File Number S360-20
Order No. GC24-5030-8

DOS
TOS

IBM Systems Reference Library

IBM System/360
Disk and Tape Operating Systems
Concepts and Facilities

This reference publication describes the concepts of Disk and Tape Operating Systems and guides the planner in the use of their various facilities. It describes the components in the Disk and Tape Operating Systems and explains the function of each. The last section of the publication discusses the design, preparation and execution of programs.

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

Preface

A related publication, IBM System/360 Basic Programming Support, Basic Operating System, Tape Operating System, and Disk Operating System, Programming Systems Summary, GC24-3420, provides an introduction to the IBM System/360.

Ninth Edition (October 1970)

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

This edition, GC24-5030-8, is a major revision of, and obsoletes, GC24-5030-7.

Summary of Amendments

This revision reflects the availability of ASCII (The American National Standard Code for Information Interchange) and Problem Determination Serviceability Aids (PDAID). All references to PL/I have been changed to PL/I (D). Miscellaneous changes are also included.

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

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

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

Contents

Disk and Tape Operating Systems are comprehensive sets of language translators and service programs operating under the supervisory coordination of an integrated control program. They require an IBM System/360 with at least 16K bytes of main storage.

The disk resident system requires at least one IBM 2311 Disk Storage Drive or one IBM 2314 Direct Access Storage Facility.

The tape resident system requires at least four magnetic tape units. One of the four units, the system residence unit, should be 9-track because of the significant system performance advantage it provides. This advantage is most significant during execution of problem-program and language translator functions that require repeated fetching of various program phases. The residence device can be a 7-track unit with the data convert feature. The IPL routine can determine the physical characteristics of the residence device and perform read backward operations (9-track) or rewind and read forward operations (7-track). The remaining tape units can be either 7- or 9-track.

The Disk and Tape Operating Systems have been designed to:

1. Shorten the period between the time a problem is submitted for solution and the time results are received.
2. Increase the volume of work that can be handled over a given period of time.
3. Assist those concerned with the system (installation managers, operators, and especially programmers).

The programmer can take advantage of a unified system that allows him to:

1. Place information, such as executable programs, in a system library and obtain access to the information by symbolic requests.
2. Divide a problem into a set of subprograms, code each in the language best suited to it, and combine them into an executable program.
3. Divide a large program into smaller sections that can be overlaid during execution to conserve main storage space.
4. Test and modify programs and data.
5. Choose between executing programs (or parts of programs) after compilation or assembly, or storing them on auxiliary devices for later use with output from other compilations or assemblies.

The Disk and Tape Operating Systems consist of a control program and a number of processing programs (Figure 1).

The control program supervises the execution of the processing programs, comprising language translators, service programs, and user-written problem programs. Most processing programs use the input/output control system (IOCS), whereas all processing programs use the facilities provided by supervisor and job control.

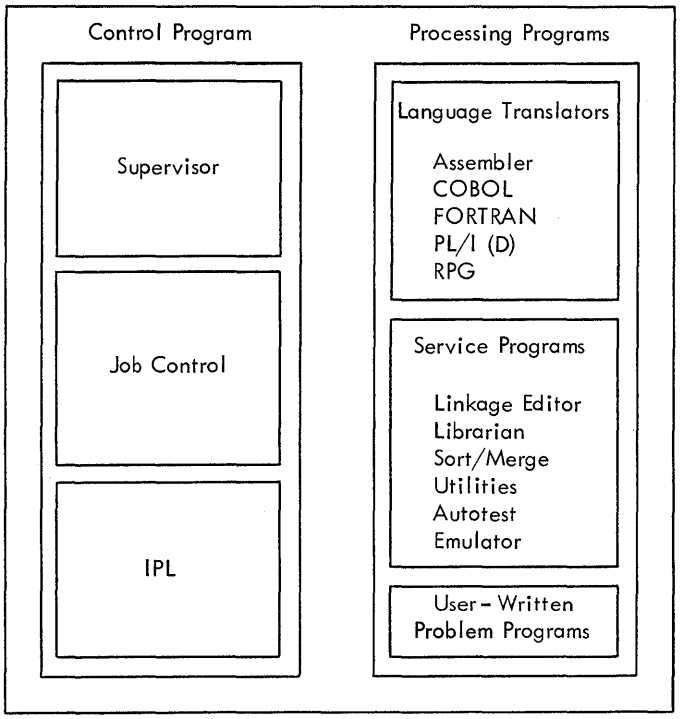

Figure 1. 16K Operating Systems

Control Program

Optimum efficiency of the system requires some automatic control. Without such control, the system is frequently idle and the operator must locate and load successive programs in addition to performing other required setup functions such as changing tape reels. An orderly and efficient flow of jobs through the system is facilitated by:

1. The permanent main storage resident supervisor.
2. The on-line residence of frequently used programs.
3. The job control program, which regains control whenever a job step terminates (normally or abnormally).

The Disk and Tape Operating Systems control program provides transition from phase to phase within a processing program (i.e., a job step), and from program to program within a job. Once the system has been initialized by the IPL routine, job after job can be included in the input job stream. The components of the control program are shown in the main storage map (Figure 2) and are discussed in the following sections.

MULTIPROGRAMMING

Multiprogramming is a technique whereby two or more programs may concurrently share the resources of a computer system in such a way that the operation of any one program is independent of the operation of any other program.

The effectiveness of any application of the technique must be measured, not in terms of the performance of any one program, but in terms of the extent to which available resources are utilized. In a multiprogramming environment no single program can necessarily run at full efficiency; nevertheless, the total efficiency and throughput of the system can be significantly improved.

Multiprogramming allows the input and output functions of one program to be overlapped with the processing functions of other programs. For instance, when one program yields control of the CPU (for example, during a wait state), control passes downwards to a partition of lower priority. When control is withdrawn from a program (for example, an I/O interrupt), control passes upwards to the partition of highest priority that is ready to execute.

DOS multiprogramming requires that programs be co-resident in main storage. To achieve this, storage is divided into partitions, with each partition capable of holding a distinct problem program. An active partition contains a program that is in course of execution. An inactive partition does not contain such a program, or is not physically present in the system.

Because each partition is of fixed physical size and is defined by fixed boundaries, DOS multiprogramming is termed fixed partition multiprogramming. Partition boundaries may not be altered while a partition is active; they may, however, be altered when the partition is inactive, thus allowing an increase or decrease in the size of a partition, provided that such an alteration does not result in the reduction in size of any other active partition. The three partitions are termed, in order of ascending priority, background, foreground-two, and foreground-one.

The amount of main storage available to the programs may be determined when the system is generated, or the operator can allot the amount of storage when the program is loaded into main storage for execution. Each program occupies contiguous areas of main storage. Multiprogramming requires 24K bytes of main storage.

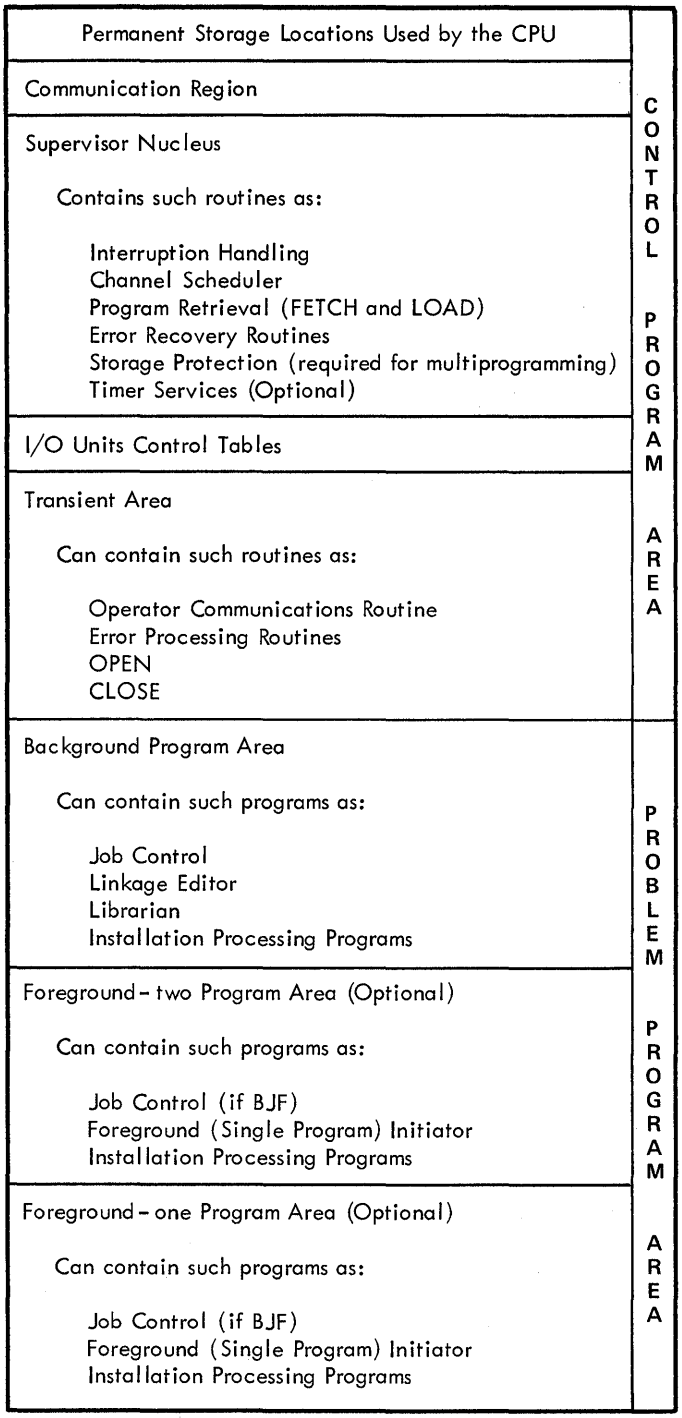

Figure 2. Main Storage Organization

Background vs Foreground Programs

There are two types of problem programs in multiprogramming: background and foreground.

Background programs are initiated by job control from a batched-job input stream.

Foreground programs executing as single programs are initiated by the operator from the printer-keyboard. Through the operator attention routine, he requests the foreground initiator program, also called the single program initiator (SPI). This program requires 2K bytes of main storage and accepts foreground initiation program commands from either the printer-keyboard or an operator-assigned card reader.

For the disk system, foreground programs may execute in batched-job foreground (BJF) mode if sufficient I/O and main storage facilities are available, i.e., 10K bytes of main storage and separate system input/output files for the partition. Batched job initiation is provided by job control if this option is specified at system generation time. A communication region is provided for each partition when batch processing in the foreground area has been specified. System logical units (except SYSLNK) may be used by either foreground or background programs. For foreground programs executing in single program initiator mode, system logical unit assignments are limited to punched card devices (card readers, card punches, and printers). Programmer logical units such as SYS000 and SYS001 may be used by either foreground or background programs.

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

The system can concurrently operate one background program and one or two foreground programs. Priority for CPU processing is controlled by the supervisor. The first foreground program has priority over the second foreground program, which in turn, has priority over the background program. All programs operate with interruptions permitted. 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 for 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.

For the disk system, object programs produced by FORTRAN, Basic FORTRAN, PL/I (D), Assembler, COBOL, and RPG may be run as batched-job foreground or background programs. Programs produced by the FORTRAN, Basic FORTRAN, and PL/I (D) compilers may not be run as single program initiator foreground programs because the object programs produced by these compilers require access to a communication region. Object programs produced by the Assembler and RPG compilers may be run as single program initiator foreground programs provided they do not require access to a communication region (See Figure 2). Under DOS, COBOL programs may run without restrictions in an SPI foreground environment. IBM-supplied programs are distributed to run in the background area. In a multiprogramming environment, telecommunication programs are normally run in the foreground-one area because this area is assigned the highest priority of the three program areas.

For the tape system, programs produced by the Basic FORTRAN and PL/I (D) compilers may not be run as foreground programs, because the object programs produced by these compilers use facilities available only to background programs. Object programs produced by the Assembler, COBOL, and RPG compilers may be run as foreground programs provided the following points are taken into consideration:

1. Object programs produced by the Assembler and RPG may not reference any system logical unit except SYSLOG.

2. Object programs produced by the COBOL compiler may not reference any system logical unit except SYSLOG, and may not contain EXHIBIT and TRACE (the output of the EXHIBIT and TRACE statements is on the system logical unit SYSLST). The DISPLAY and ACCEPT statements may be used only for the system logical unit SYSLOG, normally assigned to the printer-keyboard.

Note that IBM-supplied programs, except the disk system Tape and Disk Sort/Merge Program, are run only as background programs. The sort/merge program can be executed in the foreground when the Batch Job Foreground option is in the system.

Multiprogramming within Partitions

Using a set of macro instructions, multiprogramming may be performed in the disk system within any one or all of the partitions: background, foreground-one, and foreground-two. For multiprogramming users this, in effect, extends the capabilities of the Disk Operating System to execute twelve programs rather than three.

To perform multiprogramming within a partition, the program must consist of two parts: a main program, called the main task, and one or more subprograms, called subtasks. Subtasks share the same partition in main storage with the main task. Main storage may be divided among the subtasks and the main task within the partition in any way desired by the problem program. The main task initiates the execution of the subtasks. Subtasks have higher priority than the main task for CPU time within the partition. Up to nine subtasks can be attached to the main task within a partition. Alternately, subtasks may be attached to main task(s) in more than one partition, providing the total attached subtasks do not exceed nine.

The priority of the partition remains the same (foreground-one, foreground-two, background), but priority of a subtask within a partition is determined by when it is initiated (attached). Within a partition, the first attached subtask has the highest priority. As each subsequent subtask is attached (nine maximum), it has the next lower priority, followed by the main task that has the lowest priority. For example, if foreground-one partition contains no attached subtasks, foreground-two partition contains four attached subtasks, and the background partition contains three attached subtasks (the maximum of nine not being used), the priority order would be:

F1	Foreground-one area program
F2	Subtask 1
	Subtask 2
	Subtask 3
	Subtask 4
	Main Task
BG	Subtask 1
	Subtask 2
	Subtask 3
	Main Task

Multiprogramming within partitions is supported only by the Assembler language.

A track hold function provides protection of a DASD file being shared by more than one task or partition at the same time. The user can read a record from a disk and place a hold on that track to prevent any other task using the track hold function from accessing it. The protection remains in effect until the track is freed. Protection is also provided when more than one task will be manipulating data in the same area of main storage by preventing the data from being modified simultaneously by these tasks. The task manipulating data in the shared area first prevents other tasks from accessing it with a macro instruction. Subsequent requesting tasks using the same macro instruction are then queued and enter the wait state until the shared data is made available by the manipulating task. The requesting task with the highest priority then gains access to the data.

Supervisor

The supervisor has these functions:

- Accepts interruptions,
- Schedules the operator attention routine and all problem programs active in the system,
- Schedules input/output operations,
- Performs error recovery (and services other exceptional conditions for input/output devices),
- Notifies the operator when a specified number of tape errors has occurred, and it records the error statistics by tape volume (DOS),
- Loads programs from the system residence device (or, in the tape system, from a link-and-execute tape),
- Provides for program termination including a main storage dump facility,
- Contains translate tables for ASCII tape file processing, and
- Contains tables describing the configuration and operational status.

To minimize the main storage requirements of the supervisor, some supervisor routines are read into main storage only when needed. The area used for that purpose is called the transient area; routines executed in the transient area are referred to as transient routines. Supervisor routines needed only in exceptional cases (such as the dump routine, the recording of error statistics by tape volume routine, and routines to handle exceptional input/output conditions), are transient (Figure 2). Other transient routines are the operator communications routines and the file initiation (OPEN) and termination (CLOSE) routines of the input/output control system (IOCS). Some error recovery routines reside in main storage permanently, in case of an error in reading the resident system itself.

Executable programs are retained on the system residence device as images of main storage and are of two types: absolute or self-relocating. Absolute programs are those programs that are unable to be relocated after linkage editing. Self-relocating programs are those programs that retain the ability to be loaded and used at any main storage address, even after linkage editing. A self-relocating program in the core image library can be executed in either the background or foreground areas. An absolute program can only be executed in one of these areas without linkage editing again. The supervisor nucleus and most problem programs are absolute. The routines loaded into the transient area are self-relocating.

The Channel Scheduler initiates I/O operations and accepts I/O interruptions. The byte-multiplexing capabilities of the multiplexor channel are fully utilized by executing I/O requests for this channel as soon as each required device is free. If burst-mode devices are attached to the multiplexor channel, only non-overrunable devices are multiplexed. For example, the 2540 Read Punch is multiplexed but the 1442 Serial Read Punch is not, if tapes or direct access storage devices (DASD) are attached to the multiplexor channel. When selecting I/O requests from the queue on a selector channel, the Channel Scheduler bypasses requests for busy devices. Thus, a read request following a rewind request for the same magnetic tape unit does not tie up the channel.

A switchable device, such as a magnetic tape unit, may be attached to two channels. If a switchable tape unit is available for an I/O operation and channel one is unavailable, Channel Scheduler uses the second channel if it is available. In the disk system, the Channel Scheduler can optionally separate Seek commands from the remainder of the channel program to allow other input/output on a channel during mechanical seek arm movement.

When a record is read from SYSRDR/SYSIPT (the job input stream), the Channel Scheduler tests for either /* (end of data or end-of-job step) or /& (end of job) as the first two bytes of the record. If a record begins with either of these two character pairs, the Channel Scheduler assumes it is a delimiter and not a data record. Channel Scheduler posts an end-of-file condition to the program originating the input request. In this way, erroneous overreading of a job or a job step by a previous job or job step can be prevented.

The timer feature optionally maintains time of day and allows problem programs to set time intervals after which control is directed automatically to special routines (system or user supplied). Duration and clock time are printed with the JOB and /& control statements (the first and last control statements of a job) for accounting purposes.

The supervisor assembled during generation of the system is adapted to the configuration of the individual installation. All other components of the system adapt to the various machine and operational options by interrogating information in a communications area of the supervisor nucleus. Table areas, buffers, etc can thus be expanded to use additional main storage at program execution time.

OPERATOR COMMUNICATION

The Disk and Tape Operating Systems allow the machine operator to:

1. Control the system.
2. Query its status.
3. Respond to system-initiated requests for operator intervention.

Messages to the operator are written on the operator communication device, the printer-keyboard. Operator responses to the system are generally short, one-word answers, such as RETRY, IGNORE, or CANCEL. In addition, the operator can elect the system default option in most intervention required situations.

When an I/O error condition occurs (such as intervention required, or card read error that requires operator intervention, including readying the device), the system continues operation. Programs not dependent upon the device can continue processing. When the I/O device is readied, the pending I/O operation is automatically restarted. No operator response from the printer-keyboard is required. The operator communication routines allow the operator to issue commands to the system for:

1. Canceling execution of the current job (in case of a suspected program loop).
2. Requesting a pause to permit forms change, stacker emptying, device assignment changes, etc.
3. Requesting the suspension of batch processing.
4. Performing magnetic tape operations (such as write tape mark or rewind).
5. Assigning symbolic device names to I/O devices.
6. Requesting a list of current system I/O assignments.
7. Closing output files.
8. Obtaining a map of main storage allocation and occupancy.
9. Reallocating the interval timer or main storage among background and foreground areas.
10. Initiating the execution of a foreground program.
11. Dumping error statistics from disk to a tape or a printer (DOS).

Job Control

Job control performs various functions for batched-job foreground and background programs on the basis of information provided in job control statements. Among these functions are:

1. Assigning symbolic names to I/O addresses.
2. Establishing system options.
3. Storing volume and file label information.

Job control type functions are performed by the Foreground (Single Program) Initiator for foreground programs executing as single programs.

Because I/O device addresses vary between and within installations, usually it is impractical to use these addresses in source programs. Therefore, a standard set

of symbolic names has been established. Each relevant name is identified with an I/O device before executing a given program. This function is performed by the program initiator, based on operator commands or programmer-furnished control statements. For background and batched-job foreground programs, the program initiator is job control. For single foreground programs, the program initiator is the Foreground (Single Program) Initiator.

Some symbolic names are required by the system and must be assigned to specific types of devices. These symbolic names are in the form SYSxxx, where xxx represents specific alphabetic characters. For example, the symbolic name SYSRDR, the name of the job control input device, must always be assigned to a card reader, a magnetic tape unit, or a 2311 or 2314 disk extent.

Other symbolic names, in the form SYSnnn where nnn represents numbers ranging from 000 through the maximum number of units defined for the system, may be assigned at the discretion of the user. Some system components require assignment of one or more of these symbolic units. For example, any device supported by the Disk and Tape Operating Systems may be assigned the name SYS004.

The Tape Operating System supports only programmer logical units in the foreground areas, whereas the Disk Operating System provides both system and programmer logical units in the foreground areas. A separate set of units is defined for each of the classifications of programs: background, foreground-two, and foreground-one.

Standard assignments of symbolic device names to I/O devices are normally established when a system is generated. The operator can permanently or temporarily modify these standard assignments when the system is initialized or between job steps, permitting adjustment for configuration changes. Job control restores all temporary assignments to the standard assignments at the completion of each job for which they were modified.

For example, a specific card reader might be assigned at system generation time as SYSIPT. The programmer could change the assignment from the specified card reader to a specific magnetic tape unit for his individual job. At the completion of the job, the assignment for SYSIPT reverts to the system standard, the card reader.

For batched job operations, system input (SYSRDR and SYSIPT) and system output (SYSLST and SYSPCH) may be either tape, or punched card devices (card readers, card punches, and printers). For the Disk Operating System, these files may be assigned to an IBM 2311 Disk Storage Drive or an IBM 2314 Direct Access Storage Facility. Support of system files on disk requires 24K bytes of main storage.

System input and output files on tape may be combined, i.e., one tape unit can replace SYSRDR and SYSIPT, and another can replace SYSPCH and SYSLST.

Other system options can also be standardized for background programs when a system is generated. These options determine whether:

1. Control statements are to be listed.
2. Dumps of main storage are to be printed for abnormal terminations of jobs.
3. Language translators are to punch object decks.
4. Language translators are to print source module listings.
5. The COBOL compiler is to print object module listings.
6. Assembler and PL/I (D) are to output symbol tables, and the COBOL compiler is to output a data division map.
7. The Assembler is to print symbolic cross-reference listings.
8. The Basic FORTRAN, COBOL, and PL/I (D) compilers are to summarize all errors in source programs. (The Assembler and FORTRAN always provide this facility.)

The programmer can modify these options by using one or more of the appropriate job control statements. Such modifications are restored by job control to the system standard at the end of each job. For example, a system standard can be established that language translators are to punch object decks. The programmer can change the standard for his job by specifying // OPTION NODECK. The system standard of punching object decks is restored at the end of the job.

Additional options allow the programmer to indicate that:

1. A program is to be linkage edited.
2. A program is to be added to the core image library on the system residence device.
3. A file label is to be stored either permanently or temporarily, and the

label may be used by all partitions or by only one partition.

On a tape device, the multivolume output for SYSLST and SYSPCH facility provides automatic volume switching whenever an end of volume condition is sensed on a file.

For the tape system, checkpoint records are written on a 2400 tape unit. The tape may be a 9-track or a 7-track tape that has the data conversion feature. User data on the tape can be either in translate or data conversion mode. For the disk system, checkpoint records may be written on a 2311 or 2314 disk, or a 2400 tape unit. Checkpoint/restart may be used by any program during batch processing in a foreground or background area.

All volume-label and file-label processing is done during problem program execution. Label information is read from label statements by the program initiator (job control for batched-job foreground or background programs, Foreground (Single Program) Initiator for single foreground programs) before the execution of the problem program. The label information is subsequently stored for use by the problem program. As each file is opened, the corresponding labels are checked against data retained from the volume and file-label statements. If discrepancies are detected, a message is printed on the printer-keyboard, informing the operator of such discrepancies. The operator can terminate the job, ignore the condition, or correct the condition and continue processing.

Volume and file information can be contained in one control statement for tape. In the disk system, volume and file information can be contained in two control statements for disk. Most of the fields in the label control statements are optional. In the tape system, volume and file label information is stored in the low part of the problem program area. In the disk system, label and extent information is stored on the first cylinder after the last library of the system. The first six tracks of the cylinder can contain temporary or standard label information defined for a partition. The remaining tracks of the cylinder can contain standard label information for any or all partitions.

The standard label tracks are not overwritten until specified by the user. Thus, disk work areas may be defined, permitting compilation or assembly jobs to be submitted without label and extent information for work files. The tracks used for temporary label information can be overwritten between jobs and job steps.

Data Management (IOCS)

The data management facilities of Disk and Tape Operating Systems are provided by a group of routines collectively referred to as the input/output control system (IOCS). A distinction is made between two types of routines:

1. Physical IOCS (PIOCS). The physical unit I/O routines included in the Supervisor.

2. Logical IOCS (LIOCS). The logical unit I/O routines linked with the user's problem program.

PIOCS is used by all Disk and Tape Operating Systems programs to initiate or test I/O operations, whether or not they use LIOCS. PIOCS supervises the execution of channel programs without regard to the logical content, format, or organization of the data being read or written. (It does, however, set the mode requested by the user for tape operations: density, parity, translation, and byte-conversion.)

PIOCS includes facilities for:

1. Scheduling and queuing I/O operations.

2. Checking for and handling error conditions and other exceptional conditions related to input/output devices.

3. Handling I/O activity to maintain maximum I/O speeds without burdening the problem program.

As part of most user programs, LIOCS provides an interface between the user's file processing routines and PIOCS. (All COBOL, FORTRAN, Basic FORTRAN, RPG, and PL/I (D) programs use LIOCS; most Assembler programs use LIOCS.) Unlike PIOCS, LIOCS handles logical data files. Generalized LIOCS routines can be assembled along with problem programs, or they can be separately assembled, assigned a unique name, and cataloged into the relocatable library by the user for inclusion in problem programs.

Depending on information assembled with the user's program to define various attributes of each file (name, location, device type, organization, format, etc), the appropriate LIOCS routines then can be retrieved from the relocatable library and linked to the user's program when it is linkage edited.

Depending on file attributes, LIOCS performs the following functions where applicable:

1. Requests (of PIOCS) execution of appropriate channel programs.
2. Opens and closes files.
3. Services end-of-file and end-of-volume conditions.
4. Services user-requested label processing exits.
5. Blocks and deblocks records.
6. Manages user-defined I/O buffers.
7. Services problem program requests for control operations (that is, backspace tape, stacker selection, printer forms control, etc) on I/O devices.
8. For ASCII file processing under DOS, translates from ASCII to EBCDIC on input and from EBCDIC to ASCII on output.

For the tape system, LIOCS supports only sequential file organizations. Such data organizations are compatible with BSAM (Basic Sequential Access Method) and QSAM (Queued Sequential Access Method) data sets of Operating System, although the associated program logic is different. Instead of distinguishing basic and queued access methods, Disk and Tape Operating System IOCS permits up to two I/O buffers.

To avoid time consuming OPEN and CLOSE functions for intermediate tape work files (which are often written, reread, and rewritten several times in a single job step), NOTE/POINT macros provide limited direct access to sequential files.

For the disk system, LIOCS supports sequential, indexed sequential, and direct access file organizations. The sequential data organizations are compatible with BSAM and QSAM data sets of Operating System. The indexed sequential data organizations are compatible with BISAM (Basic Indexed Sequential Access Method) and QISAM (Queued Indexed Sequential Access Method) data sets of Operating System. The direct access data organizations are compatible with BDAM (Basic Direct Access Method) data sets of Operating System. As with the tape system, the disk system provides NOTE/POINT macros for limited direct access to sequential files.

For the disk system, IOCS file definition macro instructions are available that provide device independent sequential file processing for the system units SYSRDR, SYSIPT, SYSPCH, and SYSLST. The physical device can be assigned at execution time, allowing data files to be processed by the physical device that the user prefers at the time. For example, when the device independent macro instruction is used in the program at execution time, the system logical device SYSRDR could be assigned to a card reader, a magnetic tape, or a disk extent. The user's program need not be modified. This can be advantageous to the user when one physical device is temporarily inoperative, enabling him to process on another device. The function may be used in the user's background or foreground programs and is available to Assembler language users only.

Processing Programs

Two types of processing programs, language translators and service programs, are contained in the system supplied by IBM. The third type of processing program (user-written problem programs) can be added by the user to the system.

Language Translators

Programs in five different languages can be translated into relocatable object modules. The five translators are Assembler, COBOL, FORTRAN, PL/I (D), and RPG.

ASSEMBLER

The Assembler language provides a convenient means of solving problems by directly utilizing the IBM System/360 instruction set. It is a symbolic, machine oriented language that is applicable to any problem. The problem program coding is done with symbolic instructions that are translated into machine instructions. Program locations can be addressed through symbolic names. Data constants can be defined in several different ways, either as explicit constants or as literals coded directly into the operands of symbolic instructions.

The Assembler language enables the programmer to define and use macro instructions. Macro instructions are represented by an operation code which, in turn, actually stands for a sequence of machine and/or assembler instructions (called a macro definition) that accomplishes the desired function.

Macro definitions used with an Assembler language source program fall into two categories. System macro definitions, provided by IBM, relate the object program to facilities of the Disk and Tape Operating Systems. Other macro definitions may be created by the programmer specifically for use in the program at hand, or for incorporation into the source statement library, available for future use.

Programmer created macro definitions simplify the writing of a program and/or ensure that a standard sequence of instruction accomplishes a desired function. For example, the logic of a program may require the same instruction sequence to be executed again and again. Rather than code this entire sequence each time it is needed, the programmer creates a macro definition for the sequence. Each time the sequence is needed, the programmer simply codes the macro instruction corresponding to the macro definition.

In the Disk Operating System, a user with a 16K system has the option of using the Assembler with a disk work file variant, or an Assembler with a tape work file variant. Either variant requires 10K bytes of problem program storage. A user with at least a 24K system has an additional option of using the Assembler with a disk or tape work file variant. This variant requires 14K bytes of problem program storage. For users with at least 64K bytes of main storage, Assembler F is available. This Assembler requires 44K bytes of problem program storage. In the Tape Operating System, a 10K and a 14K variant are available, both requiring tape work files.

COBOL

COBOL is based on a well defined restricted form of English especially suited for commercial data processing problems. COBOL programs are translated by the compiler directly into object code, including most I/O functions necessary for processing the user's data files. The remaining I/O functions are included from the relocatable library by the Linkage Editor.

The source statement library may be used to store additional source language subroutines for inclusion in source programs at compile time. For example, a standard system configuration can be included in the Environment Division of the COBOL source program by using a COPY statement.

The relocatable library furnishes object language subroutines supplied by IBM, plus those created by the user. These subroutines are incorporated into various programs by the Linkage Editor. In the disk system, COBOL supports direct access files.

A 16K system is required for the Tape Operating System; at least a 24K system is

required for the Disk Operating System. In the disk system, COBOL requires 14K of problem program area and is capable of using either disk or tape work files.

A Language Conversion Program (LCP) is available under the Disk Operating System to help convert existing COBOL programs into a form acceptable to the DOS American National Standard COBOL compiler. The LCP requires a problem program area of at least 18K. The LCP is not supported under the Tape Operating System.

AMERICAN NATIONAL STANDARD COBOL. For American National Standard COBOL, 54K of problem program area is required.

BASIC FORTRAN

The Basic FORTRAN language helps solve problems that are primarily mathematical in nature. Problems containing formulas and variables can be easily described in the Basic FORTRAN language. Source statements are translated directly into object code. The user can divide his Basic FORTRAN deck into one main program and several subprograms for speed of development and ease of maintenance. Subprograms can be called by the main program and/or other subprograms. The relocatable library stores object modules for inclusion into various programs by the Linkage Editor.

The IBM Basic FORTRAN language is compatible with, and encompasses the American National Standard Basic FORTRAN. The American National Standard Basic FORTRAN language has been extended by the following features to facilitate the writing of source programs and reduce the possibility of coding errors:

1. Expressions may consist of constants and variables of the same and/or different types.
2. The T format spacing code allows input/output data to be transferred beginning at any specified position.
3. Apostrophes may be used to enclose literal data in a FORMAT statement.
4. The A format code allows reading and writing of character data.
5. The scale factor allows modification of the internal or external representation of data.
6. The first character of a record to be printed is used for carriage control.
7. Three types of data are supported: INTEGER (binary), REAL (short precision floating point), and DOUBLE PRECISION (long precision floating point).
8. An array may have one, two, or three dimensions.
9. The name of a variable may contain as many as six characters.
10. Data records may be read or written in any order on some types of input/output devices.
11. Function subprograms may return results via the argument list.

The Basic FORTRAN compiler requires a 16K system (10K of problem program area). In the disk system, its work file may be either a tape or a disk.

FORTRAN

The Disk Operating System provides an IBM FORTRAN compiler in addition to its Basic FORTRAN facility. The FORTRAN compiler requires a minimum background partition of 40K bytes for compilation. The FORTRAN language is compatible with, and encompasses the American National Standard FORTRAN. All Basic FORTRAN facilities apply to FORTRAN. In addition, the following extensions to American National Standard FORTRAN are available:

1. IMPLICIT statement allowing extended implicit classification by first character of a name.
2. Length specifications and initialization values in type statements.
3. G-conversion extended to cover all numeric and logical data types.
4. Multiple entry points in subprograms.
5. Selective returns from SUBROUTINE subprograms.
6. Arrays of up to seven dimensions.
7. Data-directed input and output via the NAMELIST statement.
8. Generalized subscripts.
9. Hexadecimal constants and FORMAT code.
10. Debugging language statements.

11. Phase load overlay for object programs.

Optionally, at system generation time, a user may specify that the FORTRAN compiler is to print object module listings.

Subprograms compiled by Basic FORTRAN may be used with FORTRAN programs. This compatibility may be provided for at system generation time or, on a job-by-job basis, during linkage editing.

The FORTRAN compiler requires two work files, each of which may be on disk or tape. Object programs may be run in a background partition or in either foreground in the batched job foreground (BJF) mode.

PL/I (D)

PL/I (D) provides the programmer with a unified problem-oriented language for efficiently programming either scientific or commercial problems, as well as problems that can best be solved with a combination of scientific and commercial computing techniques. It is particularly useful for the increasing number of semicommercial, semiscientific applications, such as information retrieval or command and control applications. The modern features of PL/I (D) make it useful for many programming applications for which other compiler languages are not suited.

The PL/I (D) compiler requires at least a 16K system. In the disk system, two variants of the PL/I (D) compiler can be built. One of the variants requires 10K bytes of problem program storage, while the other requires 12K. The 10K variant allows the system input and output files to be assigned to magnetic tape or punched card devices. The 12K variant allows the system input and output files to be assigned to a disk drive. In the disk system, PL/I (D) is capable of using either disk or tape work files.

RPG

RPG is designed specifically for report writing and file maintenance applications. The RPG language facilitates producing programs for a wide variety of reports ranging from a simple listing to a complete report that incorporates calculation and editing.

Additional facilities provided are:

1. Table lookup.
2. Branching capabilities.
3. Indicator control.
4. Split control fields.
5. Sterling conversion.
6. Designation of multiple input and output files.

IBM System/360 RPG has many more facilities than RPGs of prior systems. Special coding sheets are provided to describe the job to be performed. The RPG language is problem oriented and does not require detailed knowledge of machine functions. The main body of the program can be written in RPG, and separately assembled routines written in the Assembler can be combined with it by the Linkage Editor.

The RPG compiler requires at least a 16K system (10K of problem program area), and in the disk system, is capable of using either disk or tape work files.

Service Programs

The IBM supplied service programs perform those functions required by the majority of users. A description of the service programs follows.

LINKAGE EDITOR

The Linkage Editor combines object modules supplied in the job input stream, and/or newly compiled object modules, and/or object modules from a relocatable library. It edits these modules into executable programs. These programs then can be fetched directly from the Link-and-Execute file, or they can be cataloged into the core image library. In the tape system, the Link-and-Execute file is a separate tape. In the disk system, the Link-and-Execute file is the temporary part of the core image library.

The Linkage Editor is capable of generating simple or complex overlay structures. A phase, one element of an overlay structure, is the object of one FETCH or LOAD operation and is composed of one or more complete control sections.

One phase can contain control sections from several object modules. One object

module can supply control sections to several phases. To illustrate this, consider the following examples. One object module (MOD1) contains three control sections (CSECT1, CSECT2, and CSECT3). A second object module (MOD2) contains two control sections (CSECT4 and CSECT5).

To construct a phase (PHNAME1) containing selected control sections from each of these object modules, the following sequence of Linkage Editor control statements can be used.

```
PHASE   PHNAME1,*

INCLUDE   MOD1,(CSECT1,CSECT3)

INCLUDE   MOD2,(CSECT5)
```

To construct two phases (PHNAME2 and PHNAME3) containing control sections from the same object module, the following sequence of Linkage Editor control statements can be used.

```
PHASE   PHNAME2,*

INCLUDE   MOD1,(CSECT1,CSECT2)

PHASE   PHNAME3,*

INCLUDE   MOD1,(CSECT3)
```

The Linkage Editor also allows the same control section to occur in different phases of the same overlay structure. For example, using the same object module (MOD1), the following two phases (PHNAME4 and PHNAME5) contain the same control section (CSECT2).

```
PHASE   PHNAME4,*

INCLUDE   MOD1,(CSECT1,CSECT2)

PHASE   PHNAME5,*

INCLUDE   MOD1,(CSECT2,CSECT3)
```

Thus, a program could use the same I/O control section in its first and last phases, omitting it during intermediate phases that have no I/O activity.

The Linkage Editor program may execute as a background program only.

LIBRARIAN

Each system residence device contains one to three types of libraries: core image (required), relocatable, and source statement. As their names imply, executable programs (core image format) are stored in the core image library; relocatable object decks are stored in a relocatable library; and, source language routines are stored in a source statement library.

Each library is preceded by a directory for the library. In the tape system, the directory takes the form of header records preceding each entry in the library.

The Librarian is a group of routines that maintain and service the libraries of the systems. The maintenance routine in both systems, disk and tape, provides for cataloging (adding) and deleting. In the disk system, the maintenance function also provides the ability to rename, condense, and reallocate. The service routines provide for displaying (printing) and punching elements of the relocatable and source statement libraries. In the disk system, a service program is available for displaying and punching specified phases or programs from the core image library. The punched output is of the form acceptable as input to the Linkage Editor. Both systems provide for copying the library components. Directories for all libraries can be displayed.

The Librarian program may execute as a background program only.

Core Image Library

The core image library contains any desired number of programs, both IBM supplied and user coded. Each program comprises one or more phases created by the Linkage Editor. Hence, each phase may be either a single program or, in the case of a multiphase program, an overlay. For more rapid retrieval of the multiphase foreground programs in the disk system, 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.

All programs (except transients) in the core image library must be linkage edited to run in the problem program area (above the resident Supervisor). Non-self-relocating foreground programs must be linkage edited to run at the address of the associated foreground partition. After linkage editing, each program phase is associated with fixed locations in main storage, although self-relocating phases (usually retrieved by the LOAD macro instruction) can be loaded anywhere in the problem program area of storage.

Relocatable Library

A relocatable library can contain any number of modules. Most modules are complete object decks in relocatable format, the output from assemblies and compilations. Other modules are IOCS modules required by the various compilers. The relocatable library permits the user to retain frequently used routines on-line, available for combination with other modules without requiring recompilation.

The 80-column standard IBM System/360 object card contains up to 56 bytes of program information. Packing allows a 162-byte block in the relocatable library to contain up to 132 bytes of program information on the tape system. In the disk system, up to four 80-column cards may be contained in each 322-byte record.

Source Statement Library

A source statement library can contain any number of sequences of source language statements. Each sequence is called a book.

The source statement library provides an extension of the functions of a macro library. If an assembly source program contains macro instructions (macros), the macro definitions in the source statement library corresponding to these macro instructions are generated into the source program. If an Assembler or COBOL source program contains a COPY statement, the appropriate language translator inserts a book from the library into the source program during compilation.

Each book in the source statement library belongs to a sublibrary. Sublibraries are currently defined for two programming languages, Assembler and COBOL. Sublibrary A is used by the Assembler, and sublibrary C is used by COBOL. Classifying books by sublibrary names permits duplicate names in different sublibraries.

Each book in the source statement library is compressed. Source statements in the library are blocked into longer records after each blank field has been replaced by one or more bytes giving the count of blanks in that field and the count of nonblank characters in the preceding field. In the tape and disk systems, the records are 160 bytes. When a book is retrieved, each of its statements is expanded to its original 80-character format.

Private Libraries -- Tape Operating System

The Tape Operating System permits each user to build relocatable and source statement libraries on separate tape reels. A job control statement notifies the system when either the relocatable or source statement library is assigned to a private tape.

Private libraries are advantageous to the user in several ways:

1. The time required to perform an assembly (containing macro instructions and/or COPY statements) or a COBOL compilation (containing COPY statements) can be reduced. There is then no need to bypass a portion of the core image library and the entire relocatable library, extract the appropriate book(s) from the source statement library, and subsequently reposition the tape to the appropriate place in the core image library.

2. The time required to perform a linkage editing function can be reduced, because there is no need to bypass a portion of the core image library, extract the indicated module(s) from the relocatable library, and subsequently reposition the tape to the appropriate place in the core image library.

3. Maintenance functions (cataloging, deleting, and/or copying) for private libraries can be performed in less time because only the individual library and not the entire system tape need be updated and copied on another magnetic tape.

Private Libraries -- Disk Operating System

The Disk Operating System permits the user to build private relocatable and source statement libraries. A private library in the disk system is a disk file having a unique file name. A job control statement notifies the system when a relocatable or source statement library is private.

Private libraries are advantageous to the user in several ways:

1. The user need not be concerned with limiting the size of the core image library on the system residence pack in order to have room for a relocatable and source statement library on the system pack. The relocatable and source statement libraries can be removed from the system residence file

and placed in private libraries, leaving more room for the core image library.

2. The user may have as many relocatable or source statement libraries as he desires, each serving a particular function. By notifying the system which private relocatable and/or which source statement library is to be used in a particular job step, it will be interrogated when needed.

3. A relocatable and source statement library on the system residence file may be used in combination with a private relocatable and source statement library respectively.

4. Private libraries are useful in a testing environment because the user can keep his working programs on a system library and his modified versions of these programs on private libraries. The modified versions can thereby be tested without destroying the current working programs that are on the systems residence file or another private library.

Maintenance functions such as catalog, delete, rename, condense, and condense limit may be performed on private libraries. All service functions available to the system are also available to private libraries.

If both a private and a system relocatable library are on-line, the private library is searched first in an attempt to find a module to be linked into the user's program.

SORT/MERGE PROGRAMS

Three sort/merge programs are available for users of the Disk and Tape Operating Systems:

- The Disk and Tape Operating Systems Tape Sort/Merge Program.
- The Disk Operating System Disk Sort/Merge Program.
- The Disk Operating System Tape and Disk Sort/Merge Program.

The sort/merge programs enable the user to sort multiple files of randomly ordered records or to merge multiple files of sequenced records into one sequential file. Sequencing is performed by comparing up to 12 specified control data fields within the records. The programs assume random sequences for input files to sort operations, but take advantage of any inherent ordering. Records may be sorted or merged in ascending or descending order.

The three programs differ in the number of input files permitted, and the I/O device types supported for input, work, and output files. The following lists give the general capabilities the programs have in common, and the individual characteristics of each program.

Each of the sort/merge programs:

1. Processes standard System/360 volume and file labels, provides linkage to user label processing routines for standard, nonstandard, or user labels, and permits the use of unlabeled tapes.

2. Supports 9-track or 7-track (with or without data conversion) magnetic tape units, and some combinations of the two types.

3. Allows multivolume input and output files.

4. Provides linkages to user-written routines at various points in a sort or merge operation.

5. Provides checkpoint, interruption, and restart procedures for the sort operation.

6. Allows input files and the output file to be spread over multiple I/O devices.

7. Provides for specification of alternate input and output tape drives for either a sort or a merge operation.

8. Provides the ability to bypass unreadable data blocks, or to indicate the need for operator intervention. Provides a message to the operator indicating that a block has been bypassed.

9. Sequence checks the records during the final pass of a sort or merge run.

The Disk and Tape Operating Systems Tape Sort/Merge Program:

1. Supports 9-track and/or 7-track magnetic tape units for input, output, and work files.

2. Sorts up to 9 input files, merges up to 7 input files.

The Disk Operating System Disk Sort/Merge Program:

1. Supports 9-track and/or 7-track magnetic tape units and IBM 2311 disk storage drives for input and output files. A disk area is used for work files.

2. Sorts up to 9 input files, merges up to 4 input files.

3. Provides the option of writing an output file consisting only of the disk addresses of the ordered records, or the disk addresses and control data fields.

4. Provides a facility for calculating the number of disk work area tracks required for sorting.

The Disk Operating System Tape and Disk Sort/Merge Program:

1. Supports 9-track and/or 7-track magnetic tape units, IBM 2311 Disk Storage Drives, and IBM 2314 Direct Access Storage Facility for input, output, and work files.

2. Sorts up to 9 input files, merges up to 8 input files.

3. Provides the option of writing an output file consisting only of the disk addresses of the ordered records, or the disk addresses and control data fields.

4. Provides a facility for calculating the number of disk work area tracks required for sorting.

5. Allows a wider latitude of action by user-written exit routines than the other sort/merge programs, including the ability to lengthen, shorten, and delete records, to read an input file, and to write an output file.

6. May be attached by an executing program as well as executed as an ordinary job.

UTILITIES

The utility programs copy data files from one storage medium to the same or another medium, together with reblocking and field select options. Each utility program is generalized. To handle a specific job, the generalized program is modified by control statements. Control statements are free-form, that is, optional parameters can be supplied in any order. The programs assume a normal use for most options when a choice is not indicated in a control statement.

Special purpose programs, such as Initialize Disk, Assign Alternate Tracks, Copy-Restore (Tape or Card), Copy Disk to Disk, and Initialize Tape are available for the disk system.

For the disk system, utility programs are distributed to run in the background area but may be linkage edited to run in a batched foreground area.

AUTOTEST

The Autotest Program provides debugging aids for Assembler language programs. Its testing services are available to other programs in the system. It is most useful for symbolic debugging of assembled decks.

Some of the features of the Autotest Program are:

1. Autopatch. Instructions can be exchanged, added or deleted without reassembling or computing linkage addresses. Constants can only be replaced.

2. Display. Data from selected areas of main storage can be printed at specified points during the execution of the object program.

3. Panel. General registers, main storage positions 24-127, and floating point registers in any combination can be printed at specified points during execution of the problem program.

4. Main storage printout. The content of main storage can be dumped, if requested, on a normal end of job. On an abnormal end of job, the dump always occurs. Character, mnemonic, and symbolic information optionally supplements a hexadecimal dump.

The Autotest facility is available to all 2311 Disk and Tape Operating Systems users. However, while operating in Autotest mode, only background programs can be processed. This facility is not available to 2314 disk users.

EMULATORS

The Emulator Programs under the Disk Operating System allow 1401, 1440, and 1460 user-written programs to be executed in the background partition or in batched foreground in a stacked job environment,

and may be intermixed with System/360 programs in the job stream. The Emulator Programs require little or no reprogramming of 1400 programs that are written consistent with 1400 System Reference Manuals published by IBM.

The Emulator Programs use the physical IOCS capabilities of the Disk Operating System to simulate the 1400 I/O instructions. Automatic job transition is provided by the system when a 1400 end-of-job halt is encountered. Depending on control card options, 1400 non-end-of-job halts and 1400 program error conditions are either routed to the operator for action, or result in an abnormal end of job followed by a release to the disk system.

Three main levels of support are provided by the Emulator Programs:

- 1400 Unit Record: Support is provided for 1400 card programs, and for reading and punching both BCD and binary data.
- 1400 Tape: Support is provided for 1400 tape operations.
- 1400 Disk: Support is provided for 1311 Disk Storage Drives and 1301 and 1405 Disk Storages.

BASIC TELECOMMUNICATIONS ACCESS METHOD (BTAM)

The Basic Telecommunications Access Method (BTAM) controls transmission and reception of messages over telecommunication lines in response to READ and WRITE macro instructions issued in the user's problem program. To accomplish this function, BTAM dynamically generates and executes channel programs and, at the user's option, provides buffer allocation.

BTAM requires 32K bytes of main storage and is available in both multiprogramming and batch-only systems. BTAM programs can operate as either foreground or background programs.

The facilities of BTAM are made available through the macro generation capabilities of the Disk Operating System assembler. From a macro instruction describing the types of terminals, lines, and other facilities to be used, the user generates his BTAM logic module. It may be assembled with the user program or separately assembled and combined with it at linkage edit time. During assembly of a problem program, macro instructions coded by the user are expanded into:

1. Tabular information defining the lines, terminals, and options to be used.
2. Linkage to the BTAM routines.

When an OPEN macro instruction is executed in the problem program, Teleprocessing lines are prepared for data transmission, and initialization for buffer management is performed. When a message is to be received or sent, a READ or WRITE macro instruction causes a branch to the BTAM READ/WRITE routine. This routine builds a channel program to perform the requested operation and passes the request to the I/O Supervisor, which starts the channel program. Control passes back to the user at this point. Execution of the channel program is performed asynchronously to the problem program.

An important distinction between BTAM and other access methods of the Disk Operating System is shown in the way in which the channel program is executed. For certain types of terminals and line configurations, the BTAM channel programs may be repeatedly restarted in response to conditions on the line. This allows a single READ to successively poll a number of terminals on a line. A single WRITE can signal a number of terminals to prepare to receive a message. The functions of restarting channel program and buffering are performed asynchronously to execution of the problem program.

QUEUED TELECOMMUNICATIONS ACCESS METHOD (QTAM)

The Queued Telecommunications Access Method (QTAM) is an input/output control system that extends the techniques of logical IOCS to the telecommunications environment. Files accessed by the problem programmer are queues of messages both incoming and outgoing via remote terminals connected to communication lines. Although the time and order of the arrival of messages to and from the central processing unit (CPU) are unpredictable, the programmer can handle the messages as if they were organized sequentially.

QTAM furnishes more than the mechanics for input/output operations. In addition to the standard GET/PUT macro instruction support for message processing programs, QTAM provides a high-level and versatile message control language. QTAM-supplied macro instructions can be used to construct a complete message control program that controls the flow of message traffic from one remote terminal to another (message switching application), and between remote

terminals and any message processing programs (message processing applications). An installation-oriented message control program can thus be written in a shorter time than was previously possible.

A QTAM message control program is generated from a number of assembler macro instructions coded by the programmer. Although the assembler macro generator is used, the process followed is similar to that used by a high-level compiler. A generated message control program is completely device dependent, with all communication lines and terminals identified to the system. The user specifies his equipment configuration and the main storage areas (buffers) required for his applications through file definition and control information macro instructions. These macros generate the tables and lists of control information that define the system environment for the QTAM logic.

QTAM logic modules are also provided for many procedural functions such as message code translating, message routing, and error checking. By selecting the appropriate macro instructions, the user can specify which QTAM logic modules are to be incorporated into the message control program. In this way, the system can be tailored to the exact requirements of the particular application.

The QTAM message processing program services enable a programmer to process messages from a telecommunications network with the same macro instructions that are used for local input/output devices. Because a QTAM message control program performs the input/output operations, it is possible to write a message processing program that is device independent. Thus, the programmer is protected from the time and device dependent aspects of the telecommunications environment.

QTAM requires 64K bytes of main storage and is available only in a multiprogramming system. A QTAM message control program must be executed as a foreground-one program. Up to two QTAM message processing programs can run as foreground-two and/or background programs.

UTILITY MACRO INSTRUCTIONS

These macro instruction facilities are designed to generate self-relocating file-to-file utility programs or user-designed file-processing programs for operation in a multiprogramming environment. Macro instructions are provided for each of the following:

Disk Input	Card Input
Disk Output	Card Output
Tape Input	Printer Output
Tape Output	Console 1052 Input
Tape Cartridge Input	Console 1052 Output

A typical generated program would require two of these macros. For example, a tape-to-printer utility would use the tape-input macro and printer-output macro.

Optional exits are available to user written routines, which may provide control information, perform specialized label processing, perform record processing, handle unreadable records, add and delete records, etc.

System Generation

The 2311 and 2314 disk resident systems are received on either a magnetic tape or disk pack. The tape resident system is received on a magnetic tape. The systems contain three libraries: core image, relocatable, and source statement. Some DOS distributions also contain a private source statement library and a private relocatable library. If either disk system is received on tape, it must be copied onto disk before the system generation procedure can begin.

The tape system core image library contains the supervisor and its associated transient routines, job control, and all other IBM-supplied language translators and service programs. In the disk system, the core image library contains the supervisor and its associated transient routines, Job Control, Linkage Editor, Librarian, Assembler, and selected utility programs. All programs in the core image library of both systems are edited to run with the IBM-supplied Supervisor, the size of which may be specified by the user (6K or 8K for the tape system; 6K for the 2311 or the 2314 disk systems).

The relocatable library contains IBM program modules. Among these programs are Job Control, Linkage Editor, Librarian, the language translators (Assembler, COBOL, etc, and the IOCS modules associated with COBOL, RPG, and PL/I (D)), and other service programs (Utilities, Sort/Merge, etc).

The source statement library contains macro definitions in the Assembler sublibrary. Among these are the supervisor

communications and logical IOCS macro definitions. It also contains macro definitions for generating the supervisor and sample problems for various processor programs.

The system received by the user is capable of immediate operation. However, most installations generate supervisors adapted to their configurations. Also, system libraries should be edited according to the needs of different installations. When this process is completed, the newly created system replaces the system that was received by the user.

Briefly, the system generation process is as follows. The user codes a set of supervisor macro instructions describing his system configuration. The assembly of these macro instructions results in a new supervisor which is then cataloged in the core image library of the system. Certain books in the source statement library (such as IOCS modules) may be assembled from macro definitions in the source statement library, and added to the relocatable library. In the disk system, the language translators and other service programs that the user desires are linkage edited to the core image library. The result of these operations is the installation's operational system.

Protection Facilities

A storage protection feature is supported optionally by Disk and Tape Operating Systems. Problem programs cannot modify main storage locations not assigned to them. Storage protection is required for multiprogramming systems. Problem programs operate in the problem state (as opposed to the supervisor state), where they cannot issue privileged instructions, including the resetting of storage protection keys.

Whenever different programs use the same system function, such as the supervisor itself, or use the same magnetic tape drives, there is danger that one program will obliterate information belonging to another through either programmer or operator error. Magnetic tapes used for such read only functions as system residence can be protected from overwriting by physical removal of their file protection rings. However, to prevent one program from overwriting tape files used by another program, volume labeling is an effective protective technique. Labeling also assures proper mounting of input and output data tapes. Magnetic-tape labeling is optional in the system.

Labeling is also used to identify and protect direct access storage devices. For these devices, labeling is mandatory. Since different programs may use different extents on the same direct access volumes, the disk system provides the facility for protecting extents on direct access volumes.

For those disk systems with at least 24K bytes of main storage, file protection is available for any DASD unit supported by the system. As file protection is performed by the supervisor, this optional facility must be specified when the system is generated. In a file protected system, as each program opens an extent on a DASD, the extent limits are written in a table of the supervisor. Whenever the problem program issues a READ or a WRITE to the DASD, the supervisor checks to make sure the area is within the extents opened.

In a multiprogramming environment, a track hold function provides protection for a DASD file being shared by more than one task or partition at the same time. The user can read a record from a disk and place a hold on that track to prevent any other task using the track hold function from accessing it. The protection remains in effect until the track is freed.

For multiprogramming within a partition, protection is provided when more than one task is manipulating data in the same area of main storage by preventing the data from being modified simultaneously by these tasks. The task manipulating data in the shared area first prevents other tasks from accessing it with a macro instruction. Subsequent requesting tasks using the same macro instruction are then queued and enter the wait state until the shared data is made available by the manipulating task. The requesting task with the highest priority then gains access to the data.

Other important protection features are built into the system. For instance, records starting with /* and /& in the system input stream (SYSRDR and SYSIPT) are always considered delimiters and cannot be used as data. Thus, one job cannot erroneously overread SYSIPT or SYSRDR and destroy the succeeding job.

Availability/Serviceability Facilities

Availability is a measure of a system's ability to continue processing during a given period. Serviceability facilities are those functions that analyze and correct system malfunctions. Two serviceability facilities are available in

a disk system with a minimum of 24K bytes of storage available. If specified, these options are generated as part of the supervisor at system generation time. Certain conditions, which previously prevented the system from continuing processing, can be overcome by using these features:

1. I/O Error Logging

   When an I/O error cannot be corrected after a standard number of retries, system environment data is recorded by the Outboard Recorder (OBR). This data is stored in a recorder file, defined on a system logical unit (SYSREC). The Statistical Data Recorder (SDR) records the cumulative error status of an I/O device. This data is also stored in the recorder file. Counters are retained in main storage that correspond to error counters in the record. When an I/O error occurs, counters in main storage are updated. Whenever any of the counters in main storage are filled, the contents of all the counters are added to the counters in the record for that device.

2. Machine Check Recording and Recovery (MCRR)

   Pertinent data is also recorded after a channel or Central Processing Unit (CPU) failure has occurred. The data is analyzed and the damaged partition(s) canceled. No attempt is made to retry on any error involving this condition.

RECORDER FILE. When either of the preceding features is specified, a recorder file must be created. Data that has been stored as records in the recorder file can be edited and printed using the environment record edit and print program (EREP).

TESTING I/O UNITS

The Disk and Tape Operating Systems provide a set of programs that may be used to test I/O units. These test programs are executed under control of the On Line Test Executive Program (OLTEP). This program acts as an interface between the system and the I/O unit tests and provides communication with the operator during the running of tests. OLTEP may be run in the background partition, thus permitting the user to operate foreground programs concurrent with its execution.

DISK OPERATING SYSTEM VOLUME STATISTICS

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

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

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

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

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

Either or both of these options can be specified by the user when the system is generated.

PROBLEM DETERMINATION

Problem determination is a process or a procedure for determining the cause of an error. Some DOS facilities, such as I/O Error Logging, MCRR, and the DUMP option of job control, are problem determination tools. Problem determination provided by DOS consists of recommending a specific procedure to be followed when an error condition occurs.

One group of programs recommended for error analysis is the Problem Determination Serviceability Aids (PDAID). PDAIDs allow users to trace one of the following events when it occurs:

- Fetching or Loading of other programs
- Input/Output activity

- Supervisor Calls (SVC); that is, communications between the control program and the problem program

Tracing consists of recording pertinent data when the event occurs. This data may be used for error analysis.

Another facility for problem determination is the DUMPGEN program. This program allows the user to generate a stand-alone dump program, tailored to his requirements for displaying the entire contents of main storage when processing under the Disk Operating System cannot continue.

Program Design, Preparation and Execution

The Disk and Tape Operating Systems allow the programmer flexibility in the design and preparation of programs. He may design his programs in segments that overlay each other, or as subprograms that may be individually coded, stored, and linked before execution. He specifies his requirements for necessary facilities by using Linkage Editor control statements and by including macro instructions in his Assembler language coding. Many of the facilities are also available in FORTRAN, COBOL, RPG, and PL/I (D) programs.

For example, when coding in the Assembler language, the programmer can use the FETCH macro instruction whenever a program requires another phase to be loaded for execution. Control is given to the fetched phase. The LOAD macro instruction can be used when a phase is to be loaded into main storage but not executed immediately. The LOAD macro instruction may be used to load tables, reference material, or executable program phases. After the phase is loaded, control is returned to the calling program.

The Assembler also provides for direct linkage between routines that are in main storage at the same time. Three macro instructions are used for direct linkage between routines: CALL, SAVE, and RETURN. Equivalent functions exist in several of the other languages. Linkage between the main program and two subroutines is shown in Figure 3. Linkage can proceed through as many levels as required and each routine may be called from any level. In the standard direct linkage, a routine always returns to the next higher level.

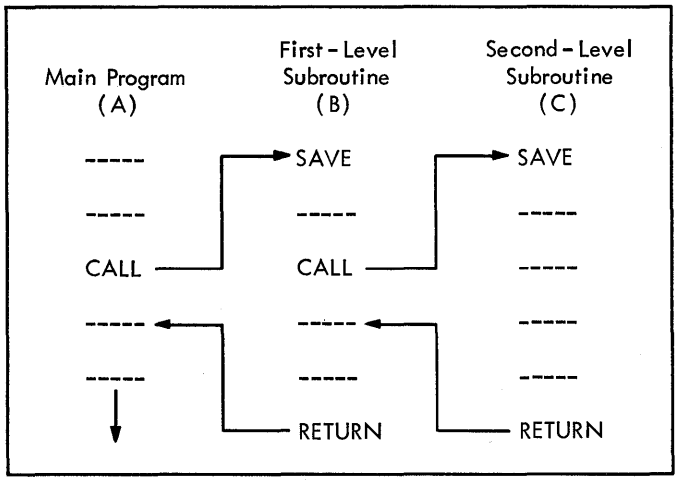

Figure 3. Direct Linkages

Figure 4 shows a program in the various stages of its development. A set of source statements that is processed by a language translator is referred to as a source program. A source program is compiled or assembled and the resulting output is called on object module. An object module can exist:

- in the form of a card deck
- written out on magnetic tape or disk, or
- cataloged into the relocatable library (through the Librarian Maintenance function which is a separate function).

All object modules must be processed by the Linkage Editor before they can be executed in the system.

The output of the Linkage Editor consists of one or more program phases. The linkage edited program can then be processed in one of three ways:

1. It can be executed immediately (from the Link-and-Execute file).
2. It can be cataloged into the core image library through the Librarian maintenance function.
3. In the tape system, it can be executed or cataloged at a later time from the "go" file.

Once a program is in the core image library, it can be retrieved easily by simple control statements or other program phases.

Program text and data can be preserved when calling one program phase from another phase. For this, the user organizes his program into an overlay structure so that one phase, the root phase, remains in main storage during the entire time the program is executed. Subsequent phases may partially or completely overlay each other, normally beginning just beyond the root phase. The user defines these and other overlay structures by using Linkage Editor control statements.

Information can also be passed between phases or from a main program to a subprogram by defining a common area. When used, the common area is in the low part of the problem program area. This facility is of particular importance when coding

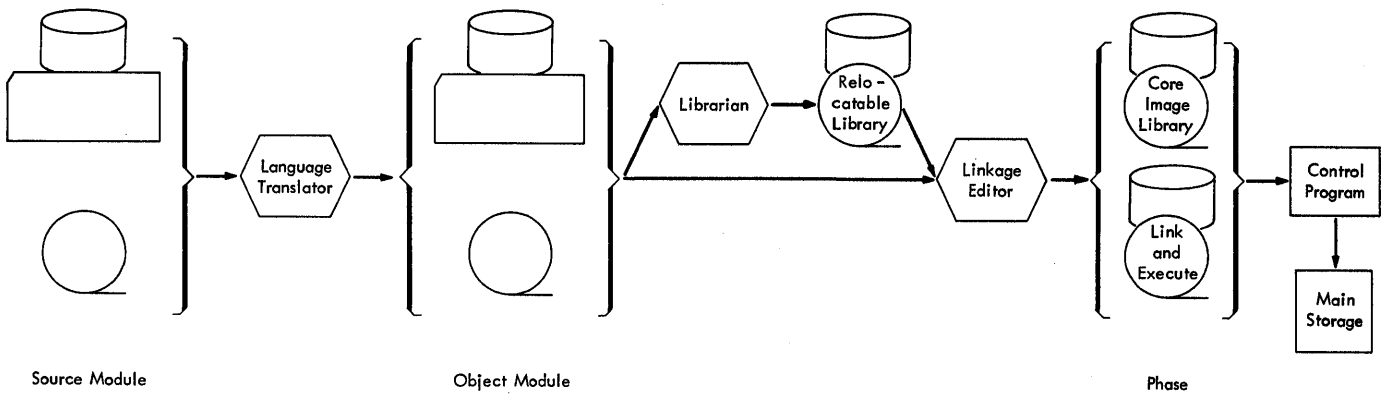

Figure 4. Program Stages

programs in the Assembler and FORTRAN languages.

An example of program overlays is shown in Figure 5. Each branch (PHASEA, PHASEB, PHASEC, PHASED, and PHASEE) of the tree structure represents a program phase. The vertical length of each branch represents the amount of main storage used by each phase.

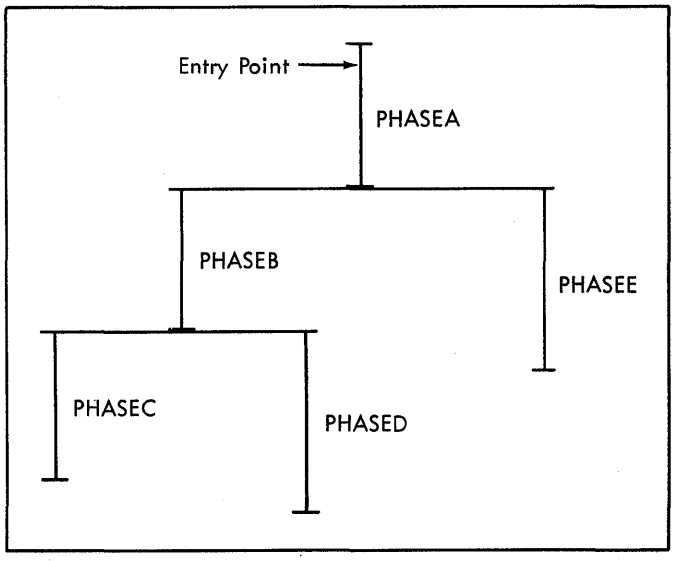

Figure 5. Overlay Tree Structure

Branch PHASEA represents a root phase. A root phase remains in main storage at all times during program execution. PHASEB and PHASEE occupy the same area of main storage, but at different times during the course of program execution. The same is true of PHASEC and PHASED. The following sequence of Linkage Editor control statements can be used to build this program overlay structure.

```
PHASE  PHASEA,ROOT

PHASE  PHASEB,*

PHASE  PHASEC,*

PHASE  PHASED,PHASEC

PHASE  PHASEE,PHASEB
```

If two programs are related only by external files (on cards, magnetic tape reels, or DASD volumes), it may be impossible (or undesirable) to connect them within an overlay structure. These files may be operated upon by two or more distinct programs, together comprising a single job; that is, the accounting unit for computer usage, or alternatively, the minimal unit submitted by one programmer at one time.

Each job comprises one or more job steps, that is, executions of single programs. For example, the only relationship existing between a compiling function and a linkage editing function is an object module, always stored on an external device (magnetic tape, DASD, or cards). Hence, consecutive compilations and linkage editing comprise job steps of a single job. (An additional job step could be the execution of the linkage edited program.) At the completion of each job step, the problem program area of main storage is set to zero.

If two programs are completely independent of each other, they may be set up as two job steps or as two separate jobs. The latter procedure is recommended because cancelation of a job step generally causes cancellation of the entire job.

Figure 6 further illustrates the job and job step concept. EXAMPLE1 consists of the executions of three programs: COBOL compiler (Step 1), Linkage Editor (Step 2), and the linkage edited object program (Step 3). These three job steps comprise Job 1. Job 2 (EXAMPLE2) consists only of the execution of the linkage edited program PAYROLL, stored in the core image library.

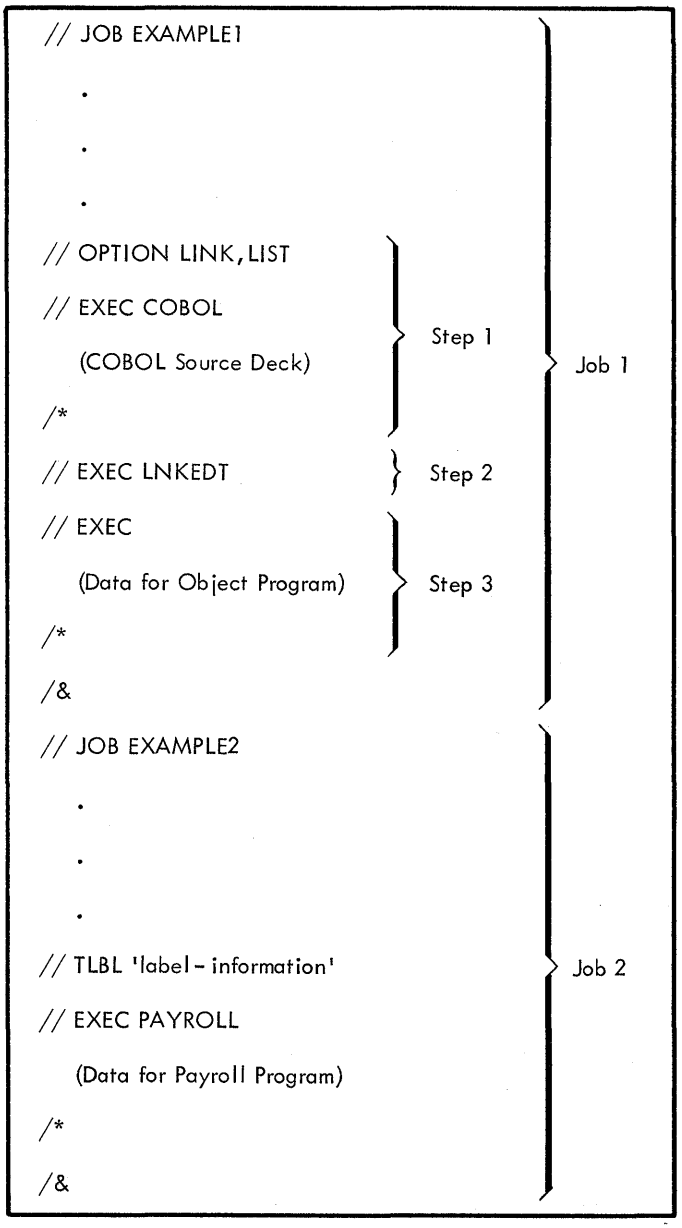

Figure 6. Job Sample

PHASE STRUCTURE

Input for an individual phase is one or more object modules. In many cases, inclusion in main storage of all control sections from a specific module is inappropriate. One alternative is to compile each control section separately, thus forming separate object modules. Because this tactic hampers use of a common symbol table and may complicate cross referencing of symbols, the Linkage Editor permits the user to select specified control sections from one or more modules for inclusion in a phase. The appropriate control statements describe these submodular phases. An example of submodular structure is shown in the section entitled Linkage Editor.

A control section is never included in a phase more than once. In addition, if a control section appears in the root phase (the phase remaining in main storage during the entire time a program is being executed), that control section appears in no other phase in the program. The Linkage Editor, however, allows the inclusion of the same control section within each of several phases other than the root phase.

PROGRAM DEBUGGING

Machine stops and "hands-on" debugging of programs are disadvantageous in any operating system. The Disk and Tape Operating Systems offer certain tools for automated, remote testing. For example, using COBOL, the user can request test information in the higher level language by using the debugging packet. Using FORTRAN, Basic FORTRAN, and PL/I (D), the user can readily insert temporary READ and WRITE statements to monitor test data. The system also offers storage dump capabilities for debugging (PDUMP and DUMP).

The Autotest debugging facility is particularly suited to assembly output, although it can also help debug object modules from other translators. Since executable programs in the system are linkage edited and loaded from the same absolute addresses, testing tools need not be fully symbolic. Main storage snapshot requests can be inserted into a program when it is linkage edited under Autotest control. As each snapshot is taken, its address and limits for any storage area to be displayed can be specified symbolically.

Other functions of Autotest are:

1. Dumping main storage at termination of a job step.
2. Recording phase retrieval.
3. Servicing test data with utility programs before or after the test execution(s).

The interruption features of IBM System/360 enable the Supervisor to interrupt many program errors immediately, whether or not Autotest is being used. If Autotest is not used, the supervisor alone can issue a diagnostic message to the programmer, and if requested by the DUMP parameter in the job control OPTION statement, dump main storage.

OPERATING CONSIDERATIONS (TAPE OPERATING SYSTEM ONLY)

Additional operating efficiency for Tape Operating System functions can be achieved by those installations with more than the minimum four tape units.

Additional magnetic tape units (if 9-track or 7-track with the data convert feature) may be used for:

1. Compile-and-execute functions, or
2. A private relocatable library, or
3. A private source statement library, or
4. System input, printing, or punching.
5. Set-up for following jobs.

If a compilation (RPG for example) is being performed, it may be advantageous to use the additional tape for a compile-and-execute function.

If a linkage editing function is being performed, the additional magnetic tape can be used effectively as a private relocatable library.

If either an Assembly or a COBOL compilation is being performed, the user may want to use extra tape unit(s) for a private source statement library and/or compile-and-execute capability.

Note that Basic FORTRAN compile-and-execute capability is extended to users with the minimum configuration (4 magnetic tape units), since FORTRAN requires only one work tape for compilations.

PERFORMANCE IN A MULTIPROGRAMMING ENVIRONMENT

The throughput of a system, the rate and efficiency at which a group of jobs may be processed, is a measure of its performance. Multiprogramming has as its principal objective the increase of throughput through more efficient use of resources. Because the system is capable of dividing processing time among three programs, the total amount of time required to process multiple jobs may be considerably reduced in a multiprogramming environment.

Performance in this environment may be enhanced or impeded, depending on several factors that fall into three categories:

1. System configuration.
2. Programming.
3. Operational considerations.

Effect of System Configuration on Performance

In multiprogramming, the number and size of the problem program areas (partitions) are established externally. The partitions are defined during the system generation process or the initialization process, but may subsequently be altered by the operator to satisfy varying requirements of problem programs during system operation. The size of a partition depends on the amount of main storage available for partition definition, and on the size of problem programs. For this reason, a large main storage capacity permits large partition definitions, each of which can permit large user programs.

In a multiprogramming environment, the system overlaps I/O operations and processing for the programs being processed. For example, at the same time high-priority (foreground) programs are performing I/O operations, a low-priority (background) program may be performing internal processing functions. The higher the speed of the CPU, the more CPU processing time is made available, resulting in increased performance of lower priority programs.

The increase in I/O activity that is a result of multiprogramming naturally results in some degree of channel contention. By having additional selector channels, the I/O load may be split, allowing parallel input/output operations.

Effect of Program Assignment on Performance

Of prime importance to multiprogramming performance is the selection of job priority. Programs that normally are more suited to run as foreground programs (that is, those programs given high priority) are those that require a considerable amount of input/output operations on low-speed devices. Teleprocessing programs are good candidates for high priority. Programs that require a considerable amount of internal processing normally should not be run in high priority as they can degrade total throughput by monopolizing CPU usage.

Programs that use information contained on the same direct access device, if run concurrently, cause speed degradation due to excessive seek time.

In a multiprogramming environment, it is particularly desirable that frequently used programs be self-relocating. A self-relocating program is one that can be executed at any location in main storage by having an initialization routine to modify all address constants at execution time. Although self-relocating programs require additional programming effort, the advantages may compensate for the extra effort, particularly when a system utilizes two foreground areas. The self-relocating program may be executed in any of the three partitions, and the operator need not be aware of the origin address established by the Linkage Editor to partition main storage correctly for execution.

Note that non-self-relocating programs may be linkage edited for more than one partition simply by changing the program name on their PHASE and FETCH statements. Subsequent EXEC statements would then refer to the appropriate program name.

Operational Effects on Performance

Performance is significantly improved if both foreground partitions are operating in batched job mode, providing, of course, that the user has sufficient main storage available. Very little operator intervention is required in this mode of operation because batched job initiation in the foreground area is provided by Job Control. Multiprogramming is more effective if sufficient planning is given to storage allocation (partition definition) and foreground initiation before the actual running of the job.

Alteration of partition definitions by the operator to satisfy problem program requirements during system operation should be kept at a minimum. Fixed storage allocation should be made at system generation time or at initialization time, with partitioning preplanned for a number of jobs. Consider requirements of many programs, rather than for a particular job. During operations, it is simplest to run with one storage allocation for the following reasons:

- Changing the storage allocation affects throughput since at least one area, and often two or three, cannot be active during a storage allocation.
- The operator need not be concerned with linkage edit addresses for programs, even if they are not self-relocating. He merely must know the area for which they were linkage edited.

When multiple allocations are used, it is desirable to restrict the allocations either to a few unique allocations, or keep the starting addresses of one or more partitions fixed. Otherwise, there might be difficulty in scheduling to achieve throughput.

Single foreground programs are initiated via the printer-keyboard. For certain foreground programs (such as teleprocessing programs that operate for extended periods of time and seldom require foreground initiation), the printer-keyboard normally is sufficient. If foreground initiation by the operator is frequently required to minimize initiation time, a card reader is probably the more desirable device.

For installations with only one card reader, it is possible to place batched job input job streams on high-speed devices, thus freeing the card reader for foreground initiation. A combined SYSRDR/SYSIPT (SYSIN) file can be created for placing both job control statements and input job streams on either disk or tape. If large volumes of varying input may be processed by one set of job control information, the job control information can be stored on a disk extent assigned to SYSRDR, and SYSIPT can be assigned to a tape unit.

If background input job streams have been placed on high-speed devices, and the user's background program makes reference to an input device as a particular device type, the program should be examined for conflicting device specifications. For example, the device type for an input file, if specified within the program as a card reader, would have to be altered to agree with the high-speed device selected for input of job streams.

The operating guide appropriate to the system contains examples of single program initiation.

Disk system foreground programs should use the Job Control standard label option. The Disk Operating System provides both standard label tracks for all partitions and a standard label track for each partition, onto which label definition statements may be placed. Any program that may use these labels can then be initiated without the need of resubmitting them.

Symbolic device assignments may be held from one foreground job to another. If the programmer uses identical assignments for operation in foreground areas, the holding of device assignments across jobs will simplify foreground initiation.

Multiprogramming Summary

Figure 7 is a summary of the characteristics and system facilities applicable to the available partitions in a multiprogramming environment.

Characteristic or Facility	Background (BG)	Either Foreground with BJF (DOS)	Either Foreground (SPI) (DOS or TOS)
Program Initiation	Automatic	Automatic[6]	Operator
Job Control Unit	SYSRDR/IPT Card/Tape/Disk	SYSRDR/IPT Card/Tape/Disk	1052 or Card Reader
Storage Protection	Yes	Yes	Yes
Storage Protect Key	1	F1: 3 F2: 2	F1: 3 F2: 2
Location (Fixed)	Adjacent to System Supervisor	F1: Adjacent to Storage End F2: Adjacent to F1	F1: Adjacent to Storage End F2: Adjacent to F1
Size: (Variable)			
Minimum (Active)	10K Bytes	10K Bytes	2K Bytes
Minimum (Inactive)	10K Bytes	0 Bytes	0 Bytes
Maximum	avail. storage	510K Bytes	510K Bytes
Intervals	2K Bytes	2K Bytes	2K Bytes
Established by	SYSGEN	SYSGEN	SYSGEN
Alterable by	Operator	Operator	Operator
Priority	3 (lowest)	F1: 1(highest) F2: 2	F1: 1(highest) F2: 2
Use of System Functions:			
System Log	Yes	Yes[1]	Yes[1]
Job Logging	Yes	Yes	No
System Units (I/O)	Yes	Yes except SYSLNK	Yes except[7] SYSLNK (DOS) No for TOS
Programmer Units	Yes	Yes	Yes
Operator Inquiry	Yes	Yes	Yes
IOCS Macros	Yes	Yes	Yes
MPS Utility Macros	Yes	Yes	Yes
BTAM Macros (DOS only)	Yes	Yes	Yes
QTAM Macros (DOS only)	Yes[4]	Yes[4]	Yes[4]
Fetch/Load	Yes	Yes	Yes
Program Check Exit	Yes	Yes	Yes

Figure 7. Multiprogramming Summary (Part 1 of 2)

Characteristic or Facility	Background (BG)	Either Foreground with BJF (DOS)	Either Foreground (SPI)(DOS or TOS)
Use of Sys. Functions (cont.)			
Interval Timer Exit[3]	Yes	Yes	Yes
Time of Day[2]	Yes	Yes	Yes
Dump (PDUMP)	Yes	Yes	Yes
Communication Region	Yes	Yes	No
Checkpoint/Restart	Yes	Yes	No
Use of System Components:			
System Service	Yes	No	No
Language Translators	Yes	No	No
Tape Sort/Merge	Yes	No	No
Disk Sort/Merge	Yes	No	No
Tape & Disk Sort Merge	Yes	Yes	No
Utilities	Yes	Yes	No
Autotest	Yes if foreground partitions are 0 bytes	No	No
Emulator	Yes	Yes	No
Use of Problem Programs: Compiled by:			
Assembler	Yes	Yes	Restricted[5]
COBOL D	Yes	Yes	Yes (DOS) Restricted[5] (TOS)
RPG	Yes	Yes	Restricted[5]
FORTRAN	Yes	Yes	No
Basic FORTRAN	Yes	Yes	No
PL/I (D)	Yes	Yes	No
ANS COBOL	Yes	Yes	Yes (DOS)
Types of Program Loading:			
Absolute	Yes	Yes	Yes
Self-Relocating	Yes	Yes	Yes
Error Logging Capabilities:			
Environment Record Edit and Print (EREP)	Yes (DOS)	No	No
OLTEP	Yes	No	No
PDAID	Yes (DOS)	Yes	Yes (DOS)
DUMPGEN	Yes (DOS)	No	No

Notes

[1]SYSLOG must be assigned to printer-keyboard.

[2]Assumes that the Timer Feature is available.

[3]Only one partition at a time selectable by the operator.

[4]QTAM Message Control Program must operate as Foreground-one. Message Processing Programs may operate as Background and/or Foreground-two.

[5]Restricted to those language functions that do not require access to a communication region, or for the tape system, symbolic system units.

[6]Foreground is initiated into Batch Job mode by operator; thereafter, program initiation is automatic from SYSRDR assigned to that partition.

[7]Assignments limited to card readers, card punches, and printers.

Figure 7. Multiprogramming Summary (Part 2 of 2)

Glossary

For a more complete list of data processing terms, refer to IBM Data Processing Techniques, A Data Processing Glossary, GC20-1699.

Access Method: Any of the data management techniques (sequential, indexed sequential, or direct) available to the user for transferring data between main storage and an input/output device.

Active Program: Any program that is loaded and ready to be executed.

Addressing: A technique by which the central processing unit prepares a remote station to receive a message.

Answering: A remote station has the ability to dial the central processing unit through the switched network facilities. The CPU answers the call.

ASCII (American National Standard Code for Information Interchange): A 128-character, 7-bit code. The high-order bit in the System/360 8-bit environment is zero.

Asynchronous: Without regular time relationship. The user's programs run asynchronously with the I/O interruptions. BTAM's channel appendage routine runs synchronously with the I/O interruptions.

Background Program: In multiprogramming, the background program is the program with lowest priority. Background programs execute from a stacked job input.

Basic Telecommunications Access Method (BTAM): A basic access method that permits a READ/WRITE communication with remote devices.

Batched Job: Programs that execute from a stacked job input. Batched jobs run under the control of job control.

Block:

1. To group records physically for the purpose of conserving storage space or increasing the efficiency of access or processing.

2. A physical record on tape or DASD.

Book: A group of source statements written in the Assembler or COBOL language.

Buffer:

1. A storage device in which data is assembled temporarily during data transfers. It is used to compensate for a difference in the rate of flow of information or the time occurrence of events when transferring information from one device to another. For example, the IBM 2821 Control Unit (a control and buffer storage unit for card readers, card punches, and printers in a System/360).

2. Program input/output. A portion of main storage into which data is read or from which it is written.

Buffer Pool: That area of main storage reserved per line group for buffering.

Burst Mode: A means of transferring data to or from a particular I/O device on either the multiplexor or selector channel. All channel controls are monopolized for the duration of data transfer.

Byte Mode: See Multiplex Mode.

Catalog: To enter a phase, module, or book into one of the system libraries.

Channel Program: One or more Channel Command Words (CCWs) that control(s) a specific sequence of channel operations. Execution of the specific sequence is initiated by a single start I/O instruction.

Channel Scheduler: The part of the supervisor that controls all input/output operations.

Checkpoint: A point in a program at which sufficient information can be stored to permit restarting the job step from that point.

Checkpoint Record: Records that contain the status of the job and the system at the time the records are written by the checkpoint routine. These records provide the necessary information for restarting a job without having to return to the beginning of the job.

Checkpoint/Restart: A means of restarting execution of a program at some point other than the beginning. When a checkpoint macro instruction is issued in a problem program, checkpoint records are created. These records contain the status of the

program and the machine. When it is necessary to restart a program at a point other than the beginning, the restart procedure uses the checkpoint records to reinitialize the system.

Checkpoint Routine: A routine that records information for a checkpoint.

Command Control Block: A sixteen-byte field required for each channel program executed by physical IOCS. This field is used for communication between physical IOCS and the problem program.

Communication Region: An area of the supervisor set aside for interprogram and intraprogram communication. It contains information useful to both the supervisor and the problem program.

Control Program: A group of programs that provides functions such as the handling of input/output operations, error detection and recovery, program loading, and communication between the program and the operator. IPL, supervisor, and job control make up the control program in the Disk and Tape Operating Systems.

Control Section: The smallest separately relocatable unit of a program; that portion of text specified by the programmer to be an entity, all elements of which are to be loaded into contiguous main storage locations.

Core Image Library: An area of the resident system device used to store programs that have been processed by the Linkage Editor. Each program is in a form identical to that which it must have to be executable in main storage.

Core Storage: See Main Storage.

Data File: A collection of related data records organized in a specific manner. For example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc) or an inventory file (one record for each inventory item, showing the cost, selling price, number in stock, etc).

Device Independence: The capability of a program to process the same type of data on different device types (punched card devices/printers, tape, or disk).

Disk Operating System: A disk resident system that provides operating system capabilities for 16K and larger System/360 systems.

DUMP: Displaying the contents of main storage.

Emulator: The combination of programming techniques and special machine features that permits a given computing system to execute programs written for another system.

Event: An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as input/output.

Event Control Block (ECB): A control block used to represent the status of an event.

Extent: The physical locations on Input/Output devices occupied by or reserved for a particular file.

External Reference: A reference to a symbol used in another module. The external references are resolved when the respective modules are combined at linkage edit time.

Fetch:

1. To bring a program phase into main storage from the core image library for immediate execution.

2. The routine that retrieves requested phases and loads them into main storage (see System Loader).

3. The name of a macro instruction (FETCH) used to transfer control to the System Loader.

File: See Data File.

Fixed Length Record: A record having the same length as all other records with which it is logically or physically associated.

Foreground Initiation: A set of system routines to process operator commands for initiation of a foreground program.

Foreground Initiator: See Single Program Initiator (SPI).

Foreground Program: In multiprogramming, foreground programs are the highest priority programs. Foreground programs may be executed from a job stack or in an SPI environment.

Inactive Program: A program that is loaded but not ready to be executed, or a program not loaded in the system.

Initial Program Loading (IPL): The initialization procedure that causes Disk and Tape Operating Systems to commence operation.

Input Job Stream: A sequence of job control statements entering the system, which may also include input data.

Input/Output Control System (IOCS): A group of macro instruction routines provided by IBM for handling the transfer of data between main storage and external storage device. IOCS consists of two parts: physical IOCS and logical IOCS.

Interruption: A break in the normal sequence of instruction execution. It causes an automatic transfer to a preset storage location where appropriate action is taken.

I/O Area: An area (portion) of main storage into which data is read or from which data is written. In Operating System publications, the term buffer is often used in place of I/O area. I/O means Input/Output.

IPL Loader: A program that reads the supervisor into main storage and then transfers control to the supervisor.

Job Control: A program that is called into storage to prepare each job or job step to be run. Some of its functions are to assign I/O devices to certain symbolic names, set switches for program use, log (or print) job control statements, and fetch the first program phase of each job step.

Job Control Statement: Any one of the control statements in the input stream that identifies a job or job step, or defines its requirements and options.

Job Statement (JOB): The control statement in the input stream that identifies the beginning of a series of job control statements for a single job, i.e., a single accounting unit of processing.

Job Step: The execution of a single processing program.

K: 1024.

Language Translators: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent machine language instructions. For example, Assembler, COBOL, etc are language translators.

Librarian: The set of programs that maintains, services, and organizes the system libraries.

Library: An organized collection of programs, source statements, or object modules maintained on the system resident device. Three libraries are used by the Disk and Tape Operating Systems: core image library, source statement library, and relocatable library.

Linkage Editor: A system service program that edits the output of language translators and produces executable program phases. It relocates programs or program sections and links together separately assembled (or compiled) sections.

Load: To fetch, i.e., to read a phase into main storage returning control to the calling phase.

Logic Module: The logical IOCS routine that provides an interface between a processing program and physical IOCS.

Logical File: A data file that has been described to the Disk or Tape Operating Systems through the use of a file definition (DTF) macro instruction. Note that a data file is described to Operating System through a different defining method. Operating System publications refer to a data file described in this different manner as a data set.

Logical IOCS: A comprehensive set of macro instruction routines provided to handle creation, retrieval, and maintenance of data files.

Logical Record: A record identified from the standpoint of its content, function, and use rather than its physical attributes. It is meaningful with respect to the information it contains. (Contrasted with Physical Record.)

MPS: Multiprogramming System.

Main Storage: All addressable storage from which instructions can be executed or from which data can be loaded directly into registers.

Main Task: The main program within a partition in a multiprogramming environment.

Module (Programming): The input to or output from a single execution of a language translator or the Linkage Editor; a separate program unit that can be combined with other units.

Multiplex Mode: A means of transferring records to or from low-speed I/O devices on the multiplexor channel, by interleaving bytes of data. The multiplexor channel sustains simultaneous I/O operations on several subchannels. Bytes of data are interleaved and then routed to or from the selected I/O devices or to and from the desired locations in main storage.

Multiplex mode is sometimes referred to as byte mode.

Multiplexor Channel: A channel designed to operate with a number of I/O devices simultaneously on a byte basis. That is, several I/O devices can be transferring records over the multiplexor channel, time sharing it on a byte basis.

Multiprogramming System: A system that controls more than one program simultaneously by interleaving their execution.

Multitask Operation: Multiprogramming; called multitask operation to express not only concurrent execution of one or more programs in a partition, but also of a single reenterable program used by many tasks.

Object Module: The output of a single execution of a language translator; it constitutes input to the Linkage Editor. An object module consists of one or more control sections in relocatable, nonexecutable form. An object module must be processed by the Linkage Editor before it can be executed in the system.

Operating System: A collection of programs that enables a data processing system to supervise its own operations, automatically calling in programs, routines, languages, and data as needed for continuous throughput of a series of jobs.

Overlap: To do something at the same time that something else is being done; for example, to perform input/output operations while instructions are being executed by the central processing unit.

Overlay:

1. A segment (phase) of a program loaded into main storage, replacing all or part of a previously retrieved section.

2. The technique of repeatedly using blocks of internal storage during different stages of a problem. For example, when one routine is no longer needed in internal storage, another routine can replace all or part of that routine.

Overrun: Overrun is the condition caused by the inability of a physical device to complete data transfer in the time allotted to the device by the system.

Phase: The smallest complete unit that can be referenced in the core image library. Each overlay of a program or (if the program contains no overlays) the program itself is a single complete phase.

Physical IOCS: Macro instructions and supervisor routines (Channel Scheduler) that schedule and supervise the execution of channel programs. Physical IOCS controls the actual transfer of records between the external storage medium and main storage, and provides I/O device error recovery.

Physical Record: A record identified from the standpoint of the manner or form in which it is stored and retrieved; that is, one that is meaningful with respect to access. (Contrasted with Logical Records.)

Polling: The process of inviting stations within a data link to transmit messages one at a time in an orderly fashion. The basic function of polling is to prevent contention by ensuring that only one station transmits at a time.

Private Library: A relocatable or source statement library that is separate and distinct from the system library.

Problem Determination: A procedure or process (provided by IBM) that the user can follow after an error message to determine the cause of that error.

Problem Program:

1. The user's object program. It can be produced by any of the language translators. It consists of instructions and data necessary to solve the user's problem.

2. A general term for any routine that is executed in the data processing system's problem state; that is, any routine that does not contain privileged operations. (Contrasted with Supervisor.)

Processing Program: A general term for any program that is both loaded and supervised by the control program. Specifically, a collection of certain IBM supplied programs: the language translators, Linkage Editor, Librarian, Autotest, Sort/Merge and Utilities. All user written programs are processing programs. The term processing programs is in contrast to the term control program.

Queue: A list of entries, usually ordered in the sequence of arrival. The entries identify things contending for service or attention.

Record: A general term for any unit of data that is distinct from all others when considered in a particular context.

Reenterable: The attribute of a set of code that allows the same copy of the set of code to be used concurrently by two or more tasks.

Relocatable: A module or control section whose address constants can be modified to compensate for a change in origin.

Relocatable Library Module: A module consisting of one or more complete control sections cataloged as a single entry in the relocatable library.

Resource: Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the central processing unit, data files, and control and processing programs.

Restart: See Checkpoint/Restart.

Selector Channel: A channel designed to operate with only one I/O device at a time. Once the I/O device is selected, a complete record is transferred one byte at a time.

Self Relocating: A programmed routine that is loaded at any double-word boundary and can adjust its address values so as to be executed at that location.

Self Relocating Program: A program that is able to run in any area of storage by having an initialization routine to modify all address constants at object time.

Service Programs: Any of the class of standard routines that assist in the use of a data processing system and successful execution of problem programs. These programs include Autotest, Sort/Merge, and Utilities.

Single Program Iniator (SPI): A program that is called into storage to perform job control type functions for foreground programs not executing in batch job mode.

Source Module: A set of source statements in the symbolic language of a language translator that constitutes the entire input to a single execution of the language translator.

Source Statement: Statements written by a programmer in symbolic terms related to a language translator.

Source Statement Library: A collection of books (such as macro definitions) cataloged onto the system by the Librarian.

Stacked Job Processing: A technique that permits multiple jobs to be grouped (stacked) for presentation to the system. This allows the system to automatically process each job in sequence.

Submodular Phase: A phase made up of selected control sections from one or more modules as compared with a normal phase that is made up of all control sections from one or more modules.

Subtask: A task in which control is initiated by a main task by means of a macro instruction that attaches it.

Supervisor: A component of the control program. It consists of routines to control the functions of program loading, machine interruptions, external interruptions, operator communications and physical IOCS requests and interruptions. The supervisor alone operates in the privileged (supervisor) state. It coexists in main storage with problem programs.

Symbolic I/O Assignment: A means by which problem programs can refer to an I/O device by a symbolic name. Before a program is executed, job control can be used to assign a specific I/O device to that symbolic name.

System Inquiry: The function of operator initiated communication to a problem program.

System Loader: One of the supervisor routines. It is used to retrieve program phases from the core image library and load them into main storage.

System Residence: The external storage space allocated for storing the basic operating system. It refers to an on-line tape reel or disk pack that contains the necessary programs required for executing a job on the data processing system.

System Service Programs: Programs that perform portions of the functions of generating the initial basic operating system, generating specialized systems, creating and maintaining the library sections, and loading and editing programs onto the resident device. These programs are: linkage editor and librarian.

Tape Operating System: A tape resident system that provides operating system capabilities for 16K and larger System/360 systems.

Tape Resident System: An operating system that uses magnetic tape for on-line storage of system routines.

Task Selection: The supervisor mechanism for determining which program should gain control of CPU processing.

Telecommunications: A general term expressing data transmission between remote locations.

Teleprocessing: A term associated with IBM telecommunication systems expressing data transmission between a computer and remote devices.

Throughput: A measure of system efficiency; the rate at which a series of jobs can be handled by a data processing system.

Trace:

1. To record a series of events as they occur.
2. The record of a series of events.

Transient Area: This is a main storage area (within the supervisor area) used for temporary storage of transient routines.

Transient Routines: These self-relocating routines are permanently stored on the system residence device and loaded (by the supervisor) into the transient area when needed for execution.

Undefined Record: A record having an unspecified or unknown length.

Variable Length Record: A record having a length independent of the length of other records with which it is logically or physically associated. (Contrasted with Fixed Length Record). It contains fields specifying physical and logical record lengths.

Volume: That portion of a single unit of storage media that is accessible to a single read/write mechanism. For example, a reel of magnetic tape on a 2400-series magnetic tape drive or a disk pack on an IBM 2311 Disk Storage Drive.

Wait Condition: As applied to tasks, the condition of a task that it is dependent on an event or events in order to enter the ready condition.

Index

Indexes to systems reference library manuals are consolidated in the publication IBM System/360 Disk Operating System Master Index, GC24-5063. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

DOS & TOS/360 Concepts and Facilities (S360-20) Printed in U.S.A. GC24-5030-8

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Reader's Comments Form

IBM System/360
Disk and Tape Operating Systems
Concepts and Facilities

GC24-5030-8

Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Please give specific page and line references with your comments when appropriate. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM branch offices.

	Yes	*No*
● Does the publication meet your needs?	☐	☐
● Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

● What is your occupation? _____

● How do you use this publication:

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Your comments:

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. *If you would like a reply, please supply your name and address on the reverse side of this form.*

Staple

GC24-5030-8

Your comments, please . . .

This publication is one of a series that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, help us produce better publications for your use. Each reply is carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in using your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13760

Attention: Programming Publications, Dept. 157

Fold Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____

Department _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Cut Along Line

DOS & TOS/360 Concepts and Facilities (S360-20) Printed in U.S.A. GC24-5030-8