GC33-5370-6
File No. S370/S4300-20

Systems

Introduction to DOS/VSE

IBM

Seventh Edition (January, 1979)

This is a major revision of, and obsoletes, GC33-5370-5.

This edition applies to the IBM Disk Operating System/Virtual Storage Extended (DOS/VSE) and to all subsequent versions and releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein. Before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

The publication has been completely revised, and chapters of interest should be read in their entirety.

Note: *For the availability dates of features and programming support described in this manual, please contact your IBM representative or the IBM branch office serving your locality.*

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 Laboratory, Programming Publication Department, Schoenaicher Strasse 220, 7030 Boeblingen, Germany. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

This Manual...

...is a general discussion of the IBM Disk Operating System/Virtual Storage Extended (DOS/VSE). Its purpose is to provide new users of DOS/VSE with a basic introduction to the major computing services available with the minimum DOS/VSE support: The available system control programming support (DOS/VSE SCP) and the licensed VSE/Advanced Functions. For users familiar with DOS or DOS/VS, it also gives a summary of the features and functions new in DOS/VSE.

The information in this manual is presented as follows:

- A brief discussion of what DOS/VSE is and what its major computing services are — **Chapter 1.**
- A discussion of the available computing services, highlighting their usefulness whenever appropriate — **Chapter 2.**
- An overview of the system generation process — **Chapter 3.**
- A discussion of the advantages of DOS/VSE over DOS — **Chapter 4.**
- A discussion of program compatibility between DOS and current DOS/VSE as well as DOS/VSE Release 34 or earlier release and current DOS/VSE — **Chapter 5.**
- An overview of licensed IBM and nonlicensed programming support packages available for use at a DOS/VSE installation — **Chapter 6.**
- An overview of devices supported by DOS/VSE — **Appendix A.**
- An overview of support added in recent SCP releases of DOS/VS and available with DOS/VSE SCP — **Appendix B.**
- An overview of the documentation for DOS/VSE, including a recommended reading path — **Appendix C.**

The reader is expected to have a basic knowledge of data processing. Supplementary information about hardware functions and machine instructions may be found in the applicable principles of operation manuals:

IBM System /370 Principles of Operation, GC22-7000

IBM 4300 Processors, Principles of Operation, GA22-7070

Contents

List of Figures

Chapter 1: The Disk Operating System/Virtual Storage Extended

The Disk Operating System/Virtual Storage Extended (DOS/VSE) consists of (1) system control programming (SCP) support, (2) the licensed VSE/Advanced Functions support and (3) any IBM supplied and user-written DOS/VSE programs that may be required to meet your installation's data processing needs. Although you may run your system without VSE/Advanced Functions in an SCP only environment, it is by installing this licensed programming support that you create a specified operating environment and thus ensure optimum utilization of your data processing system.

DOS/VSE and the hardware controlled by DOS/VSE combine to form a complete, effective computing facility. The DOS/VSE SCP with VSE/Advanced Functions provide the support needed for processing in a multi-programming environment; it includes the potential for build-up of additional computing power and for adaptation to changing data processing requirements. By including in your installation's DOS/VSE, for example, the VSE/Interactive Computing and Control Facility (VSE/ICCF), you change the installation's operating characteristics from batch to interactive. By additionally installing the VSE/Access Control — Logging and Reporting program, you can reduce considerably the risk of inadvertent or intentional misuse of data. More information about available licensed programming support is given in *Chapter 6: Licensed and Nonlicensed Programming Support.*

Through the system generation procedure, you can tailor the IBM supplied DOS/VSE SCP support to complement your installation's hardware and to meet the operational requirements of IBM supplied and user-written programs that you want to include in your DOS/VSE.

DOS/VSE operates on any IBM central processor (frequently referred to as CPU) that supports the virtual storage concept and is equipped with a processor storage of at least 160K bytes. If DOS/VSE operates in System/370 mode, that processor's storage must not exceed 8192K bytes.

Summary of Available Computing Services

The DOS/VSE SCP and VSE/Advanced Functions are designed to make efficient use of a hardware system. They relieve programmers and operators of a great deal of manual work. The programs and routines of this combined support, also referred to as components, are stored online in areas on disk, called libraries; they are immediately and directly accessible when they are needed in the course of your daily operations.

Following is a brief overview of the major computing services available with the components of this combined support.

- **Initial program load.**
  This function initiates operation with the computing system under control of DOS/VSE. The function includes automatic system services such as loading print control buffers and setting controls in accordance with the existing hardware configuration.

- **Resource management.**
  The supervisor of DOS/VSE, which is loaded into processor storage by the initial program load function, controls overall system operation. Primarily, the supervisor controls the utilization of system resources (such as processor storage, processing time, disk storage) by problem programs and programs of the operating system.

- **Job control.**
  Before a problem program can be executed, the supervisor loads the job control program, one of the components of DOS/VSE. This program determines which of the system resources were used by the previously executed problem program and are now available again. Job control also determines which system resources will be required by the problem program that is about to be executed. A problem program in this manual's context is any program whose execution is invoked through the job control program.

- **Linkage-editing of programs for execution.**
  The linkage editor, another of the components of DOS/VSE, processes language translator output to make this output executable. A language translator output processed by the linkage editor becomes one or more phases in a core image library of DOS/VSE, ready for being loaded from this library and executed.

- **Library services.**
  DOS/VSE includes a number of programs collectively referred to as librarian. The librarian provides library services such as cataloging programs and procedures (sets of control statements), deleting them (if necessary), and updating them in system and private libraries as required; condensing libraries is another service function provided by the librarian.

- **Data management.**
  The input/output and file organization routines included in DOS/VSE relieve the programmer of the detailed and cumbersome programming associated with the transfer of data between auxiliary storage and programs.

- **System-to-operator communication.**
  Devices alone do not perform any data processing. To get jobs done, a human being — the operator — must initiate system operation. DOS/VSE provides status information, some automatically, and other on operator demand. This allows the operator to monitor system operation and to respond quickly to situations that require human intervention.

- **System utilities.**
  DOS/VSE includes a number of programs that help to improve the overall efficiency of a DOS/VSE controlled data processing installation: magnetic tape or disk volumes are to be initialized; data is to be copied for backup purposes and may have to be restored. These programs are collectively referred to as system utilities.

- **System serviceability and debugging aids.**
  Reliability, availability, and serviceability of an installation are important considerations in planning for the operation of a DOS/VSE controlled installation. Program debugging needs to be done whenever a new application is being implemented. DOS/VSE includes programs and programming tools that are designed to contribute in these areas of computer operation.

- **Assembling programs.**
  DOS/VSE includes an assembler. This program efficiently translates assembler language programs into machine-readable (object) language. The assembler includes a powerful macro processor.

Each of those major computing services available through DOS/VSE is discussed in more detail in the next chapter of the manual.

Chapter 2: Available Computing Services — Discussion

This chapter expands on the information provided in the preceding chapter about the major computing services available with the DOS/VSE SCP and VSE/Advanced Functions; it mentions significant functional extensions that are available only if VSE/Advanced Functions is installed. The services discussed in this chapter are:

- Initial program load
- Resource management
- Job control
- Library services
- Linkage editing of programs
- Data management
- System-to-operator communication
- System utilities
- System serviceability and debugging aids
- Assembling programs.

Initial Program Load

To initiate system operation, the operator — after having completed initial microcode load — performs the DOS/VSE initial program load (IPL) procedure. This procedure can be automated almost 100 percent if you make use of the automated system initialization facility. This facility, which is available if VSE/Advanced Functions has been installed, allows the commands and control statements needed for the complete operating system start-up to be read from the procedure library. IPL from a card reader, a diskette I/O unit, or the console is also supported, of course.

During IPL, DOS/VSE automatically calculates and reserves the amount of virtual storage required for operating system use. DOS/VSE loads certain code into reserved storage which is located at the high end of virtual storage and referred to as shared virtual area.

Available options allow you to extend the shared virtual area and to load code of your own into this area. For more information about available IPL options and the use of the shared virtual area, see *DOS/VSE System Management Guide;* for information about the automated system initialization facility, refer to VSE/Advanced *Functions General Information.*

Resource Management and Job Control

The supervisor and the job control program frequently interact to make an installation's resources available as they are needed for problem program execution. Resources in this context are, for example, processing time and disk storage or I/O devices.

You specify the required resources either through job control statements immediately before a program is being executed or through macros (or available high-level-langauge statements) in the program itself. Based on your specifications, the supervisor allocates the requested resources dynamically as they become available and in accordance with processing priorities defined for your DOS/VSE.

This section discusses how the processing of jobs can be controlled and how the resources of an installation may be efficiently utilized.

Controlling Jobs

After the system has been successfully started by initial program load, it is ready to accept input for processing.

job and job stream

The unit of work the user submits to the system for processing is called a job. A succession of jobs presented to a computer is a job stream. Within each job, one or more programs may be executed. These programs may be IBM programs, such as a compiler that translates a user's source program into object code, or a user program that is already in executable format and processes data.

job control statements

A job, and the environment in which it is to run, must be defined to DOS/VSE by means of job control statements. Job control statements specify, for example, whether user programs are to be compiled, link-edited, and/or executed; from which library or device the system or user programs are to be loaded; what files they are to process and where these files reside. When handling jobs, DOS/VSE performs the following basic functions:

- It provides for automatic transition from job to job with a minimum of operator intervention.
- On the basis of job control statements supplied by the user, it assigns actual I/O devices to the logical device names specified in programs.
- It loads executable programs from online disk libraries into storage for execution.
- It handles program termination.

Automatic Job-to-Job Transition

Jobs are defined to DOS/VSE by a // JOB statement at the beginning and a / & statement at the end as illustrated in Figure 2-1.

job step

You may define jobs with only one job step or with two or more. Each job step of a multi-step job consists of one program (or phase), which is executed after the preceding job step is completed (see also Figure 2-1).

Defining a series of related programs as a single job may sometimes be required or it may provide specific advantages. For instance, multiple job steps within a single job are preferable if execution of a later job step depends on successful completion of an earlier one. If a job step of a

multi-step job ends abnormally, DOS/VSE bypasses the remaining steps and starts executing the next job.

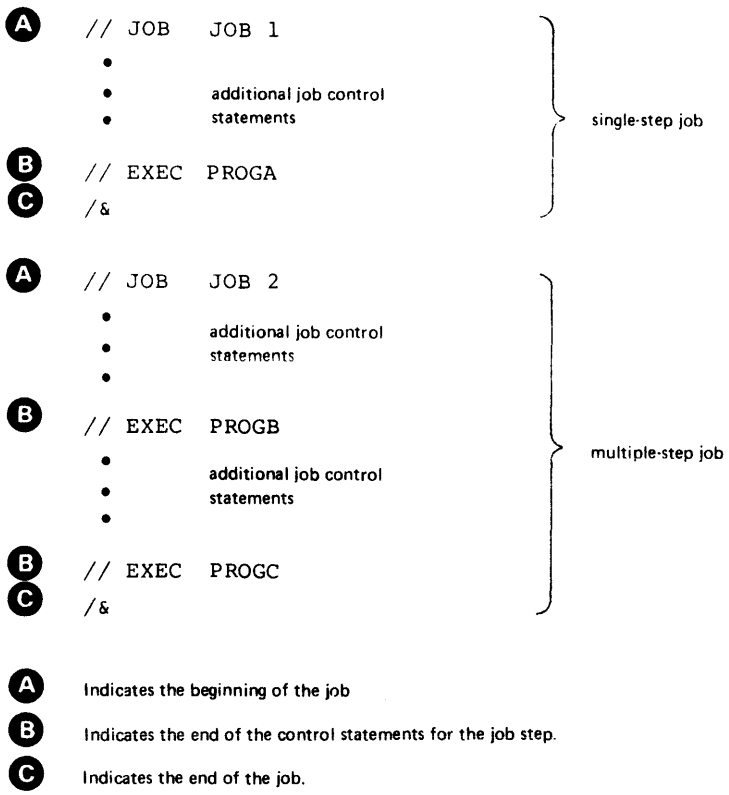


Figure 2-1. Jobs and job steps under DOS/VSE

Whenever a job or job step has been completed, the job control program is automatically reloaded. Job control reads and processes job control statements from the device assigned to the system input stream until it finds a // EXEC statement, which requests execution of a program. It then passes control to the supervisor, which locates the requested program in the core image library (see **Note** below) and loads this program into storage for execution. The program then gains control to start processing. In this way, the transition from one job to the next one in the job stream is handled by DOS/VSE without intervention by the operator.

Note: Executable programs (phases), IBM-supplied or user-written, are stored in a core image library.

Assigning Logical Unit Names to Actual I/O Devices

Programs that need access to data usually must inform DOS/VSE of the type of device involved. The actual device need not be specified in the program, only a name referring to a logical rather than physical unit must be specified.

The assignment of a logical unit name to a physical device (channel and unit number) is made either immediately after IPL or between jobs or job

steps. How to make this assignment operation is discussed in proper context later in this chapter.

You can request assignments to be permanent or temporary.

permanent assignment

A request for a permanent assignment causes a specific logical unit name to be assigned to a certain physical device for the duration of the current system run (that is, until next IPL), unless the assignment is changed again by another permanent or a termporary assignment request.

temporary assignment

A request for a temporary assignment causes a specific logical unit name to be assigned to a certain physical device only for the duration of one job or, if another temporary respecification is made, only for the duration of one or a certain number of job steps of that job. At the end of the job, job control resets a temporary assignment to the permanent assignment made previously during the same system run.

You can specify permanent or temporary assignments in two ways. You can include them in the job stream or you can enter them directly from the console keyboard that is used to communicate with the system.

Permanent and temporary device assignments offer the following advantages:

- If the configuration of an installation is changed by the addition or removal of a particular device, the installation's programs need not be altered as long as another unit of the same device type is available.
- If, before the execution of a program, a device appears to be inoperative or in use by another program, the operator can select another unit of the same device type, enter a new assignment to reflect the new physical address, and have DOS/VSE execute the job or job step.

Figure 2-2 shows how a device assignment is changed by a permanent assignment via the operator's console. Figure 2-3 gives a full list of the logical unit names recognized by DOS/VSE. The use of the logical units will be shown in proper context later in this chapter.

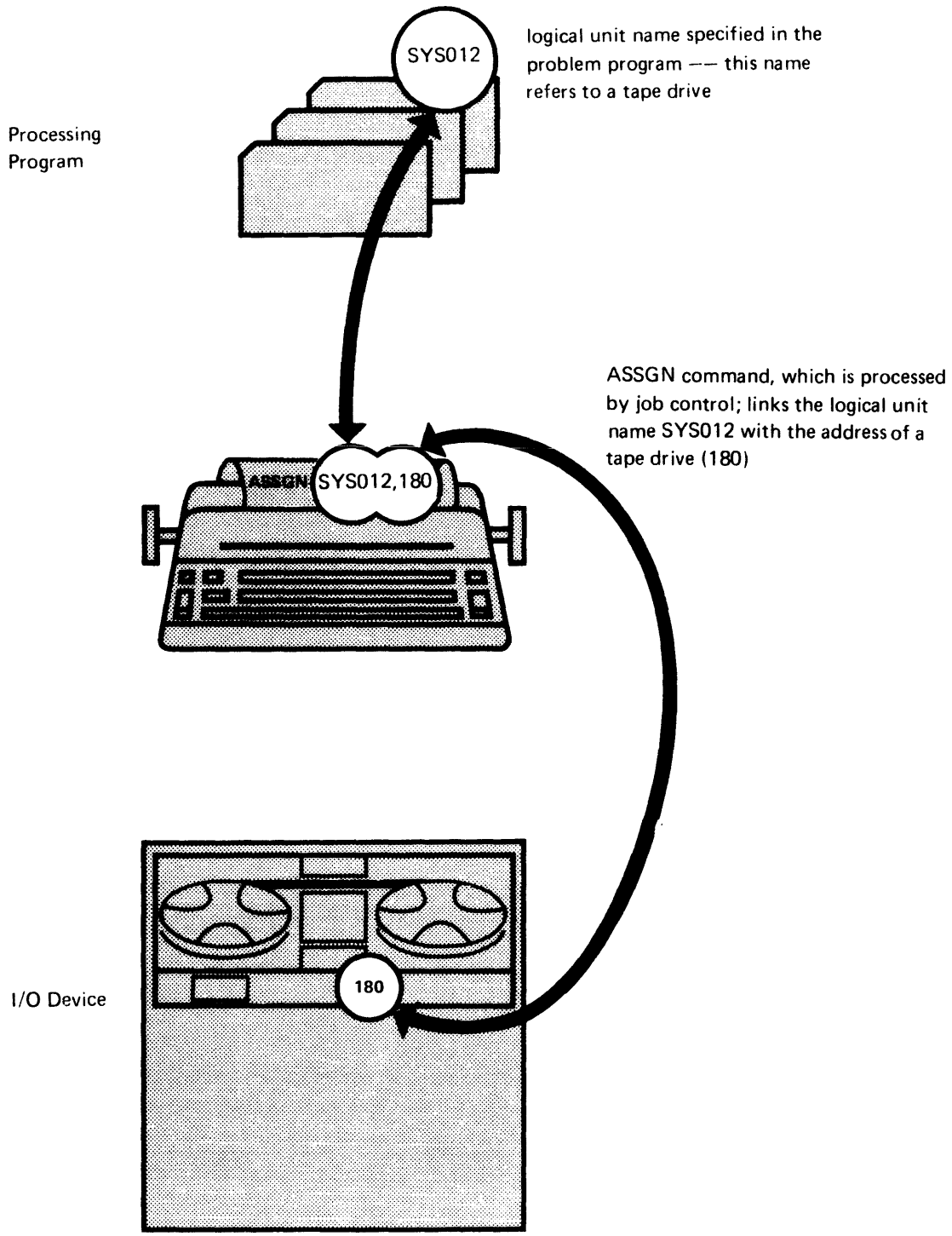


Figure 2-2. Assigning an actual I/O device to a logical unit by the operator

Logical unit	I/O device type	Used for
System logical units:		
SYSRDR	card reader, magnetic tape unit, disk or diskette extent	reading job control statements.
SYSIPT	card reader, magnetic tape unit, disk or diskette extent	input of system data such as source statements for language translators or control information for service programs.
SYSIN	card reader, magnetic tape unit, disk extent, or diskette extent	**can** be used when SYSRDR and SYSIPT are assigned to the same card reader or magnetic tape unit; **must** be used when SYSRDR and SYSIPT are assigned to the same disk extent or diskette extent.
SYSPCH	card punch, magnetic tape unit, disk extent, or diskette extent	punched output of the system.
SYSLST	printer, magnetic tape unit, disk extent, or diskette extent	printed output of the system.
SYSOUT	magnetic tape unit	**must** be used when SYSPCH and SYSLST are assigned to the same magnetic tape unit; **cannot** be used to assign SYSPCH and SYSLST to disk since these two units must refer to separate disk extents.
SYSLOG	console printer keyboard, display operator console, or printer.	communication between the system and the operator and for logging job control statements.
SYSRES	disk extent	system residence device
SYSREC	disk extent	logging error records and accommodating the (console) hard copy file and the system history file.
SYSDMP	disk extent	one or two dump files
SYSCLB	disk extent	private core image library.
SYSRLB	disk extent	private relocatable library.
SYSSLB	disk extent	private source statement library.
SYSLNK	disk extent	input to the linkage editor.
SYSCAT	disk extent	VSAM master catalog (if VSE/VSAM is used).
Programmer logical units:		
SYS000 SYS001 ⋮ SYS240	any device of the system	user program input/output.

Note: System logical units are used primarily by the components of DOS/VSE. Programmer logical units (SYS000 through SYS240) are primarily used by problem programs. Problem programs can reference up to 241 different programmer logical units per partition; in addition, these programs may reference the following system logical units: SYSRDR, SYSIPT, SYSPCH, SYSLST, and SYSLOG. For a discussion of partitions, see *Storage Organization* under *Resource Utilitzation* later in this chapter.

Figure 2-3. Logical unit names recognized by DOS/VSE

Loading Programs for Execution

All executable object programs, both IBM-supplied and user-written, must be stored in a core image library. Programs that are used frequently are stored permanently; those not often used may be stored only temporarily, just prior to loading for execution. Whether a program is to be stored permanently or temporarily can be specified in a job control statement.

The storing (or cataloging) is done by the linkage editor, which processes the output from language translators and places its output, one or more executable program phases, into a core image library.

The request for loading a phase from the core image library for execution may come from a DOS/VSE component (for example, in the case of error recovery) or it may come from the job control program after that program has read the control statement identifying the next phase to be loaded. A request may also come from any other program.

Phases that can be shared between several programs and also are used frequently may be loaded into the shared virtual area (SVA). More information on the SVA and the requirements of phases that can be contained in it can be found under *Virtual Storage Support* later in this chapter.

Figures 2-4 and 2-5 illustrate two specific sequences of job control statements and explain the sequence of operations that are caused by these statements.

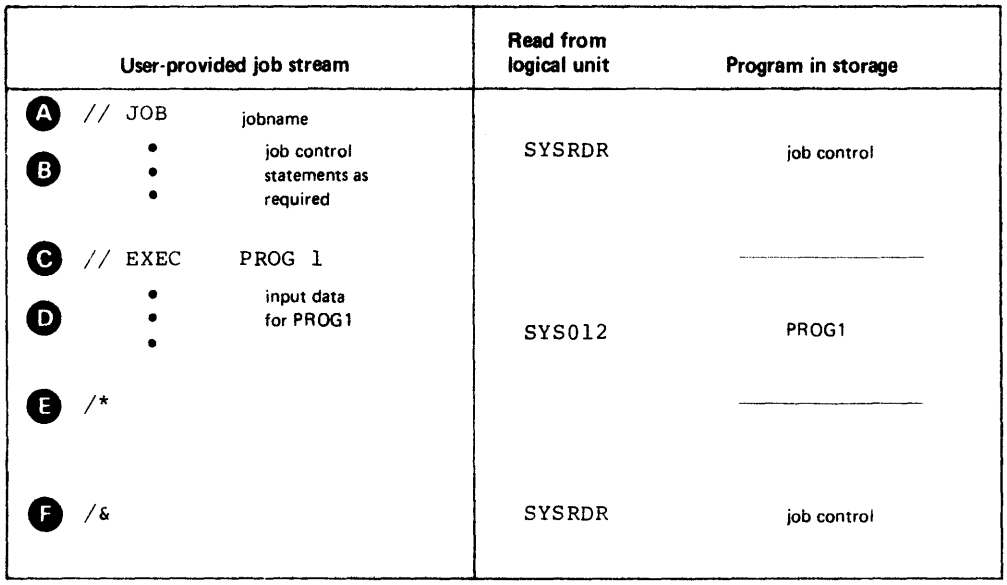
The first example (Figure 2-4) is the execution of a program that has already been cataloged in the core image library.

The second example (Figure 2-5) is more complex. It illustrates how a source program would be compiled and executed in a single job. The job consists of three job steps: one for the assembly; the second one for the link-editing of the assembled object module and cataloging the resulting phase in a core image library; the third one for the execution of the phase. This sort of job control sequence is typical for a program development and test environment. Test data is submitted during execution, but the program is placed in a core image library only temporarily. After all debugging has been completed and after successful runs with test data, the program would probably be cataloged permanently in a core image library.

Figure 2-5 more so than Figure 2-4 illustrates the role of the job control program in processing jobs under DOS/VSE

Handling Program Termination

The job control program handles normal program termination to ensure proper job-to-job transition. It also handles situations that would require abnormal program termination. Reasons for abnormal program termination are, for example:



A This is the first of a set of control statements for a job. The statement indicates the beginning of a job, gives this job a name, and (optionally) provides job accounting information.

B If no job control statements are provided, DOS/VSE continues processing using the values set immediately after IPL.

C When read by job control, this statement causes DOS/VSE to locate phase PROG1 in a core image library of the system and to load this phase into storage for execution.

D It is assumed that PROG1 has been coded to read data (in card image format) from SYS012. This logical unit may be assigned to the same physical unit to which SYSRDR, the input unit for job control statements, is assigned or to another physical unit.

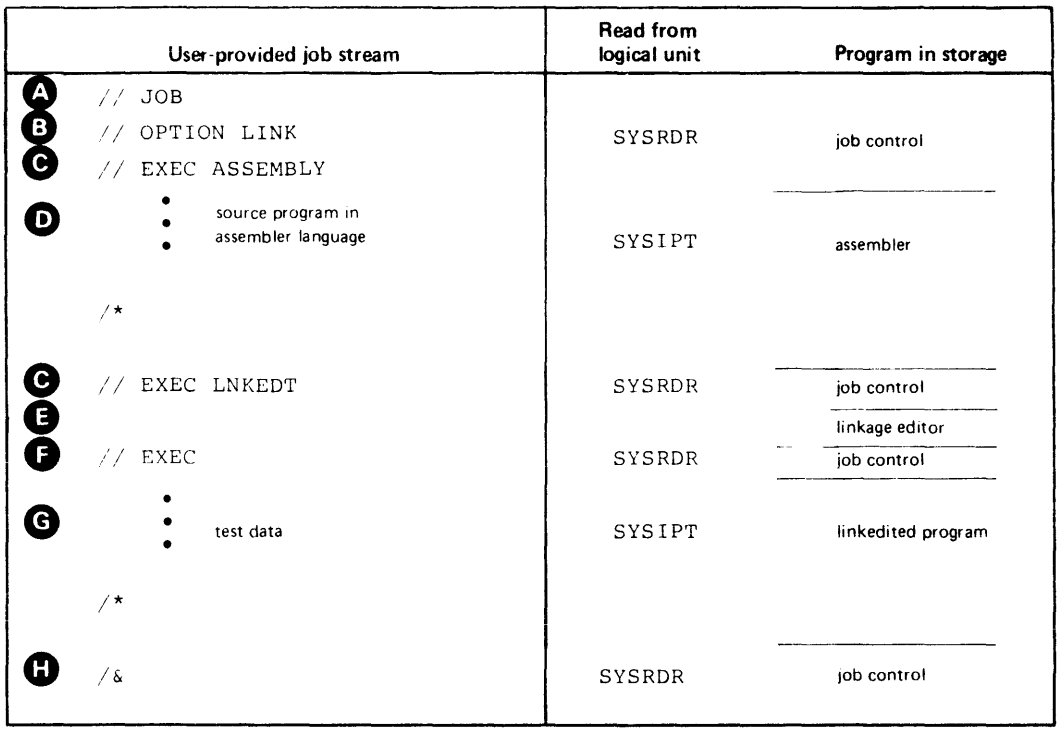
E The statement indicates to DOS/VSE the end of data on SYS012. When PROG1 finishes executing, DOS/VSE loads the job control program again.

F The statement indicates the end of the job. When the job control program reads the statement, that program performs end-of-job processing and proceeds to the next job in the job stream.

Figure 2-4. Loading and executing a phase (or program)

abnormal program termination

- The operator requests program termination from the console keyboard. He would request cancellation, for example, if he suspected that the program had entered an unending program loop.
- A program failure or illegal program action. An example might be a program's attempt to address a non-existent or non-assigned I/O device.
- An irrecoverable hardware failure. If an exceptional hardware condition occurs, the system tries to recover. Frequently, recovery is successful; but when it is not, an abnormal program termination occurs. An example would be a persistent read or write error that could not be resolved by the error recovery procedures of DOS/VSE.



A Indicates the beginning of the job.

B Indicates that any language translator output is to be link-edited and cataloged temporarily into a core image library.

C Causes DOS/VSE to locate the named phase in a core image library and to load this phase into storage for execution. As DOS/VSE loads the phase, the job control program is being overwritten.

D The assembler reads and processes this source input up to the /* statement. Normally, the logical unit SYSIPT is assigned to the same physical device as SYSRDR, from which the job control program reads the job control statements. The assembler produces an object module and writes this module on SYSLNK, a system area on disk, for subsequent processing by the linkage editor. This action is triggered by the OPTION statement (see **B** above).

When the assembly is complete, DOS/VSE loads the job control program again.

E The linkage editor retrieves the object module from SYSLNK and places its output, an executable phase, temporarily into a core image library.

When the linkage editor has finished executing, DOS/VSE again loads the job control program into storage.

F The // EXEC statement without a program name specified indicates to DOS/VSE that the executable phase temporarily placed into a core image library by the linkage editor is to be loaded into storage for execution.

G In this example, the program assembled and link-edited reads data from SYSIPT (up to and including the /* statement) and processes the data. When the phase has finished executing, DOS/VSE once more loads the job control program into storage.

H Indicates end-of-job. The job control program performs end-of-job processing and proceeds to the next job in the job stream.

Figure 2-5 Assembling, link-editing, and executing a program.

In case of an abnormal program termination, the following series of events occurs:

- If the error recovery procedures were involved, DOS/VSE records hardware and environmental data on a system recorder file (see *System Serviceability and Debugging Aids)* later in this chapter.
- DOS/VSE issues a message to the operator, indicating the cause of the failure (a malfunction of the console keyboard or display could prevent this).
- DOS/VSE may produce a storage dump (a hexadecimal printout of the contents of storage). A dump can be specified or suppressed by the programmer at any point in the job stream; a dump may also be requested by the operator. Dumps are one type of available debugging aids (for more information about debugging aids, see section *System Serviceability and Debugging Aids* later in this chapter.
- DOS/VSE bypasses the remaining data and control statements for the job in the input stream.
- DOS/VSE starts the next job in the input stream (it is possible that system operation is so extensively impaired that the IPL procedure has to be repeated).

user exit routines

If you are programming in the assembler language, you have the option of including user-written exit routines in the program. Then, if an abnormal program termination occurs, DOS/VSE passes control to the appropriate exit routine, giving information on the cause of the abnormal termination. The exit routine can examine this data and take appropriate action.

Job Accounting

You can write a simple program to keep track of processor usage and the usage of the various I/O devices by accessing the job accounting data accumulated by the system. This enables you to:

- Charge usage of the system to the various users.
- Check on efficient use of I/O devices.
- Maintain a record of system operation and use.
- Plan for new applications, additional devices, and new systems.

The necessary information (input for your program) is provided by the job accounting interface, an optional facility that you can specify for system generation. With this facility, the following information is automatically gathered by DOS/VSE for each job step and stored in a table:

- Job name, date, and the partition in which the job is running.
- Start and stop times of the job, processor time used by the problem program, processor time used by DOS/VSE (overhead), and processor idle time chargeable to the partition.

- Optionally, counts of I/O operations of the various devices used by the job.

At the end of each job step, your accounting program is automatically loaded into the partition where the job step has just finished processing, either to transfer the information from the job accounting table to auxiliary storage (tape or disk) for future use, or to format and print the information. For more details on this subject refer to *DOS/VSE System Management Guide.*

Resource Utilization

The main measure of a DOS/VSE installation's efficiency is throughput, that is, the amount of work handled in a certain period of time. The major resources to be examined when discussing an installation throughput are (1) processor time and its use, (2) virtual storage space and its exploitation by problem programs, and (3) disk library space and its employment.

DOS/VSE provides facilities that improve system throughput by allowing efficient utilization of system resources:

- Efficient use of processor time through multiprogramming or multitasking. Multiprogramming allows concurrent execution of two or more programs; multitasking, a facility available to programmers using assembler language, allows DOS/VSE to continue processing of one task of a program (or job step) if, for any reason, processing of another task of that program has been suspended.
- Efficient use of the available storage through multiprogramming and virtual storage support.
- Efficient use of disk library space through the DOS/VSE librarian programs and the relocating loader facility. This is discussed as a separate main topic.

Additional DOS/VSE components and facilities designed to improve system throughput are available through licensed programming support. VSE/POWER, for example, is designed for efficient use of processor time and unit record I/O devices. VSE/Advanced Functions, a licensed programming package, helps to improve throughput through more efficient multiprogramming. For more information about additional components and facilities, turn to *Chapter 6: Licensed and Nonlicensed Programming Support.*

The job accounting facility of DOS/VSE is available for monitoring usage of the central processor and of I/O devices. This may help you to balance your mix of concurrently running jobs to achieve better total system utilization.

In order to examine these features in more detail, it is first necessary to look briefly at the storage organization under DOS/VSE.

Storage Organization

DOS/VSE, an operating system designed for multiprogramming, can be generated to organize the available storage for optimal operating conditions at the particular installation. DOS/VSE always reserves a certain amount of the available storage for its own use leaving the remainder to problem-program use.

partitions

The storage available for the execution of problem programs is divided into areas called partitions. You can generate your DOS/VSE to support simultaneous execution of problem programs — multiprogramming — in two to five partitions in an SCP only environment and two to seven if VSE/Advanced Functions is installed. Normally, one problem program is contained and executed in each of these partitions; therefore, if you generate your DOS/VSE to support multiprogramming in five partitions, you can load up to five problem programs for concurrent execution.

sizes of partitions

At the time of system generation, you define the number of partitions to be supported by your DOS/VSE. Partition sizes, however, you define immediately after IPL or between jobs, whichever is appropriate.

You may define a partition to the minimum size, if this meets your operational requirements, or to a larger size. You may also set this size to zero if you do not need the particular partition for problem program execution and want to make more virtual storage available to one or more of your system's other partitions.

single-partition operation

In fact, by setting to zero all partitions except one, you can operate your DOS/VSE as a single-partition system although it includes multiprogramming support.

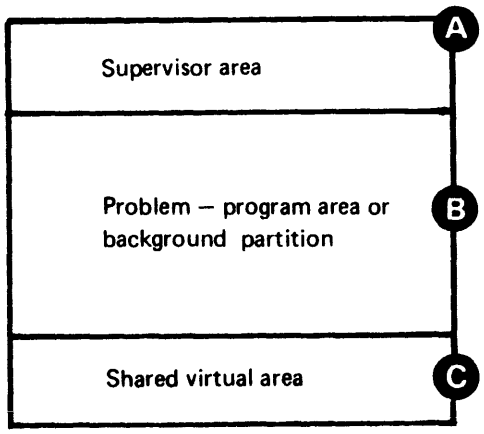
Figure 2-6 and 2-7 show the storage organization of a DOS/VSE set for single-partition and 4-partition operation, respectively. Figure 2-7 illustrates, in addition, the IBM set processing priorities for the various partitions. Processing priorities are discussed further under *Multiprogramming* below.

Multiprogramming

Multiprogramming offers one significant advantage over single-partition operation: better utilization of processor time. This is demonstrated best by comparing processor usage in the two operating environments.

processor usage in single-partition operation

In single-partition operation, only one problem program can be in storage at a time. When, for example, the program needs input or output, it issues an I/O request to the system supervisor. The supervisor passes this request to a channel, which then executes the I/O operation. During most of this time interval, the central processor remains idle, or in the wait state. (See Figure 2-8.)



A Accomodates the resident routines of the DOS/VSE supervisor, which remain in real (processor) sotrage throughout system operation under DOS/VSE.

B Is available for the execution of problem programs, one at a time. These programs may be IBM–supplied or user–written.

C Is discussed later in this chapter.

Figure 2-6 Storage organization for single-partition operation.

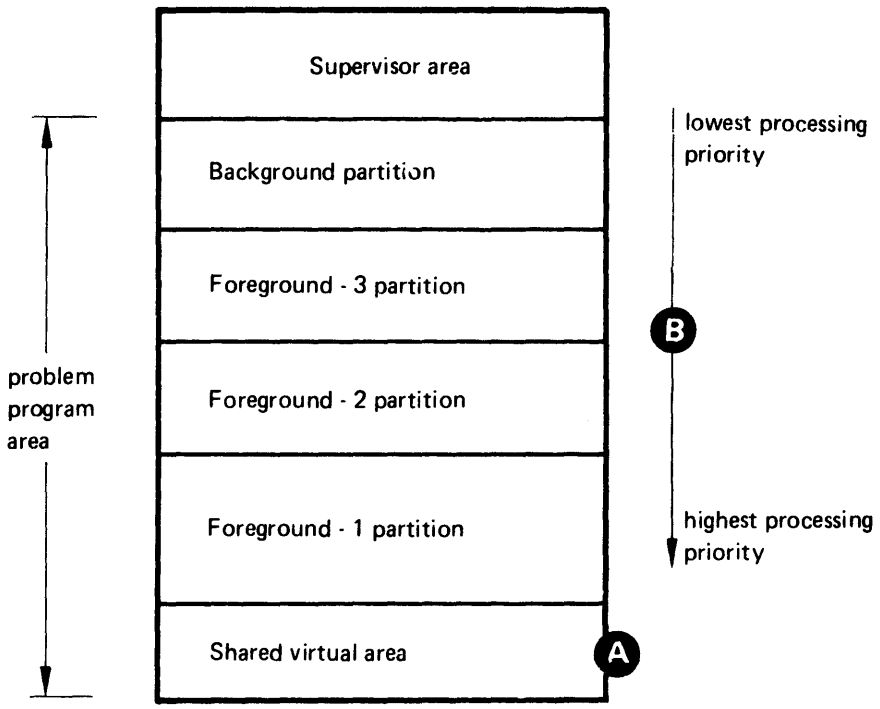
processor usage in a multiprogramming environment

Figure 2-9 illustrates the passing of control among programs in a multiprogramming environment. Note how much less processor time is left unused or idle when compared to processor usage in a single-partition system (refer to Figure 2-8).

processing priorities

With programs taking turns executing in a multiprogramming environment, processing must obviously proceed according to a set of priorities. The priority of a program for receiving processor resources is dependent upon the processing priority of the partition in which the program resides. The supervisor, of course, always has the highest priority.

The default priorities for the partitions are as shown in Figure 2-7. However, DOS/VSE allows you to change these default values during system generation or later by means of an operator command. By assigning a job to a certain partition, a programmer or an operator, therefore, assigns the priority of that partition to the job as well.



A Is discussed later in this chapter.

B The priorities per default; they may be changed by the operator.

Figure 2-7 Storage organization of and processing priorities for partitions in a 4-partition DOS/VSE

Multitasking

Multitasking, a special form of multiprogramming, allows concurrent execution of two or more sections of a program, called **task**, within one partition. The purpose of this facility again is to make more efficient use of processor time by providing a means of having various parts of one program executed concurrently.

main and subtasks

With multitasking, one main program, the main task, **attaches** one or more subprograms, or subtasks. The main task gets control from the supervisor following an EXEC statement and then initiates, or attaches, the subtasks. The main task and its attached subtasks always reside in the same partition.

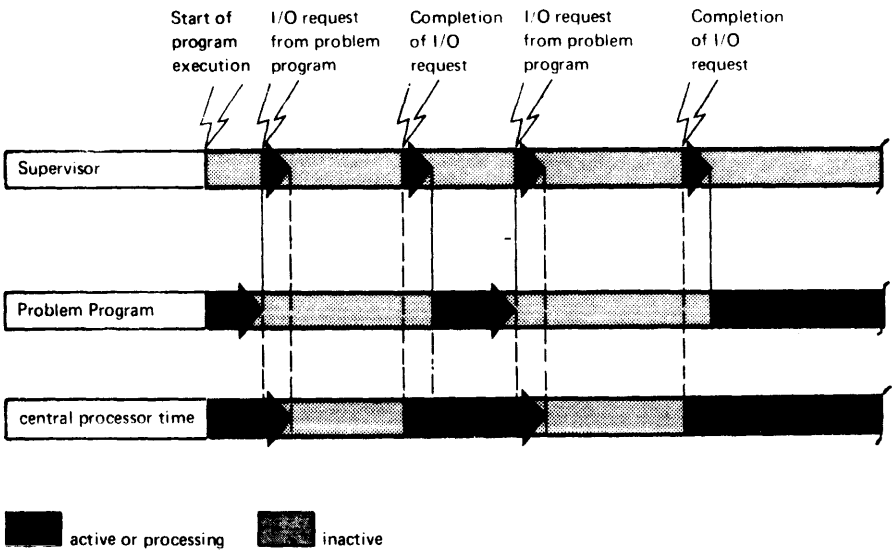


Figure 2-8. Processor usage in a single-partition operation.

Usually, the main task and its subtask(s) are parts of one program. It is, however, also possible to attach completely different programs to a common main task for execution in the same partition.

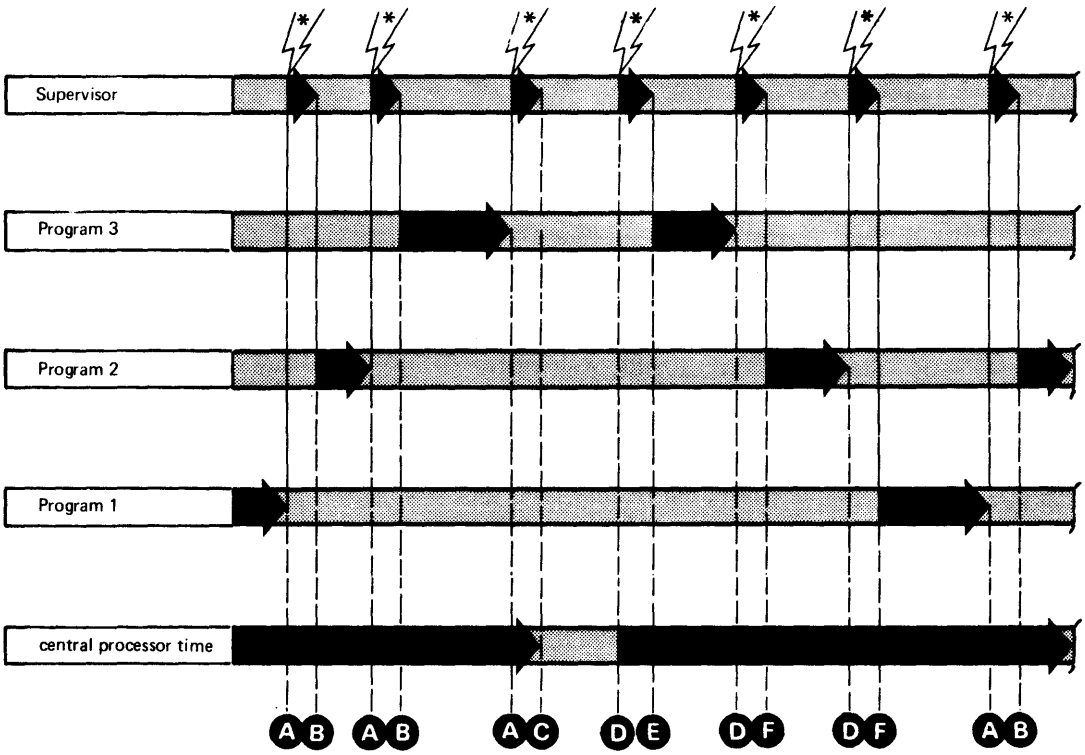
The total number of tasks in a system depends on the number of partitions. The sum of all subtasks attached at any one time and all partitions must not exceed 15. A 4-partition DOS/VSE, for example, allows you to have a maximum of eleven subtasks attached at any point in time. All of these subtasks can be attached to one main task, or they can be divided among any number of main tasks.

Subtasks have higher processing priority than their associated main tasks. The priority of a subtask in the partition is normally determined by the sequence in which it is attached by the main task. You may, however, request DOS/VSE from within your program to change the processing priority of a subtask.

That part of the main task which attaches or detaches subtasks must be written in DOS/VSE assembler language. The rest of the main task and the subtasks may be written in any high-level language, provided restrictions as documented in the pertinent language translator documentation are observed.

Virtual Storage Support

Users of DOS, an operating system that was designed primarily for operation on System/360, are restricted to an address space limited by the storage physically contained in the installation's central processor. The total space required by the supervisor and assigned partition(s) cannot exceed this constraint.



Note: *This example is based on a multiprogramming system with three active partitions. Program 1 has the highest priority, program 3 the lowest.*

(A) Programs 1, 2, 3, and 1 in this order issue an I/O request and enter the wait state pending completion of that request. The supervisor takes over to start the I/O operation, and to determine which other program can start processing.

(B) Programs 2,3, and 2, in that order, receive control because they are waiting ready to run.

(C) The supervisor is unable to find a program waiting with no I/O requests pending. The system enters the wait state, waiting for an I/O operation to finish.

(D) The pertinent channel signals the completion, in turn, of the I/O operations for programs 3, 2, and 1. The supervisor dertermines which is the highest priority program ready to resume processing.

(E) Program 3 regains control, because programs 1 and 2 are still waiting.

(F) Programs 2 and 1 resume processing, because they are higher priority programs waiting with no pending I/O requests.

In the example, the switch from one partition to another is always made upon an I/O request or I/O interrupt.

Figure 2-9. Processor usage in a multiprogramming DOS/VSE

Under DOS, programs are confined to the processor storage partition limits. Also, because the size of a partition depends on the size of the largest program to be executed in it, running any smaller program in such a partition will result in some storage space being unused. DOS/VSE, which has been designed for operation on processors that support the virtual storage concept, removes these limitations of DOS without any extra programming effort on your part.

Program Execution in a Virtual Storage Environment

The instructions of a program, and also related data, must be in processor storage in order to be executed. However, only one instruction of one program can be executed by the central processor at any point in time; therefore, all other instructions of the same or any other program need not be in processor storage at the same point in time. The design and implementation of the virtual storage concept makes use of this operating characteristic of a central processor.

program pages

page data set

Programs to be executed in a virtual storage environment are sectioned by DOS/VSE into blocks of 2K bytes called program pages. If not all of the pages of a program fit into processor storage, DOS/VSE writes pages currently not needed for program execution to a disk file called page data set and continues executing instructions contained in program pages that are still in processor storage. Figure 2-10 shows the relationship of the page data set and processor storage; the illustration may help you to understand the subsequent discussion.

page frame
page-in

If, during program execution, an instruction refers to a program page which is not in processor storage or control is transferred to such a page, DOS/VSE automatically suspends execution of the program, retrieves that page from the page data set and places it into a 2K-byte section (page frame) of processor storage. Once this operation (which is referred to as page-in) is complete, execution of the interrupted program continues.

page-out

Chances are that no page frame is available in processor storage to accommodate a program page needed for continued execution of a program. In this case, DOS/VSE automatically frees a page frame. DOS/VSE selects a program page not recently referenced and writes it to the page data set if necessary. This operation — also called page-out — is necessary if the page was altered while in processor storage or if it had not been written to the page data set previously.

page pool

Page frames that are not used by the supervisor nor are needed for the execution of programs in real mode — this is discussed below — comprise the system's page pool. Figure 2-11 illustrates the execution of four programs in a virtual storage environment.

execution in real mode

Some programs cannot tolerate page-in or page-out operations. For these programs, execution in real mode is indicated. Examples of programs that should run in real mode are time or performance critical applications (for example, programs that include MICR stacker-select routines or real-time operations).

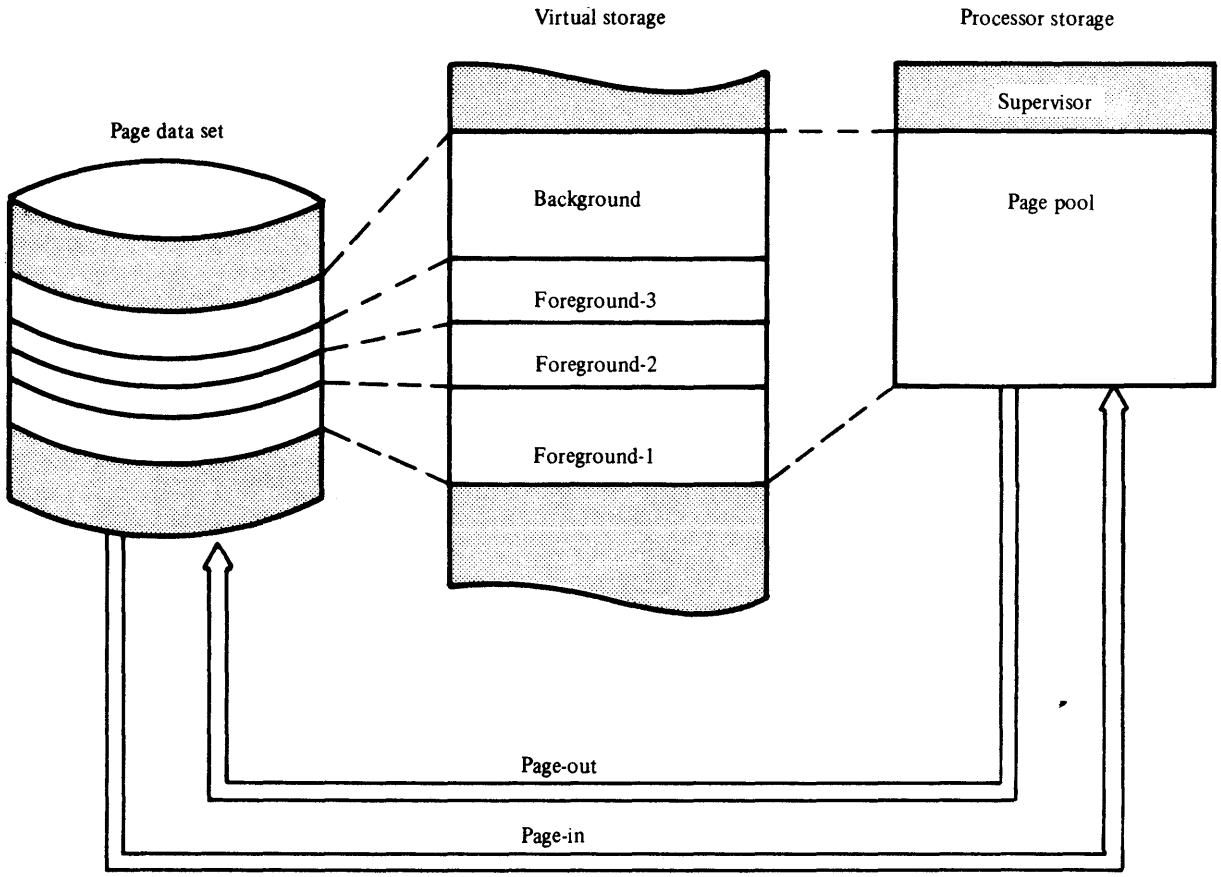


Figure 2-10. Relationship of page data set, virtual storage, and processor storage.

For the execution of a program in real mode, you must allocate a sufficiently large area of processor storage to the partitions in which the program is to run. The page frames used by that program are taken away from the page pool; they are therefore no longer available for the execution of programs which run in other partitions in virtual (not real) mode.

Allocating Storage to Partitions

Through a combination of hardware design and programming support, DOS/VSE has an address space, called virtual storage. This address space, an extent on disk, starts at zero and can extend to the maximum allowed by the system's 24-bit addressing scheme. This provides for up to 16,777,216 (or 16,384K) bytes of address space. How much of this address space will be used in a particular system depends upon a number of factors: the size of the installation's central processor's storage; the number of partitions and their size; the size of the shared virtual area (SVA) — an area available for the (virtual) execution of reentrant code that may be accessed from any partition any time; the characteristics of the installation's programs and operating environment.

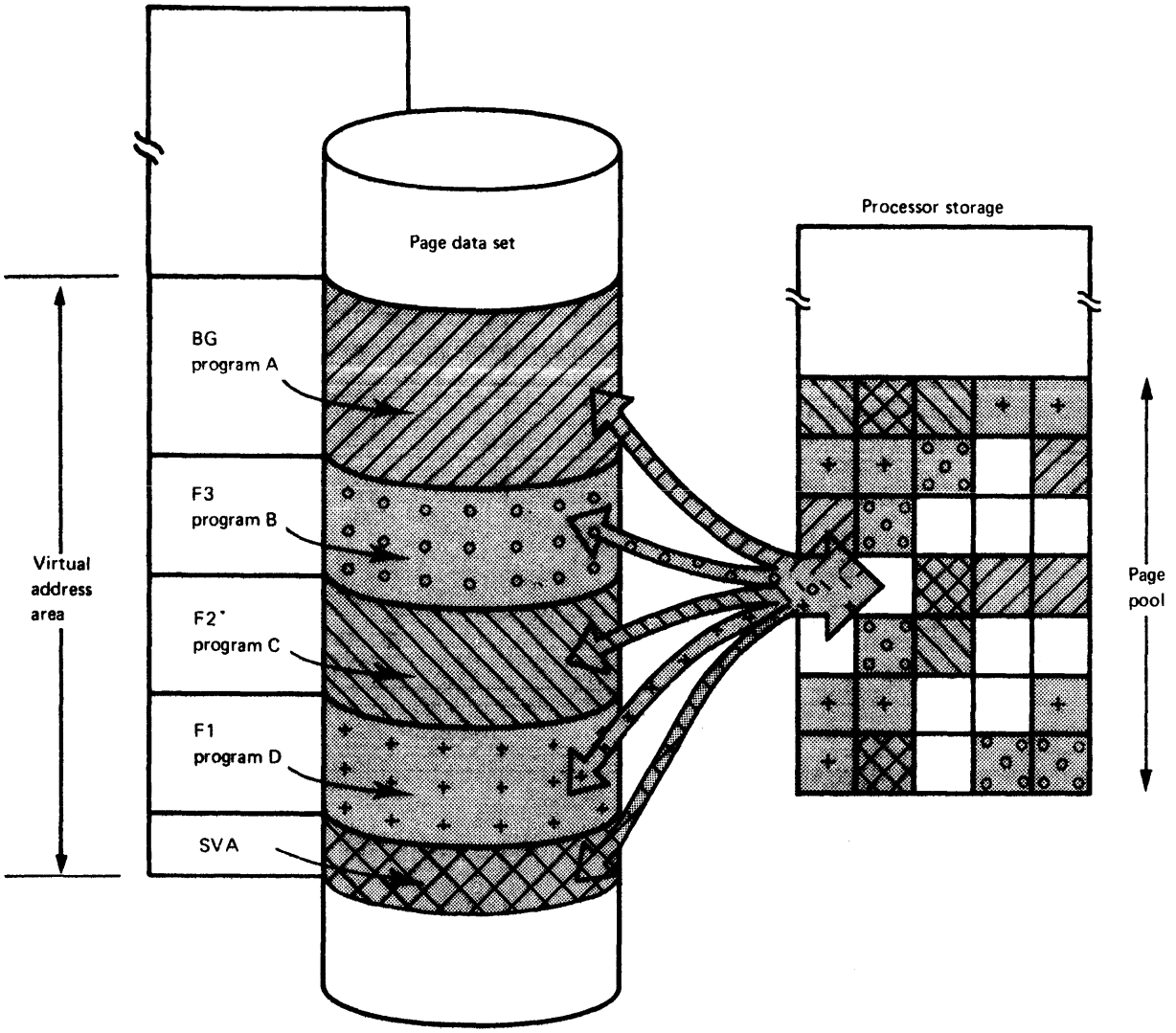


Figure 2-11. Four programs executing in a virtual environment

Based on these factors, trade-offs have to be made to arrive at an optimal virtual storage size for the requirements of a particular installation. A tailored DOS/VSE is then generated to this size, which will contain a virtual storage normally smaller than the maximum limit and larger than the processor storage installed in the system.

real address area

Assume a virtual storage system of 1,280K bytes with 512K bytes of processor storage. That part of the installation's virtual storage which can be directly equated to processor (real) storage is called the real address area if your DOS/VSE operates in System/370 mode. In this operating environment, that area is not available as address space for problem programs which are being executed in virtual mode; in other words, DOS/VSE deals with addresses on two different levels, in the real address area and the virtual address area. This relationship is shown in Figure 2-12.

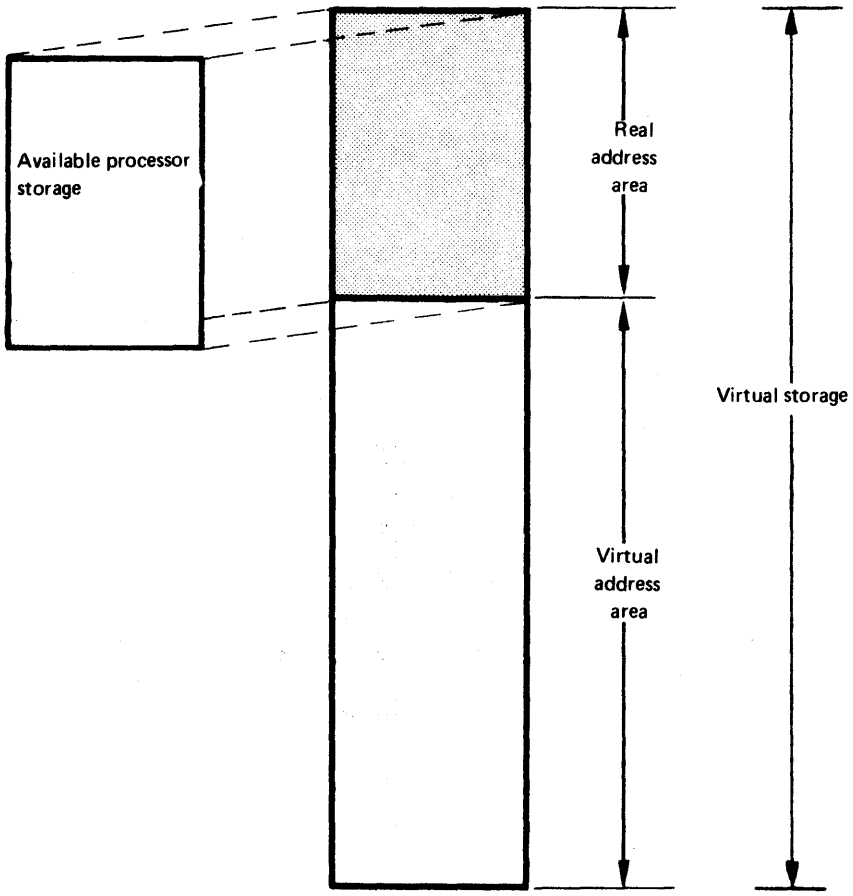


Figure 2-12. Virtual storage under DOS/VSE in System/370 mode

virtual address area

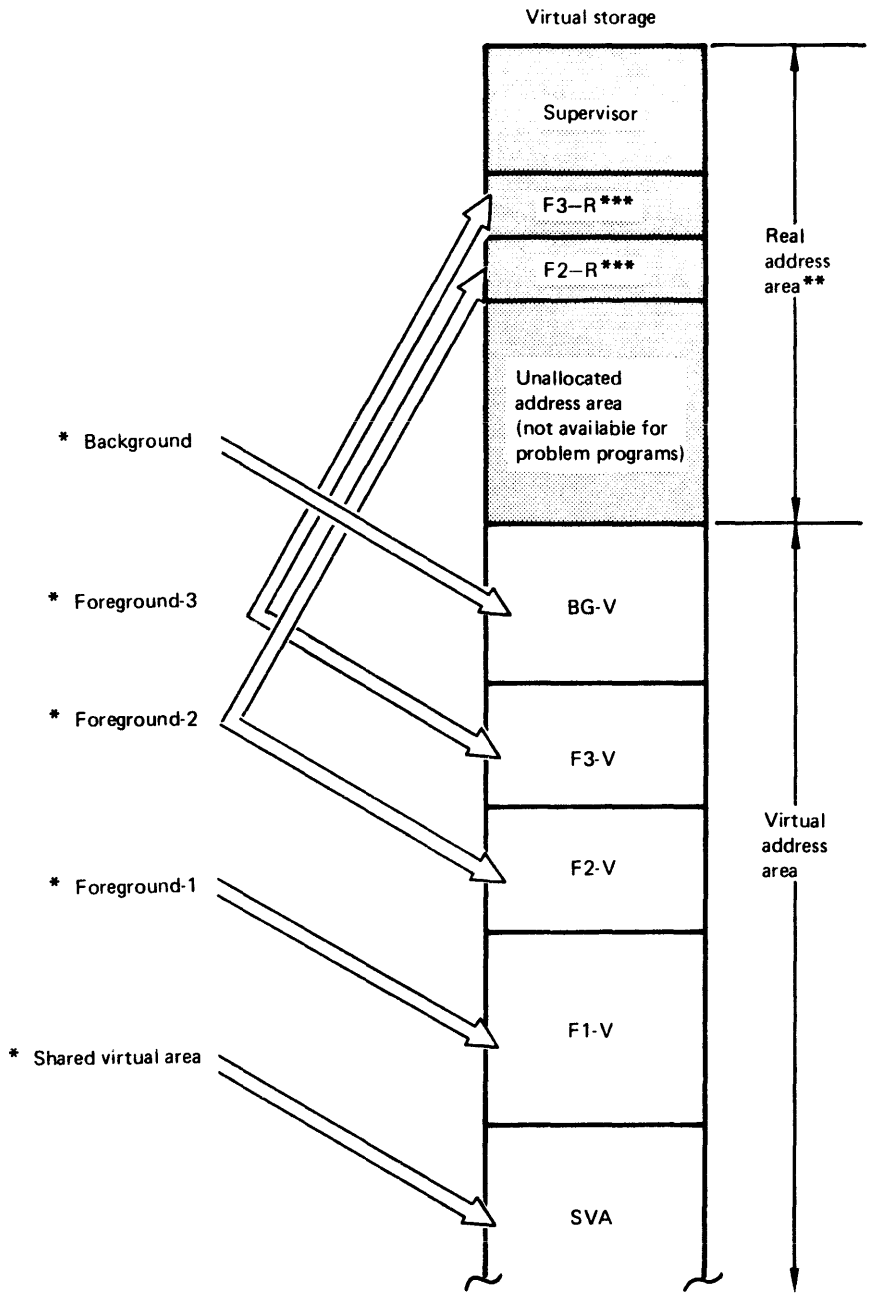
In System/370 mode, virtual storage beyond the end of the real address area and up to the limit of the system's generated virtual storage, is the system's virtual address area.

On a processor with **extended control program support: DOS/VSE** (ECPS:VSE) — an IBM 4331 processor, for example — your generated DOS/VSE can operate in either System/370 mode or ECPS:VSE mode.

When your DOS/VSE operates in ECPS:VSE mode, the generated virtual storage is considered as one single address space, and DOS/VSE deals with addresses on a single level: virtual. In ECPS:VSE mode, virtual storage equatable to processor storage is available as address space for problem programs, except for the portion that accommodates the resident routines of the supervisor. A DOS/VSE operating in ECPS:VSE mode fully exploits the advantages of ECPS:VSE.

Figures 2-13 and 2-14 exemplify the allocation of storage in DOS/VSE for operation in System/370 mode and ECPS:VSE mode, respectively.

As has been mentioned earlier in this chapter, you allocate storage to partitions after IPL or between jobs.

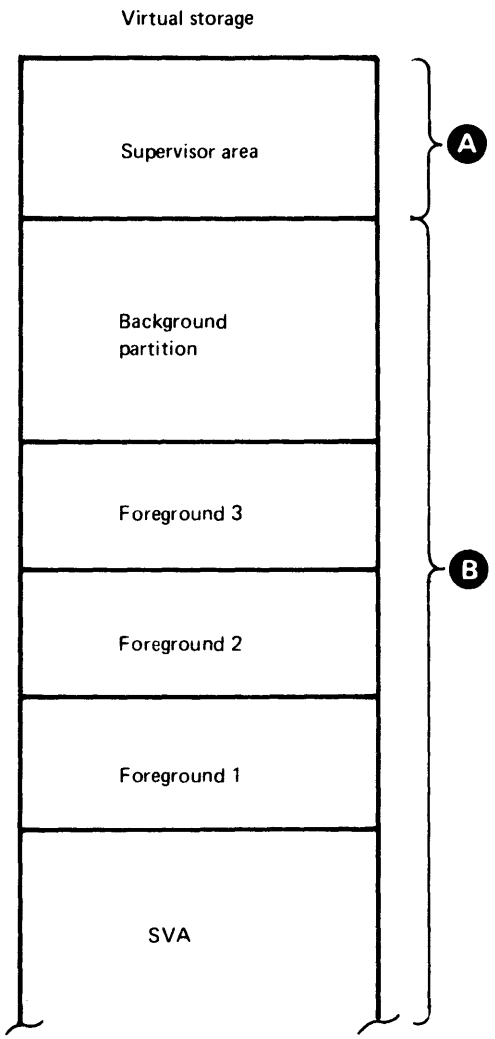


* Each partition requires a minimum of 128K bytes of virtual storage

** A 1:1 mapping of processor storage

*** Space is used for the execution of programs in real mode or for fixing pages of a program (program internal communication areas, for example)

Figure 2-13. Example of storage allocation for DOS/VSE in System/370 mode — schematic representation



(A) DOS/VSE fixes all pages that contain resident routines of the supervisor (routines of the supervisor that can tolerate paging may, during IPL, be declared as pageable).

(B) Programs that are executed from one of the partitions or from the SVA are read into page frames of the page pool for actual processing. If a program is to be executed in real mode, DOS/VSE fixes all pages of that program until execution of the program is finished.

Figure 2-14. Example of storage allocation for DOS/VSE in ECPS:VSE mode — schematic representation

Examples of Partition Allocation The examples used so far to illustrate partition allocation have not defined partition sizes. Consider two further illustrations that use specific storage allocations and, in general terms, define the use of each partition. Figure 2-15 lists the partitions, their use, and the size of the real and virtual address areas allocated to each one for a DOS/VSE operating in System/370 mode; the example represents a System/370 Model 138 with a processor storage of 512K bytes. Figure 2-16 illustrates a similar example of partition allocation for a DOS/VSE operating in ECPS:VSE mode on an IBM 4331, also with a processor storage of 512K bytes.

Partition or DOS/VSE component	Program run mode	Partition use	Real Address area allocation	Virtual Address area allocation
Supervisor area	real	System control	82K	
Background	virtual	System service, large applications, tests (not frequently used)	0K	256K
Foreground-3	virtual	Normal batch production	0K	128K
Foreground-2	virtual	Urgent jobs (non-scheduled)	0K	128K
Foreground-1	virtual	VS/POWER without remote job entry (also requires allocation of real address area for foreground-1; unused page frames are made available to the page pool).	30K	180K
SVA	virtual	Shareable phases	0K	460K
Total allocated to supervisor and real address area			112K	
Total allocated to virtual partitions, SVA, and label access work area				1152K
Storage not explicitly allocated (used for programs executed in virtual mode)			400K	

Figure 2-15. Example of storage allocation in a 4-partition DOS/VSE — System/370 mode

Partition or DOS/VSE component	Program run mode	Partition or component use	Virtual address area allocation
Supervisor	real	System control	82K*
Background	virtual	System maintenance — large applications and tests (not frequently used)	256K
Foreground-3	virtual	Normal batch processing	128K
Foreground-2	virtual	Urgent (non-scheduled, high-priority) jobs and test runs	128K
Foreground-1	virtual	VSE/POWER	210K **
SVA	virtual	Shareable phases	460K
Total of virtual storage address space allocated			1,382K
Total allocated to virtual partitions and SVA (see *, below): 1,336 - 82 =			1,310K
Storage available as page frames for actual problem program execution at any point in time (see ** below): 512 - 128 - 82 - 30 =			272

* Since DOS/VSE fixes all pages of resident supervisor routines for the duration of system operation, the page frames these pages occupy are not availabel to the page pool.

** VSE/POWER may fix pages in up to 30K bytes of processor storage; however, if VSE/POWER can operate with less than 30K bytes fixed in processor storage, it returns page frames not needed to the page pool.

Up to 128K bytes of processor storage are needed to accommodate micro code routines.

Figure 2-16. Example of storage allocation in a 4-partition DOS/VSE — ECPS:VSE mode

Library Services

One powerful feature of a DOS/VSE installation is its range of libraries that enables programming data to be stored online, readily accessible.

DOS/VSE supports four types of libraries: core image, relocatable, source statement, and procedure. The first three types of libraries exist in two different classes, system libraries and private libraries, whereas the procedure library exists only as a system library. The system libraries are contained in the system residence file (SYSRES), and can be accessed by all partitions. Private libraries can be contained on disk volumes other than the one used for SYSRES; those libraries can be accessed only by programs in the partitions to which these libraries are assigned.

The DOS/VSE librarian programs provide service, organization, and maintenance functions for all types of libraries. Additional services for efficient use of the core image library are provided by the DOS/VSE linkage editor program and the DOS/VSE loader.

This section of the manual first discusses the purpose and use of the various DOS/VSE libraries followed by an overview of the available library services.

Using the Libraries

The DOS/VSE libraries have the following main functions:

core image library

The core image library serves to catalog programs (in units called phases) that have been processed by the linkage editor and are ready for being loaded into storage for execution. Each system must include a system core image library in order to have certain system programs online whenever they are needed.

In DOS/VSE, phases can be in either non-relocatable or relocatable format. A non-relocatable phase is loaded into a partition directly at the address computed when the program was link-edited. For a relocatable phase, DOS/VSE modifies the load address, the entry points, and the address constants; such a phase can, therefore, be loaded at a storage address different from the one for which it was link-edited.

relocatable library

When the linkage editor encounters a reference to a name which is not defined within the module that is being processed, the linkage editor searches the relocatable library for a module with this name specified in the external reference. If the search is successful, the module found is linked with the module(s) being processed.

Explicitly specified modules from the relocatable library can be included with modules being link-edited. In this way, sections of code that are used by a number of different programs need be written, translated, and cataloged in relocatable object format only once.

source statement library

The source statement library contains sequences of source language statements, called books. The library consists of a number of sublibraries used, for example, to store macro definitions. When the assembler encounters a source statement with an unknown operation code, it tries to retrieve the book with the same name as this unknown operation code from the source statement library. The assembler then substitutes the code found in the library for the source statement in question.

Similarly, when a compiler encounters a reference to a book in the source statement library, it gets the specified book from the library and substitutes it for the reference in the source program that is being processed.

procedure library

The procedure library is used to catalog frequently used sets of job control and linkage editor statements in card image format. Depending on a system generation option, procedures may contain inline data, such as utility modifier or librarian control statements. Cataloged procedures can be included in the job control input stream and may be modified by **overwrite** statements as the job stream is processed.

A cataloged procedure can be invoked by specifying the procedure's name in the job control EXEC statement. For example:

```
// EXEC PROC=PAYRL
```

causes procedure PAYRL to be included in the job stream replacing the // EXEC statement.

private libraries

In addition to system libraries, DOS/VSE offers the option of private core image, relocatable, and source statement libraries. Private libraries are particularly useful, for example:

- in an installation with a large number of programs or applications.
- in an environment where, for security reasons, it is desirable that certain programs be kept under lock and key.

Available Library Service Programs

DOS/VSE contains library service programs that perform maintenance and service functions for all libraries. These functions include:

- Cataloging, renaming, and deleting any element in a library.
- Printing any library element or directory.
- Punching any library element.
- Copying all elements from one library to another library of the same type, or copying only those elements that do not yet exist in the destination library.
- Condensing a library and changing its size and location.
- Creating a new system disk pack and creating private core image, relocatable, and source statement libraries.
- Link-editing a program to a core image library.
- Relocating a program when it is loaded for execution.

The last two of the above services deserve additional attention and are discussed in the subsequent section.

Link-Editing a Program for Execution

The output of a language translator is an object module in machine language. It may be witten to the SYSPCH device and then cataloged into the relocatable library, or stored in an intermediate storage area on disk (SYSLNK) if link-editing is to follow translation immediately, or both.

Assembly or compilation of source programs is not affected by virtual storage considerations. The generated object program is the same whether it is to be executed in real or in virtual mode.

Object modules cannot yet be executed for the following reasons:

- Programs have to be relocated to the partitions in which they are to run.
- References to locations or labels in other modules (that is, external references) may still have to resolved.

Relocation and resolution of references are done by the linkage editor, which gets its input from SYSLNK and, optionally, from a relocatable library. The output (executable programs, called phases, with specific load addresses) is always stored in a core image library, either permanently or temporarily. If an output phase is reenterable and relocatable, it is also placed in the SVA, provided you submitted to DOS/VSE an appropriate request. A link-edited phase is executable, either directly or after processing by the loader of DOS/VSE.

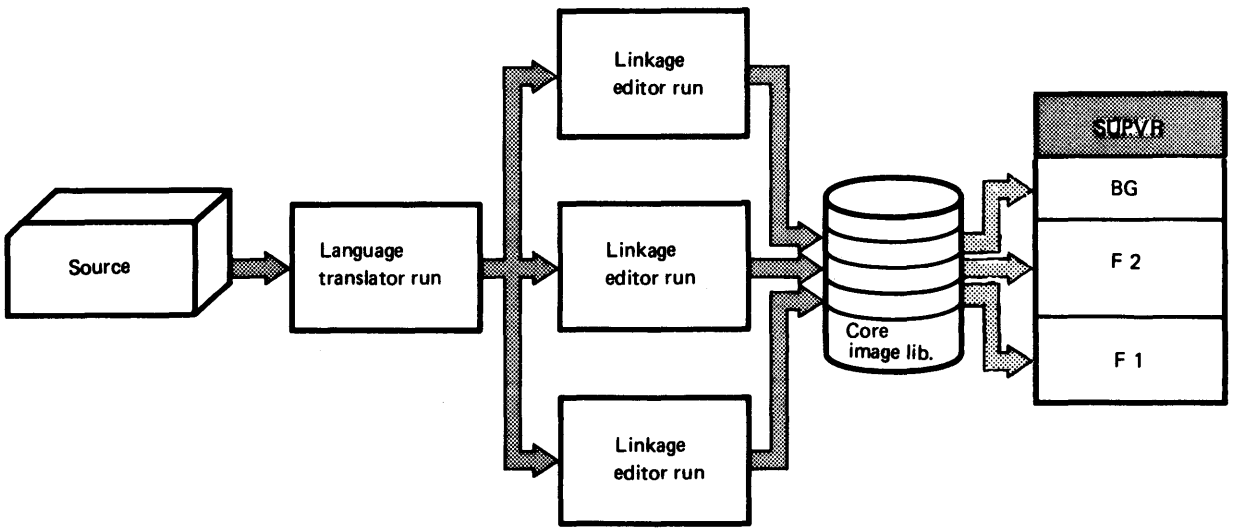
Under DOS, whenever a user program had to be able to run in any of the three partitions, the program had to be self-relocating, or three copies of the program had to be present in a core image library, each link-edited for one of the available partitions, and each with a different name.

Under DOS/VSE, however, the linkage editor is capable of producing relocatable phases; the loader in the supervisor relocates a relocatable phase, if necessary, when loading it for execution into the selected partition.

This facility is of particular advantage in a multiprogramming environment; it can save a considerable amount of processing time and disk storage space as shown in Figure 2-17. If, for any reason, the size of the supervisor increases, the facility saves relink-editing of programs that were linked with the end of the supervisor as the load address.

Figure 2-18 shows how the libraries, the language translators, and the linkage editor combine for efficient operation under DOS/VSE. Figure 2-19 summarizes the possibilities of link-editing and relocating prior to execution.

DOS or early DOS/VS without the relocation capability:



DOS/VSE with the relocation capability:

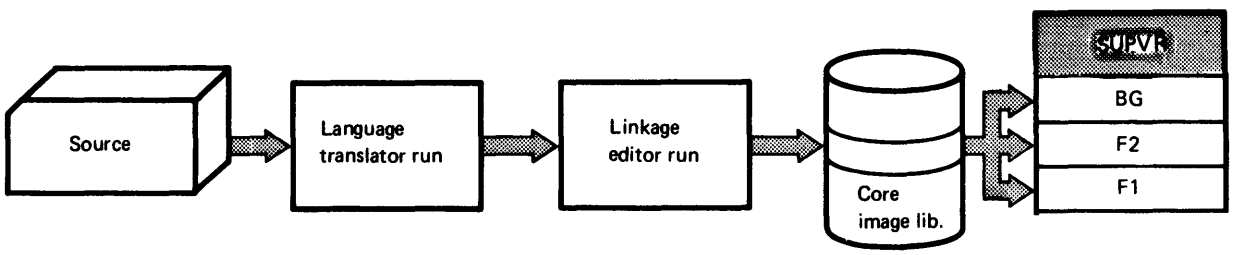
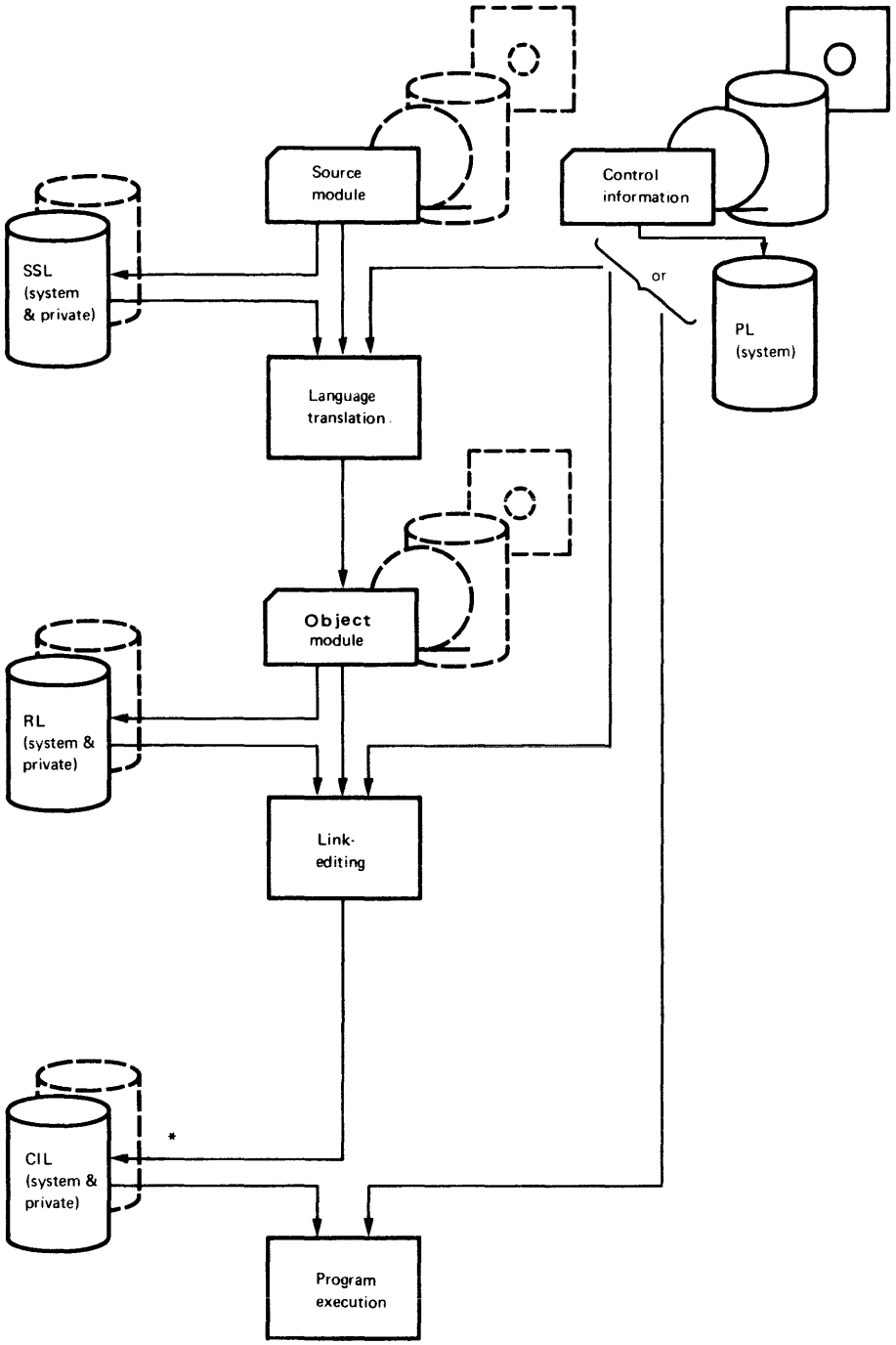


Figure 2-17. Link-editing with and without the use of the relocation capability



* If a phase is SVA-eligible and the user has requested it in the PHASE statement and in the SDL, the phase is also placed in the SVA. From there it can be executed directly whenever it is called by any of the partitions.

SSL – source statement library
RL – relocatable library
CIL – core image library
PL – system procedure library

Figure 2-18. Interrelationship of language translators, linkage editor, and libraries

Specified relocation option	Output of the linkage editor
ACTION REL (Requesting relocatability)	A relocatable phase. That phase can be loaded for execution into any of the available DOS/VSE partitions.
ACTION NOREL (Requesting non-relocatability)	A phase for execution only in one specific partition. Under a DOS/VSE operating in 370 mode, you can request that the phase be linked for execution in the processor storage area allocated as 'real address area' to the pertinent partition.
None	The same as for ACTION REL (a relocatable phase).

Figure 2-19. Link-edit options for relocatability of phases.

Data Management

Data storage and retrieval requirements, and how a data processing department responds to those requirements, are often essential concerns of everyone involved in data processing operations. Some basic questions concerning data management are:

- What are the **processing requirements** of a file:
  - — How many records of the file will be accessed each time the file is to be processed?
  - — How many records have to be added to or deleted from the file each time it is to be processed?
  - — How often within a given period of time (week or month) is there a need for processing the file?
- What type of **file organization** is best suited to the processing requirements for the file?
  - — Will a sequential organization meet all of the processing requirements?
  - — Is direct access of individual records of the file a processing requirement?
- On what **medium** (magnetic tape, cards, disk, etc.) is the file to be stored?
- What **data security** considerations should apply to the file during and after its processing cycle (for example, who besides the owner of the data may have read access or write access or both; is there a need for that data to be kept under lock and key)?

How the DOS/VSE data management facilities assist in arriving at appropriate answers to the first three of these questions is outlined in this section (for a more detailed discussion of data management topics, see

DOS/VSE Data Management Concepts). Data security is discussed in a separate section.

Data Organization and Access Methods

data organization

Data organization refers to the techniques used in placing records on an auxiliary storage device (magnetic tape or disk, for example). It involves such considerations as:

- The choice of **storage media** best suited to the processing requirements of the data.
- The **sequence of records** in the file. For example: the records could be sorted on one control field or on several, in a prescribed hierarchy; they could be in ascending or descending order.
- The **length of records.** Records in a file can be of fixed length or of variable length; they can be relatively short or relatively long.
- The **blocking factor** for the file. This determines how many logical records constitute a block of records (for transfer from the central processor to the I/O device and vice versa); and it may be an important factor in storage media utilization and in processing efficiency.
- The use of **indexes** with a file on a direct access device to provide an efficient means of randomly selecting specific records.
- The use of programmed **addressing techniques** to determine where a record is stored on a direct access device and the location from which the record can subsequently be retrieved for processing.
- The type and number of **data additions** to an already existing file. It is of major importance whether many or few data additions and updates are made to a file, whether additions and updates are made in sequential or random order, whether a file is accessed by more than one program in different partitions, or whether a file is to be a read-only file.

access method

Access method refers to the routines which assist the programmer in transferring records in a **particular data organization format** between processor storage and an I/O device. It is important to understand the relationship between data organization and access methods. Broadly speaking, how data is organized and the type of device on which this data is stored largely determine the access method that can subsequently be used to retrieve that data.

DOS/VSE provides several methods of data organization. For each of these, there is an access method which allows one or more techniques of file creation and retrieval. These data organizations and their associated access methods are briefly described below.

Sequential Access Method (SAM) and Organization

file organization

Sequential organization means that records physically follow one another in a sequence usually determined by one or more control fields within each record. Examples of control fields are name or employee-number in a personnel file, or catalog number or part number in an inventory file.

Sequential organization is the most widely used method of data organization and is supported for all device types except telecommunication terminals. Card files, print files, diskette unit files, and magnetic tape files are always organized sequentially, simply because the physical characteristics of those devices require the reading or writing of one record after another. Files on disk frequently are also organized sequentially, in control number sequence.

data access

If required, records are sorted into their prescribed sequence prior to or while creating a sequentially organized file. The Sequential Access Method (SAM) can create a sequential file from sorted records and subsequently retrieve those records for sequential processing. In addition, by utilizing certain macros (in programs written in assembler language), sequential files on disk or tape may be positioned to specific blocks prior to reading or writing. Records from sequential disk files may be **updated**, meaning that each record may be written back onto its original location after having been changed by the program.

applications

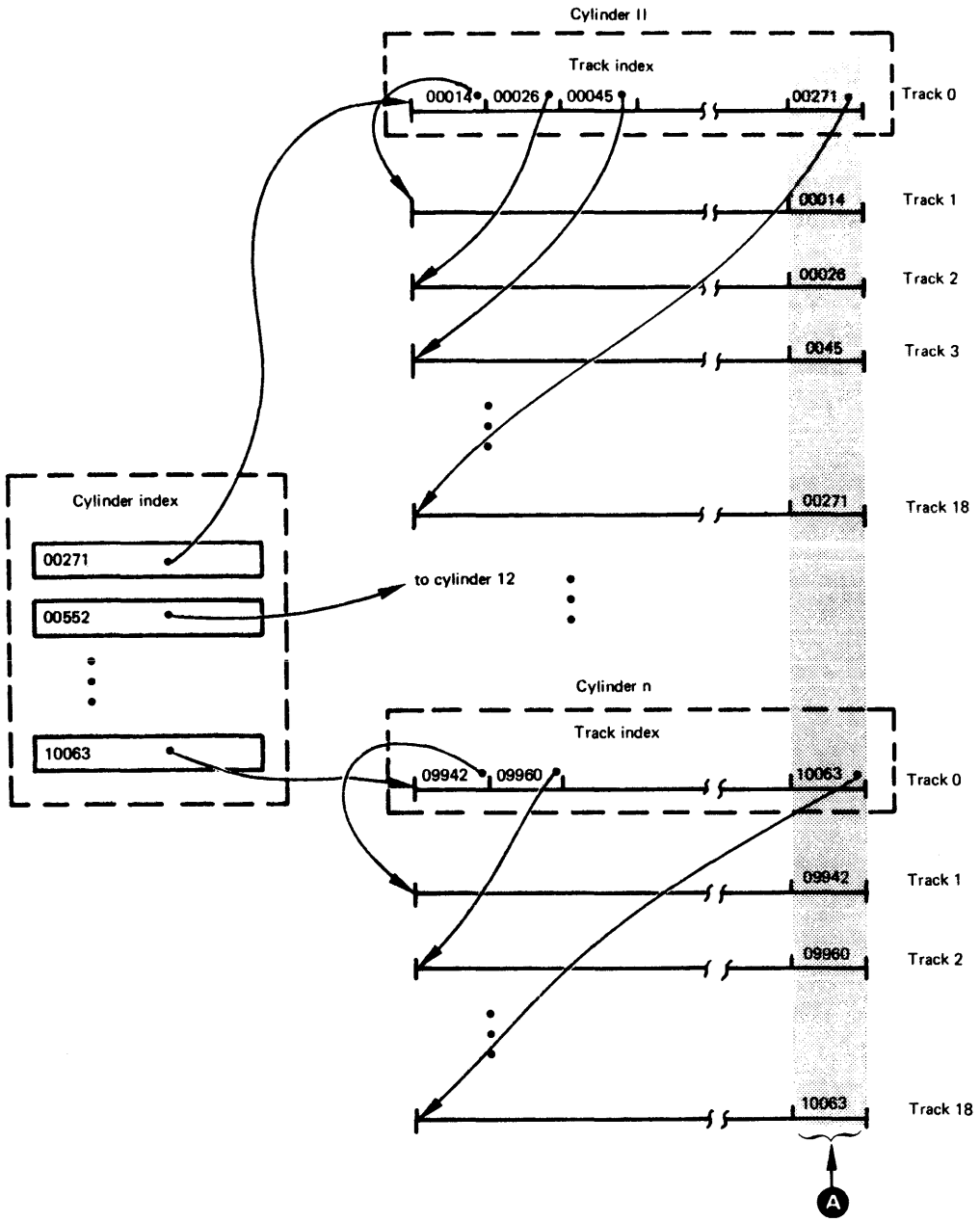
Sequential organization and access method are used for some files in most data processing installations since the requirements of many applications are met entirely by reading (or writing) all records of a file, one after the other. This means that transactions are held and batched until a number of them are available for processing against a master file. The batch of transactions is then sorted into the same sequence as the master file, which is subsequently updated at periodic intervals (such as daily, weekly, or monthly) depending on the volume of activity and the need for keeping the master records current. Payroll files, for example, are frequently organized sequentially; they are, typically, processed once per pay period with transactions consisting of employee time cards, piecework records or similar applicable data, producing checks and earnings statements and updating year-to-date figures in the master records.

Indexed Sequential Access Method (ISAM) and Organization

The physical characteristics of a disk make it practicable to retrieve a record from any location in a file instead of having to go from one record to the next starting at the beginning of the file until the desired record has been located. DOS/VSE exploits these characteristics through the indexed sequential access method and organization.

index and file organization

An indexed sequential file is made up of (1) records in logical sequence by control field (or key) and (2) indexes that are built when the file is created: a track index, a cylinder index, and (optionally) a master index. Each index entry is composed of the key of a data record, or a lower level index entry, and the physical address at which the record, or lower level index entry, is located on the disk. Figure 2-20 shows a schematic representation of the indexed sequential file organization.



A Record with highest key on the track, except on track 0

Figure 2-20. Schematic representation of the indexed sequential file organization

The programmer may process indexed sequential files sequentially when the definition of the programming application requires this approach, or he may process them randomly, retrieving a particular record from the file, if this approach is better adapted to the requirements of the job. He may even combine both approaches in the same program. Both retrieval methods are easy from the programmer's viewpoint:

data access

- For **sequential** retrieval, he simply issues the appropriate I/O statement or macro (such as GET) at the point in his program where he requires the next record, and ISAM makes the record available to the program.
- For **random** retrieval, the programmer merely provides the key of the

record he needs, issues the appropriate statement or macro, and ISAM presents the specific record to the program for processing. Index searching, deblocking records, and handling device requirements are all accomplished internally by the access method.

overflow area

ISAM also provides the routines for creating a file from sorted input, building the index, and adding records to an existing file. For the insertion of records into an existing file, additional disk space, called overflow area, is reserved. On sequential retrieval of a file that has data in the overflow area, records are retrieved in logically sequential order.

applications

This degree of processing flexibility makes ISAM attractive in many applications. It is frequently used in inventory record maintenance, where, for example, sequential retrieval is most convenient for stock status reports and batch transaction processing of non-critical items, but where random retrieval is necessary for balance of account or quantity in stock inquiries.

ISAM is available for programs that process data stored on any of the following DOS/VSE supported disk devices:

2314/2319
3330/3333 Models 1 and 2
3340 and 3344 Model B4 in 3340 simulation mode
3350 Model 1 in 3330 compatibility mode

bypassing device restrictions

In view of these device restrictions, VSE/VSAM is the better access method whenever large files have to be processed. Programs written to process data through ISAM can also process data contained in VSAM files through the IBM supplied ISAM Interface Program (IIP). The above mentioned device restrictions do not exist for ISAM programs when they access VSAM files through the IIP. For more information about VSE/VSAM, a licensed IBM programming support, turn to Chapter 6 of this manual.

Direct Access Method (DAM) and Organization

While ISAM offers both sequential and random retrieval, DAM concentrates on random retrieval only and provides an access method that can accomplish this function efficiently. DAM requires, however, that you establish a direct relationship between the keys of the records and their physical addresses on disk. This means that the program, by using the key of a record, can calculate or look up in a table the corresponding record address, and either directly store the record (on output) or directly retrieve it (on input). Greater burden and responsibility is placed on the programmer; the benefit is a potentially faster record retrieval for specialized applications such as reservation systems or transaction inquiry programs.

In a DAM file:

- Records normally are not in logical sequence by key.
- Tables to accomplish retrieval functions are not built and maintained by the access method.

- The physical location of each record on the disk is determined by an algorithm in the program that accesses the file.
- Depending on the addressing technique used by the programmer, **gaps** may exist; also, the addressing technique could produce **synonyms,** which are multiple records for the same address. The programmer is responsible for solving these problems.

DAM is available for programs that process data stored on DOS/VSE supported count-key-data disk devices.

Telecommunication Access Methods

In telecommunication, data processed by the computer system is obtained from remote locations. Input and output of data to and from the computer is performed using terminals, comparable to I/O devices, which are connected to the installation's central processor through communication lines.

The telecommunication access methods you would normally use under DOS/VSE are: the Basic Telecommunication Access Method — Extended Support (BTAM-ES) and the Advanced Communications Function/Virtual Telecommunication Access Method for VTAM Entry (ACF/VTAME). Both these access methods are available as licensed IBM programming support; they make the required telecommunication services available through assembler language macros.

Normally, you would use the services of these telecommunication access methods through other IBM licensed programming support: the VSE/Interactive Computing and Control Facility (VSE/ICCF) or the Customer Information Control System (CICS/VS) or both. This licensed programming support relieves the programmer of the burden of coding telecommunication access routines and yet allows for utilizing all of your computing system's services via a remote terminal.

For more information about the available IBM telecommunication access methods and other telecommunication-oriented licensed programming support, see *Chapter 6: Licensed and Nonlicensed Programming Support.*

Under DOS/VSE, telecommunication is used mainly for:

- Message switching. Messages from one remote station are routed to another through the central processor.
- Message processing. Messages received from remote terminals are processed by an application program running in the central processor.
- Remote job entry. Entire jobs are submitted to the central processor from remote terminals; this is a telecommunication application for which IBM provides support under DOS/VSE through a feature of VSE/POWER. The functions available with VSE/POWER are discussed briefly in Chapter 6 of this manual.
- Data collection. The central processor collects information presented by the various remote terminals for later analysis and processing.

- Inquiry/response. Remote stations request information from a central source of data.

applications

Some factors that would suggest the use of a telecommunication system are:

- Customer convenience. Brokerage firms, for example, require rapid execution and confirmation of customer orders.
- Inventory control. Manufacturing applications are common, but there are other 'inventories' — such as airline space availability — that can be effectively controlled by a telecommunication system.
- Credit Control. Central data can provide assurance that a customer has funds or credit approval.
- Management control. Immediate access to centralized files provides more timely information for control of business operations.
- Industrial control. Computer control of key production factors increases productivity of capital equipment (for example, in petroleum refineries).
- Equipment centralization. In collecting data from remote sources, either intermittently (as in production data collection) or periodically (as in central summarizing of statistical data from distant branch offices), one central processor may do the processing that would otherwise require a separate system at each remote site.
- Innovation. Some applications are just not possible without a telecommunication system (for example, remote debugging, text editing, and computer-assisted instruction).

High-Level Language Support for Data Management Functions

For use with DOS/VSE, IBM provides compilers as licensed programs for the following languages:

- COBOL
- PL/I
- RPG II.

Each language has data management facilities based on those provided by the access method available as part of DOS/VSE or as separate licensed programming support. The I/O statements in these languages are not macros as in assembler language, but are statements in the format prescribed by the language. For more information about this support, see either the pertinent IBM-provided language documentation or publication *DOS/VSE Data Management Concepts.*

In addition to the above mentioned compilers, the IBM-supplied FORTRAN IV compiler, available as a type I program, can be run under DOS/VSE. More information about this compiler is given in Chapter 6 of this manual.

Coding Your Own Channel Program

The access methods available under DOS/VSE are designed to meet the file organization and access needs of the majority of users. The data management facilities contained in these access methods are collectively called Logical Input/Output Control System (LIOCS). LIOCS functions, requested in a user-written program by macros or high level language statements, are translated by the access methods to code that accomplishes the physical transfer of data between storage and I/O devices. This code consists, essentially, of an appropriate channel program.

DOS/VSE allows the assembler language programmer to write his own channel programs if he wants to. Data management routines supporting user-written channel programs are collectively referred to as physical IOCS (PIOCS). PIOCS provides the user with a degree of flexibility in handling I/O operations greater than that provided by LIOCS, but at the same time, requires a greater understanding of and responsibility for the detailed aspects of input/output operations.

Protection of Data

It is important, if not to say vital, to protect an installation's data — programs and files, that is — against accidental or intentional misuse or destruction. Under DOS/VSE, programs in different partitions, possibly invoked from a terminal, may attempt to retrieve and update the same file or even the same record simultaneously; moreover, the installation's online disk storage may contain classified data which cannot be accessible by any or all users.

DOS/VSE provides facilities that help to protect your data against misuse or destruction. These facilities are:

- File labeling
- Protection against duplicate assignment
- DASD file protection
- Track hold specification
- Secured DASD files
- Resource protection through macros
- Access authorization checking and access logging.

File Labeling

File labels assist in achieving file protection. File labels are records associated with the files stored on magnetic tape, on disk, or on diskette. These labels provide unique identification for the **files**, and the **volumes** of these files.

For data on magnetic tape, all labels are optional; for data on disk or diskette, each volume must have one **volume label,** and each file must have a **file label.** Additional labels can be created on magnetic tape and disk volumes and can be processed as required.

For the creation of a file, you specify the contents of the labels. Then, when the file is processed as input (see Figure 2-21), you again specify, in job control statements, the contents of the label, so that the data management routines can compare the specified data with the actual label. If data management detects a mismatch between the actual label and the label information, the job is terminated.

Complete information on how DOS/VSE handles tape, disk, and diskette labels is provided in publications *DOS/VSE Tape Labels* and *DOS/VSE DASD Labels.*

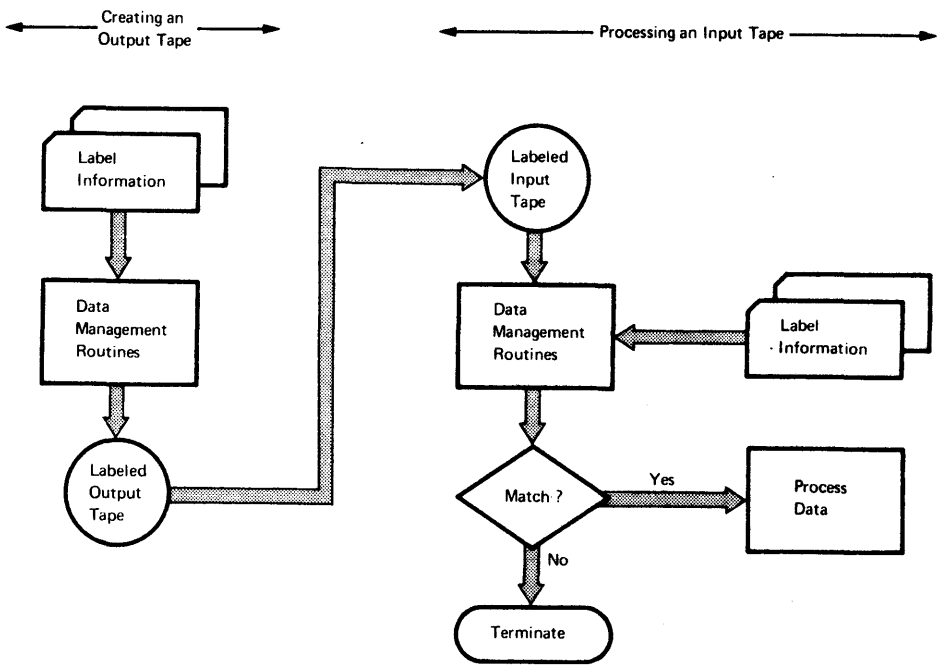


Figure 2-21. Label processing

Protection Against Duplicate Assignments

In a multiprogramming environment, it is essential that no two programs in different partitions gain access to the same device, except for disk units. A single disk unit may contain a number of files; therefore, disk units can

contain files for processing by programs in more than one partition. DOS/VSE provides the following protection against duplicate assignments:

- Unit-record devices and magnetic tape drives assigned to a program in one partition cannot be assigned to a program in another partition, except when VSE/POWER, a licensed IBM spooling program, is operational and access to the devices is under its control.
- A magnetic tape used for SYSOUT data cannot be used for any other system or programmer logical unit. If this is attempted, a message is issued to the operator, who can then take corrective action.
- If additional protection for a tape volume is required, the pertinent tape drive should be assigned with the volume serial number specified. DOS/VSE, in attempting to find this volume, bypasses unassigned tapes that do not contain the volume. If DOS/VSE cannot find a volume with the requested volume serial number, the system requests the operator to mount the volume.

DASD File Protection

This facility helps to improve data integrity for DASD files that are being processed under DOS/VSE. The facility prevents programs from writing outside the extents specified for their files. The other files on the DASD volume are thus protected against unintended destruction.

Track Hold Specification

This is another DOS/VSE facility that contributes to data integrity. The facility prevents two or more different programs from updating the same track (or block) on a DASD concurrently. All programs that make use of this facility should specify track hold. The track (block) is then held for the first program that accesses it and is not freed until that program has finished processing for this track or block.

File Security

When you create an output file on disk, you can request, by means of a job control statement, that the file is to be secured. For a diskette file, you can make this type of a request by specifying a DTF parameter, provided your program is written in assembler language. Creating a secured file means:

- The operator is notified with a message when a secured file is being used as input. He can then make the decision as to whether the file should be processed.
- The label area display program does not display the label information of a secured file.

Resource Protection through Macros

In a multitasking environment, there is essentially nothing to prevent a task from using the resources of another task in the same partition. These resources can be I/O devices, files, data areas, routines, and the like. When, however, all tasks in a partition use resource protection macros to protect shared resources, concurrent use of resources is prevented.

The **resource-share control facility,** which is not available in an SCP only environment, efficiently controls the sharing of a user-defined resource by two or more tasks through macros. You define the particular resource by a unique name of your own choosing and indicate to DOS/VSE how the resource may be shared while it is accessed by your own program. You can define the resource to be:

- nonshareable (no other program can access the named resource).
- shareable with programs that define this resource as one of type S (shareable).
- shareable with programs that define this resource as one of type S plus one program that defines the resource as one of type E (exclusive). The program that defines the resource as one of type E may be your program.

The types of resources whose sharing can be controlled with this facility are: libraries, disk volumes, catalogs, files, and control blocks in virtual storage.

Another efficient method of resource protection is the use of **ENQ and DEQ macros.** By issuing an ENQ macro, a task can gain exclusive control of a resource and relinquish control later by issuing a DEQ macro. Any other task which also does this for the same resource is placed in the wait state until the first task has given up control.

Access Authorization Checking and Access Logging

This facility is not available in an SCP only environment. The facility, which implements a high level of data security under DOS/VSE, uses VSE/ICCF with or without the VSE/Access Control — Logging and Reporting (LOGREP) program. Both these programs are licensed IBM program products.

With this facility available, DOS/VSE compares a user-specified identification and password with identifications and passwords contained in user profiles defined by you. A job execution request (or log-on request from a terminal) with a non-matching user-identification or password is rejected by DOS/VSE. In addition, DOS/VSE compares the security level defined by you for a data element that is to be accessed with the security level(s) contained in the particular user's profile. Non-matching security levels cause DOS/VSE to deny this access and to discontinue the requested processing.

The LOGREP program, if included in your DOS/VSE, logs all security violations and, optionally, also all authorized accesses to secured data

elements; it is capable, for example, to produce access reports by elements or by users.

A data element in this context can be a private library, a member of a library, or an individual file (for example a payroll master file).

Supported I/O Devices

DOS/VSE supports the full range of I/O devices that are attachable to DOS/VSE supported central processors. Some of these devices, however, are supported only on an EXCP macro level; in other words, the user, in order to use such devices, would have to code his own channel programs.

For a complete list of supported central processors and I/O devices, refer to *Appendix A: DOS/VSE Device Support.* This appendix provides also information about DOS/VSE imposed restrictions, if any. Detailed information about possible device configurations with the various DOS/VSE supported processors is given in the following publications

IBM System/370 System Summary, GA22-7001.

IBM 4300 Processors Summary, Input/Output and Data Communications Configurator, GA33-1523.

System-Operator Interaction

Operator-to-system communication plays a vital part in the efficient operation of a data processing installation. DOS/VSE provides facilities that ensure timely and effective interaction between the operator and the system.

The operator of a DOS/VSE controlled data processing installation is normally called upon to:

- start up the system.
- prepare the I/O devices. For example, mount tapes and disk packs for each individual job.
- initiate and control the execution of jobs.
- respond to DOS/VSE or program originated requests for information or action.

More specifically, the operator of a DOS/VSE controlled installation can:

- initiate and control multiprogramming operation.
- invoke inquiries to obtain information about the operating system's status.
- cancel the execution of a job; for instance, in the case of an unending program loop.

- diagnose problems with the help of the available problem determination aids.

For its part, DOS/VSE:

- alerts the operator when a condition requiring his intervention occurs.
- provides information such as start and stop times of jobs and run times.

system-to-operator communication

Communication from DOS/VSE to the operator is in the form of messages written on the operator communication device, which may be the screen of a display operator console (3277 or 125 DOC) or a printer-keyboard. The messages describe the situation that may or may not require operator action. Each message is identified by a unique number. This number serves as an entry point into *DOS/VSE Messages,* where an explanation is given for each message, along with a description of the required action, if any.

operator-to-system communication

Communication from the operator to the system is in the form of commands and replies to messages. There are three types of commands:

- Job control commands. They are almost identical in format and scope to the job control statements. These commands can be submitted for processing only to the job control program.
- Attention routine commands. They request DOS/VSE services (for example, changing the size of partitions or displaying the current processing priority assignments). To submit an attention routine command, the operator simply presses the Request key on the system's console and then keys in the command.
- Screen control commands. These are statements for manipulating the information displayed on the display operator console, if one is used.

The operator can interrupt processing at any time by pressing the Request key on the console keyboard. He can then enter a command and signal the system to resume processing, or he can instruct the system to suspend processing in a particular partition after the current job or job step (for example, in order to allow time for changing printer forms or device assignments).

Operator replies to messages are generally short 1-word answers such as RETRY, IGNORE, or CANCEL. Alternatively, the operator can allow the system default option to take effect in most situations requiring an intervention.

operator-to-problem program communication

In addition to operator-to-system and system-to-operator communication, DOS/VSE supports direct operator-to-problem program communication, provided that the problem program includes an operator communication routine. The operator can then request direct communication with this routine by issuing a specific command.

This facility of direct operator-to-problem program communication could be useful, for example, for inquiry to an inventory file at unpredictable times. The inquiry routine could be included in any program that always

occupies a partition and be invoked through the operator's communications routine when desired.

System Utilities

The primary objective of a data processing installation is to do productive work, that is, to execute the installation's problem programs whenever there is a need for processing the data these programs are concerned with. However, smooth and efficient operation of a DOS/VSE controlled installation requires the execution of system services that are not directly concerned with the control of execution of user problem programs. The subsequent paragraphs highlight some of these services, which are known as system utilities.

initialize disk or tape

load print buffers

Frequently, the execution of user problem programs involves job preparation steps such as initializing a disk volume or a volume of magnetic tape. Other examples of job preparation steps are the loading of the control buffer(s) of a printer with the proper buffer image for forms control or for using a universal character set or for both. DOS/VSE includes programming support that automates these job preparation steps to a great extent.

assign alternate track (or block)

Occasionally, DOS/VSE detects a defective track on a CKD-type disk volume or a defective block on a fixed-block architecture (FBA) disk device; in that case, an alternate track (block) must be assigned to the defective one and the data, if any, stored on that defective track or block is to be restored or rebuilt on the alternate one. DOS/VSE provides programming support that automates these operations.

surface analysis program

DASDs with non-removable disk packs (for example, the IBM 3350 or the IBM 8860) require additional surface analysis and track reclamation support. When a malfunction occurs on a DASD with non-removable disk packs, a program controlled diagnostic approach rather than the frequently used disk-volume swap technique must be applied.

backup and restore fast copy disk

A failure may occur that makes DOS/VSE inoperative (a head crash, for example). In a situation such as this, you have only one choice: start the operation anew using a backup copy of your operating system and, if necessary, also of data. DOS/VSE includes programs that copy disk volumes for backup purposes either onto disk or onto magnetic tape and that restore data so copied from magnetic tape onto disk.

More information about available system utility programs is given in *DOS/VSE System Utilities.*

System Serviceability and Debugging Aids

reliability, availability, serviceability (RAS)

The RAS facilities of DOS/VSE are designed to maintain a high degree of trouble-free performance of your DOS/VSE controlled installation. Debugging procedures are provided to help you in choosing the debugging aid best suited to obtain information about a particular type of error or malfunction. The facilities that make up RAS are:

- **Debugging aids** that provide you with information about the system at the time of a program, operator, or hardware error. This includes facilities for taking dumps of selected areas of virtual storage and for tracing events such as specific instructions or interrupts.

- **Recovery Management Support** (RMS) that, depending on the severity of the failure, enables DOS/VSE to recover from an error condition caused by any of the following hardware failures: processor storage errors, central processor instruction errors, I/O channel errors, control unit errors, and device errors. It also records hardware failures on the system recorder file (SYSREC) and prints messages on SYSLOG that keep the operator informed about the condition of the installation's hardware.

- The **Environment Recording, Edit and Print Program** (EREP) that allows the user to print the contents of the system recorder file, pre-formatted, on SYSLST. EREP has many options that provide for selective printing of summarized and detailed information about the state of an installation's hardware.

- The **Online Test Executive Program** (OLTEP), together with a set of device test programs, constitutes an online test system. Its functions include the diagnosing of I/O errors, the verification of I/O device repairs and engineering changes, and the checking of I/O devices.

- The **Reliability Data Extractor** (RDE) enables you to record the reason for an IPL procedure as well as the number of IPL procedures carried out on a system over any specified time period, for example, during an operator's shift. RDE also provides for an end-of-day time stamp to be recorded on SYSREC during system operation. This RDE information may be of great value to IBM service personnel.

Another important aspect of system reliability and availability is the correction of known problems in IBM supplied programming support. Correction of a known problem before that problem occurs at an installation may well contribute to the overall availability of a DOS/VSE controlled installation.

PTF installation

component installation

For the correction of known problems in IBM supplied programming support, IBM provides program temporary fixes (PTFs) which are to be installed on the user's DOS/VSE. The PTF installation tool, the **Maintain System History Program** of DOS/VSE, keeps track of this type of system update activity and, in addition, helps you install licensed programming support such as VSE/Advanced Functions (which is briefly discussed in Chapter 6 of this manual).

Assembling a Program

DOS/VSE includes the assembler program primarily as a supervisor generation tool. Through the program's powerful macro processor, tailoring the supervisor routines to system configuration and installation operation requirements is reduced to an automated process controlled by user-specified generation macros.

Occasionally, an application requires the utilization of system functions in a fashion that is not supported by the high-level language(s) available at the installation. An assembler-language subroutine linked to the application problem program normally is the answer to this kind of a problem.

Routines whose execution depends on the occurrence of specific events within the system (for example, routines of complex telecommunication applications or of programs with input from a magnetic ink character reader) normally are written in assembler language. By coding a routine in assembler language, you can closely control the available computing services of your installation.

A program (or routine) written in assembler language can be submitted to DOS/VSE for assembly only or for assembly, link-editing, and execution, all in one job (see also Resource Management and Job Control earlier in this chapter). However, before making use of the latter possibility, you should ensure (through assembly only runs) that the assembler-language source code is free from assembly errors.

The foregoing paragraphs on assembling programs conclude the discussion of the computing services available under DOS/VSE. The subsequent section deals with system generation.

Chapter 3: System Generation

Normally, system generation is a process concurrent with the execution of problem programs. Only the initial installation of the current release of DOS/VSE in a DOS/VS Release 34 or earlier DOS/VS release environment or in a non-DOS/VS environment requires stand-alone operation for system generation.

System generation is the creation of an installation-tailored operating system based on the DOS/VSE SCP and VSE/Advanced Functions as distributed by IBM. The process of system generation includes:

- Generating a supervisor adapted to the installation and its application needs.
- Assembling or compiling and link-editing IBM programs and programs of your own as necessary and also cataloging these programs in the appropriate libraries.
- Deleting unnecessary components to free additional disk space.
- Editing, formatting, or deleting libraries as required.

Once the DOS/VSE SCP and VSE/Advanced Functions are installed, your DOS/VSE is operational. Most likely, however, you will generate one or more supervisors that are adjusted to meet the configuration of your installation and your operational requirements.

The system core image library, the relocatable library, the source statement library, and the procedure library may also be edited. Private libraries may be created according to the user's needs.

Briefly, the system generation process is as follows: with the assistance of your IBM system engineer, you code a set of supervisor generation macros describing your installation's configuration and functional options. These macros are assembled, resulting in a new supervisor, which is cataloged into the system core image library. Certain modules, such as IOCS modules, may be assembled from the macro definitions in the source statement library and added to the relocatable library. Finally, you may link-edit certain programs of your own, catalog these into the core image library, and load them into the SVA as well. The result of these operations is an operational DOS/VSE to which additional (licensed or unlicensed) IBM programming support can be added.

Planning for System Generation

Detailed planning for system generation saves you unnecessary repetition during the process. Essentially, planning for system generation consists of:

- Planning the options and estimating the size of the supervisor. This entails selecting the options that are to be included in the supervisor and estimating the cost of these options in terms of bytes of processor

storage. The supervisor is composed of assembled supervisor generation macros. These macros describe the machine configuration, and the installation's processing options. The size of the assembled supervisor depends on the options selected in each of the supervisor generation macros and the particular hardware configuration.

- Planning the contents, organization, and ultimate size of the system and private libraries, and of the shared virtual area. This entails distributing the available DASD space among the libraries needed for daily use. You should decide on the size and number of libraries and on the size of the shared virtual area when planning for an operational system. You should consider the number of available disk drives, the number and size of the programs, which programs you want to be resident in the shared virtual area, the impact that the expansion of one library has on other libraries on the same disk pack, and your plans for future space requirements.

DOS/VSE includes programming support that automates, for example, the computing of library space requirements, if you have used DOS/VS at your installation in the past, or that merges libraries and condenses newly created libraries.

More detailed information for planning system generation is given in *DOS/VSE System Management Guide.* Implementation procedures are documented in *DOS/VSE System Generation* and in the *System Information* manual for VSE/Advanced Functions.

Shipment of DOS/VSE

The mimimum DOS/VSE is shipped in two entities: the DOS/VSE SCP and the licensed VSE/Advanced Functions. The DOS/VSE SCP is shipped in the form of a system core image library, a system relocatable library, a system source statement library, and a system procedure library. VSE/Advanced Functions is shipped in the form of private core image, relocatable, and source statement libraries. The contents of these libraries are listed in the *Memorandum to Users* that accompanies the shipment. The system core image library must be retained on the operational volume because it contains the DOS/VSE programs in executable format. The other libraries may be deleted if they are not needed. Given below is a summary of the contents of the system libraries when they are shipped:

- The core image library contains all DOS/VSE SCP support that can be invoked via job control. This support is link-edited and ready for execution, once VSE/Advanced Functions is installed. The DOS/VSE SCP as shipped is operational for system generation and hardware service purposes.

- The relocatable library contains the DOS/VSE SCP support including data management modules in object-code format, but not link-edited. You may, if you wish, link data management modules into your problem programs.

- The source statement library contains IBM-supplied macro definitions. You may assemble the macros that you require. Some of those definitions combine with macro definitions supplied as part of the

private source statement library for VSE/Advanced Functions to comprise a complete set of supervisor assembly definitions; you can assemble IOCS modules from the IOCS macro definitions. For your convenience, the source statement library also contains sample programs and system generation job streams that can be retrieved as needed.

- The procedure library contains procedures (sets of control statements) for linking DOS/VSE components to the core image library and for deleting IBM supplied libraries from any of the three types of libraries shipped with DOS/VSE; procedures for the use of MSHP, an IBM-supplied system installation and service aid,and procedures for defining standard labels are also included.

IBM supplies the DOS/VSE SCP packages on either magnetic tapes or disk packs. If they are shipped on tape, they must be copied onto disk at your installation before the system generation procedure can begin.

Additional nonlicensed or licensed IBM programming support available for use at a DOS/VSE installation must be ordered separately. For information about this additional programming support, turn to Chapter 6 of this manual. The next chapter, Chapter 4, discusses the advantages of DOS/VSE over DOS.

Chapter 4. Advantages of DOS/VSE over DOS

This section highlights major functions of DOS/VSE that were not available in DOS or are different from those in DOS. The section should be of particular interest to users of DOS who plan to move to DOS/VSE.

DOS had been designed and implemented to support the System/360 product line of computers. DOS/VSE, the successor operating system of DOS/VS, supports the System/370 product line of computers as well as central processor with Extended Control Program Support: DOS/VSE (ECPS:VSE).

While DOS supports only peripheral equipment that can be attached to System/360 CPUs, DOS/VSE now includes the support for all of the newly available equipment that is attachable to System/370 processors and to processors with ECPS:VSE. Section *New Devices Supported* later in this chapter lists a selection of newly supported devices; for a complete list of supported devices, see *Appendix A*.

The advantages of DOS/VSE over DOS become apparent as the various new and changed facilities are being discussed in the subsequent paragraphs.

virtual storage

DOS/VSE fully supports the virtual storage concept introduced with the System/370 product line of CPUs and improved in processors with ECPS:VSE. Through DOS/VSE, the advantages of virtual storage are made available to the user with virtually no extra programming effort on his part. On the contrary, by relieving the programmer of the burden of making his programs fit into a certain amount of available processor storage, DOS/VSE makes programming of application programs easier.

The virtual storage concept is discussed in Chapter 2 of this manual under *Virtual Storage Concepts*. Its advantages over the non-virtual storage organization are discussed under *Advantages of Virtual Storage* later in this Chapter.

relocating loader facility

A relocating loader facility has been added to DOS/VSE. The facility allows the linkage editor to create program phases that are relocatable and to resolve the load address to any location specified by the programmer at the time of program execution. The facility updates all of the entry points and address constants in the relocatable phase. You need retain only a single copy of a program in a core image library for execution in any partition. Programs that are link-edited to a beginning address at the end of the supervisor, for example, are now no longer affected by increases in supervisor size.

The function and use of the relocatable loader facility are also discussed under *Link-Editing a Program for Execution* in Chapter 2 of this manual. For more information about using the facility, see *DOS/VSE System Management Guide*.

additional foreground partitions

The batch job facilities (BJF) in DOS have become standard. The single program initiator (SPI) program is no longer supported. DOS/VSE with

VSE/Advanced Functions allows you to multiprogram in one background and up to six foreground partitions (BG, F6, F5, ..., F1. In an SCP only environment, you can multiprogram in one background and up to four foreground partitions.

variable partition priority

DOS/VSE allows you to modify the IBM defined sequence of partition priorities by using an operator command.

DOS/VSE assembler

The DOS/VSE assembler replaces the Assembler D. This new assembler not only supports the System/370 instruction set, and the instruction set of a processor with ECPS:VSE, it also processes definitely faster than Assembler D. This improved performance is achieved by placing all library macro definitions on a separate sublibrary in edited (pre-processed) format. All IBM-supplied macro definitions are delivered in edited format, and you can use the assembler to produce your own edited macros for inclusion in the macro sublibrary.

There are three ways in which user-written macros can be retained:

- A macro used only temporarily is normally maintained as part of the program, and is physically included in the source deck.
- A macro used in more than one program can be included as a separate book in the copy sublibrary and be maintained in this form. To call it, the programmer must first include a COPY statement at the beginning of his program to identify the macro as a source macro.
- A macro used more frequently should, after testing, be edited and kept in the macro library. Then it can be referenced directly by macro instructions.

A macro library used by previous assemblers can either be used as a copy sublibrary, or it can be converted to a macro sublibrary by letting the assembler edit all the macros and by including them in a macro sublibrary.

For a complete discussion of the DOS/VSE Assembler, refer to *Guide to the DOS/VSE Assembler* and *OS/VS-DOS/VSE-VM/370 Assembler Language.*

shared virtual area (SVA)

The SVA is located in the high address range of virtual storage. Phases that are stored in the SVA can be used concurrently by more than one partition. To be used concurrently, a phase must be relocatable and reenterable. In addition, the phase must be listed in the system directory list (SDL) with SVA eligible indicated and it must be cataloged in the system core image library. Execution of phases from the SVA is faster than loading the same phases from the core image library into a virtual partition.

Besides executable phases, the SVA contains the SDL and the system GETVIS area, an area used by DOS/VSE phases for acquiring virtual storage dynamically. The SDL contains directory information about all phases in the SVA and, optionally, other frequently-used phases in the system core image library. The SDL allows for fast retrieval of the phases listed in it. For more information about the SVA, see *DOS/VSE System Management Guide.*

extended I/O device assignment options

The ASSGN statement or command, which assigns a logical I/O unit to an actual device, has been extended to provide for greater flexibility and more ease of use in making symbolic device assignments. Frequently, a programmer is not aware of the environment in which his job will run, or he may not be concerned with what particular physical device will be used for his purposes. For these cases (among others), the following new functions are now available for making logical unit assignments.

- **Generic assignments.** Rather than specifying a physical address, you can assign a logical unit
  - — to the physical unit used in a previous statement by referring to the logical unit used in that previous statement.
  - — to a device class: for example READER, PRINTER, DISK.
  - — to a device type: for example 2400T9 for a 9-track magnetic tape drive of the 2400 series or 3340 for a disk volume on a 3340 DASD.
- **Automatic disk volume recognition.** You can specify a volume serial number to have DOS/VSE search for the specified volume. If the volume has been mounted previously, the pertinent job can access that volume immediately; if the volume is not mounted, DOS/VSE issues a mount-request message to the operator.

improved performance of the copy program

The performance of the copy (CORGZ) program, which is part of the librarian, has been improved considerably. For transferring library members from one library volume to another, the program's current version needs significantly less time than its earlier versions.

rotational position sensing (RPS)

DOS/VSE supports rotational position sensing (RPS), which is a standard feature on the IBM 3330/3333/3344/3350 devices and an optional feature on the IBM 3340 devices. The feature provides the ability to overlap rotational positioning operations on one device with service requests for other devices on the block multiplexer channel, (or its equivalent). The feature thus provides for a potential increase in channel utilization which can lead to an increase in I/O and system throughput. DOS/VSE support for RPS is provided in all access methods which support the devices.

new core image library organization

The organization of the core image library under DOS/VSE provides for fast retrieval of all phases. Subdirectories are no longer required. The restriction that private core image libraries be of the same device type as SYSRES has been removed. Private core image libraries can reside on any disk device supported by DOS/VSE as system residence device. For more details on this, refer to the *DOS/VSE System Management Guide.*

procedure library

DOS/VSE supports an additional library type as a system library: the procedure library. This library enables you to catalog frequently used sets (procedures) of job control and linkage editor statements and to include the cataloged procedures in the job input stream by submitting only one job control statement.

cardless system support

DOS/VSE installations may be configured without card reader or card punch devices, provided that an IBM diskette I/O unit is included. This minimum configuration offers a reduction in device cost and an improvement in the rate of data entry. An IBM diskette I/O unit can be used also as a system input/output device.

The stand-alone dump program (as generated by DOSVSDMP) and the stand-alone version of the Fast Copy Disk Volume system utility can be stored on diskettes and loaded from the attached diskette I/O unit into the system. The stand-alone version of the Initialize Disk utility (distributed on the PID tape) has been altered to accept control information from the Display Operator Console (DOC).

installation of and service for DOS/VSE

IBM offers licensed programming support in addition to the DOS/VSE SCP. Installing such support results in an unusable DOS/VSE if one or more of the modules that are being installed replace SCP modules to which an IBM supplied or local fix was applied. Example: SCP Modules A, C, D, and H include an IBM supplied fix to correct one specific problem; by installing a licensed programming support, SCP Modules C, D, and X are being replaced thus destroying the fix that involved the SCP Modules A, C, D, and H.

maintaining a system history

prerequisite checking

To prevent a situation such as this, DOS/VSE includes the MSHP program. This program automatically maintains and checks the system history — which is mandatory — for possible destruction of a previous fix and indicates (by message) which prerequisites for the installation of a certain programming support have not been met. The program offers various aids for automatic service activities in addition to the ones mentioned above. Examples are: restoring the IBM supplied DOS/VSE from magnetic tape onto disk and merging the shipment library with the installation's existing library; recording in the system's history file all program fixes applied to the SCP and to licensed components installed at your installation.

For details about the services available through MSHP, see publication *DOS/VSE Maintain System History Program (MSHP) User's Guide.*

Advantages of Virtual Storage

For a System/360 with DOS, the need for making programs fit into the available processor storage often requires overlay techniques. Structuring these overlays adds to the complexity of solving a problem. With the increased use of high-level languages, multiprogramming, expanded system control programs, and applications that require relatively large amounts of storage (telecommunication, data base, etc.), the need for more processor storage space and a more dynamic use of it is still growing.

True, the System/370 models, and more so processors with ECPS:VSE provide significantly more processor storage capacity than the comparable System/360 models. This additional capacity, however, does not relieve all the constraints associated with this storage. It does not eliminate the waste of storage through, for example, dormant code, as might be the case with an inactive or low activity telecommunication network. It cannot avoid the waste of storage through storage fragmentation as a result of small programs executing in big partitions; the system has no means of dynamically utilizing the fragments of free storage space.

Consider also the following situations:

1. An application is designed to operate in a 50K real storage area, which is adequate to handle current processing needs and provides room for some expansion. Some time after the application is installed, program maintenance changes and the addition of a new function causes one of the programs in the application to require 51K and another to require 52K. Installation of the next real storage increment cannot be justified on the basis of these two programs, so time must be spent restructuring the programs to fit within 50K.

2. An existing application has programs with an overlay structure. The volume of transactions processed by these programs has doubled. Additional processor storage is installed. However, the overlay programs cannot automatically use the additional storage. Therefore, reworking of the overlay structure programs is required to make them non-overlay and, thereby, achieve the better performance desired.

3. A simple, low-volume, terminal-oriented inquiry program that operates for three hours a day is to be installed. If the program is written without any overlay structure, it will require 60K of storage to handle all the various types of inquiries. However, because of a low inquiry rate, only 8K to 12K of the total program is active at any given time. The inquiry program is designed to operate in 12K with a dynamic overlay structure in order to justify its operational cost.

4. A series of new applications are to be installed that require additional computing speed and twice the amount of processor storage available on the existing system. The new application programs have been designed and are being tested on the currently installed system until the new one is delivered. However, because many of the new application programs require more storage than the existing applications, testing has to be limited to those times when the required amount of processor storage can be made available. Although another smaller scale model is also installed that has time available for program testing, it cannot be used because it does not have the amount of processor storage required by the new application programs.

5. A large terminal-oriented application is to be operative during one entire shift. During times of peak activity, four times more processor storage is required than during low activity periods. Peak activity is experienced about 20 percent of the time and low activity about 40 percent. The rest of the time, activity ranges from low to peak. Allocation of the peak activity storage requirement for the entire shift cannot be justified and the application is designed to fit into a smaller area of processor storage. As a result, a dynamic program structure must be used; certain desired functions are not included in the program and response during peak and near peak activity periods is affected.

In the situations described, processor storage is the constraining factor. However, even if more processor storage were added to a system as needed, the system could not automatically make use of it. Applications would still have to be redesigned, and the waste of storage through fragmentation and dormant code would still exist.

To assist in solving these problems, the virtual storage concept offers a means of dynamically and automatically using processor storage, storage fragments, as well as storage space added to the system at later times. With virtual storage support, programs may, to a certain extent, exceed the limits of available processsor storage and still get this storage as needed for their execution.

The time required for the execution of a program has always been and still is dependent on such factors as the mix of programs executing concurrently, their relative priorities, system and application file placement, and in some cases on the particular data being processed. Under DOS/VSE, program performance is also highly dependent on such factors as the amount of processor-storage overcommitment, the storage reference patterns of the program, and the speed of the paging device. The peformance of each program should be evaluated in the light of at least these factors.

If you operate an online or real time system with specific performance or response requirements, make sure that adequate resources (processor storage, processing time, channels, disk arms, etc.) are available. In some cases it may be necessary to test the program using the specific user workload and configuration to verify what system resources are needed to give adequate performance.

New Devices Supported by DOS/VSE

Among the new I/O devices supported by DOS/VSE (in accordance with their hardware configurability) are:

- The IBM DASDs 3330/3333, 3340/3344, and 3350.
- The IBM DASDs 3310 and 3370 which use the fixed-block architecture (FBA) concept.
- Display operator consoles — the standard operator console on System/370 Models 115 and 125 and on processors with ECPS:VSE as well as the 3277 Display Station, which may be used also as an operator console.
- The IBM 2560 Multifunction Card Machine and IBM 5425 Multifunction Card Unit.
- The IBM 3203 Printer.
- The IBM 3800 Printing Subsystem (the programming support for this subsystem must be ordered separately).
- The IBM 3540 Diskette Input/Output Unit.

A complete list of the I/O equipment supported by DOS/VSE is given in *Appendix A: DOS/VSE Device Support.*

Chapter 5: Program Compatibility

This chapter discusses program compatibility for two distinct groups of users: users of DOS/VS and users of DOS. Any incompatibilities listed below for users of DOS/VS may apply also to users of DOS.

For programs that meet the prerequisites stated below, program compatibility can be ensured only by installing VSE/Advanced Functions together with the DOS/VSE SCP.

Users of DOS/VS

A program that has been executed successfully under DOS/VS will be executed successfully also under DOS/VSE provided the program

- has not been linked to absolute addresses — if it has been, the program must be relinked to obtain a relocatable version of the program.
- interfaces with DOS/VSE through IBM provided macros or through high-level language statements as documented in the appropriate language documentation.
- accesses supported I/O devices only.
- does not contain IDA lists if your DOS/VSE is to operate in ECPS:VSE mode — if it does, the program must be changed to replace the pertinent CCWs.
- does not access the interval timer at location 80 or the system time of day (SYSTOD) at location 84 — if it does, the program must be changed to use the GETIME macro instead.
- does not access libraries by user-written library access routines — in that case, the library (or libraries) to be accessed must be retained on the previously used library device type.
- does not use seek separation.
- does not include BTAM modules — if it does, the program must be relinked with a new BTAM-ES module that has been assembled using the current BTAM-ES BTMOD.
- does not interface with QTAM (queued telecommunication access method) — if it does, the program must be changed to use BTAM-ES or ACF/VTAME instead.

Users of DOS

Current DOS users who plan to change to DOS/VSE support will have to consider the following in addition to the applicable items given for users of DOS/VS.

- Current DOS data files can be processed by DOS/VSE, if compatible I/O devices are used:

  Programs written to process data for a 2311 can be executed to process the same type of data through the use of the available 2311 compatibility feature for a

  — 3330/3333 or a 3340 attached to a System/370 Model 125.
  — 3340 attached to a System/370 Model 115 or 135.
  — 3310 or 3370 attached to a processor with ECPS:VSE.

  Programs written for the 1052 Console Printer-Keyboard can be processed on the video display and 5213 Console Printer of /370 Model 115 and 125 through the use of the available 1052 compatibility feature.

- Existing assembler language source programs can be assembled by the DOS/VSE Assembler, provided that no user written macros are called. If such macros are called, the user must either supply COPY instructions for the macro definitions at the beginning of all source decks in which the macros are used, or convert his library macros to edited macros and include them in the macro sublibrary.

- Existing high level language source programs can be compiled if the appropriate compiler is available for DOS/VSE. COBOL D programs must be changed or converted with the Language Conversion Program before they can be processed by the DOS/VSE COBOL compiler. RPG programs must be adapted to RPG II.

- Previously compiled or assembled DOS object programs can be link-edited without modification under DOS/VSE. User programs will execute provided the following points have been taken into account:

  — Devices specified by the program must be available on the system on which DOS/VSE will run.
  — Programs that depend on central processor circuitry not supported on System/370 or on processors with ECPS:VSE may not execute properly.
  — Proper processing of time dependent programs that run under earlier versions of DOS is not ensured. IBM recommends to have these programs executed in real mode (without paging).
  — Programs that deliberately create program checks may not run properly.

- Supervisor sizes will be definitely larger in DOS/VSE than with previous DOS versions. Therefore, programs will have to be relink-edited if they were not written to be self-relocating.

- User written I/O appendage routines must either run in real mode or adhere to certain restrictions which are described in *DOS/VSE System Management Guide.*

- Programs with self-modifying channel programs must run in real mode.

Chapter 6: Licensed and Nonlicensed Programming Support

IBM offers a variety of programs and programming support for inclusion in an installation's DOS/VSE. This section highlights that support, limiting the discussions to general purpose type support and programs. For a complete list of programming support available from IBM for inclusion in an installation's DOS/VSE, contact the IBM branch office serving your locality.

Licensed IBM programming support is available only through a license agreement between IBM and the user who intends to utilize such support at his installation. Licensed support may be a self-contained program which can be executed under DOS/VSE or, as in the case of VSE/Advanced Functions, it may be a set of routines or modules that enhance one or more component programs of the DOS/VSE SCP.

VSE/Advanced Functions should be installed to establish the operating environment for the use of self-contained licensed or nonlicensed programs.

Self-contained programs may be categorized as follows:

— Service programs (or subsystems)
— Language translators
— Application programs (or systems)

The last category will not be dealt with in this chapter.

VSE/Advanced Functions

This is a set of routines in source statement format plus a number of relocatable modules ready for installation on a DOS/VSE SCP. This support package should be installed for the use of self-contained licensed programs; on a System/370 Model 135 or 145, VSE/Advanced Functions requires that the optional CPU timer and clock comparator are installed.

VSE/Advanced Functions, which you install to establish a specified operating environment, provides significant additional support as highlighted below.

- **Support of up to seven partitions instead of only five.** An enhancement that may be of particular benefit for telecommunication-oriented installations which, nevertheless, have to cope with a considerable batch work load.

- **Deferred operator replies to messages.** The operator need no longer reply to messages in the same sequence as they are displayed on the console. He can defer the reply to one message per active task in the system before DOS/VSE locks the console for the transmission of messages from other tasks.

If DOS/VSE issues a request for an operator's response to a previously displayed message and this message has not yet been rolled off the screen, then DOS/VSE flags this message as not deletable. As a result, most messages that require an operator's response will be retained on the screen until the operator has entered this response; the need for the operator to have a message redisplayed will therefore arise less frequently.

- **Switchable fast CCW translate facility.** Specifying NOFASTTR in the // OPTION statement causes DOS/VSE to turn off fast CCW translation for the associated job if your system was generated to include the fast CCW translation facility. This may be of significance for jobs that are unlikely to use the same I/O areas and I/O control blocks repeatedly.

- **Job-to-job communication.** This facility is available to assembler language programmers. The facility allows you to communicate up to 256 bytes of information from one job step to another or from one job to another within the same partition.

- **Resource-Share Control.** Macros are available for efficient control of the sharing of resources. The types of resources whose sharing may be controlled are: libraries, disk volumes, catalogs, files, control blocks in virtual storage.

  You define the resource whose sharing is to be controlled by specifying a symbol for it and by indicating to DOS/VSE how this resource may be shared by other programs while your program accesses the resource. You define the resource to be either of the E (exclusive) type or of the S (shareable) type and define how the resource may be shared with other programs. You can define a resource to be

  — nonshareable—no other user can access the named resource. In this case, your program must define the resource as on of type E.
  — shareable with programs that define this resource as one of type S.
  — shareable with programs that define this resource as one of type S. and one program, possibly your own, that defines the resource as one of type E.

- **Fast transient-phase-fetch facility.** VSE/Advanced Functions includes a procedure which, when executed immediately after IPL, loads a selected number of transient phases (supervisor routines that are loaded on demand) into the SVA of your DOS/VSE. Rather than fetching these phases from the system core image library, DOS/VSE simply moves them from the SVA into that transient area. This transient area is part of the supervisor.

- **Fast creation of the hard copy file.** With VSE/Advanced Functions, the first // JOB statement after a request for the creation of a hard copy file is being processed faster.

- **SDL index in fixed storage.** This facility may be particularly useful if your SDL in the SVA contains a considerable number of entries (there are 60 entries to a 2K-page).

Since the SDL is being searched by DOS/VSE using the binary search technique, searching a long SDL might result in several, successive page faults. The facility builds an SDL index and fixes this index in processor storage. Through this index, DOS/VSE can directly locate the page that contains the desired SDL entry and thus limit its binary search operation to just that one page. Only an SDL entry crossing page boundaries might cause an extra page-in operation.

- **Library device independence.** The facility allows you to allocate DASD space for private relocatable and source statement libraries on device types other than those used for system residence.

- **Label information area as a separate file.** By defining a label area outside the SYSRES file (but on the SYSRES volume), you can allow two central processors to share one SYSRES volume. This might be of significance to users of DOS/VSE under VM/370.

  You may also define a separate label information area if the standard one within the SYSRES file is too small. In that case, however, the standard label information area will no longer be used by DOS/VSE and, as a result, standard labels stored in that standard area must be resubmitted for storing them in the separate label information area.

  A separate label information area can be defined (or redefined) during IPL. Also during IPL, you indicate to DOS/VSE whether to use the standard or the separate label information area.

- **Automated system initialization.** This support allows you to write IPL and partition start-up procedures and to have these procedures cataloged in your installation's procedure library; from then on, your operator can start up DOS/VSE with almost no interaction on his part.

- **The page data set on up to 15 extents.** If your installation's supervisor was assembled to include support for a multiple-extent page data set, you can define that data set to reside on up tp 15 extents, up to three extents per volume. This support allows, for example, to take advantage of fixed-head areas on DASD for performance critical applications in a certain partition.

- **Alternate dump files.** The DOS/VSE dump facilities support the fast writing of dump information onto a DASD volume to which a new logical unit, SYSDMP is assigned. On this DASD volume, you can create two dump files. When one of the dump files is full, the DOS/VSE dump facilities write into the other one. This alternate dump file support is available for system dumps, dumps requested by the DUMP command, and stand-alone dumps.

- **Phase-load trace tables in virtual storage.** DOS/VSE maintains one wraparound table per partition in virtual storage. Each of these tables contains one entry per load request from the associated partition, and each entry holds information as follows: name of the phase, the phase's load address, and its length. In each of these tables, DOS/VSE records up to fifteen load requests before a wraparound occurs.

- **Improved performance of the linkage editor.** A linkage editor run under DOS/VSE with Advanced Functions completes faster than under DOS/VSE without this licensed support if modules stored in the relocatable library are to be included in the program that is being link-edited.

- **Implicit invocation of the linkage editor.** By specifying GO as an operand in the // EXEC statement for an assembly or a compilation of a source program, you request DOS/VSE to link-edit the program immediately after the language translator has finished processing and to execute the program immediately after it has been link-edited.

- **Improved performance of DOS/VSE under VM/370.** This facility, which is also known as DOS/VSE — VM/370 Linkage Enhancements, is available for DOS/VSE generated to operate in System/370 mode. The facility improves performance of DOS/VSE under VM/370 by:

  — avoiding many of the instructions that are redundant in a VM/370 environment (DOS/VSE avoids functions such as load leveling and paging as well as page fixing and page freeing; DOS/VSE executes fewer privileged instructions).
  — returning control to DOS/VSE immediately after VM/370 has handled a DOS/VSE detected 'pseudo' page fault, thus enabling DOS/VSE to dispatch (give control to) another program or task that is being executed under the virtual DOS/VSE.
  — avoiding a channel-program check interrupt that previously was caused by the DOS/VS BTAM routines in order to check for modified CCWs when the BTAM CCW string was being executed.
  — automatically closing the printer and punch files that are spooled by VSE/POWER. These files are printed (punched) without a specific request.

  The facility allows your programs to process data that is stored on fixed block DASDs, although DOS/VSE is operating in System/370 mode.

For more information about these functional enhancements and their usefulness at a DOS/VSE controlled installation, refer to *VSE/Advanced Functions, General Information,* GC33-6106.

If your installation uses VSE/ICCF, you can achieve a high level of data security at your installation. VSE/Advanced Function allows for an extension of the VSE/ICCF user identification and access-authorization verification services to jobs submitted to DOS/VSE partitions. VSE/LOGREP, another licensed IBM program, lists an audit trail of security violations (attempts to access protected resources without authorization) and, optionally, of authorized access to protected resources.

Previously executed jobs may be submitted unchanged as long as they do not access protected data.

VSE/Access Control — Logging and Reporting Program (VSE/LOGREP)

This program, together with VSE/Advanced Functions and VSE/ICCF, implements, for your DOS/VSE, a high level of data security at a DOS/VSE controlled installation.

If an attempt is made to access without authorization a data element that has been defined to be secured, VSE/LOGREP records the event on disk. Optionally, VSE/LOGREP records, in addition, each authorized access of such a data element. The information logged by this program for a recorded event consists of

— identification of the invloved data element
— identification of the user accessing (or attempting to access) the data element

The program can be invoked explicitly to provide a report. For this purpose, a variety of options are available to control the scope and level of detail of that report. You can, for example, request the printout of all recorded security events or only a summary of these events, or you can request a printout of events by specified users or by specified data elements.

Other major functions of the program are: dumping the recorded security events onto magnetic tape and restoring the dumped information to disk; reset and clear the file(s) of recorded events.

A data element in this context is a private library, a member of a library, or a file.

VSE/Virtual Storage Access Method (VSE/VSAM)

VSE/VSAM is an access method covering a maximum of possibilities for direct and sequential processing of fixed and variable-length (including spanned) records on direct-access devices. VSE/VSAM offers more functions than the access methods available with the DOS/VSE SCP on a high level of performance; it provides improved data security and integrity and a more flexible data organization.

file organization

The records in a VSAM file can be organized in logical sequence by a key field (key-sequenced file), in the physical sequence in which they are written into the file (entry-sequenced file), or according to the relative record numbers in the file (relative record file). The user can read, add, delete, and modify records in a VSAM file.

VSE/VSAM allows retrieval, storage, update, and deletion of records; it can handle variable-length as well as fixed-length records. The access can be either sequential or direct. It can be by record key, record address, or relative record number. The key can be that of an individual record or it can be a generic key specifying a group of individual records. Blocking and deblocking of records is done by the access method which optimizes block length to suit the device on which the file is written.

With a key-sequenced file, several records in sequence can be inserted as a group at one point in the file (this is faster than inserting them one at a

time with direct access, as ISAM requires). Also, VSE/VSAM allows accessing several records in key-sequence and then have VSE/VSAM skip to another portion of the file and access more records in sequence without having to search the entire index to find the new group of records (this is called skip-sequential access). For keyed or addressed sequential (but not skip-sequential) processing, there is also an option to process records backwards. Key-sequenced and entry-sequenced files may be accessed via alternate indexes.

VSAM catalogs

VSE/VSAM keeps control over the creation, access, and deletion of files and over the direct-access storage space allocated to those files. This is done by keeping information on file and space characteristics in a VSAM catalog. There are two kinds of VSAM catalogs, master and user catalogs. One master catalog is required with VSE/VSAM; any number of user catalogs are optional. User catalogs increase data integrity and facilitate volume portability.

data security
data integrity

VSE/VSAM offers improved data security and integrity. A file can be protected against unauthorized use through passwords. The various password levels available grant authority to read a file, to read and update a file, or to read and update both a file and the VSAM catalog. Data integrity is improved by minimizing data movement and index updating when records are added, by preserving both new and old index paths to data until an update is completed, by special formatting to indicate the end of a file as it is being created or extended, and by a provision for recovery from damage to the catalog.

language support and implementation

Support of VSE/VSAM is provided through macros in the assembler language, and through high-level languages such as DOS/VS/COBOL, PL/I Optimizer, and DOS/VS RPG II.

file sharing

VSE/VSAM uses the DOS/VSE track hold feature to allow files to be shared accross partitions. When a record is updated, the control area containing the record is protected under exclusive control. The remainder of the file may be accessed by programs running in other partitions.

For more information about VSE/VSAM see *VSE/VSAM General Information,* GC24-5143.

VSE/POWER

In all computing systems there is a large discrepancy between the electronic switching speeds of the central processor, and the speeds of unit-record devices (such as card readers, punches, and printers) which are relatively slow. By running your jobs in the DOS/VSE multiprogramming environment, you can improve the situation to some extent.

VSE/POWER, a licensed program, offers further improvement of system performance. It is designed for virtual storage to reduce central processor dependence on the relatively slow speeds of unit-record devices.

VSE/POWER decreases the execution time of unit record I/O-bound jobs by servicing I/O requests addressed to such devices at disk I/O speed; it reads and punches cards or prints reports or accomplishes both in parallel

during the execution of other jobs. The additional processor time used by VSE/POWER is negligible. In a typical environment of jobs with mixed characteristics, throughput may be substantially improved.

VSE/POWER requires one of the generated virtual partitions and allows you to execute programs in the remaining ones without the need for separate unit-record devices for each of these partitions. The unit-record devices used by VSE/POWER can handle the unit-record I/O activities for all the partitions that are being serviced by VSE/POWER.

Processing with VSE/POWER is a follows (see Figure 5-1):

- **Input.** VSE/POWER reads the job streams (job control statements, programs, and data cards) for the individual partitions and stores these in input queues on disk. The input may be entered from:

  — A local card reader or diskette device.
  — A magnetic tape unit.
  — A DOS/VSE partition not controlled by VSE/POWER.

- **Execution.** From disk, the jobs are transferred by VSE/POWER to the designated partitions and executed.

- **Output.** Unit-record output (printer and punch) of every job is stored on disk (or tape) by VSE/POWER before it is finally processed as output:

  — For a local printer or punch device.
  — For an outside partition not controlled by VSE/POWER.

controlling VSE/POWER

- **Control.** Throughout the different steps of processing (input, execution, and output), the jobs running under VSE/POWER are within user management through the following command language facilities:

  — **Job entry control language (JECL).** Along with your job control statements and unit record input, you may insert JECL commands to describe individual job execution or I/O requirements.
  — **Central operator commands.** From input until the final list and punch output, the central operator may discharge control functions such as starting and stopping job execution or displaying or altering execution and I/O characteristics.
  — **Cross-partition communication macros.** These macros enable a problem program running outside of VSE/POWER control to supervise the execution of jobs in a way similar to commands or JECL.

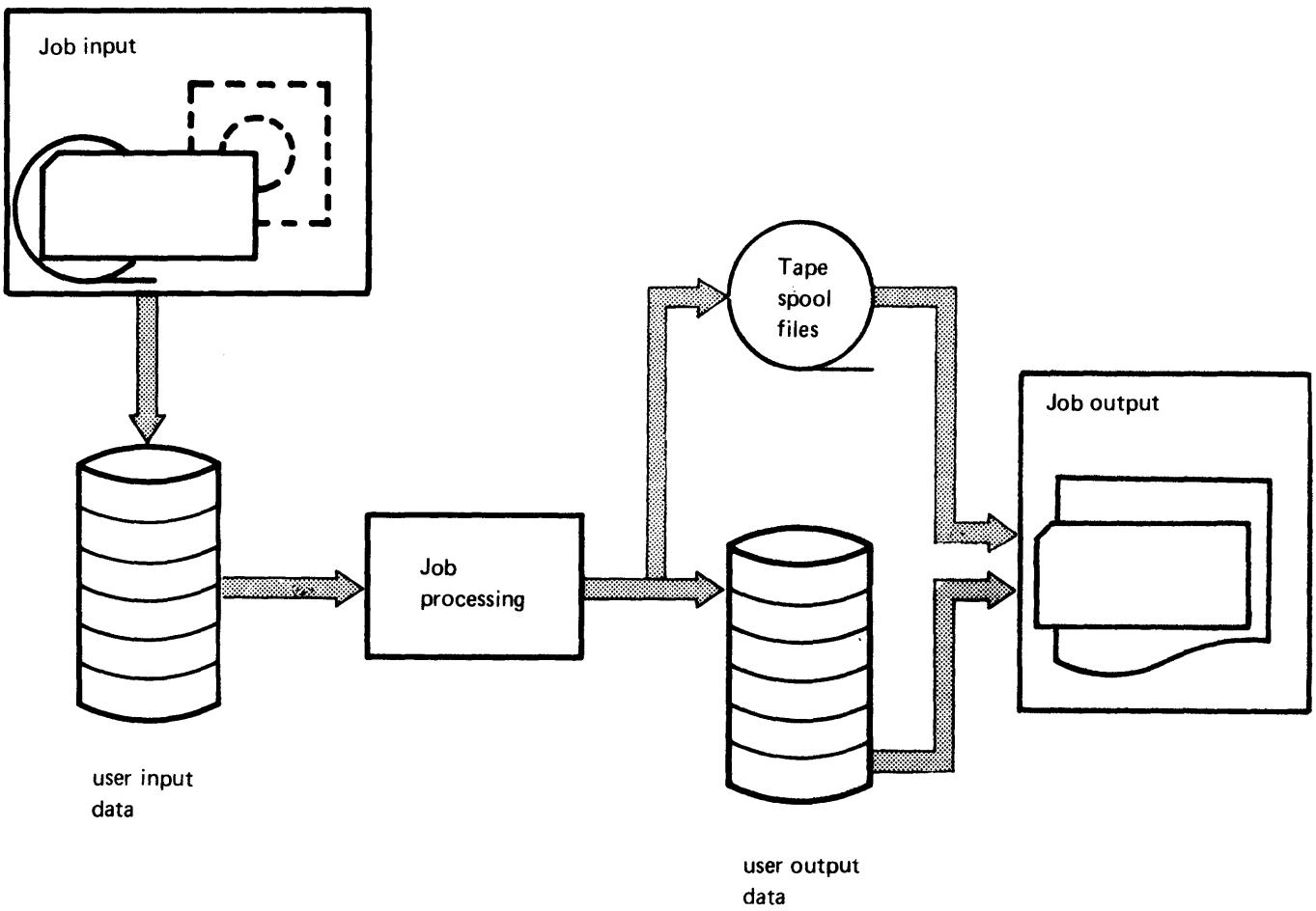


Figure 6-1. Processing with VSE/POWER

Job input as well as job output may be held in the queues for execution or printing/punching at a later time. This allows you to hold jobs that need, for example, two hours of execution or printing time until the system is less occupied.

Turnaround time for jobs with extensive printed or punched output can be improved by segmenting the output, that is, by printing or punching parts of the output before the entire job is finished.

Whenever a unit-record input or output device becomes inoperative or fails, the system can continue processing with those jobs already in the input queues on disk and store the output of these jobs on magnetic tape or in the output queues on disk. When the I/O unit becomes available again, reading, punching, or printing can continue.

remote job entry

VSE/POWER also offers a teleprocessing facility, the remote job entry feature (VSE/POWER RJE). With the feature, jobs may be submitted from remote terminals in very much the same way as if they were submitted via a unit-record device at the central location.

Jobs may be submitted remotely from:

— A remote BSC terminal.
— A remote SNA (SDLC) work station with a console and an input device (card reader or 80-column card image device).
— Another system's VSE/POWER if the multileaving facility of VSE/POWER is used.

The output of remotely submitted jobs may be directed to:

— A remote BSC Terminal (see Note below).
— A remote SNA (SDLC) workstation (see Note below).
— A local output unit at the central installation.
— Another system with VSE/POWER executing if the multileaving facility of VSE/POWER is used.

Note: Not necessarily the terminal from which the particular job was submitted.

job accounting

VSE/POWER also provides, optionally, an accounting facility. The facility combines DOS/VSE job accounting interface information and VSE/POWER job accounting information in the VSE/POWER account file for each job processed under control of VSE/POWER. You need not write your own data collection routine, but need to process the account file, sorting and summarizing the records to suit your particular accounting requirements.

For more information about VSE/POWER, see *VSE/POWER General Information Manual,* GH12-5128.

VSE/Interactive Computing and Control Facility (VSE/ICCF)

VSE/ICCF is an adaptation of the widley used DOS/VS Entry Time Sharing System, which has been available as a field developed program; it includes significant functional enhancements.

By including VSE/ICCF in your installation's DOS/VSE, you change the installation's basic operating characteristic from batch to interactive. Through VSE/ICCF, all of your installation's computing services are available, on a time-sliced basis, to the authorized users of terminals that are linked to your installation's central processor.

VSE/ICCF, which is available as licensed programming support, includes efficient editing and command-processing facilities that allow a terminal user to:

- build job streams and prepare input data for immediate or later processing.
- store such job streams or data (or both) in a library which may or may not be shareable.
- change (edit) previously stored job streams and data.
- request the execution of a job stream, monitor its processing status, and cancel execution of the job stream if this is appropriate.

- have output of processing requested by him directed to his own terminal, to an output device at the central location, or to another terminal.

VSE/ICCF includes procedures and utilities which greatly facilitate utilization and control of your installation's computing services from a terminal.

VSE/ICCF provides for a high level of data integrity and data security. Via a user profile, VSE/ICCF allows you to control that user's access to members of the DOS/VSE libraries (read-only, update, or not at all). A user's profile, and thus that user's ID and password, determines how other users may access DOS/VSE or VSE/ICCF library members created by that particular user. In addition, a terminal user can protect individual files or job streams by a password.

For more information about VSE/ICCF, refer to *DOS/VSE Computing and Control Facility, General Information,* GC33-6066.

Advanced Communications Function for VTAM Entry

This licensed programming support (ACF/VTAME) has been designed for use with DOS/VSE; its availability will be announced separately. This support allows for building and operating a variety of data communication systems under DOS/VSE.

ACF/VTAME, which can be used in a single-system as well as in a multiple-system environment, provides numerous communication functions and services. For a complete overview of ACF/VTAME support, see the publications

Advanced Communications Function for VTAM Entry (ACF/VTAME) General Information:

Introduction, GC27-0438
Concepts, GC27-0451

Some of the functions and services available with ACF/VTAME are highlighted below:

- Controlling sessions and data flow between application programs and terminals. A single request for a session or for data can be directed simultaneously to more than one terminal.
- Direct transmission of messages between application programs and terminals. It makes the communication lines and the Communications Adapter transparent to the application program.
- The sharing of network resources among the various applications, which results in a more efficient use of lines, terminals, and the Communications Adapter.
- Capability of being connected through an SDLC cross-domain link to other processors that have installed one of the following IBM program products:

— ACF/VTAME
— ACF/VTAM with the Multisystem Networking Feature (MSNF)
— ACF/TCAM with the Multisystem Networking Feature (MSNF)

This allows application programs and terminals that are controlled by ACF/VTAME to communicate with application programs and terminals in another domain of the particular multiple system network.

- An installation's network operator can monitor and control the data communications network through the IBM Network Operation Support Program (NOSP) and Network Communication Control Facility (NCCF) program products.

ACF/VTAME provides, for DOS/VSE users, many potential advantages:

— More flexibility in application processing and in accessing information between multiple system networks and the associated operating systems.
— Expansion of the scope of network facilities (such as application programs and terminals) that are available for communication of data and for processing.
— Continued operation of communication controllers and associated terminals (or devices) independently of a specific central processor.
— A wide range of multiple system configurations that support orderly applications growth under the system network architecture (SNA).
— Elimination of redundant network applications.
— Consolidation of communications management functions (such as terminal ownership control, session establishment).

Advanced Communication Function/VTAM (ACF/VTAM)

Complex communication applications may require functional support which is not available with ACF/VTAME. ACF/VTAM, whose availability for DOS/VSE will be announced separately, may be the answer to problems of this kind. For information about the functions provided by ACF/VTAM and their usefulness at a DOS/VSE installation, consult these publications:

Advanced Communications Function for VTAM (ACF/VTAM) General Information:

Introduction, GC27-0462
Concepts, GC27-0463

Introduction describes the product in a general manner, names the programs and devices that can be used with it, and lists the steps that must be accomplished to install the product.

Concepts contains extensive descriptions of the ACF/VTAM facilities and describes the installation process in more detail. Both publications describe the entire ACF/VTAM documentation library.

Basic Telecommunication Access Method — Extended Support

This licensed support package (BTAM-ES) facilitates the coding of telecommunication applications in assembler language; it is the prerequisite support for a variety of IBM supplied licensed and nonlicensed programs.

Through BTAM-ES macros, this support dynamically builds the channel programs needed for the execution of I/O operations involving the telecommunication devices that are hooked up with your installation's central processor. BTAM-ES provides a convenient method of coding efficient telecommunication routines in a problem program; however, as opposed to ACF/VTAME, BTAM-ES requires the programmer to be familiar with the installation's line configuration. Moreover, a problem program using BTAM-ES may have to be adapted to an installation's line configuration should that configuration change.

BTAM-ES is discussed in more detail in *BTAM-ES General Information,* GC38-0292.

Data Language I (DL/I)

DL/I is a licensed data management control system that executes as an application program in a virtual environment of DOS/VSE. DL/I satisfies many diverse data processing requirements. It simplifies the task of creating and maintaining large common data bases that are accessed by user-written application programs. DL/I allows growth from a batch-only processing environment to a telecommunication environment such as CICS/VS.

For more details on the DL/I library, consult the publications *Data Language/I Disk Operating System/Virtual Storage (DL/I DOS/VS) General Information,* GH20-1246.

Customer Information Control System/VS (CICS/VS)

CICS/VS, a general purpose data base/data communication (DB/DC) control system, controls online DB/DC applications. CICS/VS, a licensed program product, includes

- most of the standard functions needed by application programs for communicating with remote and local terminals and subsystems.
- controls for concurrently executing application programs that serve many online users.
- data base access capabilities.

Furthermore, data used by online applications may, in general, also be used by non-interactive programs that are being executed in other partitions.

CICS/VS can be tailored to the needs of most combinations of concurrent online applications, serving a network that consists of a wide variety of terminals and subsystems.

For further information about this licensed program, consult the publication *Customer Information Control System/Virtual Storage (CICS/VS) General Information,* GC33-0066.

Subsystem Support Services (SSS)

SSS, a nonlicensed support package, is an installation and service program for the following industry subsystems:

- IBM 3600 Finance Communication System
- IBM 3650 Retail Store System
- IBM 3660 Supermarket System
- IBM 3790 Communication System

Each of these subsystems has a control and data collection unit, called a subsystem controller, that contains control information necessary for the operation of all terminals and components attached to it.

SSS, which is available as a separately shipped DOS/VSE component, can be used to place this control information on the disk file that resides in the subsystem controller; the program provides means for servicing that data. SSS accomplishes this by maintaining, at the host processor, a library of control information required to operate the industry subsystem. This library contains user-coded application programs and user-defined control records that define the operational environment for each subsystem controller.

An operational environment is based on the configuration of all devices within the subsystem, and on the various industry-related or application-related options that are available with each industry system.

By maintaining a central library at the host processor, SSS provides the capability for installing and updating several subsystems through a single control facility. This capability is in the form of control statements, which may be used to create the subsystem library, make modifications to it, and transmit selected portions of its contents to the subsystem controllers. In addition, SSS control statements may be used to obtain printouts of selected portions of the subsystem library, for use in monitoring subsystem activity.

The facilities of SSS and instructions how to install the component are documented in *IBM System/370 Subsystem Support Services User's Guide,* GC30-3022.

DOS/VS Sort/Merge

This licensed program product enables you to sort multiple files of logical records into a predetermined sequence, or to merge files of previously sequenced records.

Besides giving improved performance in virtual mode over DOS Sort Merge, this program offers a number of additional functions. These include:

- Support of the full range of DOS/VSE supported magnetic tape and disk devices for input, output, and work files.
- Support of SAM files on magnetic tape or disk and VSAM files on disk for both input and output.
- Support of a variety of user-exits that allow the execution of user-written routines under specific conditions during a sort/merge run.
- Ability to execute an analysis run which ends after the program has finished analysing the input control stream and issuing appropriate messages.
- New control statements for
  - — specifying a selection of records to be included in the sort/merge
  - — specifying reformatting of records
  - — requesting a summary of records
  - — specifying a user-defined collating sequence.

The DOS/VS Sort/Merge program can be invoked as a problem program executing under DOS/VSE or through a call from within a program written in assembler language, COBOL, PL/I, or RPG II.

For more details on this program and the manuals available for it, consult *DOS/VS Sort/Merge General Information,* GC33-4030.

VSE/IBM System/3 — 3340 Data Import

The program, which is available as a licensed program product, reads files written into the main data area of a 3348 data module by a System/3 Model 12 or 15 and converts these files to DOS/VSE files. You may choose to have the converted files written onto a data module of the same or other type or onto a disk pack on another device, except on a 2311.

Any number of System/3 files may be converted during one run of the program, but each file requires its own set of utility control statements. If files in a 5444 simulation area of a 3348 data module are to be converted, those files must be copied into the data module's main data area prior to the conversion run. You can use the System/3 $COPY utility program for that purpose.

The program is available for use on a 4331 processor.

IBM Systems 1401/1440/1460 Emulator Program

This emulator, a licensed program product, allows programs written for execution on a 1401 (or a 1440 or 1460) to be executed on a central processor with ECPS:VSE. This emulator may be operated under DOS Release 26 or 27, under a DOS/VS release, or under DOS/VSE.

A program running under this emulator can be executed in any partition of a multiprogramming DOS, DOS/VS or DOS/VSE without affecting the processing in the other partitions. More than one program, but only one per partition, may be emulated at any one time.

If an emulated program accesses data stored on disk or magnetic tape, the following requirements must be met:

— Disk files must be converted to CKD or to FBA format, unless the CS format option (for CS30 or CS40) has been selected for emulator generation.

— Files on 7-track tapes can be processed only with the data converter feature turned on. Data on mixed-density tapes cannot be processed.

1401/1440/1460 DOS/VS Emulator on System/370

This emulator, a nonlicensed support package, combines with the 1401/1440/1460 compatibility feature to allow programs written for execution on a 1401 (or a 1440 or 1460) to be executed on a System/370 under control of DOS/VSE.

A program running under this emulator can be executed in any partition of a multiprogramming DOS/VSE without affecting the processing in the other partitions. More than one program, but only one per partition, may be emulated at any one time.

Additional emulation considerations:

— Disk files must be converted before they are used by the emulator, unless the CS30 or CS40 compatibility option has been selected.

— Tape programs may be in original or converted format. The DOS spanned variable record (VRE) format is set as standard for the System/370 emulators.

For more information about this emulator, see the publication *1401/1440/1460 DOS/VS Emulator on System/370,* GC33-5384.

VSE/Data Interfile Transfer, Testing and Operations Utility (VSE/DITTO)

This program provides file to file services for card I/O, magnetic tape, and disk devices. VSE/DITTO, a licensed program product, allows for files or portions of files to be created, listed, copied, and altered. The program's operational flexibility makes it a useful tool in a program testing environment; it reduces the need for separate special purpose utility programs and contributes to greater operational productivity.

For more details on this program and its documentation, refer to *VSE/Data Interfile Transfer, Testing and Operations Utility for General Information,* GH19-6072.

DOS/VS RPG II Compiler

The DOS/VS RPG II compiler, a licensed program product, offers significant enhancements over the previously available DOS RPG and DOS RPG II compilers. Some of the enhancements are highlighted below:

- Performance improvements over the DOS RPG compiler in two areas:
  - — Storage efficiency for object programs.
  - — Throughput for processor bound programs.

- Data communication (DC) support through an interface to CICS/VS. Application programs written in RPG II may now run as transactions under CICS/VS. The support includes:
  - — A command language translator, which is provided as part of Release 4 of CICS/VS Version 1. This translator converts to CALL and PARM statements the online requests that are made using fixed-format EXEC and RQDL1 commands for CICS/VS or DL/1 DOS/VS, respectively.
  - — Exceptional condition handling.
  - — Access to the EXEC interface block (EIB) that is used to pass information from CICS/VS to the application program.
  - — DC file definition capability for standard RPG II data transfer.
  - — Non-standard RPG II (native CICS/VS) data transfer.
  - — The CICS/VS execution (command level) diagnostic facility.

- Data base (DB) support through an interface to DL/1 DOS/VS in a batch environment and also under CICS/VS. The support includes:
  - — A translator, which is provided as part of Release 3 of DOS/VS RPG II. This translator converts to CALL and PARM statements the batch requests that are made using fixed-format RQDL1 commands.
  - — DB file definition capability for standard RPG II data transfer.
  - — Non-standard RPG II data transfer.

- Online source input. Under DOS/VSE with VSE/ICCF or VM/CMS, source input may be entered into or changed in a work space via a display terminal in one of three modes: edit, input/alter, and prompt.

  In input/alter mode, the online source-input facility displays a template of the RPG II specification that is about to be entered; it performs a syntax check of the entered values. In prompt mode, the facility prompts you, field by field of the particular RPG II specification, for the applicable values.

  The source input, when finished, is to be filed in one of the available libraries in order to be available for a subsequent compilation. The output of the compilation consists of the normal list output, error messages that are directed to the terminal from which the compile run was invoked, and (optionally) an object module in card image format.

Under VM/CMS, the compiler provides for a CALL interface to that system's end user display support (EUDS) or to its shared file manager (SFM) support.

- Support of a defined set of VSAM functions.
- Integration of the Auto Report facility.
- System/3 RPG II equivalent functions:
  - — No need to specify input array decimal positions.
  - — TIME operation code to access the system time of day.
  - — PRINT operation code to have the 2560 print the contents of punched fields.
  - — Added device independence.
- Execution of DOS/VS RPG II programs in hierarchies of RPG II, COBOL, PL/I, and/or assembler programs.
- Revised DOS/VS RPG II cycle. File description, input, and output specifications as well as primary files are no longer mandatory. In addition, the revised cycle allows your calculations to exit from a DOS/VS RPG II subprogram at any point in the calculations.
- Data structures. Data structures, sets of named areas in virtual storage, increase the flexibility in defining, using, and redefining subfields within DOS/VS RPG II programs.
- Figurative constants. The figurative constants *BLANK, *BLANKS, *ZERO, and *ZEROS may be used to clear a data structure with blanks or zeros, respectively, in a single operation.
- Symbolic dump. When a DUMP or DUMPF statement is given, you receive a printout of the contents of fields, data structures, arrays, and tables in the appropriate form (as a numeric value or a character string, whichever applies). This printout includes the names of indicators that are ON.
- Error handling. To enable you to control object time error handling (and recovery, if possible), error and exception information is made available as follows:
  - — In a program status data structure for program errors.
  - — In one or more information data structures for file errors.

  For program errors, you may specify an error subroutine which is to be executed whenever a program error occurs.

For more details on this licensed program and the manuals available for it, consult the publication *DOS/VS RPG II General Information*, GC33-6030.

DOS/VS COBOL Compiler

This licensed program product compiles source programs written in the ANS COBOL language; it is designed for use under DOS/VSE. The

compiler contains all the functions of the DOS COBOL compiler, Version 3, and includes additional support as follows:

- Support of VSAM functions available with DOS/VS Release 28.
- Device support also for devices that are supported by DOS/VSE but not by DOS.
- The FIPS flagger, which identifies areas of a user's program that do not conform to the Federal Information Processing Standard.

Note: For COBOL source programs to be in a form suitable for this COBOL compiler, they can be converted by using the *COBOL-to-American National Standard COBOL Language Conversion* Program. However, some direct programming may still be required to accomplish full conversion. The amount of this programming varies with each application program.

For more information about this compiler and the manuals available for it, consult the publication *DOS/VS COBOL, General Information,* GC28-6473.

PL/I Optimizing Compiler

Pl/I is a general purpose programming language for both commercial and scientific programs. It is particularly useful when one application requires the handling of commercial and scientific problems in one and the same program, all of which are available as licensed program products.

PL/I support under DOS/VSE is provided by the PL/I Optimizing Compiler and by two libraries, the resident and transient libraries, all of which are available as licensed program products.

the compiler

The PL/I Optimizing Compiler is designed to provide optimized object programs from a comprehensive level of PL/I. It provides, besides a high level of PL/I language, diagnostics at both compile-time and object-time.

If optimization is specified, the compiler processes the PL/I source program, reorganizing it, if necessary, so as to produce an efficient object program. If optimization is not specified, the compiler requires less compile time.

A facility is provided by the compiler to allow communication between PL/I modules produced by certain FORTRAN, RPG, and COBOL compilers.

The language level implemented by the compiler contains extensions beyond the PL/I D language level.

Source programs which were written for the PL/1 D compiler can be compiled by the new compiler provided that those programs use valid PL/I language.

the libraries

Two libraries are required for the execution of programs compiled by the optimizing compiler. These libraries contain subroutines which must be combined with the object module to produce an executable program (the PL/I resident library), and other subroutines which are required dynamically as the program is being executed (the PL/I transient library).

Both the resident and the transient libraries are separate licensed programs.

For more information about the PL/I Optimizing Compiler and the manuals available for it, consult the *PL/I Optimizer, Resident Library and Transient Library, General Information,* GC33-0004.

FORTRAN IV Library, Option 1

FORTRAN is a programming language designed for the solution of scientific and computational problems. For users of FORTRAN, the DOS FORTRAN IV compiler is available as a nonlicensed IBM-supplied Type I program.

The FORTRAN IV Library, Option 1, is available as a licensed program; together with the Type I DOS FORTRAN IV compiler, the library allows the programmer to write and have compiled FORTRAN programs that:

- access files on DASDs that have been designed for attachment to System/370 central processors.
- create and process magnetic tape files which conform to the American National Standard Code for Information Interchange (ASCII).
- use larger block sizes for EBCDIC tape records.

Appendix A: Device Support

Minimum Machine Requirements

1. A DOS/VSE supported central processor with a minimum processor storage of 160K bytes.
2. Either of the following:
   - an IBM card reader and an IBM card punch (as two separate I/O devices or combined in one device).
   - an IBM diskette I/O unit.
3. An IBM printer.
4. Either of the following:
   - two disk drives, one of which must accommodate a removable disk pack.
   - one disk drive and one magnetic tape drive.

None of the disk drives must be an IBM 2311 because this DASD type is not supported as a DOS/VSE system residence device.

Supported Devices

Any IBM supplied peripheral equipment which is attachable via a DOS/VSE supported control unit attached to an I/O channel is supported by DOS/VSE through this control unit.

Central Processing Units

For available processor storage sizes less than maximum, refer either to *IBM System/370 System Summary,* GA22-7001, or to *IBM 4300 Processors Summary Input/Output and Data Communication Configurator,* GA33-1523, or else contact your IBM sales representative or the nearest IBM branch office.

Processor Model	Max. storage in K bytes	No. of bytes	Processor Model	Max. storage in K bytes	No. of bytes
3031	6,144	6,291456	3145**	1,024	1,048,576
3115-0*	192	196,608	3145-3	1,984	2,031,616
3115-2*	384	393,216	3148	2,048	2,097,152
3125-0	256	262,144	3155-II	2,048	2,097,152
3125-2	512	524,288	3158	6,144	6,291,456
3135**	512	524,288	4331 ‡	1,024	1,048,576
3135-3	512	524,288	4341 ‡	4,096	4,194,304
3138	1,024	1,048,576			

* Must have a processor storage of 160 K bytes or more.
** Requires the optional CPU timer and clock comparator in order to establish the specified operating environment.
‡ Central processors with ECPS:VSE

Direct Access Storage Devices

IBM device type	Device name	Remarks
2311	Disk Storage Drive	See Note 1
2314	Direct Access Storage Facility	
2319	Disk Storage	See Note 2
3310	Direct Access Storage Device	A fixed-block architecture (FBA) device (see Note 5)
3330	Disk Storage	
3333	Disk Storage	
3340	Disk Storage	
3344	Direct Access Storage	See Note 3
3350	Direct Access Storage	See Note 4
3370	Direct Access Storage Device	A fixed-block architecture (FBA) device (see Note 5)

Note 1: Not supported as system residence device.

Note 2: DOS/VSE supports this device as a 2314.

Note 3: DOS/VSE supports this device as a 3340 with one head/disk assembly of the 3344 simulating four 3340s with 3348 Model 70 data modules mounted.

Note 4: DOS/VSE supports this device also in 3330-1 compatibility mode. In that case, one non-removable head/disk assembly is equivalent to two 3330-1 volumes.

Note 5: Support of this device is available in 370 mode on a processor with ECPS:VSE only if your DOS/VSE includes VSE/Advanced Functions and was generated with VM/370 linkage enhancements for operation under VM/370.

Magnetic Tape Units

IBM device type	Device name	Remarks
2401	Magnetic Tape Unit	
2415	Magnetic Tape Unit	
2420	Magnetic Tape Unit	
3410	Magnetic Tape Unit	
3411	Magnetic Tape Unit and Control	For the DOS/VSE support, this is identical with the 3410.
3420	Magnetic Tape Unit	
8809	Magnetic Tape Unit	

Punched Card Devices

Model information for the various device types is provided only if this information is of any significance in the given context.

IBM device type	model	Device name	Remarks
1442	N1	Card Read Punch	
	N2	Card Punch	
2501		Card Reader	
2520	B1	Card Read Punch	
2520	B2/B3	Card Punch	
2540		Card Read Punch	
2560		Multifunction Card Machine	
2596		Card Read Punch	SYSIPT, SYSRDR, and SYSPCH cannot be assigned to this device.
3504		Card Reader	
3505		Card Reader	
3525		Card Punch	
5424		Multifunction Card Unit	
5425		Multifunction Card Unit	

Printers

IBM device type	Device name	Remarks
1403	Printer	
1443	Printer	
2245	Printer	Kanji — programs using kanji output cannot be executed in virtual mode.
3203	Printer	Models 4 and 5 are also supported as PRT1 printers
3211	Printer	support as a PRT1 printer
3213	Console Printer	For use with the console unit of a 3158
3284	Printer	supported via a control unit as a 3277
3286	Printer	supported via a control unit as a 3277
3287	Printer	supported via a control unit as a 3277
3288	Line Printer	supported via a control unit as a 3277
3289	Line Printer	supported via a control unit as a 3277; Model 4 is supported as a PRT1 printer.
3800	Printing subsystem	The programming support must be ordered separately
5203	Printer	

Paper Tape Equipment

IBM device type	Device name	Remarks
1017	Paper Tape Reader	
1018	Paper Tape Punch	
2671	Paper Tape Reader	

Manual Controls

IBM device type	Device name	Remarks
3210	Console Printer-Keyboard	
3215	Console Printer-Keyboard	
3277	Operator Display Console	
3278 Mod.2A	Display Console	

Optical and Magnetic Character Equipment

IBM device type	Device name	Remarks
1255	Magnetic Character Reader	See Note 1
1259	Magnetic Character Reader	See Note 1
1270	Optical Reader/Sorter	See Note 1
1275	Optical Reader/Sorter	See Note 1
1287	Optical Reader	
1288	Optical Page Reader	
1419	Magnetic Character Reader	
3881	Optical Mark Reader	
3886	Optical Character Reader	
3895	Reader/Inscriber	
Note 1: DOS/VSE support for the device is the same as for the 1419 Magnetic Character Reader.		

Terminal Devices

DOS/VSE supports the full range of terminals that may communicate with one of the following control units:

IBM device type	Device name	Remarks
LCA2	3791 Local Attachment Cluster Control	
2701	Data Adapter Unit	may be integrated in central processor (see Notes 1 and 3).
2702	Transmission Control Unit	See Note 3.
2703	Transmission Control Unit	may be integrated in central processor (see Notes 2 and 3).
3704	Communication Controller	See Note 3.
3705	Communication Controller	See Note 3.
3791L	Local Communication Controller	See Note 3.
Note 1: Is integrated, for example, in System/370 Model 135. **Note 2:** Is integrated, for example, in System/370 Models 115 and 125. Support for the 2703 includes the support for a 3704/3705 Communication Controller in emulation mode. **Note 3:** For detailed information about terminal support through these control units, refer to the IBM System Library publications describing the control units, or else contact your IBM sales representative or the nearest IBM branch office.		

Locally Attachable Display Devices

IBM device type	Device name	Remarks
2260	Display Station	
3277	Display Station	

Miscellaneous Peripheral Equipment

IBM device type	Device name	Remarks
3540	Diskette Input/Output Unit	Supported by DOS/VSE as unit record I/O devices.
none	Diskette 2D drive	
7770	Audio Response Unit	

Appendix B: New in Recent DOS/VS Releases and in DOS/VSE

This appendix highlights facilities and feature support that has become available with the Releases 33 and 34 of the DOS/VS SCP and the DOS/VSE SCP. In this appendix, a facility (or feature support) included in a DOS/VS release and, if this was Release 33, expanded in Release 34 of DOS/VS or in the DOS/VSE SCP is discussed only once: as part of the release that included the expansion. Added device support is addressed only in the discussion of new support in DOS/VSE; for a complete list of devices supported by DOS/VSE refer to *Appendix A*.

New in the DOS/VS Release 33 SCP

Cross-partition event control
This facility, useful primarily for the implementation of complex applications, allows a user to set and reset cross-partition event control blocks (XECBs) and to conditionally remove a task from the wait state. For more information about cross-partition event control, refer to *DOS/VSE Macro Reference.*

Subtask priority modification
The available CHAP macro allows a subtask to request its own processing priority to be set to the lowest priority of all subtasks within the same partition.

Task timer support
The main task of the partition owning the task timer now can request that in an exit routine receives control after this task has been in control of the central processor for a specified period of time. The task timer requires, in the case of System/370 Models 135 and 145, the installation of the clock comparator and the CPU timer (an optional hardware feature).

Interval timer extension
A new parameter of the SETTIME macro allows for the specification of a smaller (1/300 of a second) interval timer stepping unit.

The IBM copy file and maintain object module (OBJMAINT) utility program
This IBM multi-purpose utility program is of particular interest in conjunction with cardless system operation for the purpose of:

— File-to-file copying, including blocking and deblocking, of card image files.
— Comprehensive listing of card image files including object programs.

For more information on this program, refer to *DOS/VSE System Utilities.*

PDZAP with logging
The CIL Patch Program (PDZAP) now includes a logging feature. The log produced by PDZAP contains information relating to changes made to

core image library phases. The information written to SYSLST consists of:

— Date and time of change, and identification of initiator.
— Name and load address of the phase changed.
— Old and new data in hexadecimal notation.

For further information on this feature, please refer to *DOS/VSE Serviceability Aids and Debugging Procedures.*

Prelinked IBM system components in the core image library
IBM system components are prelinked into the core image library to eliminate a time consuming step when installing a new release of DOS/VSE. For the deletion of unwanted components, new delete procedures are provided.

Pre-assembled versions of I/O modules
In addition to the standard IBM-supplied IOCS, pre-assembled versions of all I/O modules required for RPG II and the PL/I Optimizing Compiler are provided in the relocatable library.

Print-buffer-image modules in the relocatable library
The relocatable library as shipped includes the IBM provided standard buffer-image modules for the IBM 1403 N1 printer.

Identification of DOS/VSE supervisors
DOS/VSE supervisors include an identification to show which release level of DOS/VSE is applicable for a given supervisor. Furthermore, a new parameter (USERID) in the FOPT macro allows you to distinguish between different supervisors in the system. These identifiers eliminate ambiguities. They are listed on the console during IPL or can be derived from a supervisor dump.

The IBM Backup and Restore Utility programs
The two utility programs are of particular interest during system generation and for library service. For a detailed description of the programs, refer to *DOS/VSE System Utilities.*

The Backup program can be used to create a backup copy on tape of DOS/VSE system and private libraries. The Restore program restores the backup copy from tape to disk; it is also used to restore the DOS/VSE distribution tape to disk prior to system generation.

The Backup and Restore programs can be used together to efficiently condense libraries. There are no condense restrictions for multipogramming environments. Backup and Restore used together also facilitates the migration of libraries from one type of IBM disk device to another. A change in the DASD type used for SYSRES does not require a re-generation of the system from the IBM-supplied distribution tape, because the backup copy of the current SYSRES can be restored to a different DASD type.

Operational improvements

— The printing of relocatable dictionary listings can be suppressed by specifying the NOLIST option of the job control OPTION statement.
— The $JOBACCT dummy phase can now be loaded into the shared virtual area (SVA). This improves the performance of installations that use the job accounting option of the licensed VSE/POWER.
— The message 'AR READY FOR COMMUNICATION', formerly issued by DOS/VS after the operator pressed the Request key, is suppressed. The system enters read mode immediately after 'AR' is printed.
— The operator response 'IGNORE' to message 'INTERVENTION REQUIRED' is no longer necessary.
— All of a partitions virtual storage not occupied by the problem program is now available via the GETVIS/FREEVIS macro instructions.

Optional escape from abnormal termination

The new supervisor macro function EXIT AB provides an alternative to the function of the existing EXIT macro. The EXIT AB macro, if included in the Abnormal Termination User Exit Routine, returns control to the supervisor to allow continued processing of the problem program (main task) rather than its termination. For further information, refer to *DOS/VSE Macro Reference.*

Improved MAP command output

The upper limit addresses of the storage areas displayed by the MAP command are shown in hexadecimal instead of decimal notation. These addresses can therefore be used in commands (DUMP, for example).

Printout of the system directory list (SDL)

The new DSERV parameter DSPLY SDL can be used to obtain a printout of the SDL contents. Also, the parameter DSPLY(S) ALL is extended to include the SDL as well.

New in DOS/VS Release 34 SCP

Using the IBM 3277 Model 2 as display operator console

DOS/VSE includes support for using the IBM 3277 Model 2 Display Station as display operator console on all DOS/VSE supported central processors.

The screen of the 3277 Model 2 has 24 lines, 20 of which are used as message area. The remaining four lines are used as instruction line, entry area (2 lines), and warning line. The full length of 80 characters is supported.

Extended support for the IBM 3330-11 and 3350

Support is provided for the IBM 3330-11/3333-11 Disk Storage and the IBM Direct Access Storage. ISAM, however, does not support these devices; IBM recommends to convert existing ISAM files to VSAM files and to access the converted data via the ISAM Interface Program, which is available as part of VSE/VSAM.

An existing program can access files on the 3330-11/3333-11 and 3350 without modification if all of the following prerequisites are met:

— The supervisor supports RPS.
— Sufficient SVA space is available to load RPS logic modules.
— Sufficient space in the partition is available for the required RPS DTF extension(s).
— The program runs in virtual mode.

IPL communication device list
DOS/VSE allows to store a restrictive list of communication devices for use during initial program load (IPL). Transmission of IPL commands, then, can occur only from one of these devices. This is of interest for telecommunication installations and for installations with locally attached terminals such as IBM 2260, 2741, and 3277. During IPL, DOS/VSE checks whether the device presenting an interrupt is included in this list and rejects the interrupt if the device is not included.

Up to eight devices can be specified to make up this pool. The operator's console (SYSLOG device) must be included in the pool along with terminal devices, card reader and diskette I/O units at the user's discretion. The list of specified devices has to be linked to the system's core image library. Use of an IPL communication device list is optional.

For more information on this new DOS/VSE feature, refer to *DOS/VSE System Mangement Guide.*

IBM 3540 as IPL communication device
The IBM 3540 diskette I/O unit is now supported as an IPL communication device and may be used to submit IPL commands. The IPL commands must be written, one command per sector and in card image format, onto a single-volume diskette file.

New parameter in the DLBL statement
BLKSIZE, a new parameter in the DLBL job control statement, provides support for dynamically changing the blocking factor for a sequential disk file on an IBM 3350 or IBM 3330-11. When transferring files to such devices, it is not necessary to change the blocksize specifications in the problem program.

COPYSERV functions now in CORGZ
The CORGZ program now includes the library directories and identifies to CORGZ library elements that are missing in the target library. The added functions in the CORGZ program are invoked by the parameter NEW of the COPYx statement.

Job accounting impovements
The job accounting support has been improved in two ways:

— The date in the job accounting record of a job step will now always correspond to the date at the end of the job step. (Up to now, the date in the job accounting record for the last or only job step was that of the beginning of the job step.)
— The job accounting support will no longer be canceled if the job accounting routine is canceled.

Note: IBM ensures proper execution of the functions newly implemented in the DOS/VSE SCP only in the specified operating environment: with VSE/Advanced Functions installed.

Support of central processors with ECPS:VSE
DOS/VSE takes full advantage of extended control program support for DOS/VSE (ECPS:VSE) if generated accordingly. On a central processor with ECPS:VSE, you can operate DOS/VSE in modes as follows:

— System/370 mode, which requires dynamic translation of virtual addresses into real (processor) storage addresses by interaction of DOS/VSE with the hardware.
— ECPS:VSE mode, which requires no dynamic translation of virtual addresses into processor storage addresses by DOS/VSE because this is done by the hardware.

On a System/370 processor, DOS/VSE can be operated only in System/370 mode.

Support of 3031 processors
The support of 3031 processors, up to now available only as an independent release, is fully integrated in DOS/VSE.

Pageable supervisor routines
Not all of the supervisor routines need be in processor storage all the time. Supervisor routines that have a relatively low usage rate and can tolerate a page-in delay when their services are required have been redesigned to be pageable. An IPL option is available for you to define whether these routines should reside in processor storage at all times (which results in a larger supervisor but, on the other hand, may improve system throughput) or whether they may be retained in the page data set and read into processor storage as required.

No longer a requirement of JIBs for DASD file protection
In the past, JIBs (job information blocks) were required to have DOS/VSE store data on and retrieve data from DASDs for which file protection support was included in the supervisor. This requirement no longer exists.

GETVIS space for programs running in real mode
Programs running in real mode may now request DOS/VSE (by means of a GETVIS macro) to dynamically allocate processor storage. You can make use of this new service of DOS/VSE if the pertinent program was written in assembler language and if an area for this kind of dynamic storage allocation is available to DOS/VSE. For more information about dynamic allocation of storage, see *DOS/VSE System Management Guide.*

Up to 241 programmer logical units per partition
For each partition, you may define up to 241 logical units (SYS000 to SYS240). This adds to the flexibility of DOS/VSE in the area of late binding of I/O devices.

Removal of the label cylinder bottleneck
The label information area on disk has been increased (on CKD devices from one to two cylinders — two to three for a 3340). In addition, the

concept of allocating a fixed disk space to each of the partitions has been changed; DOS/VSE now dynamically acquires space needed for storing a partition's labels. This new concept provides for an improved exploitation of the allocated label information area.

Improved method for loading the SVA
Phases required by DOS/VSE to be resident in the shared virtual area (SVA) are loaded into that area automatically during IPL. Phases required to be SVA resident for improved performance of your own programs may be loaded into the SVA any time after IPL. Once a phase has been loaded into the SVA, that phase remains available for execution from that area until next IPL.

The SVA warm-start facility available in the past has been removed. Reloading phases into the SVA after each IPL ensures that always the latest version of a phase is contained in the SVA.

Use of a higher-resolution timer
The timer support has been modified to depend on the time-of-day clock and the clock comparator, which have a higher time resolution than the interval timer on which this support was based in the past. The job accounting routines have been modified for the same reason; they use the CPU timer.

The above mentioned modifications result in improved time-interval control and in more accurate job accounting information.

Extended device support

- **Support of fixed-block architecture disk devices:** DASDs designed for storage of data as fixed blocks are fully supported by DOS/VSE in addition to the CKD devices that have been supported in the past. DOS/VSE components that must access fixed-block architecture (FBA) DASDs have been modified accordingly. Support of FBA DASDs permits the use of these devices for:

  — system residence.
  — private libraries.
  — system files on disk.
  — work files on disk.
  — storing (and retrieving) data by using the access method SAM (DTFSD or DTFDI) or by coding your own I/O requests on an EXCP level (direct access or indexed sequential access support for fixed-block access devices is available through the licensed access method VSE/VSAM).

  System utilities are available for the execution of service functions for these new DASDs as follows:

  — Fast copy disk — this utility is provided for CKD devices and also available in a stand-alone version on the DOS/VSE distribution tape.
  — Surface analysis — this utility is provided for FBA devices and available only in a stand-alone version on the DOS/VSE distribution tape.
  — Assign alternate block.

— Format emulated extent - the utility is also available in a stand-alone version on the DOS/VSE distribution tape. The program formats a specified area of the volume for emulation of 2311 or a 2314/19 on a 3310.

An existing program can access SAM files on an FBA device without modification, providing all of the following conditions are met:

— The supervisor includes support for FBA devices.
— Sufficient SVA space is available in the partition for the required DTF extension(s).
— Sufficient workspace is available for DOS/VSE to open the pertinent FBA file(s).

The support of FBA DASDs is available under DOS/VSE in System/370 mode on processors with ECPS:VSE if:

(1) VSE/Advanced Functions has been installed.
(2) The generated DOS/VSE includes the VM/370 Linkage Enhancements facility for operation under VM/370.

For more information about the fixed-block DASD I/O support, see *DOS/VSE Data Management Concepts.*

- **The IBM analysis program-1 (AP-1):** This program, which became available with Release 33 of DOS/VS in support of CKD DASDs with non removable disk packs has been extended to support also FBA DASDs.

  The program determines whether a data error on a direct access storage device is due to a faulty recording surface or due to a hardware error. The output of AP-1 enables the system operator or programmer to decide whether the data recovery procedure or the installation's procedures for hardware problems should be initiated.

  For detailed information on AP-1, refer to *OS/VS and DOS/VSE Analysis Program-1 (AP-1) User's Guide,* GC26-3855.

- **Automatic channel switching for DASDs:** The channel switching support that has been available for magnetic tape units has been extended to be available also for DASDs. If a DASD is attached to a processor via two channels, DOS/VSE automatically switches to the second channel, should the first one be busy. You can define a DASD as switchable during IPL.

- **DASD volume recognition:** An operator command is now available for requesting DOS/VSE to provide DASD volume-status information (in use or not in use) for one, a selected group, or all disk devices that are part of your DOS/VSE controlled installation. DOS/VSE does not provide status information on mounted diskettes.

- **2311 and 2314 support in the fast copy disk volume utility:** This extended device support in the fast copy disk volume utility program is provided primarily to facilitate dump and restore operations in an emulated environment on non-removable disk packs.

- **8809 magnetic tape unit:** DOS/VSE allows operation of this 9-track, 1600 BPI magnetic tape unit in either of the following modes:

  — high speed (called streaming) mode
  — low speed (called start-stop) mode

  both with long or short record gaps. The desired operating mode can be set during IPL and may be changed between jobs or job steps. The tape unit is fully compatible and interchangeable with IBM's current half-inch magnetic tape units when it is operated in start-stop mode.

- **3289 Printer Model 4:** The new printer is supported as a system printer of the PRT1 class.

- **3800 ICR integration:** Control code required for the operation of a 3800 printing subsystem is now fully integrated in DOS/VSE. The logical IOCS support for the 3800 must be ordered separately also in the future.

- **5424 Multifunction card unit:** The functional capabilities of this new card I/O unit are fully supported. Your coding requirements for utilizing this support are similar to those for utilizing the support that has been available for the 5425 multifunction card unit.

Suppression of the channel program scan
For a problem program starting an I/O operation on an EXCP level as has been coded in the past, DOS/VSE scans the associated channel program to determine which of the problem program's areas need be fixed in processor storage for this I/O operation. By using the new IORB macro instead of the CCB macro in the problem program, you can pass to DOS/VSE the beginning and end addresses of the areas that are to be fixed for the particular I/O operation. At the same time, this indicates to DOS/VSE that there is no need for a scan of the associated channel program.

Backup/Restore improvements
If the library file identifiers for the restored library are to be the same as for the original (backed up) library, these identifiers need not be redefined.

The Restore program uses an improved sorting algorithm for the sorting of the core image library directory if this sort operation need be executed.

The stand-alone version of the Restore program, if required, need no longer be punched into cards. The Backup program now writes the stand-alone Restore program onto the backup tape.

Improved storage dumps
The DOS/VSE dump routines have been changed to execute from the SVA rather than from the logical transient area, an area of the DOS/VSE supervisor.

If a system or partition dump is taken as a result of a job-canceled situation, the DOS/VSE dump routine is executed before any resources are freed by automatic action of DOS/VSE and before an abnormal-end user exit receives control, if there is one.

DOS/VSE includes a new stand-alone dump generation and print program (DOSVSDMP) that is capable of

— generating a stand-alone dump program for IPL from cards, magnetic tape, disk, or diskette.
— printing the output of a stand-alone dump program execution. This output may be printed as is or selectively.
— printing the output of an SDAID run from that program's output tape.
— printing the storage dump obtained as a result of the execution of a DUMP command. This output may be printed as is or selectively.

The generation of a stand-alone dump program and also the printing of dump listings are requested via the console. To facilitate the required operator-system interaction, the program DOSVSDMP displays menues and, on request, explanations for the use of the program.

For more information on the new storage dump support of DOS/VSE, see *DOS/VSE Serviceability Aids and Debugging Procedures.*

Extended tracing facilities
The tracing facilities of DOS/VSE, previously available in two separate programs (PDAID and SDAID), have been consolidated in one program (SDAID) and, at the same time, extended and improved significantly.

The most important extensions and improvements are:

— Up to eight different traces may be requested for one execution of SDAID.
— Trace information is provided formatted in order to facilitate evaluation of the recorded events.
— A variety of additional output options per traced event (for example the contents of registers, the contents of certain control blocks, or the contents of specific areas) may be requested.
— Unrestricted tracing of SDAID buffer overflow with complete output of buffer contents (no records get lost).
— The tracing of an event may be limited to a specified number of occurrences or to a specific area (a partition or an area defined by two addresses, for example).
— The range of a trace may be defined symbolically.

The program is now invoked by one simple attention routine command: SDAID. When the program receives control, you can enter the required control information in one of two modes: direct or prompt.

For more information about the SDAID program, see *DOS/VSE Serviceability Aids and Debugging Procedures.*

Simplified supervisor assembly
Various supervisor assembly options are no longer required and have been deleted; the corresponding functions have become standard support. Operational flexibility is achieved through IPL and job control options.

Maintain system history program
This is an efficient tool designed to facilitate the process of installing and servicing DOS/VSE at your installation. By automating prerequisite checking for the installation of features such as VSE/Advanced

Functions or for the application of cross component program temporary fixes, this program helps to speed up system service and upgrade activities at a DOS/VSE installation, which may well result in improved system availability.

Improved performance of the copy and reorganization (CORGZ) program
Copying library members from one library to another now takes significantly less time than in the past. The control statements needed for a CORGZ run remain unchanged.

Improved version of the copy service (COPYSERV) program
The program automates the process of comparing the directory of a current library with the directory of another current or a new library.

The output of the program consists of COPYx statements for library members which are not included in the library that is defined as the target library. These statements, which the program generates in alphabetic sequence of member names, may be used as input for a later CORGZ run if, for example, the libraries compared by the program are to be merged. The program produces, in addition to the COPYx statements, output on SYSLST as follows:

— A listing of the generated statements.
— The number of additional directories needed in the new library.
— The number of additional library blocks needed to accomodate the new library.

Train-cleaning utility program
Inclusion of this utility program in DOS/VSE allows the operator to clean the mounted print train (or chain) of a printer any time between jobs or job steps in the associated partition.

Common EREP for DOS/VSE and OS/VS
The environmental recording editing and printing program included in DOS/VSE is the same as the version of this program available at OS/VS installations. This may be useful at installations with one or more computing systems operating under OS/VS along with one or more computing systems operating under DOS/VSE.

Short XREF list for assembler output listings
A new operand of the // OPTION statement, SXREF, causes the assembler to suppress the printing of unreferenced labels in the label cross-reference list.

Long SETC value
The DOS/VSE assembler now allows a character string of up to 255 characters to be assigned to a SETC symbol.

Removal of programming support
The following programming support,previously available with DOS/VS Release 34, is no longer included in DOS/VSE:

— The queued telecommunication access method (QTAM) — ACF/VTAM,an IBM licensed programming support is available to provide similar telecommunication support under DOS/VSE.

— The System/360 Model 20 emulator on System/370 — Any programs originally written for execution on a /360 Model 20 and still in use should be reassembled or recompiled.
— IOCS functions (physical and logical) and utility functions for the 2321 Data Cell; IOCS functions for the 2495 Tape Cartridge Reader.
— The deblock utility — the OBJMAINT utility is available under DOS/VSE to handle blocking and deblocking at your installation.
— Seek separation. This function is now available for block multiplexer channels through the hardware.

Appendix C: DOS/VSE Documentation

A full set of manuals and educational courses is available to describe the DOS/VSE and its use.

For education courses and manuals answering the needs of both users new to data processing and users new only to DOS/VSE or some of its applications, consult your IBM representative or contact the IBM branch office serving your locality.

The DOS/VSE Library

The DOS/VSE library is a set of manuals describing the functions and uses of DOS/VSE and the operation of the system. The manuals of the library have been revised continually to cover new and extended support, as this support has become available, and to improve information retrievability. This Appendix provides a library structure overview which you can use also as a reading guide. Manuals in dashed-line boxes are not an immediate part of the DOS/VSE support documentation for using the system.

Types of Manuals in the Library

Wherever appropriate in the library, a distinction is made between several levels of information, each level serving a different purpose:

- **Descriptive Information:** Descriptive information is aimed at developing an understanding of a component or group of components and the part they play in an operational DOS/VSE. In developing a topic, a descriptive manual or section attempts to address practical implications by means of examples and careful explanation. Descriptive information is contained primarily in this publication and in publications that are called a 'Guide' or a 'Concepts' manual.

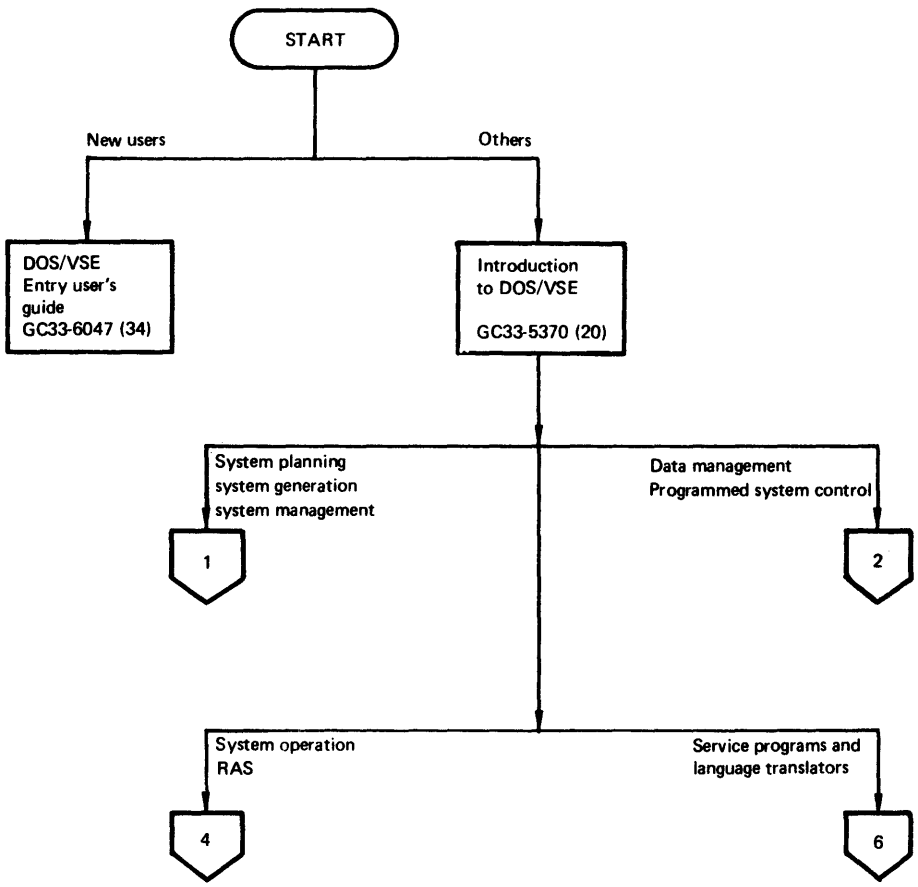
- **Reference Information:** This type of information represents the concise specifications for using the service of DOS/VSE; it is contained in manuals that are reference sources. Accompanying explanatory text is reduced to a minimum, allowing rapid retrievability of information. The *DOS/VSE System Management Guide,* for example, contains the necessary descriptive information about the available system functions while *DOS/VSE System Control Statements* is a quick-reference source for the corresponding control statements.

  In other instances, both descriptive and reference information may rightfully be contained within a single manual, one that fully covers a topic, such as *DOS/VSE Operating Procedures.*

- **Logic Information:** Logic manuals, in presenting the internal details of system programs and components, mix reference and descriptive/tutorial information. These manuals are available primarily for use by IBM programming support representatives. In the library structure overview, logic manuals are represented as dashed boxes.

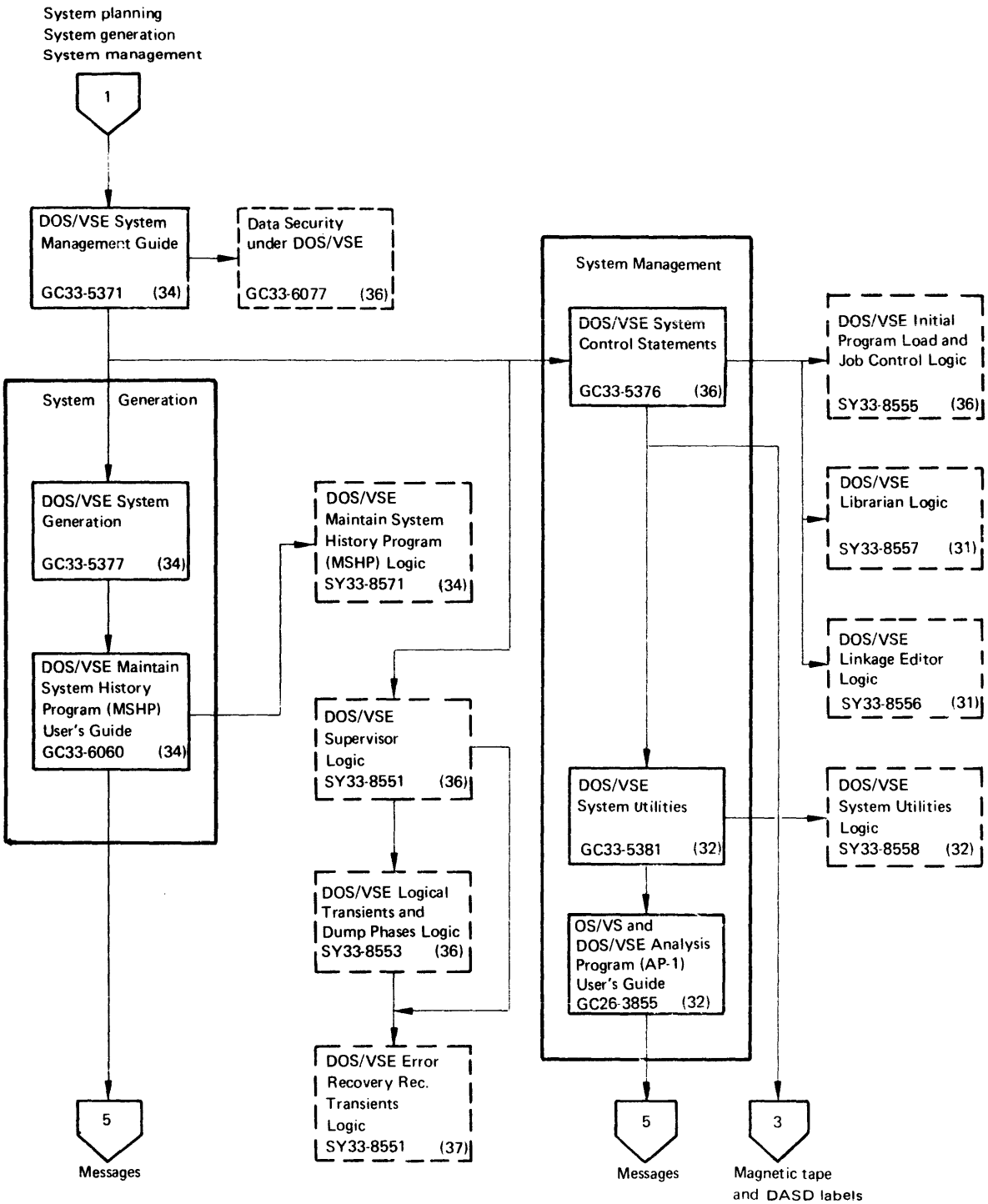
Manuals for Licensed IBM Programs

The *IBM System/370 Bibliography,* GC20-0001, lists the specific manuals available with each licensed IBM program. However, the structure overview includes only the *General Information* manuals of the licensed programs discussed in this section or, if no General Information manual is available, the order number of the program's key publication.

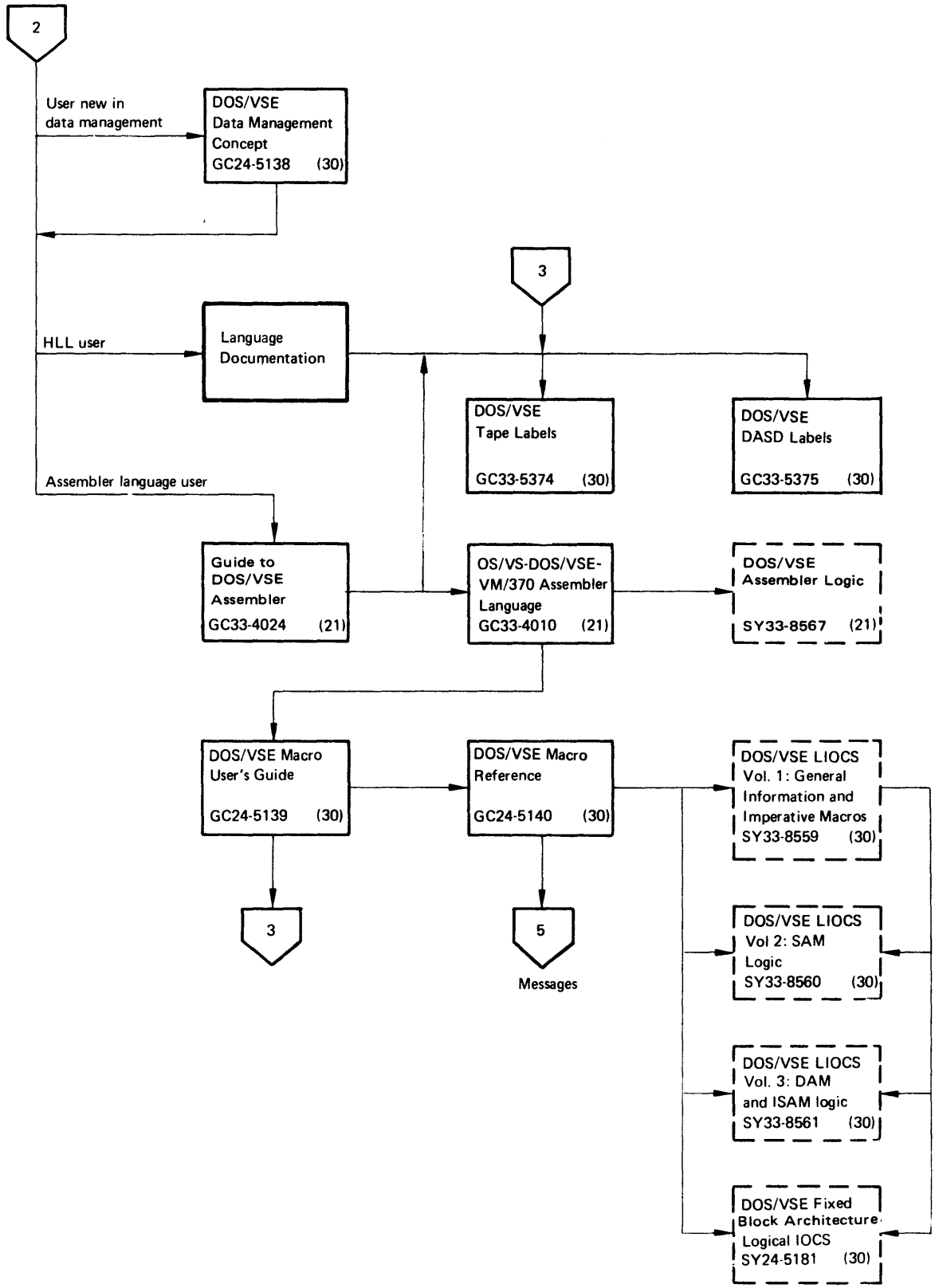


Note: For all manuals included in this library structure chart, the file number is given in parentheses in the lower right-hand corner of the particular box.

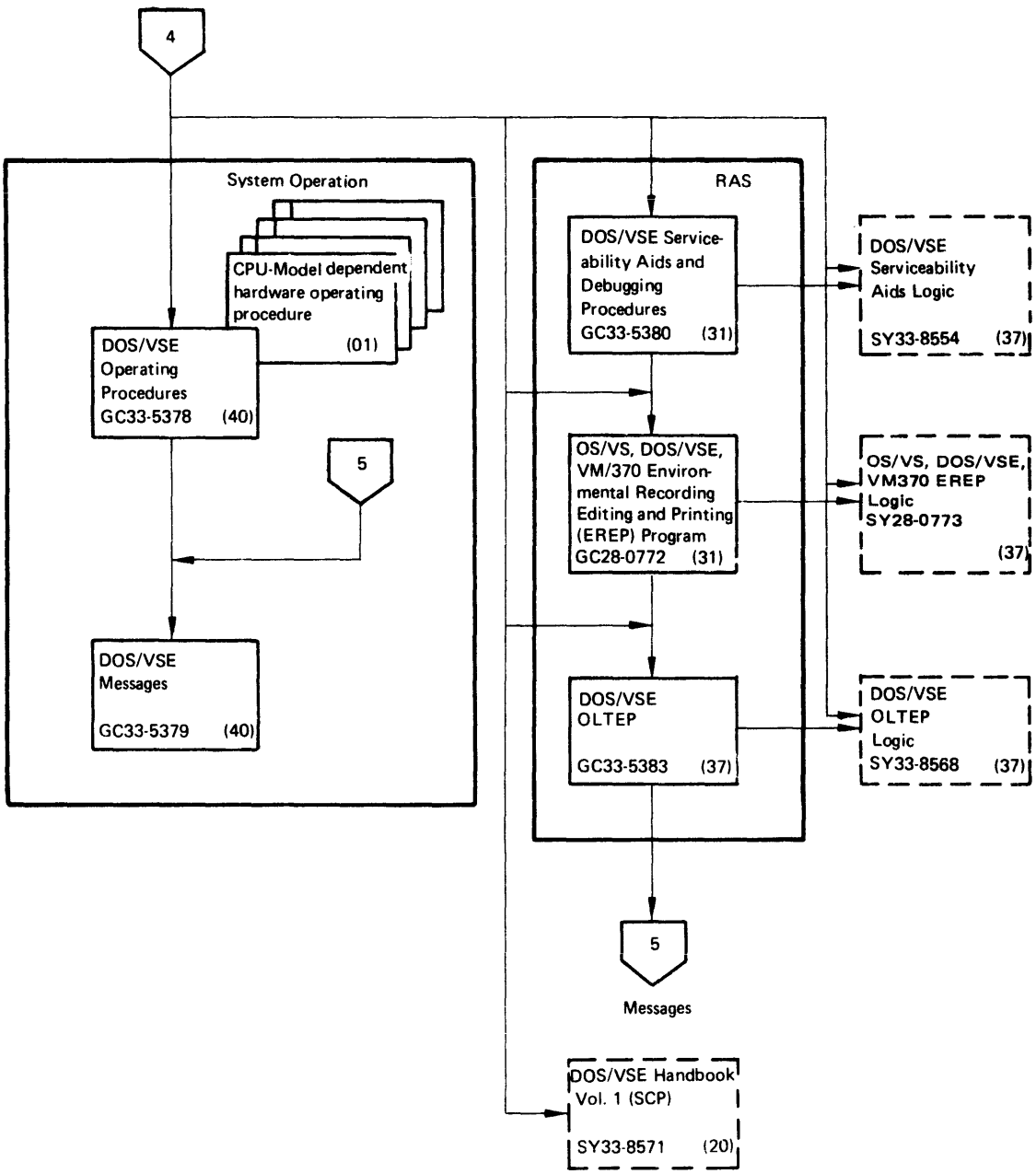
DOS/VSE library — structure overview and reading guide (Part 1 of 5)



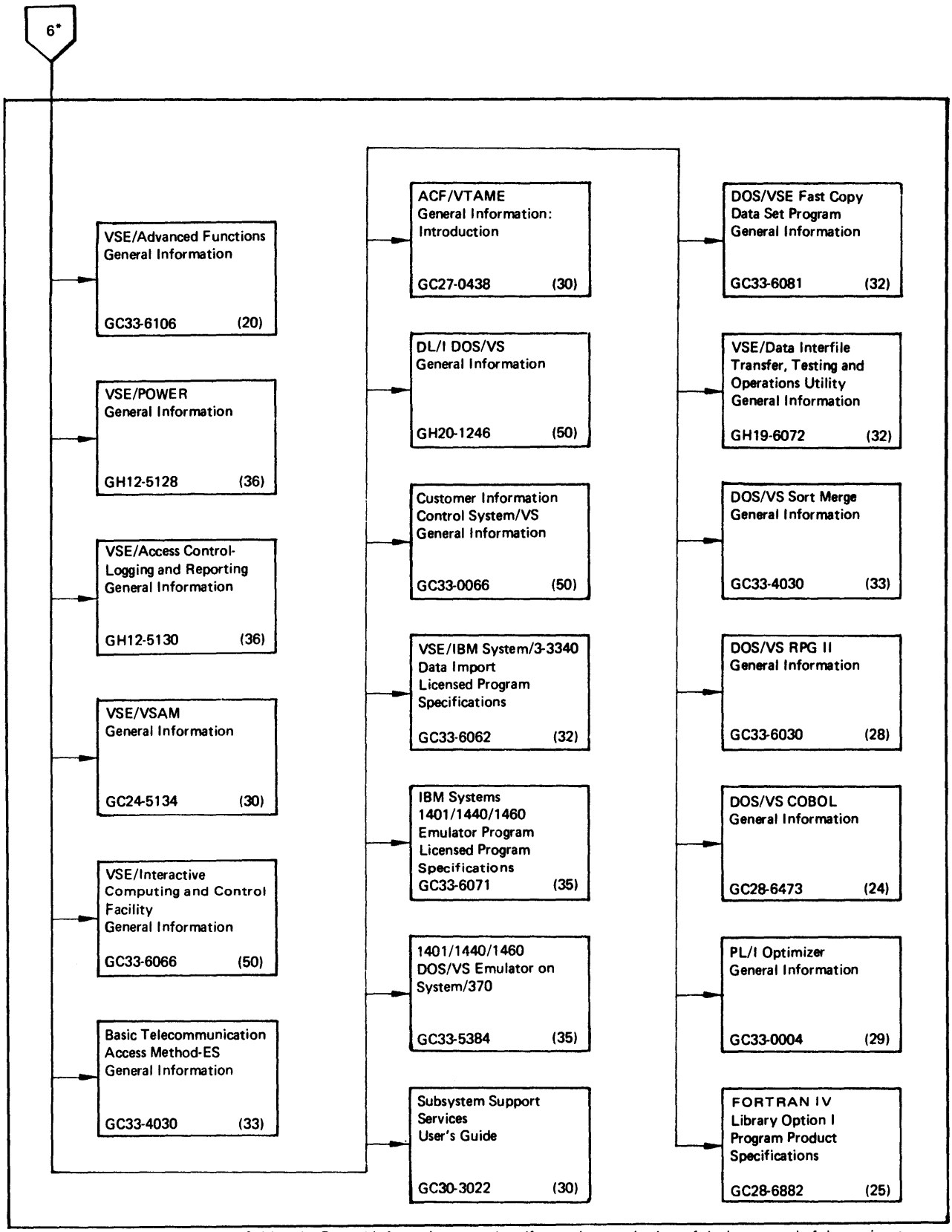
DOS/VSE library — structure overview and reading guide (Part 2 of 5)



DOS/VSE library — structure overview and reading guide (Part 3 of 5)



DOS/VSE library — structure overview and reading guide (Part 4 of 5)



* Shown here are the order number of either the General Information manual or, if no such manual exists, of the key manual of the pertinent licensed or unlicensed component.

DOS/VSE library — structure overview and reading guide (Part 5 of 5)

Glossary

This glossary includes definitions developed by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). This material is reproduced from the American National Dictionary for Information Processing, copyright 1977, by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018.

access method: A technique for moving data between virtual storage and input/output devices.

***address:** (1) An identification, as represented by a name, label, or number, for a register, location in storage, or any other data source or destination such as the location of a station in a communication network. (2) Loosely, any part of an instruction that specifies the location of an operand for the instruction.

alternate track: One of a number of tracks set aside on a disk pack for use as alternatives to any defective tracks found elsewhere on the disk pack.

application programs: A program written for or by a user that applies to his own work.

assembler language: A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

asynchronous operator communication: A DOS/VSE facility that allows the operator to defer the reply to a system message which requires a response.

attach: In data processing under DOS/VSE, to create a task and present it to DOS/VSE.

auxiliary storage: Data storage other than real storage; for example, storage on magnetic tape or disk. Synonymous with external storage, secondary storage.

batch processing: Sequential processing of computer programs, submitted to the computer as a collection (batch) of jobs that are separated from one another by job control statements.

binary synchronous communication (BSC): Data transmission via a telecommunication line in which synchronization of characters is controlled by timing signals generated at the sending and receiving stations.

blocking: Combining two or more logical records into one block.

book: A group of source statements written in any of the languages supported by DOS/VSE and stored in a source statement library.

buffer: An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area.

BSC: see binary synchronous communication.

byte: A sequence of eight adjacent binary digits that are operated upon as a unit and that constitute the smallest addressable unit of the system.

card punch: A device to record information in cards by punching holes in the cards to represent letters, digits, and special characters.

card reader: A device which senses and translates into machine code the holes in punched cards.

cardless system: A System configured without a card reader or card punch, but with an IBM diskette input/output unit.

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

central processor: *Synonym for central processing unit.

***central processing unit:** A unit of a computer that includes the circuits for controlling the interpretation and execution of instructions. Abbreviated CPU.

chaining: A method for storing records. When this method is used, each stored record has a link field which points to the next record in the processing chain. Physically, the records may be stored anywhere on the storage media.

channel: (1)* A path along which signals can be sent, for example, data channel, output channel. (2) A hardware device that connects the central processor and its associated storage with the I/O control units.

channel command word (CCW): A doubleword that directs a channel, control unit, or device to perorm an operation or a set of operations.

channel program: One or more channel command words that control a specific sequence of channel operations.

CKD: see count-key-data device.

CMS: conversational monitor system — a component of VM/370, an IBM supplied operating system.

communication line: Any physical link, such as a wire or a telephone circuit, that connects one or more remote terminals to a communication control unit, or connects one communication control unit with another.

compile: To prepare a machine language program from a computer program written in a high level language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

compiler: A program used to compile.

component: A functional part of DOS/VSE (for example: job control program).

configuration: The group of machines, devices, etc. which make up a data processing system.

control program: A program that is designed to schedule and supervise the performance of data processing work by a computing system. See also system control program.

control unit: A device that controls the reading, writing, or display of data at one or more input/output devies.

core image library: A library of phases that have been produced as output from link-editing. The phases in the core image library are in a format that is executable either directly or after processing by the relocating loader in the supervisor.

CPU: See central processing unit.

count-key-data (CKD) device: A disk storage device storing data in the format count field normally followed by a key field followed by the actual data of a record. The count field contains, among others, the address of the record in the format CCHHR (CC = cylinder number, HH = head number, R = record number) and the length of the data; the key field contains the record's key (search argument).

data base: A set of data that is sufficient for one or more purposes at one or more data processing installations.

data integrity: See integrity.

data management: A major function of DOS/VSE that involves organizing, storing, locating, retrieving, and maintaining data.

data security: See security.

deblocking: the action of making the first and each subsequent logical record of a block available for processing one record at a time.

default value: The choice among exclusive alternatives made by the system when no explicit choice is specified by the user.

diagnostic routine: A program that facilitates detection and isolation of malfunctions or mistakes.

dial line: See switched line.

dial-up terminal: A terminal on a switched teleprocessing line.

direct access: (1) Retrieval or storage of data by a reference to its location on a volume, other than relative to the previously retrieved or stored data. (2)* Pertaining to the process of obtaining data from, or placing data into, storage where the time required for such access is independent of the location of the data most recently obtained or placed in storage. (3)* Pertaining to a storage device in which the access time is effectively independent of the location of the data. Synonymous with random access.

direct file organization: Direct file organization implies that, for the purpose of storage and retrieval, there is a direct relationship between the contents of the records and their addresses on disk storage.

directory: An index that is used by the system control programs to locate one or more sequential blocks of program information that is stored on direct access storage.

disk pack: A direct access storage volume containing magnetic disks on which data is stored. Disk packs are mounted on a disk storage drive, such as the IBM 3330 Disk Storage Drive.

diskette: A flexible magnetic oxide coated disk suitable for data storage and retrieval.

dump: (1) To copy the contents of all or part of virtual storage. (2) The data resulting from the process as in (1).

dynamic address translation (DAT): (1) The change of a virtual storage address to an address in processor storage during execution of an instruction. (2) A hardware function that performs the translation.

entry sequence: The order in which data records are physically arranged in auxiliary storage, without respect to their contents. Contrast with key sequence.

entry-sequenced file: A VSAM file whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added to the end of the file.

error message: The communication that an error has been detected.

error recovery procedures: Procedures designed to help isolate and, when possible, to recover from errors in equipment. The procedures are often used in conjunction with programs that record the statistics of machine malfunctions.

extended control program support: DOS/VSE (ECPS:VSE): An implementation of the virtual storage concept which does not require software participation in the translation of virtual addresses into real addresses.

***file:** A collection of related records treated as a unit. For example, one line of an invoice may form an item, a complete invoice may form a record, the complete set of such records may form a file, the collection of inventory control files may form a library, and the libraries used by an organization are known as its data bank.

fixed block architecture (FBA) device: A disk storage device storing data in blocks of fixed size; these blocks are addressed by block number relative to the beginning of the particular file.

hard copy: A printed copy of machine output in a visually readable form, for example, printed reports, lists, documents, and summaries.

***hardware:** Physical equipment, as opposed to the computer program or method of use, for example, mechanical, magnetic, electrical, or electronic devices. Contrast with software.

idle time: That part of available time during which the hardware is not being used.

index: A table used to locate the records of a file.

indexed-sequential access method (ISAM): An access method used to retrieve data from a file whose records are stored sequentially by key.

indexed-sequential organization: The records of an indexed sequential file are arranged in logical sequence by key. Indexes to these keys permit direct access to individual records. All or part of the file can be processed sequentially.

initial program load (IPL): The initialization procedure that causes DOS/VSE to commence operation.

integrity: Preservation of data or programs for their intended purpose.

***interface:** A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

***I/O:** An abbreviation for input/output.

IPL: See initial program load.

irrecoverable error: A hardware error which cannot be recovered from by the normal retry procedures.

ISAM: See indexed sequential access method.

ISAM interface program: A set of routines that allow a processing program coded to use ISAM to access a key-sequenced VSAM file.

job: (1)* A specified group of tasks prescribed as a unit of work for a computer. By extension, a job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) A collection of related problem programs, identified in the input stream by a JOB statement followed by one or more EXEC statements.

job accounting interface: A function that accumulates, for each job step, accounting information such as job step start and stop times and counts of I/O operations.

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 step: The execution of a single processing program.

K: When referring to storage capacity, 1024 bytes.

***key:** One or more characters associated with an item of data; these characters are used to identify it or control its use.

key sequence: The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced file: A file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the file in key sequence. Relative byte addresses of records can change.

label: Identification record for a tape or disk file.

language translator: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

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

library: A collection of files or programs, that are related by some common characteristic. For example, all phases in the core image library have been processed by the linkage editor.

linkage editor: A processing program that prepares the output of language translators for execution. It combines separately produced object modules, resolves symbolic cross references among them, and produces executable code (a phase) that is ready to be fetched or loaded into virtual storage.

load: (1)* In programming, to enter instructions or data into storage or working registers. (2) In DOS/VSE, to bring a program phase from a core image library into virtual storage for execution.

macro: In assembler language programming, a statement that causes the assembler to process a predefined set of statements called macro definition. The result of this processing, a sequence of machine instructions, replaces the macro in the source program.

macro definition: see macro.

message: See error message, operator message.

multileaving facility: Programming support which provides for transmission of jobs, data, messages, and operator commands from one central processor to another via BSC lines.

***multiprogramming:** A mode of operation that provides for the interleaved execution of two or more computer programs by a single central processor.

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

multitasking: The concurrent execution of one main task and one or more subtasks in the same partition.

object code: Output from a compiler or assembler which is suitable for processing to produce executable machine code.

***object module (program):** A module (program) that is the output of an assembler or compiler and is input to a linkage editor. Contrast with source program.

***online:** (1) Pertaining to equipment or devices under control of the central processing unit. (2) Pertaining to a user's ability to interact with a computer.

operand: (1)* That which is operated upon. An operand is usually identified by an address part of an instruction. (2) Information entered with a command name to define the data on which the command processor operates and to control the execution of the command processor.

operator command: A statement to the control program, issued via a console device; it causes the control program to provide requested information, alter normal operations, initate new operations, or terminate existing operations.

operator message: A message from the operating system or a problem program directing the operator to perform a specific action such as mounting a tape reel, or informing him of specific conditions within the system such as an error condition.

***overflow:** (1) That portion of the result of an operation that exceeds the capacity of the intended unit of storage. (2) Pertaining to the generation of overflow as in (1).

overlay: (1) A program segment (phase) that, when loaded into virtual storage, it replaces all or part of a previously retrieved section. (2) The process of replacing a previously retrieved program section in virtual storage by another section.

pacing: A procedure by which the telecommunications access method controls the rate at which data is received by application programs in the central processor or by the logical units associated with SNA-SDLC terminals. Pacing is intended to protect the application program or logical unit that is receiving data from being overrun with too much input.

page: (1) A fixed-length block of instructions, of data, or of both that can be transferred between processor storage and the page data set. In DOS/VSE, a page is 2K bytes in length. (2)To transfer instructions, data, or both between processor storage and the page data set.

page data set: One or more extents in auxiliary storage in which pages are stored.

page frame: A block of processor storage that can contain a page.

page in: The process of transferring a page from the page data set to processor storage.

page out: The process of transferring a page from processor storage to the page data set.

page pool: The set of all page frames available for paging virtual-mode programs.

***parameter:** A variable that is given a constant value for a specific purpose or process.

partition: A division of the address area of that part of virtual storage which is available for the execution of problem programs.

peripheral equipment: A term used to refer to card devices, magnetic tape and disk devices, printers, and other equipment bearing a similar relation to the central processor.

phase: The smallest complete unit that can be referred to in the core image library.

printer: A device that writes output data from a system on paper.

priority: A rank assigned to a partition that determines its precedence in receiving processing time.

private library: A user-owned library that is separate and distinct from the system library.

private second level directory: A table located in the supervisor and containing the highest phase names found on the corresponding directory tracks of the private core image library.

problem program: Any program that is executed when the central processing unit is in the problem state; that is, any program that does not contain privileged instructions. This includes IBM-distributed programs, such as language translators and service programs, as well as programs written by a user.

processing program: (1) A general term for any program that is not a control program. (2) Synonymous with problem program.

processor storage: The general purpose storage of a computer. Processor storage can be accessed directly by the operating registers. Synonymous with real storage.

program temporary fix (PTF): A temporary solution or by-pass of a problem diagnosed by IBM field engineering as a result of a defect in a current unaltered release of a program.

queue: (1) A line or list formed by items in a system waiting for service; for example, tasks to be performed or messages to be transmitted in message switching system. (2) To arrange in, or form, a queue.

random processing: The treatment of data without respect to its location in auxiliary storage, and in an arbitrary sequence governed by the input against which it is to be processed.

real address: The address of a location in real (processor) storage.

real address area: The area of virtual storage where virtual addresses are equal to real (processor storage) addresses.

real mode: The mode of a program that may not be paged.

real storage: The storage of a System/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results. Commonly referred to as processor storage.

reenterable: The attribute of a load module that allows the same copy of the load module to be used concurrently by two or more tasks.

relocatable library: A library of relocatable object modules and IOCS modules required by various compilers. It allows the user to keep frequently used modules available for combination with other modules without recompilation.

relocatable phase: Output of the linkage editor containing relocation information. The relocating loader in the supervisor uses this information to relocate the phase into any partition the user selects at the time of program execution.

restore: To write data previously copied from disk (onto magnetic tape, for example) back onto disk.

rotational position sensing: A feature which permits certain DASD devices to disconnect from a block multiplexer channel during rotational positioning operations, thereby allowing the channel to service the other devices on the channel during the positioning delay.

***routine:** An ordered set of instructions that may have some general or frequent use.

SCP: See system control program.

SDL: See system directory list.

SDLC: See synchronous data link control.

secondary storage: Same as auxiliary storage.

second level directory: A table located in the supervisor and containing the highest phase names found on the corresponding directory tracks of the system core image library.

security: Prevention of access to or use of data or programs without authorization.

sequential organization: Records of a sequential file are arranged in the order in which they will be processed.

service program: A program that assists in the use of a computing system without contributing directly to the control of the system or the production of results.

shared virtual area: An area located in the highest address range of virtual storage. It can contain a system directory list (SDL) of frequently-used phases, resident programs that can be shared between partitions, and an area for dynamic allocation to components of DOS/VSE.

SNA: See system network architecture.

software: A set of programs concerned with the operation of the hardware in a data processing system.

***source program:** A computer program written in a source language. Contrast with object program.

source statement library: A collection of books (such as macro definitions) cataloged in the system by the librarian program.

spanned records: Records of varying length that may be longer than the currently used blocksize, and which may therefore be written in one or more continuous blocks. A spanned record may occupy more than one track of a CKD disk device or more than one physical block of data of an FBA device.

spooling: The reading and writing of input and output streams on auxiliary storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

stand-alone-dump: A program that runs independently of (not controlled by) DOS/VSE and displays the contents of the registers and of virtual storage.

standard label: A fixed-format identification record for a tape or disk file. Standard labels can be written and processed by DOS/VSE.

storage protection: An arrangement for preventing access to storage.

supervisor: A component of the control program. Coordinates the use of resources and controls the flow of operations in a data processing system.

SVA: See shared virtual area.

switched line: A communication line in which the connection between the computer and a remote station is established by dialing. Synonymous with dial line.

system control program: IBM—supplied programming support that is fundamental to the operation and service of the system.

system directory list: A list containing directory entries of frequently-used phases and of all phases resident in the shared virtual area. This list is placed in the shared virtual area.

system network architecture (SNA): The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the communication system. The structure of SNA allows the ultimate origins and destinations of information — that is, the end users — to be independent of and unaffected by the specific communication-system services and facilities used for information exchange.

system residence device: The direct access device on which the system residence volume is mounted.

system residence volume: The volume on which the basic operating system and all related supervisor code is located.

system service program: see service program.

system utility program: see utility program.

synchronous data link control: A discipline for managing synchronous, transparent, serial-by-bit information transfer over a communication channel.

task: a unit of work for the central processing unit from the standpoint of the control program.

telecommunication: Data transmission between a computing system and remotely located devices via a unit that performs the necessary format conversion and controls the rate of transmission.

terminal: (1)* A point in a system or communication network at which data can either enter or leave; (2) any device capable of sending and receiving information over a communication channel.

throughput: The total volume of work performed by a computing system over a given period of time.

***track:** The portion of a moving storage medium, such as a magnetic tape, or disk, that is accessible to a given reading head position.

transient area: An area within the control program and fixed in processor storage, used for temporary storage of executable high priority code. The area is a serially reusable system resource.

unit record: A card containing one complete record; a punched card. Also a line-printer output record.

universal character set (UCS): A printer feature that permits the use of a variety of character arrays.

user label: An identification record for tape or disk file; the format and contents are defined by the user; he must also write the necessary processing routines.

utility program: A problem program designed to perform a routine task, such as transcribing data from one storage device to another.

virtual address: An address that refers to virtual storage and must, therefore, be translated into a real storage address when it is used.

virtual address area: In DOS/VSE (in System/370 mode), the area of virtual storage whose addresses are greater than the highest address of the real address area.

virtual mode: In DOS/VSE, the operating mode of a program which may be paged.

virtual storage: Addressable space that appears to the user as processor storage from which instructions and data are mapped into processor storage locations.

Virtual Storage Access Method (VSAM): An access method (available as the licensed program product VSE/VSAM) for direct or sequential processing of fixed and variable length records on direct access devices; designed for use in a virtual storage environment.

Virtual Telecommunications Access Method (VTAM): A set of IBM programs (available as the licensed program product ACF/VTAME)that control communications between terminals and application programs.

volume: (1) That portion of a single unit of storage media which is accessible to a single read/write mechanism, for example, a disk pack or a reel of magnetic tape. (2) A recording medium that is mounted and dismounted as a unit.

VSAM: See Virtual Storage Access Method.

VSAM catalog: A file containing extensive file and volume information that VSE/VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a file, and to accumulate usage statistics for files.

VTAM: See Virtual Telecommunications Access Method.

work file: A file on a secondary storage medium reserved for intermediate results during execution of a particular program.

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

X

Introduction to DOS/VSE Printed in U.S.A. GC33-5370-6

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

READER'S COMMENT FORM

GC33-5370-6

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Name

Address

What is your occupation?

How did you use this manual?

As a reference source

As a classroom text

As a self-study text

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for system analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please note: Requests for copies of publications and for assistance in utilizing 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. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 BP

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

CUT ALONG THIS LINE

Introduction to DOS/VSE Printed in U.S.A. GC33-5370-6