File Number S360-36
Form C24-5036-1
Page modified 3/17/67
by TNL N24-5218

DOS

IBM

Systems Reference Library

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

This reference publication describes the IBM System/360 Disk Operating System. The system is a set of control programs and processing programs provided for IBM System/360. Using IBM 2311 Disk Storage for on-line program residence, the IBM System/360 Disk Operating System provides stacked-job processing capability, multiprogramming and telecommunications capability, controls all input/output, and provides for continuous operation of all programs run in its environment. Detailed information is given in this publication on these major topics.

1. Operation with the System Control Program:
   a. System Organization
   b. Supervisor Functions
   c. Job Control Program.

2. Using the System Service Programs:
   a. Linkage Editor
   b. Librarian.

The prerequisite for a thorough understanding of this publication is a basic knowledge of System/360 machine concepts. The publications most closely related to this one are:

1. IBM System/360 Principles of Operation, Form A22-6821.

2. IBM System/360 Disk Operating System, Data Management Concepts, Form C24-3427.

3. IBM System/360 Disk Operating System Supervisor and Input/Output Macros, Form C24-5037.

4. IBM System/360 Disk and Tape Operating Systems, Assembler Specifications, Form C24-3414.

(Text deleted)

PREFACE

The first part of this publication, under the headings Supervisor, Job Control, and Initial Program Loader (IPL), describes the control program for the Disk Operating System . This part is of interest to anyone using the system, including system analysts, programmers, and machine operators. The functions of the Supervisor are discussed, and the detailed Job Control statement formats are given. Note, however, that the macro instructions used to communicate with the Supervisor are discussed fully in the Supervisor and I/O Macros publication as listed on the front cover of this publication.

The second part of this publication, under the headings Linkage Editor and Librarian, is of particular interest to the persons responsible for maintaining the resident system. This part describes the Linkage Editor and Librarian programs fully.

The telecommunications publications most closely related to this manual are:

1. IBM System/360 Disk Operating System, Basic Telecommunications Access Method, Form C30-5001.

2. IBM System/360 Disk Operating System, QTAM Message Control Program, Form C30-5002.

3. IBM System/360 Disk Operating System, Queued Telecommunication Access Method, Message Processing Program Services, Form C30-5003.

For titles and abstracts of other associated publications, see the IBM System/360 Bibliography, Form A22-6822.

Second Edition, November 1966

This edition, C24-5036-1, is a major revision of, and obsoletes the previous editions, C24-5036-0 and C24-3428-1. Changes are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (•) next to a figure title or page number indicates that the entire figure or page should be reviewed.

Significant changes have been made throughout the Supervisor, Job Control, and Linkage Editor sections of this publication, and these sections should be reviewed in their entirety. Diagnostic messages for the Supervisor, Linkage Editor, and Librarian have been added to the Appendix.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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

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

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

The Disk Operating System consists of the following components.

Control Program

The control program constitutes 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, interruption conditions, and other functions for all problem programs. Part of the Supervisor resides in main storage at all times. Processing time is divided between the Supervisor and the program(s) being executed. This is true for the user's programs as well as other IBM-supplied components of the system. Certain functions of the Supervisor are provided by transient routines that remain in disk storage until needed and which 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, Job Control type functions are performed by the foreground initiator.

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

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.

With the exception of SYSRES and SYSLOG, system units are used only with programs running in a batched-job environment (referred to as background programs).

System Service Programs

The system service programs provide the functions of generating the system, creating and maintaining the library sections, and editing 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. All programs are edited into an area of the resident disk pack by this program. These programs can then be permanently placed in the core image library of the system, requiring only control statements to call them for execution, or they can be stored on disk temporarily, executed, and then overlaid by new programs.

2. Librarian. This is a group of programs, used for maintaining and reorganizing the disk library areas and providing 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 is used to store object modules which 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 is used to store IBM-supplied macro definitions and user-defined source statement routines (such as macro definitions) in the resident pack built to provide extended program-assembly capability.

Processing Programs

All user programs are run within the Disk Operating System environment, using the functions of the control program. Minimum resident packs may consist of only the control program and one or more user programs or, for other types of applications, the control program and the Linkage Editor, with user programs loaded and edited from cards or tape into a specified area in disk storage, and then into main storage for execution. A full system may include user's programs and the following IBM-supplied programs:

1. Language Translators: Assembler, COBOL, FORTRAN, RPG, and PL/I.
2. Sort/Merge.
3. Utilities.
4. Autotest.

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 programs are assigned to fixed locations when they are cataloged to the system. A program occupies a contiguous area of storage. The amount of main storage allotted to programs to be executed may be determined when the system is generated, or it may be determined by the operator when the program is loaded into main storage for execution.

Background vs Foreground Programs

There are two types of problem programs in multiprogramming: background and foreground. Background programs are initiated by Job Control from the batched-job input stream. Foreground programs are initiated by the operator from the printer-keyboard. Foreground programs do not execute from a stack. When one completes, the operator must explicitly initiate the next program.

Background and foreground programs initiate and terminate asynchronously from each other. Neither is aware of the other's status or existence.

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 interruptions enabled. When an interruption occurs, the Supervisor gains control, processes the interruption, 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 (i.e., no program can process), the system is placed in the wait state enabled for interruptions. Interruptions are received and processed by the Supervisor. When an interruption satisfies a program's wait condition, that program becomes active and competes with other programs for CPU processing time.

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

Note that programs produced by the FORTRAN and PL/I compilers may not be run as foreground programs, for the object programs produced by these compilers use system facilities available only to background programs. All IBM-supplied programs must also be run only as background programs, in a multiprogramming environment.

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 32K.

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

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

SYSTEM CONFIGURATION

This section presents the minimum system configuration required to operate the Disk Operating System and features in addition to the minimum 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 (24K bytes are required for multiprogramming, 1412/1419 MICR document processing, and for assigning system input/output files to disk).

Standard instruction set. See Note 1.

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

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

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

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

One 1052 Printer-Keyboard.

One 2311 Disk Storage Drive.

Note 1: Language translators may require extended instruction sets.

Note 2: Telecommunications requires a multiplexor channel and at least one selector channel.

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

Also, MICR processing requires either the Direct Control Feature or the External Interrupt Feature.

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

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
Selective Tape Listing Features (1403) for continuous paper tapes.
Dual Address Adapter (1419) to allow more stacker selection processing. Once processing with the Dual Address Adapter is established, 1412's and 1419's cannot be mixed.
Additional main storage up to 16,777,216 bytes.

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

1. 1442 Card Read Punch
2. 2501 Card Reader
3. 2520 Card Read Punch
4. 2540 Card Read Punch
5. 1403 Printer
6. 1404 Printer (for continuous forms only)
7. 1443 Printer
8. 1445 Printer
9. 1052 Printer-Keyboard. It is used for operator communication.
10. 2671 Paper Tape Reader
11. 2311 Disk Storage Drive
12. 2321 Data Cell Drive
13. 2401, 2402, 2403, 2404, and 2415 Magnetic Tape Units.
14. 1285 Optical Reader (maximum of 8 are supported)
15. 1287 Optical Reader (maximum of 8)*
16. 1030 Data Collection System
17. 1050 Data Communication System
18. 1060 Data Communication System
19. 2260 Display Station
20. AT&T 83B3 Selective Calling Stations

21. AT&T Teletypewriter Terminal, Models 33 and 35
22. Western Union Plan 115A Outstations
23. 2740 Communications Terminal
24. 7770 and 7772 Audio Response Units
25. 1412 Magnetic Character Reader* (maximum number supported is dependent upon the system configuration)
26. 1419 Magnetic Character Reader* (maximum number supported is dependent upon the system configuration)
27. IBM System/360 (Model 30, 40, 50, 65, or 75)**

*Programming specifications for using these devices may be used for planning purposes only. Source programs must not contain instructions for these devices until the Disk Operating System includes the appropriate programming. Diagnostics indicating improper operation codes or operands are issued if coding for these devices is included in a source program.

**System/360 to System/360 binary synchronous communication may take place over either switched or nonswitched communication lines. Each System/360 must be attached to the communication line through a 2701 Data Adapter Unit or a 2703 Transmission Control Unit connected to the System/360 multiplexor channel. In the case of a nonswitched line, either or both may use a 2701 attached to the System/360 selector channel.

The preceding devices (16 through 23) are attached by means of a private, leased, or common-carrier network to the multiplexor channel through a 2701 Data Adapter Unit, 2702 or 2703 Transmission Control Unit. When the 2701, 2702, or 2703 is attached to the multiplexor channel, burst-mode devices (magnetic tape and DASD) must be attached to a selector channel. The 2260 Display Station may also be attached to the System/360 channel (local attachment) via the 2848 Display Control Unit.

CONTROL STATEMENT CONVENTIONS

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

1. Uppercase letters and punctuation marks (except as described in items 3 through 5 below) 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.

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

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

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

1. Storage protection (required for multiprogramming)
2. Interruption handling
3. Channel scheduling
4. Device error recovery
5. Operator communication
6. Program retrieval
7. End-of-job handling
8. Timer services (optional).

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

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

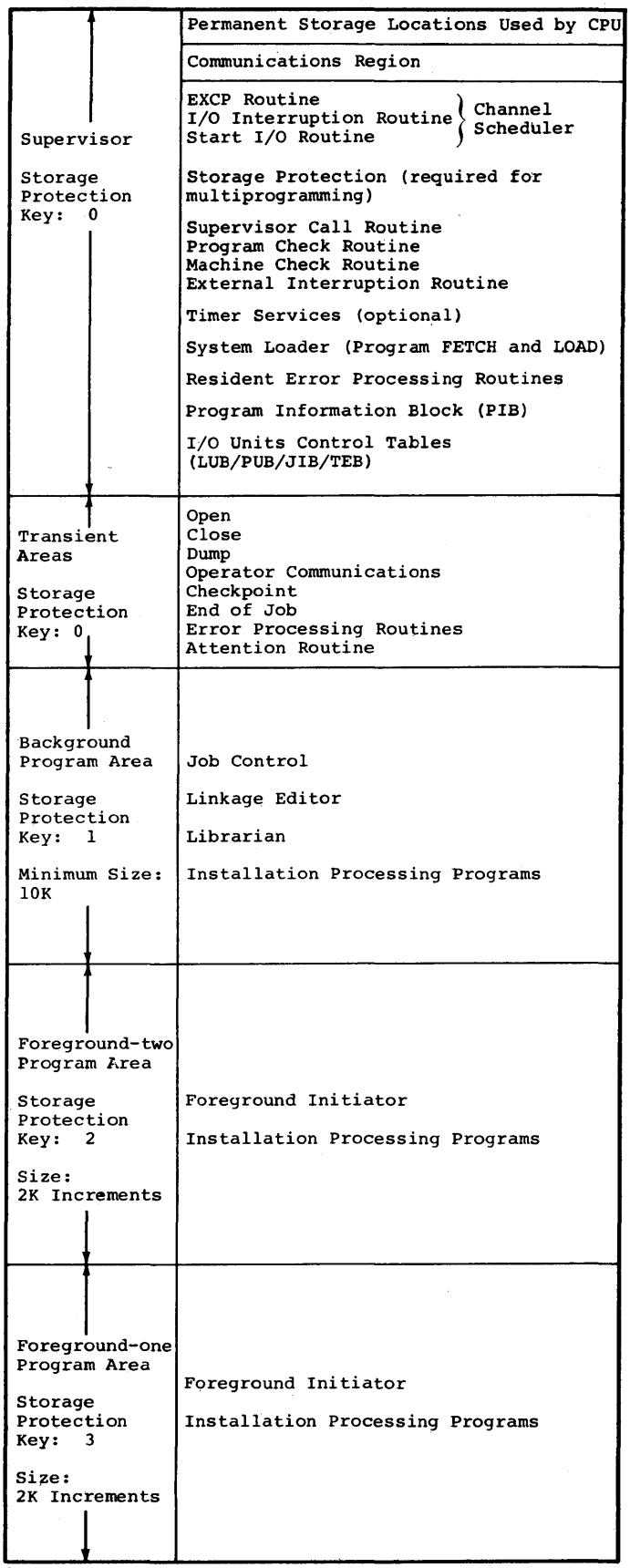
The Supervisor is generated from a set of source statements by way of an Assembler run.

MAIN STORAGE ORGANIZATION

The Supervisor occupies the low area of main storage. The transient routines are called into the transient area (overlaying the previous routine in the area) and executed when needed. The area occupied by the background program begins just past the transient area. The background program area must be a minimum of 10K bytes. (IBM-supplied programs such as the Linkage Editor require at least 10K bytes to perform their functions. Certain language translators may require a background area larger than 10K bytes.) Following the background program area is the foreground-two program area. This area must be defined in increments of 2K. (Storage protection requires that main storage be divided into blocks of 2K bytes.) Following the foreground-two program area is the foreground-one program area. As with the foreground-two area, the foreground-one area must be defined in increments of 2K. The minimum size of a foreground area is 0K; the maximum is 510K. The main storage map in Figure 1 shows the relationship between the Supervisor and the problem program areas.

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

Note that 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.



●Figure 1. Main Storage Organization

I/O UNITS CONTROL TABLES

The principal components of the I/O Units Control Tables are the Logical Unit Blocks (LUB) and Physical Unit Blocks (PUB). These tables, defined when the system is generated, are required for channel scheduling, input/output unit assignment and control, and maintenance of miscellaneous information about jobs, such as multiple I/O assignments.

Each LUB is two bytes and represents one logical (symbolic) I/O unit. Each LUB references an entry in the PUB. LUB's corresponding to logical units are ordered according to the logical units they represent. Information concerning the ordering of LUB's is contained in the subsection under Job Control entitled Logical Unit Block (LUB) and Physical Unit Block (PUB). The LUB's are grouped into two classes: system logical units and programmer logical units. The number of programmer logical units is a system generation parameter with a maximum of 245.

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 PUB's are ordered by the number of the channel to which the various devices are attached. The number of PUB's is a system generation parameter with a maximum of 255.

COMMUNICATION REGION

The communication region is a 46-byte storage area within the Supervisor region for use by the Supervisor and problem programs. Certain macro instructions are available to allow access to the information contained in this region. Fields in the communication region are addressed relative to the first byte of the region.

The layout of the communication region in the Supervisor is shown in Figure 2.

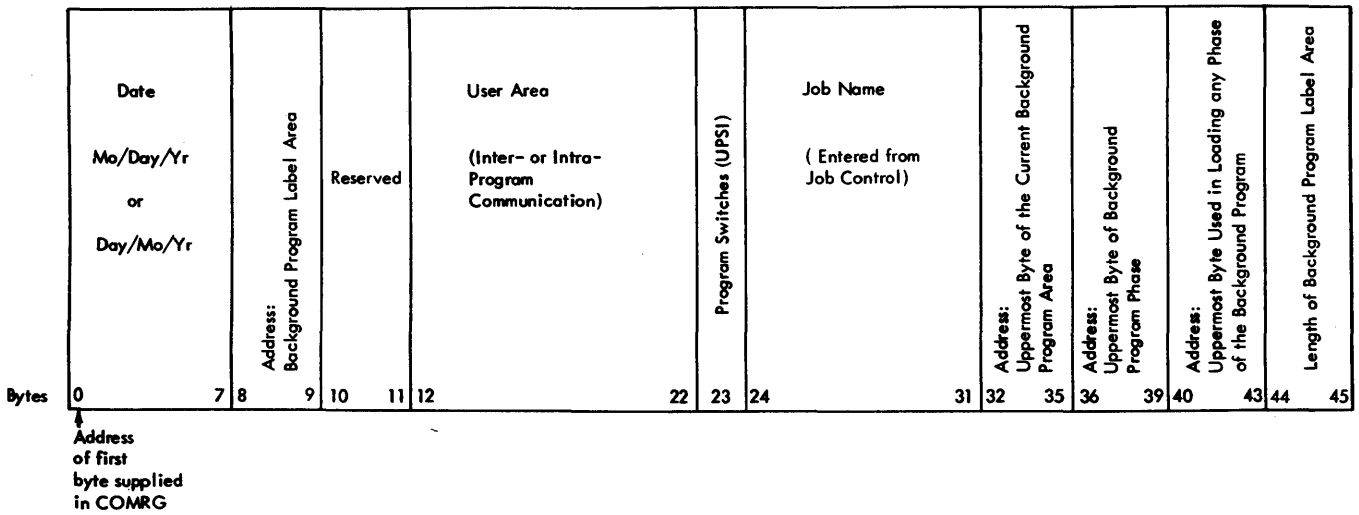


Figure 2. Communication Region in Supervisor

8 bytes — Calendar date. Supplied during IPL procedure or by DATE control statement. Can be used for dating printed output of the problem program. In one of two forms: mm/dd/yy or dd/mm/yy where mm is month, dd is day, and yy is year.

2 bytes — Address of background program label area.

2 bytes — Reserved for control program use.

11 bytes — User area (for inter- or intra-program communications). All 11 bytes set to binary zero when JOB control statement is encountered.

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

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

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

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

4 bytes — Address of the uppermost byte used in loading any phase of the background program. This value may be inaccurate if the program LOADs any phase above its linkage edited origin address, or if the program phases are not all linkage edited together.

2 bytes — Length of background program label area.

Communication Region Macro Instructions

Macro instructions are provided to allow the problem program to access the communication region. A brief discussion of these macro instructions follows. Details can be found in the Supervisor and I/O Macros publication listed on the front cover of this publication.

COMRG -- Get address of communication region:
This macro instruction allows the problem program to address information stored in the communication region (obtain date, test switches, etc). The address of the first byte of the region is placed in general register 1.

MVCOM -- Move to communication region:
This macro instruction allows the problem program to modify the content of the user area and UPSI (bytes 12

through 23) of the communication region. The operand field of the MVCOM macro instruction contains three operands. The first specifies the first communication 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. Only background programs may use the MVCOM macro instruction.

SUPERVISOR FUNCTIONS

STORAGE PROTECTION

A storage protection key of 0 is set for the Supervisor and 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.

INTERRUPTION HANDLING

An interruption can be caused by either a program instruction or a machine condition. The Supervisor automatically handles all interruptions so that the programmer need not be directly concerned with them. In most cases after an interruption is handled, control is returned to the point of interruption as if no break had occurred in the instruction sequence.

There are five kinds of interruptions. They are:

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

Figure 3 illustrates the flow of control between the Supervisor and a problem program during an interruption. Control is in the problem program initially. An inter-

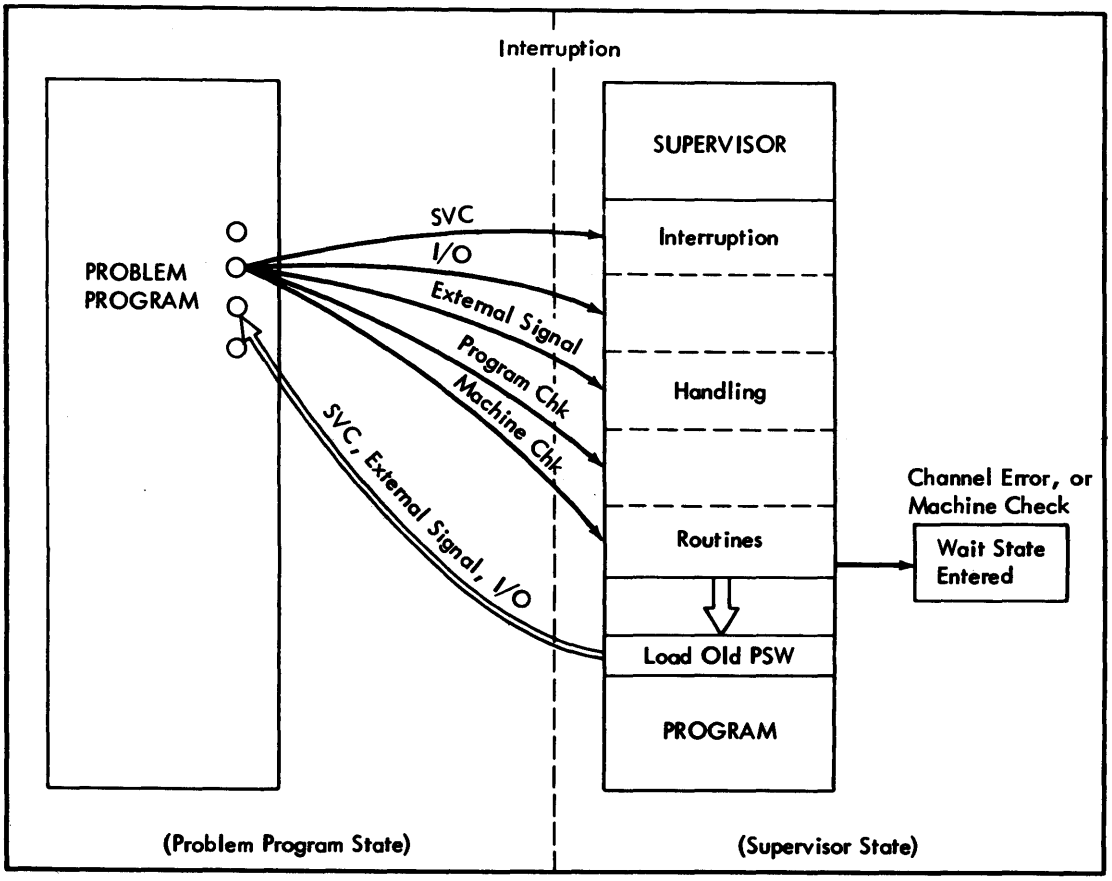


Figure 3. Flow of Control Between Supervisor and Problem Program During an Interruption

ruption 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 interruption, control is given to an appropriate handling routine. Upon completion of the routine, the program may be restored to its original condition (via the old PSW). Control is normally given back to the problem program at the point where it was interrupted. The user may have control of program check and external interruptions.

Supervisor Call

The supervisor call interruption 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 certain interruption 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 (background programs only).

CLOSE To close an input/output file.

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 interruption.

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

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

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

MVCOM To modify the content of the user area in the communication region (background programs only).

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 interruption) 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 interruption with a specific parameter. The interruption routine analyzes the parameter and gives control to another routine for the actual handling of the interruption.

External Interruption

An external interruption can be caused by the timer feature, or by the operator pressing the console interrupt key, or by an external signal. For foreground programs, the 1052 attention key is used to gain access to problem program communication routines.

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

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

1. Maintains the time of day which the user can reference at any point within the execution of the problem program.
2. Time-stamps 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. Enables the user to set the timer for a specified interval of time and either wait on it or to get control at a prespecified address after the time interval has elapsed. The interval timer can be used with only one class of program (background, foreground-one, or foreground-two) at a time.

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

Five macro instructions are provided for use with external interruptions. These macro instructions are TECB, GETIME, SETIME, STXIT, and EXIT. Figure 4 illustrates a typical sequence of events following an external interruption.

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

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

SETIME -- Set interval timer:
This macro instruction can be used to request the Supervisor to 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

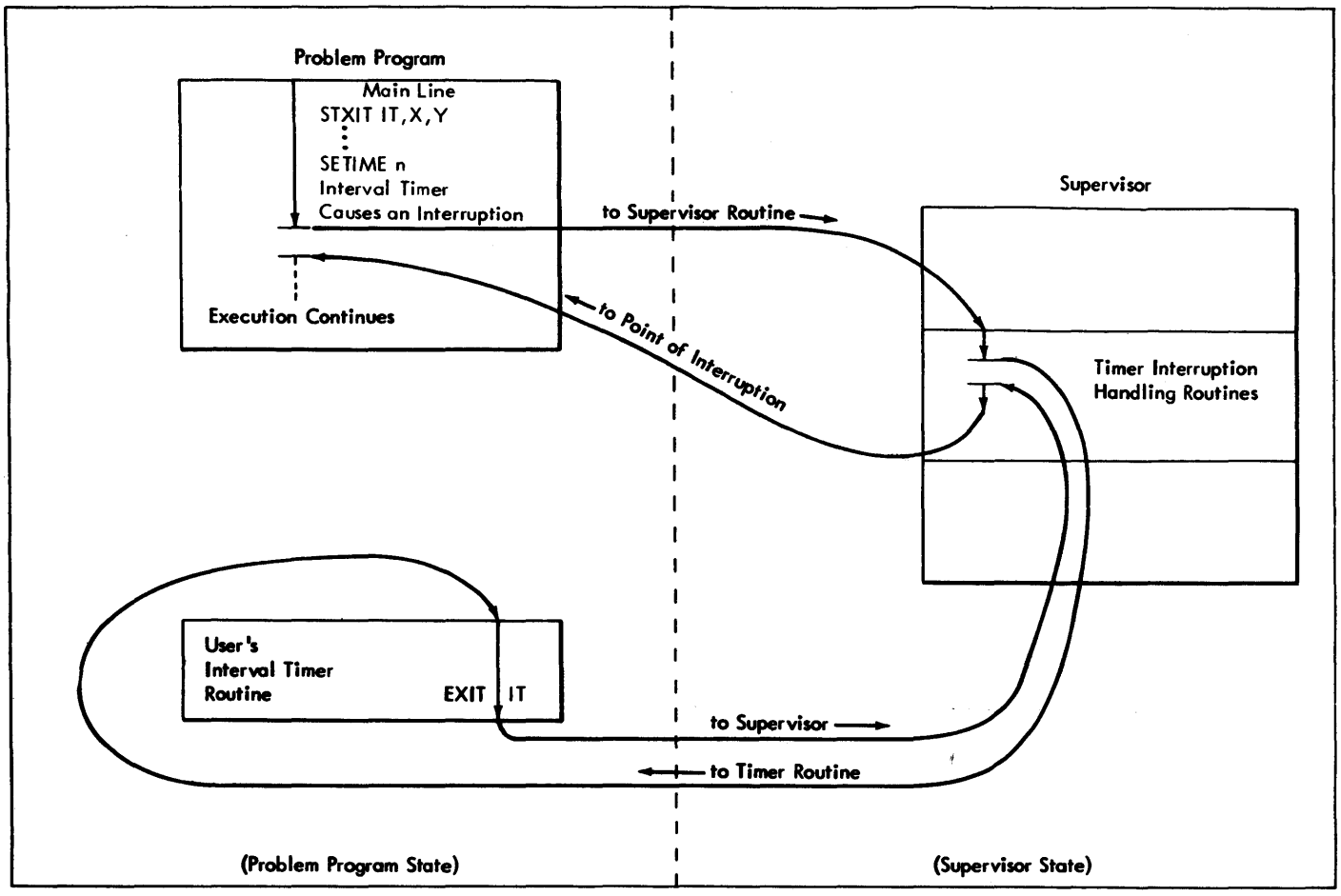


Figure 4. Typical Sequence of Events After a Timer Interruption Due to Use of SETIME

presence is indicated when the system is generated.

STXIT -- Set linkage to user interruption 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 interruptions.

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

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

Program Check

Each program can select which of the following options is to be taken in the event of a program check.

1. Abort -- The job being executed is terminated and a message output on SYSLOG and SYSLST describes the cause of the termination of a background program. For foreground programs, the output is on SYSLOG and SYS000.

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 for background programs. The job is then terminated. For foreground programs, the output is on SYSLOG and SYS000.

3. Transfer to User Routine -- If the address of a subroutine is supplied by the user by way of the STXIT macro instruction, the program-check interruption routine will branch to that subroutine when an appropriate interruption occurs. The user routine can determine the cause of the interruption. Return to the point of interruption is possible by means of the EXIT macro instruction. This facility is a system generation option.

4. Program Mask in PSW -- The program mask bits of the PSW (Program Status Word) are initially set to zero. If the user wishes to enable a program interruption, he will change this configuration by use of the Set Program Mask (SPM) instruction. The program mask bits will be reset to zero after each job step and after each FETCH of a problem program phase.

Machine Check

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

The same condition results from an I/O channel failure. The message code 1S is in bytes 0 and 1 of main storage.

Input/Output Interruptions

An input/output interruption 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 which was busy or not ready is now available for use.

3. Control unit available -- (Control Unit End). A control unit which was busy is now available for use.

4. I/O attention -- This results from pressing the request key on the 1052.

When one of these conditions is detected, control transfers to the Channel Scheduler. (Note that for non-Tele-processing devices, a program-controlled interruption (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 interruptions--normal com-

pletion of data transfer, error detection, end-of-file detection, attention (1052).

4. Perform error detection and correction.

5. Detect end-of-job or end-of-job-step control statements on SYSRDR and SYSIPT.

6. Monitor DASD channel programs for file protection and address continuity for disk system input/output.

I/O devices in the System/360 are attached to channels rather than attached 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 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.

Two types of channel are provided in this system: selector channels and the multiplexor channel. The selector channels allow I/O operations for devices on these channels to be overlapped with CPU processing and I/O operations on other channels. On the multiplexor channel, tape and DASD I/O operations cannot be overlapped with other I/O operations on the same channel. On System/360 Models 30 and 40, they may not be overlapped with processing. Card, printer and other low-speed (byte interleave mode) I/O devices on the multiplexor channel can be overlapped 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, as listed on the front cover of this publication.

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 interruptions. If the interruption 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 is returned to the problem program at the point of interruption. 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 interruptions are outstanding cannot be serviced until the device end is received. These requests will be 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 has been completely loaded (about 2ms), but device end is not received until completion of the print operation (55ms).

The Channel Scheduler detects the following specific status conditions:

1. Wrong Length Record (WLR)

2. Unit Exception

3. Channel and device errors.

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

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

For certain devices (1052), an I/O operation can be initiated by the operator. To do this, the operator 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:

1. 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 (CCW's).

2. 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 see 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 interruption to the Channel Scheduler. The program is taken out of Wait state, the switch is set to show the completion, and control is returned to the problem program.

3. 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.

DEVICE ERROR RECOVERY

Each I/O device or class of I/O devices has a unique device error recovery routine. The appropriate routine is entered from the Channel Scheduler upon detection of an error. All these routines have one function in common. That is, an attempt is made to recover from the error. This may be by programming (re-reading 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 on whether logical IOCS is used, some or all of the options just described are available. Choices 3 and 4 are available only through logical IOCS. In the absence of any other options, only choice 2 is available. Tape error statistics can also be printed on the printer-keyboard if such statistics were specified when the system was generated.

OPERATOR COMMUNICATION

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

The Supervisor permits:

1. Full-text messages to the operator. These messages are either information

only or indications of required operator action.

2. Operator-initiated instructions to the control program.

3. Communication between the operator and the problem program.

COMMUNICATION TO THE OPERATOR

The control program communicates with the operator by issuing messages on SYSLOG, normally assigned to the IBM 1052 Printer-Keyboard. If no response or action is required, an I indicator is included in the message and processing continues. 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, 4 for logical IOCS, 5 for PL/1, 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 will 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 make a choice between alternate courses of action.
I	Information:	The message does not require immediate operator action. For example, this type of message can be used to indicate the termination of a problem program.
W	Wait:	Used when a condition (such as an error on SYSRES) occurs that makes it impossible to continue processing. This indicator is not printed on the printer-keyboard. Instead, a message number is placed in byte 0 of main storage. The indicator W is placed in byte 1 of main storage. The Wait state is entered, and all interruptions are disabled. The only way that the system can be restarted is through the IPL procedure.
S	SEREP:	Used when a hardware condition occurs that makes it impossible to continue processing. This indicator is not printed on the printer-keyboard, but may be displayed on the console. A 0 in byte 0 of main storage indicates a machine-check interruption; 1 indicates an I/O channel failure. The indicator S is stored in byte 1 of main storage. A special diagnostic storage display program

(SEREP) supplied to customer engineers should be used when an S condition occurs.

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 would type on the 1052 the assignment for SYSRDR.) PLEASE ASSIGN SYSRDR is the content of the message.

COMMUNICATION FROM THE OPERATOR

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

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

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

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

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

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

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

Job Control commands may be issued only between job steps of the background program. A suspension of processing between job steps can be achieved by the programmer using the PAUSE control statement, or by the operator using the PAUSE command. When Job Control is ready to process these commands, the message READY FOR COMMUNICATIONS is printed on the printer-keyboard.

ATTN commands may be issued at any time. Pressing the request key on the printer-keyboard will cause the message READY FOR COMMUNICATIONS to be printed on the printer-keyboard. ATTN commands may then be issued.

Foreground-program initiation commands may be issued following the ATTN command START. The START command causes control to be given to the foreground initiation routines.

A description of Job Control and ATTN commands follows. Commands for initiating a foreground program are contained in the section Foreground Program Initiation. A listing of all commands is shown in Appendix C.

Job Control Commands

There are 20 Job Control commands. They are: ASSGN (assign logical name), CLOSE (close I/O unit), DVCDN (device down), DVCUP (device up), MTC (magnetic tape control), RESET (reset I/O assignments), STOP (stop background processing), LISTIO (list I/O assignments), LOG (log Job Control statements), NOLOG (suppress logging), CANCEL (cancel job), PAUSE (pause), MAP (map main storage), ALLOC (allocate main storage), SET (set value), UCS (load Universal Character Set buffer), HOLD (hold foreground unit assignments), RELSE (release foreground unit assignments at EOJ), UNA (immediately unassign foreground unit assignments), and Ⓑ (end of communications).

ASSGN -- Assign Logical Name Command: The ASSGN command is used to assign a logical I/O unit to a physical device.

Operation	Operand
ASSGN	SYSxxx,address [{,X'ss' / ,ALT}] [,TEMP]

The entries in the operand field represent the following.

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

SYSRDR
SYSIPT
SYSIN
SYSPCH
SYSLST
SYSOUT
SYSLNK
SYSLOG
SYS000-SYS244

Note that when SYSOUT is assigned, the magnetic tape device must not be the permanent assignment of either SYSLST or SYSPCH.

For 1419 Magnetic Character Readers with the dual-address adapter, two assignments are required for each reader/sorter if pocket light operations are to be performed. If no pocket light operations are to be performed, the user need only assign a logical unit to the primary control unit address (1419P).

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 cancellation of the job. A permanently assigned system logical unit must be permanently unassigned (via the ASSGN Command) in order to make both the logical unit and physical device available.

IGN -- Indicates the logical unit is to be unassigned and that all program references to the logical device are to be ignored.

X'ss' — Device specifications (used to specify mode settings for 7-track and dual density 9-track tapes). If X'ss' is not specified, 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. C0 for a 9-track single density tape unit is 800 bpi, whereas for a dual density tape unit, it is 1600 bpi. C8 is an alternate mode setting for 9-track dual density tapes only. 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	dual density 9-track		
C8	800	dual density 9-track		

Note that the first 15 entries in this table are valid only for 7-track tape. The last three entries are valid only for 9-track tape.

ALT — Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The characteristics of the alternate unit must be the same as those of the original unit. Multiple alternates may be assigned to a symbolic unit.

TEMP — Indicates the assignment for the logical unit will be destroyed by the next JOB statement. Unless this option is taken, the assign-

ment made is carried from job to job.

CLOSE -- Close Output Unit Command: The CLOSE command is used to close either a system or programmer output logical unit assigned to a magnetic tape, or a system logical unit assigned to a 2311. The logical unit may optionally be reassigned to another device, unassigned, or, in the case of a magnetic tape file, switched to an alternate unit. Note that 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-SYS244), 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 file mark, an EOV trailer record, two file marks, and rewinding and unloading the tape. For a complete description of opening and closing system input/output units, see the subsection under Job Control entitled System I/O Operations.

Operation	Operand
CLOSE	SYSxxx [,X'cuu'[,X'ss'] / ,UA / ,IGN / ,ALT]

SYSxxx — For 2311: SYSIN, SYSRDR, SYSIPT, SYSPCH, or SYSLST

For magnetic tape: SYSPCH, SYSLST, SYSOUT, or SYS000-SYS244

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 dual density 9-track tape. The specifications are shown in ASSGN -- Assign Logical Name. If X'ss' is not specified, the mode settings remain unchanged.

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

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

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

DVCDN -- Device Down Command: The DVCDN command is used to inform the system that a device is no longer physically available for system operations. It also resets all device assignments to standard. Note that this command utilizes the logical transient area, and blocks out operator communication functions until it is completed.

Operation	Operand
DVCDN	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.

DVCUP -- Device Up Command: The DVCUP command is used to inform 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.

Operation	Operand
DVCUP	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.

MTC -- Magnetic Tape Control Command: The MTC command controls magnetic tape operations. The first entry in the operand field specifies the operation to be performed.

Operation	Operand
MTC	opcode, {X'cuu' / SYSxxx} [,nn]

The first entry in the operand field can be:

Opcode	Meaning
BSF	Backspace file
BSR	Backspace record
ERG	Erase gap
FSF	Forward space file
FSR	Forward space record
RUN	Rewind and unload
REW	Rewind
WTM	Write tape mark

The second 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. The alternate second entry, SYSxxx, represents any logical unit. The optional third entry, nn, is a decimal number (01-99) representing the number of times the specified operation is to be performed.

RESET -- Reset I/O Assignments Command: The RESET command resets certain I/O assignments to the standard assignments. 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).

Operation	Operand
RESET	SYS 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.

STOP -- Stop Background Processing Command: The STOP command can be used in a multiprogramming environment to indicate that there are no more background jobs to be executed.

Operation	Operand
STOP	blank

This command removes the background job from the system's task selection mechanism. The background area remains at least 10K bytes in size. Since this is a job control command, job control remains (dormant) in the background area, and is therefore accessible without reinitialization. If no foreground program is being executed, the system is placed in the wait state. Processing of background programs can be initiated by the START command, discussed in ATTN Commands.

Note that in a multiprogramming environment, it may be advisable to use a STOP command instead of a PAUSE command. The PAUSE command causes a read to be issued to SYSLOG, tying up the 1052 until the operator responds.

LISTIO -- List I/O Assignment Command: The LISTIO command is used to cause the system to print a listing of I/O assignments. The listing appears on the printer-keyboard (SYSLOG). Note that this command utilizes the logical transient area, and blocks out operator communication functions until it is completed.

Operation	Operand
LISTIO	SYS PROG F1 F2 ALL SYSxxx UNITS DOWN UA X'cuu'

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

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

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

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

ALL — Lists the physical units assigned to all logical units.

SYSxxx — Lists the physical units assigned to the logical unit specified.

UNITS — Lists the logical units assigned to all physical units.

DOWN — Lists all physical units specified as inoperative.

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

X'cuu' — Lists the logical units

assigned to the physical unit specified.

Physical units are listed with current device specification for magnetic tape units. Logical units are listed with ownership (background, foreground-one, or foreground-two), when applicable.

LOG -- Log Command: The LOG command is used to cause the system to log columns 1-72 of all Job Control statements and foreground initiation commands on SYSLOG until a NOLOG command is sensed.

Operation	Operand
LOG	blank

The operand field is ignored by the system.

NOLOG -- Suppress Logging Command: The NOLOG command is used to cause the system to suppress the logging of all foreground initiation commands and Job Control statements except JOB, PAUSE, *, and /& until a LOG command is sensed.

Operation	Operand
NOLOG	blank

The operand field is ignored by the system.

CANCEL -- Cancel Job Command: The CANCEL command is used to cancel the execution of the current background job. Cancellation is immediate.

Operation	Operand
CANCEL	blank

PAUSE -- Pause Command: The PAUSE command is used to cause Job Control processing to pause at the end of the next background program job step, or at the end of the current background program job. At that time, the printer-keyboard is unlocked for message input. The end-of-communications indication Ⓑ causes processing to continue.

Operation	Operand
PAUSE	[any user comment]

The operand of the PAUSE command is not processed by the system. It is used only for operator documentation.

MAP -- Map Main Storage Command: The MAP command is used to cause the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment. It indicates what programs are being executed and which has access to the interval timer. For a description of the map of main storage produced by this command, see ATTN Commands: MAP Command.

Operation	Operand
MAP	blank

ALLOC -- Allocate Main Storage Command: The ALLOC command permits the operator to allocate main storage among foreground and background programs. The number of bytes to be allocated for one or both foreground areas is specified in 2K increments (2K = 2048 bytes). If only one foreground area is referenced, it is assumed that the amount of storage allocated to the other is not to change.

Operation	Operand
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 must always be 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 will not be allocated in the following instances.

1. The allocation would cause a decrease

in the storage allocated to an active foreground or background program.

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

3. The allocation would reduce the background area to less than 10K bytes.

SET -- Set Value Command: The SET command is used to initialize the date, clock, UPSI configuration, specify the number of lines to be printed on SYSLST, and specify the remaining disk capacity when SYSLST or SYSPCH is assigned to disk.

Operation	Operand
SET	[DATE=n1][,CLOCK=n2] [,UPSI=n3][,LINECT=n4] [,RCLST=n5][,RCPCH=n6]

The entries in the operand field represent the following.

DATE=n1 Sets the system date permanently to the specified value. 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 Never given at IPL time, but can be used at other times. Sets the bit configuration of the UPSI byte in the communication region. n3 consists of one to eight digits, either 0, 1, or X. Positions containing 0 will be set to 0; positions containing 1 will be set to 1; positions containing X will be unchanged. Unspecified rightmost positions are assumed to be X.

LINECT=n4 Never given at IPL time, but can be used at other times. Sets the standard number of lines to be printed on each page of SYSLST. n4 is an integer between 30 and 99.

RCLST=n5 Never given at IPL time, but can be used at other times. n5 is a decimal number indicating the minimum number of records remaining to be written on SYSLST when assigned to disk before a warning is issued to the operator that the capacity of the extent is near.

Note that this warning is given after the job ends, and that if extent limits are exceeded, the 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 Never given at IPL time, but can be used at other times. 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.

Note that this warning is given after the job ends, and that if extent limits are exceeded, the 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.

UCS -- Load Universal Character Set Buffer Command: The UCS command causes the 240-character Universal Character Set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit. The 240 EBCDIC characters correspond to the 240 print positions on 1403 chains and trains. A character sent to the printer for printing is matched against the

characters in the UCS buffer. When a match occurs, the corresponding chain/train character is printed in the print line position which the output character occupied. Thus, the user, through the UCS buffer and the many chains/trains available, can adapt his 1403 Printer to many variable printing applications.

The logical unit must be assigned to a 1403 Printer with the UCS feature. It is the user's responsibility to assemble, linkage edit, and catalog his UCS buffer phases into the core image library, and to mount the new chain or train before the UCS command is executed.

Operation	Operand
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.

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 provided to verify that the mounted chain or train is compatible with the UCS buffer contents.

The UCS phase format is as follows:

240-character UCS buffer load	80-character verification message

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

Operation	Operand
HOLD	{F1[,F2] / F2[,F1]}

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

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

RELSE -- Release Foreground Unit Assignments at EOJ Command: The RELSE command causes all I/O assignments for the foreground area(s) specified to be set to unassigned at the end of any job that is initiated for that area.

Operation	Operand
RELSE	{F1[,F2] / F2[,F1]}

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

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

Operation	Operand
UNA	{F1[,F2] / F2[,F1]}

The foreground area specified must be currently inactive. If only one foreground area is referenced, a pervious HOLD for the other area remains in effect. This command is intended to be used to free physical units currently assigned to a foreground area under the HOLD command.

Ⓑ -- End-of-Communication Command: The end-of-communications command must be issued whenever the operator is finished communicating with the system. It causes the communications routine to return control to the mainline job.

Operation	Operand
Ⓑ	blank

Ⓑ is the end-of-block character, alter code 5.

ATTN Commands

There are ten ATTN commands. They are: PAUSE (pause), CANCEL (cancel job), LOG (log Job Control statements), NOLOG (suppress logging), MAP (map main storage), ALLOC (allocate main storage), START (start program processing), MSG (transfer control), TIMER (interval timer), and Ⓑ (end-of-communications).

PAUSE -- Pause Command: The PAUSE command is used to cause Job Control processing to pause at the end of the current background program job step, or at the end of the current background program job. At that time, the printer-keyboard is unlocked for message input. The end-of-communications indication Ⓑ causes processing to continue.

Operation	Operand
PAUSE	[any user comment]

The operand of the PAUSE command is not processed by the system. It is used only for operator documentation.

CANCEL -- Cancel Job ommand: The CANCEL command is used to cancel the execution of a background, foreground-one, or foreground-two program. Cancellation is immediate.

Operation	Operand
CANCEL	{BG / F1 / F2}

BG Indicates the background job is to be cancelled.

F1 Indicates the foreground-one program is to be cancelled.

F2 Indicates the foreground-two program is to be cancelled.

If the operand field is blank, the background job is assumed to be canceled.

LOG -- Log Command: The LOG command is used to cause the system to log columns 1-72 of all Job Control statements and foreground initiation commands on SYSLOG until a NOLOG command is received.

Operation	Operand
LOG	blank

The operand field is ignored by the system.

NOLOG -- Suppress Logging Command: The NOLOG command is used to cause the system to suppress the logging of all foreground initiation commands and Job Control statements except JOB, PAUSE, *, and /& until a LOG command is received.

Operation	Operand
NOLOG	blank

The operand field is ignored by the system.

MAP -- Map Main Storage Command: The MAP command is used to cause the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment. It indicates what programs are being executed, and which has access to the interval timer.

Operation	Operand
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 - Indicates 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 were 11.2K, the map would indicate 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.

ALLOC -- Allocate Main Storage ommand: The ALLOC command permits the operator to allocate main storage among foreground and background programs. The ATTN command ALLOC cannot be used to reduce the background area at any time. The number of bytes to be allocated for one or both foreground areas are 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.

Operation	Operand
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 must always be 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 will not be allocated in the following instances.

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.

3. The allocation would reduce the background area.

START -- Start Background or Foreground Processing Command: The START command can be used to initiate a foreground program or to resume batch job processing.

Operation	Operand
START	{ BG / F1 / F2 }

BG — Causes Job Control to read the next control statement in the background program job stream. The START BG command is effective only if a STOP command was issued previously. (The STOP command is described in Job Control Commands.)

F1 or F2 — Specifies a foreground program is to be initiated and indicates the area to be used. The foreground initiation routines are given control. Commands that may be issued following the START command are described in the section entitled Foreground Program Initiation. If the specified foreground area is either being used by a program or has no area allocated to it, a message is printed on the printer-keyboard informing the operator of the condition.

MSG -- Transfer Control Command: The MSG command can be used to give control to a foreground program operator communications routine previously activated by a STXIT instruction.

Operation	Operand
MSG	{F1 / F2}

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

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

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

TIMER -- Interval Timer Command: The TIMER command causes interval timer support to be given to the program specified.

Operation	Operand
TIMER	{BG / F1 / F2}

If interval timer support is already allocated to the program 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 cancellation of that program. Once established timer support remains with an area from program to program until changed by a TIMER command.

Ⓑ -- End-of-Communications Command: The end-of-communications command must be issued whenever the operator is finished communicating with the system. It causes the communications routine to return control to the mainline job. Note that START, CANCEL, and MSG commands automatically terminate ATTN communications.

Operation	Operand
Ⓑ	blank

Ⓑ is the end-of-block character, alter code 5.

FOREGROUND PROGRAM INITIATION

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

The operator initiates a foreground program by pressing the request key on the printer-keyboard. Control is given to the message (ATTN) routine, which reads commands from the operator via the printer-keyboard. The START command (discussed in ATTN Commands) indicates a foreground 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 foreground program initiation routine; otherwise, the operator is notified that he has given an invalid command.

The foreground 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 programmer logical unit SYSnnn.
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 for self-relocating foreground programs. For non-self-relocating foreground programs, the label information area is determined by the

LBLTYP statement, described in the Job Control section of this manual.

When the EXEC command is encountered, the foreground initiator directs the Supervisor to provide loading information for the program to be executed. If the program has not been cataloged to the core image library of the system, the operator is notified. 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 foreground initiator directs the program to be loaded following the label information area in the foreground area. Its entry point is given by the Supervisor when it is loaded. A non-self-relocating program is loaded using the information obtained when the program was cataloged. Diagnostics for such conditions as the program being outside the limits of the foreground area, are issued by the Supervisor when the program is loaded.

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

A foreground 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 foreground 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 (if specified when the system was generated) are typed on the printer-keyboard for tapes used by the program.

3. 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.)

4. All I/O assignments made for the program are cancelled so the devices may be available to subsequent programs unless held across jobs by the HOLD command.

5. 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.

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

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

Foreground Initiation Commands

Foreground initiation commands are submitted by the operator following a START command to the ATTN routine. They may be submitted through the printer-keyboard (SYSLOG), or through a card reader (following a READ command). Two slashes (//) in columns 1 and 2 are optional for VOL, XTENT, DLAB, and TPLAB commands, and are accepted by the foreground initiator for these commands only.

There are 19 foreground initiation commands. They are: READ (specify reader), LISTIO (list I/O assignments), ASSGN (assign logical name), VOL (volume information), DLAB (DASD label information), XTENT (DASD extent information), TPLAB (tape label information), CANCEL (cancel initiation), EXEC (execute program), LOG (log job control statements and foreground initiation commands), NOLOG (suppress logging), HOLD (hold foreground unit assignments), RELSE (release foreground unit assignments at EOJ), UNA (immediately unassign foreground unit assignments), MAP (map main storage), MSG (transfer control), PAUSE (pause), TIMER (interval timer), and Ⓑ (end of communications).

READ -- Specify Reader Command: The READ command is used to specify a card reader from which further foreground initiation commands are to be read. The device specified must not be assigned to any other program.

Operation	Operand
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.

LISTIO -- List I/O Assignment Command: The LISTIO command is used to cause the system to print a listing of I/O assignments. The listing appears on the printer-keyboard (SYSLOG).

Operation	Operand
LISTIO	{ BG / F1 / F2 / UA / ALL }

BG Lists the physical units assigned to all background logical units.

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

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

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

ALL Lists the physical units assigned to all logical units.

ASSGN -- Assign Logical Name Command: The ASSGN command is used to assign a logical I/O unit to a physical device. All device assignments made for foreground programs are cancelled at the end of each program, unless held across jobs by the HOLD command. Except for DASD devices, a foreground program may not be assigned a device being used by another program.

Operation	Operand
ASSGN	SYSnnn,address [{,X'ss' / ,ALT}]

The entries in the operand field represent the following.

SYSnnn The symbolic unit name.

address Can be expressed as X'cuu' 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 - FE (0-254) in hexadecimal

IGN -- Indicates the logical unit is to be unassigned and that all program references to the logical device are to be ignored.

X'ss' Device specifications (used for specifying mode settings for seven track and dual density nine track tapes). If X'ss' is not specified, the system assumes X'90' for seven track tape and X'C0' for nine track tape. C0 is the normal reset mode for a 9-track tape unit, and specifies the maximum byte density for that device. C0 for a 9-track single density tape unit is 800 bpi, whereas for a dual density tape unit it is 1600 bpi. C8 is an alternate mode setting for 9-track dual density tapes only. 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	dual density 9-track		
C8	800	dual density 9-track		

Note that the first 15 entries in this table are valid only for 7-track tape. The last three entries are valid only for 9-track tape.

ALT Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The characteristics of the alternate unit must be the same as those of the original unit. Multiple alternates may be assigned to a symbolic unit.

VOL -- Volume Information Command: The VOL (volume) command is used when specifying a set of label information for a magnetic tape file or a DASD file. A VOL command

must be used for each file on a multifile volume.

Operation	Operand
VOL	SYSnnn,filename

SYSnnn — Symbolic unit name.

filename — File name. This can be one to seven characters and is identical to the symbolic address of the program DTF which identifies the file.

DLAB -- DASD Label Information Command: The DASD label command (completed on a continuation line) contains file label information for DASD label checking and creation. This statement must immediately follow the volume (VOL) command. The DLAB command and the continuation command have the following format.

Operation	Operand	
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.

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 Operating

System. It is recommended that this field be left blank.

type — Indicates the type of file label (SD, DA, ISC, or ISE). SD is assumed if this entry is omitted.

DTFSD or DTFPH with Mounted = single:
type = SD or blank

DTFDA or DTFPH with Mounted = ALL:
type = DA

DTFIS using Load Create:
type = ISC

DTFIS using other than Load Create:
type = ISE

XTENT -- DASD Extent Information Command:
The extent command defines each area, or extent, of a DASD file. One or more XTENT statements must follow each DLAB statement.

Operation	Operand
XTENT	type,sequence,lower,upper, 'serial no.',SYSxxx[,B_2]

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 is assumed to be $H_1H_2H_2$ 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 multi-extent 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
- 0 to 9 for 2321

C_1C_1 = Sub-cell number.
- 00 for 2311
- 00 to 19 for 2321

$C_2C_2C_2$ = cylinder number.
- 000 to 199 for 2311
- or
- strip number:
- 000 to 009 for 2321

H_1 = head block position.
- 0 for 2311
- 0 to 4 for 2321

H_2H_2 = head number.
- 00 to 09 for 2311
- 00 to 19 for 2321

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 that 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
0-9 for 2321

This field is optional. If missing, B_2=B_1 is assumed.

TPLAB -- Tape Label Information Command: The tape-label information command contains file label information for tape label checking and writing. This command must immediately follow the volume (VOL) command. The TPLAB command 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 TPLAB command may have either of the following two formats:

Operation	Operand
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.

CANCEL -- Cancel Initiation Command: The CANCEL command is used to cancel the initiation of a foreground program. It causes all previous foreground initiation commands to be reset and returns control to the Supervisor.

Operation	Operand
CANCEL	blank

EXEC -- Execute Program Command: The EXEC command is used to specify the foreground program to be executed. The program must be cataloged in the core image library of the system. It terminates the foreground initiation routines and causes the named foreground program to be loaded into main storage.

Operation	Operand
EXEC	progname

progname Represents the name of the program in the core image library to be executed. The program name can be one to eight alphameric characters.

When control is given to the foreground program, register 2 contains the address of the uppermost byte of storage available to the program.

LOG -- Log Command: The LOG command is used to cause the system to log columns 1-72 of all Job Control statements and foreground initiation commands on SYSLOG until a NOLOG command is sensed. Its format is as follows:

Operation	Operand
LOG	blank

The operand field is ignored by the system.

NOLOG -- Suppress Logging Command: The NOLOG command is used to cause the system to suppress the logging of all foreground initiation commands and all Job Control statements except JOB, PAUSE, *, and /& until a LOG command is sensed. Its format is as follows:

Operation	Operand
NOLOG	blank

The operand field is ignored by the system.

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

Operation	Operand
HOLD	{F1[,F2] / F2[,F1]}

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

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

RELSE -- Release Foreground Unit Assignments at EOJ Command: The RELSE command causes all I/O assignments for the foreground area(s) specified to be set to unassigned at the end of any job that is initiated for that area.

Operation	Operand
RELSE	{F1[,F2] / F2[,F1]}

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

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

Operation	Operand
UNA	{F1[,F2] / F3[,F1]}

The foreground area specified must be currently inactive. If only one foreground area is referenced, a previous HOLD for the other remains in effect. This command is intended to be used to free physical units currently assigned to a foreground area under the HOLD command.

MAP -- Map Main Storage Command: The MAP command is used to cause the system to print on SYSLOG the areas of main storage allocated to programs in a multiprogramming environment. It indicates what programs are being executed, and which has access to the interval timer. For a description of the map of main storage produced by this command, see ATTN Commands: MAP Command.

Operation	Operand
MAP	blank

MSG -- Transfer Control Command: The MSG command can be used to give control to a foreground program operator communications routine previously activated by a STXIT instruction.

Operation	Operand
MSG	{F1 / F2}

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

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

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

PAUSE -- Pause Command: The PAUSE command is used to cause Job Control processing to pause at the end of the current background program job step, or at the end of the current background program job. At that time, the printer-keyboard is unlocked for message input. The end-of-communications indication Ⓑ causes processing to continue.

Operation	Operand
PAUSE	(any user comment)

The operand of the PAUSE command is not processed by the system. It is used only for operator documentation.

TIMER -- Interval Timer Command: The TIMER command causes interval timer support to be given to the program specified.

Operation	Operand
TIMER	{BG / F1 / F2}

If interval timer support is already allocated to the program 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 cancellation of that program. Once established, timer support remains with an area from program to program until changed by a TIMER command.

Ⓑ -- End-of-Communications Command: When commands are being entered through a card reader (as a result of a READ command), and an invalid command is encountered, an error message is printed on the printer-keyboard. Further foreground initiation commands may then be read from the printer-keyboard. The end-of-communications command Ⓑ causes input reading to be switched back to the device specified in the READ command.

Operation	Operand
Ⓑ	blank

Ⓑ is the end-of-block character, alter code 5.

SYSTEM OPERATION WITHOUT A 1052

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

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

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

SYSTEM LOADER

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

FETCH Macro Instruction

This macro instruction has the format:

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

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

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

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

LOAD Macro Instruction

This macro instruction has the format:

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

This 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 can be specified either as symbols or in 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 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 provide a means of restarting at some midway point rather than at the beginning of the entire job, if processing must be terminated for any reason prior to the normal end of job. Any programmer logical unit (SYS000-SYS244) assigned to tape or 2311 can be used for recording checkpoints.

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 prior to the interruption. Therefore, the records contain everything needed to re-initialize the system when processing is restarted.

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

Only background programs may be checkpointed. Checkpoint records are written on either a 2311 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.

A detailed explanation of the CHKPT macro instruction will be found in the Supervisor and I/O Macros publication listed on the front cover.

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

NORMAL AND ABNORMAL END-OF-JOB HANDLING

When a background 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 and main storage from location 0 to the end of the background problem program area. For background programs, the printout is on SYSLST.

For foreground programs, the output unit for a storage printout is SYS000. This logical unit may be assigned to either a printer or a magnetic tape unit to obtain, in the event of an abnormal termination of a job, a printout of the Supervisor area, the appropriate foreground area, and the program registers. If SYS000 is not assigned to a printer or a non-file-protected magnetic tape unit, any printout specified by DUMP or PDUMP in the problem program is suppressed.

When a magnetic tape unit is used, there is no verification that the tape is suitable for writing (i.e., the tape is not opened). The storage print routine does not reposition the tape before writing. When the routine is completed, a tape mark is written and the tape is repositioned prior to the tape mark. 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 entitled CLOSE System Tape Output Files. Records written on SYS000 are 121 bytes in length, the first byte being an ASA control character.

The Job Control program provides job-to-job transition for background programs within the Disk Operating System. It also is 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:

1. The Initial Program Loader, to process the first job after an IPL procedure.

2. 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.

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

FUNCTIONS

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

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

Job Control clears the background program area in main storage (except the last 276 bytes) to binary zero between job steps.

The foreground-initiation routine performs for foreground programs functions similar to those performed by Job Control for background programs.

PREPARE PROGRAMS FOR EXECUTION

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

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

All foreground programs run in the system are loaded from the core image library. Thus, they will have been cataloged to the library by the Librarian.

SYMBOLIC INPUT/OUTPUT ASSIGNMENT

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

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

SYSRDR	Card reader, magnetic tape unit, or disk extent used for Job Control statements.

SYSIPT	Card reader, magnetic tape unit, or disk extent used as the input unit for programs.
SYSPCH	Card punch, magnetic tape unit, or disk extent used as the main unit for punched output.
SYSLST	Printer, magnetic tape unit, or disk extent used as the main unit for printed output.
SYSLOG	Printer-keyboard used for operator messages and to log Job Control statements. Can also be assigned to a printer.
SYSLNK	Disk extent used as input to the Linkage Editor.
SYSRES	System residence area on a disk drive.
SYS000-SYS244	All other units in the system.

Note that whenever a system logical unit (SYSRDR, SYSIPT, SYSLST, SYSPCH) is assigned to an extent of disk storage, the assignment must be permanent. Temporary assignments (via the // ASSGN statement or the ASSGN statement with the TEMP option) are not permitted.

The first seven of these names, termed system logical units, are used by the system control program and system service programs. Of these seven units, user background programs may also use SYSIPT 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 SYS244, 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 245.

For the convenience of the user, two additional names are defined for background program assignments. These names are valid parameters to Job Control only via the ASSGN, CLOSE, VOL, and XTENT statements described in this section. Reference within a program (such as in the CCB or a DTF) must name the particular logical unit to be used (SYSLST or SYSPCH, SYSRDR or SYSIPT). The additional names are as follows:

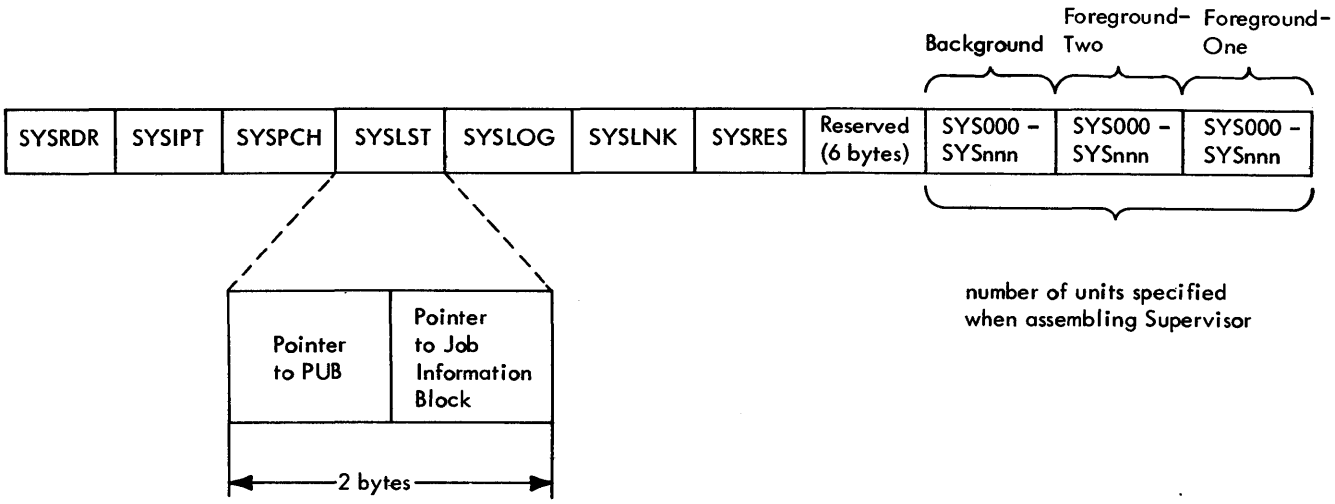
SYSIN	Name that can be used when SYSRDR and SYSIPT are assigned to the same card reader or magnetic tape unit. It may be either a temporary or a permanent assignment. This name must be used when SYSRDR and SYSIPT are assigned to the same disk extent, and may be only a permanent assignment.
SYSOUT	Name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

A tape or disk extent to be used as SYSIN can be prepared by using the IBM-supplied utility macros. Likewise, the IBM-supplied tape to printer/punch utility macros can be assembled and used to convert SYSOUT output into printed and punched card output. Examples of these two functions are shown in the utility macros publication.

With the exception of SYSLOG, foreground programs may not reference any system logical unit. (System units are reserved for the exclusive use of background programs operating in a stacked-job environment.) Foreground programs may reference any programmer logical unit, SYS000-SYSnnn.

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). The format of the device table is shown in Figure 5. The length of the table depends on the number of devices specified at system generation time. The system LUB's and the first ten programmer LUB's are always present.



●Figure 5. Sequence of LUB's in Device Table

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

Figure 6. Format of PUB Entry

A physical unit block (PUB) can be associated with each LUB. The format of a PUB is shown in Figure 6. The PUB's are ordered by priority within the channel to which the various devices are attached.

Normally, each symbolic name is assigned a physical device address at the time the Supervisor is assembled. In some cases, a single device may be assigned to two or more symbolic names. An installation can make specific assignments at system generation time and establish these as conventions to be followed by all programmers. By following the conventions, most background jobs can be submitted for execution with no ASSGN control statements. Figure 7, 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 is assigned to SYSLOG.
5. The two disk drives are addressed as SYSRES, SYSLNK, SYS001, SYS002, and SYS003. SYSLNK is used to assemble input to the Linkage Editor. SYS001, SYS002, and SYS003 are used by the language translators as work files. Language translators also can output Linkage Editor input on SYSLNK.
6. The two tape drives 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.

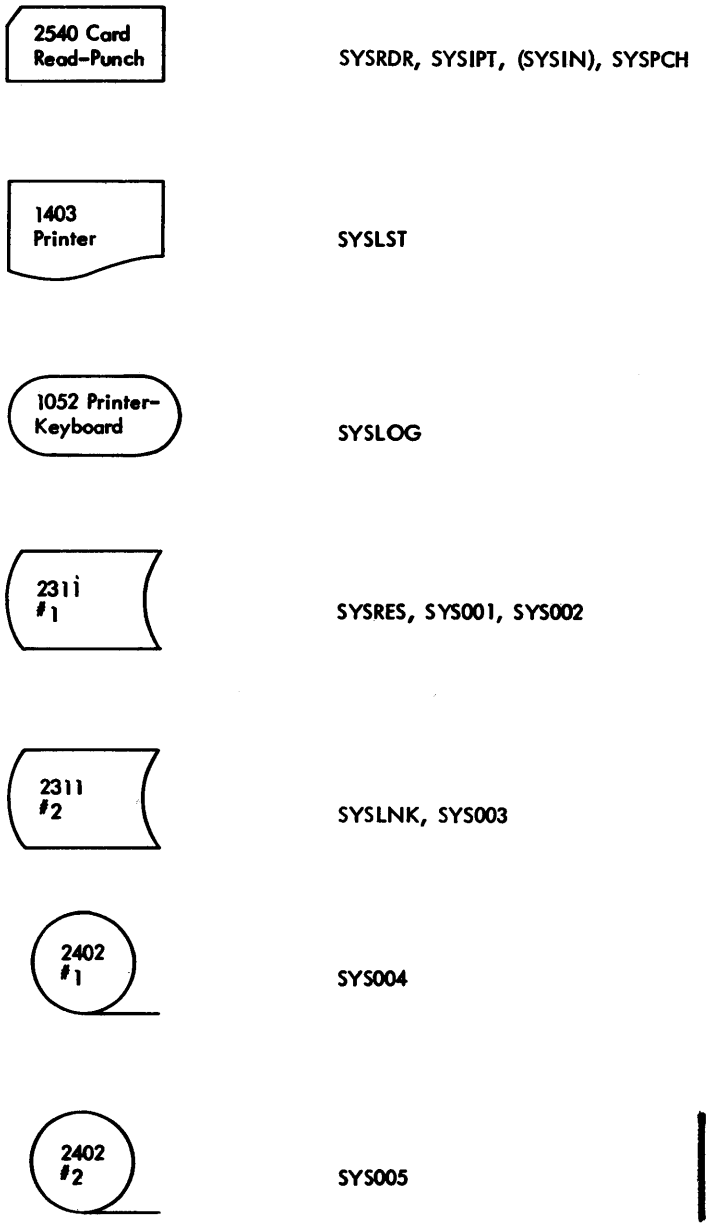


Figure 7. Example of Symbolic Device Assignment

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

SET UP COMMUNICATION REGION

Job Control takes the following information from control statements and places it in the communication region.

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

2. Job Date. Taken from the DATE statement. 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.

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 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	Standard label information
1-3	Background program label information
4-6	Foreground-two label information
7-9	Foreground-one label information

Track 0 of the label area can be used to establish extent information for the system logical unit SYSLNK, and the system component work files, SYS001, SYS002, and SYS003. This would eliminate the necessity of submitting VOL (volume information), DLAB (disk label information), and XTENT (extent information) statements each time a compilation or linkage editing function is performed. This facility can be implemented by including the operand STDLABEL in the Job Control OPTION statement. (A complete description of the Job Control statements VOL, DLAB, XTENT, and OPTION is contained in the following section.) Only label sets for sequential disk files or tape files may be written on track 0. Each time VOL, DLAB, and XTENT statements are submitted following // OPTION STDLABEL, track 0 on

the label information cylinder is completely overwritten. This facility is available to the user for any tape and sequential DASD files. The logical IOCS OPEN routine searches the standard label information area after searching the appropriate temporary label information area.

Self-relocating foreground programs may use the standard label information track, if an 80-byte area is reserved (by means of a DS statement) at the beginning of the program itself. Otherwise, label information statements must be submitted each time the program is to be executed.

Label blocks composed of information from VOL, DLAB, and XTENT statements are written on tracks 1-3 of the label information cylinder as temporary entries when:

1. The VOL, DLAB, and XTENT statements are not preceded by // OPTION STDLABEL, or

2. The VOL, DLAB, and XTENT statements are preceded by // OPTION USRLABEL.

Each set of label information submitted within a job or job step is written beginning with track 1 of the background program label information area. This information is not carried from job to job. Unless overwritten by a succeeding job step, any label information submitted at the beginning of a job may 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 may 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.

The formats of the label information statements are discussed in the following section. See the Data Management Concepts publication for a complete discussion of volume and file labels.

RESTARTING PROGRAMS FROM CHECKPOINT

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

JOB CONTROL STATEMENTS

GENERAL CONTROL STATEMENT FORMAT

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

1. Name. Two slashes (//) identify the statement as a control statement. They must be in columns 1 and 2. At least one blank immediately follows the second slash. Exception: The end-of-job statement contains /& in columns 1 and 2, the end-of-data-file statement contains /* in columns 1 and 2, and the comment statement contains * in column 1 and blank in column 2.

2. Operation. This describes the type of control statement (the operation to be performed). It can be up to eight characters long. At least one blank follows its last character.

3. Operand. This may be blank or may contain one or more entries separated by commas. The last term must be followed by a blank, unless its last character is in column 71.

All control statements are essentially free form. Information starts in column 1 and cannot extend past column 71. Exception: For the file label statements (TPLAB and DLAB) information may not be contained entirely in one card image. Any non-blank character present in column 72 specifies that information is continued in the following card image (continuation statement). Information in the continuation statement begins in column 16; columns 1-15 are ignored.

Job Control continuation statements are used only for the label statements (other statements are not checked for this possibility).

Job Control reads from the device identified by the symbolic name SYSRDR. The following statements are recognized:

Operation	Meaning
JOB	Job name
EXEC	Execute program
ASSGN	I/O assignments
RESET	Reset I/O assignments
DATE	Date
UPSI	User program switch indicators
LBLTYP	Reserve storage for label information
VOL	Volume information
DLAB	Disk file label information
XTENT	Disk file extent
TPLAB	Tape file label information
RSTRT	Restart
LISTIO	List I/O assignments
MTC	Magnetic Tape Control
OPTION	Option
PAUSE	Pause
/*	End of data file
/&	End of job
*	Comment

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:

1. Invalid symbolic unit name.
2. No space reserved in LUB table for a symbolic unit.
3. Invalid device-type.
4. Invalid length of field.
5. Invalid character.
6. Missing /& statement.
7. A volume (VOL) statement does not precede a label (DLAB or TPLAB) statement.
8. An extent (XTENT) statement does not immediately follow its associated disk label (DLAB) statement.

Note that 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 below for the label information statements. The following statements can precede the EXEC statement for a job step.

ASSGN
RESET
DATE
UPSI
LBLTYP
VOL
DLAB
XTENT
TPLAB
LISTIO
MTC
OPTION
PAUSE
*

The label statements must be in the order:

```
VOL     or  VOL
TPLAB       DLAB
            XTENT (one for each area
                   of file in volume)
```

and, must immediately precede the EXEC statement to which they apply.

The LBLTYP statement is used at Linkage Editor time and must precede the // EXEC LNKEDT statement with the exception of self-relocating background programs, for which it is instead submitted immediately preceding the // EXEC statement for the program. Self-relocating foreground programs do not use the LBLTYP statement.

DESCRIPTION AND FORMAT OF JOB CONTROL STATEMENTS

JOB Statement

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

```
// JOB jobname
```

jobname — The name of the job. Must be one to eight alphameric characters. When restarting a job, the jobname must be identical to that used when the checkpoint was taken. Any user comments can appear on the JOB statement following the job

name (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.

EXEC Statement

The EXEC (execute) control statement must be the last statement processed before a job step is executed. It indicates the end of job control statements for a job step and that execution of a program is to begin. Its format is:

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

progname — Represents the name of the program in the core image library to be executed. The program name can be one to eight alphameric characters. If the program to be executed has just been processed by the Linkage Editor, the operand of the EXEC statement is blank. When control is given to a fetched phase, general register 2 contains the address of the uppermost byte of storage available to the program.

ASSGN Statement

When programs are assembled, they use symbolic names to reference I/O devices. At execution time this statement is used to assign a specific device address to the symbolic unit name used. It contains the symbolic unit name and various parameters to describe the physical device. The format is:

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

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

SYSRDR
SYSIPT
SYSIN
SYSPCH
SYSLST
SYSLOG
SYSLNK
SYS000-SYS244

Note that assignments for SYSOUT must be permanent; that is, not reset between jobs. For this reason, it is not included in the preceding listing.

For 1419 Magnetic Character Readers with the dual-address adapter, two assignments are required for each reader/sorter if pocket light operations are to be performed. If no pocket light operations are to be performed, the user need only assign a logical unit to the primary control unit address (1419P).

deviceaddress — Can be expressed as X'cuu', UA, or IGN.

X'cuu' -- 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 cancellation of the job.

IGN -- indicates the logical unit is to be unassigned and that all program references to the logical device are to be ignored. The IGN option is not valid for SYSRDR and SYSIPT.

X'ss' — Device specifications (used for specifying mode settings for 7-track and dual density 9-track tapes). If X'ss' is not specified, 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. C0 for a 9-track single density tape unit is 800 bpi; whereas for a dual density tape unit, it is 1600 bpi. C8 is an alternate mode setting for 9-track dual density tapes only. 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	dual density 9-track		
C8	800	dual density 9-track		

Note that the first 15 entries in this table are valid only for 7-track tape. The last three entries are valid only for 9-track tape.

ALT — Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. The characteristics of the alternate unit must be the same as those of the original unit. Multiple alternates may be assigned to a symbolic unit.

All device assignments made with ASSGN control statements are reset between jobs to the configuration specified when the system was generated plus any modifications that may have been made by the operator at IPL time and between jobs or job steps.

RESET Statement

The RESET statement resets I/O assignments to the standard assignments. The standard assignments are those specified when the system was generated plus any modifications made by the operator via an ASSGN command (as opposed to an ASSGN control statement) without the TEMP option. Its format is:

```
// RESET { SYS    }
         { 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.

DATE Statement

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

UPSI Statement

This 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. The UPSI statement has the following format.

```
// UPSI nnnnnnnn
```

The operand consists of one to eight characters of 0, 1, or X. Positions containing 0 will be set to 0. Positions containing 1 will be set to 1. Positions containing X will be 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 may be tested by problem programs at execution time.

VOL Statement

The volume statement is used when checking standard labels for a DASD or tape file. A VOL statement must be used for each file on a multifile volume. Its format is:

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

SYSxxx — Symbolic unit name (present for compatibility with the Tape Operating System). The symbolic unit name is taken from the XTENT statement.

filename — File name. This can be one to seven characters and is identical to the symbolic address of the program DTF which identifies the file.

DLAB Statement

The DASD-label statement (completed in a continuation statement) contains file label information for DASD-label checking and creation. This statement must immediately follow the volume (VOL) statement. The DLAB statement and the continuation statement have the following 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 Indicates the type of file label (SD, DA, ISC, or ISE). SD is assumed if this entry is omitted.

DTFSD or DTFPH with Mounted = Single:
  type = SD or blank

DTFDA or DTFPH with Mounted = ALL:
  type = DA

DTFIS using Load Create:
  type = ISC

DTFIS using other than Load Create:
  type = ISE

XTENT Statement

The extent statement defines each area, or extent, of a DASD file. One or more XTENT statements must follow each DLAB statement. The XTENT statement has the following format.

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

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 is assumed to be $H_1H_2H_2$ 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 multi-extent 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
0 to 9 for 2321

C_1C_1 = Sub-cell number.

00 for 2311
00 to 19 for 2321

$C_2C_2C_2$ = cylinder number.

000 to 199 for 2311

or

strip number:

000 to 009 for 2321

H_1 = head block position.

0 for 2311
0 to 4 for 2321

H_2H_2 = head number.

00 to 09 for 2311
00 to 19 for 2321

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 that 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
0-9 for 2321

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

TPLAB Statement

The tape-label statement contains file label information for tape label checking and writing. This statement must immediately follow the volume (VOL) statement. The TPLAB 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 contained 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 TPLAB statement may have either of the following two formats:

1. // TPLAB 'label fields 3-10'

2. // TPLAB 'label fields 3-13'

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

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

LBLTYP Statement

The LBLTYP statement is used to define the amount of main storage to be reserved at linkage-edit time for processing of tape and non-sequential disk file labels in the problem program 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 background programs, for which it is instead submitted immediately preceding the // EXEC statement for the program. Self-relocating foreground programs do not use the LBLTYP statement. It has the following 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 file types to be used. nn specifies the largest number of extents to be used for a single file.

The LBLTYP statement should not be used if no labels are to be used, or if only SD-type disk files are to be processed.

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.

RSTRT Statement

A restart facility is available for checkpointed programs. A programmer can use the CHKPT macro instruction in his program to cause checkpoint records to be written. This allows sufficient 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. The DLAB and TPLAB statements used to restart must have the volume sequence number updated to agree with the volume sequence numbers of the units being restarted. The format of the RSTRT statement follows:

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

See the Supervisor and I/O Macros publication 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.

LISTIO Statement

This statement is used to get a listing of I/O assignments. Its format is:

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

The listing is output on the device assigned to SYSLST. The listing varies according to the operand used.

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

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

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

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

ALL — Lists the physical units assigned to all logical units.

SYSxxx — Lists the physical units assigned to the logical unit specified.

UNITS — Lists the logical units assigned to all physical units.

DOWN — Lists all physical units specified as inoperative.

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

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

Physical units are listed with current device specification for magnetic tape units. Logical units are listed with ownership (background, foreground-one, or foreground-two), when applicable. An example of a listing produced by the LISTIO SYS statement is shown in Figure 8. All channel and unit numbers are represented in hexadecimal.

```
LISTIO SYS

I/O UNITS CH. UNIT

*** SYSTEM ***
 SYSRES      1   00
 SYSLOG      0   1F
 SYSRDR      0   04
 SYSIPT      2   01
 SYSLST      2   02
 SYSPCH      IGN
 SYSLNK      UA
```

Figure 8. Example of LISTIO SYS Output

MTC Statement

The MTC statement is used to control operations on logical units assigned to magnetic tapes. Its format is:

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

The first entry in the operand field specifies the operation to be performed. It can be:

Opcode	Meaning
BSF	Backspace file
BSR	Backspace record
ERG	Erase gap
FSF	Forward space file
FSR	Forward space record
REW	Rewind
RUN	Rewind and unload
WTM	Write tape mark

The second entry, SYSxxx, represents any logical unit. The optional third entry, nn, is a decimal number 01 to 99 representing the number of times the specified operation is to be performed.

OPTION Statement

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

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

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

LOG	Causes the listing of columns 1-80 of all control statements on SYSLST. Control statements are not listed until a LOG option is encountered. Once a LOG option statement is read, logging continues from job-step to job-step until a NOLOG option is encountered or until either the JOB or /& control statement is encountered.
NOLOG	Suppresses the listing of all control statements on SYSLST except JOB and /& statements until a LOG option is encountered.
DUMP	Causes a dump of the registers and main storage to be output on SYSLST 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.)
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. An EXEC statement with a blank operand also suppresses the LINK option.
DECK	Causes language translators to output object modules on SYSPCH. If LINK is specified, the DECK option is ignored.
NODECK	Suppresses the DECK option.
LIST	Causes language translators to write the source module listing on SYSLST. Also, the Assembler writes on SYSLST the hexadecimal object module listing and a summary of all errors in the source program.
NOLIST	Suppresses the LIST option.
LISTX	Causes the COBOL compiler to output a PROCEDURE DIVISION MAP on SYSLST; causes the PL/I compiler to output the object module on SYSLST.
NOLISTX	Suppresses the LISTX option.
SYM	Causes the Assembler to output the symbol table on SYSPCH; causes the PL/I compiler to output the symbol table on SYSLST; causes the COBOL compiler to output a DATA DIVISION map on SYSLST.
NOSYM	Suppresses the SYM option.
XREF	Causes the Assembler and PL/I compiler to write the symbolic cross-reference list on SYSLST.

NOXREF	Suppresses the XREF option.
ERRS	Causes the FORTRAN, COBOL, and PL/I 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.
STDLABEL	Causes all sequential disk 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.
USRLABEL	Causes all sequential disk or tape labels submitted after this point to be written at the beginning of the user label track.
48C	Specifies the 48-character set on SYSIPT (for PL/I).
60C	Specifies the 60-character set on SYSIPT (for PL/I).

The options specified in the OPTION statement remain in effect until a contrary option is encountered or until a JOB control statement or a /& 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 cancelled after each occurrence of an EXEC statement with a blank operand.

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

PAUSE Statement

This statement has the following format.

```
// PAUSE [comments]
```

comments Optional. Any message to the operator can appear in the operand of the PAUSE statement.

The statement can be used to allow for operator action between job steps.

The printer-keyboard is unlocked for operator-message input. The end-of-communications indication, Ⓑ, causes processing to continue. The PAUSE statement is always printed on SYSLOG. If no 1052 is available, the PAUSE statement is ignored.

/* -- End of Data File Statement

This statement must be the last statement of each input data file on SYSRDR and SYSIPT. Its format is:

```
/* 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-SYS244.

/& -- End of Job Statement

This statement must be the last statement of each job. Its format is:

```
/& ignored
```

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 /&, the job is terminated. Any comments may begin in column 14.

The end-of-job statement is printed on SYSLOG and SYSLST in the following format. Columns 1-3 contain EOJ. Columns 5-12 contain the job name. Columns 14-72 contain blanks or any user comments. If the timer feature is present, columns 73-80 contain the time of day on SYSLST; the time of day is printed in columns 1-8 on the next line of SYSLOG.

* -- Comments Statement

This statement can be used as a Job Control comments statement. Its format is:

```
* 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, and the opening and closing of magnetic tape and disk devices when assigned to three programmer logical units (SYS001 - SYS003), used by the various system components.

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 and SYSIPT records must be 80 characters in length, SYSLST records are 121 characters, and SYSPCH records are 81 characters in length. 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 disk have no keys, and record lengths are the same as stated above.

OPEN System Tape Files

When the system logical unit SYSRDR, SYSIPT, SYSPCH, or SYSLST is assigned, Job Control checks to see whether the device assignment is to a magnetic tape device positioned at load point. If so, Job Control performs an OPEN having the following characteristics.

1. Accepts a leading tape mark for input or output.
2. Accepts a completely unlabeled tape (other than blank tape) for input. Non-standard labels are not recognized.
3. Accepts any label set starting with VOL1 (including user labels) for input.
4. Accepts a valid IBM-standard label set (including user labels) having a past expiration date for output.
5. Rejects all other volumes and gives the operator the option of either accepting the invalid label (IGNORE) or mounting an acceptable volume and replying RETRY.

CLOSE System Tape Output Files

When SYSPCH, SYSLST, or the combined output file, SYSOUT, is assigned to magnetic tape, the system will process 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 file mark, an EOV trailer record, and two file marks 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 had not been assigned, or if all assigned alternates were currently active as other system files, a message will be printed on SYSLOG requesting the operator to mount a new reel. The system will automatically continue 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 may be assigned to an extent of 2311 disk storage.

If both SYSRDR and SYSIPT are to be assigned to disk, they must be assigned to the same extent and be referred to as SYSIN. SYSLST and SYSPCH must be assigned to separate extents. Thus, SYSOUT may not be used to refer to a combined SYSLST/SYSPCH on 2311 disk.

The assignment of system logical units to extents of disk storage must be permanent. The operator ASSGN statement 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 the capability of 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 of this manual.) Note that this warning is given after the job ends, and that if the extent limits are exceeded before end-of-job, the job is terminated.

Assigning System Files to Disk

System input and output files are assigned to disk by providing a set of VOL, DLAB, and XTENT statements and then submitting a permanent ASSGN statement. The set of VOL, DLAB, and XTENT statements preceding the ASSGN statement may contain only one VOL statement and one XTENT statement.

The VOL statement must be one of the following:

1. // VOL SYSRDR,IJSYSIN (for SYSRDR).
2. // VOL SYSIPT,IJSYSIN (for SYSIPT).
3. // VOL SYSIN,IJSYSIN (combined SYSRDR/SYSIPT file)
4. // VOL SYSPCH,IJSYSPH (for SYSPCH)
5. // VOL SYSLST,IJSYSLS (for SYSLST)

Note that a combined SYSPCH/SYSLST file (SYSOUT) may not be assigned to disk.

In the DLAB statement, the sixth operand (filetype parameter) must specify SD (or blank, which means SD) to indicate sequential DASD file type.

In the XTENT statement, type may be 1 (data area, no split cylinder) or 128 (data area, split cylinder), and SYSxxx must correspond with the name in the ASSGN statement. There is no unique requirement for the remaining operands of the XTENT statement.

The ASSGN statement 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).

Note that all must be permanent assignments.

OPEN System Disk Files

Upon encountering a system input or output assignment to 2311, Job Control performs the following functions:

1. Rejects the assignment if it is not permanent.
2. Rejects the assignment if a previous assignment to 2311 for the same logical unit still exists (has not been closed). Also, because SYSRDR and SYSIPT must be a single combined file if both are on disk, one cannot be assigned to disk if an assignment to disk for the other (or both) already exists.
3. OPENs the file. If input, the labels are checked. If output, DASD labels are written. Also, information is placed into the Supervisor for the problem program OPEN, and for monitoring of file operations by physical IOCS.
4. If the OPEN is unsuccessful, Job Control unassigns the unit and requests further operator commands.

CLOSE System Disk Files

System logical units assigned to disk must be closed by the operator. The operator CLOSE command may be used to specify a system input or output file which has been previously assigned to a 2311. The optional second parameter (X'cuu') of the CLOSE command may be used (instead of an ASSGN command) to assign the system logical unit to a physical device. The system will notify the operator that a CLOSE is required when the limit of the file has been exhausted. If a program attempts to

read or write beyond the limits of the file, the program will be terminated and the file must be closed.

The CLOSE function:

1. Writes a file mark if the file is an output file.
2. Resets the I/O control table in the Supervisor to indicate that the file no longer exists.
3. Reassigns the logical unit to the value of the second operand of the CLOSE command.

CONTROL STATEMENT EFFECT ON I/O UNITS

Certain control statements in the Job Control input stream affect the use of system I/O units by Job Control. These statements are as follows:

// JOB. This statement must be the first statement of the job. When the JOB statement is encountered, the content of the statement is printed on SYSLOG and SYSLST. 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, a carriage eject character is prefixed to the card image, which is then written on tape.

/&. When Job Control encounters /& on SYSRDR during normal operation, the standard assignment for SYSIPT becomes effective and SYSIPT is checked for an end-of-file condition. If the standard assignments for SYSRDR and SYSIPT are not to the same device, SYSIPT is advanced to the next /& statement. The end-of-job statement is written on SYSLOG and the last line of SYSLST.

In the event of an abnormal termination, Job Control advances SYSRDR and SYSIPT to the next /&, and proceeds as above, only if a // JOB statement was provided.

The user should beware of omitting /&, for protection of one job from errors in the preceding job cannot then be guaranteed. The // JOB statement is required in all cases.

Job Control has no responsibility for the arrangement of output on any file, except that connected with control statements. Such items as page ejection and line count must be managed by the individual processing program. The first line printed on SYSLST must be preceded by a line skip or a page eject. Otherwise, an overprint may result.

WORK FILES USED BY SYSTEM COMPONENTS

The system logical unit SYSLNK and the three programmer logical units SYS001, SYS002, and SYS003 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 disk storage. SYS001, SYS002, and SYS003 can be assigned to either disk or tape. When SYS003 is assigned to disk, the assignment may be the same as that for SYSLNK; however, when the "compile-and-execute" mode of operation is desired, SYSLNK and SYS003 must be assigned to separate files.

Disk Work Files

The programmer must provide ASSGN, VOL, DLAB, and XTENT information for each file (SYSLNK and SYS001-SYS003) to be used by the system components. This 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, and SYS003 are opened and closed by each component that uses these files as work files. The filenames for SYSLNK and SYS001-SYS003 on the VOL statement are IJSYSLN, IJSYS01, IJSYS02, and IJSYS03 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.

Note that 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 99365. 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.

Tape Work Files

The work files SYS001, SYS002, and SYS003 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 9 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. Note that the PHASE, INCLUDE, and ENTRY statements are Linkage Editor control statements. These statements are described in detail in the section Linkage Editor. They are included in this discussion to present a more meaningful example.

1. JOB statement for the series of job steps to be performed.

2. ASSGN statements required for the job steps. It is assumed that the assignments differ from those specified when the system was generated. The new assignments will be carried through for the entire job and will be reassigned at the end of the job to the standards established at system generation time.

3. OPTION statement specifying that the output of the 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 FORTRAN compilation, followed by the 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 FORTRAN and Assembler object programs on SYSLNK and writes the edited program temporarily in the core image library.

7. EXEC statement for the linkage-edited object program in the core image library. The input data for the execution is followed by the end-of-data-file indicator.

8. PAUSE statement that requests operator action.

9. OPTION statement specifying that the no-dump option be exercised. The link option must be included to enable a new linkage-edit. (An EXEC statement with a blank operand suppresses the LINK option.)

10. INCLUDE statements for modules in the relocatable library that are to be included with the object deck on SYSIPT. A blank operand indicates that the program to be included follows on SYSIPT. The resulting program is edited and written in the core image library.

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

12. PAUSE statement that requests operator action.

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

14. JOB statement for the next job.

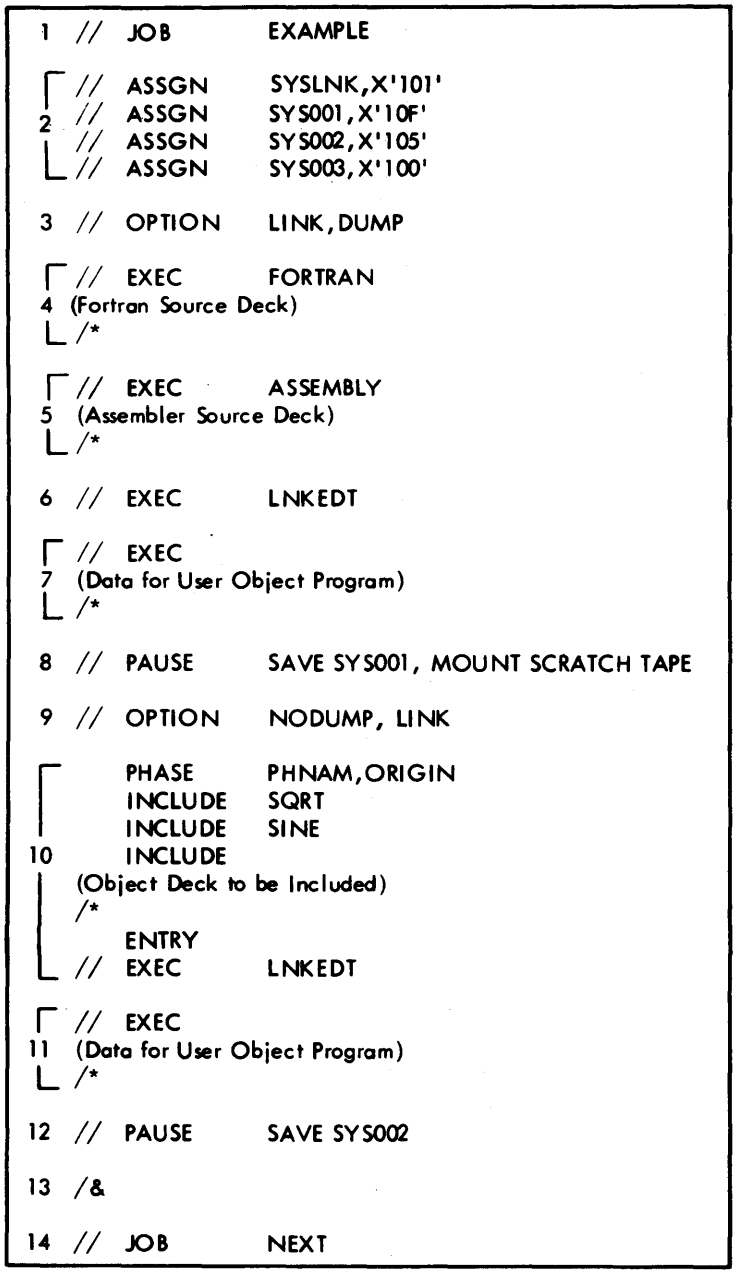


Figure 9. Job Control Statement Example

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 CCW's, which in turn cause the reading and loading of the IPL.

Operating in the supervisor state, IPL reads the Supervisor nucleus into low main storage. If a read error is sensed while reading the Supervisor nucleus, the wait state is entered and an error code is set in the first word of main storage. The IPL procedure must then be restarted.

After successfully reading in the Supervisor nucleus, IPL performs these operations.

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

2. Places the processing unit in wait state with all interruptions enabled. When the wait state is entered, the operator decides whether a 1052 or a card reader will be used to communicate with the system. If a 1052, the request key is pressed; if a card reader that is not assigned as SYSRDR at system generation time, it is brought to the ready state; if a card reader that is assigned as SYSRDR at system generation time, the interrupt key on the console is pressed.

3. At this time, the operator has several options. He may change the PUB configuration 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 has been 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 TEB's are specified as a system generation parameter.

To add a device to the PUB table, a control statement, read by the communication device (1052 or card reader), in the following format is required.

Operation	Operand
ADD	X'cuu'[(k)],devicetype[,X'ss']

where:

X'cuu' = channel and unit numbers.

k = S if the device is to be switchable (the device is physically attached to two adjacent channels). The designated channel is the lower of the two channels. If the device is not switchable, k = 0-255, indicating the priority of the device, with 0 indicating the highest priority. If k is not given, the assumed priority is 255.

devicetype = actual device (2400T9, 1443, etc). See device codes in Figure 9.1.

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 is used to specify SADxxx (Set ADdress) requirements for IBM 2702 lines:
X'00' for SAD0
X'01' for SAD1
X'02' for SAD2
X'03' for SAD3
This information is not accepted on the ASSGN statements.

X'ss' is required for 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

To delete a device from the PUB table, a control statement, read by the communication device (1052 or card reader), in the following format is required.

Operating	Operand
DEL	X'cuu'

where cuu is the channel and unit numbers of the device to be deleted.

Note that ADD and DEL statements are issued only at IPL time.

The only communication required at IPL time is the date and, if the timer is present, the time of day. It must follow any ADD or DEL statements. It is entered via the communication device (1052 or card reader) and is in the following format.

Operation	Operand
SET	DATE=value1[,CLOCK=value2]

value1 Has one of the following formats.

mm/dd/yy
dd/mm/yy

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

value2 Has the 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, 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, ASSGN statements are entered via the communication device (1052 or card reader). The ASSGN statements are as described in Job Control Commands.

Card Code	Actual Device	Device Type
2400T9	Nine Track Magnetic Tapes	Tapes
2400T7	Seven Track Magnetic Tapes	
1442N1	1442N1 Card Read Punch	Card Readers - Punches
2520B1	2520B1 Card Read Punch	
2501	2501 Card Reader	Card Readers
2540R	2540 Card Reader	
2540P	2540 Card Punch	Card Punches
2520B2	2520B2 Card Punch	
1442N2	1442N2 Card Punch	
2520B3	2520B3 Card Punch	
1403	1403 Printer	Printers
1403U	1403 Printer with UCS Feature	
1404	1404 Printer	
1443	1443 Printer	
1445	1445 Printer	
1050A	1052 Printer- Keyboard	1050 Control Unit
UNSP	Unsupported Device	Unsupported. No Burst Mode on Multiplexor Channel
UNSPB	Unsupported Device	Unsupported with Burst Mode on Multiplexor Channel
2260 (Local)	2260 Display Unit	Display Unit
2260 (Local)	A 1053 attached to a 2848. The mode operand must be entered as MODE=X'01'	Printer
2311	2311 Disk Drive	DASD
2321	2321 Data Cell Drive	DASD
2671	2671 Paper Tape Reader	Paper Tape Reader
2701	2701 Line Adapter Unit	Teleprocessing Lines
2702	2702 Transmission Control Unit	
2703	2703 Transmission Control Unit	
7770	7770 Audio Response Unit	Audio Response Unit
7772	7772 Audio Response Unit	
1285	1285 Optical Reader	Optical Readers
1287	1287 Optical Reader	
1412	1412 Magnetic Ink Character Reader	MICR Devices
1419	Single Address Adapter 1419 Magnetic Ink Character Reader	
1419P	Primary Address, Dual Address Adapter 1419 Magnetic Ink Character Reader	
1419S	Secondary Address, Dual Address Adapter 1419 Magnetic Ink Character Reader	

Figure 9.1. Device Code Entries for Devicetype Parameter in ADD Statement.

All programs executed in the Disk Operating System environment must be edited by the Linkage Editor. The Linkage Editor reads the relocatable output of the language translators and edits it into executable, nonrelocatable programs in the core image library. The linkage editor performs on one program at a time; that is, it cannot linkage edit a series of programs concurrently. Once a program has been edited, it can be executed immediately, or it can be cataloged as a permanent entry in the core image library. When a program has been cataloged in the core image library, the Linkage Editor is no longer required for that program. The program is run as a distinct job step and is loaded directly from the resident pack by the System Loader.

The extent of the editing function performed depends on the structure of the input program. The simplest case is that of a single-module program. The Linkage Editor has only to edit the program, creating a single phase entry in the core image format. This corresponds to the first diagram in Figure 10.

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 10). 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).

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), 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. Note: 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 is 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 to be discussed later) 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:

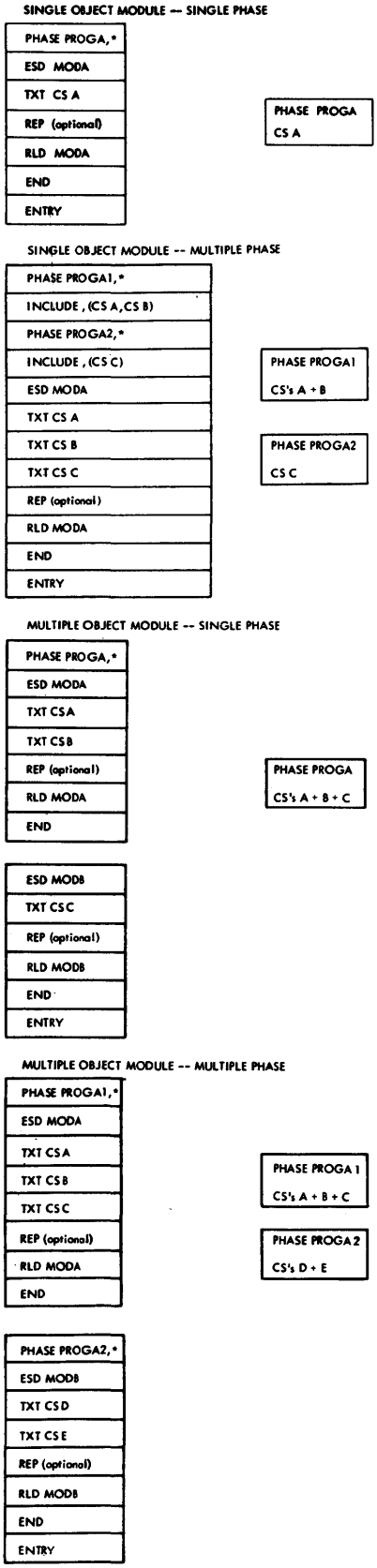


Figure 10. Linkage Editor Input and Output

Identification	Contents or Meaning
ESD	External Symbol Dictionary
TXT	Text
RLD	Relocation List Dictionary
REP	Replacement Text
END	End of a Module

With the exception of REP cards, all these cards must be present in each object module in the indicated sequence. REP cards, when required, appear after the TXT cards and before the END card and are produced by the programmer. The format of each of these card types is shown in Appendix E.

ESD (External Symbol Dictionary): The external symbol dictionary contains all the symbol and storage assignments for an object module. For example, it contains all symbols defined in this module that are referred to by some other module. It can contain all symbols referred to by this module that are defined in some other module.

The five classifications of the ESD record recognized by the Linkage Editor are:

1. SD (Section Definition): Specifies the symbolic name, the assembled origin, and the length of the control section. SD is generated by a named START or named CSECT in a source module.

2. LD (Label Definition): 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.

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. 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.

5. CM (COMMON): Indicates the amount of main storage that must be allocated by this module for a COMMON area to be shared between phases. CM is generated by COM in a source module.

TXT (Text): The program that is eventually loaded into storage for execution is contained within the TXT cards. The text card contains the assembled origin of the instructions or data included in the card, and also the count of the number of bytes contained in the card. This card includes a reference to the control section in which this information occurs that allows the relocation factor involved to be applied. TXT information will be modified as required by RLD information.

RLD (Relocation List Dictionary): The relocation list dictionary cards identify portions of the TXT that must be modified due to relocation. They provide information necessary to perform the relocation. RLD cards are generated by relocatable address constants in a source module.

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

REP (User Replace Card): The REP card allows replacement (substitution) of new text for portions of assembled text. Each REP card contains the assembled address of the first byte to be replaced, the identification of the control section to which it refers, and may contain from 2 to 22 bytes of text. The text is substituted, byte for byte, for the original text, beginning at the address specified. The address, the control section reference, and the new text must be stated in hexadecimal. The REP card must be placed within the module that it modifies.

Program Phase

A program phase, the output of the Linkage Editor, is that section of a program that is loaded as a single overlay with a single FETCH or LOAD by the System Loader. Programs may consist of many phases, the first fetched by Job Control, and each of the rest by a preceding program phase. Successive phases of a multiphase program are often called overlays.

The input for building a single phase consists of the text from one or more complete control sections. When building a phase, the Linkage Editor constructs composite ESD and composite PHASE data, known as

the control dictionary, and a composite RLD from each of the modules that make up the phase. These composite dictionaries are used to resolve all linkages between different control sections as if they had been assembled as one module. Each control section within the phase is relocated as necessary, and the entire phase is assigned a contiguous area of main storage. All relocatable address constants are modified to contain the relocated value of their symbols. The Linkage Editor always ensures that each phase or control section begins on a double-word boundary.

Each phase is constructed by building the text in the core image format. Thus, a phase may consist of one or more blocks of contiguous core image locations. Backward origin within a phase may cause slower operation of the Linkage Editor.

TYPES OF LINKAGE EDITOR RUNS

The Linkage Editor is run as a distinct job step. Because of this fact, it is meaningful to classify it as one of the system service programs. The Linkage Editor function is performed as a job step in three kinds of operations (Figure 11).

1. Catalog Programs in Core Image Library. The Linkage Editor function is performed immediately preceding the operation that catalogs programs into the core image library. By specifying the CATAL option, the Linkage Editor not only edits the programs, but also catalogs them permanently in the core image library. The sequence of events when this operation is performed is shown in the first diagram of Figure 11. Note that the input for the LNKEDT function could include modules from the 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. The sequence of events when this operation is performed is shown in the second diagram in Figure 11. 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

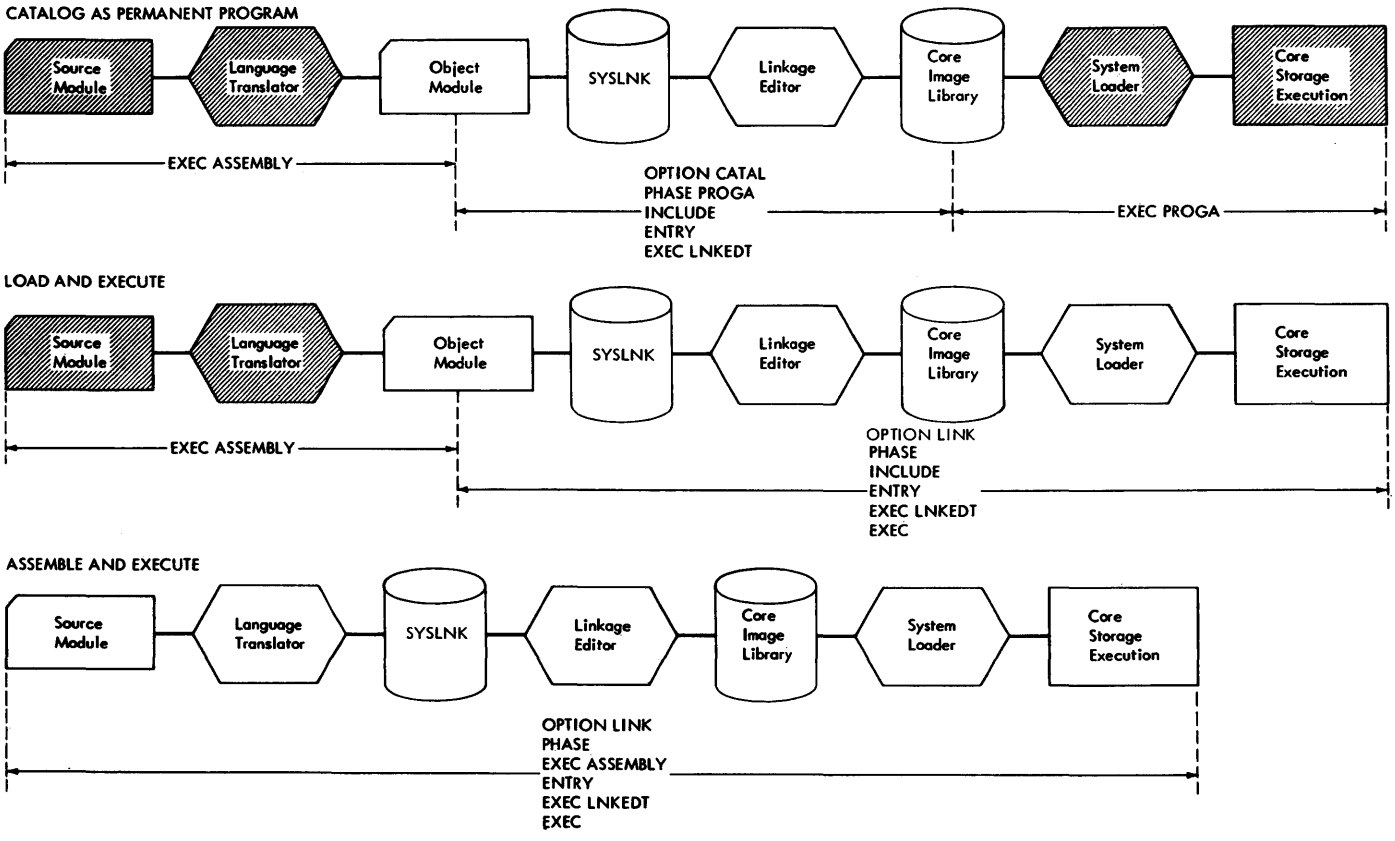


Figure 11. Three Types of Linkage Editor Operations

catalog operation, the input can consist of object modules from the relocatable library instead of, or in addition to, those modules from the card reader, tape unit, or disk extent assigned to SYSIPT. This is accomplished by including the name of the module to be included in the operand of an INCLUDE statement. After the object modules have been edited and placed in the core image library, the program is executed. The blank operand in the EXEC control statement indicates that the program just linkage-edited and temporarily stored in the core image library is to be executed.

3. Assemble- or Compile-and-Execute. Source modules can be assembled or compiled and then executed in a single sequence of job steps. In order to do this, the language translator is directed to output the object module directly in SYSLNK. This is done by using the LINK option in the OPTION control statement. Upon completion of this output operation, the Linkage Editor function is performed. The program is linkage edited and temporarily stored in the core image library. The sequence of events when this operation is performed is shown in the third diagram in Figure 11.

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 — To indicate 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 AUTOLINK feature.

INCLUDE — To signal 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 — To provide an optional transfer address for the first phase.

ACTION — To specify options to be taken.

The first (or only) object module input for the Linkage Editor should include a PHASE control statement before the first ESD item. If no PHASE statement is used, or if the PHASE statement is in error, the Linkage Editor will construct a dummy statement. This will allow the user to test the program; however, the program with the dummy PHASE statement cannot be cataloged in the core image library. The last (or only) object module may optionally be followed by an ENTRY control statement. The rules governing placement of INCLUDE and other PHASE control statements are discussed under Control Statement Placement.

SOURCES OF INPUT

Input for the Linkage Editor always begins from the area in disk assigned to SYSLNK. Object module input to the Linkage Editor can be:

1. Output from the language translator programs immediately after a compilation or assembly.
2. From the card reader, tape unit, or disk extent assigned to SYSIPT.
3. From SYSIPT and from the relocatable library.
4. From the relocatable library.

In the first case, the LINK option of the OPTION control statement indicates to Job Control and to language translators that the output resulting from an assembly or compilation will be written directly on SYSLNK.

In the second case, an INCLUDE statement with a blank operand indicates to Job Control that the input on SYSIPT up to the /* statement is to be copied on SYSLNK.

In the third case, an INCLUDE statement with a blank operand indicates to Job Control the presence of input on SYSIPT. An INCLUDE statement with an entry in the operand indicates that a module in the relocatable library is to be included in the program phase by the Linkage Editor.

In the fourth case, an INCLUDE statement with an entry in the operand indicates that a module in the relocatable library is to be included in the program phase. Modules

in the relocatable library can also contain INCLUDE statements.

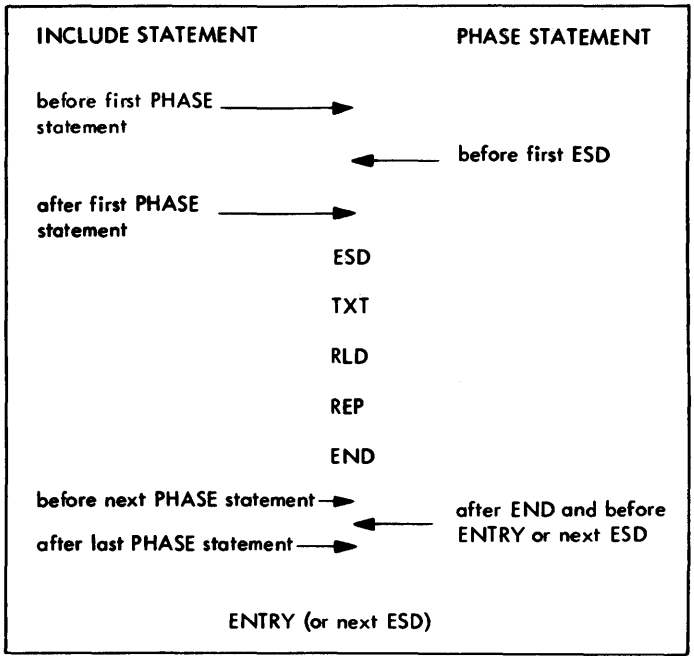
GENERAL CONTROL STATEMENT FORMAT

The Linkage Editor control statements are similar in format to statements processed by the Assembler. The operation field must be preceded by one or more blanks. The operation field must begin to the right of column 1 and must be separated from the operand field by at least one blank position. The operand field is terminated by the first blank position. It cannot extend past column 71.

CONTROL STATEMENT PLACEMENT

When preparing multiple-object modules in a single Linkage Editor run, the single ENTRY statement should follow the last object module. The ACTION statement(s) must be the first record(s) encountered in the input stream; otherwise, they are ignored.

PHASE and INCLUDE statements may be present on SYSRDR, SYSIPT, or in the relocatable library. Figure 12 shows the possible placement of the PHASE and INCLUDE statements.



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

Figure 12. Placement of PHASE and INCLUDE Statements

PHASE STATEMENT

The PHASE statement must precede the first object module of each phase processed by the Linkage Editor. Under no circumstances can a PHASE statement occur within a control section. There can be several control sections within a phase. When several PHASE statements appear before an object module, each of the statements must be followed by at least one INCLUDE statement. Any object module not preceded by a PHASE statement will be included in the current phase.

This statement provides the Linkage Editor with a phase name and an origin point for the phase. The phase name is used to catalog the phase in the core image library. This name is used in a FETCH or LOAD macro to retrieve the phase for execution. The PHASE statement is in the following format.

Operation	Operand
PHASE	name,origin[,NOAUTO]

The name field is blank. The operation field contains PHASE. Entries in the operand field must be separated by commas. The entries in the operand field represent the following.

name — Symbolic name of the phase. One to eight alphameric characters are used as the phase name. Multiphase programs should have phase names of five to eight alphameric characters. The first four characters of each phase within a multiphase background program should be the same. Otherwise, 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. For more rapid retrieval of the multiphase foreground programs, each phase may be given the prefix FGP as the first three characters of the program name. Phases with these characters are cataloged in a separate directory on the resident pack. The capacity of the directory is 144 phases.

origin — Specifies the load address of the phase. If COMMON is used, the length of the largest COMMON is added to every phase origin, even if the origin is given as an absolute value. The load address can be in one of six forms:

1. symbol[(phase)][±relocation]
2. *[±relocation]
3. S[+relocation]
4. ROOT
5. +displacement
6. F+address

The elements that make up the six forms that specify the origin signify the following.

1. symbol: may be a previously defined phase name, a previously defined control section, or a previously defined external label (the operand of an ENTRY source statement).

   (phase): If symbol is a previously defined control section or a previously defined external label that appears in more than one phase, phase (in parentheses) directs the Linkage Editor to the phase in which the origin name is to be found. The phase name must have been defined previously.

   relocation: indicates the origin of the phase currently being processed will be set relative to the symbol by a relocation term consisting of a + or a - immediately followed by a hexadecimal number X'hhhhhh' of one to six digits, a decimal number ddddddd of one to eight digits, or nK, where K=1024.

2. *: indicates the Linkage Editor phase location counter. It will cause the Linkage Editor to assign the next main-storage location (with forced double-word alignment) as an origin for the next phase. In the case of the first PHASE statement, it signifies the first double-word address after the end of the Supervisor, the label block area (if any), and the area assigned to the COMMON pool (if any).

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

3. S: indicates the origin is to be made at the end of the Supervisor, the label block area (if any), and the area assigned to the COMMON pool (if any).

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

4. ROOT: causes the Linkage Editor to consider the phase that follows as a root phase that will always be resident in main storage while the program is being executed. The main-storage address assigned to this phase is the first double-word address after the end of the Supervisor, the label block

area (if any), and the area assigned to the COMMON pool (if any). Only the first PHASE statement is permitted to specify ROOT. Any qualitative information (phase or relocation) is ignored when ROOT is specified. If a control section appears in the root phase, other occurrences of the same control section are ignored and all references are resolved to the control section in the root. Control sections are not duplicated within the same phase. If any subsequent phase overlays any part of the ROOT phase, a warning diagnostic is displayed on SYSLST if ACTION MAP is specified.

5. +displacement: allows the origin point (loading address) to be set at a specified location. The origin point is an absolute address, relative to zero. displacement is a hexadecimal number X'hhhhhh' of one to six digits, a decimal number ddddddd of one to eight digits, or nK, where K=1024. +0 would be used to denote a self-relocating program.

6. F: indicates a foreground program is being linkage edited and an area is to be reserved at the beginning of the foreground area for the program name, a register save area, and label information. F should never be used for self-relocating programs.

   +address: the positive absolute main storage address of the foreground area in which the linkage-edited program is to be executed. It may be specified by a hexadecimal number X'hhhhhh' of four to six digits, or a decimal number ddddddd of five to eight digits, or in the form nnnnK, where n is two to four digits and K=1024. For example, an address may be specified as +32K or +X'8000' or +32768. The origin of the phase is on the first double-word boundary after the sum of address, and the adjustment for the save area requirements.

NOAUTO — Indicates that the Automatic Library Look-Up (AUTOLINK) feature is suppressed. The AUTOLINK feature provides the following. AUTOLINK collects each unresolved external reference from the phase. It then searches the 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 relocatable library is edited into the phase. Note that the AUTOLINK retrieved module must have an entry point matching the external reference in order to resolve its address. Unresolved external references are processed sequentially in alphameric order. Object-module cross references with labels identical to library object-module entry-point labels are erroneous. The use of NOAUTO as the last operand in a PHASE statement causes the AUTOLINK process to be suppressed for that phase only. (Also see ACTION Statement.)

Some examples of PHASE statements follow.

```
PHASE PHNAME,*+504
```

This causes loading to start 504 bytes past the end of the previous phase.

```
PHASE PHNAME3,PHNAME2
```

This causes loading to start in the same point where the loading of the phase by the name PHNAME2 started.

```
PHASE PHNAME;ROOT
```

This causes loading to start after the end of the Supervisor, the label block area (if any) and the COMMON pool (if any) in the problem program area. When the PHASE statement contains a ROOT origin, this PHASE statement must be the first PHASE statement read by the Linkage Editor; otherwise, it is treated as a symbol.

```
PHASE PHNAME,CSECT1(PHNAME2)
```

This causes loading to start at the point where CSECT1 was loaded. CSECT1, the named control section, must have appeared in the phase named PHNAME2.

```
PHASE PHNAME,F+X'6000'
```

This causes loading to start at 24K plus the length of the save area and label area.

```
PHASE PHNAME,F+32K
```

This causes loading to start at 32K plus the length of the save area and label area.

```
PHASE PHNAME1,F+30K
PHASE PHNAME2,*
PHASE PHNAME3,PHNAME2
```

The first phase (PHNAME1) of the preceding series will be loaded starting at 30K plus the length of the save area and label area. The second phase (PHNAME2) of the series will be loaded at the end of PHNAME1. The third phase (PHNAME3) will be loaded at the same address as was PHNAME2, i.e., at the end of PHNAME1.

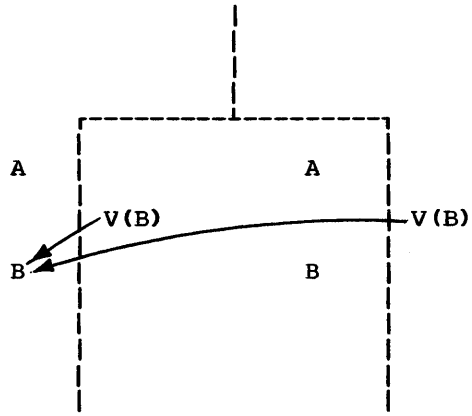
Note: In each of the preceding examples, if the origin address supplied is not on a double-word boundary, the Linkage Editor will automatically increment to the next double-word boundary.

The Linkage Editor will allow the inclusion of the same control section within each of several phases. If a control section appears in a ROOT phase, it will not appear in any other phase. A duplicate control section within the same phase will be ignored.

As external references occur in a phase, they are resolved preferentially with the entry point within the ROOT phase (if any), or the last previous occurrence of this entry point. For example, the coding

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

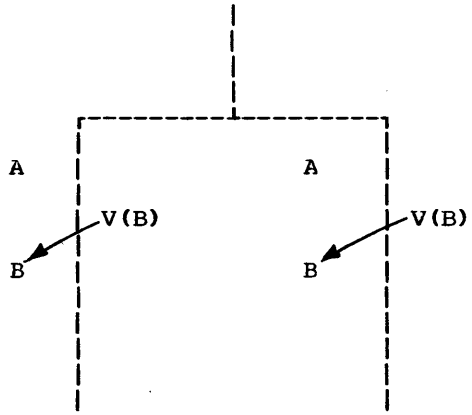
when used as a module in two phases produces



whereas the coding

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

when used as a module in two phases produces



This method of coding redefines the sequence of ESD information to allow valid cross reference by the Linkage Editor.

When linkage editing in the AUTOLINK mode, this is also true except for the case of privileged external references. Privileged external references are, by defini-

tion, those external references whose labels begin with the letters IJ. In this case, if the resolution is not possible within the current phase or ROOT phase, the AUTOLINK function is performed on this external reference at the end of the phase. The other previously defined phases are not examined for possible resolution. If NOAUTO is specified, the IJ prefix is not privileged. If the IJ module is not defined in the relocatable library, the address constant referencing this label will not be resolved.

INCLUDE STATEMENT

This statement is used to indicate that an object module is to be included for editing by the Linkage Editor. It has two optional operands. When both operands are used, they must be in the prescribed order. When the first operand is omitted and the second operand is used, a comma must precede the second operand. The first operand indicates that the input is in the relocatable library. The second operand indicates that the input is in sub-modular 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 the relocatable library. It must be the same module name that was used when the module was cataloged in the relocatable library. Including modules from the relocatable library 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 that 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 sub-modular phase is created. The counterpart of a sub-modular 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.

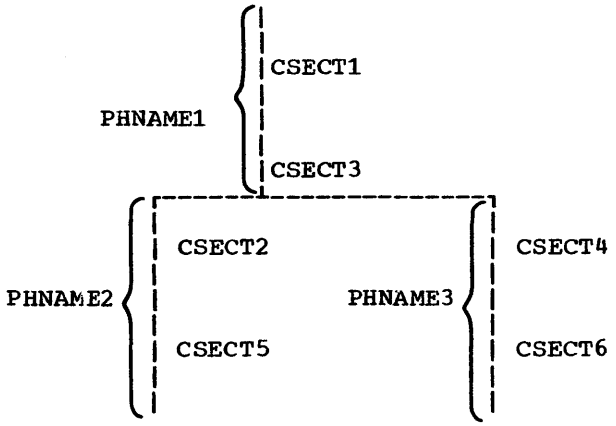
Modules in the relocatable library can be nested by using INCLUDE statements up to a depth of five (level of six). Modules included by INCLUDE statements read from SYSRDR are referred to as being in the first level. Modules included by statements in the first level are at the second level. This is illustrated in Figure 14. 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. Note that 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 implying that sub-modular 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 non-submodular INCLUDE statement may be placed before or after a sub-modular INCLUDE statement. This results in the addition of the included module into the phase at the point the INCLUDE statement is encountered. For example, if MOD1 contains CSECT4 and CSECT5, the sequence

```
PHASE     PHNAME1,*
INCLUDE   ,(CSECT1,CSECT3)
INCLUDE   MOD1
```

results in the following structure

```
            CSECT1
            CSECT3
PHNAME1     CSECT4
            CSECT5
```

while the sequence

```
PHASE     PHNAME1,*
INCLUDE   MOD1
INCLUDE   ,(CSECT1,CSECT3)
```

results in the following structure

```
            CSECT4
            CSECT5
PHNAME1     CSECT1
            CSECT3
```

Note: Both of the following statements produce the same result.

```
INCLUDE   ,(CSECT1,CSECT3)

INCLUDE   ,(CSECT3,CSECT1)
```

i.e., 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 INCLUDE's must be used:

```
INCLUDE   ,(CSECT3)

INCLUDE   ,(CSECT1)
```

As no diagnostic is given if a control section, specified in the namelist, is not present in the indicated module, the user can inspect the MAP supplied by the Linkage Editor to determine if the proper control sections are in the correct phases.

ENTRY STATEMENT

Every program, as input for the Linkage Editor, is terminated by an ENTRY statement. Its format is:

Name	Operation	Operand
blank	ENTRY	[entrypoint]

entrypoint — Symbolic name of an entry point. It must be the name of a CSECT or a label definition (source ENTRY) defined in the first phase. This address is used as the transfer address to the first phase in the program. If the operand field is blank, the Linkage Editor will use as a transfer address the first significant address provided in an END record encountered during the generation of the first phase. If no such operand is found on the END card, the transfer address will be the load address of the first phase.

It is necessary to supply the ENTRY statement only if the user wishes to provide a specific entry point. Job Control writes an ENTRY statement with a blank operand on SYSLNK when EXEC LNKEDT is read to ensure that an ENTRY statement will be present to halt linkage editing.

ACTION STATEMENT

This statement is used to indicate Linkage Editor options. When used, the statement must be the first Linkage Editor record(s) in the input stream. Its format is:

Name	Operation	Operand
blank	ACTION	CLEAR MAP NOMAP NOAUTO CANCEL

CLEAR — Indicates that the entire temporary portion of the core image library will be set to binary zero before the beginning of the Linkage Editor function. CLEAR is a time consuming function. It should be used only if it is necessary to fill areas defined by DS statements with zeros.

MAP — Indicates that SYSLST is available for diagnostic messages. In addition, a main storage map is output on SYSLST. The map contains every entry within each CSECT and every CSECT within each phase.

NOMAP — Indicates that SYSLST is not available when performing the linkage-edit function. Mapping of main storage is not performed and all Linkage Editor error diagnostics are listed on the printer-keyboard (SYSLOG).

NOAUTO — Indicates the AUTOLINK function is to be suppressed during the linkage editing of the entire program.

CANCEL Causes an automatic cancellation of the job if errors 2100I through 2170I occur. If this option is not specified, the job continues.

The ACTION statement is not required. If the MAP option is specified, SYSLST must be assigned. If the statement is not used and SYSLST is assigned, MAP is assumed and a map of main storage and any error diagnostics are output on SYSLST. If the statement is not used and SYSLST is not assigned, NOMAP is assumed.

The following information is contained in the map of main storage:

1. The name of each phase, the lowest and highest main storage locations of each phase, and the hexadecimal disk address where the phase begins in the core image library.

2. An indication if the phase is a ROOT phase, or if a phase overlays the ROOT phase in any way (designated by OVEROOT).

3. The length of COMMON, if appropriate.

4. The names of all CSECTs belonging to a phase, the address where each CSECT is loaded, and the relocation factor of each CSECT.

5. All defined entry points within a CSECT. If an entry point is unreferenced, it is flagged with an asterisk (*).

6. The names of all external references that are unresolved.

7. The transfer (execute) address of each phase.

8. Warning messages are printed if: the ROOT phase has been overlaid; a possible invalid entry point duplication occurred; the ENTRY or END statement contained an invalid (undefined) transfer label; at least one control section had a length of zero; the assembled origin on an RLD statement was outside the limits of the phase; or, an address constant could not be resolved. These messages may or may not indicate actual programming errors. The user will not be aware of these warnings if NOMAP is operational.

Note 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 13 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 13.

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 sub-modular 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.

Note that 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 13.

1. Line 2 (COMMON). The entry under REL-FR contains the length instead of the relocation factor in the case of ESD-type COMMON.

2. Lines 5 and 9 (referring to UNREFERENCED SYMBOLS). These ENTRY labels (POINT2 and POINT3) are not referenced as an external symbol, i.e., 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. Note that in the sub-modular structure, CSECTs not specified in any namelist appear as EXTRNs.

4. Lines 3, 6, 7, 11, and 15. Note that 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 sub-modular 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 compilers do not supply all of the information required by the Linkage Editor in the ESD records. Specifically, the control section length is provided in the END record. If a control section defined in the ESD information has a length of zero, it normally indicates the length will appear in the END record. It is possible for the user to generate zero-length control sections through the Assembler. Such a condition produces this message. This will not be 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 would normally correspond to referencing the EXTRNs shown in lines 17 and 18.

10. Line 24. Address constants had load addresses outside the limits of the phase in which they occurred. This normally occurs if the control section length is incorrectly defined in the input.

```
JOB  EXAMPLE              DISK LINKAGE EDITOR DIAGNOSTIC OF INPUT

ACTION TAKEN    MAP  CLEAR
LIST           PHASE PHASE1,ROOT,NOAUTO
LIST         INCLUDE  ,(NAMEONE)

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

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

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

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

21411  EX1 0002 ESD 404040 0010 0002 POINT3   1 000244 000007 NAMETWO  0 000130 001898 NAMTHREE 0 000200 0000A8

LIST        INCLUDE  RELMOD

21311        INCLUDE  RELMOD

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

LIST          PHASE   PHASE4,+16500
LIST     INCLUDE   RELMODUL

21441  REL 0015 TXT 004250 0038 F0F1 1A361A56 46D0E254 4130EF1E D500EF1E E5FA477C E2869201 FF1E0630 9509EF1D 4770E286

LIST    AUTOLINK   AUTOMOD2
LIST           PHASE PHASE5,+X'25BA',NOAUTO
LIST    REP  004016 0034130,C03A,47F0,C3GE       PATCH ASSEMBLY ERRORS
LIST    REP  00400C 003D20E,

21021           REP 004000 C3F0 0003 D2CEFCC5 6B404040 40404040 40404040 4040404C 40404040 40404040 40404040 40404040

LIST     ENTRY             INVALID TRANSFER LABEL
```

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

COMMON                                              COM                 001800  000008

ROOT     PHASE1    0018C8  0018C8  0019F7  13 2 2   CSECT     NAMEONE   0018C8  0018C8
                                                       ENTRY   POINT1    0018CC
                                                    *  ENTRY   POINT2    001930

         PHASE2    0019F8  0019F8  001A87  13 3 1   CSECT     NAMEFOUR  0019F8  001750

OVERLOT  PHASE3    0019E8  001918  001B1F  13 3 2   CSECT     NAMETWO   001918  0017E8

                                                    CSECT     NAMTHREE  0019E8  0017E8
                                                    *  ENTRY  POINT3    001A2C

                                                    CSECT     NAMEFOUR  001A90  0017E8

         PHASE4    0043A0  00414C  0059A3  13 4 1   CSECT     AUTOMOD1  004140  003A98
                                                       ENTRY   AUTOENT   004200

                                                    CSECT               0043A8  003EF8

                                                    CSECT     AUTOMOD2  0043C0  0003C0

         PHASE5    002688  002688  002767  13 6 1   CSECT               0024F8 -001808

                                                    CSECT     NAME5     0024F8 -001808

* UNREFERENCED SYMBOLS                              EXTRN     PONT2
                                                    EXTRN     POINT4

ROOT STRUCTURE OVERLAID BY SUCCEEDING PHASE

POSSIBLE INVALID ENTRY POINT DUPLICATION IN INPUT

INVALID TRANSFER LABEL ON END OR ENTRY STATEMENT IGNORED

CONTROL SECTIONS OF ZERO LENGTH IN INPUT

002 UNRESOLVED ADDRESS CONSTANTS

003 ADDRESS CONSTANTS OUTSIDE LIMITS OF PHASE
```

Figure 13. Map of Main Storage

PHASE ENTRY POINT

The Linkage Editor stores with each phase the absolute entry point of that phase. These entry points are as follows.

1. For the first phase, the entry point specified in the ENTRY statement is used. If no entry point is specified, the first significant entry address taken from an END statement encountered in the construction of the first phase is used.

2. For sub-modular phases, the entry point specified in the END record is used if the CSECT that contains this entry point was specified in the namelist. If an external label is the entry point defined by the END record, it must be previously defined and will be used for all phases constructed from this module. If no entry point is specified, the beginning address of each phase is used.

3. For all phases, the entry address (if any) specified in the first END record encountered in the construction of the phase is used.

SELF-RELOCATING PROGRAMS

A system with multiprogramming has the capability of executing self-relocating programs. A self-relocating program is one that can be executed at any location in main storage. A program that is self-relocating must initialize its address constants, including Channel Command Words (CCW's), 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'.

Because self-relocating programs must be assigned to origin at location zero when linkage edited, multiphase programs should use the LOAD macro instruction rather than the FETCH macro instruction, with the register specifying the "load address". Control is given to the LOADed phase by branching to the contents of register 1 (ENTRY point). The user should code his LOADing mechanism in a place where it is not overlaid by a new phase.

Appendix G shows an example of a self-relocating program.

LINKAGE EDITOR INPUT RESTRICTIONS

In a background area of 10K, the Linkage Editor can process at least 30 phases in a program, 221 unique ESD items in a program, and 27 ESID items in a module.

A combination of the preceding values can be accommodated by the Linkage Editor in accordance with the formulas shown in Appendix F.

In a background area greater than 10K, the additional main storage is used to produce faster linkage-edit times and accommodate a greater number of combinations of the preceding values in accordance with the formulas shown in Appendix F.

LINKAGE EDITOR JOB SET UP

When performing a linkage edit function, the following system and programmer logical units are used. Note that SYSRDR and SYSIPT may contain input for the Linkage Editor. This input is written onto SYSLNK by Job Control.

Unit	Function
SYSRDR	Control statement input (via Job Control)
SYSIPT	Module input
SYSLST	Programmer messages and listings
SYSLOG	Operator messages
SYSLNK	Input to the Linkage Editor
SYS001	Workfile

In normal operations, all preceding logical units must be assigned. In a unique circumstance (when all modules to be linkage edited are in the relocatable library), SYSIPT would not need to be assigned.

A linkage edit job is set up in the following manner.

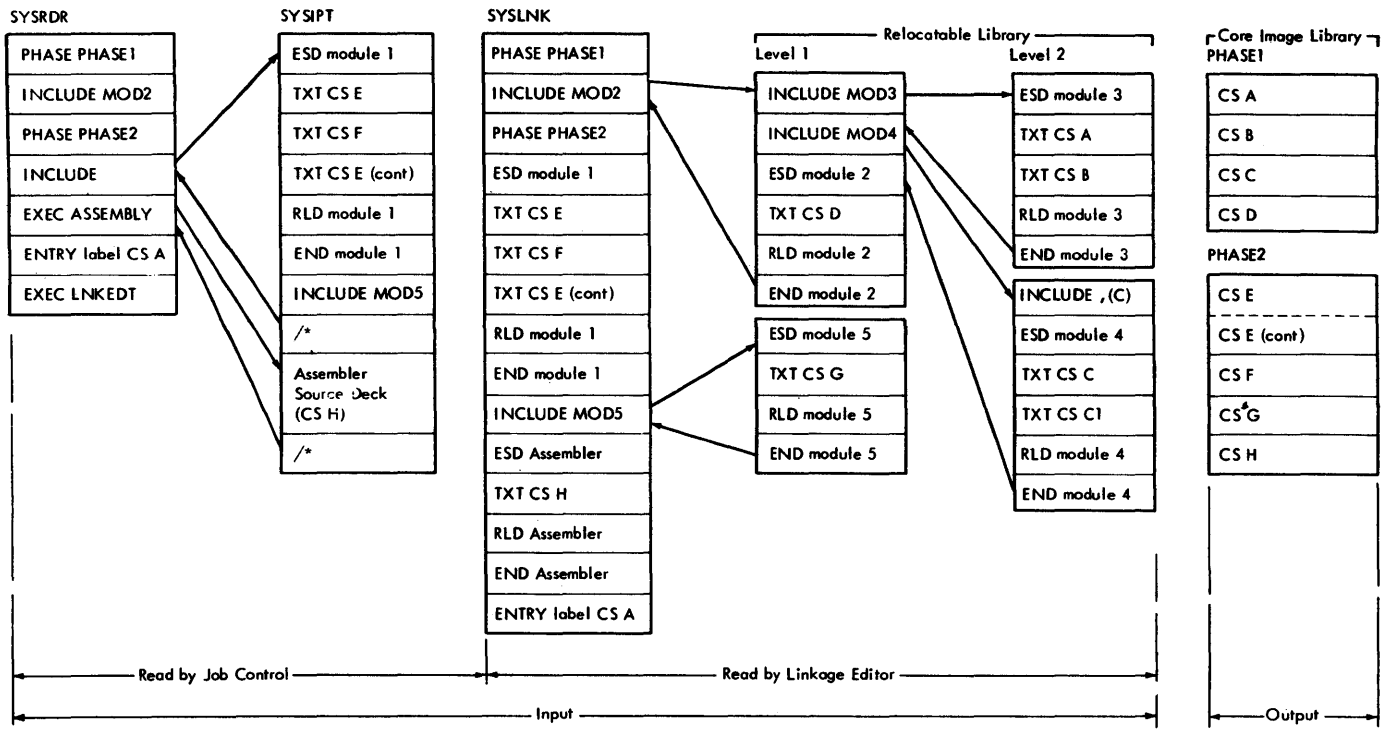
Control Statement	Remarks
// JOB	Required only if this is the first job step of a job.
// ASSGN	Required only if device assignments are to differ from the system standard assignments. Units that can be assigned are SYSRDR, SYSIPT, SYSLST, SYSLNK, and SYS001.
// OPTION	OPTION statement must follow the ASSGN statement (if any) for SYSLNK.
ACTION	Optional ACTION statement (with appropriate operand) must precede the first Linkage Editor control statement.
PHASE INCLUDE	As many PHASE and INCLUDE statements as are required are used to construct phases from the modules input to the Linkage Editor.
ENTRY	Optional statement to provide a transfer address for the first phase.
// 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.

EXAMPLE OF LINKAGE EDITOR INPUT AND OUTPUT

The program shown in Figure 14 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 14. Example of Linkage Editor Input and Output

The Linkage Editor reads from SYSLNK and starts to create a program. An INCLUDE statement with a nonblank first entry in the operand field signals the Linkage Editor to access the relocatable library. This is the first level of an INCLUDE. In the first level of the relocatable library, the Linkage Editor reads an INCLUDE (for the second level) and performs this inclusion. As no INCLUDE is present in the second level, control is returned to the calling input level. This process is repeated for the next INCLUDE. Note that the namelist specifies only CS C is wanted.

After the inclusion of the module at the first level, control is returned to SYSLNK where a new phase is encountered. The control sections are read from SYSLNK and added to PHASE2 until the next INCLUDE is read. At this time, the Linkage Editor again accesses the relocatable library, performs the inclusion of MOD5 into PHASE2, and continues reading input from SYSLNK. Processing continues until the ENTRY statement is reached.

Note that the split control section (CS E) is assigned a contiguous area of main storage.

LIBRARIAN

This section describes the set of programs that maintain, service, and copy the libraries of the Disk Operating System. This set of programs is collectively referred to as the Librarian.

The system residence (SYSRES) can contain three separate and distinct libraries:

1. Core image library
2. Relocatable library
3. Source statement library.

The core image library is required for each disk-resident system. The other two libraries, the relocatable library and the source statement library, are not required for operating a system.

CORE IMAGE LIBRARY

The core image library contains any number of programs. Each program is made up of one or more separate phases. Hence, each phase may either be a single program, or an overlay of a multiphase program. The size and number of programs in the core image library are the factors that determine the amount of space that need be allocated to the library.

All programs in the core image library are edited to run with the resident Supervisor. Each program phase is assigned a fixed location in main storage. The programs in the core image library include system programs, other IBM programs such as Assembler, RPG, COBOL, FORTRAN, PL/I, and sort programs, and user programs.

Associated with the core image library is a core image directory. The directory contains a unique descriptive entry for each phase in the core image library. Each entry includes the name of the phase, the starting disk address of the phase in the core image library, the number of records necessary to contain the phase, the number of bytes in the last record, the starting address in main storage where the phase will be loaded, and its entry point. The entries in the core image directory are used to locate and retrieve phases from the core image library.

Phases in the core image library and entries in the core image directory are in the order in which they were cataloged.

RELOCATABLE LIBRARY

The relocatable library contains any number of modules. Each module is a complete object deck in relocatable format. The size and number of modules in the relocatable library are the factors that determine the amount of space that need be allocated to the library.

The purpose of the relocatable library is to allow the user to maintain frequently used routines in residence and combine them with other modules without requiring recompilation. The routines from the relocatable library are edited in the core image library by the Linkage Editor.

Associated with the relocatable library is a relocatable directory. The directory contains a unique descriptive entry for each module in the relocatable library. Each entry includes the name of the module, the starting disk address of the module in the relocatable library, the number of records required to contain the module, 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

SOURCE STATEMENT LIBRARY

The source statement library contains any number of books. Each book in the source statement library is made up of a sequence of source language statements. The size and number of books in the source statement library are the factors that determine the amount of space that need be allocated to the library.

The purpose of the source statement library is to provide an extension of the functions of a macro library. If a source program contains a macro instruction, the macro definition in the source statement library corresponding to the macro instruction is generated in the source program. If the source program contains a COPY statement, the specific language translator will compile a book from the source statement library into the source program.

Each book in the source statement library is classified as belonging to a specific sublibrary. Sublibraries are currently defined for two programming languages (Assembler and COBOL) in the system. Classifying books by a sublibrary name allows books written in these languages to have the same name.

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, and the number of records required to contain the book. 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.

DISK STORAGE SPACE REQUIRED FOR LIBRARIES AND DIRECTORIES

The relative location of each of the library and directory areas is fixed. The amount of space allocated to each is determined by the user. Each library area consists of one or more complete disk cylinders. Each directory area consists of one or more complete disk tracks. Each directory occupies the first track(s) of the first cylinder allocated to its respective library. For example, because the core image library always begins on cylinder 1, the core image directory always begins on track 0 of cylinder 1.

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.

Core Image Directory Size

Each track allocated to the core image directory can contain entries for 144 phases with the exception of the last track, which can contain only 143 entries. Thus, the number of tracks (T_D) required for the core image directory equals:

$$T_D = \frac{P}{144}$$

where P = total number of phases in the core image library. The value of T_D is rounded up to the next highest integer if a remainder results.

Core Image Library Size

Each track allocated to the core image library contains two fixed-length blocks. Each block contains a maximum of 1728 bytes of instructions or data. The core image library contains exactly the same information as is loaded into main storage for execution. Each phase is written beginning in a new block. The number of tracks required for the core image library can be calculated as follows.

1. Determine the number of blocks (B_n) required for a phase:

   $$B_n = \frac{L}{1728}$$

   where L = total number of bytes in the phase. The value B_n is rounded up to the next highest integer.

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

   $$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$

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

   $$T_L = \frac{B_t}{2}$$

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

   $$C = \frac{T_D + T_L}{10}$$

The value C is rounded up to the next highest integer if a remainder results.

Relocatable Directory Size

Each track allocated to the relocatable directory can contain entries for 207 modules, with the exception of the last track, which can contain only 206 entries. Thus, the number of tracks (T_D) required for the relocatable directory equals:

$$T_D = \frac{M}{207}$$

where M = total number of modules in the relocatable library. The value T_D is rounded up to the next highest integer if a remainder results.

Relocatable Library Size

Each track allocated to the relocatable library contains 9 fixed-length blocks. 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 provide a fairly accurate approximation of the library area required for typical programs.

1. Determine the number of blocks (B_c) required for all cards or statements except the actual program text. Assume a separate block for each card of the following types:
   a. PHASE
   b. INCLUDE
   c. REP
   d. END
   e. SYM
   f. ENTRY

   Let B_c = total number of cards of the above types.
2. Determine the number of blocks (B_e) required for ESD and RLD cards. Assume a separate block for every two ESD or RLD cards.
3. Determine the number of blocks (B_i) required for the actual instructions or data in the TXT cards. Assume an average of 200 bytes of text in each block. (A maximum per block, for contiguously assigned text, is 264 bytes per block.) Thus,

   $$B_i = \frac{\text{total bytes of text in TXT cards}}{200}$$
4. Determine the total number of blocks (B_n) required for a module in the relocatable library:

   $$B_n = B_c + B_i + B_e$$
5. Determine the total number of blocks (B_t) required to hold all of the modules in the library:

   $$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$
6. Determine the number of tracks (T_L) required for the relocatable library:

   $$T_L = \frac{B_t}{9}$$

   The value T_L is rounded off to the next highest integer if a remainder results.
7. Determine the number of cylinders (C) required to hold the relocatable library and relocatable directory:

   $$C = \frac{T_D + T_L}{10}$$

   The value C is rounded up to the next highest integer if a remainder results.

Source Statement Directory Size

Each track allocated to the source statement directory can contain entries for 160 books with the exception of the last track, which can contain only 159 entries. Thus, the number of tracks (T_D) required for the source statement directory equals:

$$T_D = \frac{B}{160}$$

where B = total number of books in the source statement library. The value T_D is rounded up to the next highest integer.

Source Statement Library Size

Each track allocated to the source statement library contains sixteen fixed-length blocks. Each block contains a maximum of 160 bytes of source statement information. The source statements coded by the user are compressed before writing them out in the source statement library. This compression is performed by eliminating all blanks in each source statement. Several count bytes indicating the number of blanks eliminated are added to each statement before writing it in the source statement library. The number of tracks required for the source statement library can be calculated as follows:

1. Determine the number of statements (N) used to define a book.

2. Determine the average compressed-statement length (L_s) in the book. The compressed-statement length (S) approximately equals the sum of:

$$(L_1 + 1) + (L_2 + 1) + \ldots + (L_n + 1) + 5 = S$$

   where each L_n = bytes in each word of the source statement.

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

$$B_n = \frac{N(L_s)}{160}$$

   The value B_n is rounded up to the next highest integer if a remainder results.

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

$$B_t = B_1 + B_2 + B_3 + \ldots + B_n$$

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

$$T_L = \frac{B}{16}$$

   The value T_L is rounded up to the next highest integer if a remainder results.

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

$$C = \frac{T_D + T_L}{10}$$

   The value C is rounded up to the next highest integer if a remainder results.

LIBRARIAN FUNCTIONS

The Librarian programs perform three major functions:

1. Maintenance
2. Service
3. Copy.

Maintenance functions are used to 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 are used to translate information from a particular library to printed (displayed) or punched output. Information in a library directory can also be displayed. 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.

The copy function is used to either completely or selectively copy the disk on which the system resides. The CORGZ program is the program for the copy function.

Librarian functions are performed through use of control statements. The control statements are:

1. A JOB control statement.

2. A number of ASSGN control statements that may be required to change the assignment of actual input/output devices.

3. An EXEC control statement requesting a particular librarian program.

4. Librarian specification statements describing various functions to be performed.

5. A /* control statement.

6. A /& control statement.

The JOB, ASSGN, /*, and /& control statements are the same as those described in the Job Control section. The operand field of the EXEC control statement is described in this section. The other statements pertain to the Librarian and are described in this section.

Librarian functions can be performed separately, or in certain combinations as described in the following sections. All control statement information is read from

the device assigned (in the ASSGN statements) to SYSRDR. All input data is read from the device assigned to SYSLNK or SYSIPT. SYSIPT and SYSRDR can be assigned to the same physical input/output device.

Figure 15 is a table of all maintenance functions. Figure 16 is a table of all service functions. Figure 17 is a table of the copy function.

Function	Unit	Element	Control Statements Required

Catalog	Core Image Library	Phase	// JOB jobname
			// OPTION CATAL
			(Linkage Editor control statements)
			// EXEC LNKEDT
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			CATALR modulename
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			CATALS sublib.bookname
			/*
			/&

Delete	Core Image Library	Phase	// JOB jobname
			// EXEC MAINT
			DELETC phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC MAINT
			DELETC prog1.ALL[,prog2.ALL,...]
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			DELETR module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC MAINT
			DELETR prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC MAINT
			DELETR ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			DELETS sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC MAINT
			DELETS sublib.ALL
			/*
			/&

Figure 15. Maintenance Functions (Part 1 of 2)

Function	Unit	Element	Control Statements Required

Rename	Core Image Library	Phase	// JOB jobname
			// EXEC MAINT
			RENAMC oldname,newname[,oldname,newname,...]
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC MAINT
			RENAMR oldname,newname[,oldname,newname,...]
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC MAINT
			RENAMS sublib.oldname,sublib.newname[,sublib.oldname,sublib.newname,...]
			/*
			/&

Condense	Core Image Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS CL
			/*
			/&

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

	Source Statement Library	Library	// JOB jobname
			// EXEC MAINT
			CONDS SL
			/*
			/&

	Libraries	All	// JOB jobname
			// EXEC MAINT
			CONDS CL,RL,SL
			/*
			/&

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

Reallo-cation	System	Library	// JOB jobname
			// VOL SYSRES,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC MAINT
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			/*
			/&
			
			Notes: Values to be substituted for id:
			CL -- Core image library
			RL -- Relocatable library
			SL -- Source statement library
			Values to be substituted for cylin and tracks:
			Any integer

Figure 15. Maintenance Functions (Part 2 of 2)

Function	Unit	Element	Control Statements Required

Display	Core Image Library	Directory	// JOB jobname
			// EXEC DSERV
			DSPLY CD
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			DSPLY module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			DSPLY prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			DSPLY ALL
			/*
			/&

		Directory	// JOB jobname
			// EXEC DSERV
			DSPLY RD
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			DSPLY sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			DSPLY sublib.ALL
			/*
			/&

		Directory	// JOB jobname
			// EXEC DSERV
			DSPLY SD
			/*
			/&

	Transient Directory	Directory	// JOB jobname
			// EXEC DSERV
			DSPLY TD
			/*
			/&

	System Directory	Directory	// JOB jobname
			// EXEC DSERV
			/*
			/&

	Directories	All	// JOB jobname
			// EXEC DSERV
			DSPLY ALL
			/*
			/&

Figure 16. Service Functions (Part 1 of 2)

Function	Unit	Element	Control Statements Required

Punch	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			PUNCH module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			PUNCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			PUNCH ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			PUNCH sublib.book1[,sublib.book2,...][,CMPRSD]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			PUNCH sublib.ALL[,CMPRSD]
			/*
			/&

Display and Punch	Relocatable Library	Module	// JOB jobname
			// EXEC RSERV
			DSPCH module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// EXEC RSERV
			DSPCH prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// EXEC RSERV
			DSPCH ALL
			/*
			/&

	Source Statement Library	Book	// JOB jobname
			// EXEC SSERV
			DSPCH sublib.book1[,sublib.book2,...][,CMPRSD]
			/*
			/&

		Sub-library	// JOB jobname
			// EXEC SSERV
			DSPCH sublib.ALL[,CMPRSD]
			/*
			/&

Figure 16. Service Functions (Part 2 of 2)

Function	Unit	Element	Control Statements Required

Copy	Core Image Library	Phase	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC phase1[,phase2,...]
			/*
			/&

		Program	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYC ALL
			/*
			/&

	Relocatable Library	Module	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002, IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR module1[,module2,...]
			/*
			/&

		Program	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR prog1.ALL[,prog2.ALL,...]
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYR ALL
			/*
			/&

Figure 17. Copy Function (Part 1 of 2)

Function	Unit	Element	Control Statements Required

Copy	Source Statement Library	Book	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS sublib.book1[,sublib.book2,...]
			/*
			/&

		Sub-library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS sublib.ALL
			/*
			/&

		Library	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPYS ALL
			/*
			/&

	Libraries	All	// JOB jobname
			// ASSGN SYS002,X'cuu'
			// VOL SYS002,IJSYSRES
			// DLAB 'DOS SYSTEM RESIDENCE FILE ...'
			// XTENT extent information
			// EXEC CORGZ
			ALLOC id=cylin(tracks)[,id=cylin(tracks),...]
			* PRECEDING ALLOC STATEMENT REQUIRED IF NEW LIMITS TO BE ESTABLISHED
			COPY ALL
			/*
			/&

Figure 17. Copy Function (Part 2 of 2)

MAINTENANCE FUNCTIONS

The set of maintenance functions contains five subsets:

1. Catalog
2. Delete
3. Rename
4. Condense
5. Reallocate.

The catalog function adds a phase to the core image library, adds a module to the relocatable library, or adds a book to the source statement library. If control statements and books or modules to be cataloged are read from the same device, the control statement must precede its associated book or module.

Programs to be cataloged in the core image library must first be edited by the Linkage Editor. Input for the Linkage Editor can be from SYSIPT, from the relocatable library, or directly from the language translator if the CATAL option is specified in the OPTION statement. See the section entitled Linkage Editor for a description of Linkage Editor functions that are performed prior to the catalog function for the core image library.

The delete function deletes an entry from a directory that corresponds to a phase, module, or book in a library. The phase, module, or book in the appropriate

library is not removed; however, as far as the system is concerned, the phase, module, or book no longer exists. In addition, entire programs can be deleted from the core image library and the relocatable library.

The rename function is used to rename an existing phase, module, or book in the appropriate library and directory.

The condense function is used to eliminate 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 is used to redefine 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.

When the /& statement is processed at the completion of a maintenance function, the system directory is displayed on SYSLST.

Note that condense and/or reallocation functions cannot be performed while a foreground program is being executed. This is because the library directories do not accurately reflect the content of the corresponding library at various times during these functions. The control program ignores any request for the ATTN routine when a condense and/or reallocation function is being performed.

SERVICE FUNCTIONS

The set of service functions contains three subsets:

1. Display
2. Punch
3. Display and punch.

The disk-resident system can display and/or punch modules in the relocatable library, and books in the source statement library. In addition, all system directories can be displayed.

Whenever a requested service function provides punched-card output, the output is on the device assigned to SYSPCH. Whenever a requested service function provides printed output, the output is on the device assigned to SYSLST.

COPY FUNCTION

The copy function is used to copy 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 for reducing a complete system to a system that is designed to perform a specific purpose.

GENERAL CONTROL STATEMENT FORMAT

The librarian control statements are similar in format to statements processed by the Assembler. The operation field must be preceded by one or more blanks. The operation field must begin to the right of column 1 and must be separated from the operand field by at least one blank position. The operand field is terminated by the first blank position. It cannot extend past column 71. Continuation statements are not recognized.

LIBRARIAN FUNCTIONS: CORE IMAGE LIBRARY

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

MAINTENANCE FUNCTIONS

To request a maintenance function, other than the catalog function, for the core image library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of four maintenance functions (delete, rename, condense, 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 five maintenance functions (catalog, delete, rename, condense, or reallocate) for either of the other two libraries (relocatable or source statement) 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 or the library routine directory, these

subdirectories are not updated until the /& statement is read.

Catalog

The catalog function adds a phase to the core image library. The CATAL option in the Job Control OPTION statement indicates that the Linkage Editor function will catalog the phase as a permanent phase in the core image library.

Each phase that is cataloged in the core image library derives its name from the PHASE control statement.

If a phase in the core image library is to be replaced by a new phase having the same name, only the catalog function need be used because the delete function is implied.

One or more phases can be cataloged in the core image library within a single run.

For the catalog function for the core image library, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLNK must be assigned to a disk unit. SYS001 must be assigned to a tape or disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the catalog function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYSLNK, SYS001, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. The OPTION control statement, with CATAL specified in the operand field, followed by
4. The appropriate Linkage Editor control statements (PHASE, INCLUDE, and ENTRY), followed by
5. The EXEC LNKEDT control statement, followed by
6. Any job control statement.

Delete

The delete function is used to remove 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 must 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the delete function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to change the name of a phase in the core image library to another name.

The RENAMC control statement is used to achieve the rename function. 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 must 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to eliminate vacancies, resulting from delete or catalog functions, between programs in the core image library. The condense function is used whenever a number of vacancies have accumulated within the library.

The CONDS control statement, in the following format, is used to condense the core image library.

CONDS CL

The operation field contains CONDS. The operand field contains CL. The relocatable library and/or the source statement library can also be condensed in this run. If this is desired, the entry RL (for the relocatable library) and SL (for the source statement library) can appear in the operand field. Multiple entries in the operand field are separated by commas.

For the condense function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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.

LIBRARIAN FUNCTIONS: RELOCATABLE LIBRARY

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

MAINTENANCE FUNCTIONS

To request a maintenance function for the relocatable library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of the five maintenance functions 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.

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 SYSRDR and is in the following format.

```
CATALR modulename
```

The operation field contains CATALR. The entry in the operand field, modulename, is the name by which the module will be known to the control system. The modulename is one to eight characters, the first of which must not be an asterisk.

The statements composing the input for a module are described in the section entitled Linkage Editor. 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. Any statements with a blank in column 1 are acceptable. All input is diagnosed by the Linkage Editor. Note that the CATALR statement is recognized but ignored by the Linkage Editor. The END statement indicates end of module.

The ENTRY statement can only be used in a module that contains only Linkage Editor control statements and an END statement. The ENTRY statement must be the last control statement in the module, preceding the END statement.

Normally, modules in the relocatable library are output from a language translator. However, the user can construct an artificial module of Linkage Editor control statements, referred to as a calling module. The following example illustrates a valid calling module:

```
PHASE PHNAM1,ROOT
INCLUDE MODULE1
PHASE PHNAM2,*
INCLUDE MODULE2
PHASE PHNAM3,PHNAM2
      .
      .
      .
ENTRY CSECTNME
END
```

Operands in INCLUDE statements refer to modules in the relocatable library. If, for example, the preceding calling module is cataloged by the name BIGPROG, all modules referred to in BIGPROG can be linkage edited by using the following control statements:

```
// OPTION CATAL
   INCLUDE BIGPROG
// EXEC LNKEDT
```

Note also that a calling module may consist only of INCLUDE statements. In this case, the PHASE statements would precede the appropriate 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSIPT must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard. If SYSRDR and/or SYSIPT are 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 device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYSIPT, 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 /* control statement if the other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Delete

The delete function is used to delete references to specific modules in the relocatable library. Any number of modules may be deleted during a single run. The modules are not physically deleted from the library; rather, the entry in the relocatable directory describing the module is deleted. Modules can be physically removed from the library (after the delete function has removed the entry from the directory) by performing a condense function. See the subsection entitled Condense.

The DELETR control statement in one of the following formats is used to delete a module from the relocatable library.

```
DELETR modname[,modname,...]

DELETR prog1.ALL[,prog2.ALL,...]

DELETR ALL
```

The first format is used when a specific module is to be deleted. The entry in the operation field is DELETR. The entry in the operand field, modname, is the name of the module to be deleted. If more than one module is to be deleted, the module names are separated by a comma. modname is one to eight characters, the first of which must not be an asterisk.

The second format is used when an entire program is to be deleted. The entry in the operation field is DELETR. In the operand field, prog refers to the first three characters of the modules used to build the program. (All IBM-supplied modules in the relocatable library making up a program have the same first three characters, such as IJQ for the Assembler and IJS for COBOL.) The three characters are followed by a period and ALL.

The third format is used if the entire library is to be deleted. The entry in the operation field is DELETR. The entry in the operand field is ALL. When this function is performed, the system status record is reset to show that all library blocks are now available to the system. Therefore, it is unnecessary to perform a condense function after a DELETR ALL has been performed.

Any number of DELETR control statements can be used for the relocatable library within a single run.

For the delete function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the delete function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to change the name of a module in the relocatable library to another name.

The RENAMR control statement is used to achieve the rename function. 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to eliminate 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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.

SERVICE FUNCTIONS

To request a service function for the relocatable library, use the following EXEC control statement.

// EXEC RSERV

One or more of the three service functions can be requested within a single run. Any number of modules within the relocatable library can be acted upon in this run.

Display

The display function is used to get a printout of a module in the relocatable library. Any number of modules can be displayed within a single run. The printed output consists of a header and the module.

Contained in the printed header is the module name and the number of records needed to contain the module.

The printed output of the module is represented by hexadecimal characters and EBCDIC, depending on the type of record and the information contained within the record. The fields of the printed output correspond to the card columns of the output cards shown in Appendix E.

The DSPLY control statement in one of the following formats is used to display modules in the relocatable library.

```
DSPLY module1[,module2,...]

DSPLY prog1.ALL[,prog2.ALL,...]

DSPLY ALL
```

The first format is used if only specific modules are to be displayed. The entry in the operation field is DSPLY. module in the operand field represents the name of the module to be displayed. If more than one module is to 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 LJQ 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the device assigned to SYSRDR, is as follows.

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to convert 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 LJQ 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the punch function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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, 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 SYSRDR and also to SYSPCH, a number of blank cards sufficient for punching the module must follow each PUNCH control statement. This is to prevent erroneously punching the cards of a following job step. Any extra cards that are not needed are automatically bypassed.

Display and Punch

The display-and-punch function is used to combine 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display-and-punch function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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, 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 SYSRDR and also to SYSPCH, a number of blank cards sufficient for punching the module must follow each DSPCH control statement. This is to prevent erroneously punching the cards of a following job step. Any extra cards that are not needed are automatically bypassed.

LIBRARIAN FUNCTIONS: SOURCE STATEMENT LIBRARY

This section describes the maintenance and service functions that relate to the source statement library. The copy function for the source statement library is discussed in the section entitled Copy Function.

MAINTENANCE FUNCTIONS

To request a maintenance function for the source statement library, use the following EXEC control statement.

```
// EXEC MAINT
```

One or more of the five maintenance functions 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.

Catalog

The catalog function adds a book to a sub-library 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 sub-library 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 sub-library with the same name as a book to be cataloged, that module in the sub-library 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 SYSRDR and is in the following format.

```
CATALS sublib.bookname
```

The operation field contains CATALS. The qualifier sublib in the operand field represents the sublibrary to which the book is to be cataloged and can be:

A for the Assembler sub-library

C for the COBOL sublibrary.

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.

Books that are to be cataloged in a sub-library 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 either the Assembler

or COBOL sublibrary of 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 would be 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSIPT must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 /* control statement if other job steps are to follow, or
6. The /& control statement, which is the last control statement of the job.

Card image input, read from the device assigned to SYSIPT, is as follows:

1. The BKEND statement, and/or the MACRO header statement if the book is a macro definition, followed by
2. The book to be cataloged, followed by
3. 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.

If SYSRDR and SYSIPT are assigned to the same device, the books to be cataloged immediately follow their respective CATALS control statements.

Delete

The delete function is used to remove references to specific books in a sub-library of the source statement library. The function can also be used to delete an entire sublibrary. 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
```

The first format is used if only specific books are to be deleted. The entry in the operation field is DELETS. The qualifier sublib in the operand field represents the sublibrary containing the book to be deleted and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

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. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The name of the book can be of any length; however, a maximum of the first eight characters is used to locate and delete the book. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be deleted. The entry in the operation field is DELETS. The first entry in the operand field is the name of the sublibrary to be deleted. The qualifier sublib represents the sublibrary containing the book to be deleted and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

The second entry in the operand field is ALL. The two entries must be separated by a period.

Whenever a DELETS statement causes the active blocks of the library to reach zero, 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 on the library when there are no active blocks.

For the delete function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the delete function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to change the name of a book in the source statement library to another name.

The RENAMS control statement is used to achieve the rename function. As soon as the statement is processed, the system recognizes only the new book-name. The RENAMS statement is in the following format.

```
RENAMS sublib.oldname,sublib.newname

    [,sublib.oldname,sublib.newname,...]
```

The operation field contains RENAMS. The qualifier sublib in the operand field represents the sublibrary containing the book to be renamed and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

oldname and newname represent the old book-name and the new book-name. If the sublibrary is specified for the newname, it must be the same as for the oldname. If no sublibrary is specified for the newname, it is assumed to be the same as for the oldname. If no sublibrary is specified for the oldname, the book is assumed to be in the Assembler sublibrary. The entries in the operand field must be separated by commas. The names in the operand field can be of any length; however, only a maximum of the first eight characters is used by the system to locate and rename the book.

Any number of RENAMS control statements can be used for the source statement library within a single run.

For the rename function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the rename function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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.

Condense

The condense function is used to eliminate 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, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to the printer-keyboard.

Control statement input for the condense function, read from the device assigned to SYSRDR, is as follows:

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 SYSRDR, 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.

SERVICE FUNCTIONS

To request a service function for the source statement library, use the following EXEC control statement.

```
// EXEC SSERV
```

One or more of the three service functions can be requested within a single run. Any number of books within the source statement library can be acted upon in this run.

Display

The display function is used to get 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 sublib.ALL
```

The first format is used if only specific books are to be displayed. The entry in the operation field is DSPLY. The qualifier sublib in the operand field represents the sublibrary containing the book to be displayed and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

book in the operand field represents the name of the book in the sublibrary to be displayed. If more than one book is to be displayed, the entries must be separated by commas. If books to be displayed are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters in length. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be displayed. The entry in the operation field is DSPLY. The first entry in the operand field is the name of

the sublibrary to be displayed. The qualifier sublib can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

The second entry in the operand field is ALL. The two entries must be separated by a period.

For the display function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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 is used to convert 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 sub.ALL[,CMPRSD]

The first format is used if only specific books are to be punched. The entry in the operation field is PUNCH. The qualifier sub in the operand field represents the sublibrary containing the book to be punched and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

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 non-qualified books are in the last sub-library specified. If a sub-library is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters long. Continuation statements are not recognized.

The second format is used if an entire sublibrary is to be punched. The entry in the operation field is PUNCH. The first entry in the operand field is the name of the sublibrary to be punched. The qualifier sub can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

The second entry in the operand field is ALL. The entry CMPRSD is used if the books are to be punched in the compressed format. A /* statement will always be punched at the end of the output. When SYSPCH is assigned to a tape unit or disk unit, each card image is preceded by a stacker-select character.

For the punch function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the punch function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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, 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 SYSRDR and also to SYSPCH, a number of blank cards sufficient for punching the book must follow each PUNCH control statement. This is to prevent erroneously punching the cards of a following job step. Any extra cards that are not needed are automatically bypassed.

Display and Punch

The display-and-punch function is used to combine 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 sub.ALL[,CMPRSD]

The first format is used if only specific books are to be displayed and punched. The entry in the operation field is DSPCH. The qualifier sub in the operand field represents the sublibrary containing the book to be displayed and punched and can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

book in the operand field represents the name of the book in the sublibrary to be displayed and punched. The entry CMPRSD is used if the books are to be punched in the compressed format, but printed in the original card image format. If more than one book is to be displayed and punched, the entries must be separated by commas. If books to be displayed and punched are in the same sublibrary, subsequent book names need not be qualified. (The Librarian assumes that nonqualified books are in the last sublibrary specified. If a sublibrary is never specified, the Librarian assumes the book is in the Assembler sublibrary.) The names of the books in the operand field can be from one to eight characters long. Continuation statements are not recognized. A /* statement will be punched at the end of the output.

The second format is used if an entire sublibrary is to be displayed and punched. The entry in the operation field is DSPCH. The first entry in the operand field is the name of the sublibrary to be displayed and punched. The qualifier sub can be:

A for the Assembler sublibrary

C for the COBOL sublibrary.

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.

For the display and punch function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSPCH must be assigned to a card punch, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display-and-punch function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, 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, 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 SYSRDR and also to SYSPCH, a number of blank cards sufficient for punching the book must follow each DSPCH control statement. This is to prevent erroneously punching the cards of a following job step. Any extra cards that are not needed are automatically bypassed.

LIBRARIAN FUNCTIONS: DIRECTORIES

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

To request the display service function for a directory (core image directory, relocatable directory, source statement directory, transient directory, or system directory), use the following EXEC control statement.

```
// EXEC DSERV
```

The display function is used to print the status of the directories defined for the system. Any number of directories can be displayed within a single run. No directory is displayed more than once in the same job step. The system directory is unconditionally displayed.

Each printed line contains one entry in the directory. All fields in the directory are headed by the title DEC (decimal) or HEX (hexadecimal).

Each printed directory is preceded by a header that contains the name of the directory in EBCDIC characters.

The DSPLY control statement in the following format is used to display specific directories or all directories.

```
DSPLY dir1[,dir2,...]

DSPLY ALL
```

The first format is used if only specific directories are to be displayed. The entry in the operation field is DSPLY. The entry in the operand field, dir, represents the name of the directory to be displayed. It can be:

1. TD for the transient directory. The transient directory shows the routines processed in the transient area and some frequently used library routines.

2. CD for the core image directory.

3. RD for the relocatable directory.

4. SD for the source statement directory.

If more than one directory is to be displayed, the symbols for the directories must be separated by a comma and can be in any order.

The second format is used if all five directories are to be displayed. The entry in the operation field is DSPLY. The entry in the operand field is ALL. In the absence of DSPLY, only a system directory is displayed. No more than one display of each directory is provided during each execution of DSERV.

For the display function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit. SYSLOG must be assigned to a printer-keyboard.

Control statement input for the display function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by

2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYSLST, and SYSLOG. The ASSGN statements are followed by

3. The EXEC DSERV control statement, followed by

4. The DSPLY control statement, followed by

5. The /* control statement if other job steps are to follow, or

6. The /& control statement, which is the last control statement of the job.

REALLOCATION FUNCTION

The reallocation function is used to redefine the number of tracks and cylinders allotted to the libraries and directories on a disk resident system. Any number of libraries or directories can be reallocated within a single run. The reallocation function can be used to increase, decrease, eliminate, or add specified areas in the

disk resident system. Each area that is reallocated is automatically condensed. The EXEC control statement required to perform a reallocation function is in the following format. Note that any other maintenance function (catalog, delete, or rename) for the three libraries may be performed in this run.

```
// EXEC MAINT
```

Associated with the EXEC statement for the reallocation function is the ALLOC control statement. The ALLOC control statement is in the following format.

```
ALLOC id=cylin(track)[,id=cylin(track),...]
```

The operation field contains ALLOC. The entry, id, in the operand field refers to the specific library and directory being reallocated and can be one of the following entries.

CL for the core image library and directory
RL for the relocatable library and directory
SL for the source statement library and directory.

The entry, cylin, in the operand field refers to the number of cylinders that contain the specified library. The entry, track, is enclosed within parentheses and refers to the number of tracks that contain the specified library directory. The tracks allocated to the directory are contained in the cylinders allocated to the library. The keyword operands are separated by a comma if more than one operand is present. For maximum efficiency, all requested operands should be entered on one statement.

When reallocation is being performed, only the areas being changed in the resident disk pack need be specified.

Consider this example.

```
ALLOC CL=16(1),SL=4(1)
```

In the 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.

For the reallocation function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

When executing the reallocation function of MAINT, a file must be defined for IJSYSRES on SYSRES via VOL, DLAB, and XTENT control statements. The information needed for this file is the same as for the IJSYSRES file of the copy disk function (see Copy Function), except that the symbolic unit is SYSRES instead of SYS002.

Control statement input for the reallocation function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. VOL, DLAB, and XTENT 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 is used to copy the system residence either selectively or completely. Along with the actual copying, the number of tracks and cylinders allocated to libraries and/or directories of the new system pack can be redefined.

The device number of the disk pack on which the disk resident system is to be copied is assigned to SYS002.

By using the ALLOC statement, described in the section entitled Reallocation Function, the number of tracks and cylinders allocated to the libraries and the directories can be respecified, if required, for the new system. The ALLOC statement in no way affects the old system. If the number of tracks and cylinders allocated to the new libraries and/or directories is not to differ from the old system, no ALLOC statement is required. If the ALLOC statement is used, only the areas being changed in the new system pack need be specified.

A complete copy consists of copying each directory, library, and any other area on the residence pack. The selective copy consists of copying only particular libraries, or specific programs, modules, or books within libraries. When either kind of copy (complete or selective) is performed, all libraries are automatically condensed.

The EXEC statement for the copy function is in the following format.

```
// EXEC CORGZ
```

Associated with the EXEC statement for the copy function is the COPYC, COPYR, or COPYS control statement for the core image library, relocatable library, or source statement library, respectively. A fourth control statement, COPY, is provided when the complete system is to be copied.

The COPYC control statement is used to specify the phases or programs in the core image library that are to be copied. It is in one of the following formats.

```
COPYC phase1[,phase2,...]

COPYC prog1.ALL[,prog2.ALL,...]

COPYC ALL
```

The first format is used when specific phases are to be copied. The entry in the operation field is COPYC. The entry, phase, in the operand field represents the name(s) of the phase(s) to be copied. Entries in the operand field must be separated by commas.

The second format is used when specific programs are to be copied. The entry in the operation field is COPYC. The entry, prog.ALL, in the operand field represents the name of the program to be copied. prog is the first four characters of the program name. (All phases within a program have the same first four characters.) prog is followed by a period and ALL. Entries in the operand field must be separated by commas.

The third format is used to copy the complete core image library. The entry in the operation field is COPYC. The entry in the operand field is ALL.

The COPYR control statement is used to specify the modules in the relocatable library that are to be copied. It is in one of the following formats.

```
COPYR module1[,module2,...]

COPYR prog1.ALL[,prog2.ALL,...]

COPYR ALL
```

The first format is used when specific modules are to be copied. The entry in the operand field is COPYR. The entry, module, in the operand field represents the name(s) of the module(s) to be copied. Entries in the operand field must be separated by commas.

The second format is used when specific programs are to 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/1.) prog is followed by a period and ALL. Entries in the operand field must be separated by commas.

The third format is used to copy the complete relocatable library. The entry in the operation field is COPYR. The entry in the operand field is ALL.

The COPYS control statement is used to specify the books in the source statement library that are to be copied. It is in one of the following formats.

```
COPYS sublib.book1[,sublib.book2,...]

COPYS sublib.ALL

COPYS ALL
```

The first format is used when specific books are to be copied. The entry in the operation field is COPYS. The qualifier sublib in the operand field represents the name of the sub-library containing the book and can be:

A for the Assembler sub-library

C for the COBOL sub-library.

book represents the name(s) of the book(s) to be copied.

The second format is used when an entire sub-library is to be copied. The entry in the operation field is COPYS. The entry, sublib, in the operand field represents the name of the sub-library to be copied and can be A or C. The qualifier sublib is followed by a period and ALL.

The third format is used to copy the complete source statement library. The entry in the operation field is COPYS. The entry in the operand field is ALL.

The COPY control statement is used to

copy the complete system. It is in the following format.

```
COPY ALL
```

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

Any number of elements of a particular library can be specified in one control statement. Continuation statements are not valid. All entries in the operand field must be separated by commas. Each library that is to be selectively copied requires a separate group of control statements. When a selective copy is performed, all elements of a particular library must be acted upon before the elements of any succeeding library.

The following functions are performed automatically by the CORGZ program:

- All programs essential to a minimum system will be copied. These programs include all logical and physical transients, IPL, Supervisor, Job Control, and Linkage Editor.
- The standard labels of track zero of the label cylinder of SYSRES will be copied to track zero of the label cylinder of SYS002.

When executing the copy disk (CORGZ) function, a file must be defined for IJSYSRES on SYS002 via VOL, DLAB, and XTENT control statements. The filename on the VOL statement must be IJSYSRES. The file identification portion of the DLAB 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 immediately after the last cylinder of the source statement library.

The following is an example of a valid job setup for the copy function:

Example:

```
                                                                    Col.
                                                                    72
//  JOB   COPY
//  ASSGN SYS002,X'191'
//  VOL   SYS002,IJSYSRES
//  DLAB  'DOS SYSTEM RESIDENCE FILE                        1111111',   C
            0001,66001,99365,'              ',SD
//  XTENT 1,000,000000001,000121009,'111111',SYS002
//  EXEC  CORGZ
    ALLOC CL=60(10),RL=30(10),SL=30(10)
    COPY  ALL
/*
/&
```

For the copy function, SYSRDR must be assigned to a card reader, a tape unit, or a disk unit. SYS002 must be assigned to a disk unit. SYSLST must be assigned to a printer, a tape unit, or a disk unit, and SYSLOG must be assigned to a printer-keyboard.

Control statement input for the copy function, read from the device assigned to SYSRDR, is as follows:

1. The JOB control statement, followed by
2. The ASSGN control statements, if the current assignments are not those required. The ASSGN statements that can be used are SYSRDR, SYS002, SYSLST, and SYSLOG. The ASSGN statements are followed by
3. VOL, DLAB, and XTENT 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.

PUNCH SERVICE FUNCTION: SPECIAL CONSIDERATIONS

As noted in the preceding sections, it is possible to perform a punch service function for the relocatable library or the source statement library and assign SYSPCH to a tape unit or disk extent. The output from the relocatable library contains not only the module(s), but also a CATALR control statement preceding each module. A /* statement follows the last module output. For the source statement library, the output contains not only the book(s), but also a CATALS and BKEND statement preceding each book and a BKEND statement following each book. A /* statement follows the last BKEND statement of the last book output.

This output can be used in two ways. A utility program (tape-to-card or disk-to-card) can be used to punch the modules or books, or the tape or disk can be used as input for a relocatable or source statement library. When the latter is true, a special control statement (IPTCTRL) is required to transfer control from SYSRDR (from which all librarian control statements are normally read) to SYSIPT (on which the librarian control statements -- CATALR or CATALS -- are actually located). This control statement is in the following format.

IPTCTRL

The operation field contains IPTCTRL; the operand field is blank.

The placement of the IPTCTRL statement in the control statement input stream depends on the requirements of the user. When the statement is read, it immediately transfers control from SYSRDR to SYSIPT. SYSIPT should be assigned temporarily for this operation when the input is on tape. When input is on disk, however, SYSIPT must be assigned permanently.

When a permanent assignment for SYSIPT is made, the IPTCTRL statement must be followed by a CLOSE command on SYSRDR that reassigns SYSIPT to an alternate device. This command can be followed in the input stream by a /& statement, if it is the last statement of the job, or by any other job control statement.

Note that if this reassignment is not made, SYSIPT is advanced to a nonexistent /& after reading the /& statement on SYSRDR.

The user also may create a tape or disk such as is output by the punch service function. An additional facility is provided allowing control to be returned to SYSRDR without ending the job step currently being processed. With a tape or disk produced by the punch service function, the last record is a /* control statement which ends the job step. Instead of the /*, the user can have a RDRCTRL statement as the last record, permitting control to return to SYSRDR. This statement is in the following format:

RDRCTRL

The operation field contains RDRCTRL; the operand field is blank.

CONDENSE MAINTENANCE FUNCTION: SPECIAL CONSIDERATIONS

The condense maintenance function can be performed automatically at the end of a catalog or delete maintenance function under the control of an installation specified parameter. The parameter is stored in the system directory. It indicates that when the number of blocks available in the corresponding library is less than the number specified by the parameter, the condense function is performed for that library. The system interrogates the parameter at the completion of each maintenance function for the library. If a condense function is to be performed, a message is printed on the printer-keyboard (SYSLOG) to inform the operator that the library is to be condensed. If multiprogramming is in progress and the core image library should be condensed, the automatic condense function will be suppressed. (Multiprogramming has no effect on a condense function for either the relocatable or source statement library.)

The CONDL control statement (as opposed to the CONDS control statement for a user-specified condense function) informs the MAINT librarian program that a parameter to specify an automatic condense is to be set. The CONDL control statement is in the following format.

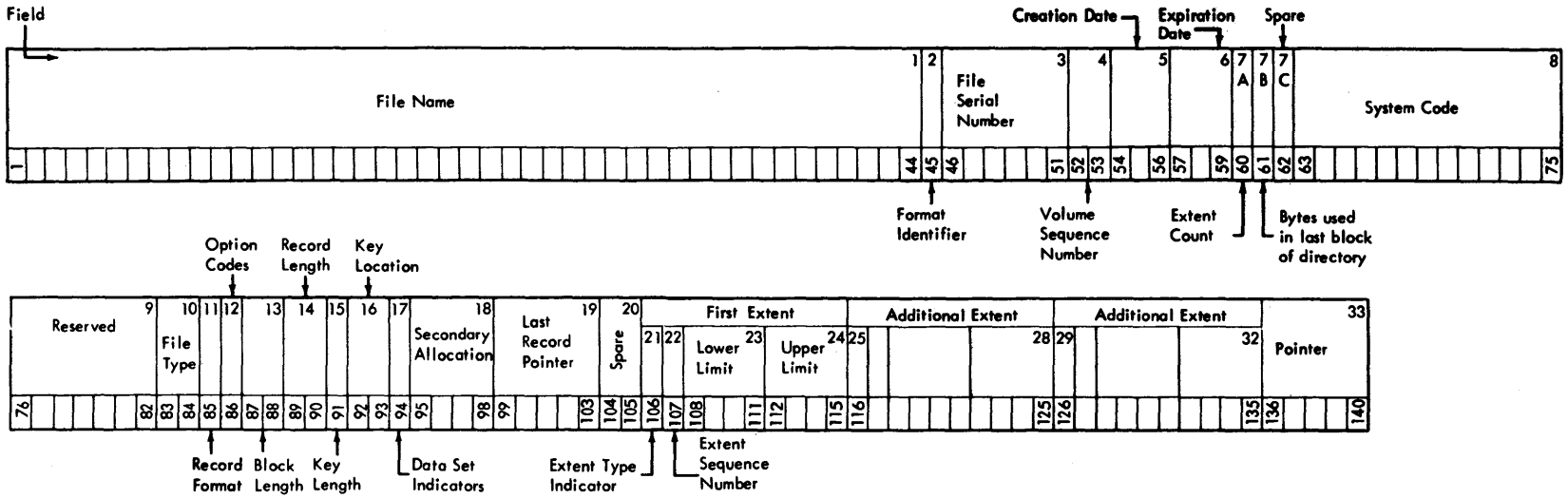
CONDL lib=nnnnn[,lib=nnnnn[,lib=nnnnn]]

The entry in the operation field is CONDL. In the operand field, the entry lib is CL for the core image library, RL for the relocatable library, and SL for the source statement library. The entry nnnnn represents the number of blocks specified for the specific library and is from one to five decimal digits. The maximum value of nnnnn is 65536. Note that each track of the core image library contains 2 blocks, each track of the relocatable library contains 9 blocks, and each track of the source statement library contains 16 blocks.

If 0 (zero) is specified for nnnnn, an automatic condense will not be performed for the specific library. If the number of blocks specified exceeds the number of blocks allocated for the library, a condense is performed each time deleted blocks appear in the library at the end of a maintenance function. When the system is copied onto another pack, the condense limit on SYSRES is also copied.

The condense limits will be displayed with the system status on a DSERV and at the end of a maintenance job.

The control statement input to establish a value for an automatic condense is the same as that for a user-specified condense. See any of the subsections entitled Condense for a description of the control statement input.



Format 1: This format is common to all data files on Direct Access Storage Devices.

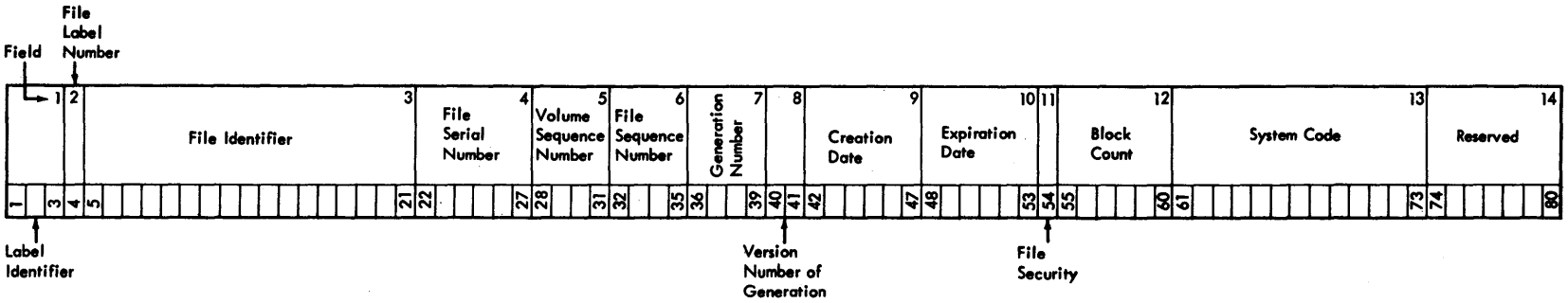
FIELD	NAME AND LENGTH	DESCRIPTION
1.	FILE NAME 44 bytes, alphameric EBCDIC	This field serves as the key portion of the file label. Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections: 1. File ID is an alphameric name assigned by the user and identifies the file. Can be 1-35 bytes if generation and version numbers are used, or 1-44 bytes if they are not used. 2. Generation Number. If used, this field is separated from File ID by a period. It has the format Gnnnn, where G identifies the field as the generation number and nnnn (in decimal) identifies the generation of the file. 3. Version Number of Generation. If used, this section immediately follows the generation number and has the format Vnn, where V identifies the field as the version of generation number and nn (in decimal) identifies the version of generation of the file.

FIELD	NAME AND LENGTH	DESCRIPTION
		Note: The Disk Operating System compares the entire field against the file name given in the DLAB card. The generation and version numbers are treated differently by Operating System/360.
	The remaining fields comprise the DATA portion of the file label:	
2.	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	1 = Format 1
3.	FILE SERIAL NUMBER 6 bytes, alphameric EBCDIC	Uniquely identifies a file/volume relationship. It is identical to the Volume Serial Number of the first or only volume of a multi-volume 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	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 Operating System/360 only for partitioned (library Structure) data sets. Not used by the Disk Operating System.
7C	SPARE 1 byte	Reserved.
8	SYSTEM CODE 13 bytes	Uniquely identifies the programming system. The character codes that can be used in this field are limited to 0-9, A-Z, or blanks.
9	RESERVED 7 bytes	Reserved.
10	FILE TYPE 2 bytes	The contents of this field uniquely identify the type of data file: Hex 4000 = Consecutive organization Hex 2000 = Direct-access organization Hex 8000 = Indexed-sequential organization Hex 0200 = Library organization Hex 0000 = Organization not defined in the file label.
11.	RECORD FORMAT 1 byte	The contents of this field indicate the type of records contained in the file: Bit Position / Content / Meaning 0 and 1 / 01 / Variable length records 10 / Fixed length records 11 / Undefined format 2 / 0 / No track overflow 1 / File is organized using track overflow (Operating System/360 only) 3 / 0 / Unblocked records 1 / Blocked records 4 / 0 / No truncated records 1 / Truncated records in file 5 and 6 / 01 / Control character ASA code 10 / Control Character machine code 00 / Control Character not stated 7 / 0 / Records have no keys 1 / Records are written with keys.
12.	OPTION CODES 1 byte	Bits within this field are used to indicate various options used in building the file. Bit 0 = If on, indicates data file was created using Write Validity Check. 1-7 = unused
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 postion 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. This bit is used by the Disk Operating System.

FIELD	NAME AND LENGTH	DESCRIPTION
		BIT 1 If on, indicates that the data set described by this file must remain in the same absolute location on the direct access device. 2 If on, indicates that Block Length must always be a multiple of 7 bytes. 3 If on, indicates that this data file is security protected; a password must be provided in order to access it. 4-7 Spare. Reserved for future use.
18.	SECONDARY ALLOCATION 4 bytes, binary	indicates the amount of storage to be requested for this data file at End of Extent. This field is used by Operating System/360 only. It is not used by the Disk Operating System routines. The first byte of this field is an indication of the type of allocation request. Hex code C2 (EBCDIC B) blocks (physical records), hex code E3 (EBCDIC T) indicates tracks, and hex code C3 (EBCDIC C) indicates cylinders. The next three bytes of this field is a binary number indicating how many bytes, tracks or cylinders are requested.
19.	LAST RECORD POINTER 5 bytes discontinuous binary	points to the last record written in a sequential or partition-organization data set. The format is TTRLL, where TT is the relative address of the track containing the last record, R is the ID of the last record, and LL is the number of bytes remaining on the track following the last record. If the entire field contains binary zeros, the last record pointer does not apply.
20.	SPARE 2 bytes	Reserved.

FIELD	NAME AND LENGTH	DESCRIPTION
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 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. 8n Shared cylinder indicator, where n 1, 2, or 4.
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 will point 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.



The standard tape file label format and contents are as follows:

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, EBCDIC	identifies the type of label HDR = Header -- beginning of a data file EOF = End of File -- end of a set of data EOV = End of Volume -- end of the physical reel
2.	FILE LABEL NUMBER 1 byte, EBCDIC	always a 1
3.	FILE IDENTIFIER 17 bytes, EBCDIC	uniquely identifies the entire file, may contain only printable characters.
4.	FILE SERIAL NUMBER 6 bytes, EBCDIC	uniquely identifies a file/volume relationship. This field is identical to the Volume Serial Number in the volume label of the first or only volume of a multi-volume file or a multi-file set. This field will normally be numeric (000001 to 999999) but may contain any six aphameric characters.
5.	VOLUME SEQUENCE NUMBER 4 bytes	indicates the order of a volume in a given file or multi-file set. The first must be numbered 0001 and subsequent numbers must be in proper numeric sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	assigns numeric sequence to a file within a multi-file set. The first must be numbered 0001.
7.	GENERATION NUMBER 4 bytes	uniquely identifies the various editions of the file. May be from 0001 to 9999 in proper numeric sequence.
8.	VERSION NUMBER OF GENERATION 2 bytes	indicates the version of a generation of a file.
9.	CREATION DATE 6 bytes	indicates the year and the day of the year that the file was created: Position / Code / Meaning 1 / blank / none 2-3 / 00-99 / Year 4-6 / 001-366 / Day of Year (e.g., January 31, 1965, would be entered as 65031).
10.	EXPIRATION DATE 6 bytes	indicates the year and the day of the year when the file may become a scratch tape. The format of this field is identical to Field 9. On a multi-file reel, processed sequentially, all files are considered to expire on the same day.
11.	FILE SECURITY 1 byte	indicates security status of the file. 0 = no security protection 1 = security protection. Additional identification of the file is required before it can be processed.
12.	BLOCK COUNT 6 bytes	indicates the number of data blocks written on the file from the last header label to the first trailer label, exclusive of tape marks. Count does not include checkpoint records. This field is used in trailer labels.
13.	SYSTEM CODE 13 bytes	uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved. Should be recorded as blanks.

PART 1. JOB CONTROL COMMANDS

ASSGN

Operand: SYSxxx,address[{,X'ss' / ,ALT}] [,TEMP]

Remarks:

SYSxxx: can be SYSRDR
SYSIPT
SYSIN
SYSLST
SYSPCH
SYSOUT
SYSLOG
SYSLNK
SYS000-SYS244

address: can be X'cuu', UA, or IGN

X'cuu': c=0-6
uu=00-FE(0-254) in hex

UA: unassign

IGN: unassign and ignore

X'ss': used for magnetic tape only

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 tape		
C0	1600	dual density 9 track tape		
C8	800	dual density 9 track tape		

ALT: specifies alternate unit

TEMP: assignment for logical unit will be destroyed by next JOB statement

CLOSE

Operand: SYSxxx[{,X'cuu'[,X'ss'] / ,UA / ,IGN / ,ALT}]

Remarks:

SYSxxx: for 2311 - SYSIN
SYSRDR
SYSIPT
SYSPCH
SYSLST

for magnetic tape -
SYSPCH
SYSLST
SYSOUT
SYS000-SYS244

X'cuu', X'ss', UA, IGN, ALT: values as described in ASSGN command

DVCDN

Operand: X'cuu'

Remarks:

X'cuu': c = 0-6
uu = 00-FE (0-254) in hex

Operation	Operand	Remarks
DVCUP	X'cuu'	X'cuu': c = 0-6 uu = 00-FE (0-254) in hex
MTC	opcode,{X'cuu' SYSxxx}[,nn]	opcode: BSF, BSR, ERG, FSF, FSR, RUN, REW, or WTM X'cuu': c = 0-6 uu = 00-FE (0-254) in hex SYSxxx: any logical unit nn: decimal number (01-99)
RESET	{SYS PROG ALL SYSxxx}	Resets specified I/O device assignments
STOP	blank	Stops background program processing
LISTIO	{SYS PROG F1 F2 ALL SYSxxx UNITS DOWN UA X'cuu'}	Causes listing of specified I/O assignments
LOG	blank	Causes logging of job control statements and foreground initiation commands on SYSLOG
NOLOG	blank	Suppresses logging of job control statements and foreground initiation commands on SYSLOG
CANCEL	blank	Cancels execution of background job
PAUSE	[any user comment]	Causes pause at end of current job step
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG
ALLOC	{F1=nK[,F2=nK] F2=nK[,F1=nK]}	Allocates foreground program areas Value of n is an even number
UCS	SYSxxx,phasename[,FOLD] [,BLOCK][,NULMSG]	Causes the 240-character universal character set contained in the core image library phase specified by phasename to be loaded as buffer storage in the IBM 2821 Control Unit. SYSxxx must be assigned to a 1403 Printer with the UCS feature.
HOLD	{F1[,F2] F2[,F1]}	Causes assignments for foreground logical units to be held across jobs.
RELSE	{F1[,F2] F2[,F1]}	Causes foreground logical units to be unassigned at EOJ.
UNA	{F1[,F2] F2[,F1]}	Causes immediate unassignment of foreground logical units.

Operation	Operand	Remarks
SET	[DATE=value1][,CLOCK=value2] [,UPSI=value3][,LINECT=value4] [,RCLST=value5][,RCPCH=value6]	value1: in one of the following formats mm/dd/yy or dd/mm/yy mm: month (01-12) dd: day (01-31) yy: year (00-99) value2: in the following format hh/mm/ss hh: hours (00-23) mm: minutes (00-59) ss: seconds (00-59) value3: 0, 1, or X value4: standard number of lines for output on each page of SYSLST value5: decimal number indicating minimum number of SYSLST disk records remaining to be written before operator warning value6: decimal number indicating minimum number of SYSPCH disk records remaining to be written before operator warning
Ⓑ	blank	Ⓑ is alter code 5

PART 2: ATTN COMMANDS

Operation	Operand	Remarks
PAUSE	[any user comments]	Causes pause at end of current job step
CANCEL	{BG / F1 / F2}	Cancels execution of current job in specified area
LOG	blank	Causes logging of job control statements and foreground initiation commands on SYSLOG
NOLOG	blank	Suppresses logging of job control statements and foreground initiation commands on SYSLOG
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG
ALLOC	{F1=nK[,F2=nK] / F2=nK[,F1=nK]}	Allocates foreground program areas Value of n is an even number
START	{BG / F1 / F2}	Initiates a background or foreground program
MSG	{F1 / F2}	Transfers control to foreground program message routine
TIMER	{BG / F1 / F2}	Causes interval timer support to be given to program specified
Ⓑ	blank	Ⓑ is alter code 5

PART 3: FOREGROUND INITIATION COMMANDS

<table>
<tr><th>Operation</th><th>Operand</th><th>Remarks</th></tr>
<tr><td>READ</td><td>X'cuu'</td><td>X'cuu': c = 0-6
uu = 00-FE (0-254) in hex

Note: Device must be a card reader</td></tr>
<tr><td>LISTIO</td><td>{BG
F1
F2
UA
ALL}</td><td>Causes listing of specified I/O assignments</td></tr>
<tr><td>ASSGN</td><td>SYSnnn,address [{,X'ss'
,ALT}]</td><td>SYSnnn: can be SYS000,SYS001,...

address: can be X'cuu' or IGN

X'cuu': c = 0-6
uu = 00-FE (0-254) in hex

IGN: unassign and ignore

X'ss': used for magnetic tape only

<table>
<tr><th>ss</th><th>Bytes per Inch</th><th>Parity</th><th>Translate Feature</th><th>Convert Feature</th></tr>
<tr><td>10</td><td>200</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>20</td><td>200</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>28</td><td>200</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>30</td><td>200</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>38</td><td>200</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>50</td><td>556</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>60</td><td>556</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>68</td><td>556</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>70</td><td>556</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>78</td><td>556</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>90</td><td>800</td><td>odd</td><td>off</td><td>on</td></tr>
<tr><td>A0</td><td>800</td><td>even</td><td>off</td><td>off</td></tr>
<tr><td>A8</td><td>800</td><td>even</td><td>on</td><td>off</td></tr>
<tr><td>B0</td><td>800</td><td>odd</td><td>off</td><td>off</td></tr>
<tr><td>B8</td><td>800</td><td>odd</td><td>on</td><td>off</td></tr>
<tr><td>C0</td><td>800</td><td colspan="3">single density 9 track tape</td></tr>
<tr><td>C0</td><td>1600</td><td colspan="3">dual density 9 track tape</td></tr>
<tr><td>C8</td><td>800</td><td colspan="3">dual density 9 track tape</td></tr>
</table>

ALT: specifies alternate unit</td></tr>
<tr><td>VOL</td><td>SYSnnn,filename</td><td>SYSnnn: can be SYS000,SYS001,...

filename: one to seven alphabetic characters</td></tr>
</table>

Operation	Operand	Remarks
DLAB	'label fields 1-3', xxxx,yyddd,yyddd,'system code' [,type]	'label fields 1-3': first three fields of Format 1 DASD file label. Is a 51-byte character string, contained within apostrophes and followed by a comma. Entire 51-byte field must be contained in the first of the two commands. A continuation character is in column 72. Field 1 is the file name (44-byte alphameric); field 2 is the format identifier (1-byte numeric); field 3 is the file serial number (6-byte alphameric). xxxx: volume sequence number (4-digit numeric). Must begin in column 16 of the continuation command. Columns 1-15 are blank. yyddd,yyddd: file creation date followed by file expiration date. Each is 5-digit numeric. 'system code': not required. When used, a 13-character string, within apostrophes. type: SD, DA, ISC, or ISE. If omitted, SD is assumed.
XTENT	type,sequence,lower,upper, 'serial no.',SYSxxx[,B_2]	type: 1 for data area (no split cylinder) 2 for overflow area (for indexed sequential file) 4 for index area (for indexed sequential file) 128 for data area (split cylinder) sequence: sequence number of extent within multi-extent file. Can be 0-255. lower: lower limit of extent in the form $B_1C_1C_1C_2C_2C_2H_1H_2H_2$ where: B_1 = 0 for 2311; 0-9 for 2321 C_1C_1 = 00 for 2311; 00-19 for 2321 $C_2C_2C_2$ = 000-199 for 2311; 000-009 for 2321 H_1 = 0 for 2311; 0-4 for 2321 H_2H_2 = 00-09 for 2311; 00-19 for 2321 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-SYS244 B_2: 0 for 2311; 0-9 for 2321

Operation	Operand	Remarks
TPLAB	'label fields 3-10'	'label fields 3-10': indicated fields of the standard tape file label. A 49-byte character string, contained within apostrophes.
TPLAB	'label fields 3-10 label fields 11-13'	'label fields 3-10: same as above label fields 11-13': 20-character direct continuation of the same character string begun with fields 3-10 (no blanks, apostrophes, or commas separating). A continuation character must be present in column 72.
CANCEL	blank	Cancels initiation of foreground program
EXEC	progname	progname: one to eight alphameric characters
LOG	blank	Causes logging of job control statements and foreground initiation commands on SYSLOG.
NOLOG	blank	Suppresses logging of job control statements and foreground initiation commands on SYSLOG.
HOLD	{F1[,F2]} {F2[,F1]}	Causes assignments for foreground logical units to be held across jobs.
RELSE	{F1[,F2]} {F2[,F1]}	Causes foreground logical units to be unassigned at EOJ.
UNA	{F1[,F2]} {F2[,F1]}	Causes immediate unassignment of foreground logical units.
MAP	blank	Causes a map of areas in main storage to be printed on SYSLOG.
MSG	{F1} {F2}	Transfers control to foreground program message routine.
PAUSE	[any operator comments]	Causes pause at end of current job step
TIMER	{BG} {F1} {F2}	Causes interval timer support to be given to the specified program.
Ⓑ	blank	Ⓑ is alter code 5

Name	Operation	Operand	72	Remarks
//	JOB	jobname	⌀	jobname: one to eight alphameric characters
//	EXEC	[progname]	⌀	progname: one to eight alphameric characters. Used only if the program is in the core image library.
//	ASSGN	SYSxxx,address [{,X'ss' / ,ALT}]	⌀	SYSxxx: can be SYSRDR SYSIPT SYSIN SYSPCH SYSLST SYSLOG SYSLNK SYS000-SYS244 address: can be X'cuu', UA, or IGN X'cuu': c = 0-6 uu = 00=FE (0-254) in hex UA: unassign IGN: unassign and ignore X'ss': used for magnetic tape only

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 tape		
C0	1600	dual density 9-track tape		
C8	800	dual density 9-track tape		

ALT: specifies alternate unit

//	RESET	{SYS / PROG / ALL / SYSxxx}	⌀	Resets I/O device assignments
//	DATE	mm/dd/yy or dd/mm/yy	⌀	mm: month (01-12) dd: day (01-31) yy: year (00-99)
//	UPSI	nnnnnnnn	⌀	n: 0, 1, or X
//	VOL	SYSxxx,filename	⌀	SYSxxx: can be SYS000-SYS244 filename: one to seven alphabetic characters

Name	Operation	Operand	72	Remarks
//	DLAB	'label fields 1-3' xxxx,yyddd,yyddd, 'system code'[,type]	C	'label fields 1-3': first three fields of Format 1 DASD file label. Is a 51-byte character string, contained within apostrophes and followed by a comma. Entire 51-byte field must be contained in the first of the two statements. Field 1 is the file name (44-byte alphameric); field 2 is the format identifier (1-byte numeric); field 3 is the file serial number (6-byte alphameric). C: any non-blank 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.
//	XTENT	type,sequence,lower, upper,'serial no.', SYSxxx[,B_2]	Ø	type: 1 for data area (no split cylinder) 2 for overflow area (for indexed sequential file) 4 for index area (for indexed sequential file) 128 for data area (split cylinder) sequence: sequence number of extent within multi-extent file. Can be 0 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; 0-9 for 2321 C_1C_1 = 00 for 2311; 0-19 for 2321 $C_2C_2C_2$ = 000-199 for 2311; 000-009 for 2321 H_1 = 0 for 2311; 0-4 for 2321 H_2H_2 = 00-09 for 2311; 00-19 for 2321 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-SYS244 B_2: 0 for 2311; 0-9 for 2321

Name	Operation	Operand	72	Remarks
//	TPLAB	'label fields 3-10'	⌀	'label fields 3-10': Indicated fields of the standard tape file label. A 49-byte character string, contained within apostrophes.
//	TPLAB	'label fields 3-10 label fields 11-13'	C	'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)
//	LBLTYP	{ TAPE[(nn)] } { NSD(nn) }	⌀	TAPE: used when tape files requiring label information are to be processed and no nonsequential disk files are to be processed. (nn): optional and is present only for future expansion (it is ignored by Job Control) NSD: nonsequential disk files are to be processed (nn): largest number of extents per single file
//	RSTRT	SYSxxx,nnnn	⌀	SYSxxx: symbolic unit name of the device on which the checkpoint records are stored. Can be SYS000-SYS244. nnnn: four character identification of the checkpoint record to be used for restarting
//	LISTIO	{ SYS PROG F1 F2 ALL SYSxxx UNITS DOWN UA X'cuu' }	⌀	Causes listing of I/O assignments on SYSLST
//	MTC	opcode,SYSxxx[,nn]	⌀	opcode: BSF, BSR, ERG, FSF, FSR, REW, RUN, or WTM SYSxxx: any logical unit nn: decimal number (01-99)

Name	Operation	Operand		Remarks
//	OPTION	option1[,option2,...]	ø	option: can be any of the following LOG Log control statements on SYSLST NOLOG Suppress LOG option DUMP Dump registers and main storage on SYSLST in the case of abnormal program end NODUMP Suppress DUMP option LINK Write output of language translator on SYSLNK for linkage editing NOLINK Suppress LINK option DECK Output object module on SYSPCH NODECK Suppress DECK option LIST Output listing of source module on SYSLST NOLIST Suppress LIST option LISTX Output listing of object module on SYSLST NOLISTX Suppress LISTX option SYM Punch symbol deck on SYSPCH NOSYM Suppress SYM option XREF Output symbolic cross-reference list on SYSLST NOXREF Suppress XREF option ERRS Output listing of all errors in source program on SYSLST NOERRS Suppress ERRS option CATAL Catalog program or phase in core image library after completion of Linkage Editor run STDLABEL Causes all sequential disk or tape labels to be written on the standard label track USRLABEL Causes all sequential disk or tape labels to be written on the user label track 48C 48-character set 60C 60-character set
//	PAUSE	[comments]	ø	PAUSE statement is always printed on 1052 (SYSLOG). If no 1052 is available, the statement is ignored.
/*	ignored	ignored	ø	Columns 1 and 2 are the only columns checked.
/&	ignored	ignored	ø	Columns 1 and 2 are the only columns checked.
*		comments	ø	Column 2 must be blank.

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

The format of the ESD card follows:

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 to indicate SD, PC, LD, CM or ER 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.

The format of the TXT card follows:

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.

The format of the RLD card follows:

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 - pointer (relocation identifier) to the relocation factor of the contents of the load constant. b. Two positions - pointer (position identifier) to the relocation factor of the control section in which the load constant occurs. c. One position - flag indicating type of constant, as follows:

Bits	
0-2	ignored
3	0 - a non-branch 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.

The format of the END card follows:

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.

The format of the REP (user replace card) follows:

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 F: LINKAGE EDITOR INPUT RESTRICTIONS

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 will normally be incorporated into a unique ESD item. However, if the same symbol occurs in different phases (e.g., control sections), each resolved occurrence of the symbol within a different phase is a unique ESD item.

The number of ESID items is the number of ESD items within a module not counting the LD (source ENTRY) type.

An RLD item is related to the address constants within a phase. An address constant can have one or more RLD items, each item corresponding to a symbol in the relocatable expression of an address constant. For example, A(X + Y) is an address constant of two relocatable expressions that will have two RLD items.

The following formula can be used for determining the minimum number of PHASE, ESD, and ESID items that can be processed in the control dictionary of a 16K system.

$$16(x + y) + 3z \leq 4000$$

where x = total number of PHASE statements, not to exceed 120
y = total number of unique ESD items
z = total number of ESID items per module.

A background area greater than 10K but less than 14K does not change this formula. However, if the background program area is greater than 14K, the Linkage Editor uses additional storage to increase the dictionary area. The value 4000 in the formula for a background area greater than 14K increases linearly to a value of 33,500 at a background area of 40K.

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.

```
  SOURCE STATEMENT

         PRINT NOGEN
PROGRAM  START 0
         BALR  15,0
         USING *,15
*     ROUTINE TO RELOCATE ADDRESS CONSTANTS
         LA    1,PRINTCCW              RELOCATE CCW ADDRESS
         ST    1,PRINTCCB+8             IN CCB FOR PRINTER
         LA    1,TAPECCW               RELOCATE CCW ADDRESS
         ST    1,TAPECCB+8              IN CCB FOR INPUT TAPE
         LA    1,EOFTAPE              *RELOCATE*****
         ST    1,AEOFTAPE             * PROGRAM    *
         LA    1,CHA12                *  ADDRESS   *
         ST    1,ACHA12               ****CONSTANTS*
         IC    2,PRINTCCW              SAVE PRINT CCW OP CODE
         LA    1,OUTAREA               RELOCATE OUTPUT AREA ADDRESS
         ST    1,PRINTCCW               IN PRINTER CCW
         STC   2,PRINTCCW              RESTORE PRINT CCW OP CODE
         LA    1,INAREA                RELOCATE INPUT AREA ADDRESS
         ST    1,TAPECCW                IN TAPE CCW
         MVI   TAPECCW,2               SET TAPE CCW OP CODE TO READ
*     MAIN ROUTINE...READ TAPE AND PRINT RECORDS
READTAPE LA    1,TAPECCB               GET CCB ADDRESS
         EXCP  (1)                     READ ONE RECORD FROM TAPE
         WAIT  (1)                     WAIT FOR COMPL. OF I/O
         L     10,AEOFTAPE             GET ADDRESS OF TAPE EOF ROUTINE
         BAL   14,CHECK                GO TO UNIT EXCEPTION SUBROUTINE
         MVC   OUTAREA(10),INAREA      EDIT RECORD
         MVC   OUTAREA+15(70),INAREA+10  IN
         MVC   OUTAREA+90(20),INAREA+80  OUTPUT AREA
         LA    1,PRINTCCB              GET CCB ADDRESS
         EXCP  (1)                     PRINT EDITED RECORD
         WAIT  (1)                     WAIT FOR COMPL. OF I/O
         L     10,ACHA12               GET ADDRESS OF CHAN 12 ROUTINE
         BAL   14,CHECK                GO TO UNIT EXCEPTION SUBROUTINE
         B     READTAPE
CHECK    TM    4(1),1                  CHECK FOR UNIT EXC. IN CCB
         BCR   1,10                     YES-GO TO PROPER ROUTINE
         BR    14                       NO-RETURN TO MAINLINE
CHA12    MVI   PRINTCCW,X'8B'          SET SK TO CHAN 1 OP CODE
         EXCP  (1)                     SK TO CHAN 1 IMMEDIATELY
         WAIT  (1)                     WAIT FOR COMPL. OF I/O
         MVI   PRINTCCW,9              SET PRINTER OP CODE TO WRITE
         BR    14                      RETURN TO MAINLINE
EOFTAPE  EOJ                           END OF JOB
         CNOP  0,4                     ALIGN CCB'S TO FULL WORD
PRINTCCB CCB   SYS004,PRINTCCW,X'0400'
TAPECCB  CCB   SYS001,TAPECCW
PRINTCCW CCW   9,OUTAREA,X'20',110
TAPECCW  CCW   2,INAREA,X'20',100
AEOFTAPE DC    A(EOFTAPE)
ACHA12   DC    A(CHA12)
OUTAREA  DC    CL110' '
INAREA   DC    CL100' '
         END   PROGRAM
```

APPENDIX H: LINKAGE EDITOR DIAGNOSTICS

All I-type diagnostic messages are output on SYSLST if the MAP option is assumed or specified. If NOMAP is operational, all I-type messages are output on SYSLOG. In addition, diagnostic messages 2171I to 2195I are output on SYSLOG if MAP is operational.

Message 2199I is output on SYSLOG to inform the operator that errors occurred in the Linkage Editor input and were output on SYSLST (if MAP is operational). When these errors (2100I through 2170I) occur, the job continues unless ACTION CANCEL is specified.

The error diagnostic listing on SYSLST is intended to provide the programmer with a complete summary of user-prepared input, the names of the modules AUTOLINKed from the relocatable library, and all error messages.

Number	Message	Cause	Action
2100I	Content of statement in error. See the note following 2170I.	Invalid input card type.	Correct the statement in error and resubmit the job.
2101I	Content of statement in error. See the note following 2170I.	Invalid operation in control statement.	Correct the statement in error and resubmit the job.
2102I	Content of statement in error. See the note following 2170I.	Non-decimal or non-hexadecimal character in decimal or hexadecimal field.	Correct the statement in error and resubmit the job.
2110I	Content of statement in error. See the note following 2170I.	Invalid or missing field delimiter on control statement.	Correct the statement in error and resubmit the job.
2111I	Content of statement in error. See the note following 2170I.	An operand field is greater than the maximum length on a user-prepared control statement or REP card.	Correct the statement in error and resubmit the job.
2112I	Content of statement in error. See the note following 2170I.	An operand field is missing.	Correct the statement in error and resubmit the job.
2113I	Content of statement in error. See the note following 2170I.	Control statement extends beyond column 71.	Correct the statement in error and resubmit the job.
2114I	Content of statement in error. See the note following 2170I.	Sub-modular namelist is too long.	Correct the statement in error and resubmit the job.
2115I	Content of statement in error. See the note following 2170I.	NOAUTO expected, but not found.	Correct the statement in error and resubmit the job.

Number	Message	Cause	Action
2116I	Content of statement in error. See the note following 2170I.	Control statement present between first ESD and END statements of a module.	Correct the statement in error and resubmit the job.
2120I	Content of statement in error. See the note following 2170I.	Phase name duplicated.	Correct the statement in error and resubmit the job.
2121I	Content of statement in error. See the note following 2170I.	Phase name lower in sequence than $$A or phase name begins with *.	Correct the statement in error and resubmit the job.
2122I	Content of statement in error. See the note following 2170I.	Symbol or phasename, designated in origin, not previously defined.	Correct the statement in error and resubmit the job.
2123I	Content of statement in error. See the note following 2170I.	Previous phase processed contained no valid storage assignment.	Correct the statement in error and resubmit the job.
2124I	Content of statement in error. See the note following 2170I.	Phase origin is negative.	Correct the statement in error and resubmit the job.
2125I	Content of statement in error. See the note following 2170I.	PHASE statement encountered during AUTOLINK.	Correct the statement in error and resubmit the job.
2130I	Content of statement in error. See the note following 2170I.	Relocatable library not present.	Specify that the correct system pack be used when the job is resubmitted.
2131I	Content of statement in error. See the note following 2170I.	Module requested by INCLUDE statement not present in the relocatable library.	Catalog the requested module in the relocatable library and resubmit the job.
2132I	Content of statement in error. See the note following 2170I.	Too many nesting levels of INCLUDE attempted.	Correct the statement in error and resubmit the job.
2133I	Content of statement in error. See the note following 2170I.	Nested sub-modular INCLUDE.	Correct the statement in error and resubmit the job.
2135I	Content of statement in error. See the note following 2170I.	ACTION statement has invalid operand.	Correct the statement in error and resubmit the job.
2136I	Content of statement in error. See the note following 2170I.	ACTION MAP specified, but SYSLST was not assigned.	Assign SYSLST when the job is resubmitted.

Number	Message	Cause	Action
2140I	Content of statement in error. See the note following 2170I.	ESD item of invalid type.	Correct the statement in error and resubmit the job.
2141I	Content of statement in error. See the note following 2170I.	Duplicated ESID number: 1. No END statement in last module. 2. Duplicates ESD card. 3. Extraneous ESD card.	Correct the statement in error and resubmit the job.
2142I	Content of statement in error. See the note following 2170I.	ESD entry point label does not point to ESD named control section or COMMON.	Correct the statement in error and resubmit the job.
2143I	Content of statement in error. See the note following 2170I.	Invalid duplication of entry point label.	Correct the statement in error and resubmit the job.
2144I	Content of statement in error. See the note following 2170I.	Invalid ESID number, of control dictionary and linkage table overlap (see Appendix F).	Correct the statement in error and resubmit the job.
2145I	Content of statement in error. See the note following 2170I.	Origin of control section not on a double-word boundary.	Correct the statement in error and resubmit the job.
2146I	Content of statement in error. See the note following 2170I.	COMMON has the same label as a named control section or an entry point label.	Correct the statement in error and resubmit the job.
2147I	Content of statement in error.	ESD entry point label does not belong to a defined control section.	Correct the statement in error and resubmit the job.
2150I	Content of statement in error. See the note following 2170I.	Load address encountered outside phase.	Correct the statement in error and resubmit the job.
2151I	Content of statement in error. See the note following 2170I.	Invalid delimiter on REP card.	Correct the statement in error and resubmit the job.
2155I	Content of statement in error. See the note following 2170I.	Address constant does not belong to any control section defined.	Correct the statement in error and resubmit the job.
2156I	Content of statement in error See the note following 2170I.	Invalid format of RLD card.	Correct the statement in error and resubmit the job.
2158I	Content of statement in error. See the note following 2170I.	END statement should contain the length of the control section, but does not.	Correct the statement in error and resubmit the job.
2170I	Content of statement in error. See the following note.	ESID number not previously processed.	Correct the statement in error and resubmit the job.

Note: Statements in error (Error Numbers 2100I to 2170I) are printed in the following formats:

1. If no 12-2-9 code in column 1 of card image, columns 2-80 of card image printed in EBCDIC.

2. If 12-2-9 code in column 1 of card image:

Print Positions	Contains Card Image Columns
8-15	73-80 (identification) in EBCDIC
17-19	2-4 (card type) in EBCDIC
21-26	6-8 (assembled origin) in hexadecimal
28-31	11-12 (number of bytes in card image) in hexadecimal
33-36	15-16 (ESID number) in hexadecimal

Remainder of line depends on type of card image (ESD or non-ESD).

If non-ESD type card image, print positions 38-128 are printed in columns 17-52. Printed in hexadecimal in blocks of 9 words (36 bytes), separated by one blank.

If ESD type card image, print positions 38-128 contain 3 fields of ESD information. Each field is 16 columns, broken down as follows.

Columns	Contains
17-24	ESD item name in EBCDIC
25	ESD type in EBCDIC
26-28	Assembled origin in hexadecimal
30-32	Length/ESID number in hexadecimal

Number	Message	Cause	Action
2181I	LINKAGE EDITOR CANNOT CONTINUE	No valid storage assignment in final phase.	Correct previous errors and resubmit the job.
2182I	LINKAGE EDITOR CANNOT CONTINUE	No END record encountered before ENTRY statement.	Supply END record and resubmit the job.
2185I	LINKAGE EDITOR CANNOT CONTINUE	An error occurred during the linkage editing of a $ phase.	Correct the errors and resubmit the job.
2191I	LINKAGE EDITOR CANNOT CONTINUE	1. End of file or extents on SYS001. 2. SYS001 not assigned to disk or tape.	Correct the SYS001 usage and resubmit the job.

Number	Message	Cause	Action
2192I	LINKAGE EDITOR CANNOT CONTINUE	End of Librarian work area. Too many phases to process.	Break program down into less than 120 phases and resubmit the job.
2193I	LINKAGE EDITOR CANNOT CONTINUE	Core image Library capacity reached.	Condense the core image library or reallocate limits to create enough room, and resubmit the job.
2194I	LINKAGE EDITOR CANNOT CONTINUE	Disk error. Invalid no-record-found condition.	Resubmit the job.
2195I	LINKAGE EDITOR CANNOT CONTINUE	Multiprogramming in process while attemping to Linkage edit and catalog a new supervisor.	Resubmit the job
2199I	ERROR HAS OCCURRED DURING LINKAGE EDITING	Printed on SYSLOG if any of the errors 2100I through 2170I have occurred.	None.

Number	Message	Cause	Action
3C10I	INVALID CONTROL CARD	Card read is not ALLOC, COPY, COPYC, COPYR, or COPYS.	Card is ignored. Program continues.
3C20I	ALLOCATION SPECIFIED TWICE FOR THE ------ LIBRARY	More than one ALLOC statement received for this library.	Additional card is ignored. Program continues.
3C21I	INVALID OPERAND	The operand in an ALLOC or COPY-type statement is invalid.	On a COPY-type statement, the operand is ignored and the remainder of the card is processed. On an ALLOC statement, correct the operand and resubmit the job.
3C30I	CARD OUT OF ORDER	ALLOC card received after COPY, COPYC, COPYR, or COPYS card.	Card is ignored. Program continues.
3C33I	----- NOT IN LIBRARY	The name requested is not found in the requested library.	Operand is ignored. Remainder of card is processed.
3C40I	NO { CORE IMAGE / RELOCATABLE / SOURCE STATEMENT } LIBRARY ALLOCATED	A request was made of a library but no library is available.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C60I	{ CORE IMAGE / RELOCATABLE / SOURCE STATEMENT } LIBRARY ALLOCATED BUT NO DIRECTORY	An allocation for the library was received, but not for the directory.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C61I	ZERO ALLOCATION SPECIFIED FOR CORE IMAGE LIBRARY	No allocation for the Core Image Library was received.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C62I	TRACKS EXCEED CYLINDERS IN { CORE IMAGE / RELOCATABLE / SOURCE STATEMENT } LIBRARY	The number of tracks allocated for the directory exceeds the total cylinder allotment for the directory/library.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.

Number	Message	Cause	Action
3C63I	{CORE IMAGE / RELOCATABLE / SOURCE STATEMENT} DIRECTORY OVERFLOW	Insufficient number of tracks have been allocated for this directory.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C64I	{CORE IMAGE / RELOCATABLE / SOURCE STATEMENT} LIBRARY OVERFLOW	Insufficient number of cylinders have been allocated for this library.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C65I	INVALID EXTENTS DEFINED FOR SYS002	The extents defined do not cover the entire file.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3C66I	FILE IJSYSRES NOT DEFINED ON SYS002	The file IJSYSRES has been defined, but not on SYS002.	Correct the file definition or the allocation for the output file (SYS002) and resubmit the job.
3D10D	INVALID OPERATION	Operation field of control statement contains something other than DSPLY.	Correct the control statement and resubmit the job.
3D20D	INVALID OPERAND	Operand field of control statement contains something other than CD, RD, SD, TD, or ALL.	Correct the control statement and resubmit the job.
3D43I	RELOCATABLE LIBRARY HAS NO ENTRIES	(1) No active entries in relocatable library, or (2) No relocatable library is present on the disk and either RD or ALL appeared in the control statement.	None.
3D47I	SOURCE STATEMENT LIBRARY HAS NO ENTRIES	(1) No active entries in source statement library, or (2) No source statement library is present on the disk and either SD or ALL appeared in the control statement.	None.
3M10D	INVALID OPERATION	Operation field of control statement contains something other than CATALR, CATALS, DELETC, DELETR, DELETS, RENAMC, RENAMR, RENAMS, CONDS, ALLOC, IPTCTRL, or RDRCTRL.	Correct the control statement and resubmit the job.

Number	Message	Cause	Action
3M11D	INVALID CARD IN MODULE	Indicates that the module to be cataloged in the relocatable library contains an invalid statement. Valid statements are in one of two formats: 1. 12-2-9 code in column 1 of an 80-byte record, or in column 2 of an 81-byte record. These records may be of the ESD, RLD, TXT, REP, END or SYM types. 2. A record with a blank in column 1 of an 80-byte record, or in column 2 of an 81-byte record. Any combination of valid characters can follow.	Correct the statement in error and resubmit the module to be cataloged.
3M20D	INVALID OPERAND	Blank or invalid operand field following a CATALR or CATALS statement.	Indicate the name of module or book to be cataloged and resubmit the job.
3M21I	INVALID OPERAND	Blank or invalid operand field following DELETC, DELETR, DELETS, RENAMC, RENAMR, or RENAMS statement	Indicate the name(s) of the program(s) module(s), or book(s) to be deleted or renamed and resubmit the job.
3M22I	PHASE xxx INVALID PHASE NAME PROGRAM NOT CATALOGED	Name of the phase to be cataloged was in error or not given in a phase statement. The program containing the phase is not cataloged in the core image library.	Add PHASE card or change phase name and resubmit the job.
3M23D	xxxxxxxx INVALID OPERAND -- BKEND OPERAND n	BKEND statement contains invalid entry xxxxxxxx as operand n within the statement.	Correct the BKEND statement and resubmit the job.
3M24I	xxxxxxxx {CATALOGED / DELETED / RENAMED} IN SUBLIBRARY x	Book xxxxxxxx was missing its sublibrary identification (A for Assembler; C for COBOL) and was cataloged/deleted/renamed in sublibrary x (A or C).	None.
3M25D	ERROR IN CARD SEQUENCE NO. -- CARD NO. xxxxx	Card out of sequence in book to be cataloged in source statement library.	Sequence the book correctly and resubmit the job.

Number	Message	Cause	Action
3M26D	ERROR IN CARD COUNT -- ACTUAL COUNT xxxx	Card count in the BKEND statement does not correspond to actual count of cards, including BKEND statements.	Correct the operand of the BKEND statement and resubmit the job.
3M33I	xxxxxxxx NOT IN LIBRARY	The phase, module, or book to be deleted or renamed did not exist in the library.	None.
3M34I	EOF ON SYSIPT -- END STATEMENT MISSING	Module to be cataloged did not have an END statement.	Supply an END statement and resubmit the job.
3M35D	NO HEADER BKEND OR MACRO STATEMENT	First statement of a book to be cataloged in source statement library is not a BKEND or MACRO statement.	Supply the appropriate statement(s) (BKEND and/or MACRO) and resubmit the job.
3M43I	NO { RELOCATABLE / SOURCE STATEMENT } LIBRARY	The library called for does not exist.	None.
3M52I	{ RELOCATABLE / CORE IMAGE / SOURCE STATEMENT } DIRECTORY IS FULL	Not enough space in directory when trying to catalog.	Reallocate for directory.
3M53I	{ RELOCATABLE / SOURCE STATEMENT } LIBRARY IS FULL	Not enough space for new entry in library when trying to catalog.	Reallocate for library.
3M54I	xxxxxxxx ALREADY IN LIBRARY	The phase, module, or book to be renamed is already in the library.	Rename and resubmit job.
3M60I	ZERO ALLOCATION SPECIFIED FOR CORE IMAGE LIBRARY	Allocation of zero cylinders was made for the core image library in an ALLOC statement.	Correct ALLOC statement and resubmit the job.

Number	Message	Cause	Action
3M61I	INVALID ZERO ALLOCATION	Only one of a pair of parameters is zero.	Correct ALLOC statement and resubmit job.
3M62I	TRACKS FOR DIRECTORY EXCEED CYLINDERS FOR LIBRARY	Number of tracks specified for a directory is greater than the number of tracks contained in the cylinders specified for the corresponding library.	Correct ALLOC statement and Resubmit the job.
3M63I	{CORE IMAGE / RELOCATABLE / SOURCE STATEMENT} DIRECTORY ALLOCATION IS TOO SMALL	Specified allocation is too small to contain information existing in {Core Image / Relocatable / Source Statement} directory.	Increase specified directory allocation and resubmit job.
3M64I	{CORE IMAGE / RELOCATABLE / SOURCE STATEMENT} LIBRARY ALLOCATION IS TOO SMALL	Specified allocation is too small to contain information existing in {Core Image / Relocatable / Source Statement} library.	Increase specified library allocation and resubmit job.
3M65I	ALLOCATION EXCEEDS SYSRES EXTENT	Parameter on ALLOC statement requires a larger extent than provided by XTENT card for SYSRES.	Correct ALLOC or XTENT statement and resubmit job.
3M66I	BEGIN REALLOCATION	System is not operable until reallocation is finished.	None.
3M67I	END REALLOCATION	Reallocation is finished and system is now operable.	None.
3M68I	STATEMENT IGNORED DUE TO MULTIPROGRAMMING IN PROGRESS	User asked for a condense or reallocation of the core image library when multiprogramming (F1 or F2) was active.	Wait for F1 and/or F2 to finish, and resubmit the job.
3M69I	STATEMENT IGNORED DUE TO CONTROL PROGRAM BEING CATALOGED	User asked for a condense of the core image library after cataloging the Supervisor in the same job.	Submit the condense as a different job.
3M70A	POTENTIAL DISASTER ERROR. REBUILD SYSTEM	Disk error on SYSRES.	Rebuild system and re-IPL.
3R10D	INVALID OPERATION	Operation field of control statement contains something other than DSPLY, PUNCH, or DSPCH.	Correct the control statement and resubmit the job.

Number	Message	Cause	Action
3R21I	INVALID OPERAND	Operand field of control statement contains something other than a valid entry. Valid operands are an alphameric module name and ALL.	Correct the control statement and resubmit the job.
3R27I	xxxxxxxx NOT IN LIBRARY	Module xxxxxxxx is not present in the relocatable library.	None.
3R43I	NO RELOCATABLE LIBRARY	(1) A relocatable library is not defined on the disk pack, or (2) The relocatable library contains no active entries.	None.
3R44I	RELOCATABLE LIBRARY HAS NO ACTIVE ENTRIES	Request made of an empty relocatable library.	Job is canceled.
3S10D	INVALID OPERATION	Operation field of control statement contains something other than DSPLY, PUNCH, or DSPCH.	Correct the control statement and resubmit the job.
3S21I	INVALID OPERAND	Operand field of control statement contains something other than a valid entry. Valid operands are the sublibrary name (A for Assembler and C for COBOL) followed by a period and the book to be operated upon, the sublibrary name followed by a period and ALL, and CMPRSD as the last operand of the statement.	Correct the control statement and resubmit the job.
3S33I	x.xxxxxxxx NOT IN LIBRARY x. ALL NOT IN LIBRARY	1. Book xxxxxxxx in sublibrary x not present in the source statement library. 2. Sublibrary x not present in source statement library.	None.
3S43I	NO SOURCE STATEMENT LIBRARY	1. A source statement library is not defined on the disk pack, or 2. The source statement library contains no active entries.	None.

Number	Message	Cause	Action
0P70I	JOB xxxxxxxx CANCELLED DUE TO UNDEFINED LOGICAL UNIT	Program issued an EXCP for a logical unit for which there is no LUB. If a dump is taken, general register 1 will contain a pointer to the CCB in question.	Resubmit the job.
0P71I	JOB xxxxxxxx CANCELLED DUE TO DEVICE NOT ASSIGNED	Program issued an EXCP for a logical unit which is not assigned to a physical device. If a dump is taken, general register 1 will contain a pointer to the CCB in question.	Resubmit the job.
0P72I	JOB xxxxxxxx CANCELLED DUE TO READING PAST /& STATEMENT	Program ignored the occurrence of a /& (end-of-job) statement on SYSRDR or SYSIPT.	Resubmit the job.
0P73I	JOB xxxxxxxx CANCELLED DUE TO I/O ERROR	Program does not accept I/O error.	Resubmit the job.
0P74I	JOB xxxxxxxx CANCELLED DUE TO I/O OPERATOR OPTION	Operator typed CANCEL on 1052 in response to an I/O error message, or X'03' was inserted in main storage byte 4.	Resubmit the job.
0P75I	JOB xxxxxxxx CANCELLED DUE TO I/O ERROR QUEUE OVERFLOW	Number of I/O errors pending simultaneously has exceeded Supervisor capacity.	Resubmit the job.
0P76I	JOB xxxxxxxx CANCELLED DUE TO INVALID DASD ADDR	1. DASD file protect limits exceeded. 2. Incorrect record reference for SYSIN or SYSOUT on 2311.	Resubmit the job.
0P77I	JOB xxxxxxxx CANCELLED DUE TO INVALID ADDRESS	Address parameter given by problem program refers to an address outside main storage, or outside the requester's area (background or foreground).	Resubmit the job.
0P78I	JOB xxxxxxxx CANCELLED DUE TO UNRECOGNIZED CANCEL-CODE	An IBM-supplied component failed to post a valid CANCEL-Code.	Resubmit the job.
0P79I	JOB xxxxxxxx CANCELLED DUE TO NO LONG SEEK	A DASD command chain in a a file protected environment does not start with a command code X'07'.	Resubmit the job.

Number	Message	Cause	Action
0S00I	JOB xxxxxxxx CANCELLED	Error in problem program caused job termination.	Resubmit the job.
0S01I	JOB xxxxxxxx CANCELLED DUE TO OPERATOR INTERVENTION	Operator typed CANCEL on 1052.	Resubmit the job.
0S02I	JOB xxxxxxxx CANCELLED DUE TO PROGRAM REQUEST	CANCEL macro instruction issued by problem program.	Resubmit the job.
0S03I	PROGRAM CHECK INTERRUPTION - HEX LOCATION nnnnnn - CONDITION CODE m - interruption cause	Program check interruption caused job termination.	Resubmit the job.
0S04I	ILLEGAL SVC - HEX LOCATION nnnnnn - SVC CODE nn	When nn is 02: 1. The phase name given does not start with $$B. 2. An SVC 8 has been issued from the logical transient area, and the SVC 2 is given from the problem program area before an SVC 9 has been given. --- When nn is 05: 1. The "to"-range specified in the MVCOM macro instruction is invalid, or 2. MVCOM was issued by a foreground program. --- When nn is 11: The call was not given by a logical transient routine. --- When nn is 22, 23, or 26: The caller did not have a PSW key of zero. This is applicable only in a multiprogramming system.	Resubmit the job.

Number	Message	Cause	Action
		When nn is 32: 1. For LIOCS - an imperative macro (such as WRITE or PUT) was issued to a module that does not contain the requested function. 2. For LIOCS - an invalid ASA first character forms control character for the printer was used. 3. For COBOL object programs - a wrong length record was detected. When nn is any other value: The Supervisor function requested by the operand of the SVC is not defined for the Supervisor being used.	
0S05I	PHASE xxxxxxxx NOT FOUND	Phase named in a FETCH (SVC 1) or LOAD (SVC 4) macro instruction, or referred to by an SVC 2, or SVC 3 cannot be found.	Resubmit the job.
0S06I	JOB xxxxxxxx CANCELLED DUE TO PHASE NOT FOUND	See message 0S05I. This message is always preceded by message 0S08I.	Resubmit the job.
0S07I	PROBLEM PROGRAM PSW nnnnnnnnnnnnnnnn	Gives the condition of the problem program immediately before its cancelation. Message 0S07I is printed in conjunction with a descriptive cancelation message.	Resubmit the job.

Number	Message	Cause	Action
0S08I	LOG. TRANS. AREA CANCELLED	Indicates that the cancellation described by an associated message occurred while a logical transient routine was executing. Further details such as phase name, hex location, SVC code, condition code, and interruption cause are not available when cancellation occurs in a logical transient routine.	Resubmit the job.
0S09I	JOB xxxxxxxx CANCELLED DUE TO ILLEGAL SVC	See message 0S04I. This message is always preceded by message 0S08I.	Resubmit the job.
0S10I	PROGRAM xxxxxxxx COMPLETED	Message issued at the normal completion of a foreground program.	None.
0S11I	JOB xxxxxxxx CANCELLED DUE TO PROGRAM CHECK	See message 0S03I. This message is always preceded by message 0S08I.	Resubmit the job.

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

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

IBM **Technical Newsletter**

File No. S360-36 (DOS)

Re: Form No. C24-5036-1

This Newsletter No. N24-5248

Date: 6/2/67

Previous Newsletter Nos. N24-5218
N24-5289

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

This Technical Newsletter supports an intermediate storage size (24K) for System/360, Model 30.

The pages attached to this Newsletter replace the following pages of your publication:

5 - 7
51 and 52

Changes on modified pages are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (●) next to a figure title or page number indicates that the entire figure or page should be reviewed. Please insert this page to indicate that your publication now includes the following modified pages:

Modified Pages

6, 7, 51

IBM Corp., Programming Publications Dept., Endicott, N. Y. 13760.

PRINTED IN U.S.A.

IBM **Technical Newsletter**

System S360-36 (DOS)

Re: Form No. C24-5036-1

This Newsletter No. N24-5218

Date: March 17, 1967

Previous Newsletter Nos. N24-5214,
obsoleted by N24-5218

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

This Technical Newsletter reflects the availability of Disk Operating System support for the 1285 Optical Reader, and provides information on the Queued Telecommunications Access Method (QTAM).

The pages attached to this Newsletter replace the following pages of your publication:

Replace Pages

1 and 2
7 and 8
19 and 20
31 and 32
45 - 48
51 and 52
55 - 58
113 and 114
117 and 118
131 and 132
135 and 136

Changes on modified pages are indicated by a vertical line to the left of the affected text and to the left of the affected parts of figures. A dot (●) next to a figure title or a page number indicates that the entire figure or page should be reviewed. Please insert this page to indicate that your publication now includes the following modified pages:

Modified Pages

1, 2, 7, 7A, 8, 20, 32,
46, 47, 51, 52, 56, 57A,
114, 117, 131, 135.

IBM Corp., Programming Publications Dept., Endicott, N. Y. 13760.

PRINTED IN U.S.A.

IBM **Technical Newsletter**

File No. S360-36 (DOS)

Re: Form No. C24-5036-1

This Newsletter No. N24-5302

Date: 6/16/67

Previous Newsletter Nos. N24-5218
N24-5248
N24-5289

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

This Technical Newsletter reflects the availability of system support for System/360 to System/360 binary synchronous communication.

The pages attached to this Newsletter replace the following pages of your publication:

Replace Pages

7A and 8

Changes on modified pages are indicated by a vertical line to the left of the affected text and to the left of affected parts of figures. A dot (●) next to a figure title or page number indicates that the entire figure or page should be reviewed. Please insert this page to indicate that your publication now includes the following modified pages:

Modified Pages

8

IBM Corp., Programming Publications Dept., Endicott, N. Y. 13760.

PRINTED IN U.S.A.

READER'S COMMENT FORM

IBM System/360 C24-5036-1
Disk Operating System
System Control and System Service Programs

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate.

COMMENTS:

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

C24-5036-1

Staple

Fold

Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold

Fold

Cut Along Line

IBM

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

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

Additional Comments:

IBM **Technical Newsletter**

File No. S360-36 (DOS)

Re: Form No. C24-5036-1

This Newsletter No. N24-5289

Date: April 28, 1967

Previous Newsletter Nos. N24-5218
N24-5208 is obsoleted by
N24-5289

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

This Technical Newsletter contains information concerning four foreground initiation commands that were not previously included in the publication and additional information concerning the PL/I Compiler. Planning information in the supplemental TNL N24-5208 is also included in this TNL, in the form of page replacement. Programming support for the IBM 1412 and the IBM 1419 Magnetic Character Readers will be included in a future release of the Disk Operating System.

The pages attached to this Newsletter replace the following pages of your publication:

Replace Pages

7 and blank
7A and 8
19 and 20
29 and 30
33 and 34
43 and 44
49 and 50
55 and 56
57 and blank
57A and 58
113 - 118
139 - 142

Changes on modified pages are indicated by a vertical line to the left of the affected text and to the left of the affected parts of figures. A dot (●) next to a page number indicates that the entire page should be reviewed. Please insert this page to indicate that your publication now includes the following modified pages:

Modified Pages

7, 8, 20, 20A, 29, 34, 34A, 43, 43A, 49, 50, 56, 57, 57A, 114, 115, 117, 140, and 141

IBM Corp., Programming Publications Dept., Endicott, N.Y. 13760.

PRINTED IN U.S.A.