSW-000061780800003A
```

Figure 41. Sample Output for QTAM Trace

Macro Supported	SVC		Function
	Dec.	Hex.	
EXCP	0	0	Execute channel programs.
FETCH	1	1	Fetch any phase.
	2	2	Fetch a logical transient (B-transient).
	3	3	Fetch or return from a physical transient (A-transient).
LOAD	4	4	Load any phase.
MVCOM	5	5	Modify supervisor communications region.
CANCEL	6	6	Cancel a problem program or task.
WAIT	7	7	Wait for a CCB or TECB.
	8	8	Transfer control to the problem program from a logical transient (B-transient).
LBRET	9	9	Return to a logical transient (B-transient) from the problem program after an SVC 8.
SETIME	10*	A	Set timer interval.
	11	B	Return from a logical transient (B-transient).
	12	C	Logical AND (Reset) to second job control byte (displacement 57 in communications region).
	13	D	Logical OR (Set) to second job control byte (displacement 57 in communications region).
EOJ	14	E	Cancel job and go to job control for end of job step.
	15	F	Same as SVC 0 except ignored if CHANQ table is full. (Primarily used by ERP).
STXIT (PC)	16*	10	Provide supervisor with linkage to user's PC routine for program check interrupts.
EXIT (PC)	17*	11	Return from user's PC routine.
STXIT (IT)	18*	12	Provide supervisor with linkage to user's IT routine for interval timer interrupts.
EXIT (IT)	19*	13	Return from user's IT routine.
STXIT (OC)	20*	14	Provide supervisor with linkage to user's OC routine for external or attention interrupts (operator communications).
EXIT (OC)	21*	15	Return from user's OC routine.
	22*	16	The first SVC 22 seizes the system for the issuing program by disabling multiprogram operation. The second SVC 22 releases the system (enables multiprogram operation).
	23*	17	Load phase header. Phase load address is stored at user's address.
SETIME	24*	18	Provide supervisor with linkage to user's TECB and set timer interval.
	25*	19	Issue HALT I/O on a teleprocessing device, or HALT I/O on any device if issued by OLTEP. For a MPS=YES system, dequeue an unstarted OLTEP I/O request to a shared device.
	26*	1A	Validate address limits.
	27*	1B	Special HIO on teleprocessing devices.

* optional

Figure 42. DOS Supervisor Calls (Part 1 of 2)

Macro Supported	SVC		Function
	Dec.	Hex.	
EXIT (MR)	28*	1C	Return from user's stacker select routine (MICR type devices only).
	29*	1D	Provide return from multiple wait macros WAITF and WAITM (except MICR type devices).
QWAIT	30*	1E	Wait for a QTAM element.
QPOST	31*	1F	Post a QTAM element.
	32 33 34	20 21 22	(Reserved). Reserved for internal macro COMRG. Reserved for internal macro GETIME.
HOLD	35*	23	Hold a track for use by the requesting task only.
FREE	36*	24	Free a track held by the task issuing the FREE.
STXIT (AB)	37*	25	Provide supervisor with linkage to user's AB routine for abnormal termination of a task.
ATTACH	38*	26	Initialize a subtask and establish its priority.
DETACH	39*	27	Perform normal termination of a subtask. It includes calling the FREE routine to free any tracks held by the subtask.
POST	40*	28	Inform the system of the termination of an event and ready any waiting tasks.
DEQ	41*	29	Inform the system that a previously enqueued resource is now available.
ENQ	42*	2A	Prevent tasks from simultaneous manipulation of a shared data area (resource).
	43* 44* 45* 46* 47* 48 49 50 51* 52*	2B 2C 2D 2E 2F 30 31 32 33 34	Provide supervisor support for external creation and updating of SDR records. Provide supervisor support for external creation of OBR records. Provide emulator interface. Provide OLTEP with the facility to operate in supervisory state. Provide return from wait multiple WAITF for MICR type device. (Reserved) (Reserved) Reserved for LIOCS error recovery. Return phase length at OLTEP request. Provide an interface between supervisor and end-of-job routines.

* optional

Figure 42. DOS Supervisor Calls (Part 2 of 2)

Environmental Recording, Editing, and Printing Program (EREP)

The EREP program edits and prints data that has been stored in the recorder file (SYSREC) by the I/O error logging and either MCRR or MCAR/CCH functions.

For IBM System/370 it can create and maintain a history tape and if specified a RDE tape of OBR/MCAR/CCH and if specified IPL/EOD Data.

Three variants of the EREP program are: one that processes System/360 records, one that processes System/370 records, and one that processes both System/360 and System/370 records. You must catalog the variant that is required for your SYSREC to the core image library before its execution. One of the job streams that follow can be used to catalog the variant that you need.

For System/360 Records

```
// JOB CATLEREP
// OPTION CATAL
   INCLUDE IJBECALL
// EXEC LNKEDT
/&
```

For System/370 Records

```
// JOB CATLEREP
// OPTION CATAL
   INCLUDE IJBECAL2
// EXEC LNKEDT
/&
```

For Both System/360 and System/370 Records

```
// JOB CATLEREP
// OPTION CATAL
   INCLUDE IJBECALL
// EXEC LNKEDT
   INCLUDE IJBECAL2
// EXEC LNKEDT
/&
```

If the variant that processes both System/360 and System/370 records is cataloged, then (after cataloging) you should execute MAINT and condense the core image library to eliminate unused space. EREP is not self-relocating. Thus, to execute EREP in a foreground partition, specify a load address when it is cataloged.

Note: If a variant of EREP (other than that which processes both types of records) is executed for a SYSREC file that contains both type of records, the job is canceled when EREP encounters a record it cannot process. If the correct variant is not used, the data on SYSREC is not processed or cleared.

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 when EREP is run in a 10K partition. 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. When the environment data is needed or the SYSREC file becomes full, EREP can be executed from SYSLOG or SYSRDR by:

```
// EXEC EREP
```

Then EREP issues a message to the operator via SYSLOG requesting the logical unit, either SYSLOG or SYSIPT, that is to be used for entering the EREP options. The operator must respond with one of the following:

- C followed by EOB or END for SYSIPT
- S followed by EOB or END for SYSLOG
- N followed by EOB or END or the latter for the default option, EDIT.

The EREP options are:

Name	Operation	Option
blank for SYSIPT and not used by SYSLOG	OPTION	$\left[\begin{Bmatrix}\text{EDIT}\\ \text{CLEAR}\\ \text{HIST}\left[\begin{Bmatrix}[\text{,NEW}][\text{,2}]\\ \text{,UPNEW}\end{Bmatrix}\right]\end{Bmatrix}\right]$

When entering the EREP options via SYSIPT column 1 must be blank and only one option per card is allowed (HIST with UPNEW or with NEW and/or 2 is considered one option). SYSIPT input must be delimited by an end of data file card (/* in columns 1 and 2).

When entering the EREP options via SYSLOG, the entry must not exceed 80 positions. Start with:

1. The operation, OPTION, followed by
2. A blank, followed by

3. The option, followed by

4. EOB or END.

Repeat this procedure for each option: When all the options have been specified, enter an EOB or END to continue processing.

Embedded blanks within the operation and the option are not allowed. A misspelled word, system error, duplicate option, or unsupported option can be corrected at SYSLOG by the operator. However, if they are not corrected, they are ignored. Multiple options are allowed by EREP. See Figure 40 for a summary of the EREP options.

EDIT: This option causes EREP to edit and print the contents of the IJSYSRC file on SYSLST. EREP displays SDR records first (in the order they appear on the disk) followed by OBR records (grouped by device addresses). Then either channel inboard records (for IBM System/360) or channel check handler records (for IBM System/370) followed by CPU machine check records (MCRR or MCAR for 360 and 370 respectively). The record counters are reset to zero after each SDR record is processed. Retain these printouts for those persons involved with problem determination. EDIT is forced if CLEAR, HIST, or HIST with any optional operands is specified.

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).

CLEAR: This option causes EREP to clear (reset) the OBR and either the MCRR or MCAR/CCH portion of the SYSREC file. This option forces the EDIT function. CLEAR is always the last EREP function performed. CLEAR is forced if HIST or HIST with any optional operands is specified.

HIST,NEW[,2]: This causes EREP to create a history file on the tape unit assigned to SYS009. If 2 is also specified a second history file is created on the same tape unit for RDE data. The data contained on both tapes is identical. The tape(s) contains the OBR/MCAR/CCH portion of the SYSREC file. Then the EDIT and CLEAR functions are forced (Note).

HIST[,2]: EREP updates the history file (SYS009) when the HIST option is specified. If 2 is also specified, the second history file for RDE data is updated. After the file is updated, the EDIT and CLEAR functions are forced (Note).

HIST,UPNEW: This causes EREP to first update a tape file assigned to SYS009 (either history or RDE), and then create a new tape file. After the tape files are processed, the EDIT and CLEAR functions are forced (Note). If UPNEW is specified, TLBL cards for updating and creating must be included in the job stream.

> Note: HIST or HIST with any optional operands are not valid for the 360 varient.

EREP History Tape(s)

The tapes (history and RDE) are created and updated by using the EREP history options. A magnetic tape unit assigned to SYS009 must be used for this function. EREPNEW must be the filename that is used when a tape is created, and EREPUP when a tape is updated (both TLBL cards must be included with UPNEW). When a tape becomes full or a second tape must be mounted, the operator is notified via SYSLOG.

Retain the history tape for those persons involved with problem determination. It can be used as input for certain on-line test programs of OLTEP. (See the OLTEP manual listed in the Preface.) Retain the RDE tape; it will be used by IBM.

Figures 43 and 44 show the use of the history options. Figure 45 summarizes the logical units used by EREP.

OPTION	Comments
EDIT	Edits and prints SYSREC on SYSLST
CLEAR	1. Edits and prints SYSREC on SYSLST 2. Clears SYSREC
HIST,NEW[,2] (Note)	1. Creates OBR/MCAR/CCH history file(s) on SYS009 2. Edits and prints SYSREC on SYSLST 3. Clears SYSREC
HIST[,2] (Note)	1. Updates OBR/MCAR/CCH history file(s) on SYS009 2. Edits and prints SYSREC on SYSLST 3. Clears SYSREC
EDIT followed by HIST,NEW or HIST (Note)	1. Edits and prints SYSREC on SYSLST 2. Creates or updates OBR/MCAR/CCH history file on SYS009 3. Clears SYSREC
HIST,UPNEW (Note)	1. Updates an OBR/MCAR/CCH file (either history or RDE) on SYS009 2. Creates an OBR/MCAR/CCH file on SYS009 3. Edits and prints SYSREC on SYSLST 4. Clears SYSREC
(none)	Edits and prints SYSREC on SYSLST

Note: Only valid for IBM System/370

Figure 43. EREP Options

```
// JOB EXAMPLE1
// TLBL EREPNEW
// ASSGN SYS009,X'cuu'
// EXEC EREP
   OPTION HIST,NEW
/*
/&
// JOB EXAMPLE2
// TLBL EREPUP
// ASSGN SYS009,X'cuu'
// ASSGN SYSLST,X'cuu'
// EXEC EREP
   OPTION EDIT
   OPTION HIST
/*
/&
EREPNEW and EREPUP must be the filenames
for a new history file(s) or for
updating.
```

Figure 44. Examples of IBM System/370 EREP Options

EREP, Special Conditions

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 non-SDR error records are printed, execute EREP again to print them.
- If an I/O error occurs during the reading of a particular record the entire record is skipped and processing continues with the next sequential record. EREP does not CLEAR the SYSREC file in such a case. Then to prevent redundant data from appearing, clear the file by re-IPLing and setting RF=CREATE.

SYSIPT	(Optional)
SYSLOG	(Required, must be assigned to a 1052 or a 3210 or 3215)
SYSLST	(Required)
SYSREC	(Required)
SYS009	(Optional, for history options of IBM S/370 and must be assigned to a magnetic tape unit)

Figure 45. Logical Units used by EREP

Error Statistics by Tape Volume Utility Programs

When DOS is generated, you have the option of requesting the collection of error statistics by tape volume (ESTV). This ESTV data can be printed on SYSLOG or stored on a direct access storage device.

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. At system generation time specify TEBV=(DASD,n) in the FOPT macro of the Supervisor.

2. Permanently assign SYSREC to a 2311, 2314, or 2319 device. You must do this at every IPL, if volume statistics are to be collected, unless an ASSGN for SYSREC has been included in the Supervisor assembly.

3. Catalog ESTVFMT into the core image library with the name IJBESTFM. Do this with the following statements:

```
// JOB CATALOG
// OPTION CATAL
   INCLUDE IJBESTFM
// EXEC LNKEDT
/&
```

4. Determine the size of the recorder file. (One track of a 2311 holds 33 ESTV records; one track of a 2314 or 2319 holds 48 ESTV records.)

5. Make the ESTVFLE accessible to the ESTV programs by including standard label information. Do this with the following statements:

```
// OPTION STDLABEL
// DLBL ESTVFLE,'error stat by tape
   volume',99/365,SD
// EXTENT SYSREC,nnnnnn,1,1,nnnn,nnnn
```

6. Format ESTVFLE immediately after the first IPL after system generation, or whenever new label information is entered for the file. Do this by entering from the console:

```
// EXEC ESTVFMT
```

When ESTVFMT completes its functions, the following message is printed on SYSLOG:

```
4R00I  ESTV FILE INITLZD, ENTRY IN
VTOC
```

No response is required to this message.

If ESTV data 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 gives you 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 you to keep the 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 SYSLOG. An operator must be at SYSLOG, 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 SYSREC,X'191'
// ASSGN SYS005,X'183'
// ASSGN SYS006,X'184'
// ASSGN SYSLST,X'00E'
// TLBL TAPEIN
// TLBL TAPEOUT,'VOLUME STATISTICS',99/365
// 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: Any symbolic device needed for a particular execution of ESTVUT must be assigned, either temporarily for the job or permanently.

- SYSLOG must be assigned to a 1052 or a 3210 or 3215 for all executions of ESTVUT to log system inquires and accept replies.
- SYSLST must be specified if the response to message 4R04A OUTPUT= is either PRC or PRNC. SYSLST may be assigned to the printer, to magnetic tape, or to disk.
- SYSREC is the symbolic device of the ESTV Recorder File (ESTVFLE). SYSREC must be permanently assigned in order to be available to the system for collection of statistics. The procedure for establishing ESTVFLE is discussed under ESTV Format Data Set Program (ESTVFMT).
- SYS005 must be assigned to a magnetic tape unit when the response to message 4R02A INPUT= is TAPE.
- SYS006 must be assigned to a magnetic tape unit when the response to message 4R03A OUTPUT= is TAPE.

LABEL INFORMATION: Label information for the disk file and the two tape files must be available to the system whenever the devices are used in the execution of ESTVUT.

- If a DLBL statement is included for the recorder file, the first operand must be ESTVFLE.
- The first operand of an input tape TLBL card must be TAPEIN.
- The first operand for an output tape TLBL card must be TAPEOUT.
- You may select values for all other label operands.
- A LBLTYP for tape is required only if the program 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 notifies 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 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 (for 2314 or 2319), 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 SYSLOG 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 you 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.

If you wish to keep all of the error data on one tape, then make 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 SYSLOG 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 SYSLOG.

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 or S
   INCLUDE IJBESTUT
   PHASE ESTVPR,+0 or S
   INCLUDE IJBESTPR
   PHASE ESTVMT,+0 or S
   INCLUDE IJBESTMT
// EXEC LNKEDT
/&
```

- +0 is used for a multiprogramming system.

- S is used for a nonmultiprogramming (batched-job) system.

DOS Stand-Alone Dump (DUMPGEN)

DUMPGEN produces a stand-alone dump program tailored to system requirements. The stand-alone dump that is generated is either a conventional dump program or a formatting dump program. The conventional dump displays the contents of each main storage position consecutively, but does not display the floating point registers. The formatting dump displays the supervisor table in a more readable format after displaying main storage in the same manner as the conventional dump, and displays the floating point registers.

Both dumps have the translate feature. During the conventional display, if the remaining portion of any line or a group of lines is identical to the first word to be printed, the first word and the word SAME is printed and printing is suspended until the contents of main storage changes. If any line to be printed is identical to the last line previously printed and the words of that line are different from each other, then printing is suspended until the contents of the line changes.

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 (X'00'-X'17') and the 640 bytes it requires 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 either its absolute or self-relocating version 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
   PHASE DUMPGEN,S or +0
   INCLUDE IJBDMPGN
// EXEC LNKEDT
/&
```

- S is used for non-MPS (batched-job) system
- +0 is used for a MPS system

DUMPGEN must be executed in any partition by the command:

```
       // EXEC DUMPGEN
```

DUMPGEN then reads its control statements from SYSIPT. Code the two types of control statements, ASSGN or OPTN.

ASSGN Statement: ASSGN defines the output device for the stand-alone dump. The format of the ASSGN statement is:

Name	Operation	Operands
blank	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:

Name	Operation	Operand

blank	OPTN	CORE=nnnnnK
		INTR={NO, YES}
		DECKS=nnnnnnnn
		LOADADR={D'nnnnnnnn', X'xxxxxx'}
		FORMAT={NO, YES}
		TAPEIPL={NO, YES}

CORE Defines the area of main storage, starting at position 0, 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).

INTR NO produces a stand-alone dump program which, when loaded, prints out the contents of main storage on the SYSLST printer defined with the ASSGN statement or X'00E'.

YES 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 9999999. 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 16766576. However, if the load address is not aligned to a doubleword, then the specified address is rounded high to the next double word. The specified address is checked for validity.

FORMAT If NO is specified or FORMAT is omitted, then a conventional dump is generated.

YES generates a formatting dump which formats and displays low core, displays the contents of main storage, and then formats and displays the DOS supervisor tables. The formatted display of the tables depends on the location of the BG communications region. If the stand-alone dump is loaded into the location of the communications region, the program is terminated when the formatted display of the tables is to occur.

TAPEIPL If NO is specified or TAPEIPL is omitted and SYSPCH is assigned to a tape unit, then the stand-alone dump records are written on tape preceded by an ASA character.

If YES is specified and SYSPCH is assigned to a tape unit, the stand-alone dump is written on tape and it may be IPLed directly from the tape unit.

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 DECKS=2 is followed by DECKS=5, then 5 stand-alone programs are 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. The name field must be blank.

5. Only one operation and only one operand per control statement is allowed.

6. One or more blanks must follow the operand if comments are desired.

7. 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 INTR=NO 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 DOS Messages publication listed in the Preface.

Label Cylinder Display (LSERV)

The label cylinder display program, LSERV, displays on SYSLST the TLBL and the DLBL and EXTENT information (except for secured data files) contained on the SYSRES label cylinder. The nonmultiprogramming (MPS=NO) version of LSERV is in the core image library(s) of the IBM-supplied DOS system. It is also supplied in the relocatable library (see the section LSERV Catalog Statements).

LSERV may be executed in any partition, with a minimum of 8192 bytes of main storage, by the statement

```
        // EXEC LSERV
```

LSERV requires no other control statements for its operation other than the normal job step or job termination controls, /* or /&.

LSERV assumes the SYSRES label cylinder is formatted as it is described in the DOS DASD Labels manual listed in the Preface.

LSERV Catalog Statements

You can use the following job stream to catalog LSERV to the core image library.

```
    // JOB
    // OPTION CATAL
       PHASE LSERV,S or +0
       INCLUDE IJBLSERV
    // EXEC LNKEDT
    /&
```

- S is required for a non-MPS system.
- +0 is required for an MPS system.

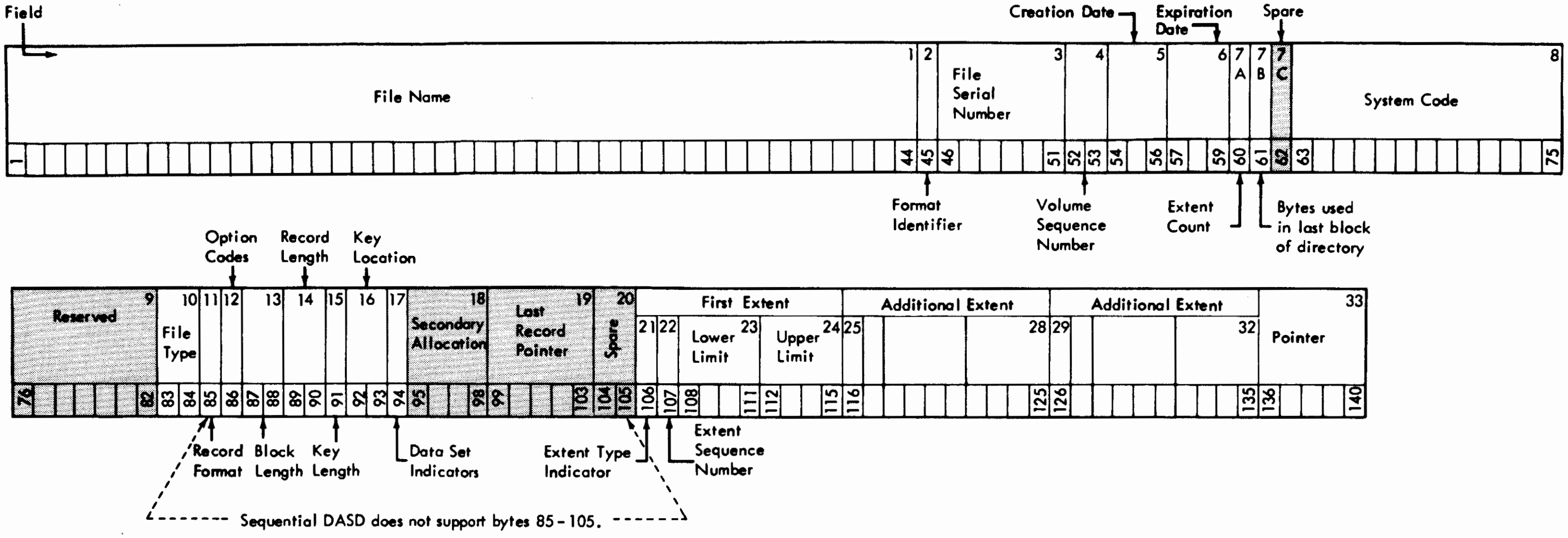


Figure 46. Standard DASD File Labels, Format 1 (Part 1 of 3)

Format 1: This format is common to all data files on Direct Access Storage Devices. (Shaded areas are not processed by DOS).

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.

FIELD NAME AND LENGTH DESCRIPTION

Note: The Disk Operating System compares the entire field against the file name given in the DLBL card. The generation and version numbers are treated differently by the Operating System.

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

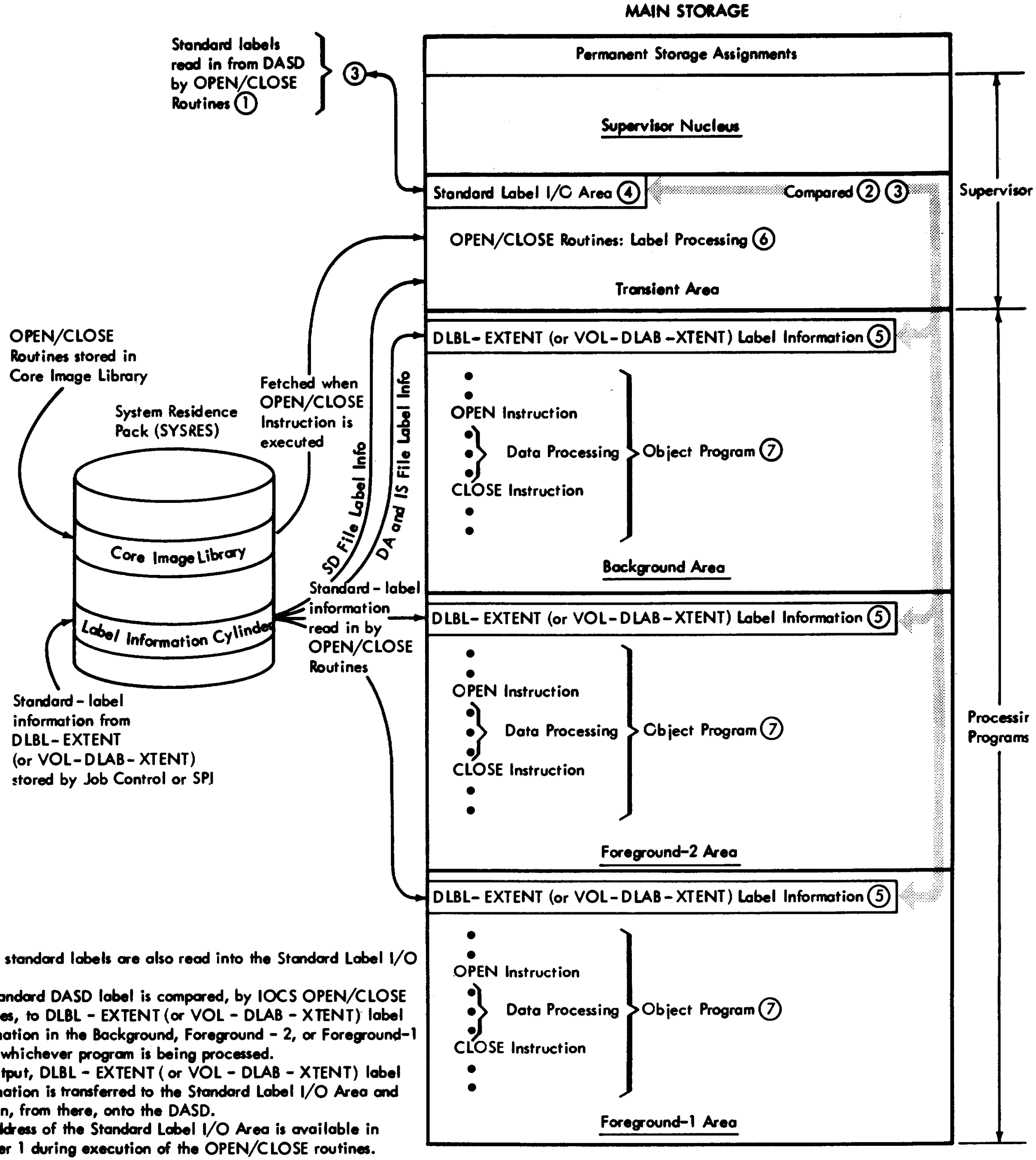
Indicates the year and the day of the year the file was created. It is of the form YDD, where Y signifies the year (0-99) and DD the day of the year (1-366).

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 EBCDIC characters. On input, IOCS ignores this field. On output, IOCS writes the information supplied in DLAB. If you use DLAB, IOCS writes: DOS/360 Ver. 3.
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, 2, 4, 6, 7: 0 for DOS Used by O/S 3 If on, DOS data security is invoked. 5 Used by DOS and 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 46. Standard DASD File Labels, Format 1 (Part 2 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 46. Standard DASD File Labels, Format 1 (Part 3 cf 3)


MAIN STORAGE
Permanent Storage Assignments
Supervisor Nucleus
Standard Label I/O Area ④
Compared ② ③
Supervisor
OPEN/CLOSE Routines: Label Processing ⑥
Transient Area
DLBL- EXTENT (or VOL-DLAB-XTENT) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑦
CLOSE Instruction
Background Area
DLBL- EXTENT (or VOL-DLAB-XTENT) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑦
CLOSE Instruction
Foreground-2 Area
DLBL- EXTENT (or VOL-DLAB-XTENT) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑦
CLOSE Instruction
Foreground-1 Area
Processir Programs
Standard labels read in from DASD by OPEN/CLOSE Routines ①
③
OPEN/CLOSE Routines stored in Core Image Library
System Residence Pack (SYSRES)
Fetched when OPEN/CLOSE Instruction is executed
SD File Label Info
DA and IS File Label Info
Core Image Library
Label Information Cylinder
Standard - label information read in by OPEN/CLOSE Routines
Standard - label information from DLBL- EXTENT (or VOL-DLAB- XTENT) stored by Job Control or SPI

Notes:

1. User - standard labels are also read into the Standard Label I/O Area.
2. The standard DASD label is compared, by IOCS OPEN/CLOSE routines, to DLBL - EXTENT (or VOL - DLAB - XTENT) label information in the Background, Foreground - 2, or Foreground-1 area, whichever program is being processed.
3. On output, DLBL - EXTENT (or VOL - DLAB - XTENT) label information is transferred to the Standard Label I/O Area and written, from there, onto the DASD.
4. The address of the Standard Label I/O Area is available in register 1 during execution of the OPEN/CLOSE routines.
5. The standard - label information areas (from DLBL - EXTENT
or or VOL - DLAB - XTENT) are determined by the Linkage Editor with the specifications of //LBLTYP statements.
6. For SD files, label information is read from label information cylinder into transient area. For DA or IS files, label information is read from label information cylinder into the problem program area.
7. If a self - relocating program is executed, OPENR/CLOSER instructions are used instead of OPEN/CLOSE.

Figure 47. Overview of DOS DASD Label Processing

Appendix B: Standard Tape File Labels

Field	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	Label Identifier	File Label Number	File Identifier	Set Identifier	File Section Number	File Sequence Number	Generation Number	Version Number of Generation	Creation Date	Expiration Date	Accessibility	Block Count	System Code	Reserved
	1 – 3	4	5 – 21	22 – 27	28 – 31	32 – 35	36 – 39	40 – 41	42 – 47	48 – 53	54	55 – 60	61 – 73	74 – 80

The American National Standards Institute Inc. standard tape file label format and contents are as follows: (Shaded areas are not processed by DOS/360)

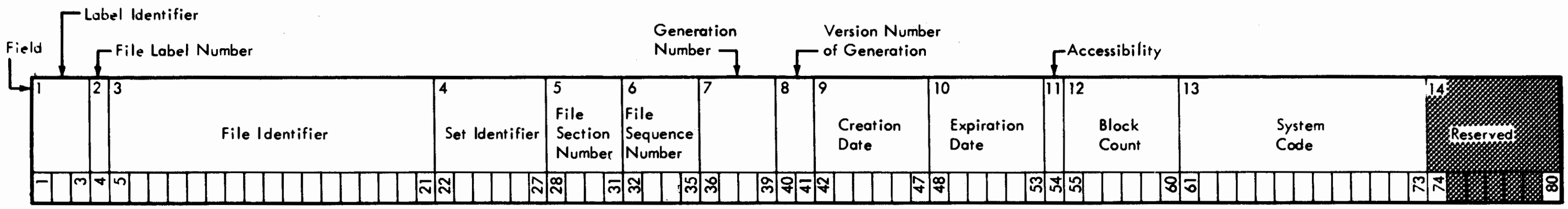
FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, ASCII	Identifies the type of label. HDR = Header--beginning of a data file. EOF = End of File--end of a set of data. EOV= End of Volume--end of the physical reel.
2.	FILE LABEL NUMBER 1 byte, ASCII	Indicates the sequence of this label within 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 contain any characters.
4.	SET IDENTIFIER 6 bytes, ASCII	Identifies the volume/file relationship. This field is identical to the volume serial number in the volume label of the first or only volume of a multivolume file or a multifile set. This field will normally be numeric (000001 - 999999) but may contain any 6 alphameric characters.
5.	FILE SECTION NUMBER 4 bytes	Indicates the order of a volume in a given file or multifile set. The first volume can be any number (default=0001). Subsequent numbers must be in proper numeric sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	Assigns numeric sequence to a file within a multifile set. The first file can be any number (default = 0001). Subsequent numbers must be in proper numeric sequence.
7.	GENERATION NUMBER 4 bytes	Uniquely identifies the various editions of the file. Must be spaces* or from 0001 to 9999 in proper numeric sequence.
8.	VERSION NUMBER OF GENERATION 2 bytes	Indicates the version of the generation of a file. Must be spaces* or from 00 to 99 in proper numeric sequence. (default=00)

9.	CREATION DATE
	6 bytes
		Indicates the year and the day of the year that this file was created. Format is (ɓyyddd):
		ɓ = space*
		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. Format is identical to 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 on the file from the last header label to the first trailer label, exclusive of tapemarks.
13.	SYSTEM CODE 13 bytes	Uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved for future use; should be recorded as spaces.*

*ASCII (American National Standard Code for Information Interchange) space correlates to EBCDIC blank.

Figure 48. IBM Standard Tape File Label



The American National Standards Institute Inc. standard tape file label format and contents are as follows: (Shaded areas are not processed by DOS/360)

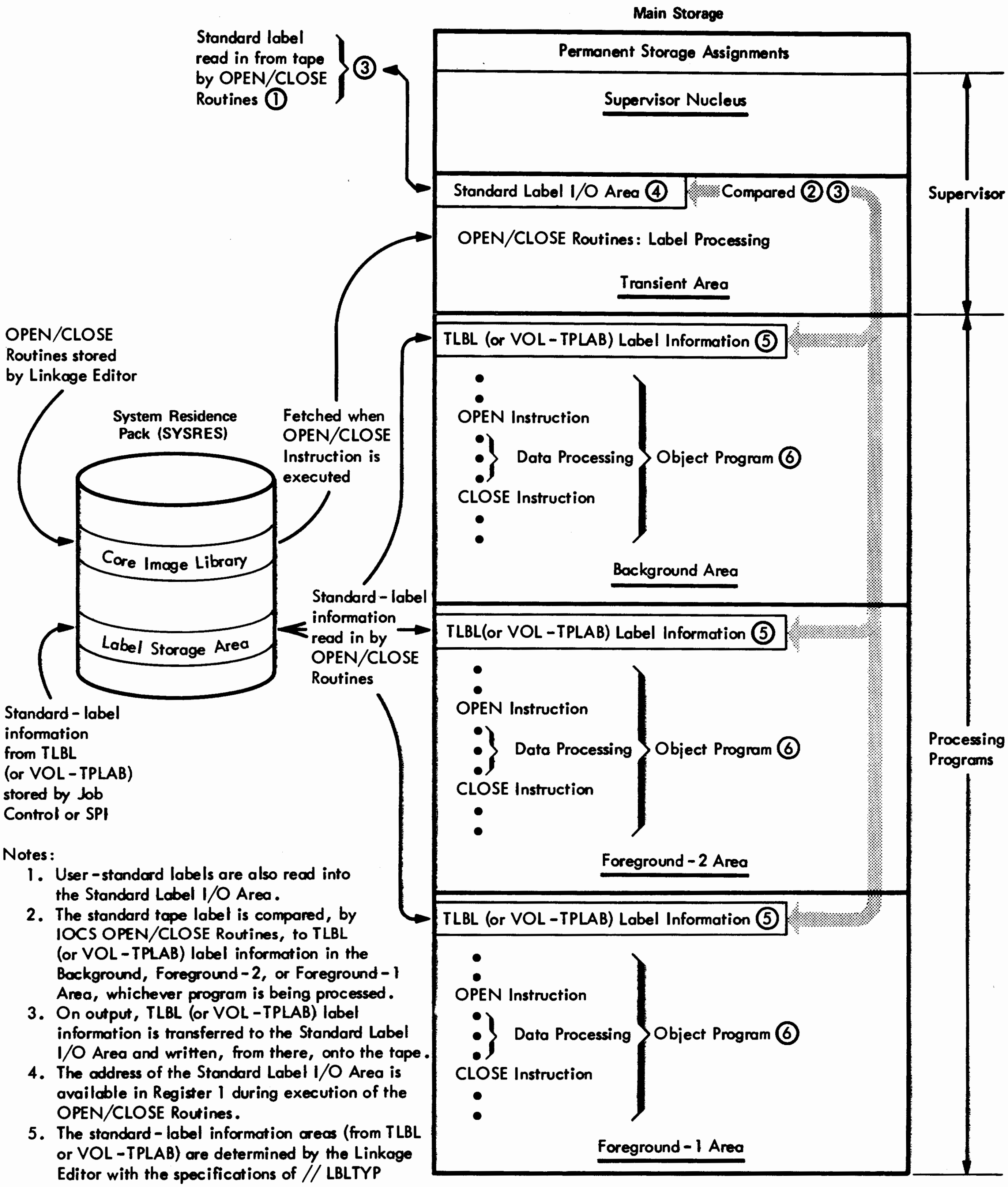
FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, ASCII	Identifies the type of label. HDR = Header--beginning of a data file. EOF = End of File--end of a set of data. EOV = End of Volume--end of the physical reel.
2.	FILE LABEL NUMBER 1 byte, ASCII	Indicates the sequence of this label within 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 contain any characters.
4.	SET IDENTIFIER 6 bytes, ASCII	Identifies the volume/file relationship. This field is identical to the volume serial number in the volume label of the first or only volume of a multivolume file or a multifile set. This field will normally be numeric (000001 - 999999) but may contain any 6 alphameric characters.
5.	FILE SECTION NUMBER 4 bytes	Indicates the order of a volume in a given file or multifile set. The first volume can be any number (default = 0001). Subsequent numbers must be in proper numeric sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	Assigns numeric sequence to a file within a multifile set. The first file can be any number (default = 0001). Subsequent numbers must be in proper numeric sequence.
7.	GENERATION NUMBER 4 bytes	Uniquely identifies the various editions of the file. Must be spaces* or from 0001 to 9999 in proper numeric sequence.
8.	VERSION NUMBER OF GENERATION 2 bytes	Indicates the version of the generation of a file. Must be spaces* or from 00 to 99 in proper numeric sequence. (default = 00)

9.	CREATION DATE
	6 bytes
		Indicates the year and the day of the year that this file was created. Format is (b̸yyddd):
		b̸ = space*
		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. Format is identical to 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 on the file from the last header label to the first trailer label, exclusive of tapemarks.
13.	SYSTEM CODE 13 bytes	Uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved for future use; should be recorded as spaces.*

*ASCII (American National Standard Code for Information Interchange) space correlates to EBCDIC blank.

Figure 49. ANSI (American National Standards Institute, Inc.) Standard Tape File Label


Main Storage
Permanent Storage Assignments
Supervisor Nucleus
Standard label read in from tape by OPEN/CLOSE Routines ① ③
Standard Label I/O Area ④
Compared ② ③
OPEN/CLOSE Routines: Label Processing
Transient Area
Supervisor
OPEN/CLOSE Routines stored by Linkage Editor
System Residence Pack (SYSRES)
Fetched when OPEN/CLOSE Instruction is executed
Core Image Library
Label Storage Area
Standard - label information read in by OPEN/CLOSE Routines
Standard - label information from TLBL (or VOL - TPLAB) stored by Job Control or SPI
TLBL (or VOL - TPLAB) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑥
CLOSE Instruction
Background Area
TLBL(or VOL -TPLAB) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑥
CLOSE Instruction
Foreground - 2 Area
TLBL (or VOL -TPLAB) Label Information ⑤
OPEN Instruction
Data Processing
Object Program ⑥
CLOSE Instruction
Foreground - 1 Area
Processing Programs

Notes:

1. User-standard labels are also read into the Standard Label I/O Area.
2. The standard tape label is compared, by IOCS OPEN/CLOSE Routines, to TLBL (or VOL-TPLAB) label information in the Background, Foreground-2, or Foreground-1 Area, whichever program is being processed.
3. On output, TLBL (or VOL-TPLAB) label information is transferred to the Standard Label I/O Area and written, from there, onto the tape.
4. The address of the Standard Label I/O Area is available in Register 1 during execution of the OPEN/CLOSE Routines.
5. The standard-label information areas (from TLBL or VOL-TPLAB) are determined by the Linkage Editor with the specifications of // LBLTYP statements.
6. If a self-relocating program is executed, OPENR/CLOSER instructions are used instead of OPEN/CLOSE.

Figure 50. Overview of DOS Standard Tape Label Processing

Appendix C: Summary of Statements and Commands

Appendix C contains a summary of the following:

- Job Control Statements (JCS) which must be preceeded by // blank in positions 1, 2, and 3.

- Job Control Commands (JCC).

- Attention Routine Commands (AR).

- Single Program Initiation Commands (SPI).

Following Figure 51 are examples of job streams.

<table>
<tr><th>Name</th><th>Operation</th><th>Operand</th><th>Remarks</th></tr>
<tr><td></td><td>ALLOC
accepted by
JCC, AR</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></td><td>ALTER
accepted by
AR</td><td>XXXXXX</td><td>Alters 1 to 16 bytes of main storage.
XXXXXX is the hexadecimal address where alteration is to start.</td></tr>
<tr><td>[//]</td><td>ASSGN
accepted by
JCS
JCC
SPI</td><td>SYSxxx, address [{,X'ss'}
[{,ALT}] [TEMP]</td><td>SYSxxx: can be SYSRDR
SYSIPT
SYSIN - Invalid for SPI
SYSPCH
SYSLST
SYSOUT
SYSLOG
SYSLNK } Invalid for SPI
SYSREC
SYSRLB
SYSSLB
SYSCLB - Only valid for JCC
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 (Invalid for SYSCLB, SYSRDR, SYSIPT, and SYSIN)

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 (Invalid for SYSCLB)
TEMP: only valid for JCC. Assignment for logical unit is destroyed by next JOB statement.</td></tr>
</table>

Figure 51. Job Control Summary (Part 1 of 8)

Name	Operation	Operand	Remarks
	BATCH accepted by: AR	{ BG F1 F2 }	Start, or continue batched-job operation
	CANCEL accepted by: AR	{ BG F1 F2 }	Cancels execution of current job in specified area
	CANCEL accepted by: JCC & SPI	blank	Cancels execution of current job
[//]	CLOSE accepted by: JCS JCC	SYSxxx [{ , X'cuu' [, X'ss'] , UA , IGN , ALT }]	SYSxxx: for magnetic tape - SYSPCH, SYSLST, SYSOUT, SYS000 - SYSmax for DASD (JCC only) - SYSIN, SYSRDR, SYSIPT, SYSPCH, SYSLST X'cuu', X'ss', UA, IGN, ALT: Values as described in ASSGN command
//	DATE accepted by: JCS	mm/dd/yy or dd/mm/yy	mm: Month (01-12) dd: day (01-31) yy: year (00-99)
[//]	DLAB accepted by: JCS SPI	'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 following 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 alphmeric). 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 accepted by: JCS SPI	filename, ['file-ID'] , [date] , [codes] , [data security] (See note 1)	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 data security: one to three characters
	DSPLY accepted by AR	XXXXXX	Displays 16 bytes of main storage

Figure 51. Job Control Summary (Part 2 of 8)

Name	Operation	Operand	Remarks
	DUMP accepted by: AR	{ blank S BG F1 F2 BGS F1S F2S CEAREA XXXXXX,XXXXXX } { (BG) (F1) (F2) }*	Dumps specified areas of main storage *Optional parameter. Dumps on SYSLST of specified partition. Default is BG SYSLST. blank: main storage without registers and supervisor S: main storage with registers and supervisor BG, F1, F2: Applicable partition and registers BGS, F1S, F2S: Applicable partition, registers and the supervisor. CEAREA; CE table, CE area, and AAA XXXXXX,XXXXXX: Specified area of main storage between these hexadecimal addresses
	DVCDN accepted by: JCC	X'cuu'	X'cuu': c=0-6 uu=00-FE (0-254) in hex
	DVCUP accepted by: JCC	X'cuu'	X'cuu': c=0-6 uu =00-FE (0-254) in hex

	EOB or END
	accepted by:
	JCC, AR
	SPI
		blank	end of SYSLOG communications
			EOB for System/360
			END for System/370

[//]	EXEC accepted by: JCS, SPI	[progname]	progname: one to eight alphameric characters. Used only if the program is in the core image library
[//]	EXTENT accepted by: JCS SPI	[symbolic unit] , [serial number] , [type] , [sequence number] , [relative track] , [number of tracks] , [split cylinder track] , [B=bins] (See Note 1)	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 accepted by: JCC, SPI	{ F1 [, F2] F2 [, F1] }	Causes assignments for foreground logical units to be held accross jobs
//	JOB accepted by: JCS	jobname [accounting information]	jobname: one to eight alphameric characters accounting information: one to 16 characters
[//]	LBLTYP accepted by: JCS SPI	{ 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 51. Job Control Summary (Part 3 of 8)

Name	Operation	Operand	Remarks
[//]	LISTIO accepted by: JCS JCC	$\left\{\begin{matrix}\text{SYS}\\ \text{PROG}\\ \text{F1}\\ \text{F2}\\ \text{ALL}\\ \text{SYSxxx}\\ \text{UNITS}\\ \text{DOWN}\\ \text{UA}\\ \text{X'cuu'}\end{matrix}\right\}$	Causes listing of I/O assignments on SYSLST for JCS and SYSLOG for JCC.
	LISTIO accepted by: SPI	$\left\{\begin{matrix}\text{BG}\\ \text{F1}\\ \text{F2}\\ \text{UA}\\ \text{ALL}\end{matrix}\right\}$	Causes listing of specified I/O assignments on SYSLOG
	LOG accepted by: JCC AR SPI	blank	Causes logging of job control statements and single program initiation commands on SYSLOG.
	MAP accepted by: JCC AR SPI	blank	Causes a map of areas in main storage to be printed on SYSLOG.
	MODE accepted by: AR	$\left\{\begin{matrix}\text{R}\\ \text{STATUS}\\ \\ \text{HIR}\\ \text{ECC}\left[,\left\{\begin{matrix}\text{M}\\ \text{C}\end{matrix}\right\}\right]{}^{*}\left\{\left[,\left\{\begin{matrix}\text{R}\\ \text{Q}\\ \text{TH}\end{matrix}\right\}\right][,\text{E=eeee}][,\text{T=tttt}]\right\}^{*}\end{matrix}\right\}$	Changes the mode of operation, changes the EFL threshold values, and gives status information. Available only for IBM System/370. *Note: When HIR or ECC is specified, at least one (1) of the optional operands within these braces must be selected. TH is only valid for the Model 145 when ECC,C is specified with the MODE command.
	MSG accepted by: AR SPI	$\left\{\begin{matrix}\text{F1}\\ \text{F2}\end{matrix}\right\}$	Transfers control to single program message routine.
[//]	MTC accepted by: JCS JCC	opcode,$\left\{\begin{matrix}\text{SYSxxx}\\ \text{X'cuu'}\end{matrix}\right\}$[,nn]	opcode: BSF, BSR, ERG, FSF, FSR, REW, RUN, or WTM SYSxxx: any logical unit X'cuu': (only valid for JCC) c=0-6 uu=FE(0-254) in hex nn: decimal number (01-99)
	NOLOG accepted by: JCC AR SPI	blank	Supresses logging of job control statements and single program initiation commands on SYSLOG.

Figure 51. Job Control Summary (Part 4 of 8)

Name	Operation	Operand	Remarks
//	OPTION accepted by: JCS	option 1 [, option 2, . . .]	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 sumbol 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 SYSPARM='string' specifies a value for assembler system variable symbol and SYSPARM
[//]	PAUSE accepted by: JCS JCC SPI	[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.
	PAUSE accepted by: AR	{BG F2 F1} [, EOJ]	Causes pause at end of current job step or at end of job.
	READ accepted by: SPI	X'cuu'	X'cuu': c = 0-6 uu = 00-FE (0-254) in hex Note: Device must be a card reader.
	RELSE accepted by: JCC SPI	{F1 [, F2] F2 [, F1]}	Causes single program logical units to be unassigned at EOJ.

Figure 51. Job Control Summary (Part 5 of 8)

Name	Operation	Operand	Remarks
[//]	RESET accepted by: JCS JCC	SYS PROG ALL SYSxxx	Resets I/O device assignments
	ROD accepted by: JCC	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.
//	RSTRT accepted by: JCS	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
	SET accepted by: JCC	[DATE=value 1] [, CLOCK=value 2] [, UPSI=value 3] [, LINECT=value 4] [, RCLST=value 5] [, RCPCH=value 6] [, RF=value 7]	value 1: 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) value 2: in the following format hh/mm/ss hh: hours (00-23) mm: minutes (00-59) ss: seconds (00-59) value 3: 0, 1, or X value 4: standard number of lines for output on each page of SYSLST value 5: decimal number indicating minimum number of SYSLST disk records remaining to be written before operator warning value 6: decimal number indicating minimum number of SYSPCH disk records remaining to be written before operator warning value 7: defines to the system the status of the recorder file (IJSYSRC) on SYSREC used by the I/O error logging (OBR/SDR) the machine check recording and recovery (MCRR), and the machine check analysis and recording and channel check handler (MCAR/CCH) features. RF = YES - File Exists NO - File does not Exist CREATE - Create a File Note: RF=NO is invalid if a IBM System/370 model is specified during system generotion.

Figure 51. Job Control Summary (Part 6 of 8)

Name	Operation	Operand	Remarks
	START accepted by: AR	{ BG F1 F2 }	Initiates a single program foreground program or resumes batched-job operations.
	STOP accepted by: JCC	blank	Stops batched-job program processing
	TIMER accepted by: AR SPI	{ BG F1 F2 }	Causes interval timer support to be given to the specified partition.
[//]	TLBL accepted by: JCS SPI	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 or d-dddd) { [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 accepted by: JCS SPI	'label fields 3-10'	'label fields 3-10': Indicated fields of the standard tape file label for either EBCDID or ASCII files. A 49-byte character string, contained within apostrophes.
[//]	TPLAB accepted by: JCS SPI	'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)
	UCS accepted by: JCC SPI	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 accepted by: JCC, SPI	{ F1 [, F2] F2 [, F1] }	Causes immediate unassignment of foreground logical units.
	UNBATCH accepted by: JCC	blank	Terminates batched-job foreground processing.

Figure 51. Job Control Summary (Part 7 of 8)

Name	Operation	Operand	Remarks
//	UPSI accepted by: JCS	nnnnnnnn	n: 0, 1, or X
[//]	VOL accepted by: JCS, SPI	SYSxxx, filename	SYSxxx: can be SYS000-SYSmax filename: one to seven alphameric characters, the first of which must be alphabetic
[//]	XTENT accepted by: JCS SPI	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 2314/2319; 0 - 9 for 2321 C_1C_1 = 00 for 2311 or 2314/2319; 00 - 19 for 2321 $C_2C_2C_2$ = 000-199 for 2311 or 2314/2319; 000-009 for 2321 H_1 = 0 for 2311 or 2314/2319; 0-4 for 2321 H_2H_2 = 00-09 for 2311; 00-19 for 2321 or 2314/2319 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 2314/2319; 0 - 9 for 2321
/*	ignored accepted by: JCS	ignored	columns 1 and 2 are the only columns checked.
/&	ignored accepted by: JCS	[comments]	columns 1 and 2 are the only columns checked. Comments are printed on SYSLOG and SYSLST at EOJ.
*	accepted by: JCS	comments	column 2 must be blank.

Note 1. If the DLBL and EXTENT statements for a private core image library are in the input stream (that is, the information is not contained on the label cylinder), they must precede the ASSGN SYSCLB command.

Figure 51. Job Control Summary (Part 8 of 8)

Standard Label Storage Example

```
    // JOB STDLABEL
    *  WRITE STANDARD LABELS ON THE STANDARD LABEL TRACK
1.  // OPTION STDLABEL
2.  // DLBL IJSYSRS,'DOS SYSTEM RESIDENCE FILE',99/365,SD
3.  // EXTENT SYSRES,123456,1,1,1,1269
    // DLBL IJSYSLN,'SYSTEM WORK FILE NO.  0',99/365,SD
    // EXTENT SYSLNK,123456,1,1,1340,140
    // DLBL IJSYS01,'SYSTEM WORK FILE NO.  1',99/365,SD
    // EXTENT SYS001,123456,8,1,1490,150,2
    // DLBL IJSYS02,'SYSTEM WORK FILE NO.  2',99/365,SD
    // EXTENT SYS002,123456,8,1,1493,150,5
    // DLBL IJSYS03,'SYSTEM WORK FILE NO.  3',99/365,SD
    // EXTENT SYS003,123456,8,1,1496,200,9
4.  // DLBL IJSYSIN,'COMBINED SYSRDR & SYSIPT',99/365,SD
    // EXTENT SYSIN,123456,1,1,1270,26
5.  /&
6.  ASSGN SYSIN,X'190'
```

Explanation for Standard Label Storage

The purpose of this job is to build in the standard (permanent) label storage area sets of label storage data, which are accessible to any partition and which remain until replaced by another job using option STDLABEL. These particular labels represent the SYSRES volume, the work files required for compilations and linkage editing, and a combined SYSRDR-SYSIPT file on disk. Assume that only the latter file is to be added to those already in the label storage area.

Statement 1: The STDLABEL option causes all label data subsequently submitted to be written at the beginning of the standard label track. Therefore, although only the label data for SYSIN needs to be added to the information already present, all label data sets must be resubmitted for any file still needed that was previously on the standard track.

Statement 2: The following points are of interest concerning the parameters in the DLBL and EXTENT statements:

- 123456 corresponds to the volume serial number assigned when the volume was initialized.
- 99/365 is used as the expiration date on the work files to protect against use of the area by another file.
- SYSLNK and SYS003 have separate extents to permit compile-and-execute mode. Faster compile time would result if two work files, such as SYS001 and SYS002, were placed on a second drive.
- SD could have been omitted, because the parameter is assumed. See the label set for SYSIN. Only sequential files are acceptable for standard label storage.

Statement 3: The EXTENT statement follows the DLBL statement. Work files 1-3 use split cylinders. File 1 occupies tracks 0-2, file 2 has tracks 3-5, and file 3 has tracks 6-9. The concept of split cylinders takes advantage of two facts.

1. An entire cylinder can be accessed with no loss of time for access mechanism movement, and
2. Movement time is related to the number of cylinders involved.

Splitting cylinders between two or more files reduces the access arm movement. Only sequential files can occupy split cylinders. Because the beginning of all files is on the same cylinder, one setting services all files initially. The advantage of split cylinders is greatest when the files are processed at about the same rate. However, even if they are not, the total number of cylinders moved may be less than that required when files are located in entirely separate cylinders. The VTOC for this volume is in cylinder 199.

Statement 4: This data set represents a combined SYSRDR-SYSIPT file on disk. If the data set were for SYSIPT or SYSRDR, only the SYSxxx entries would change; the IJSYSIN would remain. This indicates that file storage data can exist at one time for only one of the three possibilities: SYSRDR, SYSIPT, or SYSIN. SYSIN is the required name for a combined file.

Statement 5: Each DLBL statement causes the preceding label set to be written into the label storage area. The /& writes the final label set and restores the system to the user label mode.

Statement 6: If the job input stream is on disk at this time, the ASSGN statement cpens the file making use of the label storage data. Execution of the job stream on disk follows, because job control looks immediately to the disk for its next statement.

Creation of SYSIN on Disk Example

```
    // JOB BUILDIN
    *  BUILD COMBINED SYSRDR & SYSIPT ON DISK
1.  // ASSGN SYS004,X'00C'
    // ASSGN SYS005,X'190'
2.  // DLBL UOUT,'COMBINED SYSRDR & SYSIPT',99/365,SD
    // EXTENT SYS005,123456,1,1,1270,26
3.  // EXEC CDDK
4.  // UCD TC,FF,A=(80,80),B=(80,80),OY
    // END
    // JOB A
          .
          .   (Other job control statements as required)
          .
    // EXEC PROGA1
          .
          .   (Data for A1 if required)
          .
    /* SOME PUNCH
          .
          .   (Job control statements as required)
          .
5.  // EXEC PROGA2                                                   Input to SYS004
          .
          .   (Data for A2 if required)
          .
    /* SOME PUNCH
    /& SOME PUNCH
    // JOB B
          .
          .   (Other job control statements as required)
          .
    // EXEC PROGB
          .
          .   (Data for B if required)
          .
    /* SOME PUNCH
    /& SOME PUNCH
6.  CLOSE SYSIN,X'00C'
7.  /*
    /&
```

Explanation for Creation of SYSIN on Disk

The card-to-disk utility in this example creats a combined SYSRDR and SYSIPT file on disk. The resultant file can be used in conjunction with the previous example.

Statement 1: Temporary assignments are made for the input and output files in accordance with the utility program specifications. Because disks can be assigned to any SYSnnn, the disk could have been assigned to any SYSnnn other than SYS004, which is reserved for theinput unit. The system units are assigned already because of the assumed configuration.

Statement 2: DLBL and EXTENT statements are submitted to provide label data. The system file is sequential. It must occupy a single extent, but the extent type may be 1 (no split cylinder) or 8 (split cylinder).

Statement 3: The program name for the card-to-disk utility in this example is CDDK.

Statement 4: The // UCD and // END statements represent the control statement input, which specifies the particular file conditions to the generalized utility program. Details on the control statement requirements are given in IBM System/360 Disk and Tape Operating Systems, Utility Program Specifications, C24-3465.

Statement 5: Following the END statement, the utility program looks to SYS004 for the input file data. Because this is the card reader, the data to create SYSIN follows immediately.

- /& statements define the end of input for each job to handle abnormal end-of-job situations. All /* and /& statements must have some punch in columns 4-80. Otherwise, the utility program does not accept them as data. Position 3 must be blank to permit proper action when the data serves as SYSIPT.

Statement 6: A CLOSE command is included at the end of the file to perform the requirement of closing the system file and returning the assignment to the card reader.

Statement 7: The /*, without any punch in positions 3-80, signals the end of card data for the uility program. The /& performs the usual end-of-job functions.

Operating Considerations

When SYSRDR, SYSIPT, or SYSIN are on tape or disk, certain operating conditions should be considered.

- After a SYSIN or SYSRDR job stream has been prepared on tape or disk, it may be necessary to interrupt the normal schedule to execute a special rush job. By inserting a PAUSE statement (//) between jobs (that is, between the /& and the // JOB statement) at the time the job stream is created, control is given to the operator immediately, before the // JOB statement is read. At this point a temporary assignment for SYSIN can be made to the card reader. The temporary assignment is not reset by the // JOB statement in the card reader but reverts to the permanent tape or disk assignment when the /& statement is read in the card reader. The original stream restarts at the point of interruption. Because this method anticipates interruption, the pause may have to be nullified by entering a EOB/END through SYSLOG.

- Care must be taken in the use of the ATTN PAUSE or PAUSE command for the purpose of changing assignments, because they bring control to the operator the next time job control is in core. This occurs at the end of a job step and not necessarily at the end of a job.

- Remember that when the system opens a SYSRDR, SYSIPT, or combined SYSRDR/SYSIPT file on disk, it always refers to IJSYSIN. Therefore, although several such files may be present on disk simultaneously, each must have a unique file name, and the label storage area may reference only one of the files at a time.

Sequential Disk File Labels, Single Drive Example

```
    // JOB SAMLABEL
    *  LABELS FOR A MULTI-VOLUME SEQUENTIAL DISK FILE WITH ONE DRIVE
1.  // ASSGN SYS005,X'191'
2.  // DLBL FILENAME,'ALPHAMERIC FILE NAME',99/365,SD
    // EXTENT SYS005,000020,1,0,1320,190
    // EXTENT SYS005,000020,1,1,8,740
    // EXTENT SYS005,000100,1,2,1275,64
    // EXTENT SYS005,000006,8,3,50,636,6
3.  // EXEC PROG100
4.  /&
```

Explanation for Sequential Disk File Labels, Single Drive

PROG100 requires access to a sequential disk file located on three different packs that are to be mounted successively on the same drive (SYS005). The job stream shown in this example shows the label statements required.

Statement 1: The statement is included only to emphasize the relationship between assignment and the label statements. The EXTENT statements indicate the SYSxxx required.

Statement 2: Although this file consists of four extents on three packs, only one DLBL statement is required. The volume serial number in the first EXTENT statement corresponds to the volume serial number for the first extent. One EXTENT statement is submitted for each of the four extents, and the statements must be placed into the job stream in extent number sequence. One of the extents occupies a split cylinder area to illustrate that this is acceptable for sequential files. All of the label statements are written on the background temporary track, because USRLABEL mode is in effect. This is the only data in that area, because writing starts at the beginning of track 1.

Statement 3: Logical IOCS in PROG100 opens the first extent by using the filename in the DLBL statement, the alphameric filename in the DLBL statement, and the volume serial number and SYSxxx unit supplied in the first EXTENT statement (0 sequence number) to locate the actual label in the VTOC. When PROG100 has processed all of the data for the first extent, logical IOCS opens the second extent, based on the extent sequence number.

Note: The first two extents are on the same volume.

When the open routine has checked the volume serial number (000100) in the third extent statement against the volume serial number (000020) in the volume label of the disk volume currently mounted on SYS005, it notifies the operator of the discrepancy. The first volume (000020) can be removed, and the new volume (000100) mounted on SYS005. The operator's reply causes the cpen routine to process the new volume.

Statement 4: The background label storage area is cleared when the /& statement is encountered. Therefore, if the next job requires the same file, the label statements must be resubmitted.

Sequential Disk File Labels, Multiple Drives Example

```
    // JOB SAMLABEL
    *  LABELS FOR A MULTI-VOLUME SEQUENTIAL DISK FILE WITH MULTIPLE DRIVES
    // ASSGN SYS005,X'191'
1.  // ASSGN SYS006,X'192'
    // ASSGN SYS007,X'193'
    // DLBL FILENAME,'ALPHAMERIC FILE NAME',99/365,SD
2.  // EXTENT SYS005,000020,1,0,1320,190
    // EXTENT SYS005,000020,1,1,8,740
    // EXTENT SYS006,000100,1,2,1275,64
    // EXTENT SYS007,000006,8,3,50,636,6
3.  // EXEC PROG100
    /&
```

<u>Explanation for Disk File Labels, Multiple Drives</u>

This example has the same requirements as the <u>Single Drive</u> example except that the three volumes are mounted on three different drives.

<u>Statement 1</u>: To make this example realistic the assigned statements for two more disk drives are added. The drives involved are in no way restricted. The SYSnnn and the actual device address need not be consecutive. However, SYSnnn must correspond to the SYSnnn represented in the EXTENT statements.

<u>Statement 2</u>: All label statements submitted are identical to the <u>Single drive</u> example except for SYSnnn in the EXTENT statements.

<u>Statement 3</u>: Logical IOCS opens each extent in the same way as described in <u>Single Drive</u> example except that processing does not stop for removal and mounting of packs, because enough devices are on-line to contain the file. A combination of this and the <u>Single drive</u> example could be used to reduce handling time without excessively increasing the total drive requirements.

Direct Access File Labels Example

```
    // JOB DALABEL
    *  LABELS FOR A MULTI-VOLUME DIRECT ACCESS FILE
    // ASSGN SYS005,X'191'
    // ASSGN SYS006,X'193'
    // ASSGN SYS007,X'192'
    // DLBL FILENAME,'ALPHAMERIC FILE NAME',99/365,DA
1.  // EXTENT SYS005,000065,1,0,1320,190
    // EXTENT SYS005,000065,1,1,80,740
    // EXTENT SYS006,000025,1,2,50,906
    // EXTENT SYS007,000002,1,3,1275,64
2.  // EXEC PROG101
    /&
```

Explanation for Direct Access Example

This example shows the label statements as they apply to a direct access file.

Statement 1: Only one DLBL statement is required. These statements follow the same pattern as for sequential files, except that the DA specification is required to identify the file as direct access. One EXTENT statement is submitted for each of the extents, and the statements are placed in the job stream in consecutive ascending SYSnnn sequence. No skips in the SYSnnn numbers are allowed, although the sequence may begin with any SYSnnn. Only type 1 extents (data area, no split cylinders) are permitted, although up to 16 extents per volume are acceptable. All label storage records are written in the temporary background area, because USRLABEL mode is assumed in this example.

Statement 2: PROG101 opens all extents in all volumes. When all of the volumes have been opened, the file is ready for processing. Consequently, all volumes must be on-line and ready at the same time.

Index Sequential File Labels, Load Create Example

```
    // JOB INDEX SEQUENTIAL TO LOAD A FILE
    *  LABELS FOR A MULTI-VOLUME INDEXED SEQUENTIAL FILE
    *  MASTER INDEX, CYLINDER INDEX ON SAME VOLUME AS DATA RECORDS
    *  INDEPENDENT OVERFLOW AREA, LOAD CREATE
    // ASSGN SYS005,X'190'
    // ASSGN SYS006,X'191'
    // ASSGN SYS012,X'192'
    // ASSGN SYS019,X'193'
1.  // DLBL LOAD,'INDEX SEQUENTIAL EXAMPLE FILE',70/001,ISC
    // EXTENT SYS005,111111,4,0,1520,2  MASTER INDEX EXTENT
    // EXTENT SYS005,111111,4,1,1522,8  CYLINDER INDEX EXTENT
    // EXTENT SYS005,111111,1,2,1530,470 FIRST PRIME DATA EXTENT
    // EXTENT SYS012,123456,1,3,0010,1990  SECOND PRIME DATA EXTENT
    // EXTENT SYS019,123457,1,4,0010,80  THIRD AND FINAL PRIME DATA EXTENT
    // EXTENT SYS006,123458,2,5,1900,20  INDEPENDENT OVERFLOW EXTENT
2.  // EXEC LOADIS
    /&
```

Explanation for Index Sequential File Labels, Load Create

In this example LOADIS contains routines to create an indexed sequential file that is to be located on multiple volumes with an independent overflow area and a master index. The job stream reflects the label statements required for this operation.

Statement 1: The DLBL statement fcllows the usual pattern except that ISC defines the file operation as indexed sequential load create. The following points are relative to the EXTENT statements:

- The master index and the cylinder index use the option of being on the same volume as data records. However, they cannot occupy the same cylinder as data records nor extend into another volume. They can occupy successive cylinders. The extent type is 4, and the sequence numbers are 0 for the master index, and 1 for the cylinder index.

  Once the master index location is selected, the cylinder index must follow it immediately.

- The prime data area follows the rules to provide a single continuous extent, although three EXTENT statements are required. These rules are:

  a. Extent type 1.

  b. Utilization of entire cylinders (tracks 0-9).

  c. When multiple volumes are needed, location through cylinder 199 of a volume and continuation with cylinder 1 of the next volume.

- The independent overflow area is identified by extent type 2. In this example, it is located on the same volume as the data records but, like the master and cylinder indices, it can be on a separate volume. It can occupy successive cylinders but cannot extend into another volume. The EXTENT statement could be placed in the job stream ahead of the prime data EXTENT statements. This would require a change in the sequence number.

- The EXTENT statements are submitted in extent number sequence.

Index Sequential File Labels, Other than Load Create Example

```
    // JOB ISLABEL
    *  LABELS FOR A MULTI-VOLUME INDEXED SEQUENTIAL FILE
    *  MASTER AND CYLINDER INDEX ON SEPARATE VOLUME FROM DATA RECORDS
    *  INDEPENDENT OVERFLOW AREA, NOT LOAD CREATE
    // ASSGN SYS005,X'190'
    // ASSGN SYS006,X'191'
    // ASSGN SYS012,X'192'
    // ASSGN SYS019,X'193'
1.  // DLBL DISK,'INDEX SEQUENTIAL EXAMPLE FILE',70/001,ISE
    // EXTENT SYS005,111111,4,0,1520,2  MASTER INDEX EXTENT
    // EXTENT SYS005,111111,4,1,1522,8  CYLINDER INDEX EXTENT
    // EXTENT SYS005,111111,1,2,1530,470 FIRST PRIME DATA EXTENT
    // EXTENT SYS012,123456,1,3,0010,1990  SECOND PRIME DATA EXTENT
    // EXTENT SYS019,123457,1,4,0010,80  THIRD AND FINAL PRIME DATA EXTENT
    // EXTENT SYS006,123458,2,5,1900,20  INDEPENDENT OVERFLOW EXTENT
    // EXEC ADDRTR
2.  /&
```

Explanation for Index Sequential File Labels, other than Load Create

This example shows the label statements required to access the index sequential file created in the Load Create example. It has both a master index and an independent overflow area and requires multiple volumes for the data records.

Statement 1: The DLBL statement is the same as for any DASD file except for the ISE parameter, which indicates the file type as indexed sequential other than load create. The EXTENT statements follow the rules outlined in the Load Create example.

Statement 2: All volumes must be on-line at the same time. The logical IOCS of ADDRTR opens all volumes and extents before making the file available for processing.

Catalog UCS Buffer Images and Load Buffer Example

```
     // JOB UCSLOAD
     *  ASSEMBLE AND CATALOG TO CORE IMAGE LIBRARY PHASES FOR HN AND RN
     *  ARRANGEMENTS FOR USE WITH UNIVERSAL CHARACTER SET BUFFER LOAD
     // OPTION CATAL,NOSYM,NOXREF
1.      ACTION CANCEL
        PHASE UCSHN,*,NOAUTO
     // EXEC ASSEMBLY
              START 0
              DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
              DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
2             DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
              DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
              DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
              DC     CL43'HN ARRANGEMENT TYPE IF EQUAL SIGN ='
              DC     40C' '
              END
     /*
3.   // EXEC LNKEDT
        ACTION CANCEL
        PHASE UCSRN,*,NOAUTO
4.   // EXEC ASSEMBLY
              START 0
              DC        C'1234567890XY/STUVW''@$*,=JKLMNOPQR-A(ABCDEFGHI+.)'
              DC        C'1234567890XY/STUVW%@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
              DC        C'1234567890XY/STUVW#@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
              DC        C'1234567890XY/STUVW<@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
              DC        C'1234567890XY/STUVW&&@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
              DC        CL40'RN ARRANGEMENT TYPE IF POUND SIGN #'
              DC     40C' '
              END
     /*
     // EXEC LNKEDT       R
     /&
5.   // PAUSE MOUNT HN TRAIN
        UCS SYSLST,UCSHN,FOLD,BLOCK
                      RN
```

Explanation for UCS Buffer Load Example

This example shows the assembly, linkage edit and catalog of UCS buffer phases to the core image library. The buffer is then loaded.

Statement 1: The usual rules for setting up a catalog operating using the output of a language translator are followed. See OPTION and PHASE statements. ACTION CANCEL is used, because no cataloging should occur if any statement errors exist.

Statement 2: The Assembler prepares a relocatable module for the printer image data that follows on SYSIPT. Output is written on SYSLNK.

> Note: The use of two ampersands and two apostrophes to cause acceptance of those characters in the string.

The characters are submitted in the same sequence in which they occur on the train/chain. The standard arrangements are given in the manual IBM 2821 Control Unit GA24-3312). An 80-character message follows the 240-character printer image. The /* signifies the end of the data.

Statement 3: The EXEC LNKEDT statement link edits and catalogs the phase to the core image library.

Statement 4: Assuming no errors in the previous assembly, OPTION CATAL is still in effect. The linkage edit control statements that pertain to the next buffer arrangement are read, and the cycle is repeated.

Statement 5: Now that the buffer images are in the core image library, they can be loaded by the UCS job control command.

PHASE statement stops the system so that the proper chain/train can be mounted. In this example the HN chain/train arrangement represented by UCSHN is loaded into the UCS buffer for the printer represented by SYSLST, which is X'00E' in the assumed configuration. Of course, the printer must be equipped with the UCS feature.

The fold option is specified so that any lowercase letters will print as uppercase letters. This is necessary if lowercase codes are expected, because the HN arrangement has only uppercase letters. Otherwise, these lowercase letters can find no matching code in the buffer and cannot print. The BLOCK parameter permits any code not represented in the UCS buffer to print as a blank without causing a data-check stop. If BLOCK is not specified, any unmatched code causes a data check. Remember that the blocking of data checks is also controlled by the programmer through logical IOCS. Hence, the effect of this parameter may be transient, depending on the processing programs coming up in the job stream. A verification message will print, since NULMSG is not specified.

Appendix D: Linkage Editor Summary

Linkage Editing and Execute Example

```
    // JOB LINKEXEC
    *  LINK EDIT AND EXECUTE, SINGLE PHASE, SINGLE OBJECT MODULE
    *  RELOCATABLE MODULE NOT CATALOGED, BACKGROUND PROGRAM
    *  NONSEQUENTIAL DASD & LABELED TAPE FILES TO BE PROCESSED
1.  // ASSGN SYSLNK,X'190'
2.  // OPTION LINK
3.     PHASE PROGA,*
       INCLUDE
4.     (Relocatable object deck)
    /*
5.  // LBLTYP NSD(2)
6.  // EXEC LNKEDT
7.     Any job statements required for executicn as ASSGN or lakel statements.
8.  // EXEC
       Data input as required.
    /*
    /&
    *  1.  TO CATALOG AND EXECUTE, CHANGE STATEMENT 2 TO // OPTION CATAL.
    *  2.  TO CATALOG ONLY, CHANGE STATEMENT 2 TO // OPTION CATAL AND
    *      REMOVE ALL STATEMENTS FOLLOWING LNKEDT EXCEPT /&
    *  3.  TO USE MODULE FROM RELOCATABLE LIBRARY, CHANGE STATEMENT 3
    *      TO INCLUDE MODULES AND REMOVE ALL STATEMENTS UP TO // LBLTYP.
```

Explanation for Linkage Edit and Execute

This example illustrates the basic concept of linkage editing and executing by using a single phase that is constructed from a single relocatable object deck contained in punched cards. The program is executed in the background partition. Labeled tape and nonsequential DASD files are to be processed when the phase is executed. No more than two extents are used by any DASD file.

Statement 1: No assignments are necessary, because the system units required for linkage editing are in the assumed configuration. However, an ASSGN for SYSLNK is included to illustrate its position relative to the OPTION statement in case assignment is required.

Statement 2: The OPTION LINK statement sets switches to indicate that a linkage editor operation is to be performed. If SYSLNK has not been assigned, the statement is ignored. Linkage editor control statements are not accepted unless the OPTION statement is processed. Only linkage editing is performed; cataloging to the core image library does not occur.

Statement 3: The PHASE statement is copied cn SYSLNK, because position 1 is blank and the LINK switch is on. The operands are not examined until SYSLNK becomes input to the linkage editor program.

When the PHASE statement is processed by the linkage editor, only one phase is constructed, because only one PHASE statement is submitted for the entire LNKEDT. The name of this phase is PROGA, as specified in the first operand. The second operand indicates the origin point for the phase. Because an * has been used, the phase tegins in the next main storage location available, with forced doubleword alignment. Because this is the first and cnly phase, it is located at the end of the supervisor plus length of the lakel save area (reserved by LBLTYP) plus length of any area assigned to the COMMON pool (as designated by a CM entry in the relocatable rodule).

A relocation factor, either plus or minus, is used with the *, such as *+1024.

This causes the origin point of the phase to be set relative to the * by the amount of the relocation term. This term can be expressed as:

```
X'hhhhhh' -- 1 to 6 hexadecimal digits
dddddddd  -- 1 to 8 decimal digits
nK        -- where K = 1024
```

*+1024 uses the second format and adds 1024 bytes to the origin location. +1K or +X'400' gives the same result as +1024.

Statement 4: The INCLUDE statement has no operands, so the system reads the records from SYSIPT and writes them on SYSLNK until SYSIPT has an end-of-data (/*) record. The data on SYSIPT is expected to be the object module in card image format that is used in this linkage editor operation. If the output of the language translator (SYSPCH) is placed on tape or disk instead of cards, it cannot be used directly as SYSIPT in a linkage editor operation because the records contain a stacker select code in position 1. SYSPCH must be converted to an 80-position card image record.

Statement 5: The LBLTYP statement causes a computation of the number of bytes that are required for label storage data in the program to be linkage edited. In this example, 124 bytes are reserved (84 + [2x20]). The calculation is saved by job control and passed on first to the linkage editor and later to LIOCS.

Statement 6: The EXEC LNKEDT writes an ENTRY statement with no operand on SYSLNK and causes the system loader to bring in the linkage editor program.

Using the data just placed on SYSLNK as input, the linkage editor develops executable code. The output is placed in the next available space of the core image library (immediately after the last cataloged phase). This is true regardless of whether the program is cataloged or not. Cataloging causes the updating of the directory to reflect a new ending point for the library. If cataloging does not occur, the next program that is linkage edited overlays it. For this reason, a linkage edited program that is not cataloged is said to be placed in the temporary area of the core image library. Also, a program that is linkage edited without cataloging must be linkage edited whenever it is used. No ACTION options are specified. Therefore, in resolving the external references, the system makes use of the AUTOLINK feature. Error diagnostics and a main storage map are written on SYSLST, because SYSLST is assigned.

Statement 7: Because the program is not cataloged, it must be executed immediately. Any pertinent job control statements are entered at this point.

Statement 8: An EXEC statement with no operand indicates that the phase to be executed was just linkage edited. Therefore, no search of the core image directory is required, and the system loader brings the program into main storage from the temporary area and transfers control to its entry point. In this example, the entry point is either the address specified in the END record, or the phase load address if the END address is omitted, because the automatic ENTRY statement is in effect.

This example can be modified to illustrate the following:

1. Catalog and execute. To cause this phase to be cataloged rather than merely linkage edited, change the OPTION (statement 2) from LINK to CATAL. The name in the PHASE statement may be added to the EXEC statement (// EXEC PROGA).

2. Catalog only. To catalog only, change the OPTION (statement 2) from LINK to CATAL and remove all data following the EXEC LNKEDT (statement 6) up to the /& statement.

3. Object module in relocatable library. The name to catalog the object module into the relocatable library must be added to the INCLUDE statement. If the name is RELOCA, the statement becomes INCLUDE RELOCA. The relocatable object deck and /* statement are removed. This form of the INCLUDE statement is written on SYSLNK when it is read by job control. The linkage editor retrieves the object module when it encounters the INCLUDE statement because it uses SYSLNK for input.

Catalog to Core Image Library Example

```
    // JOB CATALCIL
    *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
    *  SINGLE PHASE, MULTIPLE OBJECT MODULES, FOREGROUND PROGRAM
    *  MIXTURE OF CATALOGED AND UNCATALOGED RELOCATABLE MODULES
    *  LABELED TAPE FILES AND SEQUENTIAL DASD FILES TO BE PROCESSED
1.  // ASSGN SYSLNK,X'190'
2.  // OPTION CATAL
3.     PHASE PROGB,F+32768
4.     INCLUDE
       (Relocatable object deck)
    /*
       INCLUDE SUBRX
       INCLUDE SUBRY
       INCLUDE
       (Relocatable object deck)
    /*
5.  // LBLTYP TAPE
6.  // EXEC LNKEDT
7.  /&
```

Explanation for Catalog to Core Image Library

This example illustrates the cataloging of a single phase composed of multiple relocatable object modules. These modules are located in the input stream and in the relocatable library. Labeled tape files and sequential DASD files are processed when the phase is executed. The program is executed in a foreground partition. Assume that the foreground partition begins at location 32768.

Statement 1: The SYSLNK assignment indicates the relationship to the OPTION statement, although it is not required because of the assumed configuration.

Statement 2: The OPTION CATAL statement sets the LINK switches, as well as a CATAL switch. If SYSLNK is not assigned, the statement is ignored. The linkage editor control statements are not accepted unless the OPTION statement is processed. Linkage editing and cataloging to the core image library will occur.

Statement 3: Only one PHASE is constructed. It is cataloged to the core image library and retrieved by the name PROGB. Because this is to be a foreground phase, F plus the location address in the foreground partition must be specified.

A program may be linkage edited to any address that falls within a foreground partition. The load address does not have to coincide with the partition address. However, the program must be of such a size that it can reside in the available core defined from the load address to the end of the partition.

The address may be expressed in one of three forms:

```
X'hhhhhh' -- A hexadecimal number of 4
             to 6 digits
dddddddd  -- A decimal number of 5 to
             8 digits
nnnnK     -- Where K = 1024 and n is 2
             to 4 digits
```

+X'8000', +32768, and +32K are equivalent. The actual origin point of the phase is adjusted upward from the address specification to allow for the partition save area, and the label information (LBLTYP statement reservation).

Statement 4: Four modules make up this phase. The first and last are not cataloged in the relocatable library; therefore the object decks must be on SYSIPT, and each must be followed by the end-of-data record (/*). SUBRX and SUBRY are cataloged previously to the relocatable library by those names. Job control puts the uncataloged modules on SYSLNK in place of their INCLUDE statements. Job control copies the INCLUDE statements for the cataloged modules.

Statement 5: The LBLTYP statement has the operand TAPE, rather than NSD because labeled tapes and sequential DASD files are processed when the phase is executed. Eighty bytes are reserved ahead of the actual phase for label information. LBLTYP NSD is also satisfactory because it generates a minimum of 104 bytes and tapes require only 80.

Statement 6: The EXEC LNKEDT statement causes the system loader to bring in the linkage editor program. SYSLNK now becomes input to the linkage editor. It contains the following:

```
PHASE PROGB,F+32768
First uncataloged relocatable deck
INCLUDE SUBRX
INCLUDE SUBRY
Second uncataloged relocatable deck
ENTRY
```

The modules are linkage edited so that they occupy contiguous areas in main storage in the sequence in which they appear in the input stream. When the linkage editing is completed, cataloging to the core image library occurs because of the CATAL option. The core image directory is checked to make sure the new phase entries fit. If not, the job is canceled. The directory is scanned for any match to a phase being cataloged. A match is deleted from the directory. The system directory is updated to reflect the changes. Job control is brought into main storage.

Statement 7: Because CATAL was specified, a special routine is executed when the /& control statement is read by job control. This routine updates the transient, library-routine, and foreground-program directories. A system status report is printed to reflect the usage and available space in each of the libraries and directories. These operations do not occur in a LINK situation. The /& resets the CATAL option, that is, it turns off the LINK and CATAL switches.

The example can be modified to illustrate a catalog-and-execute operation by inserting the following data between the EXEC LNKEDT and /& statements:

1. Any job control statements required for execution or PROGB

2. A // EXEC PROGB or // EXEC statement

3. Any card reader input for PROGB

Note that the actual update of the directories and the system status report are delayed until completion of the execution of PROGB, when /& is read. From a system design standpoint this is not desirable because of possible operational problems. Making the execution of PROGB a separate job avoids any difficulties.

Compile and Execute Example

```
    // JOB COMPEXEC
    *  COMPILE OR ASSEMBLE, LINK EDIT AND EXECUTE
    *  SINGLE PHASE, MULTIPLE OBJECT MODULES, BACKGROUND PROGRAM
    *  SEQUENTIAL DASD FILES TO BE PROCESSED
    *  INPUT TO LINKAGE EDITOR FROM LANGUAGE TRANSLATOR, RELOCATABLE LIBRARY AND SYSIPT.
1.  // ASSGN SYSLNK,X'190'
2.  // OPTION LINK
3.     PHASE PROGA,S
4.  // EXEC COBOL
       (COBOL source statements)
    /*
       INCLUDE SUBRX
5.     INCLUDE
       (Relocatable object module)
    /*
6.     ENTRY BEGIN1
    // EXEC LNKEDT
7.     (Any job control statements required for PROGA execution)
    // EXEC
       (Any input data required for PROGA execution)
    /*
    /&
```

Explanation for Compile and Execute

The language translators provide the option of placing their output on SYSLNK rather than SYSPCH. Because the linkage editor uses SYSLNK for input, a program can be assembled or compiled, then linkage edited and executed. This operation, known as assemble/compile and execute, is illustrated by this example.

All three sources of object module input to the linkage editor are used: SYSIPT, the relocatable library, and the output from a language translator. It is assumed that the phase is executed in the background partition, and that only sequential DASD files or unlabeled tape files are processed.

Statement 1: The SYSLNK assignment is given to illustrate the relationship to the OPTION statement, although it is not required because of the assumed configuration.

Statement 2: Because SYSLNK is assigned, the OPTION LINK statement sets the link indicator switches.

Statement 3: The PHASE statement must always precede the relocatable modules to which it applies; therefore, it is written on SYSLNK first for later use by the linkage editor. S is the origin point, that is, the phase originates with the first doubleword at the end of the supervisor plus length of the label save area (as defined by LBLTYP) plus length of the area assigned to the COMMON pool (if any). This gives the same effect as * gives for a single phase or the first phase. As with the *, the S may be used with a relocation factor, for example, S+1024. The factor must always be positive, because a negative factor could cause the origin point to overlay the supervisor.

Statement 4: The appropriate language translator is called (in this case, COBOL). The normal rules for compiling are followed; the source deck must be on the unit assigned to SYSIPT and the /* defines the end of the source data. Because the LINK switches are set, the output of the language translator is written on SYSLNK. Except for PL/I, FORTRAN (F) and the ASSEMBLER (F) and 14K variant, the DECK option is ignored when SYSLNK is used.

Statement 5: The INCLUDE SUBRX statement is written on SYSLNK. The linkage editor retrieves the named module from the relocatable library. Because the operand is blank, the next INCLUDE statement signifies that the relocatable module is on SYSIPT. The data on SYSIPT is copied on SYSLNK until the /* statement.

Statement 6: The ENTRY statement is written on SYSLNK as the last linkage control statement. The symbol BEGIN1 must be the name of a CSECT or a label definition defined in the first phase. The address of BEGIN1 becomes the transfer address for the first phase of the program. The ENTRY statement is used because the user wishes to provide a specific entry point rather than to use the point specified in the END record or the load address of the phase. The ENTRY statement affects the first, or only, phase.

Statement 7: No LBLTYP statement is required, because only sequential DASD files are to be processed. The rest of the statements follow the same pattern as discussed in the Linkage Editing and Execute example. The input from SYSLNK to the linkage editor is:

```
PHASE PROGA,S
Relocatable module produced by COBOL
  compilation
INCLUDE SUBRX
Relocatable module from SYSIPT
ENTRY BEGIN1
```

If an error is detected during compilation or assembly of a source program, the LINK option is suppressed. Under these circumstances the EXEC LNKEDT and EXEC statements are ignored in this example. This LINK option suppression should be kept in mind if a series of programs is to be compiled and cataloged as a single job. Failure of one job step would cause failure of all succeeding steps. Remember that an OPTION LINK cannot be given if OPTION CATAL is in effect, because message 1S1nD (STATEMENT OUT OF SEQUENCE) results. This is an error in instruction to the system because CATAL has functions that must be performed when the next /& statement is read. Therefore, the CATAL switch must remain on, and linkage editing only cannot be performed.

Catalog for Phase Overlay Example

```
    // JOB MULTPHAS
    *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
    *  MULTIPLE PHASES, MULTIPLE OBJECT MODULES, BACKGROUND PROGRAM
    *  NO LABELED TAPE OR NONSEQUENTIAL DASD FILES TO BE PROCESSED
    // ASSGN SYSLNK,X'190'
1.  // OPTION CATAL
2.     PHASE PHASEA,ROOT
       INCLUDE MOD1
3.     PHASE PHASEB,*
       INCLUDE MOD2
4.     PHASE PHASEC,PHASEB
       INCLUDE MOD3
5.  // EXEC LNKEDT
    /&
```

Explanation for Catalog for Phase Overlay

Sometimes it is not possible to bring an entire program into main storage because it requires more bytes of main storage than are available. To solve this problem the program can be broken into separate phases that can be brought into main storage as required, overlaying all or part of another phase if desired. The linkage editing of three cataloged relocatable modules into phases for overlay is illustrated by this example.

Statement 1: The OPTION CATAL sets the switches so that the phases can be linkage edited and cataloged into the core image library.

Statement 2: PHASEA is considered the ROOT phase, that is, it is always resident in main storage during program execution. The origin point is the first doubleword address after the supervisor plus length of the label save area (if any) plus length of the area assigned to the COMMON pool (if any). Only the first phase statement is permitted to specify ROOT.

Statement 3: The * in the PHASE card for PHASEB causes MOD2 to be linkage edited at the end of PHASEA.

Statement 4: Because PHASEB is specified as the load address of PHASEC, it is linkage edited into the same address as PHASEB. The symbol that designates the origin point may be a previously defined phase name as in this example, a previously defined control section, or a previously defined external label. A plus or minus relocation factor may be used (for example, PHASE2+100).

Statement 5: The EXEC LNKEDT causes all three phases to be linkage edited and cataloged. When the phases are executed, the ROOT phase normally is loaded by the system loader. The other phases would probably be brought into main storage by means of the FETCH or LOAD macro issued by the calling phase.

Submodular Structure Example

```
      // JOB SUBMOD
      *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
      *  MULTIPLE PHASES, ONE OBJECT MODULE (SUBMODULAR STRUCTURE)
      *  BACKGROUND PROGRAM, NO LABEL STORAGE RESERVATION
      // ASSGN SYSLNK,X'190'
      // OPTION CATAL
1.       PHASE PHASE1,ROOT
         INCLUDE ,(CSECT1,CSECT3)
2.       PHASE PHASE2,*
         INCLUDE ,(CSECT2,CSECT5)
3.       PHASE PHASE3,PHASE2
         INCLUDE ,(CSECT4,CSECT6)
4.       INCLUDE
         (Relocatable object deck)
      /*
      // EXEC LNKEDT
      /&
```

Explanation for Submodular Structure

Several relocatable modules are structured into several phases. In this example, a single object module is broken into several phases. The object module is composed of CSECT1-CSECT6. It is structured into three phases with overlay. The module is not cataloged in the relocatable library. Only the PHASE AND INCLUDE statements are discussed.

Statement 1: The INCLUDE statement tells the linkage editor to place CSECT1 and CSECT3 into PHASE1. The sequence in which the CSECTs are linkage edited is determined by the sequence in the input module rather than the sequence in the INCLUDE statement. (CSECT3,CSECT1) would give the same result as (CSECT1,CSECT3). The sequence can be controlled by issuing separate INCLUDE statements. For example, INCLUDE ,(CSECT3) followed by INCLUDE ,(CSECT1) causes CSECT3 to be linkage edited before CSECT1, regardless of the sequence in the object module.

Note that the first operand is missing in the INCLUDE statement, as indicated by the leading comma. This format of the INCLUDE statement searches the next succeeding object module on SYSLNK to locate the named CSECTs. See Statement 4.

PHASE1 is located at the end of the supervisor plus length of the label save area, and the COMMON area (if any).

Statement 2: The INCLUDE statement causes CSECT2 and CSECT5 to be used for PHASE2. This phase is located at the end of PHASE1.

Statement 3: PHASE3 is made up of CSECT4 and CSECT6 and overlays PHASE2 because its origin point is at the same address as PHASE2.

Statement 4: This INCLUDE statement with a blank operand is required to write the object module that follows in the card reader onto SYSLNK, to satisfy the INCLUDE statements with a blank first operand.

With the sequence of statements shown in the example, the PHASE and INCLUDE statements are read from SYSRDR. However, it is permissible to read PHASE and INCLUDE statements from SYSIPT. To do this, Statement 4 (INCLUDE blank) is placed ahead of Statement 1. The INCLUDE with the blank operand directs job control to read the following data (which includes the PHASE, INCLUDE, and then the object module) on

SYSLNK from SYSIPT to the /* statement. If SYSRDR and SYSIPT are separate devices, take care to place the PHASE and INCLUDE statements on the correct device.

PHASE and INCLUDE statements can also be in the relocatable library. If the object module is in the relocatable library under the name MOD1, the following changes are made:

1. Remove Statements 1 through 3, and add module name to Statement 4.

```
// JOB SUBMOD
// OPTION CATAL
   INCLUDE MOD1
// EXEC LNKEDT
```

2. When the relocatable module is cataloged to the library, precede it with the following statements:

```
PHASE PHASE1,*
INCLUDE MOD1,(CSECT1,CSECT3)
PHASE PHASE2,*
INCLUDE MOD1,(CSECT2,CSECT5)
PHASE PHASE3,PHASE2
INCLUDE MOD1,(CSECT4,CSECT6)
Relocatable object deck
```

This form of the INCLUDE statement causes the linkage editor to search the module that follows the last INCLUDE statement in the library for the required control sections.

Self-Relocating and Multiple Link Edits Example

```
    // JOB MULTCATL
    *  SEVERAL LINK EDITS AS A SINGLE JOB
    // OPTION CATAL
1.     PHASE PROG1,+0
       INCLUDE PROG1
    // EXEC LNKEDT
2.     PHASE PROG2,*
       INCLUDE
       (Relocatable object module)
    /*
    // EXEC LNKEDT
       PHASE PROG3,*
       INCLUDE PROG3
    // EXEC LNKEDT
    /&
    // JOB LINKGO
3.  // OPTION LINK
       PHASE PROG4,*
       INCLUDE
       (Relocatable object module)
    /*
    // EXEC LNKEDT
       (Any job control statements required for PROG4 execution)
    // EXEC
       (Any input data required for PROG4)
    /*
       PHASE PROG5,*
       INCLUDE PROG5
    // EXEC LNKEDT
       (Any job control statements required for PROG5 execution)
    // EXEC
       (Any input data required for PROG5)
    /*
    /&
```

Explanation for Self-Relocating and MultipleLink Edits

The linkage editing requirements for a self-relocating program and the combining of several cataloging or link-and-execute job steps into a single job are illustrated in this example. Use discretion in deciding how many steps should be combined before a /&, because a failure in one step can cause successive steps to be bypassed unnecessarily.

Statement 1: The +0 displacement as the origin part for PROG1 designates this program as self-relocating. Once this program is cataloged to the core image library, it may be executed in any partition. It can be initiated into the background partition by job control, if the supervisor was generated with multiprogramming capabilities. A non-MPS supervisor will give message 0P77I (CANCELED DUE TO INVALID ADDRESS). However, the program can be loaded by using the LOAD macro in a calling phase.

Statement 2: PROG2 and PROG3 are linkage edited and cataloged as job steps within the job MULTCATL. Note that OPTION CATAL holds for these steps.

Statement 3: A new job is initiated because the succeeding job steps are linked and executed without cataloging. (OPTION LINK cannot be issued when OPTION CATAL is in effect.) Note that OPTION LINK need not be reissued before the next job step.

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.

Appendix E: Multiprogramming Summary

Figure 52 is a summary of the characteristics and system control and service facilities applicable to the available partitions in a multiprogramming environment.

Characteristic of Facility	Background (BG)	Either Foreground with BJF MPS=BJF	Either Foreground (SPI) MPS=YES
Program Initiation	Automatic	Automatic[1]	Operator
Job Control Unit	SYSRDR/IPT Card/Tape/Disk	SYSRDR/IPT Card/Tape/Disk	SYSLOG[2] or Card Reader
Storage Protection	Yes	Yes	Yes
Storage Protect Key	1	F1: 3 F2: 2	F1: 3 F2: 2
Location (Fixed)	Adjacent to System Supervisor	F1: Adjacent to Storage End F2: Adjacent to F1	F1: Adjacent to Storage End F2: Adjacent to F1
Size: (Variable)			
Minimum (Active)	10K Bytes	10K Bytes	2K Bytes
Minimum (Inactive)	10K Bytes	0 Bytes	0 Bytes
Maximum	avail. storage	510K Bytes	510K Bytes
Intervals	2K Bytes	2K Bytes	2K Bytes
Established by	SYSGEN	SYSGEN	SYSGEN
Alterable by	Operator	Operator	Operator
Priority	3 (lowest)	F1: 1(highest) F2: 2	F1: 1(highest) F2: 2
Use of System Functions:			
System Log	Yes	Yes[2]	Yes[2]
Job Logging	Yes	Yes	No
System Units (I/O)	Yes	Yes	Yes[3]
Programmer Units	Yes	Yes	Yes
Operator Inquiry	Yes	Yes	Yes
IOCS Macros	Yes	Yes	Yes
Fetch/Load	Yes	Yes	Yes
Program Check Exit	Yes	Yes	Yes
Interval Timer Exit[4]	Yes	Yes	Yes
Time of Day[5]	Yes	Yes	Yes
Dump (PDUMP)	Yes	Yes	Yes
Communication Region	Yes	Yes	No
Checkpoint/Restart	Yes	Yes	No
Types of Program Loading:			
Absolute	Yes	Yes	Yes
Self-Relocating	Yes	Yes	Yes

Figure 52. Multiprogramming Summary (Part 1 of 2)

Characteristic of Facility	Background (BG)	Either Foreground with BJF MPS=BJF	Either Foreground (SPI) MPS=YES
Linkage Editor[6] (LNKEDT)	Yes	Yes	No
Librarian			
MAINT[6]	Yes	Yes[8]	Yes[8]
CORGZ	Yes	No	No
CSERV[6]	Yes	Yes	Yes
RSERV[7]	Yes	Yes	Yes
SSERV[7]	Yes	Yes	Yes
DSERV[6]	Yes	Yes	Yes
SYSBUFLD[6]	Yes	Yes	Yes
Problem Determination:			
ESTV	Yes	Yes[7]	Yes[6]
EREP[7]	Yes	Yes	Yes
PDAID[6]	Yes	Yes	Yes
DUMPGEN[6]	Yes	Yes	Yes
LSERV[6]	Yes	Yes	Yes

Notes:

[1] Foreground is initiated into Batch Job mode by operator; thereafter program initiation is automatic from SYSRDR assigned to that partition.

[2] SYSLOG must be assigned a 1052 cr a 3210 or 3215.

[3] Assignments limited to card readers, card punches, and printers.

[4] Only one partition at a time selectable by cperator.

[5] Assumes that the Timer Feature is available.

[6] Self-relocating

[7] Non-self-relocating, may be linkage edited to a core image library (either system or private) for execution in background and/or either foreground.

[8] Only for private core image library.

Figure 52. Multiprogramming Summary (Part 2 of 2)

Glossary

For a more complete list of data processing terms, refer to IBM Data Processing Techniques, A Data Processing Glossary, GC20-1699.

active program: Any program that is loaded and ready to be executed.

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.

background program: In multiprogramming the background program is the program with lowest priority. Background programs execute from a stacked job input.

Basic Telecommunications Access Method (BTAM): A basic access method that permits a READ/WRITE communication with remote devices.

batched job: Programs that execute from a stacked job input. Batched jobs run under the control of job control.

block:

1. To group records physically for the purpose of conserving storage space or increasing the efficiency of access or processing.

2. A physical record on tape or DASD.

book: A group of source statements written in the Assembler language.

CCH (channel check handler): A feature that assesses System/370 channel errors to determine if the system can continue operations.

channel inboard error: An error that occurs between one I/O device and the central processing unit.

catalog: To enter a phase, module, or book into one of the DOS libraries.

channel program: One or more Channel Command Words (CCWs) that control(s) a specific sequence of channel operations. Execution of the specific sequence is initiated by a single start I/O instruction.

channel scheduler: The part of the supervisor that controls all input/output operations.

checkpoint record: Records that contain the status of the job and the system at the time the records are written by the checkpoint routine. These records provide the necessary information for restarting a job without having to return to the beginning of the job.

checkpoint/restart: A means of restarting execution of a program at some point other than the beginning. When a checkpoint macro instruction is issued in a problem program, checkpoint records are created. These records contain the status of the program and the machine. When it is necessary to restart a program at a point other than the beginning, the restart procedure uses the checkpoint records to reinitialize the system.

checkpoint routine: A routine that records information for a checkpoint.

command control block: A sixteen-byte field required for each channel program executed by physical IOCS. This field is used for communication between physical IOCS and the problem program.

communication region: An area of the supervisor set aside for interprogram and intraprogram communication. It contains information useful to both the supervisor and the problem program.

control program: A group of programs that provides functions such as the handling of input/output operations, error detection and recovery, program loading, and communication between the program and the operator. IPL, supervisor, and job control make up the control program in the Disk and Tape Operating Systems.

control section: The smallest separately relocatable unit of a program; that portion of text specified by the programmer to be an entity, all elements of which are to be loaded into contiguous main storage locations.

core image library: An area of the resident system device used to store programs that have been processed by the Linkage Editor. Each program is in a form identical to that which it must have to be executable in main storage.

core storage: See main storage.

core wrap mode: The method of operation that records the events of a trace in main storage.

data conversion: The process of changing data from one form of representaticn to another.

data file: A collection of related data records organized in a specific manner. For example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc) or an inventory file (one record for each inventory item, showing the cost, selling price, number in stock, etc).

data set security: A feature that provides protection for disk files. A data secured file cannot be accidentally accessed by a problem program.

device independence: The capability of a program to process the same type of data on different device types (punched card devices/printers, tape, or disk).

Disk Operating System: A disk resident system that provides operating system capabilities for 16K and larger System/360 and System/370 systems.

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).

DTF (define the file) macro instruction: A macro instruction that describes the characteristics of a logical input/output file, indicates the type of processing for the file, and specifies the main storage areas and routines to process the file. Tc do this, use the appropriate entries in the keyword operands associated with the DTF macro instruction.

DUMP: Displaying the contents of main storage.

EREP (Environmental Recording, Editing, and Printing): A program that processes data that has been stored in the system recorder file.

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 vclume 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.

extent: The physical locations on Input/Output devices occupied by or reserved for a particular file.

external reference: A reference to a symbol used in another module. The external references are resolved when the respective modules are combined at linkage edit time.

fetch:

1. To bring a program phase into main storage from the core image library for immediate execution.

2. The routine that retrieves requested phases and loads them into main storage (see system loader).

3. The name of a macro instruction (FETCH) used to transfer control to the System Loader.

file: See data file.

fixed length record: A record having the same length as all other records with which it is logically or physically associated.

foreground initiation: A set of system routines to process operator commands for initiation of a foreground program.

foreground initiator: See single program initiator (SPI).

foreground program: In multiprogramming, foreground programs are the highest priority programs. Foreground programs may be executed from a job stack or in an SPI environment.

Initial Program Loading (IPL): The initialization procedure that causes the DOS System to commence operation.

input job stream: A sequence of job control statements entering the system, which may also include input data.

Input/Output Control System (IOCS): A group of macro instruction routines provided by IBM for handling the transfer of data between main storage and external storage device. IOCS consists of two parts: physical IOCS and logical IOCS.

interruption: A break in the normal sequence of instruction execution. It causes an automatic transfer to a preset storage location where appropriate action is taken.

I/O area: An area (portion) of main storage into which data is read or from which data is written. In Operating System publications, the term buffer is often used in place of I/O area. I/O means input/output.

I/O (input/output) error logging: The process of recording OBR and SDR records on the system recorder file.

IPL loader: A program that reads the supervisor into main storage and then transfers control to the supervisor.

job accounting interface: A function that accumulates accounting information for each job step to: charge usage of the system, help plan new applications, and help supervise system operation more efficiently.

job control: A program that is called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to certain symbolic names, set switches for program use, log (or print) job control statements, and fetch the first program phase of each job step.

job statement (JOB): The control statement in the input stream that identifies the beginning of a series of job control statements for a single job, i.e., a single accounting unit of processing.

job step: The execution of a single processing program.

K: 1024

language translators: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent machine language instructions. For example, Assembler, COBOL, etc are language translators.

librarian: The set of programs that maintains, services, and organizes the system libraries.

library: An organized collection of programs, source statements, or object modules maintained on the system resident device.

linkage editor: A system service program that edits the output of language translators and produces executable program phases. It relocates programs or program sections and links together separately assembled (or compiled) sections.

load: To fetch, i.e., to read a phase into main storage returning control to the calling phase.

logical IOCS: A comprehensive set of macro instruction routines provided to handle creation, retrieval, and maintenance of data files.

main storage: All addressable storage from which instructions can be executed or from which data can be loaded directly into registers.

main task: The main program within a partition in a multiprogramming environment.

MCAR (machine check analysis and recording): A feature that records System/370 machine check interrupt error information on the system recorder file and then attempts to recover from the interrupt.

MCI (machine check interrupt): The interrupt that occurs if the central processing unit fails to operate.

MCRR (machine check recording and recovery): The recording of pertinent data on the system recorder file after either a machine check interrupt or a channel inboard error occurred on System/360 Model 30, Model 40, or Model 50.

module (programming): The input to or output from a single execution of a language translator or the Linkage Editor; a separate program unit that can be combined with other units.

MPS: Multiprogramming System.

multifile volume: A unit of recording media, such as a tape reel or disk pack, that contains more than one data file.

multiprogramming system: A system that controls more than one program simultaneously by interleaving their execution.

multitask operation: Multiprogramming; called multitask operation to express not only concurrent execution of one or more programs in a partition, but also of a single reenterable program used by many tasks.

multivolume file: A data file that, due to its size, requires more than one unit of recording media (such as a tape reel or disk pack) to contain the entire file.

nonstandard labels: Labels that do not conform to the System/360 standard label specifications. They can be any length, need not have a specified identification, and do not have a fixed format.

OBR (outboard recorder): A feature that records pertinent data on the system recorder file when an unrecoverable I/O error occurs.

operating system: A collection of programs that enables a data processing system to supervise its own operations, automatically calling in programs, routines, languages, and data as needed for continuous throughput of a series of jobs.

overlap: To do something at the same time that something else is being done; for example, to perform input/output operations while instructions are being executed by the central processing unit.

phase: The smallest complete unit that can be referenced in the core image library. Each overlay of a program or (if the program contains no overlays) the program itself is a single complete phase.

physical IOCS: Macro instructions and supervisor routines (Channel Scheduler) that schedule and supervise the execution of channel programs. Physical IOCS controls the actual transfer of records between the external storage medium and main storage, and provides I/O device error recovery.

private library: A core image, relocatable, or source statement library that is separate and distinct from the system library.

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.

problem program:

1. The user's object program. It can be produced by any of the language translators. It consists of instructions and data necessary to solve the user's problem.

2. A general term for any routine that is executed in the data processing system's problem state; that is, any routine that does not contain privileged operations. (Contrasted with Supervisor.)

processing program: A general term for any program that is both loaded and supervised by the control program. Specifically, a collection of certain IBM supplied programs: the language translators, Linkage Editor, Librarian, Autotest, Sort/Merge and Utilities. All user written programs are processing programs. The term processing programs is in contrast to the term contrcl program.

record: A general term for any unit of data that is distinct from all others when considered in a particular context.

reenterable: The attribute of a set of code that allows the same copy of the set of code to be used concurrently by two or more tasks.

relocatable: A module or control section whose address constants can be modified to compensate for a change in origin.

relocatable library module: A module consisting of one or more complete control sections cataloged as a single entry in the relocatable library.

restart: See checkpoint/restart.

RMS (recovery management support): A feature for System/370 that consists of the MCAR (machine check analysis and recording) and CCH (channel check handler) functions. RMS gathers information about System/370 hardware reliability and attempts certain error recovery operations. RMS is a part cf the entire reliability, availability, and serviceability support for System/370.

SDR (statistical data recorder): A feature that records the cumulative error status of an I/O device on the system recorder file.

self-relocating: A programmed routine that is loaded at any doubleword boundary and can adjust its address values so as to be executed at that location.

self-relocating program: A program that is able to run in any area of storage by having an initialization routine to modify all address constants at object time.

service programs: Any of the class of standard routines that assist in the use of a data processing system and successful execution of problem programs.

single program initiator (SPI): A program that is called into storage to perform job control type functions for foreground programs nct executing in batch job mode.

source module: A set of source statements in the symbolic language of a language translator that constitutes the entire input to a single execution of the language translator.

source statement: Statements written by a programmer in symbolic terms related to a language translator.

source statement library: A collection of books (such as macro definitions) cataloged onto the system by the Librarian.

stacked job processing: A technique that permits multiple jobs to be grouped (stacked) for presentation to the system. This allows the system to automatically process each job in sequence.

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.

submodular phase: A phase made up of selected control sections from one or more modules as compared with a normal phase that is made up of all control sections from one or more modules.

subtask: A task in which control is initiated by a main task by means of a macro instruction that attaches it.

supervisor: A component of the control program. It consists of routines to control the functions of program loading, machine interruptions, external interruptions, operator communications and physical IOCS requests and interruptions. The supervisor alone operates in the privileged (supervisor) state. It coexists in main storage with problem programs.

symbolic I/O assignment: A means by which problem programs can refer to an I/O device by a symbolic name. Before a program is executed, job control can be used to assign a specific I/O device to that symbolic name.

system generation: The process of tailoring the IBM supplied operating system to user requirements.

system loader: One of the supervisor routines. It is used to retrieve program phases from the core image library and load them into main storage.

system recorder file: The data file that is used to record hardware reliability data.

system residence: The external storage space allocated for storing the basic operating system. It refers to an on-line tape reel or disk pack that contains the necessary programs required for executing a job on the data processing system.

system service programs: Programs that perform portions of the functions of generating the initial basic operating system, generating specialized systems, creating and maintaining the library sections, and loading and editing programs onto the resident device. These programs are: linkage editor and librarian.

task selection: The supervisor mechanism for determining which program should gain control of CPU processing.

telecommunications: A general term expressing data transmission between remote locations.

teleprocessing: A term associated with IBM telecommunication systems expressing data transmission between a computer and remote devices.

throughput: A measure of system efficiency; the rate at which a series of jobs can be handled by a data processing system.

transient area: This is a main storage area (within the supervisor area) used for temporary storage of transient routines.

transient routines: These self-relocating routines are permanently stored on the system residence device and loaded (by the supervisor) into the transient area when needed for execution.

variable length record: A record having a length independent of the length of other records with which it is logically or physically associated. (Contrasted with Fixed Length Record). It contains fields specifying physical and logical record lengths.

volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism. For example, a reel of magnetic tape on a 2400-series magnetic tape drive or a disk pack on an IBM 2311 Disk Storage Drive.

wait condition: As applied to tasks, the condition of a task that it is dependent on an event or events in order to enter the ready condition.

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, DOS 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.

GC24-5036-7

DOS System Control and Service (S360-36) Printed in U.S.A. GC24-5036-7

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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