GC28-0600-1

Systems

OS/VS2 Planning Guide

IBM

Correction: The indexed sequential access method (ISAM) is an optional facility of OS/VS2 that is included in the system during system generation via the DATAMGT macro instruction. In this publication, ISAM is identified erroneously as a standard facility of the data management component of the system control program.

In this publication, references to the IBM 3330 Disk Storage should be interpreted as meaning the IBM 3330 Series which consists of the IBM 3330 Disk Storage Model 1 and the IBM 3333 Disk Storage and Control Model 1.

First Edition (July, 1972)

This edition supports the announcement of and subsequent releases of OS/VS2 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/360 and System/370 Bibliography,** Order No. GA22-6822, and current **SRL Newsletter,** Order No. GN20-0360, for the editions that are applicable and current.

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 Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

© Copyright International Business Machines Corporation 1972

Preface

This publication describes OS/VS2. The purpose of the publication is to introduce VS2 concepts and provide planning information for users preparing to have VS2 installed. The publication assumes basic knowledge of OS/MVT. (MVT is described in **IBM System/360 Operating System: MVT Guide,** GC28-6720.)

This publication is for planning purposes only. The functions and capabilities described herein reflect the information currently available. This information is subject to change prior to the availability of OS/VS2. Further information will be published when OS/VS2 becomes available.

This publication is divided into the following chapters:

- Introduction -- This chapter presents an overview of VS2. The chapter also discusses the advantages of a virtual storage system, the new functions provided by VS2, and the basic configuration and supported devices of the system.
- System Control Program -- This chapter presents an overview of the five major components of the VS2 system control program: job management, task management, input/output supervision, data management, and recovery management. For each item discussed, a summary of the major changes from MVT is provided.
- Standard Support Programs -- This chapter presents an overview of VS2 standard support programs (such as utilities, service aids, and linkage editor) that interact with the system control program. For each item discussed, a summary of the major changes from MVT is provided.
- Options -- This chapter presents an overview of the optional facilities that can be added to the VS2 system during and after system generation. For each item discussed, a summary of the major changes from MVT is provided.
- Compatibility -- This chapter describes the differences and incompatibilities that exist between VS2 and MVT. Where applicable, differences and incompatibilities that apply to MFT and VS1 are also discussed. The chapter provides basic information that the user needs in converting from MVT to VS2.
- Defining the System -- This chapter provides preliminary planning information that the user needs in defining his system. For example, the chapter describes the VS2 system generation procedures and macro instructions, and the VS2 system initialization procedures and parameters.
- Glossary -- This chapter defines new VS2 terms used in this publication. The chapter does not define data processing and MVT terms commonly in use.

The following items are described in this publication for planning purposes only, and will not be available with the initial release of VS2:

- Dynamic support system (DSS)
- Virtual storage access method (VSAM)

Availability dates for support of the above items may be obtained from the local IBM branch office.

Related Publications

Introduction to Virtual Storage in System/370, GR20-4260

This publication provides an overview of the advantages and concepts of virtual storage systems.

IBM System/370 System Summary, GA22-7001

This publication provides an overview of the system concepts and features of System/370, including a discussion of the components comprising OS/VS.

IBM Data Processing Glossary, GC20-1699

This publication defines terms and concepts used in IBM publications.

OS/VS Virtual Storage Access Method (VSAM) Planning Guide, GC26-3799

This publication contains a detailed description of the virtual storage access method (VSAM) that is optional in VS2.

IBM System/370 VS1 Planning and Use Guide, GC24-5090

This publication describes OS/VS1. It can assist installation personnel in selecting and evaluating VS1, and can assist system programmers in implementing, modifying, and extending the capabilities of the VS1 control program.

Contents

Figures

Introduction

Operating System/Virtual Storage 2 (VS2) is a compatible extension of OS/MVT (multiprogramming with a variable number of tasks) that is specially modified to take advantage of relocation hardware and the extended control features of System/370.

Virtual Storage

One of the primary differences between VS2 and MVT is that VS2 provides 16,777,216 bytes of addressable space, called virtual storage. It is addressable space that appears to the user's program as real storage. When a user's instructions and data are ready for processing by the CPU, they are mapped into real storage locations.

When instructions and data are said to be "virtual storage", they are actually in a portion of auxiliary storage called external page storage. When they are needed for execution, the instructions and data are loaded into real storage through a process called paging. The data and instructions are contained in 4K-byte blocks of storage, called pages.

In VS2, the term real storage is used for what was called main storage in MVT; that is, real storage is the storage of System/370 from which the central processing unit directly obtains instructions and data, and to which it directly returns results. Instructions and data are brought into real storage from virtual storage only when they are actually required by an executing program, and altered instructions and data are returned to virtual storage when they are no longer being accessed. Therefore, at any given time, real storage contains only a portion of the total contents of virtual storage. The relationships between virtual storage, real storage, and external page storage are illustrated in Figure 1.

The map of virtual storage in VS2 resembles that of main storage in MVT, except that the addressable space is enlarged. Figure 2 shows the similarities. In both systems, the dynamic area (from which regions are allocated) occupies the middle of storage, while each end is used for programs established at IPL time. In VS2, the virtual storage areas that correspond to the fixed areas of main storage in MVT are called the nondynamic areas.

Theoretically, the size of virtual storage is limited only by the addressing scheme of the computing system. (In System/370, a 24-bit address is used, allowing up to 16,777,216 bytes of addressable storage.) In reality, the size of virtual storage is limited by the amount of auxiliary storage available in the system. This concept is illustrated in Figure 3.

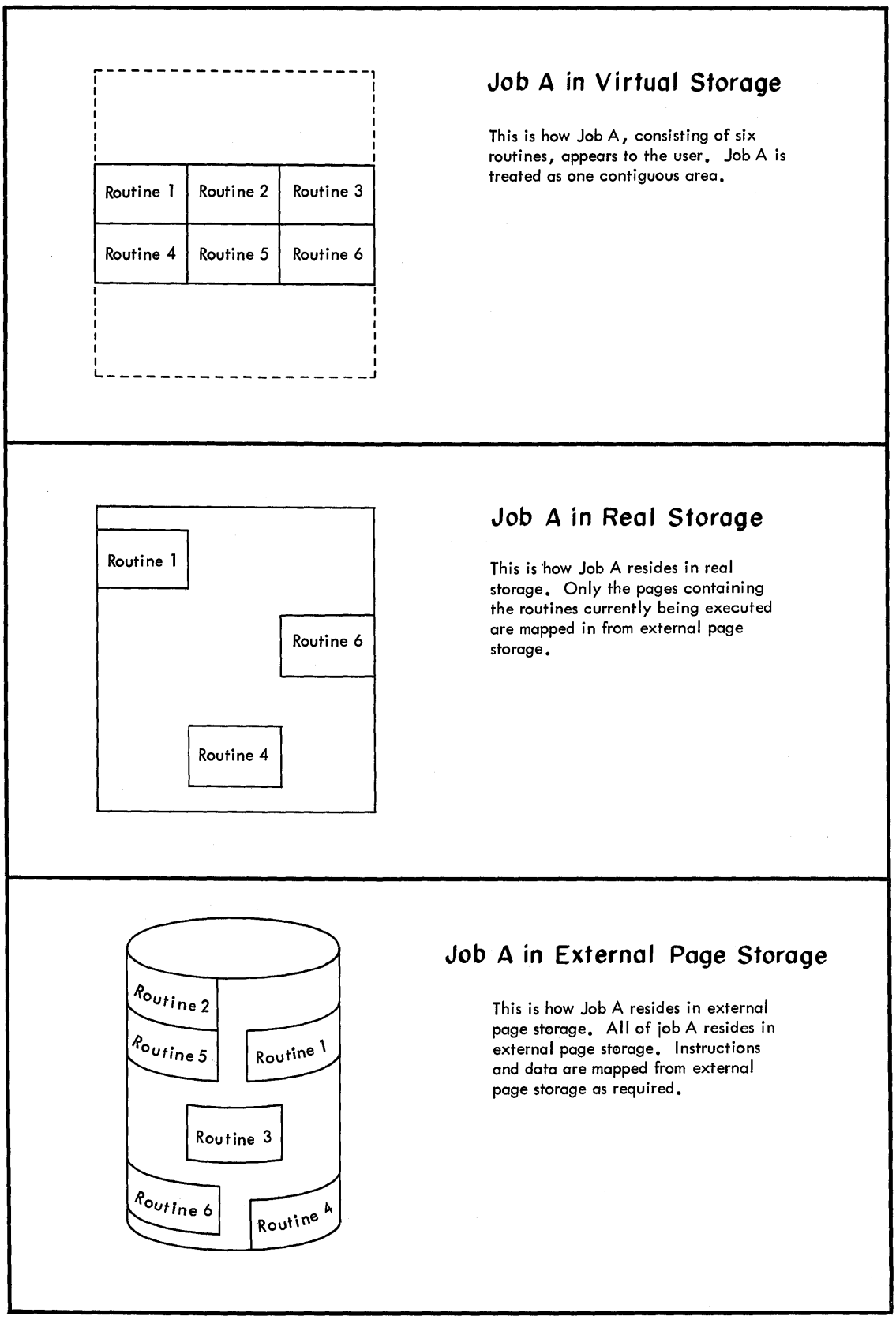
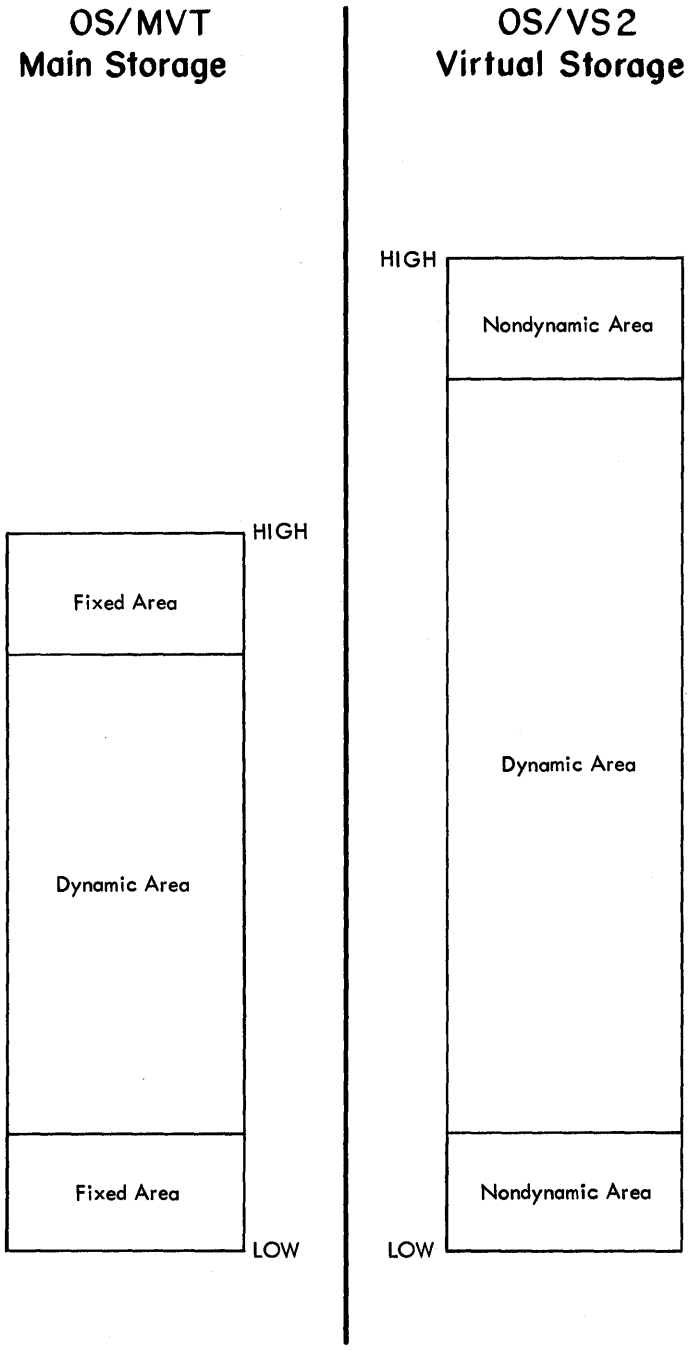


Figure 1. Relationship Between Virtual Storage, Real Storage, and External Page Storage



LOW, HIGH

Ends of storage with low and high address.

Figure 2. MVT and VS2 Storage Overviews

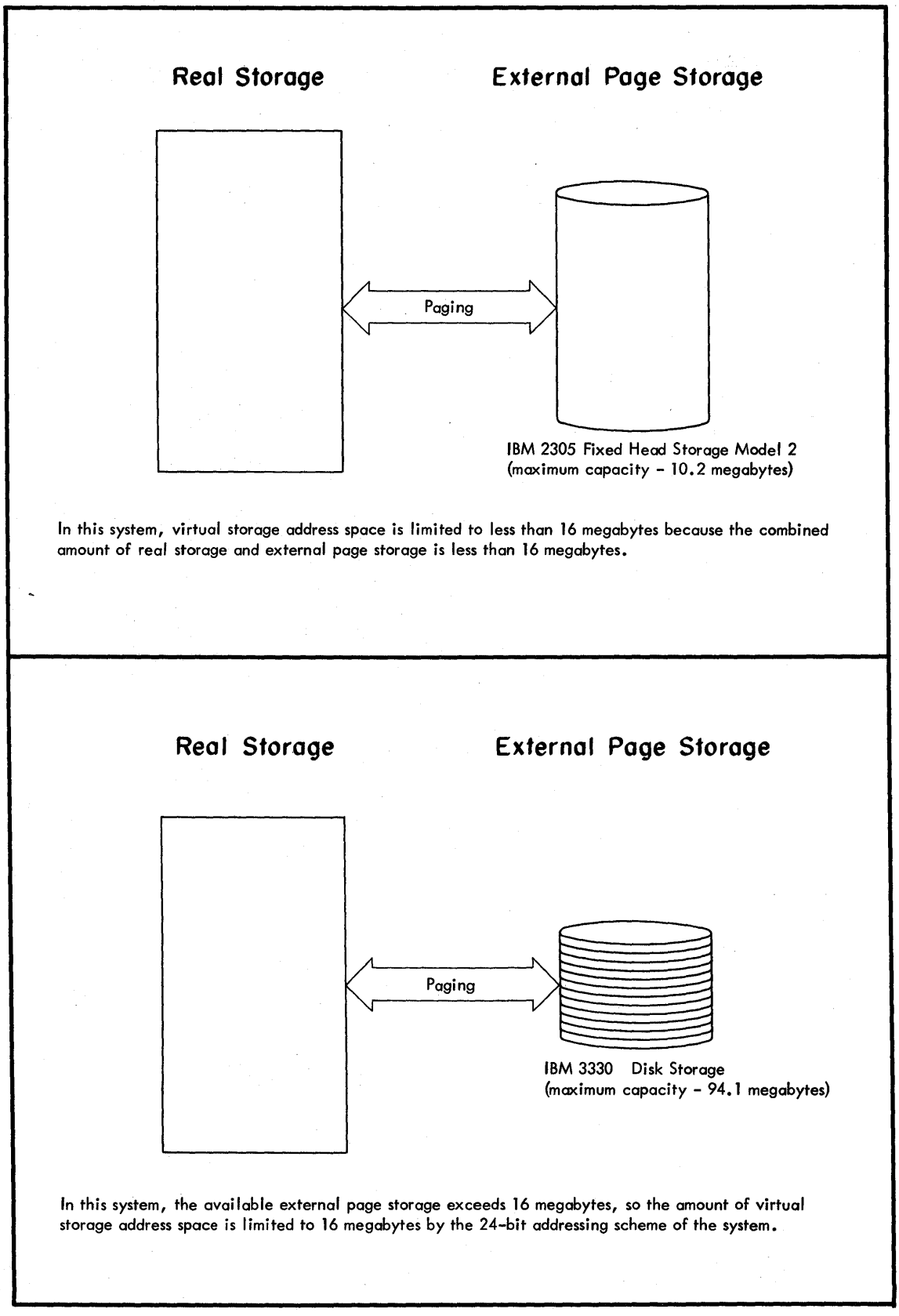


Figure 3. Relationship Between Virtual Storage Address Space and Available External Page Storage

New Functions

In addition to the benefits of using a virtual storage system (described in the following section), VS2 provides new functional enhancements over MVT. They include:

- Simplified system generation in which IBM performs various functions (including installation verification procedures) previously performed by the user.
- Additional standard facilities in VS2 that were previously optional:

  Basic direct access method (BDAM)
  Channel check handler (CCH)
  Checkpoint/restart
  Dynamic device reconfiguration (DDR)
  Hardcopy log
  Indexed sequential access method (ISAM)
  Job step timing
  Machine check handler (MCH)
  Missing interruption checker
  Multiple console support (MCS)
  Online test executive program (OLTEP)
  PCI fetch
  System log
  System management facilities (SMF)
  Volume statistics
- Additional parameter lists that can be specified during system generation and used at initialization time.
- New debugging tool for authorized maintenance personnel (dynamic support system) to help correct causes of software failures.
- New timing facilities (CPU timer and clock comparator).
- Enhanced I/O load balancing.
- New dispatching facility to provide more efficient use of CPU and I/O resources (automatic priority group).
- Improved system security and integrity (authorized program facility, fetch protection. data extent block validity checking, local system queue area, missing interruption checker).
- Enlarged link pack area (fixed and pageable) that contains reenterable routines used concurrently by all tasks in the system.
- Operator capability to cancel a job waiting for a region or a data set.
- New optional access method (virtual storage access method).
- Enhanced support programs (VS2 assembler, linkage editor, IEBCOPY utility program).
- Enhanced system management facilities (SMF).

Compatibility

VS2 is an extension of MVT; VS1 is an extension of MFT. Although the major thrust for compatiblility discussed in this publication is between VS2 and MVT, VS2 is also a an extension of VS1. Compatibility between MVT, MFT, VS2, and VS1 is discussed in detail in the chapter "Compatibility".

The main compatibility objective of VS2 is that all problem programs that run under MVT also run under VS2. In VS2, a part of virtual storage -- the nonpageable dynamic area -- was established to accommodate those programs that can not be executed in a virtual storage environment. Another objective is that capabilities added to VS2 not affect currently existing interfaces between problem programs and the operating system.

The known incompatibilities that do exist, and the modifications necessary to compensate for them, are described later in this publication. The following list briefly states the incompatibilities:

- MVT operator commands contain some new parameters, and some old parameters that should not be specified because they specify nonsupported functions. In several cases, the output generated by the commands has changed.
- TSO operator commands START and MODIFY contain both new parameters and changes to the existing parameters; also, all operator parameters now contain unique abbreviations.
- Job control language (JCL) is basically unchanged, and all jobs that run in MVT can be executed in VS2 without modification to the JCL.
- All problem programs that execute under Release 21 of MVT also execute, with no modifications, under VS2.
- Reassembly/recompilation of programs is unnecessary except for users of TCAM message control programs and message processing programs, and users of programs which have unique system dependencies.
- System data sets, except for three new data sets (SYS1.LPALIB, SYS1.PAGE, and SYS1.DSSVM), are the same as their MVT counterparts.
- Certain major functions in MVT are not supported in VS2. See Figure 4.

MVT Function Not Supported	Comparable VS2 Function
Automatic SYSIN batching (ASB) reader	---
Conversational remote job entry (CRJE)	Time sharing option (TSO)
Direct system output (DSO) writer	---
Graphic job processor (GJP)	---
IEBUPDAT utility program	IEBUPDTE utility program
IEHIOSUP utility program	Pageable link pack area
Main storage hierarchy support	---
Multiprocessing (MP65)	---
Queued telecommunications access method (QTAM)	Telecommunications access method (TCAM)
Remote job entry (RJE)	---
Rollout/rollin	Paging
Satellite graphic job processor (SGJP)	---
Scatter load	Paging
System environment recording routines (SER0 and SER1)	Recovery management support
TESTRAN program	---
Transient areas	Pageable link pack area

Figure 4. Major MVT Functions Not Supported in VS2

Planning aids for defining a system, optimizing the performance of a system, and modifying a system are vital to successfully meeting the needs of an installation.

In VS2, the process of defining the system has been made easier by the improved system generation and system initialization facilities. These processes are discussed in the chapter "Defining the System".

The performance of a system depends to a large extent on the specifications of job classes, job priorities, and system output classes. In VS2, these facilities are the same as their counterparts in MVT.

Modification of a system, including maintenance and updating, is primarily the responsibility of the system programmers at an installation. Except for those MVT facilities that are not

supported in VS2 or are no longer needed, the external implementation of the MVT facilities is the same for VS2. The facilities that fall into this category include:

- Cataloged procedures.
- System output writers.
- Job queue formatting.
- Output separation.
- Message routing exit routines.
- PRESRES volume characteristics list.
- Must complete function.
- Shared direct access storage device.
- Time slicing.

Advantages of VS2

This chapter has already briefly described the functional enhancements VS2 provides over MVT. However, the most important new feature of VS2 is its support of a virtual storage environment. The advantages that can result from using a virtual storage system are described below.

System Flexibility

In scheduling, installations strive to match work to be done against available real storage and input/output devices. In VS2, virtual storage can make this installation preplanning easier. The result is:

- Less planning to prevent storage fragmentation. In systems without virtual storage, storage fragmentation was a primary consideration when selecting jobs to be multiprogrammed. Detailed planning of job mixes to avoid storage fragmentation in real storage is no longer necessary. (In virtual storage, some fragmentation will exist, but its effect will be minimal.)
- More flexible backup. Since a small CPU can be used to execute programs designed to run on a larger CPU, storage size is now less of a consideration when planning for a backup CPU.
- More efficient use of system resources. Since real storage is obtained and released dynamically, only the real storage that is needed is tied to a job currently executing. In systems without virtual storage, all real storage belonging to a job could not be reassigned by the system until the end of the job.
- Easier handling of priority jobs. In systems without virtual storage, a high priority job would have to to wait to be executed if there was insufficient real storage to contain the job (for example, if large production jobs were occupying most of storage). In VS2, virtual storage space increases the probability that high priority jobs can begin execution without waiting for completion of the large, low priority jobs.

System Throughput

Throughput is the total volume of work performed by a computing system over a given period of time. In VS2, virtual storage management has considerable influence on the system throughput. As a result, the following enhancements may result in improved system throughput:

- More system functions that can be shared by all users are resident in virtual storage. Resident reenterable functions eliminate duplicate copies of code, reduce the need to access system libraries, and save the time required to load these functions with each program.

- More job steps can be initiated, thus achieving a higher degree of multiprogramming. Because of virtual storage and demand paging, entire jobs do not occupy real storage at any one time.
- Better balance in the use of the CPU and the input/output devices is achieved. In VS2, the optional dynamic dispatching facility automatically allocates varying slices of time to jobs having the same dynamic dispatching priority, giving preference to I/O-bound jobs over CPU-bound jobs.
- More regions can be started because of the larger addressable space. Thus, if the time sharing option is included in the system, the added regions can be used to serve more terminals.

Programmer Productivity

The additional virtual storage available with VS2 helps to reduce many of the former constraints of program design, thus allowing the programmer to concentrate on applications. As a result:

- More addressable space is available for programs. Since jobs requiring more real storage than is actually available can run in VS2, storage constraints have been eased for the programmer. In most cases, programming practices such as overlays have been made unnecessary.
- Programs with large storage requirements can be tested on machines with small real storage. Since programs are loaded into virtual storage and only parts of the programs are paged on demand into real storage, applications intended for large machines can be tested on smaller machines.
- Programs can be changed more easily. Since storage size is not a primary concern of the programmer, he can avoid intricate coding when changing a program.

System Integrity

Integrity is preservation of data or programs for their intended purpose. In VS2, system integrity has been improved in the following ways:

- The authorized program facility (APF) limits the use of sensitive system and (optionally) user services and resources to authorized system and user programs. The authorization consists of a code that is used with programs residing in the password protected SYS1.LINKLIB and SYS1.SVCLIB data sets, and in the link pack area. The SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB data sets are the only data sets in which an authorized program can reside.

  The linkage editor permits an installation to establish authorization of the programs either through a new parameter in the linkage edit step or through a new linkage editor control statement. Authorization is at the job step level -- that is, the authorization of the first program executed in the job step determines the authorization of the job step. If the job step is authorized and a program residing in the link pack area, SYS1.LINKLIB, or SYS1.SVCLIB data set is invoked (via a LINK, LOAD, ATTACH, or XCTL macro instruction), authorization is retained. However, if the job step is authorized and any program invoked does not reside in the link pack area, SYS1.LINKLIB , or SYS1.SVCLIB data set, the authorization is lost for the remainder of the job step. A program that is link edited as authorized but does not reside in the link pack area, SYS1.LINKLIB, or SYS1.SVCLIB is not considered to be authorized by the system.

  The system services and resources that are restricted in use include, for example, SVC 28 (CVOL), SVC 82 (DASDR), SVC 107 (MODESET), and SVC 113 (the new PGFREE, PGFIX, and PGLOAD system paging macro instructions). The system programs that are authorized under APF to use the restricted SVCs are the IEHDASDR, IEHATLAS, and IEHPROGM utility programs, and the AMASPZAP service aid program. The user programs

that must be authorized include those programs that use the authorized system programs and those programs that use the restricted SVCs directly. (Any problem programs executing in Release 21 of MVT that use these restricted SVCs or authorized programs must be relink-edited and moved to the proper libraries to become authorized under APF. During system generation, the user can designate the user SVCs that will require authorization.) All programs executing in the supervisor state or under protection key zero are authorized.

A new system macro instruction, TESTAUTH, has been defined to support APF. The macro instruction tests the authorization of the caller and informs the caller if it is authorized to perform a particular function.

- The data extent block (DEB) validity checking facility prevents a user from unauthorized access to data on an external device and alleviates several possibilities of transferring control in the supervisor state to an unauthorized routine.

  A new system macro instruction, DEBCHK, has been defined to support DEB validity checking. The macro instruction is used to verify that a DEB is valid.
- The local system queue area (LSQA) prevents excessive use of system queue area (SQA) space. The LSQA consists of one or more segments that are associated with each virtual storage region and contain job-related system control blocks. The isolation of job-related system control blocks makes it less likely for job failures to cause system failures. The LSQA is protected via read-only access from the problem program.
- The storage protection feature, consisting of both store and fetch protection, prevents unauthorized or unintentional access to virtual or real storage by other than the intended user. (See the discussion of storage protection later in this chapter.)

VS2 Overview

The functional capability of VS2 consists of the MVT functional base which has been extended to take advantage of virtual storage. The virtual storage concept is implemented through facilities called address translation and paging.

Address Translation

In VS2, references in a program to virtual storage locations must be translated into references to real storage locations. The process of changing the address of a data item or an instruction to its real storage address is called address translation. In VS2, the dynamic address translation (DAT) facility of the System/370 machines performs this function.

During execution of a program, each virtual storage address is translated as the instruction is executed. Translation occurs only when the central processing unit is operating in translation mode; that is, bit 5 of the extended control (EC) mode program status word (PSW) is 1. Storage addresses referenced by channels for input/output operations are not translated by the DAT feature; they are translated by the I/O supervisor.

New program interruptions indicate when the translation process cannot be completed. They are:

- Page translation exception, which occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set. The page table indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.
- Segment translation exception, which occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set. The segment table is used to control user access to virtual storage segments.
- Translation specification exception, which occurs when a page table entry, segment table entry, or the control register pointing to the segment table contains information in an invalid format.

Paging

A program can be executing even though some of its pages are not in real storage. The specific pages needed in real storage at any given time depend upon the particular program being executed. The process of transferring pages between real storage and external page storage (the portion of auxiliary storage used to contain pages) is called paging.

When an instruction is executed and addresses are translated, an interruption occurs if a page is referred to and it is not in real storage. The paging supervisor brings the page into real storage from an external page storage device. (The paging supervisor is discussed in the chapter "System Control Program".)

When real storage is needed for a page being paged in from external page storage the real storage will be made available by the paging supervisor. If a page in real storage was modified during execution, it is written out to an external storage device to make room for the required page. If a page in real storage was not modified and an exact copy of the page already exists on an external storage device, then the page need not be written out; the page awaiting real storage space is loaded into real storage overlaying the unmodified page.

The system sometimes commits more real storage that can be supported efficiently, and excessive paging occurs in an effort to satisfy that commitment. If the paging rate becomes excessive, there is a constant request queue at the paging device, and active programs resident in storage become idle, waiting for service to their paging requests. This condition, called thrashing, results in little useful work being done. To control thrashing, the dispatcher selects and marks tasks nondispatchable. When the paging rate subsides, the tasks that were marked nondispatchable are reset to dispatchable status.

Some programs cannot be paged. For example, programs that modify channel programs while they are executing cannot be paged since the I/O supervisor duplicates and uses a copy of the channel command words (CCWs) as part of its translation process; any changes made to the original CCW during execution would not be known to the I/O supervisor.

Programs that are highly time dependent (such as the magnetic ink character recognition programs) cannot be paged because the time required to translate the channel programs cannot be handled in the amount of time available. Such programs must be run in an area of virtual storage that has the same range of addresses as real storage, called nonpageable dynamic storage (or virtual equals real or V=R storage).

Storage Maps

Figure 5 shows a map of virtual storage. Figure 6 shows a map of real storage. The maps are divided into the following areas:

System Queue Area - The system queue area is an area of virtual storage reserved for control blocks not related to jobs and job steps and tables (such as the segment tables) maintained by the control program. The amount of fixed real storage required to back up the system queue area varies with the demands of the system.

Pageable Link Pack Area - The pageable link pack area is an area of virtual storage that contains reenterable routines that can be used concurrently by all tasks in the system. As needed, parts of this area are paged into real storage. The fixed link pack area, described below, is an extension of this area.

Pageable BLDL Table - The BLDL table can be either fixed or paged, but not both. (The fixed BLDL table is described below.) The pageable BLDL table occupies an area in the upper portion of virtual storage. It consists of the list of entries for the SYS1.LINKLIB data set that are to be paged into real storage as needed.

Master Scheduler Region - The master scheduler region is an area of virtual storage that contains the master scheduler routine. As needed, parts of this area are paged into real storage.

Master Scheduler Local System Queue Area - The master scheduler LSQA is an area of virtual storage that contains system control blocks (such as the task control block) for the master scheduler. At least one page of this area always remains in real storage.

Pageable Dynamic Area - The pageable dynamic area is an area of virtual storage whose virtual storage addresses are not necessarily identical to real storage addresses. It is used for programs that can be paged during execution. Reader/interpreter regions, system output writer regions, initiator/terminator regions, regions for pageable problem programs, and LSQAs for each of these types of regions are all allocated from the pageable dynamic area.

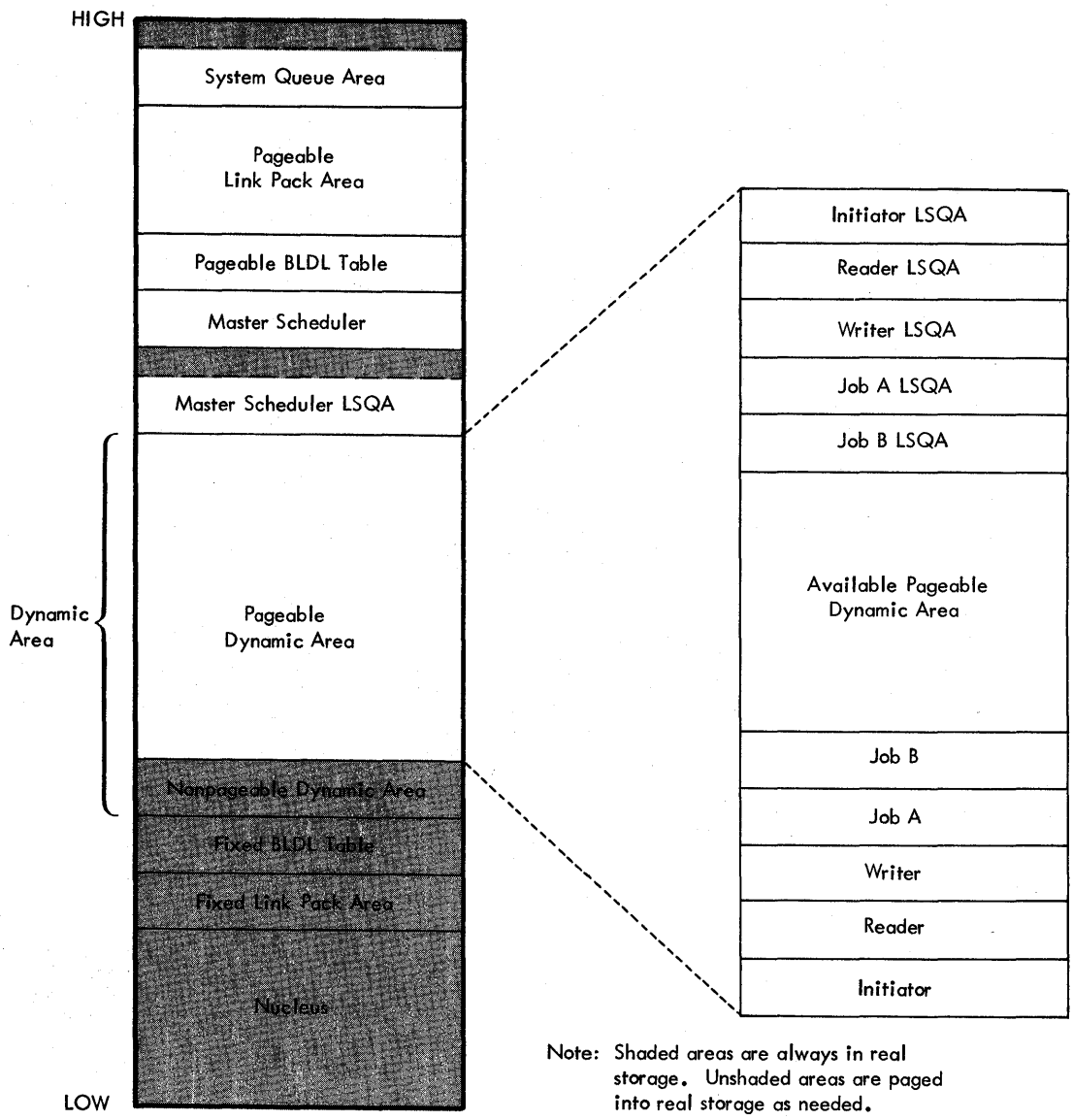
Nonpageable Dynamic Area - The nonpageable dynamic area (also called virtual equals real or V=R storage) is an area of virtual storage whose virtual storage addresses are identical to real storage addresses. It is used for programs that are not to be paged during execution. If the virtual storage in this area is not assigned for nonpageable programs, the real storage with the same storage addresses is available for paging. The minimum size of this area is 64K bytes.

Fixed BLDL Table - The BLDL table can be either fixed or paged, but not both. The fixed BLDL table is present only if the user specifies it to occupy an area in the lower portion of virtual storage. It consists of the list of entries for the SYS1.LINKLIB data set that are to remain in real storage.

Fixed Link Pack Area - The fixed link pack area is present only if the user specifies it at IPL time. It is an extension of the pageable link pack area and occupies an area in the lower portion of virtual storage. It is used for frequently used reenterable routines which are to remain in real storage.

Nucleus - The nucleus, mapped into virtual storage at location zero, consists of those system programs which must remain in real storage while the system is in use.

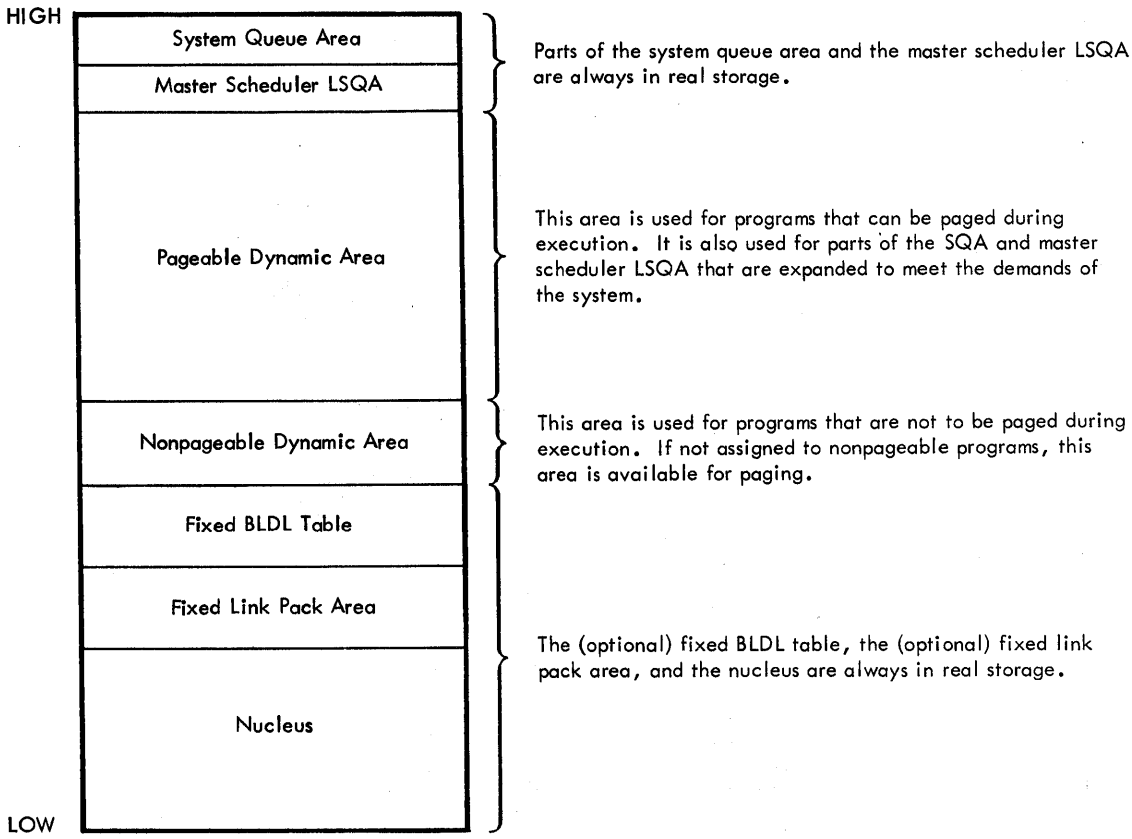
Virtual storage is divided into 256 segments of 64K bytes each. Each segment is divided into 16 pages of 4K bytes each. Regions in the pageable area of virtual storage are allocated in multiples of 64K bytes (segments); regions in the nonpageable dynamic area of virtual storage are allocated in multiples of 4K bytes (pages).



LOW, HIGH

Ends of storage with low and high address.

Figure 5. VS2 Virtual Storage Map



LOW, HIGH

Ends of storage with low and high address.

Figure 6. VS2 Real Storage Map

Storage Protection

Storage protection is a feature that prevents unauthorized or unintentional access to virtual storage by other than the intended user. Because of the larger addressable space that can be used to start more regions in a virtual storage system, storage protection is extended to allow for the additional number of regions that have to be protected. In VS2, storage protection limits both store and fetch access to storage. (Store protection prevents the contents of storage from being altered by storage addressing errors in programs or input from I/O devices. Fetch protection prevents the unauthorized retrieval of data and instructions from storage.)

Programs executing in pageable dynamic storage have read-only access to the areas of the system that are always in real storage (for example, nucleus and system queue area), and full access to only their own regions. All programs executing in pageable dynamic storage are assigned the same non-zero protection key.

Programs executing in nonpageable dynamic storage have the same access as programs executing in pageable dynamic storage. However, these programs each have unique non-zero protection keys.

System tasks have access to all allocated areas of virtual storage. These tasks have a protection key of zero.

Programs executing in pageable dynamic storage are protected from each other by the valid/invalid bits in the segment table entries; these bits control the address range the

programs can reference. Programs executing in nonpageable dynamic storage are protected from each other through their protection keys. The protection keys are also used to provide protection between programs executing in pageable and nonpageable dynamic storage.

Configuration

OS/VS2 supports the System/370 models listed below. The Model 145 will be available with the initial release of OS/VS2; the local IBM branch office should be contacted for availability of the other models.

Model 145
Model 155II
Model 158
Model 165II
Model 168

A batch and TSO system will operate in 512K bytes of real storage.[1]

Basic Configuration

The minimum configuration required for VS2 is:

- One IBM System/370 of the type listed above, with the minimum storage as indicated above. (The clock comparator and CPU timer feature (#2001) is required on the Model 145.)
- One multiplexer channel.
- One selector or block multiplexer channel.
- Three IBM 3330 Disk Storage devices, or four IBM 2314 Direct Access Storage Facility or IBM 2319 Disk Storage devices.
- One card reader and punch.
- One line printer.
- One system console.

If IBM 3330 Disk Storage devices are used for system generation, a total of four devices is necessary. A 9-track magnetic tape is required to restore the OS/VS2 system from magnetic tape to a disk drive for system generation and maintenance.

External Page Storage

External page storage is the portion of auxiliary storage that is used to contain pages. There should be sufficient external page storage in the system for all of virtual storage except that part of virtual storage (essentially, the nucleus and other nonpageable storage) that permanently resides in real storage. If the space is inadequate, a user may have to wait for extend page storage when his job is initiated.

If the time sharing option is included in the system, external page storage must also accomodate swapping requirements. Although only 16,777,216 bytes are addressable, several TSO users can share the same region and therefore external page storage in excess of 16,777,216 bytes may be needed for backup.

In VS2, a primary paging device is an auxiliary storage device that is used in preference to secondary paging devices for paging operations; portions of a primary paging device can be used for purposes other than paging operations. A secondary paging device is an auxiliary storage device that is not used for paging operations until the available space on primary

---

[1]A minimum OS/VS2 system will operate in 384K bytes of real storage, batch and reader/writer operating concurrently.

paging devices falls below a specified minimum; portions of a secondary paging device can be used for purposes other than paging operations.

Figure 7 shows the devices that may be used for external page storage, and their capacities. If both fixed-head and moveable-head devices are used, the fixed-head devices should compose the primary external page storage. External page storage may be allocated on as many as 16 devices of the types eligible.

When the primary paging device or devices become full, page migration occurs. Page migration is the transfer of pages from the primary paging device to the secondary paging device to make more space available on the primary device. The pages transferred are assoicated with the region in the pageable dynamic area with the lowest priority.

Device	Maximum number of pages	Maximum capacity in megabytes
IBM 2305 Model 1, Fixed Head Storage	1,146	4.7
IBM 2305 Model 2, Fixed Head Storage	2,483	10.2
IBM 2314 Direct Access Storage Facility	6,392	26.2
IBM 2319 Disk Storage	6,392	26.2
IBM 3330 Disk Storage	22,968	94.1

Figure 7. External Page Storage Devices

Input/Output Devices

The following input/output devices will be supported in VS2. (Any changes to the following list of supported devices are available from the local IBM branch office.)

Unless otherwise noted in this list, terminals are supported in both TCAM and BTAM. Availability of TCAM support for those terminals indicated by asterisk (*) may be obtained from the local IBM branch office.

Note: Terminals which are equivalent to those explicitly supported may also function satisfactorily. The customer is responsible for establishing equivalency. IBM assumes no responsibility for the impact that change to the IBM-supplied products or programs may have on such terminals.

Direct Access Storage Devices

IBM 2305 Fixed Head Storage Model 1 (Models 165II and 168 only)
IBM 2305 Fixed Head Storage Model 2
IBM 2314 Direct Access Storage Facility
IBM 2319 Disk Storage
IBM 3330 Disk Storage

Note: All of the above devices are supported as system residence, input/output, paging, and spooling devices.

Direct Access Storage Control Units

IBM 2835 Storage Control Model 1 (Models 165II and 168 only)
IBM 2835 Storage Control Model 2
IBM 2844 Auxiliary Storage Control
IBM 3345 Storage and Control Frame Models 3, 4, and 5 (Model 145 only)
IBM 3830 Storage Control Models 1 and 2 (including 2 channel switch additional #8171)
IBM Integrated File Adapter (IFA) Feature #4650 (Model 145 only)
IBM Integrated Storage Controls (Models 158 and 168 only)

Magnetic Tape Devices

IBM 2401 Magnetic Tape Unit
IBM 2420 Magnetic Tape Unit

IBM 2495 Tape Cartridge Reader
IBM 3410 Magnetic Tape Unit (Models 145, 155II, and 158 only)
IBM 3411 Magnetic Tape Unit and Control (Models 145, 155II, and 158 only)
IBM 3420 Magnetic Tape Unit

Control Units

IBM 2803 Tape Control
IBM 2804 Tape Control
IBM 3803 Tape Control

Tape Switch

IBM 2816 Switching Unit

Paper Tape Devices

IBM 2671 Paper Tape Reader

Printers

IBM 1403 Printer Models 2,7, and N1
IBM 1443 Printer Model N1
IBM 3211 Printer

Printer Control Units

IBM 2821 Control Unit Model 2 and 3
IBM 3811 Printer Control Unit

Card Readers and Punches

IBM 2501 Card Reader Models B1 and B2
IBM 2520 Card Read Punch
IBM 2540 Card Read Punch
IBM 3505 Card Reader
IBM 3525 Card Punch

Reader and Punch Control Units

IBM 2821 Control Unit Models 1, 5, and 6

Optical Character Recognition (OCR)/ Magnetic Ink Character Recognition (MICR) Devices

IBM 1287 Optical Reader
IBM 1288 Optical Page Reader
IBM 1419 Magnetic Character Reader (Dual Address Adapter #7730 and Expanded Capability #3800 features required)

Consoles

IBM 2150 Console/IBM 1052 Printer-Keyboard Model 7
IBM 2250 Display Unit Models 1 and 3
IBM 2260 Display Station Model 1
IBM 2740 Communication Terminal Model 1
IBM 3066 System Console (Models 165II and 168 only)
IBM Model 158 Display Console
IBM 3210 Console Printer-Keyboard
IBM 3215 Console Printer-Keyboard
IBM 3270 Information Display System
IBM 3213 Printer (Model 158 only)

Start/Stop Terminals

IBM 1030 Data Collection System
IBM 1050 Data Communication System
IBM 1060 Data Communication System
IBM 2260 Display Station Models 1 and 2
IBM 2265 Display Station

IBM 2740 Communication Terminal Models 1 and 2
IBM 2741 Communication Terminal
IBM 2760 Optical Image Unit
IBM System/7 (as an IBM 2740 Communication Terminal Model 1 with checking)
AT & T Model 83B3 Selective Calling Stations
Teletype[1] Models 33 and 35 (Paper tape is not supported with Teletype.[1])
Western Union Plan 115A Outstations
IBM World Trade Telegraph Terminals

Binary Synchronous Terminals

IBM 1130 Computing System Processor Station
IBM 1800 Data Acquisition and Control System Processor Station (BTAM support only)
IBM 2770 Data Communication System
IBM 2780 Data Transmission Terminal
*IBM 2790 Data Communication System (BTAM support only)/2715 Transmission Control Unit Model 2
IBM 2972 General Banking System Models 8 and 11
*IBM 3270 Information Display System
*IBM 3670 Brokerage Communication System (TCAM support only)
*IBM 3735 Programmable Buffered Terminal
IBM System/3 Processor Station
IBM System/360 Processor Station (including Model 25 Integrated Communications Adapter)
IBM System/360 Model 20 Processor Station
IBM System/370 Processor Station (including Model 135 Integrated Communications Adapter)

Locally - Attached Terminals

IBM 2250 Display Unit Models 1 and 3 (GAM and GSP support only)
IBM 2260 Display Station Models 1 and 2 (GAM, GSP, and TCAM support only)
*IBM 3270 Information Display System

Telecommunication Control Units

IBM 2701 Data Adapter Unit
IBM 2702 Transmission Control
IBM 2703 Transmission Control
*IBM 2715 Transmission Control Unit Model 1
*IBM 3705 Communications Controller (Emulation program support only) (TCAM support only)
IBM 7770 Audio Response Unit Model 3 (IBM 2721 Portable Audio Terminal and IBM 2730 Transaction Validation Terminal Model 1 support only) (TCAM support only)

Devices Supported by World Trade Only

IBM 1275 Optical Reader Sorter (Dual Address Adapter and Expanded Capability feature required)
IBM 1419 Magnetic Character Reader Models 31 and 32 (Dual Address Adapter and Expanded Capability feature required)

---

[1]Trademark of Teletype Corporation, Skokie, Illinois.

System Control Program

The VS2 system control program consists of the following major components:

- Job management, which schedules all work done by the computing system.
- Task management, which allocates system resources and controls system execution.
- Input/output supervision, which performs all I/O operations.
- Data management, which coordinates data flow.
- Recovery management, which attempts to recover from system malfunctions.

Job Management

Job management is a term that is generally used to describe the functions of the master scheduler and the job scheduler. The master scheduler and the job scheduler are major parts of the operating system that control the processing of jobs. The master scheduler initializes the system and responds to operator commands by initiating the requested actions. The job scheduler reads and interprets job definitions, schedules the jobs for processing, initiates and terminates the processing of jobs and job steps, and records job output data.

Figure 8 provides a summary of the VS2 changes to the job management functions.

Facility	Change
Master scheduler	Changes to operator commands to reflect virtual storage and nonsupport of some MVT parameters.
Job scheduler	ASB reader, DSO writer, and column binary not supported; new I/O load balancing facility; new maximum number (63) of initiators that can be started; ability of operator to cancel a job waiting for a region or a data set.
Multiple console support (MCS)	Now standard (previously optional).
System log	Now standard (previously optional).
Hardcopy log	Now standard (previously optional).
Job step timing	Now standard (previously optional).
Checkpoint/restart	Now standard (previously optional); only 2K block size supported.
System management facilities (SMF)	Now standard (previously optional); tape and OUTLIM facility not supported; new exit for system output; new accounting information to record virtual storage usage.
Automatic volume recognition (AVR)	None. (See the chapter "Options".)
Time slicing	None. (See the chapter "Options".)
Device independent display operator console support (DIDOCS)/status display support (SDS)	None. (See the chapter "Options".)
Track stacking	None. (See the chapter "Options".)

Figure 8. VS2 Changes to Job Management

Master Scheduler

The master scheduler is one of the system tasks established when the system is loaded. Its functions can be divided into two categories: initialization and command processing.

Initialization

Master scheduler initialization consists of the following functions:

- Initializing the communications task to handle all communication with the operator console.
- Scheduling execution of the initial SET command.

- Initializing the time-of-day clock.
- Initializing the job queue (SYS1.SYSJOBQE) data set.
- Setting volume attributes of all volumes listed in PRESRES.
- Scheduling execution of the automatic START commands.
- Scheduling execution of the SEND command.
- Initializing the system log.
- Initializing the system management facilities (SMF).
- Initializing the missing interruption checker task.

Major Changes from MVT: None.

Command Processing

Command processing is the reading, scheduling, and executing of operator commands issued via either a console device or an input job stream.

The reading of commands entered via a console device is performed by routines operating under a console communication task; the reading of commands entered via an input job stream is performed by routines operating under a reader task associated with that input job stream.

The scheduling of a command is the storing of the command and the readying of a task to continue processing the command. A command scheduling routine operates under either the console communications task (when the command was issued via a console device), or the reader task (when the command was issued via an input job stream).

The executing of a command is the performance of the function specified in the command. The functions are performed either as new tasks established by the master scheduler or as parts of existing system tasks. Commands are classified, accordingly, as task-creating commands and existing-task commands. In VS2, the task-creating commands START and MOUNT are attached and run in a separate region.

Major Changes from MVT: In VS2, the size of the master scheduler region can be specified at IPL time. Also, some operator commands have been changed; these changes are discussed in the chapter "Compatibility".

Job Scheduler

The job scheduler is divided into three major parts: the reader/interpreter, the initiator/terminator, and the output writer. Each part is a separate task and can thus be executed concurrently with and independently of the others.

Reader/Interpreter

The reader/interpreter reads job and step definitions from an input job stream, analyzes the definitions, and builds control blocks and tables that are used during initation and execution of the job steps. The reader/interpreter also reads and analyzes commands encountered in the input stream. When the reader/interpreter encounters data in the input stream, it writes the data on a direct access device.

The control blocks and tables constructed by the reader/interpreter contain the following information:

- Job attributes.
- Job step attributes.
- Information needed to assign devices to data sets.
- Data set attributes.

Major Changes from MVT: In VS2, the automatic SYSIN batching (ASB) reader is not supported. Therefore, column binary is also not supported.

Initiator/Terminator

The initiator/terminator selects jobs and job steps to be executed. It analyzes the I/O device requirements of the job steps, allocates devices to them, creates tasks for them, and at completion of the jobs, supplies control information for writing job output on a system output unit.

After selecting an interpreted job to be executed, the initiator/terminator examines the types of regions requested for the job. If a job or job step requires nonpageable dynamic storage for execution, the initator/terminator reserves a unique non-zero protection key for the job. (See the discussion of storage protection in the chapter "Introduction".)

During allocation, in order to reduce contention for I/O devices, a new algorithm for I/O load balancing is used. The new algorithm allocates devices for data sets that have nonspecific device requests. Rather than basing the algorithm on a count of allocated data sets on a device (as in MVT), in VS2 the actual number of I/O requests to a tape or direct access device will be monitored to get a more accurate picture of I/O load. The device determined to be the best candidate for allocation to a given data set is then selected.

The following factors can contribute to the efficient allocation of I/O devices and thus help the user realize benefit from I/O load balancing:

- Devices not dedicated to specific applications should be distributed evenly across channels; within channels, they should be distributed evenly across control units.
- DD statements entered at the time of job initiation should be sequenced in the order of expected activity.

Major Changes from MVT: In MVT, a maximum of 15 initiators could be started; in VS2, a maximum of 63 initiators can be started. In VS2, the operator has the ability to cancel a job waiting for a region or a data set. VS2 provides a more accurate algorithm for I/O load balancing.

Output Writer

The output writer transfers system messages and system output data sets from the direct access volume on which they were initially written by the system to a specified output device.

Output data sets can be directed to a class of devices, and references to the data are then placed on an output work queue. Because the queue is maintained in priority sequence, the system output writers can select jobs in the output work queue on a priority basis.

Major Changes from MVT: In VS2, the direct system output (DSO) writer is not supported.

Facilities

The major facilities provided by job management are:

- Multiple console support.
- System log.
- Hardcopy log.
- Checkpoint/restart.
- System management facilities.
- Job step timing.

Multiple Console Support (MCS)

Multiple console support (MCS) allows one operating system to use many operator consoles. Each console in a multiple console configuration is defined by specifying:

- The operator commands the system will accept from that console.
- A console to act as an alternate if a failure occurs.
- The types of messages the console will receive.

In a system with MCS, one console acts as the master console and the rest (up to thirty-one) are secondary consoles. The master console is the basic console required for operator-system communication; it alone can accept all possible operator commands, change the status of the hardcopy log and the messages to be recorded on it, switch to a different master console, and receive all messages not specifically assigned to any other console. A secondary console is any console other than the master console; it handles one or more functions assigned to it (for example, it might handle tape activity).

Major Changes from MVT: In VS2, MCS is a standard facility; in MVT, MCS was optional.

System Log

The system log consists of data sets on which the communication between problem programs, operators, and the system is recorded. It may contain the following kinds of information:

- Job time, job step time, and data from the JOB and EXEC statements of a job that has ended.
- Operating data entered by problem programs using a write-to-log (WTL) macro instruction.
- Descriptions of unusual events that occurred during a shift.
- Write-to-operator (WTO) and write-to-operator with reply (WTOR) messages.
- Accepted replies to WTOR messages.
- Commands issued through operator's consoles and the input stream, and commands issued by the operating system.

Major Changes from MVT: In VS2, the system log is a standard facility; in MVT, it was optional.

Hardcopy Log

The hardcopy log is a permanent record of system activity that is mandatory for systems with an active graphic console or multiple active consoles; for other systems, the primary console device serves as the hardcopy log.

Since multiple console support allows more than one console in a system, an installation might find it helpful to record all the messages issued by and to a system. The hardcopy log is a place to collect these messages, and therefore an installation can review system activity by reviewing message activity.

Major Changes from MVT: In VS2, the hardcopy log is a standard facility; in MVT, it was optional.

Checkpoint/Restart

If a job step is terminated before successful completion, checkpoint/restart can make it possible to resume execution from the beginning of the step or from a place within the step. Either way, the restart can be made to occur after resubmission of the job by the programmer or it can be made to occur automatically when the failure occurs.

The CHKPT macro instruction is coded in the user's program at a checkpoint to be taken. A checkpoint is the point at which information about the status of a job can be recorded so that the job step can be later restarted.

Checkpoint/restart includes a checkpoint routine and several restart routines.

The checkpoint routine gathers and records on a checkpoint data set enough information about the status of the job step and its related control blocks to allow a restart from the place where the checkpoint is taken.

The restart routines can be invoked when a job step is resubmitted for restart, or they can be invoked automatically when a failure occurs. The functions performed by restart routines depend upon the type of restart that is requested.

If the restart is to be made from the beginning of a job step, the RESTART parameter of the JOB statement must contain the name of the step to be restarted, and routines of the ready task simply bypass preceding steps and begin processing with the named step.

If a step is to be restarted from the beginning, automatically, then restart processing begins during step termination. The step termination routine of job management invokes routines to verify that a restart can be performed and requests the operator to authorize the restart.

If a step is to be restarted from a place where a checkpoint was taken and the job is resubmitted, the RESTART parameter of the JOB statement must identify the step and checkpoint identifier, while a SYSCHK DD statement must describe the checkpoint data set.

If a step is to be restarted automatically from a place where a checkpoint was taken, the step termination routine invokes routines to ensure that all data sets for the step are kept.

Major Changes from MVT: In VS2, checkpoint/restart is a standard facility; in MVT, it was optional. Also, in VS2, only 2K block size data sets are supported for checkpoint/restart data sets.

System Management Facilities (SMF)

System management facilities (SMF) collect and record system information. The information obtained can be used in management information reports that describe system efficiency, performance, and usage. The SMF records contain such data as:

- System configuration.
- Job and job step identification.
- CPU wait time.
- CPU and input/output device usage.
- Temporary and non-temporary data set usage and status.
- Virtual and real storage usage.
- Status of removable direct access volumes.
- Allocation recovery records.
- Paging statistics.

SMF provides exits to installation-supplied routines that can monitor the operation of a job or job step and generate the installation's own SMF records. The exit routines can cancel jobs, write records to the SMF data set, open and close user-defined data sets, suppress the writing of certain SMF records, and enforce installation standards (such as identification of users). Dummy routines are automatically provided for all unused exits.

Major Changes from MVT: SMF has been changed in the following ways:

- SMF is a standard facility of VS2.
- SMF records in VS2 contain additional accounting information to record new system environmental characteristics.

- SMF in VS2 provides one new exit from the system control program that receives control each time an SMF logical record has been formatted and is ready to be written out. The exit may prevent the record from being written.
- In VS2, SMF does not support tape for recording SMF data.
- In VS2, the OUTLIM facility is not supported.

Job Step Timing

Each job step can be timed and the time limits enforced. The amount of time used is recorded after a job step is finished. In addition, the following are included in this facility: the ability to request the date plus the time of day, to change the time at midnight, and to request, check, and cancel intervals of time.

Major Changes from MVT: In VS2, the job step timing facility is standard; in MVT, it was optional.

Task Management

Task management is a major function of the operating system that coordinates the use of resources and maintains the flow of central processing unit (CPU) operations. It assigns resources to perform tasks, keeps track of all such assignments, and ensures that the resources are freed upon task completion. VS2 changes to task management functions are summarized in Figure 9.

Facility	Change
Interruption supervision	New timer facilities (CPU timer and clock comparator); new program interruptions (translation exceptions, program event recording, set system mask, and monitor call); new extended control (EC) mode architecture.
Paging supervision	New support for virtual storage; support for TSO swapping function.
Task supervision	New dispatching facility (automatic priority group).
Contents supervision	Scatter load, TESTRAN, storage hierarchies, and transient areas not supported; PCI fetch now standard (previously optional).
Virtual storage supervision	Support for virtual storage (similar to MVT main storage supervision).
Timer supervision	New timer facilities (CPU timer and clock comparator); location 80 timer not supported.

Figure 9. VS2 Changes to Task Management

Interruption Supervision

All supervisory activity begins with an interruption (a break in the normal sequence of instruction execution). An interruption can occur either because a program has requested a control program service or because an event has occurred which requires supervisory processing. There are five types of interruptions:

- Supervisor call (SVC) interruptions occur when a supervisor call (SVC) instruction is executed.
- Input/output interruptions occur when an I/O device is readied or an I/O operation terminates, or during an I/O operation such as a program controlled interruption (PCI).
- Timer/external interruptions occur when a specified time interval expires or when the interruption key of the system control panel is pressed.

- Program interruptions occur when a program attempts an invalid action, when a data error is detected, or where a page fault occurs.
- Machine check interruptions occur when the CPU detects hardware malfunctions.

Any interruption causes the current program status word (PSW) to be replaced by a new PSW. The new PSW causes an appropriate interruption handler to receive control, depending on the type of interruption. The interruption handler saves critical information (such as register contents and PSW information) necessary to return control to the interrupted program after the interruption is processed. In most cases, the interruption handler analyzes the interruption and passes control to a special purpose routine for processing the interruption.

Major Changes from MVT: In VS2, the interruption handlers will support the new timer facilities and react to the new program interruptions (translation exceptions, program event recording, set system mask, and monitor call). All of the interruption handlers have been modified to support the new extended control (EC) mode architecture; for a discussion of EC mode, see "Problem Programs" in the chapter "Compatibility".

Paging Supervision

The paging supervisor allocates and releases real storage space for pages, and transfers pages between real storage and external page storage. Whenever a virtual address in a page is referred to and that page is not in real storage, an interruption occurs and control is given to the paging supervisor.

First, page reclamation is attempted in order to satisfy the request if the page is already in storage. In this case, the page may be available (previously released but not yet paged out), in page-out processing (being transferred from real storage to external page storage), or in page-in processing (being transferred from external page storage to real storage). If the required page is found, appropriate action is taken to use the page. Otherwise, to move pages into real storage, the paging supervisor maintains a list of page frames that are not receiving a high reference rate by the programs that are currently using them.

When the number of available page frames falls below a predetermined value specified at NIP time, the paging routines will select the page frames of the least-used pages and make them available for use. The selection process is primarily based on the change and reference bit settings in the storage key associated with each page frame.

If a page in a selected frame has been changed, the changed page is moved to external page storage before the page frame is made available. If a page in a selected frame has not been changed, it is not moved to external page storage if a copy already exists in external page storage.

The selecting of page frames continues until a predetermined maximum number of available frames is reached. During this process, page frames that contain unreferenced pages are selected before page frames that contain referenced pages. Also, page frames that contain unchanged pages are selected before page frames that contain changed pages.

The paging process utilizes an efficient slot sorting technique to optimize paging operations by minimizing page data set seek or rotational delay time, or both. A slot is a continuous area on a paging device in which a page can be stored. The capability to access the next slot sequentially forms the basis of the slot sorting algorithm.

In summary, the paging supervisor:

- Recognizes when a page must be transferred to real storage.
- Selects a page frame in which to place the page.
- Maintains a supply of page frames.
- Saves pages that have been changed in real storage.
- Recognizes when a page must be made non-transferable to external page storage.

Major Changes from MVT: The paging supervisor is necessary to support virtual storage, and was not needed in MVT. When the time sharing option (TSO) is included in the system, the paging supervisor performs the swapping function performed by the TSO supervisor in MVT.

Task Supervision

The task supervisor performs services requested by tasks and allocates CPU time among competing tasks. A task is described to the operating system by a task control block (TCB) and many services provided by task supervision relate to the TCB. The services supplied are listed below. Where applicable, the macro instructions used to request the services are shown in parentheses.

- Attaching/detaching a subtask (ATTACH/DETACH).
- Changing the dispatching priority of a task (CHAP).
- Extracting information from a TCB (EXTRACT).
- Specifying a user program interruption exit routine (SPIE).
- Synchronizing a program with one or more events (WAIT,POST).
- Serializing the use of one or more resources (ENQ, DEQ, RESERVE).
- Scheduling asynchronous exit routines.
- Testing the authorization of a user to perform a given function (TESTAUTH).
- Modifying information contained in the program status word (MODESET).
- Returning control by the dispatcher.

Major Changes from MVT: In VS2, a single task priority level may be identified at system generation or system initialization time for the dynamic dispatcher. Tasks within this priority level will be dispatched in a way that makes better use of the system's CPU and I/O resources. (The priority level cannot be identified also as a time-slicing group).

Contents Supervision

The contents supervisor locates requested programs, fetches the programs to virtual storage if necessary, and schedules their execution. The major contents supervision functions are requested by the LINK, LOAD, DELETE, XCTL, and ATTACH macro instructions. The functions are:

- Maintaining directories which describe all modules located in various portions of virtual storage.
- Searching directories for requested modules.
- Testing the status of modules to determine if they are available for use.
- Deferring requests for unavailable modules and restarting the deferred requests when the modules become available.
- Requesting the use of modules that are not in virtual storage.
- Scheduling execution of modules.

Program controlled interruption (PCI) permits the program to cause an I/O interruption during execution of an I/O operation. PCI provides an means of alerting the program of the progress of chaining during an I/O operation. It also permits programmed dynamic storage allocation.

PCI fetch is able to bring a program into storage with only one seek of the disk if:

- A buffer is always available for relocation dictionaries.
- No errors occur during the I/O operation.
- No cylinders are crossed while bringing in the program.

- The speed of the central processing unit allows PCI to modify the channel command word before it reaches the channel.

An additional WAIT and seek are required each time a buffer is not available. A seek is required each time an error occurs or a cylinder is crossed. If the speed of the central processing unit does not allow PCI to perform its function in time, the number of seeks needed by the standard fetch are required.

Major Changes from MVT: VS2 does not support scatter loading, TESTRAN attributes, or hierarchy support; these functions will be ignored if requested. Also, in VS2, transient areas are no longer needed because the pageable link pack area contains all those modules that formerly executed out of transient areas. PCI fetch is standard in VS2; in MVT, it was optional.

Virtual Storage Supervision

The virtual storage supervisor allocates address space within virtual storage. The supervisor routines service two macro instructions: GETMAIN (used to allocate virtual space) and FREEMAIN (used to free allocated virtual space):

- The GETMAIN routines service requests for virtual storage space, including requests for a region, space within a region, space within a local system queue area, and space within the system queue area.
- The FREEMAIN routines service all requests for making space available for reallocation, including requests for an entire region, space within a region, space within a local system queue area, and space within the system queue area.

Various supervisor services use areas in the system queue area and local system queue areas called quickcells. During execution of GETMAIN requests, these areas reduce the time required to allocate space for control blocks, and thus service the requests more quickly.

Major Changes from MVT: The virtual storage supervisor in VS2 perform the same function as the main storage supervisor in MVT. Basically, only internal changes have occurred as a result of the use of virtual storage. However, GETMAIN has been extended to service requests for virtual storage beginning on a page boundary. Also, subpools in VS2 are allocated in 4K-byte blocks.

Timer Supervision

The timer supervisor uses three timers, each with one microsecond precision, to service requests of programmers:

- The time-of-day clock is used to calculate the time of day and maintain the current date. This clock can only be set at IPL time.
- The clock comparator is used to measure elapsed time and schedule activity for specific times of the day. A value is placed in the clock comparator and when the time-of-day clock reaches that value, an external interruption occurs.
- The CPU timer is used to time tasks. A value is placed in the timer and when the timer is decremented to zero, an external interruption occurs.

Timer supervision is performed by four major routines:

- The TIME routine supplies the current date and time of day.
- The STIMER routine processes requests for interval timing based on task execution or real time.
- The TTIMER routine supplies the time remaining in a previously requested interval or it cancels previous timing requests.
- The timer second-level interruption handler processes timer interruptions.

Major Changes from MVT: In VS2, timer supervision routines support the CPU timer and the clock comparator, which are new features. The location 80 timer of MVT is not supported since the time-of-day clock is used instead.

Input/Output Supervision

The input/output (I/O) supervisor starts, terminates, and (where necessary) restarts activity on input/output devices. Its overall objective is to ensure that requested I/O operations are performed. The VS2 changes to I/O supervision are summarized in Figure 10.

Facility	Change
Starting I/O	Translation of virtual channel programs to real channel programs; page fixing.
Terminating I/O	None.
Restarting I/O	Translation of virtual channel programs to real channel programs; page fixing.

Figure 10. VS2 Changes to Input/Output Supervisor

Starting I/O Operations

Two parts of the control program are normally involved in starting I/O operations: access method routines and the I/O supervisor.

Each access method routine prepares information required by the I/O supervisor to start I/O operations. It gathers information used to initiate the I/O operations and places the information in control blocks. It then issues an EXCP macro instruction, causing entry to the I/O supervisor.

The I/O supervisor determines if the I/O device associated with the operation is not busy, and if so, whether any channel associated with the device is not busy. When both the device and an associated channel are not busy, the I/O supervisor prepares for the execution of channel programs and issues a START I/O instruction to initiate the operation.

The I/O supervisor also handles EXCPVR requests. These requests reduce the time required to initiate an I/O operation because the I/O supervisor does not perform page fixing, validity checking, or channel program translation; it is assumed that these functions have been provided for by the user. EXCPVR requests can only be issued by programs executing in the supervisor state, programs operating under protection key zero, or programs authorized by APF. In addition, the virtual storage access method (VSAM) and the telecommunications access method (TCAM) issue EXCPVR requests; however, programs using VSAM or TCAM do not have to be authorized themselves.

The I/O supervisor must distinguish between two classes of channel programs:

- Channel programs that will run without address translation. These channel programs are submitted for execution from programs executing in nonpageable dynamic storage.
- Channel programs that will not run without address translation. These channel programs are submitted for execution from programs executing in pageable dynamic storage. The I/O supervisor provides the necessary translation by building another copy of the channel program. This copy contains real storage addresses instead of virtual storage addresses, and takes into account discontiguous pages frames in real storage. In addition, the I/O supervisor fixes all required pages for the duration of the I/O operation. (Translation of virtual storage channel programs to real storage channel programs is required because channels do not use the segment and page tables for translation.)

Major Changes from MVT: The I/O supervisor in VS2 translates virtual storage channel programs to real storage channel programs; this feature was not needed in MVT. Also, the I/O supervisor performs page fixing.

Terminating I/O Operations

I/O operations terminate either normally because the operation is completed, or abnormally because an error is detected. When an I/O operation terminates, an I/O interruption occurs, causing CPU control to be passed first to the I/O interruption handler and then to the I/O interruption supervisor portion of the I/O supervisor to process the interruption.

The I/O supervisor posts the completion of the I/O operation, schedules error routines when the operation terminated abnormally, and, if possible, starts another I/O operation on the channel. Then the I/O interruption supervisor returns control to the interruption handler.

Major Changes from MVT: None.

Restarting I/O Operations

When an error occurs during I/O activity, the I/O supervisor invokes an I/O error routine which records the error on the SYS1.LOGREC data set and begins a cycle of restarts. The cycle continues either until the error is corrected or until it is declared to be permanent (uncorrectable).

An error is considered to be corrected by an I/O error routine when no errors occur during a retry of the I/O activity. When an error is corrected, the I/O supervisor continues processing normally as if no error had been found.

I/O error routines count the number of retries and indicate that the error is permanent when the maximum allowable number of retries has been reached. When a permanent error is found none of the related requests for the involved data set are started.

Major Changes from MVT: The I/O supervisor in VS2 translates virtual storage channel programs to real storage channel programs; this feature was not needed in MVT. Also, the I/O supervisor performs page fixing.

Data Management

Data management is a major function of the operating system that involves organizing, cataloging, storing, retrieving, and maintaining data. The data management routines are primarily responsible for moving information between virtual storage and external storage. Figure 11 provides a summary of the VS2 changes to the data management facilities.

Facility	Change
Access methods	Standard support of QSAM, BISAM, QISAM, and BDAM (previously optional); optional support of new VSAM (see the chapter "Options"); QTAM not supported; support of chained scheduling only in nonpageable storage (BSAM and QSAM).
Catalog management	None.
Direct access device space management (DADSM)	None.
I/O support	DEB validity checking performed.
Shared direct access storage devices (shared DASD)	None. (See the chapter "Options".)
Direct access volume serial number verification	Now standard (previously optional).

Figure 11. VS2 Changes to Data Management

Standard Access Methods

Access methods are techniques for moving data between virtual storage and input/output devices. In VS2, there are six standard access methods, each of which pairs a data set organization with a retrieval technique. The choice of which access method to use depends upon which best suits a particular application or installation.

VS2 data sets can be organized in four ways:

- Sequential: Records are arranged in physical sequence, and are usually read or updated in the same order in which they appear. This organization is used for all magnetic tapes, but may also be selected for direct access devices. Punched tape, punched cards, and printed output are considered to be sequentially organized.

  In the sequential organization, individual records cannot be located quickly. Also, records usually cannot be deleted or added easily unless the entire data set is rewritten. This organization is generally used when most records are processed each time the data set is used.
- Indexed Sequential: Records are arranged in collating sequence on the tracks of a direct access volume according to a key that is part of every record. The location of each record is computed through the use of indexes maintained by the system. This organization permits direct as well as sequential access to any record.

  In this organization, since the system has control over the location of the individual records, the user needs to do very little input/output programming. Also, since a separate area of the data set is set aside for added records, the data set need not be rewritten to accommodate new records.
- Direct: Records are arranged in random order on a direct access volume. Each record is stored or retrieved directly with addressing as specified by the user. This organization is generally used for data sets whose characteristics do not permit the use of sequential or indexed sequential organization, or for data sets where the time required to locate individual records must be kept to an absolute minimum.

  In this organization, the user is largely responsible for the programming required to locate the records.
- Partitioned: This organization has characteristics of both the sequential and the indexed sequential organizations. Independent groups of sequentially organized data, called members, are in direct access storage. Each member has a unique name stored in a directory that is part of the data set and contains the location of the member's starting point.

Partitioned organization is used mainly to store programs, subroutines, and tables. As a result, partitioned data sets are often referred to as libraries.

VS2 data access techniques are divided into two categories:

- Queued: This technique provides GET and PUT macro instructions to handle individual records. It offers a maximum amount of automatic input/output facilities. It may be used only to retrieve records in a sequential order (for example, records on magnetic tape).

  The GET and PUT macro instructions cause automatic blocking and deblocking of the records stored and retrieved. Look-ahead buffering and synchronization of input and output operations with CPU processing are automatic features of this technique.
- Basic: This technique provides READ and WRITE macro instructions to handle blocks (not records). It places some of the responsibility for data handling on the programmer but gives him more control of input/output operations. It is used when the operating system cannot predict the sequence in which the records are to be processed, or when some or all of the automatic functions performed by the queued access technique are not desired.

  Since READ and WRITE macro instructions process blocks, the blocking and deblocking of records is the responsibility of the programmer. Although look-ahead buffering and synchronized scheduling are not automatically included in this technique, macro instructions are provided to perform these functions.

Following are brief descriptions of the six access methods that are standard in VS2; Figure 12 summarizes their characteristics. Optional access methods that are available in VS2 (telecommunications access method, virtual storage access method, basic telecommunication access method, and graphics access method) are described in the chapter "Options".

Organization / Access Method / Characteristics	Sequential		Partitioned	Indexed Sequential			Direct
	QSAM	BSAM	BPAM	QISAM		BISAM	BDAM
				LOAD	SCAN		
Primary macro instructions	GET PUT PUTX	READ WRITE	READ WRITE FIND STOW	PUT	SETL GET PUTX	READ WRITE	READ WRITE
Synchronization of program with I/O	Automatic	CHECK*	CHECK*	Automatic	Automatic	WAIT*	WAIT*
Record formats transmitted	Logical - F,V Block - V	Block - F,V,U	Block (part of member) - F,V,U	Logical - F,V	Logical - F,V	Logical - F,V	Block - F,V,U
Buffer creation and construction	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic	BUILD*, GETPOOL*, Automatic
Buffer techniques	Automatic, Simple, Exchange	GETBUF*, FREEBUF*	GETBUF*, FREEBUF*	Automatic, Simple	Automatic, Simple	GETBUF*, FREEBUF*, Dynamic FREEDBUF*	GETBUF*, FREEBUF*, Dynamic FREEDBUF*
Transmittal modes (work area/buffer)	Move, Data, Locate, Substitute			Move, Locate	Move, Locate		

Note: * denotes a macro instruction.

Figure 12. Summary of Standard VS2 Access Method Characteristics

Basic Sequential Access Method (BSAM)

BSAM can be used for storing or retrieving data blocks arranged sequentially on sequential access or direct access devices. (See descriptions of basic and sequential above.)

Major Changes from MVT: In VS2 BSAM, chained scheduling is only available to programs executing in nonpageable dynamic storage. Requests for chained scheduling from programs executing in pageable regions will be ignored, and normal scheduling will be substituted.

Queued Sequential Access Method (QSAM)

QSAM is an extended version of BSAM where a queue is formed of input data awaiting processing or output data awaiting transfer to auxiliary storage or an output device. (See descriptions of queued and sequential above.)

Major Changes from MVT: In VS2 QSAM, chained scheduling is only available to programs executing in nonpageable dynamic storage. Requests for chained scheduling from programs executing in pageable regions will be ignored, and normal scheduling will be substituted.

Basic Indexed Sequential Access Method (BISAM)

BISAM is used in one form to directly retrieve or update particular blocks of a data set on a direct access device. An index, which is used to locate the data set, is stored with the data set. Other forms of this method can be used to store or retrieve sequential blocks of the same data set. (See descriptions of basic and indexed sequential above.)

Major Changes from MVT: None.

Queued Indexed Sequential Access Method (QISAM)

QISAM is an extended version of BISAM where a queue is formed of input data awaiting processing or output data awaiting transfer to auxiliary storage or output device. (See descriptions of queued and indexed sequential above.)

Major Changes from MVT: None.

Basic Direct Access Method (BDAM)

BDAM is used to directly retrieve or update particular blocks of a data set on a direct access device. (See descriptions of basic and direct above.)

Major Changes from MVT: None.

Basic Partitioned Access Method (BPAM)

BPAM can be used to create program libraries in direct access storage for convenient storage and retrieval of programs. (See descriptions of basic and partitioned above.)

Major Changes from MVT: None.

Catalog Management

Catalog management routines maintain the collection of data set indexes (the catalog) that is used by the control program to locate volumes. The catalog management routines also locate the cataloged data sets.

The catalog, itself a data set (SYSCTLG), resides on one or more direct access volumes. It contains indexes that relate data set names to the serial numbers and device types of the volumes containing the data sets.

In maintaining the catalog, catalog management routines create and delete indexes, and add or remove entries. To locate a data set, catalog management routines search through the indexes for the index entry containing the last part of the qualified name of the data set.

The catalog management routines are used primarily by the job scheduler and the IEHPROGM utility program, although they can be used by any processing program:

- The scheduler invokes the catalog management routines during the initiation and termination of a job step. During initiation, a catalog management routine locates cataloged data sets. During termination, a catalog management routine may catalog or uncatalog data sets referred to during the job step and specified for the catalog.
- The IEHPROGM utility program invokes catalog management routines to create and delete indexes, and to add or remove entries. The IEHPROGM program does not locate data sets.
- Processing programs can invoke the catalog management routines via the CATALOG, INDEX, and LOCATE assembler language macro instructions. These macro instructions provide access to all the catalog management routines.

Major Changes from MVT: None.

Direct Access Device Space Management (DADSM)

Direct access device space management (DADSM) consists of routines that allocate space on direct access volumes to data sets. The routines are used primarily by job management routines during the initiating of job steps when space is obtained for output data sets. They are also used by other data management routines for increasing the space already assigned to a data set, and for releasing space no longer needed.

The DADSM routine controls allocation of space through the volume table of contents (VTOC). The VTOC is built when a volume is initialized by the direct access storage device initialization (IBCDASDI) utility program. The VTOC indicates the current usage of the space on the volume.

When space is needed on a volume, the DADSM routines check the VTOC for enough contiguous, available tracks to satisfy the request. If there are not enough contiguous tracks, the request is filled using up to five noncontiguous groups of free tracks.

Major Changes from MVT: None.

Input/Output Support

Input/output (I/O) support routines perform three functions associated with I/O operations:

- Opening a data control block before a data set is read or written.
- Closing a data control block after a data set has been read or written.
- Processing end-of-volume (EOV) conditions when an end-of-volume or end-of-data (EOD) set condition occurs during an I/O operation.

Open Processing

Before access can be gained to a data set, the data control block (DCB) for that data set must be opened by means of an OPEN macro instruction. When a processing program issues an OPEN macro instruction, the Open routine of the control program performs:

- Volume mounting and verification.
- Merging of data set attributes from the DD statement and the data set label into the control blocks.
- Determination of access method routines.

Major Changes from MVT: Open processing in VS2 performs DEB validity checking.

Close Processing

After processing has been completed for a data set, the data control block (DCB) for that data set must be closed by means of a CLOSE macro instruction. When a processing program issues a CLOSE macro instruction, the Close routine of the control program performs:

- Input and output label processing.
- Volume disposition.
- Restoration of data control block to its original condition by removing the information that was merged from the DD statement.

Major Changes from MVT: Close processing in VS2 performs DEB validity checking.

End-of-Volume Processing

End-of-volume (EOV) processing is performed when end-of-data set or end-of-volume conditions occur during I/O operations on sequentially organized data sets. When a routine of a sequential access method encounters a tape or file mark (end-of-data set) or an end-of-volume condition, the routine issues an SVC instruction to pass control to the EOV routine.

Major Changes from MVT: None.

Direct Access Volume Serial Number Verification

Direct access volume serial number verification is standard in VS2. The volume serial number of a direct access device is checked after an unsolicited device-end interruption condition has been corrected and the volume has been put back online again.

When an unsolicited device-end interruption is received from a direct access device, the I/O supervisor ensures that the volume serial number of the mounted volume agrees with the volume serial in the unit control block (UCB).

Major Changes from MVT: In VS2, direct access volume serial number verification is a standard facility; in MVT, it was optional.

Recovery Management

Recovery management facilities gather information about hardware reliability and allow retry of operations that fail because of CPU, I/O device, or channel errors. The facilities of VS2 that record the environment of the system at the time of a machine malfunction and attempt to recover from the malfunction are the machine check handler and the channel check handler; the facility that attempts recovery from various I/O errors is dynamic device reconfiguration. Figure 13 provides a summary of the VS2 changes to the recovery management programs.

Facility	Change
Machine check handler (MCH)	Now standard (previously optional); enhanced MODE command.
Channel check handler (CCH)	Now standard (previously optional).
Dynamic device reconfiguration (DDR)	Now standard (previously optional); system residence device and page data set not supported.
Alternate path retry (APR)	None. (See the chapter "Options".)

Figure 13. VS2 Changes to Recovery Management

Machine Check Handler (MCH)

Machine check handler (MCH) is a standard facility of VS2 that attempts to reduce lost computing time due to machine malfunctions.

Recovery from machine malfunctions is initially attempted by the hardware instruction retry (HIR) and error correction codes (ECC) facilities of the machine. If the machine recovery attempts are unsuccessful, a machine check interruption will occur. MCH then analyzes the data and attempts to keep the system as fully operational as possible. It may:

- Attempt to repair malfunctions and, if possible, resume operation, leaving no adverse effects on the system.
- Attempt to terminate affected tasks and resume operation.
- Isolate the failure to a page frame in real storage, and mark the page frame as invalid or unavailable for use by the paging supervisor.
- Place the system in a wait state.

In all cases, whether or not recovery is successful, MCH constructs records of the error environment on the SYS1.LOGREC data set and issues diagnostic messages to the operator.

Major Changes from MVT: In VS2, MCH is a standard facility; in MVT, it was optional. In VS2, there is an enhanced MODE command that is used for all machine models. The MCH facility in VS2 has been expanded to allow MCH to mark a page frame unavailable.

Channel Check Handler (CCH)

Channel check handler (CCH) is a standard facility of VS2 that allows the user to recover from errors in the execution of channel programs. CCH receives control from the I/O supervisor when a channel data check, channel control check, or interface control check occurs.

For all three checks, it provides the operator with information about channel errors that enables him to keep statistics about the channel or help bring about recovery from system termination conditions.

In addition, for channel control checks and interface control checks, it provides the device-dependent error recovery procedures (ERPs) of the I/O supervisor with information needed to attempt a retry of a channel operation that has failed.

Major Changes from MVT: In VS2, CCH is a standard facility; in MVT, it was optional.

Dynamic Device Reconfiguration (DDR)

Dynamic device reconfiguration (DDR) is a standard facility of VS2 that allows the system and the user to circumvent an I/O failure, if possible, by moving a demountable volume from one device to another. The device swap is accomplished without abnormal termination of the affected job and without another initial program load (IPL).

A request to move a volume may be initiated by either the system or the operator. The system requests DDR after a permanent (uncorrectable) I/O error has occured. The operator may request DDR at any time by issuing the SWAP command. He may substitute one device for another, or simply interrupt processing on a device to carry out cleaning procedure.

Major Changes from MVT: In VS2, DDR is a standard facility; in MVT, it was optional. In VS2, DDR for the system residence device is not supported. Also, DDR does not support the page data sets.

Standard Support Programs

In addition to the job management, task management, input/output supervision, data management, and recovery management components of the system control program, VS2 contains a number of other standard support programs that are necessary to run the system. Figure 14 provides a summary of the VS2 changes to these programs.

Facility	Change
OS/VS2 assembler	New assembler with improved diagnostics and extended language capability.
Linkage editor F and loader	New control statements and parameters to order and align CSECTs and common areas, and to designate APF codes.
Utilities	IEHATLAS - restricted by APF; IEHPROGM - restricted by APF; IEHDASDR - restricted by APF, and automatically invoked by AMDSADMP; IEBCOPY - supports new LOAD/UNLOAD functions for library distribution.
Dynamic support system (DSS)	New system debugging tool for authorized maintenance personnel.
Missing interruption checker	Now standard (was previously optional).
Online test executive program (OLTEP)	Executes in nonpageable storage, except for logout analysis; now standard (previously optional).
Problem determination	None.
Reliability data extractor (RDE)	None. (See the chapter "Options".)
Service aids	AMAPTFLE - supports independent component releases; AMASPZAP - restricted by APF.
Storage dumps	Provides dumps of real and/or virtual storage; new DSS dump facility.

Figure 14. VS2 Changes to Standard Support Programs

OS/VS2 Assembler

The OS/VS2 assembler is a standard facility that translates source statements into machine language, assigns virtual storage locations to instructions and other elements of the program, and performs auxiliary assembler functions designated by the programmer. (The auxiliary functions assist the programmer in such areas as checking and documenting programs, generating macro instructions, and controlling the assembler itself.)

The output of the assembler program is the object program, a machine-language translation of the source program. The assembler produces a printed listing of the source statements and object program statements, and additional information useful to the programmer in analyzing his program. The object program is in the format required by the linkage editor.

The OS/VS2 assembler is the only language translator distributed with the VS2 system control program.

Major Changes from MVT: This facility is not available in MVT. It provides improved diagnostics and extended language capability over assembler F available in MVT.

Linkage Editor F and Loader

The linkage editor F and the loader are standard facilities of VS2 that are used to prepare the output of language translators for execution.

Linkage editor processing is a necessary step that follows source program assembly. Input to the linkage editor may consist of a combination of object modules, load modules, and control statements. The primary purpose of the linkage editor is to combine and edit these modules into a single load module that can be brought into virtual storage by program fetch and then executed.

The loader combines the basic editing and loading functions of the linkage editor and program fetch in one job step. It prepares the executable program in virtual storage and passes control to it prior to execution. The loader is designed for high-performance loading of modules that do not require the special processing facilities of the linkage editor and program fetch. It does not produce load modules for program libraries.

Major Changes from MVT: In VS2, several new control statements and parameters are available to order CSECTs and common areas in a load module, to align CSECTs and common areas on a page boundary, and to designate the authorization code for APF. In VS2, linkage editor E is not distributed with the system control program; that is, only linkage editor F is distributed with VS2.

Utilities

The VS2 utilities are standard programs that organize and maintain data. They are divided into three categories: system utilities, data set utilities, and independent utilities. Figure 15 lists these utilities.

System Utilities	Data Set Utilities	Independent Utilities
IEHATLAS	IEBCOMPR	IBCDASDI
IEHDASDR	IEBCOPY	IBCDMPRS
IEHINITT	IEBDG	ICAPRTBL
IEHLIST	IEBEDIT	
IEHMOVE	IEBGENER	
IEHPROGM	IEBISAM	
IFHSTATR	IEBPTPCH	
	IEBTCRIN	
	IEBUPDTE	

Figure 15. VS2 Utilities

System Utilities

System utility programs manipulate collections of data and system control information. The programs are executed or invoked through the use of job control statements and utility control statements.

IEHATLAS

The IEHATLAS program is used when a defective track is indicated by a data check or missing address marker condition. It locates and assigns an alternate track to replace the defective track. Usable data records on the defective track are retrieved and transferred to the alternate track. The bad record from the defective track is replaced on the alternate by a correct copy. (The correct copy must be provided by the user.)

Major Changes from MVT: In VS2, use of this utility is restricted through the authorized program facility.

IEHDASDR

The IEHDASDR program prepares direct access volumes for VS2 use and ensures that any permanent hardware errors (i.e., defective tracks) encountered on direct access volumes will not seriously degrade the performance of those volumes. During generation of the stand-alone dump service aid program via the AMDSADMP macro instruction, IEHDASDR is invoked to dump the program on a residence volume.

In addition, the IEHDASDR program can write the entire contents or portions of a direct access volume onto a volume or direct access device type, onto a magnetic tape volume or volumes, or onto a system output device. Data that is written onto a magnetic tape volume is arranged so that it can subsequently be "restored" to its original organization by the IEHDASDR program. The direct access device types supported by the IEHDASDR program are IBM 2305 Fixed Head Storage Models 1 and 2, IBM 2314 Direct Access Storage Facility, IBM 2319 Disk Storage, and IBM 3330 Disk Storage.

IEHDASDR has been enhanced to allow the user to construct his own IPL program, and have it (and all IPL records necessary to initialize it) written on track 0 of a supported device.

The IEHDASDR program can be used to:

- Check tracks, assign alternate tracks for defective tracks, and perform initialization and formatting functions to make a direct access volume suitable for VS2 use. Surface analysis does not apply for the IBM 3330 Disk Storage.
- Initialize and format direct access devices.
- Assign alternate tracks for specified defective or questionable tracks on disk volumes.
- Create a backup or transportable copy of a direct access volume, or list the contents on a system output device.
- Copy data from a magnetic tape volume onto a direct access volume.

Major Changes from MVT: In VS2, use of this utility is restricted through the authorized program facility. Also, in addition to the enhancements noted above, the AMDSADMP service aid program now creates the job stream for the IEHDASDR program.

IEHINITT

The IEHINITT program places VS2 volume labels written in EBCDIC, in BCD, or in ASCII (American Standard Code for Information Interchange) onto any number of magnetic tapes mounted on one or more tape drives. Each volume label created by the program contains:

- A standard volume label with a user specified serial number and user identification.
- A dummy header label (an 80-byte record containing HDR1 and an ASCII character).
- A tapemark.

Major Changes from MVT: None.

IEHLIST

The IEHLIST program can be used to list:

- Entries in a catalog.
- Entries in the directory of one or more partitioned data sets.
- Entries in a volume table of contents, in edited or unedited form.

Major Changes from MVT: None.

IEHMOVE

The IEHMOVE program moves or copies logical collections of VS2 data.

The program can be used to move or copy:

- A data set residing on one or as many as five volumes.

- A group of cataloged data sets.
- A catalog, or portions of a catalog.
- A volume of data sets.

The scope of a basic move or copy operation can be enlarged by:

- Merging members from two or more partitioned data sets.
- Including or excluding selected members.
- Renaming moved or copied members.
- Replacing selected members.

Major Changes from MVT: None.

IEHPROGM

The IEHPROGM program provides facilities for modifying system control data and for maintaining data sets at an organizational level.

The program can be used to:

- Scratch a data set or a member.
- Rename a data set or a member.
- Catalog or uncatalog a data set.
- Build or delete an index or an index alias.
- Connect or release two volumes.
- Build and maintain a generation data group index.
- Maintain data set passwords.

Major Changes from MVT: In VS2, use of this utility is restricted through the authorized program facility.

IFHSTATR

The IFHSTATR program selects, formats, and writes information from type 21 (error statistics by volume) records. The records exist on the IFASMFDP tape. They can also be retrieved directly from SYS1.MANX or SYS1.MANY data sets (on a direct access storage device); however, the IFHSTATR program does not clear the SYS1.MANX or SYS1.MANY data sets and therefore does not make them available for additional records.

Major Changes from MVT: None.

Data Set Utilities

Data set utility programs manipulate partitioned, sequential, or indexed sequential data sets. Data ranging from fields within a logical record to entire data sets can be manipulated. The programs are executed or invoked through the use of job control statements and utility control statements.

IEBCOMPR

The IEBCOMPR program compares two identically organized data sets at the logical record level. Data sets to be compared can be either sequential or partitioned.

The program can:

- Verify a back-up copy of a sequential or partitioned data set.
- Verify portions of records within a sequential or partitioned data set.

User exits are provided at appropriate places for optional user routines that process user labels, handle error conditions, and modify source records.

Major Changes from MVT: None.

IEBCOPY

The IEBCOPY program can copy or merge partitioned data sets. Specified members of partitioned data sets can be selected for, or excluded from a copy operation. The program can also compress a data set in place, optionally replace identically named members on data sets, or optionally rename selected members.

The IEBCOPY program automatically lists the number of unused directory blocks and unused tracks available for member records in the output partitioned data set. By means of the LIST=NO operand the program can be made to suppress the names of copied members listed by input partitioned data set.

Major Changes from MVT: In VS2, this utility supports LOAD/UNLOAD functions which can be used for VS2 library distribution.

IEBDG

The IEBDG (data generator) program provides a pattern of test data to be used as a programming debugging aid. A (test) data set, containing records of any format, can be created by utility control statements, with or without input data.

An optional exit is provided to a user routine that can monitor each output record before it is written. Sequential, indexed sequential, and partitioned data sets can be used for input or output.

Major Changes from MVT: None.

IEBEDIT

The IEBEDIT program can create an output data set containing selected jobs or job steps. At a later time, the data set can be used as an input stream for job processing.

Input to the IEBEDIT program is obtained from a sequential data set. The input data set can reside on any input device supported by VS2 (e.g., magnetic tape, direct access, or card reader). The program can select JOB statements, JOBLIB statements, and job steps from the input data set and can include them in the output data set.

Major Changes from MVT: None.

IEBGENER

The IEBGENER program can copy a sequential data set or a partitioned member, or it can create a partitioned data set from a sequential data set or a partitioned member. The program expands existing partitioned data sets by creating partitioned members and merging them into the data set to be expanded. The program can also reblock or change the logical record length of a data set, or create user labels on sequential output data sets.

The IEBGENER program provides optional editing facilities with all applications. In addition, it provides user exits at appropriate places for routines that process labels, manipulate input data, create keys, and handle uncorrectable input/output errors.

Major Changes from MVT: None.

IEBISAM

The IEBISAM program can copy an indexed sequential data set directly from one direct access volume to another. Alternatively, the IEBISAM program can reorganize an indexed sequential data set into a sequential data set and place that data set on a direct access or on a magnetic tape volume. The data set is in a form that can be subsequently loaded; that is, it can be converted back into an indexed sequential data set.

Optionally, the IEBISAM program can be used to print the records of an indexed sequential data set.

Major Changes from MVT: None.

IEBPTPCH

The IEBPTPCH program prints or punches all, or selected portions, of a sequential or partitioned data set. Records can be printed or punched to meet either standard specifications or user specifications.

The IEBPTPCH program provides optional editing facilities. In addition, user exits are provided at appropriate places for routines that process labels and/or manipulate input or output records.

Major Changes from MVT: None.

IEBTCRIN

The IEBTCRIN program reads input from the IBM 2495 Tape Cartridge Reader (TCR), edits the data as specified by the user, and produces a sequentially organized output data set. The input consists of cartridges written by either the IBM Magnetic Tape SELECTRIC Typewriter (MTST) or the IBM 50 Magnetic Data Inscriber. An input data set (one or more cartridges) must consist of either all MTST cartridges or all Magnetic Data Inscriber cartridges.

The program can construct records from the stream of data bytes read sequentially from the Tape Cartridge Reader. The user has the option of gaining temporary control (via a user-supplied exit routine) to process each logical record.

The output produced by the program is a sequential data set that can be written on any QSAM-supported output device (for example, a system output device, a magnetic tape volume or a direct access volume). A second sequential data set may be produced for error records.

Major Changes from MVT: None.

IEBUPDTE

The IEBUPDTE program incorporates both IBM- and user-generated source language modifications into sequential or partitioned data sets. The input and output data sets may contain blocked or unblocked logical records with record lengths of up to 80 bytes. Exits are provided at appropriate places for user routines that process header and trailer labels.

The program can:

- Add, copy, and replace members or data sets.
- Add, delete, replace, and renumber the records within an existing member or data set.
- Assign sequence numbers to the records of a member or data set.
- Convert sequential input into partitioned output or vice versa.

In general, the program may be used to:

- Create and update symbolic libraries.
- Incorporate changes to partitioned members or sequential data sets.
- Change the organization of a data set from sequential to partitioned or vice versa.

Major Changes from MVT: None.

Independent Utilities

Independent utility programs are used to prepare direct access devices for system use and to ensure that any permanent hardware errors incurred on a direct access device do not seriously degrade the performance of that device. They operate outside, and in support, of System/370. They do not support the IBM 3066 System Console, which is only used with the System/370 Models 165II and 168. The user controls the operation of independent utility programs through utility control statements. Since the programs are independent of the operating system, job control statements are not required.

IBCDASDI

The IBCDASDI (DASDI) program performs two separate functions: it initializes direct access volumes for use with the operating system, and assigns alternate tracks on non-drum, direct access storage volumes. A single job can initialize one volume or assign alternates for specified tracks on one volume. DASDI jobs can be performed continuously by stacking complete sets of control statements.

Major Changes from MVT: None.

IBCDMPRS

The IBCDMPRS (DUMP/RESTORE) program dumps and restores the data on direct access volumes. The contents of a direct access volume (all data except the home address and the count field of record zero (R0)) can be "dumped" onto IBM 2314 Direct Access Storage Facility, IBM 2319 Disk Storage, or IBM 3330 Disk Storage devices or onto magnetic tapes, and restored onto a direct access volume that resides on the same type of device as the source volume. Both the source volume and the volume onto which data is to be restored must have been initialized to System/370 specifications. This utility is useful for preparing transportable copies and backup copies of direct access volume contents.

Major Changes from MVT: None.

ICAPRTBL

The ICAPRTBL (IBM 3211 Buffer-Loader) program loads the universal character set (UCS) buffer and the forms control buffer (FCB) for a IBM 3211 Printer.

When the IBM 3211 Printer is assigned as the output portion of a composite console and an unsuccessful attempt has been made to IPL because the UCS and FCB buffer contains improper bit patterns, this utility can load the buffers so the system can be initialized by the IPL program.

Under normal circumstances, where an operable console printer-keyboard is available, the buffers will be loaded under control of the operating system.

Major Changes from MVT: None.

Reliability, Availability, Serviceability (RAS)

Reliability, availability, and serviceability (RAS) facilities consist of recovery and repair procedures that are designed to reduce the frequency and impact of system interruptions caused by hardware failures. The RAS facilities are designed to improve the reliability of the hardware, to increase the availability of the computing system, and to improve the serviceability of the system hardware components.

Dynamic Support System (DSS)

The dynamic support system (DSS) is a debugging tool that can be used by authorized maintenance personnel, such as an IBM program systems representative, to help identify and correct causes of software failures.

DSS has its own input/output capability, and has access to all of virtual storage as well as real storage. When running, it is stand-alone and has control of the system; however, DSS can return control to the VS2 control program for further operations without system restart processing. Since DSS takes control from the system on each activation, time dependencies cannot be maintained. Thus, DSS should not be used while a time-sensitive program (for example, a teleprocessing or time sharing task) is running. DSS could be used in development testing of these programs or in locating and repairing a critical problem which prevents the execution of these programs, but it should not be invoked during any time-dependent production run.

No permanent changes can be made by DSS. That is, any modification made to the system will not be carried over to the next IPL. Also, DSS cannot be used to modify itself, IPL, or NIP.

Communication with DSS is by commands entered through the integrated operator's console. The DSS language provides capabilities to:

- Display and alter real storage, virtual storage, and registers.
- Provide control for the program event recording (PER) hardware of System/370.
- Stop operation of the system at any instruction or PER interruption, perform maintenance functions (that is; identify and correct the causes of the failure), and resume operations.
- Save information within DSS for later use, or write out the information on tape or high-speed printers.
- Use tape or card readers for secondary command input devices.
- Write procedures that are used for reiterative command sequences.

Major Changes from MVT: This facility is not available in MVT.

Missing Interruption Checker

Missing interruption checker is a standard facility of VS2 that notifies the operator if a device-end or channel-end interruption is not received within a specified period of time. The absence of such interruptions may mean that a mount message has not been satisfied or that a device has malfunctioned.

Specific actions an operator may have to take depend upon the conditions he encounters. He may be required to ready a device on which a volume has been mounted, examine indicator lights on the device for abnormal signs, or terminate the job.

Major Changes from MVT: This facility is available in MVT only as a program temporary fix (PTF).

Online Test Executive Program (OLTEP)

The online test executive program (OLTEP) acts as an interface between the operating system and online test programs that are to be run. OLTEP schedules and controls the activities of the test programs. The programs can be used to:

- Test control units and input/output devices.
- Diagnose equipment malfunctions.
- Verify repairs.
- Test engineering changes.
- Perform periodic maintenance checks.

While OLTEP is being run, continuous communication is maintained with the operator. The programs can be run to obtain printed diagnostic information, or they can be run to exercise a device while it is being tested with hardware testing equipment.

Major Changes from MVT: In VS2, OLTEP is a standard facility; in MVT, it was optional. In VS2, OLTEP executes in pageable storage only when it is invoked to do logout analysis (LOA); otherwise, OLTEP can be executed only in nonpageable storage.

Problem Determination

Problem determination can be thought of as an activity that is required to identify a failing hardware unit or program and determine who is responsible for maintenance. It helps to establish whether a failure is in the system control program, in a software component added to the system by the user (type I programs and program products), or in the user's application programs.

Problem determination is accomplished with procedures specified by IBM. In some cases, these procedures may be initiated by a message or code issued by the system control program; the message or code, in turn, may direct the user to take certain definitive actions such as running a service aid or utility program. In all cases, a definite action will be specified and should be taken before a service call is to be placed to IBM for either hardware or programming assistance.

Major Changes from MVT: None.

Service Aids

The service aids help in diagnosing and repairing system or application program failures.

AMAPTFLE

The AMAPTFLE service aid is a problem program that is used to apply program temporary fixes (PTFs). AMAPTFLE has two functions:

- An application function generates control statements and applies PTFs in one operation.
- A generate function produces the JCL and control statements needed to apply PTFs. The JCL must be executed in a later, separate step.

Major Changes from MVT: In VS2, AMAPTFLE will support independent component releases.

AMASPZAP

The AMASPZAP service aid is a problem program that allows the user to inspect and modify data at the time a problem is diagnosed. Various combinations of AMASPZAP control statements enable the user to:

- Inspect and modify instructions and data in any load module that exists as a member of a partitioned data set.
- Inspect and modify data in a specific data record that exists in a data set residing on a direct access device.
- Dump an entire data set, a specific member of a partitioned data set, or any portion of a data set residing on a direct access device.
- Update the system status index (SSI) in the directory entry for a load module.

Major Changes from MVT: In VS2, the use of this service aid is restricted through the authorized program facility.

AMBLIST

The AMBLIST service aid is a problem program that assists in problem determination by providing:

- Formatted listings of linkage editor/loader input and output (that is, object and load modules).
- A cross-reference listing of symbolic references within a load module without re-processing that module through the linkage editor.
- A formatted listing of all information in a module's control section (CSECT) identification records (IDR).
- A formatted listing of all IDR indications of program modifications for a load module or library.
- A map of a system's nucleus.
- A map of the active link pack area (LPA) or the total LPA.

Major Changes from MVT: None.

AMDPRDMP

The AMDPRDMP service aid is a problem program that formats and prints the contents of the AMDSADMP output data set, the SYS1.DUMP data set, the TSO DUMP data set, and the GTF output data set. Users can control AMDPRDMP output with control statements. Some of the areas and traces that can be printed are:

- Link pack area map.
- QCB trace.
- System control blocks for all active tasks.
- Allocated pageable storage.
- Virtual storage ranges.
- Real storage ranges.
- Page data set dumps.
- System nucleus.
- GTF trace data.

Major Changes from MVT: In VS2, areas of virtual storage are printed; in MVT, only areas of main storage were printed.

AMDSADMP

The AMDSADMP macro instruction is used to generate a stand-alone service aid program that provides the user with a flexible, installation-tailored dump facility. It can generate one of the following types of dump programs:

- *Low-speed* - The low-speed dump program dumps the contents of real storage to a printer or tape. The dump output is translated and deblocked. Output which is directed to tape may be printed by the AMDPRDMP service aid or the IEBPTPCH utility program. (IPL of the program must be from a direct access volume.)
- *High-speed* - The high-speed dump program dumps the contents of real storage to tape. Optionally, it can also produce dumps of page data sets. The dump output is in large (2K), untranslated hexadecimal blocks. The AMDPRDMP service aid must be used to format, translate, and print the output. (IPL of the program may be from either a tape or a direct access volume.)

Major Changes from MVT: In VS2, the high-speed dump program can produce dumps of the page data set(s).

Generalized Trace Facility (GTF)

The generalized trace facility (GTF) service aid is a problem determination aid that can be used to trace software (system or problem program) behavior. GTF uses the monitor call (MC) instruction to detect occurrences of system events and to create trace records. Problem programs may also use a GTF macro instruction (GTRACE) to record problem program data in the trace data set.

GTF can single out those programming activities to be traced within the system, including I/O interruptions, program interruptions, and supervisor call interruptions.

The output from GTF can be a trace data set that is used with the edit function of the AMDPRDMP service aid. The AMDPRDMP service aid provides the capability to format specific trace activities. It operates as a problem program that can be called via the job control language.

Major Changes from MVT: None.

IFCDIP00

The IFCDIP00 service aid is a problem program that is used to:

- Initialize the SYS1.LOGREC data set during system generation.
- Reinitialize the SYS1.LOGREC data set if an error has resulted in the destruction of the SYS1.LOGREC header record and makes the data set unusable.
- Reallocate the SYS1.LOGREC data set to increase or decrease the space allocation.

This service aid is run and controlled by job control statements; no user or utility control statements are needed.

Major Changes from MVT: None.

IFCEREP0

The IFCEREP0 service aid is a problem program that is used to:

- Select and format environment records from the SYS1.LOGREC data set and write them to an output device.
- Select environment records from the SYS1.LOGREC data set and accumulate them on a history data set.
- Write the accumulated records on the history data set to an output device.
- Summarize the information contained in the records on the SYS1.LOGREC or history data sets.
- Process (edit, write, accumulate, and summarize) records produced on different machine models.

This service aid is run and controlled by job control statements; no user or utility control statements are needed.

Major Changes from MVT: None.

IMCOSJQD

The IMCOSJQD service aid is a problem program that produces a formatted copy of the contents of the job queue data set. It does not alter the status of the records to be written out. The user does not need to know the explicit address of the job queue data set, only the address assigned to the direct access device on which the volume containing the job queue is mounted.

When the program locates the job queue, the records are read and, either serially or selectively, identified by type and address, formatted, and written out. The job queue may be written out by job, job step, or in its entirety.

This service aid is run and controlled by job control statements; no user or utility control statements are needed.

Major Changes from MVT: None.

Storage Dumps

Storage dumps give the user a meaningful image of his programs. The routines available to provide dumps include:

- ABDUMP, which provides a formatted dump of selected areas of a problem program's virtual storage. ABDUMP is invoked by the SNAP and ABEND macro instructions.
- DSS DUMP, which allows the user of the dynamic support system (DSS) to dump both real and virtual storage. DSS DUMP is invoked by DSS.
- CONSOLE DUMP, started by the console operator, invokes SVC DUMP, described below. See the discussion of the DUMP command in the chapter "Compatibility".
- TSO DUMP, invoked by the time sharing option (TSO), invokes SVC DUMP, described below.
- SVC DUMP, provides a dump of selected areas of virtual storage (the nucleus, system queue area, local system queue area, region, and link pack area modules used). SVC DUMP is invoked by a key zero routine. The dump provided by SVC DUMP can be printed by the AMDPRDMP service aid.
- SADMP DUMP, which provides an installation-tailored dump of virtual storage, and can be printed by the AMDPRDMP service aid. SADMP DUMP is invoked by the AMDSADMP service aid.

Major Changes from MVT: In VS2, dumps are provided of real storage and/or selected areas of virtual storage; in MVT, dumps are for main storage only. Also, the DSS facility is not available in MVT.

Options

The items discussed in the preceding chapter are standard support programs of the VS2 system control program. In addition to the standard support programs, the user can add optional facilities to his system. There are two types of options: options included in the system during system generation and options included in the system after system generation.

Options Included During System Generation

Options included in the system during system generation are contained in the distribution library and added to the system during the IBM installation process. Figure 16 provides a list of these options and their changes from MVT.

Facility	Change
Alternate path retry (APR)	None.
Automatic priority group (APG)	New dispatching facility.
Automatic volume recognition (AVR)	None.
Device independent display operator console support (DIDOCS)/status display support (SDS)	None.
Reliability data extractor (RDE)	None.
Shared direct access storage devices (shared DASD)	None.
Time sharing option (TSO)	Support for swapping performed by VS2 paging supervisor; new use of virtual storage to support more TSO regions; new parameters on START and MODIFY commands; increased number of regions (from 14 to 42) for foreground jobs; parallel swapping allowed to four devices.
Time slicing	None.
Track stacking	None.
Access methods	Support for TCAM, VSAM, BTAM, and GAM.

Figure 16. VS2 Changes to Options

Time Sharing Option (TSO)

The time sharing option (TSO) of VS2 adds general purpose time sharing to the VS2 control program. The installation can dedicate the system to time sharing operations, or it can run concurrent time sharing and batch operations.

TSO adds a number of capabilities to the facilities already available through the VS2 control program:

- It gives users access to the system through a *command language* which is entered at *remote terminals* -- typewriter-like keyboard-printer or keyboard-screen devices connected through telephone or other communication lines to the computer.
- It gives those who may not be programmers the use of data entry, editing, and retrieval facilities.
- It makes the facilities of the operating system available to programmers at remote terminals to develop, test, and execute programs conveniently, without the job turnaround delays typical of batch processing. Both terminal-oriented and batch programs can be developed at terminals.
- It allows the management of an installation to dynamically control the use of the system's resources from a terminal.

- It creates a time-sharing environment for terminal-oriented applications. Some applications, such as problem-solving languages, terminal-oriented compilers, and text-editing facilities, are available as IBM program products. Installations can add others suited to their particular needs.

TSO consists of control routines, service routines and command processors, and a number of IBM program products. All the facilities available with TSO in MVT are available with TSO in VS2. VS2 enhances TSO in the following ways:

- The use of virtual storage makes it possible to run more foreground regions.
- The use of virtual storage allows the installation to place frequently used TSO commands or service routines in the pageable link pack area or TSO link pack area without reducing the amount of real storage available. When TSO is not active, the TSO LPA does not require address space.
- The paging supervisor, which does all swapping, uses external page storage for swapping, thereby eliminating the swap data set.

Differences in TSO for VS2

TSO in VS2 differs from TSO in MVT in several respects:

- The number of regions that can be assigned to foreground jobs is increased from 14 for TSO in MVT to 42 for TSO in VS2.
- The paging supervisor executes all swaps.
- The LSQA is no longer taken from the region; each TSO region is allocated one segment of LSQA.
- Swapping is now a process by which the valid pages (addressable pages) in a user's region are usually block paged to external page storage and always block paged from external page storage.
- The amount of external page storage required in VS2 must be sufficient to accomodate ('back up') the paging of background pages and the paging and swapping of foreground pages.
- Swapping can be directed to specific page data set devices. The parallel swapping capability that existed for TSO in MVT is expanded to allow swapping to as many as four devices. These can be the same devices that receive pages from background jobs.
- Background regions must have 100% backup in external page storage. For example, a background job that requests a 128K region is allocated 128K (32 pages) of external page storage as backup.
- Most TSO jobs do not continually need an amount of external page storage equivalent to the region sizes requested. Consequently, foreground jobs are backed up by an installation specified amount of external page storage that is a percentage of the total amount of external page storage required.

TSO in VS2 uses eight new parameters, including parameters to handle the assignment of external page storage and establish thresholds to guarantee that both background and foreground jobs have minimum amounts of external page storage. Figure 17 defines the parameters and Figure 18 shows the relationship between the TSOAUX and TSOMAX parameters in a system.

Parameter	When Specified	Default	Effect
TSOAUX	System generation, or system initialization, or START TSO, or MODIFY TSO	0 %	Specifies the minimum percentage of virtual storage dynamic area reserved for foreground jobs and backed up by external page storage. By implication, this parameter also specifies the maximum percentage of virtual storage dynamic area available for background jobs and backed up by external page storage.
TSOMAX	START TSO or MODIFY TSO	100 %	Specifies the maximum percentage of virtual storage dynamic area available for foreground jobs and backed up by external page storage. By implication, this parameter also specifies the minimum percentage of virtual storage dynamic area reserved for background jobs and backed up by external page storage.
BACKUP	START TSO or MODIFY TSO	100 %	Specifies the percentage of virtual storage dynamic area assigned to foreground jobs that will be backed up by external page storage.
SWAP	START TSO	None	Specifies up to 4 volume serial numbers of volumes containing page data sets to which swapping is to be directed. Parallel swapping is accomplished when two or more devices are specified. If no devices are specified, the primary page data set with the most page slots available will be used, or, if a secondary device is defined, it will be used.
NOSWAP	START TSO	NOSWAP	Specifies that the regular page data set is to be used for swapping. This parameter nullifies the SWAP parameter.
AUXLIST	START TSO or MODIFY TSO	None	Specifies that information concerning the use and availability of auxiliary storage is to be listed.
LPAR	START TSO	None	Specifies TSO LPA modules required. The START TSO command will be rejected unless the named modules are found in SYS1.LINKLIB or the TSO link pack area.
LPAF	START TSO	None	Specifies that, in addition to the processing for the LPAR parameter, the named TSO modules will be fixed in the TSO link pack area as long as TSO is active.

Figure 17. New TSO Operator Parameters for VS2

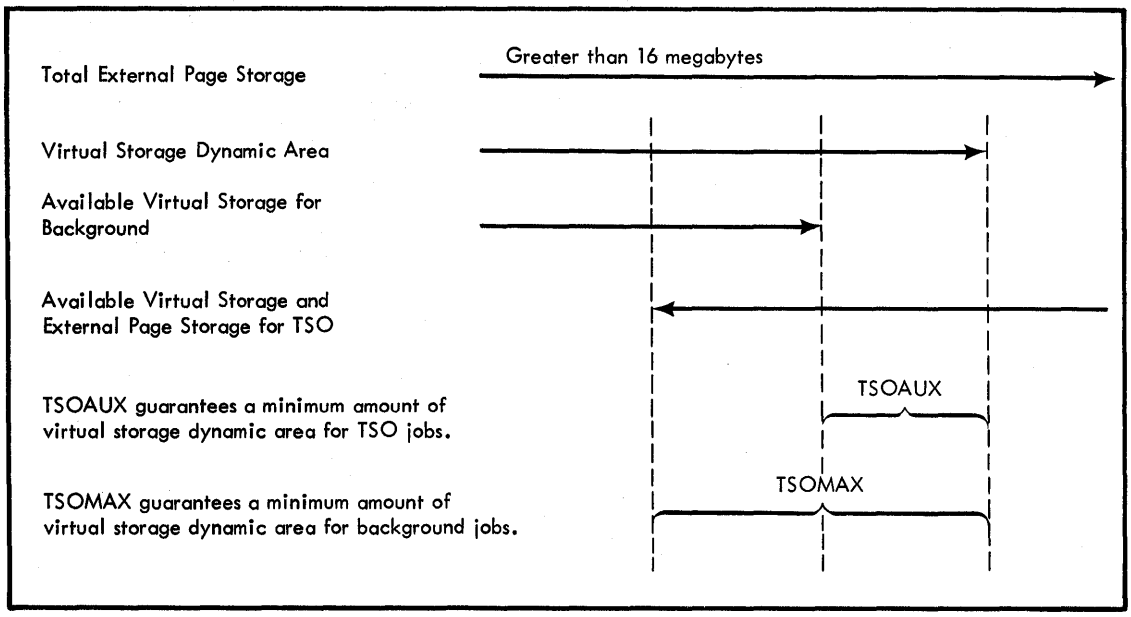


Figure 18. Comparison of TSOAUX and TSOMAX Parameters

Basic Preparation

The procedure to prepare TSO in VS2 is basically the same as it was in MVT. To prepare TSO, the installation must:

- Generate a system including TSO.
- Tailor a TSO-TCAM message control program.
- Define a member of SYS1.PARMLIB to contain TSO system parameters.
- Define LOGON procedures.
- Define the user attribute data set (UADS) to contain user identification records.

Major Changes from MVT: See the section "Differences in TSO for VS2" above. For a summary of all changes to the TSO operator commands, see "TSO Commands" in the chapter "Compatibility."

Automatic Volume Recognition (AVR)

Automatic volume recognition (AVR) is an option of VS2 that permits the operator to premount tape volumes on any available devices prior to the initiation of the job step that requires the volumes. The operator can therefore reduce the time lost in performing job setups.

If the operator does not premount tape volumes, AVR allows him to select devices he wishes to use during job step initiation. AVR also allows him to manually restart an I/O job without demounting volumes and then remounting the same volumes.

In VS2, the system attempts to balance channel loads by using the I/O load balancing algorithm to allocate data sets with nonspecific device requests. In some installations, however, a system slowdown may result if AVR causes one channel to be overloaded with most of the data handling.

Major Changes from MVT: None.

Device Independent Display Operator Console Support (DIDOCS)/Status Display Support (SDS)

Device independent display operator console support (DIDOCS)/status display support (SDS) is an option of VS2 that allows graphic display devices to be used as operator consoles. Its use can result in faster communication between the system and the operator than can be achieved with standard printer-keyboard or composite console devices. DIDOCS/SDS is not a standard facility of VS2; however, it will be included in the system when a graphic console is included. It provides the following advantages to the operator:

- He can respond to a message or enter a command while messages are being written to the screen.
- He is shown all action messages to be answered, and can delete any he no longer needs.
- He can use the light pen or cursor, when available, to delete messages and perform other display-oriented functions.
- He can obtain an in-line display or an out-of-line display within specified screen display areas.
- He can obtain a dynamic status display from a MONITOR ACTIVE command.
- He can initiate automatic command entry either by the use of the selector pen or from the program function keyboard (PFK) by an operator command.

Although multiple console support (MCS) is required for DIDOCS, MCS is standard in VS2, and need not be specified separately.

Major Changes from MVT: None.

Time Slicing

Time slicing is an option of VS2 that lets each task of a specified priority have control of the central processing unit (CPU) for a specified interval of time. Normally a task maintains control either until it is complete, until a higher-priority task becomes ready, or until it must wait for some event (such as an I/O operation). With time slicing, a group of tasks are allotted an interval of time which is divided among them.

When a member of the time-slice group has been active for the allotted length of time, it is interrupted and control is given to another member of the group which will, in turn, have control of the CPU for the same amount of time. In this way, all member tasks are given an equal slice of CPU time and no task within the group can monopolize the CPU. Only tasks in the group are time-sliced, and they are time-sliced only when the priority level of the group is the highest priority level that has a ready task.

When a time-sliced task loses control prior to the expiration of its interval, the remainder of the interval is not saved. If the task was waiting for an event, the next task is dispatched. If a task of higher priority became ready, the higher priority task is dispatched when control is returned to the group, not the task that lost control.

The priority level identified for the time slice group cannot also be identified for the automatic priority group (APG).

Major Changes from MVT: None.

Shared Direct Access Storage Devices (Shared DASD)

Shared direct access storage devices (shared DASD) is an option of VS2 that enables independent operating systems (VS1, VS2, MVT, and MFT) to share common data on a shared direct access storage device.

The sharing is made possible by a two-channel switch that allows the control unit for the device to be switched between two channels, each from a different system. (A 4-channel switch is supported by the IBM 3330 Disk Storage device.) The VS2 control program

protects data being used in one central processing unit (CPU) from modification by a program in the other CPU, and minimizes access-arm contention between the devices.

Shared DASD offers the following advantages:

- Reduction of time spent in data set creation and maintenance, and a reduction in space requirements (whenever a single data set can service multiple users).
- Flexibility in scheduling jobs on CPUs that have access to the common data base.
- Availability of a system through the flexibility of job scheduling in the event of a malfunction in any CPU.
- Increase in the amount of work processed, since two CPUs are using common data at the same time.

In a shared DASD environment, volume handling is conducted in parallel on both sharing systems. Therefore, the operator's responsibility is increased, and good communication between operators of the systems must be maintained.

For a listing of the system data sets and libraries that can be shared, see System Data Sets in the chapter "Compatibility".

Major Changes from MVT: None.

Alternate Path Retry (APR)

Alternate path retry (APR) is an option of VS2 that ensures that an alternate path to a device is tried (whenever possible) when a failing path is detected. Until a retry is successful, APR places any failing paths offline to prevent further retries from being initiated on these paths. When retry succeeds, APR restores all of the original paths, and resumes normal system operation.

APR also permits the operator to vary a path to a device online or offline. The operator, however, cannot vary the last remaining path to a device offline, nor can he vary teleprocessing paths or paths to shared direct access storage devices.

Major Changes from MVT: None.

Reliability Data Extractor (RDE)

The reliability data extractor (RDE) is an option of VS2 that generates system initialization (IPL) and system termination (EOD) records, and collects them on SYS1.LOGREC. The IPL record contains the reason for restarting the system and names the type of equipment or program (if any) that is responsible for the restart; the EOD record defines the time span of processing for the restart. The IPL and EOD records can be examined after use of the IFCEREP0 service aid program. The program can:

- Print the IPL and EOD records, and print a summary of all the IPL and EOD records that are on SYS1.LOGREC; this permits examination of the records to determine the reasons for repeated system initializations.
- Write the records from SYS1.LOGREC to a measurement data set (generally a tape). The information on SYS1.LOGREC includes, in addition to the IPL and EOD records, error recording records from the SDR, MCH, TPER, CCH, and OBR error recording routines.

Major Changes From MVT: None.

Track Stacking

Track stacking is an option of VS2 that is used to handle the data brought in from the SYS1.SYSJOBQE data set. It permits one or more logical tracks for a particular job to reside temporarily in storage as an ordered series.

If a larger part of the computer time at an installation is spent in job scheduling, performance can be improved by using track stacking. Track stacking allows the moving of an

entire locical track from the input queues into storage for use by the initiators; this reduces the number of accesses and thus reduces arm interference. Track stacking may be specified at system generation time or at IPL time; if the track stacking facility is used, region sizes for the initiators are increased to include the additional buffer space.

Major Changes From MVT: None.

Automatic Priority Group (APG)

Automatic priority group (APG) is an option of VS2 that assigns a single priority level to a group of tasks in an attempt to provide optimum use of CPU and I/O resources by these tasks. The priority level may be identified at system generation or system initialization time.

Tasks are dispatched and shifted within the APG based on their characteristics. Tasks are considered CPU-oriented if they appear to be oriented more towards use of the CPU resource. Tasks are considered I/O-oriented if they seem more inclined towards use of I/O (paging I/O is excluded from this determination).

All tasks in the APG are dispatched with a time interval. A task is considered CPU-oriented in APG if it uses its entire interval of allotted time - that is, if it does not give up control voluntarily. A task is considered I/O-oriented in APG if it does not use its entire interval of allotted time.

Important features of the APG group are:

- Tasks within the I/O subgroup are ordered such that those which use smaller portions of their interval are ranked higher in the queue.
- CPU tasks receive control in a cyclic manner, thus ensuring that any available CPU time is distributed equitably among them.
- Potential I/O tasks are not locked at the bottom of the CPU subgroup indefinitely.
- APG is self-adjusting within user-specified limits to maintain a mix of CPU-bound jobs to I/O-bound jobs.

The priority level identified for the APG cannot also be identified as a time slice group.

Major Changes from MVT: This facility is not available in MVT.

Access Methods

Access methods are techniques for moving data between virtual storage and input/output devices. The access methods that are optional in VS2 are the telecommunications access method (TCAM), the virtual storage access method (VSAM), the basic telecommunications access method (BTAM), and the graphics access method (GAM).

Telecommunications Access Method (TCAM)

The telecommunications access method (TCAM) is an option of VS2 that is a generalized I/O control system that extends the techniques of data management to the teleprocessing environment. The data sets addressed by the user (via GET, PUT, READ, and WRITE macro instructions) are queues of messages coming from, or going to, remote terminals via communications lines. If the time sharing option (TSO) is included in the system, TCAM provides the terminal support.

In addition to controlling the transfer of messages to user-written application programs, TCAM provides a flexible, message control language. TCAM macro instructions can be used to construct an installation-oriented message control program for controlling message traffic among remote terminals, and between remote terminals and application programs.

A teleprocessing control system created through the use of the TCAM message control language can:

- Establish contact and control message traffic between computer and terminals.
- Delete and insert line control characters automatically, thus freeing the user from line control considerations.
- Assign, use, and free buffers dynamically.
- Edit incoming and outgoing messages.
- Forward messages to destination terminals and application programs.
- Take corrective action and provide special handling for messages containing errors.
- Maintain statistical information about message traffic.

TCAM offers an extensive set of facilities. They include:

- Online testing of teleprocessing terminals and control units.
- Program debugging aids.
- Network reconfiguration facilities.
- Checkpoint/restart.
- I/O error recording.
- Operator control to examine and alter network status.
- Alternate destination capability.

Major Changes from MVT: TCAM message control programs and message processing programs must be reassembled and linkage edited before execution in VS2. Also, VS2 TCAM supports fetch protection.

Virtual Storage Access Method (VSAM)

The virtual storage access method (VSAM) is a new access method that is used with direct access storage devices. VSAM creates and maintains two types of data sets.

One type of data set is sequenced by a key field within each record and is called a key-sequenced data set. Data records are located by using the key field and an index that records each key field and the address of the associated data, similar to ISAM.

The other type of data set is sequenced by the time of arrival of each record into the data set and is called an entry-sequenced data set. Data records are located by using the record's displacement from the beginning of the data set. The displacement is called the relative byte address, similar to the relative block address used with BDAM.

VSAM stores, retrieves, and updates user data records in these types of device independent data sets. The data records are stored in a new data format designed for device independence. This ensures long term data stability and suitability for use in data base applications. Data in both types of data sets can be accessed either sequentially or directly.

VSAM enhances may ISAM capabilities including performance, device independence, concurrent processing, data portability, and kinds of accessing supported. It provides multiple levels of password security protection. It also creates and maintains separate catalogs that contain information about each VSAM data set and are used to link a data set with its index.

VSAM provides a multifunction utility program that will define, delete, print, copy, and provide backing and portability of VSAM data sets. An ISAM interface routine to allow most existing ISAM programs access to VSAM data sets is also provided.

For a detailed description of VSAM, see *OS/VS Virtual Storage Access Method (VSAM) Planning Guide,* GC26-3799.

Major Changes from MVT: This facility is not available in MVT.

Basic Telecommunications Access Method (BTAM)

The basic telecommunications access method (BTAM) is an option of VS2 that provides the basic modules for constructing a teleprocessing program, including routines for controlling a variety of terminal units, communications lines, and transmission control units. BTAM can be used as a component in the development of more sophisticated teleprocessing systems, and can be modified to support special configurations not supported by other access methods.

BTAM provides the capabilities to:

- Poll terminals and receive messages.
- Address terminals and send messages.
- Dynamically chain input buffers.
- Dial and answer.
- Detect and correct errors.
- Write output buffer chains.
- Perform code translation.

BTAM controls terminal I/O operations initiated by READ and WRITE macro instructions in the user's problem program. Although BTAM controls data transmission, it does not provide elaborate message queuing or process messages.

Major Changes from MVT: None.

Graphics Access Method (GAM)

The graphics access method (GAM) is an option of VS2 that consists of I/O and control routines that transfer data to and from graphic devices. GAM, part of the graphic programming services (GPS), provides the following facilities:

- Buffer management routines that allocate, control, and protect buffer storage.
- Data management routines that store graphic orders and data in user-specified output areas.
- READ/WRITE level macro instructions that transfer data between virtual storage and the graphic device buffers.
- Basic and express attention handling routines that facilitate man-machine communication.
- Error handling routines that diagnose synchronous errors and perform the necessary error handling.

When GAM is included at system generation time, problem oriented routines (POR) and the graphic subroutine package (GSP) may also be included.

Major Changes from MVT: None.

Options Included After System Generation

Type I programs are free and can be added to the system after system generation. They are not distributed with the control program, and must be ordered separately.

Program products, available from IBM for a license fee, provide the user with specialized problem programs; they are added to the system after system generation and installation.

Information on the availability of all type I programs and program products, and documentation on their use, is available from the local IBM sales office.

Major Changes from MVT: Any restrictions on use of the type I programs and program products are available from the local IBM sales office.

Compatibility

VS2 is an extension of the MVT configuration; VS1 is an extension of the MFT configuration. This chapter describes the differences and incompatibilities between VS2 and MVT, with references, where applicable, to VS1 and MFT. It is divided into the following sections:

- Operator commands.
- Job control language.
- Problem programs.
- Coding guidelines.
- Emulators.
- Reassembly/recompilation.
- System data sets.
- System macro instructions.
- Major VS2 - MVT differences.
- Major VS2 - VS1 differences.

Operator Commands

Operator commands are statements to the control program, issued via a console device or the input stream, which cause the control program to provide requested information, alter normal operations, initiate new operations, or terminate existing operations.

This section does not describe all of the operator commands needed to run jobs in VS2. Rather, it describes the parameters that have changed from MVT to VS2.

MVT Commands

Except for the major changes noted below, the operator commands for VS2 are the same as the commands for MVT. Figure 19 provides a summary of the changes.

Command	Parameter	Change
CANCEL		Operator can cancel job waiting for region or data set.
DISPLAY	A N Q	Output changed to reflect virtual storage. RJE and ASB reader queues not supported. RJE and ASB reader queues not supported.
DUMP	ALL	Parameter not supported.
MODE		One simplified command for all models.
MONITOR	A	Output changed to reflect virtual storage.
MOUNT	H	Parameter not supported.
SET	GMT	New parameter for time-of-day clock.
START	H LSQA	Parameter not supported. New parameter for local system queue areas.
VARY	F M OFFGFX ONGFX S	Parameter not supported. Parameter not supported. Parameter not supported. Parameter not supported. Parameter not supported.

Figure 19. VS2 Changes to MVT Operator Commands

CANCEL Command

The specification of the CANCEL command is the same for VS2 and MVT. However, the command can now be used to cancel a job waiting for a region or a data set.

DISPLAY Command

The specification of the DISPLAY command is the same for VS2 and MVT. However, the output generated by this command is different in the following instances:

- If the parameter A is specified in the command, the output will include the starting and ending virtual storage addresses of the region, the number of pages allocated for the local system queue area (LSQA) for the job step, and an indication of whether the job step is executing in pageable or nonpageable storage. Also, the output now permits up to 255 active tasks to be displayed.
- If the parameters N or Q are specified in the command, the output will no longer include the remote job entry or automatic SYSIN batching reader queues, since these facilities are no longer supported.

DUMP Command

The specification of the DUMP command is the same for VS2 and MVT. However, the parameter ALL is no longer supported. Since the output generated by the DUMP command in VS2 is virtual storage, if ALL was specified the output would consist of all 16,777,216 bytes of virtual storage.

MODE Command

The specification of the MODE command is different because in VS2 one simplified MODE command can be used for all machine models.

MONITOR Command

The specification of the MONITOR command is the same for VS2 and MVT. However, the output generated by the command is different in the following instance:

- If the parameter A is specified in the command, the output will include the starting and ending virtual storage addresses of the region, the number of pages allocated for the local system queue area (LSQA) for the job step, and an indication of whether the job step is executing in pageable or nonpageable storage.

MOUNT Command

The specification of the MOUNT command is the same for VS2 and MVT. However, the parameter H (hierarchy) should not be specified. If H is specified, the system will issue a message stating that H is an invalid keyword; the operator will then have to reissue the command, omitting the H parameter.

SET Command

The specification of the SET command at IPL time is the same for VS2 and MVT. However, a new parameter, GMT, can be specified. If GMT is specified, the DATE and CLOCK values in the command are Greenwich Mean Time.

START Command

The specification of the START command is the same for VS2 and MVT. However, the parameter H (hierarchy) should not be specified. If H is specified, the system will issue a message stating that H is an invalid keyword; the operator will then have to reissue the command, omitting the H parameter.

The START command also has a new parameter, LSQA, which specifies the number of segments of LSQA to be allocated to the task being started.

VARY Command

The specification of the VARY command is the same for VS2 and MVT. However, the parameters ONGFX and OFFGFX (and F, M, and S) should not be specified. If the parameters are specified, a rejection message will be issued and the operator will have to reissue the command, omitting the parameters. These parameters have been eliminated since the graphic job processor (GJP) and the satellite graphic job processor (SGJP) are no longer supported.

TSO Commands

This section does not describe all the changes to the TSO operator commands. However, it does list the significant changes to the MODIFY and START commands for VS2. It also lists the new abbreviations that can be specified for the operator parameters.

MODIFY Command

The following parameters allow installation control over auxiliary storage, and can be specified on the MODIFY command. The parameters are described in detail in the discussion of TSO in the chapter "Options".

AUXLIST
BACKUP
TSOAUX
TSOMAX

The following parameters have been changed on the MODIFY command:
REGNMAX=nnn, where nnn=0-42 -- can now be specified.

REGSIZE(region-number)=xxxK, where the maximum is 896K -- replaces old REGSIZE parameter.

START Command

The following new parameters allow installation control over auxiliary storage, and can be specified on the START command. The parameters are described in detail in the discussion of TSO in the chapter "Options."

AUXLIST
BACKUP
LPAF
LPAR
NOSWAP
SWAP
TSOAUX
TSOMAX

The following parameters have been deleted from the START command:

FORM
MAP

The following parameters have been changed on the START command:

DUMP=DUMP -- replaced by DUMP
DUMP=NODUMP -- replaced by NODUMP
LIST -- no longer positional.
LPA -- can now be specified.
REGNMAX=nnn, where nnn=0-42 -- can now be specified.

REGSIZE(region-number)=xxxK, where the maximum is 896K -- replaces old REGSIZE parameter.

Parameter Abbreviations

All operator parameters are listed in Figure 20. Following each parameter is its unique abbreviation for VS2. These abbreviations are new for VS2.

Parameter	Abbreviation	Parameter	Abbreviation
ACTIVITY	AC	NOPRIORITY	NOPRI
AUXLIST	AU	NOSWAP	NOSWAP
BACKGROUND	BACKG	NOSWAPLOAD	NOSWAPL
BACKUP	BACKU	NOWAIT	NOW
CYCLES	C	OCCUPANCY	O
DECAYACT	DECAYA	PREEMPT	PRE
DECAYWAIT	DECAYW	PRIORITY	PRI
DRIVER	DRIVER	REGNMAX	REGN
DSPCH	DS	REGSIZE	REGS
DUMP	DU	SERVICE	SE
HOLD	H	SMF	SMF
LIST	LI	SUBMIT	SUBM
LPA	LP	SUBQUEUES	SUBQ
MAXOCCUPANCY	MAXO	SWAP	SWAP
MAXSWAP	MAXS	SWAPLOAD	SWAPL
MINSLICE	MI	TERMAX	TE
NOACTIVITY	NOAC	TSCREGSZ	TSC
NOAVGSERVICE	NOAV	TSOAUX	TSOA
NOBACKGROUND	NOB	TSOMAX	TSOM
NODUMP	NOD	USERS	U
NOOCUPANCY	NOO	WAIT	W
NOPREEMPT	NOPRE		

Figure 20. Parameter Abbreviations for TSO Operator Commands

Job Control Language

Every job submitted for execution by the operating system must include job control language statements. These statements contain information required by the operating system to initiate and control the processing of jobs.

This section does not describe all of the job control language needed to run jobs in VS2. Rather, it describes the parameters and subparameters that have changed from MVT to VS2. However, most jobs that run in Release 21 of MVT can be executed in VS2 without changing the job control language. Also, if the default region size in VS2 is at least as large as the partition size in VS1 and MFT, VS1 and MFT jobs that run in Release 21 of MVT can be executed in VS2 without changing the job control language. Figure 21 provides a summary of the JCL changes.

Statement	Parameter	Change
JOB	ADDRSPC	Assigns pageable or nonpageable storage space, and defaults to pageable storage.
	REGION	Assigns real storage space in 4K multiples and virtual storage space in 64K multiples; reassigns MVT hierarchy 1 storage.
	ROLL	Ignores this parameter.
EXEC	ADDRSPC	Assigns pageable or nonpageable storage space, and defaults to pageable storage.
	REGION	Assigns real storage space in 4K multiples and virtual storage space in 64K multiples; reassigns MVT hierarchy 1 storage.
	ROLL	Ignores this parameter.
DD	DCB	Specifies BTAM online terminal test option in EROPT subparameter; ingores HIARCHY subparameter.
	OUTLIM	Ignores this parameter.
	SEP	Ignores unit separation for nonspecific devices.
	UNIT	Ignores unit separation for nonspecific devices as specified in SEP subparameter.

Figure 21. VS2 Changes to Job Control Language

JOB Statement

Except for the changes noted below, the JOB statement for VS2 is the same as the JOB statement for MVT.

ADDRSPC Parameter

A new parameter, ADDRSPC, is added to the JOB statement in VS2. It may be specified as either ADDRSPC=VIRT or ADDRSPC=REAL:

- If ADDRSPC=VIRT is specified, the job is assigned a region in pageable storage.
- If ADDRSPC=REAL is specified, the job is assigned a region in nonpageable storage.

If the ADDRSPC parameter is not coded, the default value is VIRT. Therefore, if the current job control language for a job is not changed, the job will be executed in pageable storage. If the job must be executed in nonpageable storage, ADDRSPC=REAL must be specified.

If the ADDRSPC parameter is coded on the JOB statement, its value overrides any value in the ADDRSPC parameters that may be specified in the EXEC statements for that job.

REGION Parameter

In MVT, the size requested in the REGION parameter of the JOB statement was rounded up to the next 2K multiple. In VS2:

- If ADDRSPC=REAL is specified, the requested region size is rounded up to the next 4K (page) multiple.
- If ADDRSPC=VIRT is specified or if the ADDRSPC parameter is not coded, the requested region size is rounded up to the next 64K (segment) multiple.

In VS2, hierarchy 1 storage does not exist. Therefore, if the current job control language for a job is not changed, the hierarchy 1 storage specified is added to the amount of hierarchy 0 storage specified. Hence, the job will get the amount of storage requested, but it will all be in contiguous virtual storage.

Since VS2 subpools are allocated in 4K blocks (vs 2K blocks in MVT), it may be necessary to increase the region size to accomodate this difference.

ROLL Parameter

In VS2, rollout/rollin is not supported since paging gives real storage to tasks as they need it. Therefore, if the current job control language for a job is not changed, the ROLL parameter of the JOB statement will be ignored. That is, the ROLL parameter will be treated as if ROLL=(NO,NO) had been specified.

If the current job control language for a job specifies ROLL=(,YES), it may be necessary to increase the value in the REGION parameter to account for the larger addressing space needed.

EXEC Statement

Except for the changes noted below, the EXEC statement for VS2 is the same as the EXEC statement for MVT.

ADDRSPC Parameter

A new parameter, ADDRSPC, is added to the EXEC statement in VS2. Its function is the same as that described under the ADDRSPC parameter of the JOB statement.

If the ADDRSPC parameter is coded on the EXEC statement, its value applies only to the job step specified on that EXEC statement. However, if the ADDRSPC parameter was coded on the JOB statement, its value on the JOB statement overrides any values specified in the EXEC statements for that job.

REGION Parameter

Changes to the REGION parameter of the EXEC statement for VS2 are the same as those described under the REGION parameter of the JOB statement.

ROLL Parameter

Changes to the ROLL parameter of the EXEC statement for VS2 are the same as those described under the ROLL parameter of the JOB statement.

DD Statement

Except for the changes noted below, the DD statement for VS2 is the same as the DD statement for MVT.

DCB Parameter

In VS2, a new value can be specified for the EROPT subparameter of the DCB parameter of the DD statement. If EROPT=T is coded, it indicates a request for the basic telecommunication access method (BTAM) online terminal test option.

VS2 does not support MVT main storage hierarchy support. Therefore, if the current job control language for a job is not changed, the HIARCHY subparameter of the DCB parameter will be ignored.

OUTLIM Parameter

In VS2, the function provided by the OUTLIM parameter of the DD statement is not supported. Therefore, if the current job control language for a job is not changed, the OUTLIM parameter will be ignored.

SEP Parameter

The SEP parameter of the DD statement is used to request channel separation. In VS2, if a nonspecific volume is requested for a data set and channel separation is also requested for that data set, the channel separation will be ignored; space allocation will be automatically controlled by the I/O load balancing facility. Therefore, in this situation, if the current job control language for a job is not changed, the SEP parameter will be ignored.

UNIT Parameter

Changes to the SEP subparameter of the UNIT parameter of the DD statement for VS2 are the same as those described under the SEP parameter of the DD statement.

Problem Programs

VS2 was designed to be an extension of MVT. Thus, all problem programs (MFT, MVT, and VS1) that execute under Release 21 of MVT, except as noted below, also execute, with no modifications, under VS2. The following problem programs do not have to be modified to execute in VS2:

- Problem programs that are not system or environment dependent.
- Problem programs that are coded according to the guidelines described in the OS System Reference Library publications.
- Problem programs that do not modify the system.
- Problem programs that do not access control block fields not available through normal system functions (that is, through EXTRACT or DEVTYPE macro instructions).

Basic and Extended Control Mode PSWs

Each interruption has associated with it a program status word (PSW). System/370 has two PSW formats, differentiated by bit 12, the former USASCII bit:

- When the bit is off (=0), the format is of basic control (BC) mode in which the features of a System/360 computing system and additional System/370 features, such as new machine instructions, are operational on a System/370 computing system. BC mode is used in VS2 at IPL time.
- When the bit is on (=1), the format is of extended control (EC) mode in which all the features of a System/370 computing system, including dynamic address translation, are operational.

Because of the two PSW formats, the following problem programs that execute under the BC mode PSW may have to be modified to execute under the EC mode PSW:

- Problem programs that reference the following PSW fields directly:

  system mask
  bit 12
  interruption code
  instruction length code (ILC)
  condition code (CC)
  program mask
- Programs that use the load PSW (LPSW) macro instruction to enable or disable the system. The LPSW macro instructions should be replaced with MODESET macro instructions; otherwise, system integrity may be impaired or addressability may be inaccurate.
- Programs that use absolute addresses to refer to locations in low storage above 127 decimal, since the system reserved area in low storage has been expanded.
- Programs that use the set storage key (SSK) or insert storage key (ISK) macro instructions, since the register containing the storage key now contains additional information.

Execute Channel Program (EXCP) Macro Instruction

The execute channel program (EXCP) macro instruction provides a device-dependent means of performing I/O operations. The user of EXCP must provide control information regarding the channel program to be executed; he has the option of specifying the use of I/O appendages during the progress of I/O operations associated with his data set.

Because of virtual storage, the following problem programs that contain EXCP macro instructions may have to be modified to execute in pageable storage:

- Problem programs that modify channel programs while they are executing. Since the I/O supervisor builds a translated copy of the channel program, any changes made to the original channel program would not be known to the I/O supervisor. If the problem programs are not modified, they must be executed in nonpageable storage.
- Problem program appendages. In order to avoid a disabled page fault, the appendages must ensure that referenced pages are fixed in storage before they are entered; this can be accomplished by using the new page fix appendage. If the appendages are not modified, they should be executed in nonpageable storage.

Coding Guidelines

Many coding techniques that are used in MVT can also be used in VS2. Because of virtual storage, however, VS2 necessitates new techniques that should be considered. Although the new techniques are not necessary in order for the programs to run, they may improve system performance.

Although paging occurs frequently in the system, a key objective in new coding techniques is to reduce paging whenever possible. The following guidelines may help achieve this goal:

- 4K-byte blocks (pages) should be coded. Frequently-used programs or loops within programs should be kept on one page whenever possible.
- Seldom used routines (such as initialization, error, and termination routines) should be kept separate from those which are frequently used.
- Frequently-used subroutines should be coded inline each time they are needed. In this case, any increase in total program size will be offset by the decrease in paging.
- Data areas should be arranged on a single page. Multiple-page lists, chains, and tables should be avoided.
- Input/output data areas and event control blocks (ECBs) should be kept on a single page. This prevents referring to several pages when searching for an ECB.
- Pages for buffers should be released as soon as they are all used.
- Dynamically changeable code should be kept on one page if possible. If a page has been changed, it must be paged out before the real storage it occupies can be reused; otherwise, the contents of the page can just be overlaid.

Emulators

Integrated emulator programs allow object programs written for another system to be executed in VS2 with little or no reprogramming. Utilizing the compatibility feature (consisting of hardware and microprogrammed routines that aid emulation), the emulator programs operate as problem program in VS2.

The emulators allows multiprogramming of emulator jobs with other jobs. However, since the emulators employ both programming and circuitry to imitate the processing of other systems, their availability is model-dependent. The emulators supported by the different models are listed in Figure 22.

Information on restrictions on use of the emulators and dates on availability are available from the local IBM branch office.

Emulator	Model 145	Models 155II and 158	Models 165II and 168
1401/1440/1460	x	x	
1410/7010	x	x	
DOS	x	x	
7070/7074		x	x
7080			x
709/7090/7094/7094-II			x

Figure 22. Emulators Supported in VS2

Reassembly/Recompilation

Essentially all programs executing under MVT can execute under VS2 without reassembly or recompilation. The primary exception is that users of the telecommunications access method (TCAM) must reassemble and linkage edit their message control programs and message processing programs before execution.

Also, PL/I F programs executing under the queued telecommunications access method (QTAM) prior to Release 20 of MVT must be reassembled and linkage edited before execution in VS2.

System Data Sets

Some system data sets are basic to VS2 and are required for every operating system. Other system data sets are optional and are required only when selected program options are included in the system.

Most system data sets in VS2 are the same as their counterparts in MVT. However, the SYS1.LPALIB data set, new in VS2, now contains all of the SVCs and most of the modules that were formerly contained in the SYS1.LINKLIB and SYS1.SVCLIB data sets; all the modules contained in this new data set appear in the fixed or pageable link pack areas in VS2.

VS2 also provides two other new data sets: SYS1.PAGE and SYS1.DSSVM. The SYS1.PAGE data set is used to satisfy demand paging requirements; the SYS1.DSSVM data set is used to contain the DSS command language modules.

Required Libraries and Data Sets

The following libraries and data sets are required for every operating system, and must be on the system residence volume.

- SYSCTLG (system catalog) -- The system catalog contains pointers to all the cataloged data sets in an operating system.
- SYS1.LOGREC -- This data set is used by recovery management to record statistical data about machine errors.
- SYS1.NUCLEUS (nucleus library) -- The nucleus library contains the resident portion (nucleus) of the control program and modules selected and link edited during system generation.
- SYS1.SVCLIB (SVC library) -- The SVC library contains recovery management support (RMS) modules required for system error recovery.

- PASSWORD -- This data set contains records that associate the names of protected data sets with the passwords assigned to the data sets.

The following libraries and data sets are required for every operating system, and must be on direct access volumes. They may be on the system residence volume.

- SYS1.DSSVM -- This data set contains the command language modules used by the dynamic support system (DSS).
- SYS1.IMAGELIB -- This library contains the 1403 and 3211 universal character set (UCS) and the forms control buffer (FCB) image modules.
- SYS1.LINKLIB (link library) -- The link library contains programs and routines that are not in the link pack area and are referred to by ATTACH, LINK, LOAD, or XCTL macro instructions.
- SYS1.LPALIB -- This library contains all modules appearing in the link pack area (LPA). Since all modules residing in this library are brought into the LPA at IPL time, this library may be placed on a demountable volume that may be removed after IPL.
- SYS1.MAN (SMF data set) -- The SMF data set contains the data collected by the system management facilities (SMF) routines. A primary data set (SYS1.MANX) and an alternate data set (SYS1.MANY) are required.
- SYS1.PAGE -- This data set is used to satisfy demand paging requirements. It is also used for block paging of TSO users.
- SYS1.PARMLIB -- This library contains the PRESRES list used by the master scheduler, the SMFDEFLT list used by the SMF routines, the BLDL list, and various parameters used by NIP.
- SYS1.PROCLIB (procedure library) -- The procedure library contains cataloged procedures used to perform specific system functions.
- SYS1.SAMPLIB -- This library contains the independent utility programs, the IPL text, the SMF exit routines, and the installation verification procedures needed to verify the operation of the generated system control program.
- SYS1.SYSJOBQE -- This data set is used as a work area by the job scheduler.

The following data sets are required if the time sharing option (TSO) is included in the system, and must be on direct access volumes.

- SYS1.BRODCAST -- This data set contains two types of TSO messages: notices and mail.
- SYS1.CMDLIB -- This library contains TSO command processors, service routines, and utilities.
- SYS1.HELP -- This library is required if the TSO HELP command is used. It contains information regarding the syntax, operands, and functions for each TSO command.
- SYS1.UADS -- This library contains a list of terminal users who are authorized to use TSO. It also contains information about each of them.

Optional Libraries and Data Sets

The following libraries and data sets are optional and, if selected, must be on direct access volumes. They may be on the system residence volume.

- SYS1.DUMP -- This data set is used to contain a dump recorded in the event of abnormal termination of a critical task. This data set may reside on a tape device.
- SYS1.MACLIB (macro library) -- The macro library contains macro definitions for the system macro instructions.
- SYS1.SYSVLOGX and SYS1.SYSVLOGY (system log data sets) -- The system log data sets are used to contain write-to-log (WTL) messages before they are printed on a system output device.
- SYS1.TELCMLIB (telecommunications library) -- The telecommunications library contains telecommunication subroutines in load module form.

The following data sets are optional if the telecommunications access method (TCAM) is included in the system, and, if selected, must be on direct access devices. They may be on the system residence volume.

- TCAM checkpoint data set -- This data set contains all information needed to reconstruct the message control program environment upon restart.
- TCAM message queues data set -- This data set contains the messages between the central computer and the remote stations.

Shared Data Sets

If the shared DASD option is included in the system, systems can share common data. Although any of the installation's application data sets can be shared, not all system data can be shared. Following is a list of those system data sets that cannot be shared.

- PASSWORD
- SYS1.LOGREC
- SYS1.LPALIB
- SYS1.MANX
- SYS1.MANY
- SYS1.NUCLEUS
- SYS1.PAGE
- SYS1.SVCLIB
- SYS1.SYSJOBQE
- SYS1.SYSVLOGX
- SYS1.SYSVLOGY
- SYSCTLG

System Macro Instructions

Several new macro instructions have been added to VS2 to support the new functions available. Other macro instructions have been changed from MVT. Although this section does not provide a detailed description of the new and changed parameters in the macro instructions, it does provide a brief overview of the functions and the uses of the macro instructions.

New VS2 Macro Instructions

The following macro instructions are new and are restricted to authorized programs, programs executing in the supervisor state, and programs operating under protection key zero:

PGFIX -- This macro instruction prevents virtual storage from being paged. It can be used to support I/O operations and to avoid integrity problems that may result from disabled page faults.

PGFREE -- This macro instruction frees virtual storage to be paged. It is used to cancel the effect of the PGFIX macro instruction.

PGLOAD -- This macro instruction moves virtual storage pages into real storage page frames. It can be used to ensure that a page will be in real storage when needed.

MODESET -- This macro instruction is used to change the status of programs between supervisor state and problem program state, key zero and non-key zero, and enabled and disabled.

EXCPVR -- This macro instruction indicates that an I/O request is not to be translated by the I/O supervisor. It is used to reduce the time required to initiate an I/O operation.

The following macro instruction are new and are not restricted in use to specific programs:

DEBCHK -- This macro instruction supports the data extent block (DEB) validity checking facility. It can be used to verify that a DEB is valid.

PGRLSE -- This macro instruction deallocates page frames and the slots allocated to the virtual storage pages. It is used to free up real storage and the external page storage associated with it.

TESTAUTH -- This macro instruction supports the authorized program facility (APF). It is used to determine if a program is authorized to use a program or function restricted in use by APF.

Changed MVT Macro Instructions

The following macro instructions exist in MVT, but have been changed in VS2.

ENQ/DEQ -- These macro instructions now support an ECB option.

SPIE -- This macro instruction now supports program interruptions that result from page translation exceptions.

Major VS2 - MVT Differences

This section describes the major functional differences and incompatibilities that exist between VS2 and MVT. However, it does not include those areas (for example, job control language or operator commands) previously mentioned in this chapter.

Unsupported MVT Functions

Some of the major functions in MVT are not supported in VS2. Some of the functions have been replaced by improved functions in VS2; others have been eliminated and have no comparable replacements.

The following MVT functions are not supported in VS2, but are provided for by comparable VS2 functions.

- Conversational remote job entry (CRJE) -- This function is provided for by the time sharing option (TSO).
- IEBUPDAT utility program -- This function is provided for by the IEBUPDTE utility program.
- IEHIOSUP utility program -- This function is no longer needed because of the deletion of most modules from the SVC library.
- Queued telecommunications access method (QTAM) -- This function is provided for by the telecommunications access method (TCAM).
- Rollout/rollin -- This function is no longer needed because demand paging gives real storage to tasks as they need it.
- Scatter load -- This function is no longer needed because of the VS2 paging function.
- System environment recording routines (SER0 and SER1) -- This function is provided for by recovery management routines.
- Transient areas -- This function is no longer needed because all SVC routines reside in either the fixed or pageable link pack area.

The following major MVT functions are not supported in VS2, and have no comparable VS2 functions.

- Automatic SYSIN batching (ASB) reader.
- Direct system output (DSO) writer.
- Graphic job processor.
- Main storage hierarchy support.
- Multiprocessing.
- Remote job entry.
- Satellite graphic job processor (SGJP).
- TESTRAN.

VS2 - MVT Differences

Following are other differences that exist between VS2 and MVT.

- In VS2, lists of system parameters may be preformatted for use during system initialization. For more detailed information, see the following chapter "Defining the System".
- In VS2, part of MVT NIP has been replaced by a new system generation option, DEVSTAT.
- IN VS2, the authorized program facility (APF), DEB validity checking, and LSQA provide increased system integrity.

- In VS2, chained scheduling is supported only in nonpageable storage. If chained scheduling is requested in pageable storage, the request is ignored and normal scheduling is substituted.
- The VS2 SMF data set is not supported on tape. Also, SMF records in VS2 contain additional information.
- In VS2, priority queuing must be specified for all devices that contain the page data set.
- VS2 subpools will be allocated in 4K (page) blocks vs 2K blocks in MVT. VS2 also uses a "best fit" algorithm (i.e., storage closest in size to the request) vs a first fit algorithm (i.e., storage that is large enough to accomodate the request) in MVT.
- VS2 has no maximum channel program length.
- In VS2, storage protection implies both store and fetch protection.
- In VS2, all modules residing in the SYS1.LPALIB data set are brought into the link pack area at IPL time. If a BLDL macro instruction is issued for any of the modules, a return code indicating 'not found' is returned since the SYS1.LPALIB data set is no longer considered part of the system after IPL.
- Any modification (via AMASPZAP or the linkage editor) to modules that were made resident at IPL will require a subsequent re-IPL to cause the modification to become part of the system. This would include any module selected at system initialization via the FIX and MLPA options as well as all modules from SYS1.LPALIB.

  In addition, if a module contained in SYS1.LPALIB is modified as above, at re-IPL either the pageable LPA must be rebuilt or the modified LPA must be used to cause the modification to become part of the system.
- In VS2, an OPEN macro instruction must be issued for every data extent block (DEB) created in order to support DEB validity checking.
- VS2 does not support the 6-hour and 24-hour pseudo-clocks, and interruptions from location 80; these functions are replaced by the time-of-day clock and the clock comparator.
- In VS2, the LPA directory cannot be modified after system initialization.
- VS2 allows the operator to cancel a job waiting for storage or a data set.
- In VS2, SVC DUMP has been expanded to alow dumping of selected area of virtual storage and selective setting of the system to non-dispatchable.
- VS2 assembler replaces both assembler E and assembler F.
- In VS2, dynamic device reconfiguration for the system residence device and the page data set is not supported.

Major VS2 - VS1 Differences

This section does not attempt to give a functional description of VS1. Rather, it serves to list the functional differences and incompatibilities that exist between VS1 and VS2. For a complete description of VS1, see *IBM System/370 VS1 Planning and Use Guide,* GC24-5090.

- VS1 Release 1 is an extension of MFT Release 20.1; VS2 Release 1 is an extension of MVT Release 21.
- VS1 allows the user to specify up to 16,777,216 bytes of address space; VS2 always provides 16,777,216 bytes of address space.
- VS1 divides storage into pages of 2K bytes each; VS2 divides storage into pages of 4K bytes each.
- VS1 allocates external page storage on as many as 8 devices; VS2 allocates external page storage on as many as 16 devices.

- VS1 supports job processing through the job entry subsystem (JES) facility; VS2 supports job processing via MVT job management.
- VS1 reader procedure specifies PGM=IEFVMA; VS2 reader procedure specifies PGM=IEFIRC.
- VS1 supports the direct system output (DSO) writer; VS2 does not.
- VS1 uses transient areas; VS2 uses the link pack area.
- VS1 supports the function provided by the OUTLIM parameter of the DD statement; VS2 does not.
- VS1 service aid HMDPRDMP formats and prints VS1 dumps only; VS2 service aid AMDPRDMP formats and prints VS2 dumps only.
- VS1 service aid HMDSADMP reads records of 2K bytes each; VS2 service aid AMDSADMP reads records of 4K bytes each.
- VS1 does not support the time sharing option (TSO); VS2 does.
- VS1 does not support dynamic dispatching; VS2 does.
- VS1 does not support I/O load balancing; VS2 does.

Unsupported Devices

Figure 23 describes the VS1 support of those System/370 MVT hardware devices not supported in VS2.

For a list of all the devices that are supported in VS2, see Input/Output Devices in the chapter "Introduction".

Device	Supported in VS1
IBM 1017 Paper Tape Reader	
IBM 1018 Paper Tape Punch	
IBM 1255 Magnetic Character Reader	
IBM 1259 Magnetic Character Reader	
IBM 1270 Optical Reader Sorter	
IBM 1442 Model N1, Card Read Punch	x
IBM 1442 Model N2, Card Punch	x
IBM 2245 Printer	
IBM 2301 Drum Storage	
IBM 2303 Drum Storage	
IBM 2311 Disk Storage Drive	
IBM 2321 Data Cell Drive	
IBM 2402 Magnetic Tape Unit	
IBM 2403 Magnetic Tape Unit and Control	
IBM 2404 Magnetic Tape Unit and Control	
IBM 2415 Magnetic Tape Unit and Control	x
IBM 2596 Card Read Punch	x
IBM 2841 Storage Control Unit	
IBM 3881 Optical Mark Reader	
IBM System/370 Model 135	x

Figure 23. VS1 Support of MVT Devices Not Supported in VS2

Defining the System

During system generation and system initialization, the user is able to define or change the operation of a standard or optional feature of the system control program. By defining the system to meet the needs of the installation, he can increase the performance and overall efficiency of the system. This chapter provides preliminary planning information on defining the system, and is divided into the following categories:

- System generation.
- System initialization.
- System restart.
- System libraries.
- PRESRES volume characteristics list.

System Generation

System generation is the process of selecting VS2 modules from IBM-distributed libraries and tailoring them to create an operating system for an installation. There are five major steps in generating a new operating system:

- Planning -- The distribution libraries containing the VS2 modules must be ordered from IBM. The VS2 options must be selected that, with the standard VS2 features, will constitute the desired system. If an existing VS2 operating system is not available to generate the new system, the VS2 starter system must also be ordered.
- Stage I -- The macro instructions needed to specify the options and standard features of the new system must be selected and coded. These macro instructions are assembled and expanded into job control language and utility control statements.
- Initialization -- The direct access volumes that will contain the distribution libraries (and the starter system) must be initialized. The volumes on which the new system will be generated must also be initialized, and data sets on the system volumes must be allocated.
- Stage II -- The job control language and utility control statements from stage I assemble, link-edit, and specify modules to be moved or copied from the distribution libraries into the new operating system.
- Testing -- After the new system is generated, the IBM Program Systems Representative verifies via the installation verification procedures that the system is operational.

System generation is a process that results in an operating system adapted to both the machine configuration and the data processing needs of an installation. After installation needs have been determined, the "tailored" operating system to meet those needs is specified through system generation macro instructions.

The first time VS2 is installed, the IBM-supplied VS2 starter system is required. (The starter system consists of a VS2 operating system that is used to generate VS2.) For subsequent installations, either the starter system or an existing VS2 operating system can be used.

The IBM Program Systems Representative will perform the installation verification procedures (IVP) to ensure that the system is operational on the hardware configuration. This procedure will be performed as part of IBM's system control program installation procedure for the initial installation of VS2 and will also be performed for subsequent updates.

Two other types of system generation can also be performed: nucleus only and I/O device only. The nucleus generation is performed when only changes to the nucleus of the control program need to be made. The I/O device generation is performed when only I/O devices and channels are to be added, deleted, or modified. (Note: The processor-only generation available to users of MVT is not available in VS2.)

In VS2, system generation is improved in the following ways:

- Many facilities that were optional in MVT are now standard. This results in fewer macro instructions and parameters to be specified.
- IBM generates (from the user's system generation statements) and installs one system control program per CPU. IBM installation also includes verification procedures to check that the installed system functions properly. (In MVT, these functions were performed by the user.)
- Additional system initialization lists are available to be preformatted and taken from the SYS1.PARMLIB data set for use at IPL time.

Macro Instructions

System generation macro instructions are used to describe the new system. The macro instructions describe the machine configuration, the control program, the data management routines, and the user-written routines. They are also used to specify the program options that are to be included in the system.

The macro instructions that can be specified in VS2 are:

- CENPROCS -- specifies the central processing unit and secondary model support.
- CHANNEL -- specifies the channel characteristics. A CHANNEL macro instruction is required for each channel in the installation's computing system.
- CKPTREST -- specifies standard system completion codes not eligible for automatic restart, and user completion codes that are eligible for automatic restart. If the CKPTREST macro instruction is not specified, the standard set of codes will be used.
- CTRLPROG -- specifies control program options. If the CTRLPROG macro instruction is not specified, the default values are assumed.
- DATAMGT -- specifies optional access methods (BTAM and TCAM) to be included in the system. (GAM is specified via the GRAPHICS macro instruction.)
- DATASET -- specifies system data sets and the volumes on which they reside. A DATASET macro instruction is required for each data set defined that is not located on the system residence device.
- EDITOR -- specifies linkage editor options. If the EDITOR macro instruction is not specified, the default values are assumed.
- GENERATE -- specifies the data sets, volumes, and I/O devices required for the system generation process, the system generation output options, and the type of generation being performed.
- GRAPHICS -- specifies inclusion of graphic programming services (problem oriented routines and graphic subroutine package) and the GAM access method.
- IODEVICE -- specifies characteristics of an I/O device and its operating system requirements. An IODEVICE macro instruction is required for each uniquely addressable I/O device.
- LINKLIB -- specifies user-written routines, in load-module form, to be added to the link library or the fixed link pack area of the new system.
- LOADER -- specifies options to be included in the loader processing program. If the LOADER macro instruction is not specified, the default values are assumed.
- MACLIB -- specifies functional macro instructions to be excluded from the macro library of the new system.
- PAGE -- specifies the page data set(s). The PAGE macro instruction may be used to define up to 16 page data sets by being specified once for each data set required. The specifications in the PAGE macro instruction may be overridden at NIP time.

- RESMODS -- specifies user-written routines, in object or load-module form, to be added to the system nucleus member being generated.
- SCHEDULR -- specifies job and master scheduler options.
- SECONSLE -- specifies secondary consoles for the multiple console support function.
- SVCTABLE -- specifies the number, type, APF authorization, and entry status of the user written SVC routines to be added to the new system.
- TSO -- specifies a system with the time sharing option.
- UCS -- specifies the IBM standard character set images for an IBM 1403 Printer with the universal character set feature or an IBM 3211 Printer. If the UCS macro instruction is not specified, it is expected that the character set images will be included by a process other than system generation.
- UNITNAME -- specifies a group of I/O devices. A UNITNAME macro instruction is required for each named group of I/O devices in the system, except for unique device types.

Options

System control program options are included in the system via the system generation macro instructions. Figure 24 lists the options and the macro instructions by which they are specified. For a description of each individual option, see the chapter "Options".

Option	Macro Instruction
Alternate path retry (APR)	IODEVICE
Automatic priority group (APG)	CTRLPROG
Automatic volume recognition (AVR)	SCHEDULR
Basic telecommunications access method (BTAM)	DATAMGT
Device independent display operator console support (DIDOCS)/status display support (SDS)	SCHEDULR
Graphics access method (GAM)	GRAPHICS
Reliability data extractor (RDE)	CTRLPROG
Shared direct access storage devices (shared DASD)	IODEVICE
Telecommunications access method (TCAM)	DATAMGT
Time sharing option (TSO)	TSO
Time slicing	CTRLPROG
Track stacking	SCHEDULR

Figure 24. VS2 Options and System Generation Macro Instructions

System Initialization

Before VS2 can process jobs, the control program and its associated control blocks and work areas must be loaded into virtual storage and prepared for operation. Loading these control program modules is the function of the initial program loader (IPL).

After the IPL program completes it loading functions, control passes to the nucleus initialization program (NIP), which performs functions necessary to initiate operation of the control program. (For example, NIP defines storage areas and initializes certain tables, work areas, and control blocks.) NIP also loads and initializes routines (such as fixed LPA routines) selected by the user.

The nucleus initialization program (NIP) provides initialization of both real and virtual storage in VS2. To give the installation control over virtual storage, NIP recognizes several new parameters. The installation is able to specify these new parameters, as well as the supported MVT parameters, via the master console or a new SYS1.PARMLIB member. The parameters specified are variable with each execution of the initial program loader (IPL) program.

SYS1.PARMLIB

The SYS1.PARMLIB data set in VS2 includes both IBM-supplied and user-supplied parameter lists. The formats for the lists have been revised from MVT to allow better space utilization within parameter records and to afford more flexibility in their definition. However, all MVT parameter lists that are still valid lists in VS2 need not have their formats changed.

IBM-Supplied Lists

The following IBM-supplied lists are currently included in SYS1.PARMLIB for MVT, but are not included in VS2:

- IEAIGE00 -- resident error recovery procedures list.
- IEAIGG00 -- resident access methods list.
- IEARSV00 -- resident SVC list.

The following lists are currently included in SYS1.PARMLIB for MVT, and are also included in VS2:

- IEABLD00 -- resident BLDL list.
- LNKLST00 -- link library list.

The IEASYS00 list is new for VS2. It is created during stage II of system generation and includes all system parameters defined during system generation. If not modified in response to the " SPECIFY SYSTEM PARAMETERS" message, the IEASYS00 list of parameters is used for the IPL in process.

User-Supplied Lists

The SYS1.PARMLIB lists described below are not supplied by IBM, but are defined by the VS2 user via the IEBUPDTE utility program. They are all new for VS2.

The installation is able to specify system parameters though the IEASYSxx list in SYS1.PARMLIB; in fact, multiple sets of system parameters can be maintained in SYS1.PARMLIB. The set of system parameters which is to be effective for any given IPL process is determined by the operator's response to the "SPECIFY SYSTEM PARAMETERS" message. The operator's reply may include a list of parameters to be used or one of the SYS1.PARMLIB parameter lists. If the SYS1.PARMLIB list being used does not prohibit operator intervention, the operator can override all parameters. By using a SYS1.PARMLIB parameter list, the operator is freed from typing in all the necessary entries.

The IEALPAxx list is used to indicate reenterable routines from the SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB data sets that are to be made resident as a pageable extension to the link pack area. The routine in this area (called the modified link pack area) are resident only for the current IPL. If a routine is searched for, the modified link pack area is searched first, followed by the pageable link pack area. If a system restart is initiated and a previously created link pack area is used, the routines in the modified link pack area must be respecified if desired. The IEALPAxx list is specified by the MLPA system parameter.

The IEAFIXxx list is used to indicate reenterable routines from the SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB data sets that are to be made resident as a nonpageable extension to the link pack area. The routines in this area (called the fixed link pack area) are resident only for the current IPL. If a routine is searched for, the fixed link pack area is searched first, the modified link pack area is searched second, and the pageable link pack area is searched last. If a system restart is initiated and a previously created link pack area is used, the routines in the fixed link pack area must be respecified if desired. The IEAFIXxx list is specified by the FIX system parameter.

VS2 System Parameters

Each parameter in the SYS1.PARMLIB lists is specified in the same format as that required by the "SPECIFY SYSTEM PARAMETERS" message. Figure 25 provides a summary of these parameters and, for convenience, groups them into nine general categories. Following the figure is a more detailed description of the parameters.

Parameter	NIP Categories: Virtual Storage Initialization	Real Storage Initialization	Page Data Set Initialization	Paging Supervisor Initialization	Task Dispatching	System Queue Area Definition	Master Scheduler Region Definition	System Parameter Definition	Other Definitions	Value Specified
APG					x					Automatic priority group for dispatcher.
BLDL	x									Pageable directory for SYS1.LINKLIB.
BLDLF	x	x								Nonpageable directory for SYS1.LINKLIB.
CLPA	x									Link pack area to be recreated.
CPQE				x						Channel programs for paging supervisor.
DUMP									x	Tape volume for SYS1.DUMP.
FIX	x	x								Reenterable routines for nonpageable LPA.
HARDCPY									x	Hard copy log.
LSQACEL						x				Quickcells for local system queue area.
MLPA	x									Reenterable routines for pageable LPA.
MPA							x			Master scheduler region size.
OPI								x		Operator intervention restrictions in SYS1.PARMLIB.
PAGE			x							Location and allocation parameters for page data set.
PAL				x						Paging algorithm limits.
REAL	x	x								Nonpageable storage size.
SQA						x				Pageable system queue area size.
SQACEL						x				Quickcells for system queue area.
SYSP								x		SYS1.PARMLIB parameter list to be merged with IEASY00.
TMSL					x					Time slice groups.
TRACE									x	Entries in system trace table.
TSOAUX			x							Minimum auxiliary storage backup for TSO jobs.

Figure 25. VS2 System Initialization Parameter Summary

APG

specifies an automatic priority group for the dispatcher.

If the APG and TMSL parameters specify the same priority, the APG specification of that priority will be accepted.

BLDL

specifies a directory for the SYS1.LINKLIB data set in pageable storage.

The value specified is appended to IEABLD to form the name of a SYS1.PARMLIB member to contain the list of modules for which entries are to be built in the resident BLDL table.

The directory created in response to the BLDL parameter will reside in pageable storage.

If the BLDL and BLDLF parameters are both specified, the BLDL parameter will be ignored. A warning message will be issued to identify this condition.

If specified during system generation and not modified, the IEABLD00 list is used.

BLDLF
specifies a directory for the SYS1.LINKLIB data set in nonpageable storage.

The value specified is appended to IEABLD to form the name of a SYS1.PARMLIB member to contain the list of modules for which entries are to be built in the resident BLDL table.

The directory created in response to the BLDLF parameter will reside in nonpageable storage.

If the BLDLF and BLDL parameters are both specified, the BLDLF parameter will be accepted. A warning message will be issued to identify this condition.

If specified during system generation and not modified, the IEABLD00 list is used.

CLPA
specifies that the link pack area is to be recreated.

If a previously created link pack area is found in one of the page data sets, it is deleted and the space it occupied is made available for system paging use.

CPQE
specifies the number of channel programs to be used by the paging supervisor.

The value specified is the number of channel programs to be added to the minimum number of channel programs (10 for a single page data set, and 15 for two or more page data sets).

If the CPQE parameter is not specified and the time sharing option is included in the system, the minimum numbers described above will be increased by 80.

DUMP
specifies whether a tape volume is to be used for the SYS1.DUMP data set.

If the SYS1.DUMP data set is cataloged on a direct access device, the DUMP parameter will be ignored.

If the DUMP parameter is not specified and if the SYS1.DUMP data set is not cataloged, a message will be issued requesting that a tape volume be provided and specified for the SYS1.DUMP data set.

FIX
specifies reenterable routines from the SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB data sets or from a user-specified library to be made resident as a nonpageable extension to the link pack area.

The value specified is appended to IEAFIX to form the name of a SYS1.PARMLIB member to contain the list of modules to be loaded.

If specified during system generation and not modified, the IEAFIX00 list is used.

For more information on the IEAFIXxx list, see "User-Supplied Lists" in this chapter.

HARDCPY
specifies that the hard copy log is desired, whether the system log is to be used as hardcopy log, and whether messages are to be recorded on the hardcopy log.

If the console configuration contains an active graphic console or more than one active console, a hardcopy log is required. In this case, if routing codes 1, 2, 3, 4, 7, 8, or 10 are desired, they need not be specified since all of these codes are automatically assigned by the system.

LSQACEL
specifies quickcells (reserved space used to reduce time required to allocate space for a control block) in the local system queue area.

The size of a quickcell will be rounded up by the system to an even multiple of 8 bytes. The total size of the quickcell area is limited to 4K bytes.

MLPA
specifies reenterable routines from the SYS1.LINKLIB, SYS1.LPALIB, and SYS1.SVCLIB data sets to be made resident as a pageable extension to the link pack area.

The value specified is appended to IEALPA to the form the name of a SYS1.PARMLIB member to contain the list of modules to be loaded.

For more information on the IEALPAxx list, see "User-Supplied Lists" in this chapter.

MPA
specifies the master scheduler region in pageable storage.

The value specified is the number of 64K byte segments to be added to the minimum size (128K bytes) of the master scheduler region.

If the specified amount of virtual storage is not available, a warning message will be issued. The MPA parameter may be respecified at that time.

OPI
specifies operator intervention restrictions of the system parameters in the SYS1.PARMLIB list.

The OPI parameter can only be included in the SYS1.PARMLIB list. It may be specified for individual system parameters or for the entire set of system parameters.

PAGE
specifies the location and allocation parameters for the page data set(s).

The value specified may include the unit address or the volume serial number of the device containing the page data set, the number of storage blocks to be pageable to the data set, the number of tracks or cylinders to be assigned to the data set, the largest single area currently available on the device assigned to the data set, whether the page data set is to be reformatted, whether the page data set is to contain the link pack area and its directory, and whether the page definition is to be displayed on the console.

PAL
specifies page fix count limits, page replacement threshholds, and intervals and limits for the task disable algorithm.

REAL
specifies nonpageable storage.

The value specified is the number of 4K byte pages to be added to the minimum address space (64K bytes) reserved for nonpageable storage.

If the specified amount of real storage is not available, a warning message will be issued. The REAL parameter may be respecified at that time.

SQA
specifies the system queue area in pageable storage.

The value specified is the number of 64K byte segments to be added to the minimum size (64K bytes) of the system queue area.

If the specified amount of virtual storage is not available, a warning message will be issued. The SQA parameter may be respecified at that time.

The size of the system queue area cannot be increased during a restart that reuses the previously initialized link pack area. If the SQA parameter is specified on the restart, and the size is less than that specified on the previous system start, a warning message will be issued and the size will be increased to that of the previous system start. If the SQA parameter is specified on the restart, and the size is greater than that specified on the previous system start, a message will be issued requesting that the SQA parameter be canceled or a system start be initiated.

SQACEL
specifies quickcells (reserved space used to reduce time required to allocate space for a control block) in the system queue area.

The size of a quickcell will be rounded up by the system to an even multiple of 8 bytes. The total size of the quickcell area is limited to 4K bytes.

SYSP
specifies the SYS1.PARMLIB list of system parameters which is to be merged with the default list of system parameters, IEASYS00.

The value specified is appended to IEASYS to form the name of a SYS1.PARMLIB member to contain the list of system parameters to be used for the current IPL.

If not modified via this parameter, only the IEASYS00 list is used. The IEASYS00 member contains the values selected during system generation for various system parameters unless the contents of the list have been modified by the system programmer at the installation.

The members specified by the SYSP parameter are treated in an ascending priority sequence -- that is, parameters defined in a member specified near the beginning of the list will be replaced by the same parameters defined in a member specified near the end of the list. In the process of merging parameters with the IEASYS00 list, IEASYS00 assumes the lowest priority.

TMSL
specifies time slice groups.

The minimum acceptable time slice is 20 milliseconds and any specification less than this value will be increased to 20 milliseconds.

If the TMSL and APG parameters specify the same priority, the TMSL specification of that priority will be ignored.

TRACE
specifies entries in the system trace table.

The value specified here in the TRACE parameter overrides the value specified during system generation. If zero is specified, the system trace option is canceled for the current IPL.

If the specified amount of real storage is not available, a warning message will be issued. The TRACE parameter may be respecified at that time.

TSOAUX
specifies minimum auxiliary storage backup for TSO jobs.

The value specified is the pages of auxiliary storage required to back up the pageable area to be reserved for paging TSO-related tasks.

The storage indicated in this parameter will not be available for allocation to non TSO-related tasks for the duration of the IPL; however, the TSOAUX value may be respecified when TSO is started.

If the number of pages remaining for non-TSO use is less than 500, the number of TSO-reserved pages will be decreased to provide the system with 500 pages. If the total number of pages is less than 500, the TSOAUX parameter will be ignored. In both cases, a warning message will be issued to identify the conditions.

Unsupported MVT Parameters.

Some of the system parameters in MVT are not supported in VS2. Some of the parameters have been replaced by new parameters in VS2; others have been eliminated because their functions are not supported.

If an unsupported parameter is specified, a message will be issued identifying the condition; all parameters specified after the nonsupported parameter will have to be respecified for the system.

The following MVT system parameters are not supported in VS2, but are provided for by comparable VS2 functions.

- MIN -- This parameter is no longer needed in VS2 since the minimum space is now determined by the scheduler.
- MOD -- The CPU model identification is determined in VS2 by the store CPU identification instruction.
- MPS -- This parameter is replaced by the VS2 MPA parameter.
- RAM -- Routines from SYS1.SVCLIB and SYS1.LINKLIB can be added to the VS2 link pack area via the SYS1.LPALIB data set or via the MLPA or FIX parameters.
- RERP -- In VS2, error recovery procedure (ERP) routines are always resident in the link pack area.
- RESVC -- In VS2, SVC routines are always resident in the link pack area.
- SQS -- This parameter is replaced by the VS2 SQA parameter.

The following MVT system parameters are not supported in VS2, and have no comparable VS2 functions.

- ALTSYS -- Dynamic device reconfiguration for the system residence device is not supported.
- HRAM -- MVT hierarchy support is not needed in VS2.
- HSVC -- MVT hierarchy support is not needed in VS2.
- QBF -- The buffer space specified by this parameter can now be specified only at system generation time.

System Restart

It is sometimes necessary to shut down the system because of end-of-shift, end-of-day, normal maintenance, or system malfunction. At these times, in MVT, system restart allows the system to resume operations without redefining the job queues.

In VS2, as in MVT, system restart also allows reuse of the previously initialized job queue. However, in VS2, another type of system restart is available that allows reuse of the previously-initialized link pack area. Unless the CLPA parameter is specified requesting that the link pack area be recreated or unless the link pack area is reformatted via the PAGE parameter, this second type of restart is performed.

System Libraries

To increase system throughput, libraries should be balanced on devices, and devices should be balanced on channels. Ideally, each library should be on a different device, and each device should be on a different channel.

If all libraries are placed on the same device, throughput will be decreased because of excessive arm interference. However, when more than one library is placed on an IBM 2314 Direct Access Storage Facility, IBM 2319 Disk Storage, or IBM 3330 Disk Storage device, arm movement can be reduced by placing the volume table of contents (VTOC) approximately midway between the first and last cylinders being used. The libraries, starting with the most frequently referenced, can then be alternately placed on both sides of the VTOC with the least referenced libraries farthest from the VTOC.

For improved system efficiency in VS2, it is recommended that the following libraries be allocated on cylinder boundaries:

- SYS1.LINKLIB
- SYS1.MACLIB
- SYS1.PROCLIB
- SYS1.SYSJOBQE

The PRESRES Volume Characteristics List

This topic describes the creation and use of a direct access volume characteristics list that is placed in the system parameter library under the member name PRESRES (permanently resident and reserved).

Characteristics of the PRESRES List

The PRESRES volume characteristics list defines the mount characteristics (permanently resident, reserved) and allocation characteristics (storage, public, private) for any, or all, direct access volumes used by an installation. Use of the PRESRES list can only be suppressed by deleting the member from the parameter library (SYS1.PARMLIB).

The scheduler, after receiving control from the nucleus initialization program (NIP), compares the volume serial numbers in the PRESRES characteristics list with those of currently mounted direct access volumes. Each equal comparison results in the assignment to the mounted volume of the characteristics noted in the PRESRES entry. (Fields in the unit control block for the device on which the volume is mounted are set to reflect the desired characteristics.) If the volume is the IPL volume or the volume containing the data sets SYS1.LINKLIB, SYS1.PROCLIB, SYS1.SYSJOBQE, SYS1.PAGE, or a volume that cannot physically be demounted, the mount characteristic (permanently resident) has already been assigned and only the allocation characteristic is set.

A mounting list is issued for the volumes in the PRESRES characteristics list that are not currently mounted (except those for which mounting messages have been suppressed), and the operator is given the option of mounting none, some, or all of the volumes listed. The mount and allocation characteristics for the volumes mounted by the operator are set according to the PRESRES list entry for the volume; the operator mounts the unit on the volume he selects. The mounting list is subject to the following restrictions:

- A PRESRES entry identifying a volume that cannot physically be demounted will appear in the mounting list issued to the operator if the volume (device) is OFFLINE or is not present in the system.
- Only the first 102 volumes on the PRESRES list can be placed on the mount list.

After the scheduler has finished PRESRES processing, reading of the job input stream begins, and the PRESRES list is not referred to again until the next IPL.

To assign allocation characteristics other than "public" to volumes whose mount characteristic is "permanently resident", users can use a PRESRES characteristic list entry.

Selection of the volumes for which PRESRES entries are to be created should be done so that critical volumes are protected. Since the combination of mount and allocation characteristics assigned to a specific volume determines the types of data sets that can be placed on the volume and the volume's usage, the user can exercise effective control over the volume through a PRESRES list entry.

Writing the PRESRES Entry

Each PRESRES entry is an 80-byte record, consisting of a 6-byte volume serial number field, a 1-byte mount characteristic field, a 1-byte allocation characteristics field, a 4-byte device type field, a 1-byte mount-priority field, and an optional information field. Commas are used to delimit the fields, except that the optional information field is always preceded by a blank. All character representation is EBCDIC. The format of the entry is described below.

- Volume serial number -- up to six characters, left justified.
- Mount characteristic -- 0 to denote permanently resident, 1 to denote reserved. The default characteristic is "permanently resident" and is assigned if any character other than 0 or 1 is present in the field.
- Allocation characteristic -- 0 to denote storage, 1 to denote public, 2 to denote private. The default characteristics is "public" and is assigned if any character other than 0, 1, or 2 is present in the field.
- Device type -- a four-digit number designating the type of direct access device on which the volume resides, e.g. the IBM 2319 Disk Storage is indicated by the notation 2319. Note that this field only indicates the basic device type for the associated volume. The operator should be advised if the device requires special features to process the data on the designated volume.
- Mount priority -- N to denote that mount messages at IPL time for a volume should be suppressed. This field allows the user to list seldom used volumes in the PRESRES list without having a mount message issued at each IPL. When these volumes are required, they may be mounted and attributes will be set from the PRESRES list entry. If the mount message is not to be suppressed, the mount priority field and the preceding comma may be omitted.
- Optional information -- descriptive information about the volume. This information is not used by the system, but will be available to the user on a printout of the list. If necessary, comments may start in the second byte after the mount priority field or, if the mount priority field is omitted, in the second byte following the comma after the device type field.

Embedded blanks are not permitted in the volume serial, mount, allocation, or device type fields.

Adding the List

The IEBUPDTE utility program places the list (under the member name PRESRES) in the system parameter library, SYS1.PARMLIB. This utility is also used to maintain the list.

Glossary

This glossary defines new OS/VS2 terms that are used in this publication. For definitions of terms not included in this glossary, see *IBM Data Processing Glossary,* GC20-1699.

address translation: The process of changing the address of a data item or an instruction from its virtual address to its real storage address. See also dynamic address translation.

automatic priority group: A group of tasks at a single priority level that are dispatched according to a special algorithm that attempts to provide optimum use of CPU and I/O resources by these tasks. See also dynamic dispatching.

basic control (BC) mode: A mode in which the facilities of a System/360 computing system and additional System/370 features, such as new machine instructions, are operational on a System/370 computing system. See also extended control (EC) mode.

BC mode: Basic control mode.

DAT: Dynamic address translation.

demand paging: Transfer of a page from external page storage to real storage at the time it is needed for execution.

disabled page fault: A page fault that occurs when I/O and external interruptions are disallowed by the CPU.

DSS: Dynamic support system.

dynamic address translation (DAT): (1) The change of a virtual storage address to a real storage address during execution of an instruction. See also address translation. (2) A hardware feature that performs the translation.

dynamic area: The portion of virtual storage that is divided into regions that are assigned to job steps and system tasks. See also pageable dynamic area, nonpageable dynamic area. Contrast with nondynamic area.

dynamic dispatching: A facility that assigns priorities to tasks within an automatic priority group to provide optimum use of CPU and I/O resources.

dynamic support system (DSS): An interactive debugging facility that allows authorized maintenance personnel to monitor and analyze events and alter data.

EC mode: Extended control mode.

enabled page fault: A page fault that occurs when I/O and external interruptions are allowed by the CPU.

extended control (EC) mode: A mode in which all the features of a System/370 computing system, including dynamic address translation, are operational. See also basic control(BC) mode.

external page storage: The portion of auxiliary storage that is used to contain pages.

fixed: Not capable of being paged out.

fixed BLDL table: A BLDL table that the user has specified to be fixed in the lower portion of real storage.

fixed link pack area: An extension of the link pack area that occupies fixed pages in the lower portion of real storage.

fixed page: A page in real storage that is not to be paged out.

frame: Same as page frame.

link pack area (LPA): An area of virtual storage containing reenterable routines that are loaded at IPL time and can be used concurrently by all tasks in the system.

local system queue area (LSQA): One or more segments associated with each virtual storage region that contain job-related system control blocks.

LPA: Link pack area.

LSQA: Local system queue area.

nondynamic area: The area of virtual storage occupied by the resident portion of the control program (the nucleus and the link pack area). Contrast with dynamic area.

nonpageable dynamic area: An area of virtual storage whose virtual addresses are identical to real addresses; it is used for programs or parts of programs that are not to be paged during execution.

page: (1) A fixed-length block of instructions, data, or both, that can be transferred between real and external page storage. (2) To transfer instructions, data, or both between real storage and external page storage.

page data set: A data set in external page storage, in which pages are stored.

page fault: A program interruption that occurs when a page that is marked not in real storage is referred to by an active page. Synonymous with page translation exception.

page fixing: Marking a page nonpageable so that it remains in real storage.

page frame: A block of real storage that can contain a page. Synonymous with frame.

page-in: The process of transferring a page from external page storage to real storage.

page-out: The process of transferring a page from real storage to external page storage.

page table (PGT): A table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

page translation exception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the page table entry for that address is set. Synonymous with page fault. See also segment translation exception, translation specification exeception.

pageable dynamic area: An area of virtual storage whose addresses are not identical to real addresses; it is used for programs that can be paged during execution.

paging: The process of transferring pages between real storage and external page storage.

paging device: A direct access storage device on which pages (and possibly other data) are stored.

paging supervisor: A part of the supervisor that allocates and releases real storage space (page frames) for pages, and initiates page-in and page-out operations.

PER: Program event recording.

primary paging device: An auxiliary storage device that is used in preference to secondary paging devices for paging operations. Portions of a primary paging device can be used for purposes other than paging operations.

program event recording (PER): A hardware feature used to assist in debugging programs by detecting program events.

real address: The address of a location in real storage.

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.

secondary paging device: An auxiliary storage device that is not used for paging operations until the available space on primary paging devices falls below a specified minimum. Portions of a secondary paging device can be used for purposes other than paging operations.

segment: A continuous 64K area of virtual storage, which is allocated to a job or system task.

segment table (SGT): A table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

segment translation exeception: A program interruption that occurs when a virtual address cannot be translated by the hardware because the invalid bit in the segment table entry for that address is set. See also page translation exception, translation specification exception.

SQA: System queue area.

swapping: In VS2 with TSO, a paging technique that writes the active pages of a job to external page storage and reads pages of another job from external page storage into real storage.

system queue area (SQA): An area of virtual storage reserved for system-related control blocks.

translation specification exception: A program interruption that occurs when a page table entry, segment table entry, or the control register pointing to the segment table contains information in an invalid format. See also page translation exception, segment translation exception.

virtual address: An address that refers to virtual storage and must, therefore, be translated into a real storage address when it is used.

virtual equals real (V=R) storage: Same as nonpageable dynamic area.

virtual storage: Addressable space that appears to the user as real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, rather than by the actual number of real storage locations.

V=R storage: Same as nonpageable dynamic area.

Indexes to OS/VS publications are consolidated in the *OS/VS Master Index*, GC28-0602, and the *OS/VS Master Index of Logic*, GY28-0603. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

When more than one page reference is given, the first page number indicates the major reference.

OS/VS2 Planning Guide

GC28-0600-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. Elsewhere, an IBM office or representative will be happy to forward your comments.

Cut or Fold Along Line

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 comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

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

Cut or Fold Along Line