GC20-1752-0

Systems

OS/Virtual Storage 1 Features Supplement

This supplement discusses OS/Virtual Storage 1 (OS/VS1) features and organization. Only concepts and functions of OS/VS1 that are new to and significantly different from those of OS MFT are presented in detail. Transition from OS MFT to OS/VS1 is discussed also. Facilities announced prior to the date of this supplement are included in the discussions. Features that are not part of the first release of OS/VS1 are identified.

This supplement is an optional section that is designed to be inserted in its entirety in any one of the following base publications, each of which contains the conceptual and System/370 hardware information required to understand the OS/VS1 discussion presented:

- *A Guide to the IBM System/370 Model 135* (GC20-1738)
- *A Guide to the IBM System/370 Model 145* (GC20-1734)
- *A Guide to the IBM System/370 Model 158* (GC20-1754)
- *A Guide to the IBM System/370 Model 168* (GC20-1755)

Readers who possess more than one of the above base publications need add this module to only one of the documents as the OS/VS1 information presented applies to System/370 Models 135, 145, 158, and 168 unless otherwise indicated in the text.

The contents of this supplement are designed to acquaint the OS MFT knowledgeable reader with the new facilities and the advantages of OS/VS1.

IBM

PREFACE

This supplement is stocked in the IBM Distribution Center, Mechanicsburg as a separate form-numbered item and is not automatically distributed as part of any other publication. Subsequent updates to the supplement must also be ordered separately. Those who are familiar with a System/370 model and OS MFT, and who require information about OS/VS1, should obtain this supplement and insert it as Section 90 of one of the appropriate base publications listed below.

Base publications for the OS/VS1 supplement are:

- A Guide to the IBM System/370 Model 135 (GC20-1738-4 or later editions)
- A Guide to the IBM System/370 Model 145 (GC20-1734-2 or later editions)
- A Guide to the IBM System/370 Model 158 (GC20-1754)
- A Guide to the IBM System/370 Model 168 (GC20-1755)

This supplement is self-contained. It begins with page 1 and includes its own table of contents and index. The title of the supplement is printed at the bottom of each page as a means of identifying the optional supplement to which the page belongs. Knowledge of information contained in other optional supplements that can be added to the base publications listed above is not required in order to understand the OS/VS1 features as they are presented. However, comprehension of virtual storage concepts and dynamic address translation hardware and terminology, as described in any one of the base publications, is assumed.

First Edition (August 1972)

This publication is intended for planning purposes only. It will be updated from time to time; however, the reader should remember that the authoritative sources of system information are the system library publications for OS/VS1. These publications will first reflect any changes.

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

Address comments concerning the contents of this publication to: IBM Corporation, Technical Publications Department, 1133 Westchester Avenue, White Plains, New York 10604.

CONTENTS (Section 90)

FIGURES (Section 90)

TABLES (Section 90)

SECTION 90: OS/VIRTUAL STORAGE 1 FEATURES

90:05 FUNCTIONS AND FEATURES SUPPORTED

OS/VS1 is a growth operating system for OS MFT and DOS installations. OS/VS1 includes features equivalent to and compatible with those of OS MFT and offers major new functions and feature enhancements. The most significant new items of OS/VS1 are:

- Support of one virtual storage of up to 16,777,216 bytes using dynamic address translation hardware
- More efficient peripheral I/O operations processing provided by the new job entry subsystem (JES), which replaces OS MFT input readers and output writers and incorporates many of the spooling features of MFT HASP II
- Improved remote job entry provided by the new remote entry services (RES) facility, a logical and functional extension of JES. RES replaces OS RJE and provides functions similar to those available in HASP II RJE.
- Job scheduling enhancements that include elimination of small partition scheduling and reduction in contention for the job queue
- An additional access method called Virtual Storage Access Method (VSAM) that is designed to offer more function and to be more suitable to online and data base environments than ISAM
- Operational enhancements
- Improvements in system integrity and data security protection, such as additional protection of control blocks within a user partition

OS/VS1 supports one partitioned virtual storage of up to 16 million (16,777,216) bytes with segments of 64K and pages of 2K. The organization of virtual storage in VS1 is similar to that of main storage in MFT. However, virtual rather than real storage is divided into partitions. The management of virtual, real, and external page storage, and the paging activity of the system are handled entirely by the VS1 control program, and are transparent to the programmer.

OS MFT is upward compatible with OS/VS1 to the extent that moving to OS/VS1 resembles moving from one release of MFT to another that contains significant new features. (See Section 90:50 for a discussion of MFT to OS/VS1 transition.) Except for JES and RES, OS/VS1 is upward compatible with OS/VS2 in the same way MFT is upward compatible with MVT. DOS Version 3 and 4 users can make the transition to OS/VS1 with the aid of the OS/DOS emulator, which also operates under OS/VS1 control. Compatibility that exists between DOS files and MFT data sets also exists between DOS files and OS/VS1 data sets. Because of the compatibility between OS/VS1 and MFT, the effort required to convert from DOS to OS/VS1 is similar to that required to convert from DOS to MFT.

OS/VS1 is an intermediate-level operating system, classified as system control programming (SCP) and hereafter referred to as VS1 or OS/VS1, that supports System/370 Models 135, 145, 158, and 168 operating in EC and translation modes. VS1 also supports purchased Models 155 and 165 with the optional Dynamic Address Translation Facility installed, which are designated as Models 155 II and 165 II, respectively. Models

158, 155 II, 168, and 165 II are not supported by the first release of VS1. VS1 does not support System/370 models operating in EC mode without address translation operative, System/370 models operating in BC mode, or any System/360 models.

The following minimum system configuration and hardware features are used by VS1:

- 144K of real storage for a Model 135, 160K of real storage for a Model 145, and minimum real storage size for other System/370 models supported by VS1 (configurations with less than 160K are subject to certain restrictions)
- Byte multiplexer channel with associated I/O devices, including one reader, one punch, one printer, and one console
- One selector or block multiplexer channel, or IFA for Models 135 and 145, with associated direct access devices that include 2314/2319, 3330-series, or 2305 (Model 1 or 2) direct access storage. The preceding direct access devices are supported for system residence. At least three 2314/2319 disk drives or three 3330-series disk drives are required. In addition, at least one nine-track tape unit is required in order to perform a system generation.
- Dynamic address translation and channel indirect data addressing
- Storage protection
- Interval timer (at location 80) and time of day clock
- Monitoring facility
- Program event recording

The following restrictions apply when VS1 is used on a Model 135 with 144K:

- For most installations, the recommended maximum number of partitions is one.
- The Generalized Trace Facility cannot be used if only one partition is defined.
- If two partitions are defined, GTF can operate but the external trace option cannot be used.
- OLTEP cannot be used (in the first release of VS1).

The standard features of VS1 and a minimal I/O configuration are supported in a system with 160K of real storage. Inclusion of optional features in the generated VS1 control program, support of a larger than minimal I/O configuration, and improved system performance require a system with more than 160K of real storage. Note that a Model 145 with a GE storage size will have less than 160K of real storage available when the control storage requirement exceeds 32K. The advantages and functions offered by a VS1 virtual storage environment and optimal system performance can best be attained for a Model 135 if 240K is installed, and for a Model 145 if 256K or more is installed.

Table 90.05.1 lists the standard and optional features of OS/VS1 and Table 90.05.2 lists the I/O devices, consoles, and terminals supported. Items that are not available in the first release of VS1 are identified. Just as for an OS MFT operating system, the desired installation-tailored OS/VS1 control program must be generated, at which time user-selected optional features are included in the resulting system. More

features are standard in VS1 than in MFT. This can reduce the number of options that must be specified and, thereby, reduce system generation preparation and execution time.

OS/VS1 is a functional extension of OS MFT as of Release 20.1. However, the following MFT features are not available in VS1:

- Storage hierarchies (2361 Core Storage cannot be attached to any System/370 model). Hierarchy parameters are processed at link-edit time but are ignored during program loading.
- TESTRAN
- QTAM (function provided by TCAM), Graphic Job Processor (GJP), and Satellite Graphic Job Processor (SGJP)
- RJE (function provided by RES)
- IEBUPDAT utility (replaced by IEBUPDTE)
- IMAPTFLE and IMDMDMAP replaced by HMBLIST
- IFGUAP utility

The following I/O devices, some of which are suppported by MFT, are not supported by VS1:

1017/1018 Paper tape reader/punch
1285 Optical Reader
2301 Drum
2303 Drum
2311 Disk Storage
2321 Data Cell Drive
7772 Audio Response Unit

Table 90.05.1. Standard and optional features of OS/VS1. Standard features are automatically included during system generation. Optional features must be requested during system generation or added after the generation is performed.

OS/VS1	
Standard Features	Optional Features
• One virtual storage of up to 16 million bytes with 64K segments and 2K pages • Up to 52 partitions 1-15 problem program 1-37 system task • Demand paging for allocation of real storage • Execution of programs in paged and nonpaged (V=R) modes • Independent job scheduling • Job Entry Subsystem (JES) • Multitasking (resident ATTACH) • Storage protection • Timing facilities • Resident access methods • Resident reentrant SYS1.LINKLIB modules • Resident BLDL table • Resident IDENTIFY,EXTRACT,SPIE • Standard Fetch and Multiple Wait • Operator communication at IPL • Validity Checking • Access methods: BSAM, QSAM, BDAM, BPAM • Direct access volume serial number identification • Dynamic Support System (DSS)* • Online Test Executive Program (OLTEP) • Machine Check Handler (MCH) • Channel Check Handler (CCH) • Integrated emulator interface • Linkage Editor/Loader • System Assembler • System Utilities IEHDASDR IEHPROGM IEHMOVE IEHIOSUP IFHSTATR IEHATLAS IEHLIST IEHINITT • Data Set Utilities IEBCOPY IEBTCRIN IEBPTPCH IEBGENER IEBCOMPR IEBEDIT IEBUPDTE IEBISAM IEBDG Access Method Services (for VSAM)* • Independent Utilities IBCDMPRS IBCDASDI ICAPRTBL • Service Aids HMAPTFLE HMASPZAP HMDSADMP HMBLIST HMDPRDMP IMCJOBQD IFCDIP00 IFCEREP0 Generalized Trace Facility (GTF)	• Transient SVC Table • Resident Type 3 and 4 SVC routines • Resident ERP's • PCI Fetch • TRACE function • Alternate or Composite Console • Multiple Console Support (MCS) • Device Independent Display Operators Console Support (DIDOCS) • Remote Entry Services (RES)* • Automatic Volume Recognition (AVR) • Dynamic Device Reconfiguration (DDR) • Alternate Path Retry (APR) • Checkpoint/Restart • Time Slicing • System Management Facilities (SMF) • Access methods: VSAM*, BTAM, TCAM, GAM, BISAM, QISAM • Shared Direct Access Storage Devices (DASD) for 2314/2319, 3330, 2305-2 • Graphics Subroutine Package (GSP) • Conversational Remote Job Entry (CRJE) • Integrated emulators (independent component releases) 1401/1440/1460 emulator for Models 158*, 155 II*, 145, and 135 1410/7010 emulator for Models 158*, 155 II*, and 145 7070/7074 emulator for Models 158*, 155 II*, 168*, and 165 II* 7080 and 709/7090/7094/7094II emulators for Models 168* and 165 II* DOS emulator for Models 158*, 155 II*, 145, and 135 • Reliability Data Extractor • Reduced error recovery for magnetic tape • System Log • Volume statistics for magnetic tape • MSP/7 Host Program Preparation Facility II (HPPF)
*Not available in the first release of VS1	

Table 90.05.2. I/O devices, consoles, and terminals supported by OS/VS1

Readers and Punches

1442 Reader Punch, Models N1 and 2
2501 Card Reader, Models B1 and B2
2520 Card Read Punch, Models B1, B2, B3
2540 Card Read Punch
2596 Card Reader (96 column) as 1442 equivalent

Printers

1403 Printer, Models N1, 2, 3, 7
1443 Printer, Model N1
3211 Printer

Direct Access Storage

2314 Direct Access Storage Facility (Models 1, A, and B) and 2844 Auxiliary Storage Control
2319 Disk Storage, A and B models
3330 Disk Storage
2305 Fixed Head Storage Facility, Model 1* and Model 2

Magnetic and Paper Tape

2400-series magnetic tape, all models, and 2816 Tape Switching
3400-series magnetic tape, all models (2420 equivalent in first release)
2495 Tape Cartridge Reader
2671 Paper Tape Reader

Optical and Magnetic Character Readers

1287, 1288 Optical Character Readers
1419 Magnetic Character Reader (Dual Address Adapter and Expanded Capability feature required)

Display Units

2250 Display Unit
2260, 2265 Display Stations
3270 Display System*

Consoles

3210, 3215 Console Printer-Keyboards
Display Console for the Model 158* (display console not available for Model 155 II) - 3215 mode only in first release
3066 System Console for Models 168* and 165 II*
2150 Console with 1052 Model 7
Composite console (card reader and printer)
2250 and 2260 (local) display units (DIDOCS is required)
2740 Communication Terminal (MCS required)
3277 (DIDOCS is required)*
3213 Printer (for hard copy when the Model 158 display console is used)*

Transmission Control Units

2701, 2702, 2703, 2715 Transmission Control Units
2772 Multipurpose Control Unit
2955 Data Adapter Unit
3271 Control Unit*
3705 Communications Control Unit*
7770 Audio Response Unit

Table 90.05.2. (continued)

Terminals (Start/Stop)
1030 Data Collection System
1050 and 1060 Data Communication Systems
2260 and 2265 Display Stations
2721 Portable Audio Terminal
2740 Models 1 and 2, and 2741 Model 1 Communication Terminals
2760 Optical Image Unit
83B3 AT&T Terminal
WU115A Teletype
TWX-33/35** AT&T Teletype Terminal
System/7 Sensor-Based Information System
Terminals (Binary Synchronous)
2770 and 2790 Data Communication Systems
2780 Data Transmission Terminal
2972-8, -11 General Banking Stations
3270 Information Display System*
3670 Brokerage Communication System*
3735 Programmable Buffered Terminal*
1130 System (as a processor station)
1800 System (as a processor station)
System/3 (as a processor station)
System/360 Models 20 and up (as a processor station)
System/370 models (as a processor station)

*Not supported in the first release of VS1

**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 any changes to the IBM-supplied products or programs may have on such terminals.

90:10 ORGANIZATION AND INITIALIZATION OF STORAGE

VIRTUAL STORAGE ORGANIZATION

The organization of virtual storage in VS1 is reflected in tables and control blocks, similar to those used in MFT, that are established at system initialization and maintained throughout system operation. Virtual storage is organized, allocated, and freed in much the same way as main storage is in MFT. However, in VS1, virtual storage allocated to pageable programs does not require the allocation of real storage until the virtual storage is actually referenced by executing code. The size of the virtual storage supported, up to a maximum of 16,777,216 bytes, can be specified at system generation and can be changed by the operator at IPL.

Virtual storage in VS1, like main storage in MFT, is divided into two main areas: a nonpageable area in lowest addressed virtual storage and a pageable area in highest addressed virtual storage, as shown in Figure 90.10.1. The two areas are divided by the V=R line. Storage protection, functionally equivalent to that implemented in MFT, is provided as a standard feature. Protect key values are assigned to virtual storage areas. Each time a page frame is allocated, its protect key is set equal to the protect key value assigned to the virtual

storage page to which the page frame is allocated. Fetch protection is not supported.

The virtual storage in the nonpageable area is mapped on a virtual equals real (V=R) basis with real storage. That is, each virtual storage page has a page frame assigned such that virtual and real storage addresses are equal. The nonpageable area contains the resident control program, the system queue area, and the virtual equals real (V=R) area. Included in the resident control program are the generated nucleus, which is a minimum of 54K (for one partition and the minimum I/O configuration), and RMS routines (MCH and CCH). The two RMS routines require 6K of resident control program storage and 2K of pageable supervisor area when less than 192K of real storage is present, and 8K of resident control program storage when 192K or more is available. The resident control program is assigned storage protect key 0, and is a multiple of 2K in size.

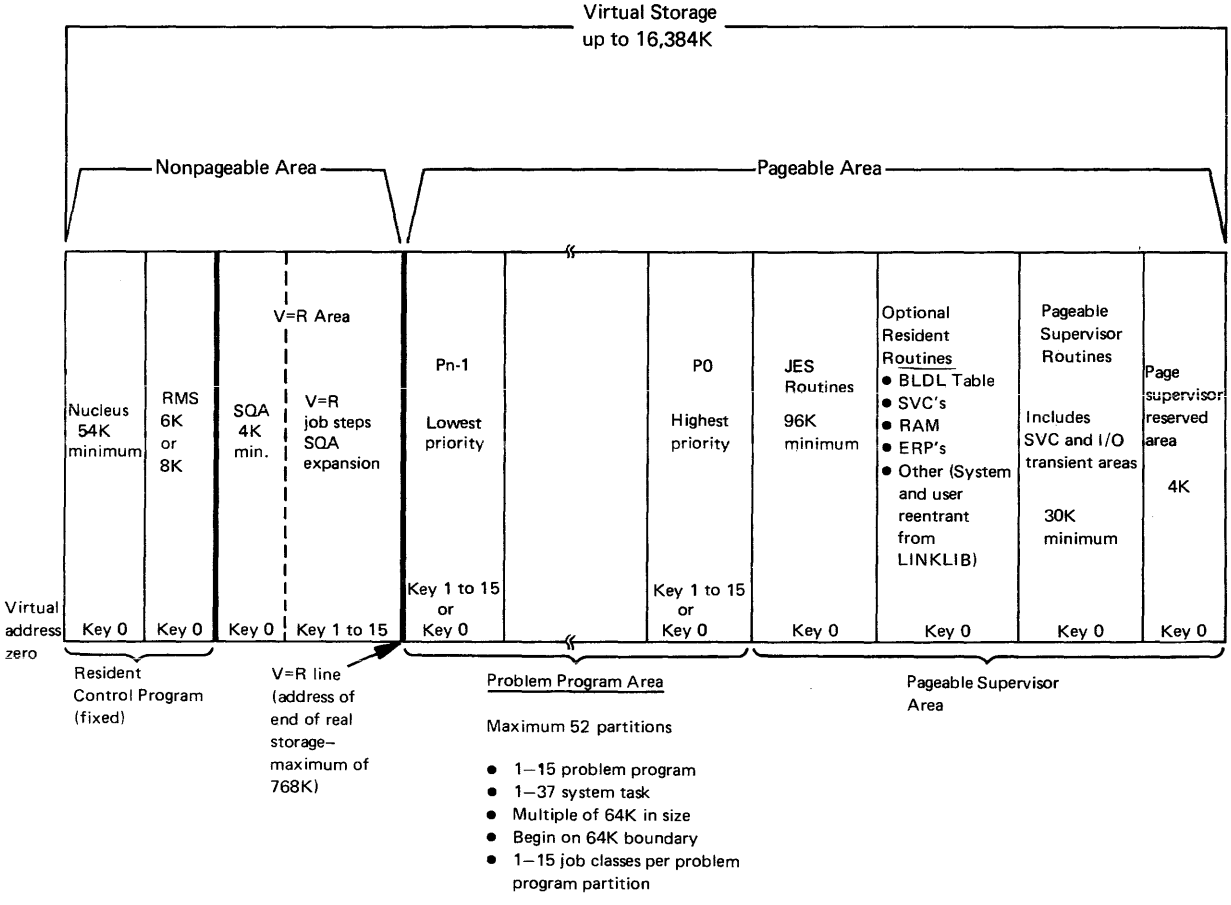


Figure 90.10.1. Virtual storage organization in OS/VS1

The system queue area (SQA) is adjacent to the resident control program. SQA, with a protect key of zero, is used for control blocks and work areas that are not job or job step related or that must have virtual equals real storage addresses. The size of SQA is specified at system generation and must be a minimum of 4K (for a one partition system). An additional 2K of SQA should be allocated for each 64K of real storage present in excess of 128K. The amount of SQA specified is reserved in the nonpageable area of virtual storage during system initialization. However, during system operation, SQA is dynamically

expanded and contracted as needed, a virtual storage page (and page frame) at a time. When an allocated virtual storage page of SQA is no longer required, it and its allocated page frame are freed and become available for reassignment.

The V=R area, located adjacent to SQA defined at system generation, is used for SQA expansion. Any available virtual storage pages (and their correspondingly addressed available page frames) within the V=R area can be allocated to the system queue area, since SQA need not consist of contiguous virtual storage pages. Dynamically expandable system queue space is not supported by MFT, and system operations must be terminated if the amount allocated during IPL is exhausted. The approach implemented in VS1 is designed to minimize system terminations that occur because of a lack of SQA.

The size of the V=R area is variable by system. The V=R area consists of all the virtual storage from the end of defined SQA to the location of the V=R line. The V=R line is established during system initialization when the amount of real storage present has been determined. The address of the V=R line in virtual storage is 768K or the address of the end of real storage, whichever is less.

The V=R area is used for the execution of programs that operate in nonpaged (V=R) mode, as well as for SQA expansion. A nonpageable job or job step is identified by the new ADDRSPC parameter in the JOB or EXEC statement. When ADDRSPC=REAL is specified, the REGION parameter is used to indicate the amount of virtual and real storage required. Storage can be allocated on a job or a job step basis and a job can contain both paged (ADDRSPC=VIRT) and nonpageable job steps. The ADDRSPC and REGION defaults are specified in the PARM field of the EXEC statement in the reader procedure.

A minimum area of 2K can be requested for execution of a nonpageable job step. The maximum amount of real storage that can be allocated to a nonpageable job step is a function of the amount of real storage present in the system, the size of the resident control program, and the amount of real storage allocated to fixed pages at the time the nonpageable job step is initiated. The amount requested must be a multiple of 2K in size. When a nonpageable job step is initiated, enough contiguous virtual and real storage must be available at that time to satisfy the REGION parameter request. More than one nonpageable job step can be active concurrently, up to a maximum of 15, subject to the availability of the contiguous virtual and real storage areas required.

Jobs containing one or more steps that are to execute in nonpaged mode are scheduled (interpreted, initiated, terminated) using a pageable problem program partition in the pageable area that handles the job class indicated. That is, the scheduler operates paged in a pageable partition even though the job steps it schedules operate nonpaged in a contiguous area in the V=R area. Although V=R job steps are not paged, they operate with translation mode operative. This is done because they reference virtual storage addresses contained in the pageable supervisor area. Page tables associated with a nonpageable job step are established such that the real storage address that results from the translation of an address contained within the V=R area allocated to the job step is equal to the virtual storage address. Channel program address translation is not performed on CCW lists contained in a nonpaged program. A nonpaged job step must be restarted from a checkpoint in the same V=R area that was used for the checkpoint.

The only VS1 SCP component that is not part of the resident (fixed) control program and that must operate in nonpaged mode is OLTEP when it is controlling execution of OLT's. In addition, in VS1, a program must operate in nonpaged mode if it:

- Contains a channel program that is modified while the channel program is active
- Is highly time-dependent (involves time-dependent I/O operations, for example, such as magnetic ink character reader programs)
- Must have all of its pages in real storage when it is executing (for performance reasons, for example)
- Must use the chained scheduling facility of BSAM or QSAM
- Uses the EXCP macro and executes user-written I/O appendages that can encounter a disabled page fault (Section 90:25 discusses disabled page faults)
- Uses the EXCP macro and has CCW chains with more the 240 CCW's

Existing user-written programs that are operating in MFT and that must operate in nonpaged mode in VS1 need not be modified in order to run in this mode. Existing optical character reader and certain types of card reader programs can be run in V=R mode under VS1 to maintain the performance currently achieved in MFT. A program that involves operations on a 1287 or a 1288 optical character reader will run slower in paged mode than in nonpaged mode. A program that accesses a card reader directly (does not use JES) and that uses the CNTRL macro, say for stacker selection, will probably run slower in paged mode than in nonpaged mode. The reduction in performance is the result of additional control program processing that is required to perform channel program translation and page fixing.

Note that executing such programs in nonpaged mode can improve the performance of that individual program but can also degrade total system performance by making less real storage available for paging. Nonpaged mode should be used only when really necessary, as indicated in the performance discussion contained in the prerequisite base publications for this supplement (Section 15:15 or 30:15).

The pageable area consists of all the virtual storage above the V=R line. It contains a pageable supervisor area and a partitioned area for the execution of pageable problem programs. The pageable supervisor area contains control program routines that are resident in virtual storage (have virtual storage allocated and are contained in external page storage) and, therefore, are subject to paging. Optionally, certain control program functions can be made fixed instead of pageable.

The pageable supervisor area is located in highest addressed virtual storage and has a protect key of zero. It has:

- An area of 4K reserved for the page supervisor
- The pageable supervisor routine area of 30K minimum, which contains supervisor routines that can be paged instead of fixed with minimal effect on system performance. The pageable supervisor routine area contains ATTACH, the communications task, DISABLE, enqueue/dequeue, EXTRACT, FIND/BLDL/CONVTTR, IDENTIFY, LINK/LOAD/XCTL/FINCH, the master scheduler, standard program fetch (unless PCI fetch is included), SEGLD/SEGWT, SPIE, SYNCH, and, if only one partition is defined, TIME. This area also contains the SVC transient area, which is 2K bytes (instead of 1K, as in MFT) and the I/O transient area of 1K.
- The resident supervisor routines area, which contains certain access methods, and the BLDL table. The BLDL table can be pageable or fixed. Optionally, reentrant routines from SYS1.LINKLIB, Type 3 and 4 SVC routines, and ERP routines can be made resident in this area

also. The contents of this area are indicated at system initialization. The routines included can be pageable or fixed except for ERP's which must be fixed.

- The job entry subsystem area, which contains JES routines and buffers. The JES virtual storage area is a minimum of 96K (one reader, one writer, one spool device, and minimum buffers) and is pageable. At system initialization, the JES virtual storage area is made large enough to support the maximum number of readers and writers (for both JES and RES) that can be started for this IPL.

The problem program virtual storage area is divided into contiguous partitions, just as in MFT. A maximum of 52 partitions (P0 to P51) can be defined: one to 15 problem program partitions, with a nonzero protect key (1-15), and one to 37 system task partitions, with a protect key of zero. P0, in the highest addressed portion of the problem program area, has the highest execution (task dispatching) priority and P51 has the lowest. A partition in virtual storage must be a multiple of 64K in size (the segment size used) and begin on a 64K virtual storage boundary. Small partitions, as defined in OS MFT (those too small to contain the job scheduler), do not exist in OS/VS1.

Jobs are scheduled to execute in problem program partitions by job priority within job class (A to O), just as in MFT. However, a VS1 problem program partition can have up to 15 job classes assigned, instead of the MFT limit of three. A given job class can be assigned to more than one partition, as in MFT. The number of partitions, their size, their type, etc., is defined at system generation. These parameters can be altered by the operator at IPL or during system operation as long as the maximum number of partitions or the maximum amount of virtual storage specified at system generation or initialization is not exceeded.

Problem program partitions are used for the scheduling and execution of pageable job steps and direct system output (DSO) writers. A pageable job step can be initiated in a problem program partition as long as the size of the program to be executed is at least 4K less than the partition size. (The 4K is required for system control blocks and tables.) Paged job steps execute with storage addresses in instructions translated by the DAT hardware and storage addresses in channel programs translated by the control program. Direct system output writers in VS1 are functionally equivalent to those in MFT.

Virtual storage within a problem program partition consists of:

- Fixed protected queue area (PQA) of 2K minimum with a protect key of zero

- Pageable protected queue area (PQA) of 2K minimum with a protect key of zero

- Problem program area consisting of all virtual storage in the partition not allocated to fixed or pageable PQA. This area has a nonzero protect key.

The fixed PQA contains partition-related control information that is not paged for system integrity and reliability reasons. For example, the fixed PQA contains the page tables required for the partition and request blocks (PRB's, IRB's, etc.). When a partition is defined, either during IPL or when the operator enters the DEFINE command, at least one virtual storage page is allocated for PQA from the highest addressed virtual storage in the partition, and a page frame is assigned to it and fixed. The number of virtual storage pages required for fixed PQA at partition initialization time depends on the size of the virtual partition and whether or not PCI fetch is included in the control

program. (The amount of fixed PQA required at partition initialization in bytes is 500, plus 1600 if PCI fetch is present, plus one byte for each 1024K bytes of virtual storage in the partition rounded to the next multiple of 2048.) If additional fixed PQA space is needed during problem program execution, it is taken, a virtual storage page at a time, from the highest available problem program virtual storage in the partition.

The pageable PQA contains control blocks that can be paged with minimal effect on performance. No pageable PQA is allocated when the partition is initialized. When a job step is initiated, a 2K page of pageable PQA is allocated from the highest addressed available virtual storage in the partition. Pageable PQA can be expanded during problem program execution. Pages are allocated from highest available problem program virtual storage in the partition.

At partition initialization time, the problem program area contains all virtual storage in the partition below fixed PQA. When a job step is initiated, virtual storage is allocated to it beginning with the lowest addressed virtual storage in the partition. Virtual storage above this is then available for allocation to the fixed and pageable PQA's and to the problem program as needed. The maximum amount of virtual storage available to a pageable problem program is, therefore, at least 4K less than the size of the partition in which it operates.

Problem program partition organization in VS1 offers integrity advantages over MFT partition organization in that all control blocks contained within a VS1 virtual partition are protected from modification by the problem program.

System task partitions are designated with the identification STP instead of with job classes. They have the same organization as problem program partitions and operate in problem program state; however, system task partitions have protect key zero assigned. Problem programs cannot be executed in a system task partition. The Generalized Trace Facility, the MOUNT command, START RDR/WTR, STOP RDR/WTR, and conversational remote job entry readers are system tasks that are authorized to operate in a system task partition.

Authorized system tasks can also be executed in problem program partitions; hence, a system task partition does not have to be defined. A system task partition can be defined for the purpose of executing operator commands, such as MOUNT, START RDR/WTR, and STOP RDR/WTR, that must operate in a partition. If a system task partition is available when one of these commands is issued, the command can be processed immediately without waiting for a problem program partition to become available.

Unlike main storage areas in MFT, certain virtual storage areas in VS1 need not be contiguous with each other. There can be undefined virtual storage between PO, which must be a multiple of 64K located on a 64K boundary, and the pageable supervisor area, which is allocated from the top of virtual storage down and need not be a multiple of 64K in size. There cannot be any undefined virtual storage between virtual partitions but there can be between the nonpageable area (V=R line) and the pageable area (lowest priority pageable partition starting address). This condition can occur when the real storage size of the system is less than 768K and not a multiple of 64K in size (144K, 240K, for example). In a Model 145, available real storage may not be a multiple of 64K in size because more than 32K is required for control storage.

REAL STORAGE ORGANIZATION

Real storage is also divided into a nonpageable and a pageable area. The nonpageable area in lowest addressed real storage is allocated to the nonpageable area of virtual storage on a V=R basis. It contains the nonpaged (fixed) resident control program (nucleus, RMS, and defined SQA). With a few exceptions, resident control program routines operate with translation mode specified even though they are not paged. This approach is taken because the resident control program accesses virtual storage addresses at various times during its execution and address errors would occur at these times if translation was not operative.

Page frames in the nonpageable area above the resident control program up to the V=R line are allocated to both pageable and nonpageable virtual storage pages. Page frames are allocated to SQA and nonpageable job steps only from available real storage below the address of the V=R line. However, available real storage below the V=R line can also be allocated to pageable supervisor routines, pageable partitions, and PQA, when necessary. Page frames above the V=R line are allocated to pageable supervisor routines, pageable partitions, and PQA but cannot be allocated to SQA or nonpageable job steps. (Real storage allocation is discussed in more detail in Section 90:35.) Figure 90.10.2 shows the organization of real storage.

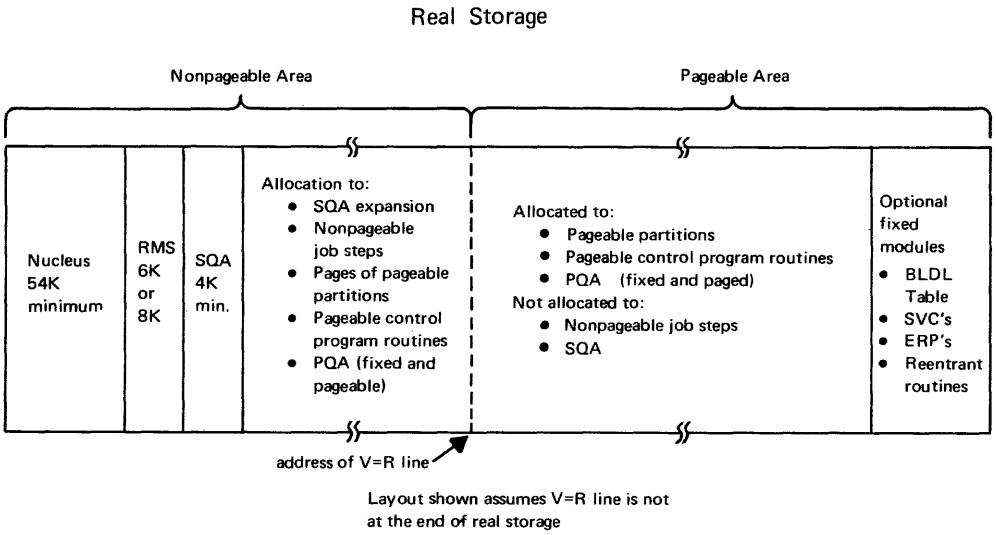


Figure 90.10.2. Real storage organization in OS/VS1

EXTERNAL PAGE STORAGE ORGANIZATION

External page storage is used to contain the contents of the pageable virtual storage area, which consists of all virtual storage between the V=R line and the end of defined virtual storage. Only fixed PQA pages and any fixed control program routines, such as SVC's, etc., allocated within the pageable area are not written on external page storage. The direct access devices supported as paging devices are the 2314/2319, 3330-series, and 2305 Model 2.

The direct access storage allocated as external page storage is called the page file (SYS1.PAGE data sets). The page file can consist of up to eight page data sets, each of which must be a single extent only and totally contained on one direct access device. A maximum of

two direct access device types is supported for the page file. While direct access devices that contain a page data set need not be dedicated to paging, this approach is recommended for performance reasons. Page file disk volumes must be permanently resident. The IOS priority queuing option should be specified for direct access devices that contain page data sets to ensure that paging I/O requests receive the highest priority on their associated channels.

A track in a VS1 page data set contains a number of 2K record areas called slots. Regardless of the direct access device type used, page data set tracks are formatted with a dummy record written after each 2K slot. The dummy records are added to increase paging performance by allowing time for electronic head switching while accessing multiple pages contained within the same cylinder using a command-chained channel program. The track overflow feature is not used because, for a 2K record size, it yields no significant space benefit on the supported devices. The number of 2K slots available per device is shown below.

	2314/2319	3330	2305-2
Slots per track	3	5	6
Slots per cylinder	60	95	-
Slots per pack	12,000*	38,285*	4,608

* Maximum number of 2K slots required is 8,192 (for 16 million bytes of virtual storage) less 384 or the number of page frames in real storage, whichever is less.

The page file must be able to contain a number of 2K slots equal to or greater than the number of 2K virtual storage pages that exist between the V=R line and the end of virtual storage as follows:

$$\text{Number of 2K slots required} = \frac{(\text{defined virtual storage size}) - (\text{real storage size or 768K, whichever is less})}{2K}$$

External page storage is statically mapped on a one-to-one basis with virtual storage above the V=R line. That is, the contents of any given virtual storage page are always placed in the same slot and the first virtual storage page is associated with the first slot in external page storage, etc. (See Figure 90.10.3.)

SYSTEM INITIALIZATION

At the completion of the VS1 IPL procedure, which is much like that required for MFT, EC and translation modes are operative. During IPL, virtual, real, and external page storage are initialized as follows.

Virtual Storage

At IPL time, a virtual storage of the size specified at system generation is initialized, unless the size is smaller than 512K plus real storage size or the operator overrides the system generation specification. The operator can increase or decrease the amount specified at system generation but cannot specify less than the size of real storage plus 512K. If a virtual storage size is not specified at system generation, the default is 1024K or real storage size plus 512K, whichever is larger. Virtual storage size must be a multiple of 64K.

The initialization of virtual storage consists of building the control blocks and tables required to define the various areas of virtual storage that are shown in Figure 90.10.1. Virtual storage is mapped according to system generation parameters and any additional

definitions specified in SYS1.PARMLIB or overrides indicated by the operator. Control program modules that are to be made resident in virtual storage and paged are allocated virtual storage, fetched from load module libraries, and paged out to external page storage as a result of normal paging activity. There is no routine in VS1 that forces the page out of pageable load modules that are fetched during IPL or thereafter. If real storage becomes full during the loading of modules, pages are written out as per the page replacement algorithm.

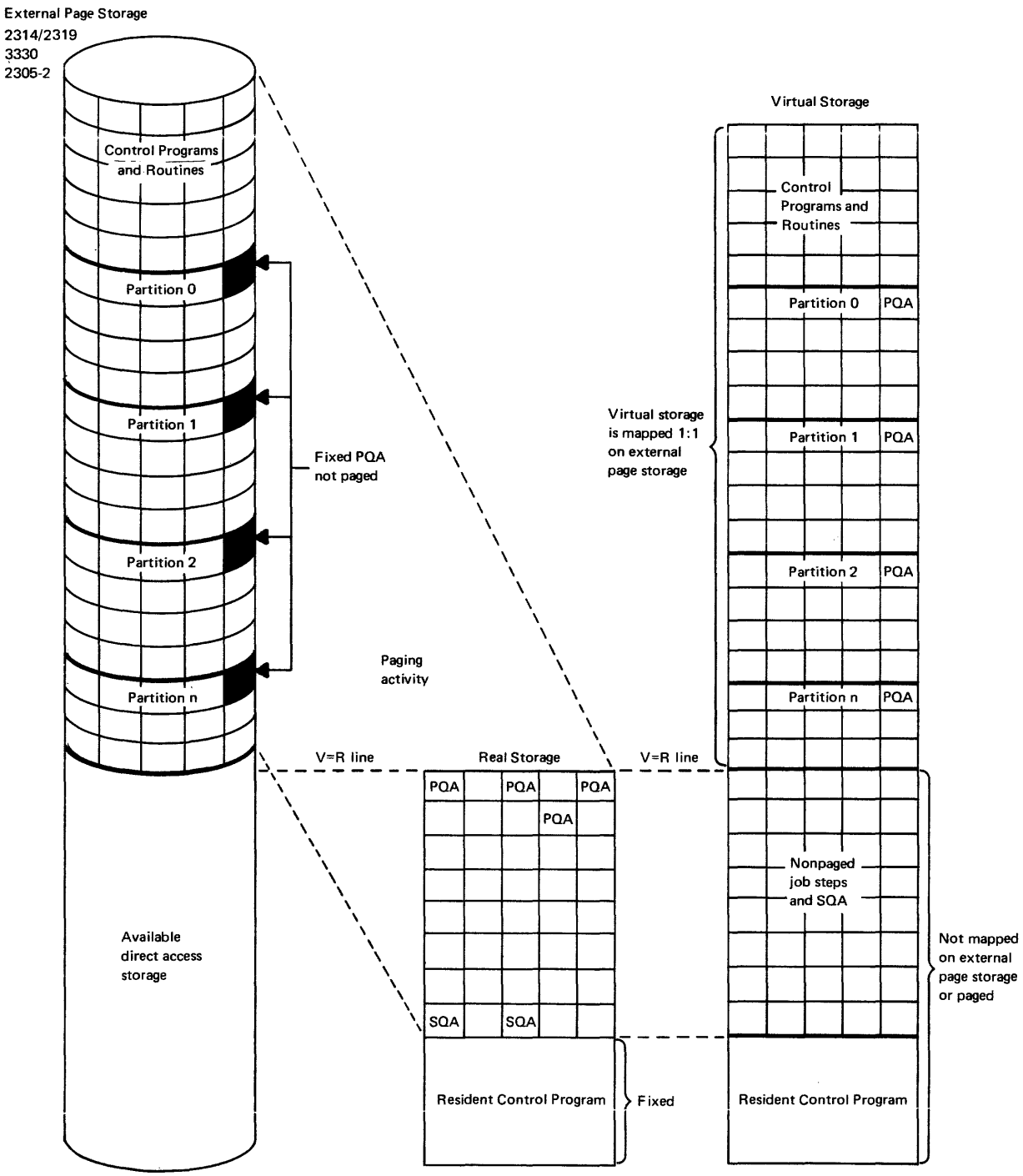


Figure 90.10.3. External page storage, real storage, and virtual storage relationship in OS/VS1

During IPL, the segment table is built at the end of the nucleus. The size of the segment table is dependent on the size of the virtual storage established for this IPL. Virtual storage for page tables is also allocated during IPL. The page tables for all virtual storage areas except the problem program area (pageable partitions) are allocated in SQA. Page tables for the pageable partitions defined are allocated within the partitions themselves. One or more virtual storage pages of fixed PQA are allocated in the high end of each initialized partition with space reserved in them for the page tables required to address the entire partition.

Real Storage

At the completion of IPL, real storage contains:

- The resident control program (in the nonpageable area in lowest addressed real storage) and any resident supervisor routines that are fixed (in the high end of real storage)
- 4K or more of SQA (adjacent to the resident control program)
- At least one page frame of fixed PQA for each partition defined and initialized
- A minimum of three fixed pages for JES

The minimum VS1 control program (one partition and no fixed supervisor routines) requires 72K of fixed real storage (54K for the nucleus, 6K for RMS, 4K of SQA, 2K of fixed PQA, and 6K of fixed real storage for JES).

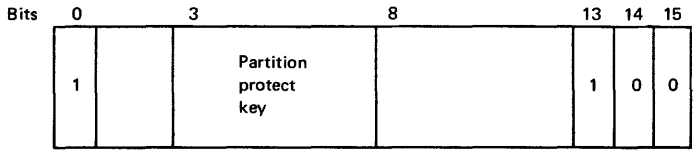
In VS1, a certain amount of real storage is reserved to be available for paging, for SQA or PQA expansion, and for short-term fixing. A minimum of eight page frames must be available for paging. Two page frames must be available to extend SQA or PQA, if necessary, and enough page frames must be available to satisfy the largest expected I/O request for short-term page fixing. In VS1, 36K of real storage is always reserved to remain available for these operations, regardless of the amount of real storage in the system.

Control blocks and tables are initialized to reflect the size and organization of the virtual storage defined, as well as the real storage present and allocated. The segment table in the nucleus reflects the current size of virtual storage, and the segment entries have their invalid bit off to indicate that page tables have been built and initialized. The page table entries for the virtual storage that has real storage allocated (resident control program, fixed PQA, etc.) have their invalid bit off, while the entries for problem program virtual storage have their invalid bit on.

A page table entry for a problem program partition is initialized as shown in Figure 90.10.4. Bit 0 is on to indicate that bits 3 to 7 contain the nonzero protect key of the partition. The invalid bit is on and the user bit is off. When off, the user bit indicates that a page-in is not required after a page frame has been allocated to the virtual storage page. The user bit is turned on the first time real storage is allocated to this virtual storage page after job step initialization. At job step termination, the user bit is turned off.

The first time any problem program virtual storage page is referenced, a page frame is allocated without a page-in and cleared to zeros (for security protection). The protect key value from the page table entry is inserted in the protect key of the page frame allocated. Bit 0 and the invalid bit are turned off and the user bit is turned on

in the associated page table entry. The high-order bits of the address of the allocated page frame are placed in the page table entry.



Bit	
0	When 1, bits 3 to 7 contain partition protect key
3–7	When bit 0 is a 1, these bits contain the partition protect key
13	Invalid bit on to indicate real storage page not allocated
15	User bit off to indicate a page-in is not required to allocate a page frame

Figure 90.10.4. Page table entry contents for an initialized, inactive problem program partition

External Page Storage

The location, in terms of unit address(es) or volume serial number(s), and the size of the page data sets in the page file must be specified at system generation. The operator can override this specification at IPL. The first time a given volume is used for a page data set, space is allocated and slots are formatted. Thereafter, the page data set can be used without reformatting as long as the same or a lesser amount of space is allocated. Formatting can be requested by the operator.

If a specified page data set volume is found to be unmounted during IPL, it is bypassed with a message to the operator indicating the volume was not mounted. The operator is notified if the page file space allocated is not large enough to contain the virtual storage space above the current V=R line.

At the completion of IPL, external page storage may contain some of the pageable control program load modules that are resident in virtual storage (if any were paged out during IPL).

90:15 MAJOR COMPONENTS

The major control and problem program components of OS/VS1 are shown in Table 90.15.1. Except for the integrated emulator programs and TCAM, components identified as SCP are distributed as part of VS1. Integrated emulators and TCAM are distributed separately. Type I programs and program products are not distributed as part of VS1 and must be obtained individually.

The division of control program routines in VS1 and MFT is similar. Both have job, task, data, and recovery management functions. However, OS/VS1 also has a page management function that is responsible for managing both real and external page storage. Virtual storage is allocated and maintained by the storage supervisor of task management that manages main storage in an MFT environment.

Table 90.15.1 OS/VS1 control and processing program components

OS/VS1	
CONTROL PROGRAM COMPONENTS (SCP)	
Job Management	**Task Management**
• Master scheduler and communications task • Job Entry Subsystem (JES) Job entry peripheral services (JES readers, JES writers) Job entry central services • Job queue manager • Job scheduler Initiator Interpreter Allocation Terminator Direct SYSOUT writers • System Management Facilities (SMF) • Remote Entry Services (RES) • Conversational Remote Job Entry (CRJE)	• Interruption supervisor • Task supervisor • Virtual storage supervisor • Contents supervisor • Timer supervisor • Overlay supervisor **Page Management** • Page exception handler • Page supervisor Real storage management External page storage management
Data Management	**Recovery Management**
• Input/output supervisor • Access methods QSAM, BSAM, QISAM, BISAM, VSAM, BDAM, BPAM, BTAM, TCAM, GAM • Catalog management • Direct Access Device Space Management (DADSM)	• Machine Check Handler (MCH) • Channel Check Handler (CCH) • Alternate Path Retry (APR) • Dynamic Device Reconfiguration (DDR) • Online Test Executive Program (OLTEP)* • Problem determination facilities
PROBLEM PROGRAMS (SCP AND PP)	
Language Translators	**Service Programs**
• System Assembler (SCP) • Assembler H (PP) • Full ANS COBOL V3, V4, and Libraries (PP) • PL/I Optimizing Compiler (PP) • PL/I Checkout Compiler (PP) • PL/I Resident and Transient Libraries (PP) • FORTRAN IV G (PP) • FORTRAN IV H Extended (PP) • FORTRAN IV Libraries - Models 1 and 2 (PP) • Code and Go FORTRAN (PP) • ITF PL/I - Release 2 (PP)* • ITF BASIC - Release 2 (PP)* • System/7 FORTRAN IV System/370 Host Compiler and Library	• Linkage Editor (SCP) • Loader (SCP) • Utilities System and data set utilities (SCP) Data set utilities with ASCII (PP) • Basic Unformatted Read System (PP) • Sort/Merge 5734-SM1 (PP) **Integrated Emulators** • DOS Emulator (SCP) • 1401/1440/1460 (SCP) • 1410/7010 (SCP) • 7070/7074 (SCP) • 7080 (SCP) • 709/7090/7094/7094II (SCP)
General • Application-oriented program products (some operate in paged mode and some in nonpaged mode)	
*Must operate in nonpaged mode	

Table 90.15.1 (continued)

PROBLEM PROGRAMS - TYPE I AND USER-WRITTEN	
Language Translators	Service Programs
• COBOL to ANS COBOL LCP (360-CV-713) • COBOL F (360S-CB-524) • COBOL F Library (360-LM-525) • PL/I Syntax Checker (360S-PL-552) • Full ANS COBOL Version 2 (360S-CB-545) and Library (360S-LM-546) • FORTRAN G (360S-FO-520) • FORTRAN H, Version 2 (360S-FO-500) • FORTRAN Library (E,G,H) (360S-LM-501) • FORTRAN Syntax Checker (360S-FO-550) • PL/I F (360S-NL-511) • PL/I Subroutine Library (360S-LM-512) • PL/I Syntax Checker (360S-PL-552)	• Sort/Merge (360S-SM-023) General • User-written programs compiled using the Type I language translators listed • User-written programs compiled using program product language translators

The new features of VS1 and the most significant differences between VS1 and MFT components are presented in the discussions that follow. VS1 uses the same system data sets and libraries as are used in MFT. VS1 also uses one new required library, SYS1.DSSVM, which is required by DSS and discussed in Section 90:40, and new required data sets--SYS1.PAGE (for the page file), SYS1.SYSPOOL (for JES spool data sets), and SWADS (a scheduler work area data set for each initiator). If RES is used, the new SYS1.UADS and SYS1.BRODCAST data sets are required also. All the direct access devices supported by VS1 for disk data sets are also supported as system residence devices.

VS1 supports all the primary operator console devices required for Models 135 to 168. The DIDOCS option with 3270 support must be included in a VS1 control program to support 3270 (display) mode operations on the Model 158 display console. DIDOCS is also required to support the display console contained in the 3066 standalone console unit for the Models 168 and 165 II. The 3213 printer is supported only as a hard-copy output device for the Model 158 display console and not for input operations.

90:20 JOB MANAGEMENT

VS1 and MFT job management functions are logically the same, and externally the VS1 job management interface with the operator is upward compatible with that of MFT. The internal organization of job management in VS1 and MFT differs considerably, however. VS1 job management has been modified to operate in a paging environment, and it is designed to offer reduced real storage requirements, improvements in performance, and new functions. The organization and new features of VS1 job management are designed to provide the following:

- More efficient handling of peripheral I/O operations (JES)
- Enhanced support of remote job entry (RES)
- More system configurability without regeneration
- Enhanced operator command processing

- Additional operator control (WRITER command)

- Improved job scheduling via elimination of small partition scheduling and significant reductions in contention for the job queue (implementation of SWADS)

MASTER SCHEDULER AND COMMUNICATIONS TASK

As in MFT, the master scheduler and the communications task in VS1 handle initialization functions at IPL, and operator/system communication. These routines operate in the pageable supervisor area. Most commands are processed in the 2K SVC transient area, which is pageable. Certain of these command processing routines are repackaged in VS1 to execute in 2K multiples (instead of 1K) to increase their performance. Some commands must operate in a partition (as in MFT) and a system task partition can be defined for this purpose, as indicated previously.

All MFT operator commands and parameters and their formats are accepted in VS1 except those associated with MFT features that are not supported in VS1. Only the SET command has an additional parameter, SPOOL, which can be used to change the spool configuration specified at system generation and to cause formatting of spool volumes. In VS1, the SET command is issued after IPL to change the time or date, but not during IPL.

Modifications or extensions to the functions performed by the following commands have been made:

- DEFINE - This command handles virtual storage partition allocation as defined for VS1.

- MODIFY - Up to 15 job classes are accepted.

- DISPLAY - The class, priority, queue location, and position on the queue for an active job are displayed (not given in MFT for active jobs).

- MODE - A simplified format is used that is applicable to all System/370 models. The operator can no longer establish threshold values for ECC errors and instruction retry errors.

- START - This command provides the capability of specifying a reader or a writer procedure without a partition indicated. The procedure is initiated using the next available partition.

VS1 supports new commands associated with RES (discussed under "Remote Entry Services" later in this subsection). One other new operator command, WRITER, is supported that enables the operator to communicate requests to a JES writer. Like DEFINE and HALT, a WRITER command can be entered only via the operator console (not via an input stream). The WRITER command gives the operator significantly more control over executing writers than is provided in MFT. Using the WRITER command, the operator can:

- Request up to 255 additional copies of output (printed, punched, or written to tape), on a data set or a job basis

- Stop the writing (to printer, punch, or tape) of a data set immediately and begin writing it again from the beginning

- Stop the writing (to printer, punch, or tape) of a data set and have the writer continue with the next data set

- Request that printing continue up to 255 pages ahead of the current logical page (forward space) or up to 100 pages before the current logical page (backspace)

- Terminate the printing of a data set and reenqueue it on the class queue from which it was dequeued or on a different SYSOUT queue. Printing can be resumed at the beginning of the data set or at the point at which it was terminated.

- Alter printer line spacing (single, double, triple) for the current data set

JOB ENTRY SUBSYSTEM

The job entry subsystem (JES) is a significant new feature of VS1. It replaces MFT readers and writers and HASP II. JES provides centralized management of system input and system output data. It handles local system input and system output streams, allocates and manages intermediate direct access storage for this data, and interfaces with RES to handle remote input and output streams. JES is designed to maximize utilization of the unit record and direct access devices involved in peripheral I/O processing. It also supports a full checkpoint/restart capability.

In VS1, JES places system input and output on direct access volumes called the SYS1.SYSPOOL data set. Logically, the data stored in SYS1.SYSPOOL is placed in spool data sets. However, as discussed later, a spool data set within the SYS1.SYSPOOL data set does not have the same characteristics as an OS data set, and it is processed by JES routines instead of OS access methods. Reading input streams and writing the data onto spool volumes, and reading system output data from spool volumes and writing the data to system output devices is called spooling in VS1.

Spooling operations in JES are centralized such that all system input reading, system output writing, and spool volume processing are controlled by one set of modular routines. Centralization eliminates duplication of functions within the system and improves the performance of spooling operations. JES reader code and writer code are reentrant, which reduces the amount of storage (both virtual and real) required to service multiple input and output streams. Because all JES routines are totally pageable, real storage is allocated only to active JES tasks and without operator intervention. (In MFT, for example, main storage allocated to an inactive reader or writer partition cannot be used by other active partitions unless the operator intervenes to redefine partition allocation or to change the type of the reader/writer partition.) The entire JES area is pageable, except for two or more pages that contain tables required by JES. Two pages are fixed in the JES area when one reader and one writer are active. Additional pages are dynamically fixed as required when more readers and writers are started. JES also requires one page of fixed PQA.

Centralization of control also enables JES spooling operations to be performed more efficiently. Buffer storage for all readers and writers is contained in one pool, buffer storage for all spool volumes is contained in one pool, and direct access spool space (SYS1.SYSPOOL data set) for all system input and output data is shared. Buffer storage and spool space are managed (allocated, opened, closed, and deallocated) by JES routines that are tailored to provide efficient spooling operations in a paging environment. Buffer storage is allocated for JES operations such that the number of page faults incurred is minimized, and direct access storage is allocated such that data transfer time for JES operations is minimized.

JES readers do not interpret job control statements as do MFT reader interpreters. The interpreter in VS1 is a subroutine of the initiator. This organization, together with support of command chaining, can allow a card reader to operate near its rated hardware speed, since reading is not delayed by interpretation. Therefore, jobs can be placed in the job queue more quickly.

Any number of readers and writers are supported by JES, subject only to the availability of system resources. MFT supports three readers and 36 writers maximum. The maximum number of readers and the maximum number of writers that can be started for a given VS1 control program (both JES and RES requirements) can be specified at system generation. The size and number of spool buffers and the number of spool volumes can also be indicated at system generation. Defaults are assumed for parameters not specified. At IPL, the system generation JES parameters are used unless overridden by the new JESPARMS entry in SYS1.PARMLIB. Thus, the number of readers and writers supported can be increased or decreased without a system generation. During IPL, enough virtual storage is allocated to the JES area to support the maximum JES configuration indicated during IPL. Virtual storage from this JES area is allocated as readers and writers are started.

The virtual storage requirement for the minimum JES configuration (one reader, one writer, one spool volume, and minimum buffering) handling one partition is 96K.

In summary, JES offers the following significant overall advantages when compared with MFT readers and writers:

- More efficiently managed peripheral I/O operations through centralization of control and use of resource allocation algorithms that are specifically designed to improve spool performance
- Reduced virtual and real storage requirements for spooling operations involving multiple readers and writers
- More efficient use of real storage for peripheral operations since real storage is allocated to a JES component only when it is active
- Ability to handle more readers and writers
- Ability to increase the number of readers and/or writers handled by a VS1 control program without generating a new system
- Continuously available reader for unit record SYSIN devices
- Additional operator control over writers

JES functions are performed by job entry peripheral services (JEPS) and job entry central services (JECS) routines. The components of JEPS and JECS are:

JEPS	JECS
• Monitor task	• Spool management
• JES readers	• Buffer management
• JES writers	• DASD work area management

The components, functions, and data flow of JES are shown in Figure 90.20.1.

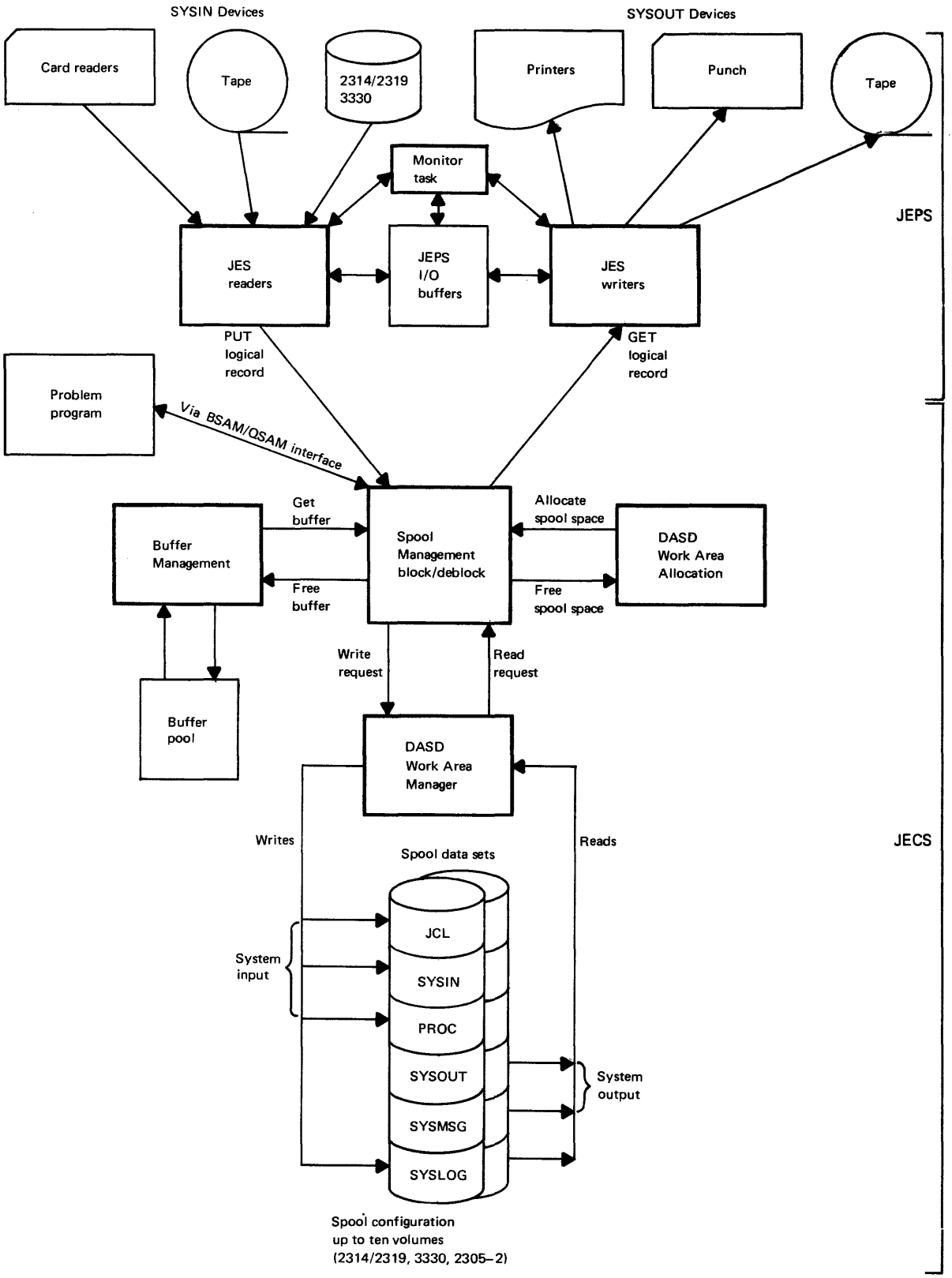


Figure 90.20.1. Components, functions, and data flow of JES

JEPS Monitor Task

The monitor task is responsible for initializing JES readers and writers. When a START or a STOP command for a reader or a writer procedure is entered, an initiator or terminator is brought into the first partition available in order to invoke the monitor. The monitor task is also automatically invoked to terminate a reader task when end-of-file occurs on a SYSIN device other than a card reader. The monitor task obtains or releases buffers for the reader/writer from the reader/writer virtual storage area allocated in the JES area during IPL, and attaches or detaches the reader/writer task. A reader/writer is started as long as this request does not cause the maximum number of readers/writers specified at IPL to be exceeded. OS MFT reader and writer procedures are not compatible with those of JES.

JES Readers

An input stream can be contained on any card reader or tape unit supported by VS1 or on disk (2314/2319 and 3330-series). Card input streams are read by a special JES access method (JAM). Column binary reading is not supported. Tape and disk input streams, which can be blocked, are read using a special interface to QSAM. Command-chained reads are initiated for a card SYSIN device if the block size specified is a multiple of the logical record length or if block size is not specified in the reader procedure. For example, if a block size of 400 is specified, a channel program designed to read five cards is initiated for each I/O operation to the card SYSIN device.

SYSIN reading starts at the beginning of the input stream or at the job name indicated in the START command. Reading continues until a STOP RDR command is issued by the operator or, for tape and disk SYSIN only, until end-of-file occurs. A reader task that is handling a tape or disk SYSIN device is terminated on an end-of-file condition. When end-of-file occurs on a card reader, the JES reader enters the wait state (and the real storage assigned to it tends to become available for allocation to other tasks). When the card reader is made ready again, reading automatically continues. This continuously available reader facility is not provided in MFT for card SYSIN devices.

Input stream data can consist of job control statements (including in-stream procedures and requests to execute procedures), input data sets (multiple per job step), and operator commands. A JES reader inspects the JOB statement, and defaults for job class and priority are supplied, if necessary. A unique job number is appended to the job name specified to eliminate the possibility of duplicate job names. This job number is used internally only. Job control statements, commands contained within jobs, input data, and any requested procedures from user procedure libraries are passed to the spool management routine of JECS to be written in spool data sets. Procedures that are contained in SYS1.PROCLIB are not written in spool data sets. Commands not contained within a job are processed when encountered.

When a job has been completely read, the JES reader builds a disk entry record (DER) that describes the job and passes it to the job queue manager, which places the DER in a job class queue or in the hold queue, as indicated in the JOB statement. The total number of statements read and the real time taken to read the job is made available to SMF, if SMF is included in the control program.

JES Writers

A JES writer writes SYSOUT spool data sets created by job steps and system message spool data sets that contain job scheduler messages and

job control statements. The SYSOUT devices supported are printers, punches, and tape units supported by VS1. A special JES access method is used to support printers and punches. Column binary punching is not supported. An interface to QSAM is used to handle a tape SYSOUT device. As with card readers, command chaining is used for print and punch operations when the block size specified is a multiple of the logical record length or if block size is not specified in the writer procedure to enable these devices to operate near rated speeds.

A JES writer is initiated using a START command and terminated using a STOP command. Each JES writer can handle up to eight SYSOUT classes and more than one writer can be assigned to the same SYSOUT class. Thirty-six output classes (A-Z, 0-9) are supported, as in MFT. The writing of SYSOUT data for a job does not begin until the job itself is terminated, just as in MFT. A JES writer handles all the SYSOUT spool data sets of the same class that are present for a given job before attempting to process SYSOUT spool data sets of the same class that belong to another job. All the job control statements and system messages for a job are printed before all the SYSOUT spool data sets for the job. (MFT writers intersperse the printing of job control statements and system messages with the printing of SYSOUT data sets.) After all the SYSOUT data for a job has been written, the job is purged from the system, accounting data is supplied to SMF, if appropriate, and the spool space allocated to its SYSOUT spool data sets is released.

Multiple copies of a given SYSOUT spool data set can be requested via the new SYSOUT DD statement parameter COPIES, in addition to via the WRITER command.

User-written output writers that use BSAM/QSAM and job separator routines that operate with MFT do not require modification for operation under VS1.

Problem Program Access to SYSIN and SYSOUT Data

In VS1, problem programs access SYSIN and SYSOUT spool data sets via QSAM and BSAM, as in MFT. However, in VS1, these sequential access methods interface with a device independent JES translator which interfaces with JECS to access SYSIN and SYSOUT spool data sets on spool volumes. The JES translator module is automatically invoked when a SYSIN/SYSOUT data set is specified by a job step. The translator module is entered each time the problem program requests the reading of a SYSIN record or the writing of a SYSOUT record. The JES translator reformats the request as necessary and passes it to JECS. When JECS has processed the request, this fact is indicated to the JES translator which posts the appropriate control blocks and, in the case of a SYSIN request, makes the record available to the problem program.

The interface to the JES translator is transparent to the problem program. Thus, the MFT approach of using QSAM or BSAM to access SYSIN and SYSOUT spool data sets is valid in VS1, and modifications to the SYSIN/SYSOUT data set processing contained in existing MFT programs are not required in order to execute these programs under VS1. SYSIN and SYSOUT spool data sets cannot be accessed via the EXCP macro in VS1 because there is no interface to the JES translator from this macro.

JECS Spool Management

Spool management is the central facility that controls all access to spool data sets, as was shown in Figure 90.20.1. It receives and processes service requests from JES readers, JES writers, job scheduler components, and executing problem programs. Spool management processing

consists of blocking and deblocking system input and output records and requesting services from other JECS components.

Spool management interfaces with JECS buffer management to obtain the I/O buffers required to read and write spool data sets. Spool management interfaces with DASD work area management to obtain and to free direct access space for spool data sets and to request I/O operations on spool data sets.

The OUTLIM facility, available to MFT users only via SMF, is a standard feature of JES. This facility allows the user to indicate the maximum number of logical records that are to be placed in a SYSOUT spool data set. Spool management also ensures that the OUTLIM quantity for SYSOUT spool data sets is not exceeded and maintains control blocks that indicate all the system input and output data sets associated with each job.

JECS Buffer Management

The pool of I/O buffers available to be used for reading and writing all spool data sets is maintained by buffer management, which services requests from spool management only. The buffer pool is contained in the pageable JES area of virtual storage. The pool is allocated during IPL and its size cannot be increased without a re-IPL. The size and number of buffers can be specified at system generation and these values can be overriden by changing the JESPARMS member in SYS1.PARMLIB.

The number of buffers required for optimum spool performance is a function of the number of JES readers, JES writers, partitions, and opened spool data sets that can be active concurrently. If too few buffers are provided, loss of spool performance can occur. Hence, it is better to overestimate than to underestimate buffer requirements. The allocation of more spool buffers than are actually required does not affect system performance since real storage is allocated to spool buffers only if they are used. A maximum of 999 buffers can be specified. (See OS/VS1 Storage Estimates, GC24-5094, for estimating spool buffer needs.)

The formula given for estimating buffer requirements provides one buffer for each spool data set possible. The buffer is allocated when the spool data set is opened and, normally, is released when the spool data set is closed. When allocating a buffer, buffer management always looks first for an available buffer that is contained in a virtual storage page that already has buffers allocated. This is done to ensure that the minimal number of virtual storage pages are used for allocated spool buffers. This approach minimizes both page faults and the amount of real storage allocated for buffers at any given time during JES activity.

If the buffer pool is empty, buffer management attempts to obtain a buffer that is assigned to another spool data set. However, if all assigned spool buffers are currently in use, the buffer request cannot be satisfied until an assigned buffer becomes available or until a buffer is freed and returned to the buffer pool. This buffer preempting for the purpose of buffer sharing can occur only if the total spool configuration is active and the number of spool buffers allocated is less than the number calculated using the spool buffer requirements formula.

DASD Work Area Management

The allocation and deallocation of direct access space to spool data sets, and the reading and writing of spool data sets are handled by the

DASD work area manager and the DASD work area allocation routine, respectively. Spool volumes contain JCL, SYSIN, PROC, SYSOUT, and system message spool data sets, write-to-programmer messages, and the two system log data sets.

The spool volume configuration (DASD work area) can consist of up to ten permanently mounted direct access volumes. Any mixture of the following direct access device types can be included in the spool configuration (SYS1.SYSPOOL data set): 2314/2319, 3330-series, 2305-2. The volume serial numbers of the volumes in the spool configuration are indicated at system generation. Spool volumes are mounted prior to IPL. If the specified volumes are not mounted, they are deleted from the spool configuration during IPL. The new SPOOL parameter of the SET command permits the operator to alter (add to, delete from, change) the spool configuration at IPL, after which a re-IPL is required to alter the spool volume configuration. Spool devices need not be dedicated to spooling; however, arm movement can be minimized and increased performance obtained when spool devices are dedicated.

Spool data sets are sequentially organized and contain variable length blocked spanned records. For card input, blanks after the last character punched are deleted from the resulting logical record. For printer and punch output, blanks after the last character to be printed in a line or punched in a card are deleted from the logical record. This truncation eliminates using spool space to store insignificant blank characters.

Spool volumes must be preformatted with physical records the size of the spool buffers. Preformatting is done during an initial IPL when the operator requests spool volume formatting, or because it is determined that spool volumes require formatting. Reformatting is not required thereafter unless spool buffer size is changed. The buffer size chosen must be a minimum of 436 bytes and cannot be larger than the full track capacity of the smallest capacity track in the spool device configuration. Therefore, if a 2314/2319 is part of the spool configuration, buffer size cannot exceed 7294 bytes regardless of the other direct access device types in the spool configuration. The track overflow feature is not supported for spool data sets, but rotational position sensing is used when it is present for a spool device.

Direct access space on spool volumes is allocated to spool data sets in terms of a logical cylinder instead of a physical cylinder. A logical cylinder consists of a number of tracks, all of which need not be contained in the same physical cylinder. The number of tracks in a logical cylinder is fixed by device type in VS1 and is shown below. The logical cylinder sizes assigned in VS1 result in a logical cylinder having a comparable amount of space regardless of the device type. The number of tracks allocated to spool space on a given direct access volume should be a multiple of the number of tracks per logical cylinder to avoid wasting tracks.

Device	Number of Physical Tracks in a Logical Cylinder	Logical Cylinder Capacity in Bytes
2314/2319	5	36,460
3330-series	3	39,090
2305-2	3	43,980

The DASD work area allocation routine maintains a logical cylinder bit map of the spool space defined. This map is contained in virtual storage. It indicates which logical cylinders are allocated and which are available. When a permanent I/O error occurs on any track in a logical cylinder, the logical cylinder is marked unavailable for allocation.

The DASD work area allocation routine allocates spool space such that the spool I/O load is balanced across the available spool volumes as much as possible. During processing, a count of the number of accesses and the total spool access time are maintained for each spool volume. When a spool space request is received, average access time is calculated for those spool volumes that have available space. The spool device chosen to satisfy a request is the one with available space and the smallest average access time.

One logical cylinder at a time is allocated to a given spool data set, and each time this space becomes filled, one additional logical cylinder is allocated. Because of the I/O load balancing approach used, the logical cylinders assigned to any given spool data set can be contained on more than one spool volume. The available logical cylinder allocated on the spool device selected is the one that is closest (on either side) to the current location of the access arm on the device. This is done to group allocated logical cylinders together so that access arm movement is minimized. Therefore, the logical cylinders allocated to a given spool data set on a given spool volume are not necessarily contiguous.

At system generation, a threshold value percentage for spool capacity can be specified or the default percentage of 80 can be used. The operator is notified when the threshold percentage of spool capacity becomes allocated during system operation. At this time, the operator should hold the input queue, ensure that writers are started to those SYSOUT classes that have data, and start another writer, if possible. When the operator indicates that these operations have been completed, the control program stops the operation of all active readers. The remaining spool space is then allocated only for starting another writer, processing jobs currently initiated, and terminating problem programs and system readers.

If the percentage of spool allocation continues to rise above the threshold value, the operator is informed of every five percent increase. When the spool volumes become so full that only a special reserve of logical cylinders is available, the operator is asked if the job currently requesting spool space should be canceled. Depending on the reply, the job is canceled or placed in a wait state until a logical cylinder becomes available. The reserve cylinders are allocated only for the purpose of starting another writer or canceling a job as a result of an affirmative operator reply to the cancel request.

The operator continues to be informed of the percentage of allocated logical cylinders until the percentage is reduced to the threshold value. When the allocation percentage decreases to a value of ten percent less than the threshold value, the operator is informed that spool space is no longer critical. JES readers that were stopped can be restarted and the input queue can be released.

The advantages of the spool techniques used by DASD work area management routines are:

- Spool space is allocated and deallocated more quickly via use of an in-storage map rather than by DADSM routines, which must process VTOC's to locate and return direct access space. OPEN and CLOSE processing is also eliminated.
- Spool space is allocated to minimize direct access device arm movement.
- Spool space for a given spool data set is allocated across I/O devices if possible to enable spool I/O operations to be overlapped and to help balance the I/O load.

- Space is allocated a logical cylinder at a time as required so there is less chance of wasting direct access space because of overestimating the requirement for a given spool data set.

- The operator is automatically informed that spool space is running out prior to a full condition that causes job cancellation. The operator can take steps to prevent a full spool condition.

JOB QUEUE MANAGEMENT

The job queue organization and management implemented in VS1 is a modified version of that used in MFT. It is designed to reduce contention for the job queue (SYS1.SYSJOBQE) and to eliminate duplicate job queue processing code through centralization of job queue processing, which also reduces paging activity.

The basic difference between job queue organization in MFT and VS1 is that much of the information for problem program jobs that is contained in the SYS1.SYSJOBQE data set in MFT is placed in other data sets in VS1. Specifically, SYS1.SYSJOBQE in VS1 does not contain system message blocks and the job scheduler control blocks (JFCB's, SCT's, SIOT's, etc.) created by the interpreter from the job control statements for problem program jobs (unless they have been initiated via a START command). System messages are placed in spool data sets on spool volumes, and scheduler control blocks are placed in new data sets called scheduler work area data sets (SWADS). The control blocks in the job queue are primarily job related, instead of job step related, and are not updated as frequently as scheduler control blocks. The job queue contains scheduler control blocks only for system tasks and generalized start jobs. Records in SYS1.SYSJOBQE are 176 bytes, as in MFT.

There is one SWADS for each active initiator. A SWADS is allocated and formatted when the initiator is started. Parameters in the SWADS DD statement in the initiator procedure are used. They can be overridden by the operator. The SWADS for a given initiator contains the scheduler control blocks for the job currently being handled by the initiator. The control blocks are placed in the SWADS by the interpreter at the time the job is selected for initiation and interpreted. Thereafter, a job scheduler routine accesses its SWADS to initiate and terminate job steps rather than the job queue, as is done in MFT, thereby eliminating much of the contention for the job queue. The scheduler control blocks for successive jobs processed by the same initiator overlay one another in the SWADS assigned to the initiator.

Scheduler work area data sets (up to 15) and SYS1.SYSJOBQE are maintained by job queue management routines which allocate and deallocate disk space in the job queue, enqueue and dequeue work entries, delete work queue entries, and read and write queue records. All system routines that access the job queue and the SWADS (JES readers and writers, job schedulers, master scheduler, etc.) do so via centralized job queue management. The job queue manager is contained in the pageable area of virtual storage.

SYS1.SYSJOBQE and SWADS can be contained on 2314/2319, 3330-series, 2305-2, or 2305-1 direct access storage. RPS is supported when present for the device. All SWADS must be allocated to the same device type (which can be different from the job queue device type) in order for automatic restart and system start to function correctly. If channel or device separation is requested for the SWADS, contention among initiators can be further reduced.

JOB SCHEDULER

The basic design changes embodied in the VS1 job scheduler are inclusion of the interpret function as part of job scheduling and access to a SWADS instead of the job queue for job step scheduling.

The components of the job scheduler (initiator, interpreter, allocation, terminator) have been modified to operate in a paging environment, interface with JES, support modifications to other system routines, and provide some functions not available in MFT. All scheduler components can operate paged in 64K of virtual storage and are structured to minimize the occurrence of page faults.

Initiator

A VS1 initiator is pageable and a large portion of it is reentrant. As in MFT, the initiator operates in a partition to perform its scheduling function. Initiators schedule problem program job steps (both pageable and nonpageable), system tasks, and JES readers and writers. Initiators interface with the job queue manager to access SYS1.SYSJOBQE and SWADS.

The VS1 initiator supports a queued problem program start facility which enables the operator to start more than one cataloged procedure to the same partition. The started procedures are queued and initiated on a first-in, first-out basis. System task starts must be single-step procedures while problem program starts may be multistep procedures.

The VS1 initiator also supports an operator option that is not provided in MFT. If all the data sets required by the job that is being initiated are not currently available, the operator can request that the job be placed in the hold queue. The operator can then release the job at a later time when the data sets become available. In MFT, the operator can only cancel the job or request another allocation attempt.

Interpreter

The interpreter is pageable and a large portion of it is reentrant. It operates as a subroutine of the initiator. The interpreter is invoked at the initiation of each job. The interpreter reads procedures directly from SYS1.PROCLIB (via data management) and interfaces with JECS spool management to read all the JCL and any PROC spool data sets associated with the job to be scheduled. It interprets all the job control for the job, constructs the required scheduler control blocks, and writes them in the SWADS (using job queue management) for use by the other job scheduler components. Interpreter messages are placed in system message spool data sets. Commands are sent to the master scheduler for processing when they are encountered. Jobs that were being interpreted when abnormal system termination occurred do not have to be resubmitted during the warm start procedure, as they do in MFT. The interpreter accepts all job control statements supported in MFT, as well as the following new parameters:

- ADDRSPC=VIRT or REAL and the REGION parameter on JOB and EXEC statements (discussed in Section 90:10)
- TYPRUN=SCAN on a JOB statement to indicate that the job control for the job is to be analyzed for errors but that the job is not to be executed
- COPIES=nnn on SYSOUT DD statements to request multiple copies of system output data sets

- DLM=cc on SYSIN data set DD statements (DD* and DD DATA). This parameter can be used to specify a delimiter other than /* or // to indicate the end of job step data in the input stream.

- DEST and HOLD parameters on SYSOUT DD statements submitted via RES (See Remote Entry Services discussion)

UNIT and SPACE parameters on a SYSOUT DD statement are ignored as they are no longer required. New DD statement parameters for VSAM have been added to the job control language as well (discussed in Section 90:30).

Allocation

The allocation routine operates as a subroutine of the initiator to allocate and deallocate I/O devices to job steps, issue mounting messages to the operator, etc., as in MFT. In addition, the allocation routine supports dedicated work data sets (supported in MVT but not in MFT). This facility enables a job step to use disk data sets that are assigned to the initiator that schedules their execution. Job scheduling time is reduced by the elimination of temporary disk data set allocation and deallocation processing.

Terminator

The terminator is pageable and a large portion of it is reentrant. Like the initiator, the terminator places messages in system message spool data sets. No other functions different from those of MFT are supported by a VS1 terminator (except those related to supporting a paging environment).

Direct SYSOUT (DSO) Writers

The same functions are supported by DSO writers in VS1 as in MFT, except that a VS1 DSO writer can handle up to eight job classes instead of three.

System Management Facilities (SMF)

SMF provides all the same functions it does in MFT and is expanded to include new accounting data provided by JES and page management. The SMF option desired is chosen at system generation from the following:

- NOTSUPPLIED - no SMF data is provided. However, the OUTLIM facility is still available (without the OUTLIM exit which is supported only if SMF is present).

- BASIC - user-written accounting routines are to be provided. These routines can be newly written or those currently being used with MFT. The latter need not be modified for operation in VS1. The new JES accounting information is made available as are two user exits not provided in MFT.

- FULL - SMF routines are to be included. This option should be selected if SMF is currently being used in MFT. The same options are supported as in MFT. New accounting data and two new exits that are not available in MFT are provided also.

SMF records can be written only on direct access volumes in VS1. They cannot be written on tape, as in MFT. The SMF record types and formats produced by SMF routines in VS1 are compatible with those

produced in MFT, for the most part. Additional accounting information is supplied, minor changes to existing fields have been made, and certain fields have a different meaning in VS1. For example, in the job step termination record the storage-requested and storage-used fields reflect the virtual storage used. If the job step ran in nonpaged mode, these fields also reflect the real storage used. SMF records that are modified in VS1 are the system measurement record (Type 1), the step termination record (Type 4), and the end of day record (Type 12).

The additional job accounting information provided by JES at job purge time is the following:

- Time required to read the job (elasped time calculated from JES reader start and stop times)
- Number of cards read
- Job priority and job class
- Elapsed time for SYSOUT print processing and number of lines printed
- Elapsed time for SYSOUT punch processing and number of cards punched
- Elapsed time for SYSOUT tape processing and number of SYSOUT records written to tape

The page supervisor provides the following new data to SMF:

- Number of page-ins per job step (including user and system page-ins) and number of page-ins for the entire system (reclaimed pages are not included in this count)
- Number of page-outs per job step (including user and system page-outs) and number of page-outs for the entire system
- Number of reclaimed pages for the entire system

General Flow of Job Scheduling

Figure 90.20.2 illustrates JES and job scheduling flow in VS1. Active JES readers read input streams. Data read (job control statements, procedures, input data, commands) is passed to spool management without interpretation of job control statements. Spool management requests the allocation of spool space (one logical cylinder is allocated per spool data set) and blocks input stream data which is written in spool data sets by the DASD work area manager. Whenever a complete job has been read, the JES reader creates control blocks to describe the job and passes them to the job queue manager. Queue space is allocated for the job, and the job is enqueued in SYS1.SYSJOBQE by priority within job class or placed in the hold queue. The JES reader continues reading its input stream.

Started initiators inspect the job queue for a job with a job class they are assigned to handle. When an initiator selects a job, it passes control to the interpreter routine. The interpreter obtains the job control statements for the job from the appropriate JCL spool data set via spool management. Job control statements are interpreted and scheduler control blocks are built for the job which are placed (by job queue management) in the SWADS associated with the initiator. Spool space for SYSOUT spool data sets is allocated. One logical cylinder is allocated to each SYSOUT spool data set at this time. Commands within the job are routed to the master scheduler and interpreter messages are placed in a system message spool data set. Control is given to the allocation routine which attempts to allocate I/O devices to the first

step. Allocation messages are written in a system message spool data set and the job step is begun.

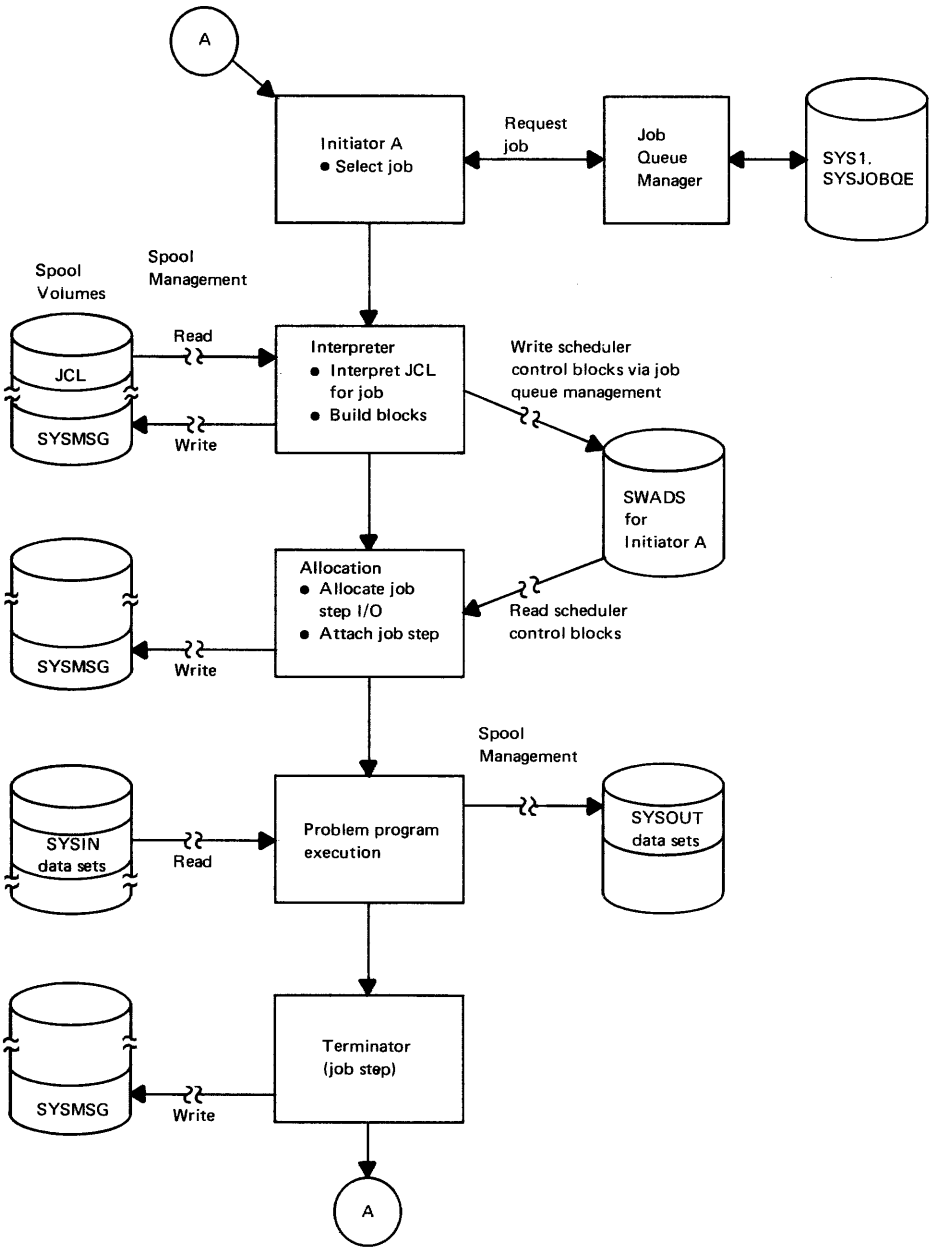


Figure 90.20.2. General flow of JES and job scheduling in OS/VS1

During job step execution, the problem program obtains SYSIN data using QSAM or BSAM, which interfaces with spool management via the JES translator to read the appropriate SYSIN spool data sets. The problem program can write SYSOUT spool data sets in the same way. When the job step completes, the terminator performs I/O device deallocation and places messages in a system message spool data set.

Initiation of successive steps of the job continues until end of job occurs. The job terminator requests that the job queue manager enqueue the SYSOUT spool data sets for the job in the job queue by priority

within SYSOUT class. The spool space allocated for SYSIN spool data sets for the job is released. The initiator attempts to select another job.

Started JES writers interface with job queue management to select a SYSOUT spool data set with a class they are assigned to handle. A JES writer obtains SYSOUT logical records via spool management. When all the SYSOUT spool data sets for a job are processed, the job queue manager purges the job from the system and spool management frees the SYSOUT spool space allocated to the job.

REMOTE ENTRY SERVICES

Remote entry services (RES) is a fully integrated functional extension of JES. Using binary synchronous communications, RES enables remote users to submit jobs to a central computing system via a work station (terminal) and to receive the output from these jobs. RES presents remotely submitted jobs to JES readers, which in turn place the jobs in the system input queue. JES writers present output from completed remotely submitted jobs to RES, which transmits the output to remote work stations. Hence, the unit record devices at remote work stations (readers, printers, punches) are logically operated by JES as if they were part of the central system.

The following terminals are the remote work stations supported by RES:

- 1130 System
- System/3
- System/360 Models 20 to 195
- System/370 Models 135 to 168 (operating in BC mode only)
- System/370 Model 195
- 2770 Data Communications System
- 2780 Data Transmission Terminal

The maximum number of terminals supported is equal to the maximum number of tasks supported in the VS1 control program. USASCII code is supported only for the 2770 and the 2780. Terminals can be attached to point-to-point leased and point-to-point dial-up lines. Multidropped leased lines and dial-up lines are not supported. Lines can be two-wire half-duplex or four-wire half-duplex. Full duplex lines are not supported. A 2701 or 2703 is required in the central system configuration. A Model 135 can use the integrated communications adapter (ICA) instead of a 2701.

Intelligent remote work stations (those that include a CPU) operate under the control of standalone work station programs. Work station programs are distributed with the VS1 control program. Generation procedures must be performed to tailor a work station program to the configuration of each intelligent work station to be supported by RES. The standalone programs provided for RES remote work stations are the same ones that are provided for HASP II RJE remote work stations.

RES enables a user at a remote work station to:

- Transmit jobs to a central system for processing, using standard VS1 job control statements and operator commands. Two additional job control parameters and five additional operator commands are provided also.

- Route the output from each completed job to the central system or a specific remote work station, which need not be the one from which it was submitted. Output destination can be indicated in job control statements or via a new RES command.

- Display the status of his work station and the status of his submitted jobs
- Send messages to and receive messages from other remote work station users and the central system operator

RES functions are provided by the following components of VS1:

- The remote terminal access method (RTAM), which handles all data transmission to and from remote work stations
- New data sets, SYS1.UADS and SYS1.BRODCAST, which define the attributes of RES work station users and contain work station messages
- New commands (LOGON, LOGOFF, ROUTE, SEND, LISTBC), which enable remote users and the central operator to control remote job processing and to communicate with each other. WTO and WTOR macros are extended to allow system tasks to support remote users.
- JES readers and JES writers that interface with RTAM and support remote input and output streams. (Column binary is not supported.)

Figure 90.20.3 shows how RES components interact with each other and interface with JES to support high-speed remote job entry.

RTAM

If RES is to be used, this fact must be indicated during system generation. The RTAM module required to support the RES terminal configuration must be generated via a separate RTAM generation procedure that can be performed any time after STAGE I of the VS1 generation procedure is completed.

RTAM is the only access method used by RES. RTAM executes in the pageable supervisor area in VS1. It supports only the RES terminal network and does not interface with any other terminal networks (TCAM or BTAM, for example) that may be included in the system configuration. RTAM directly controls work stations without CPU's (2770 and 2780 terminals) and interfaces with intelligent work stations using a MULTI-LEAVING technique that is not supported by OS RJE.

MULTI-LEAVING is a more efficient way to transmit data between two computers using binary synchronous communication facilities because it reduces the number of line turnarounds required and enables more data to be transmitted before line turnaround occurs. MULTI-LEAVING permits data from more than one unit record device in a work station configuration to be sent during a single transmission, and allows transmission acknowledgment to be sent together with data in the same record. For example, a work station can transmit a record containing text from two or more input stream unit record devices to the central system. The central system can respond by sending a record that acknowledges receipt of the text and that includes text for one or more output stream unit record devices in the work station configuration. This eliminates an individual transmission for each text record and each acknowledgment.

Multitasking support must be included in RTAM when intelligent work stations are part of the RES terminal configuration. MULTI-LEAVING support is a standard feature of the standalone programs provided for intelligent work stations.

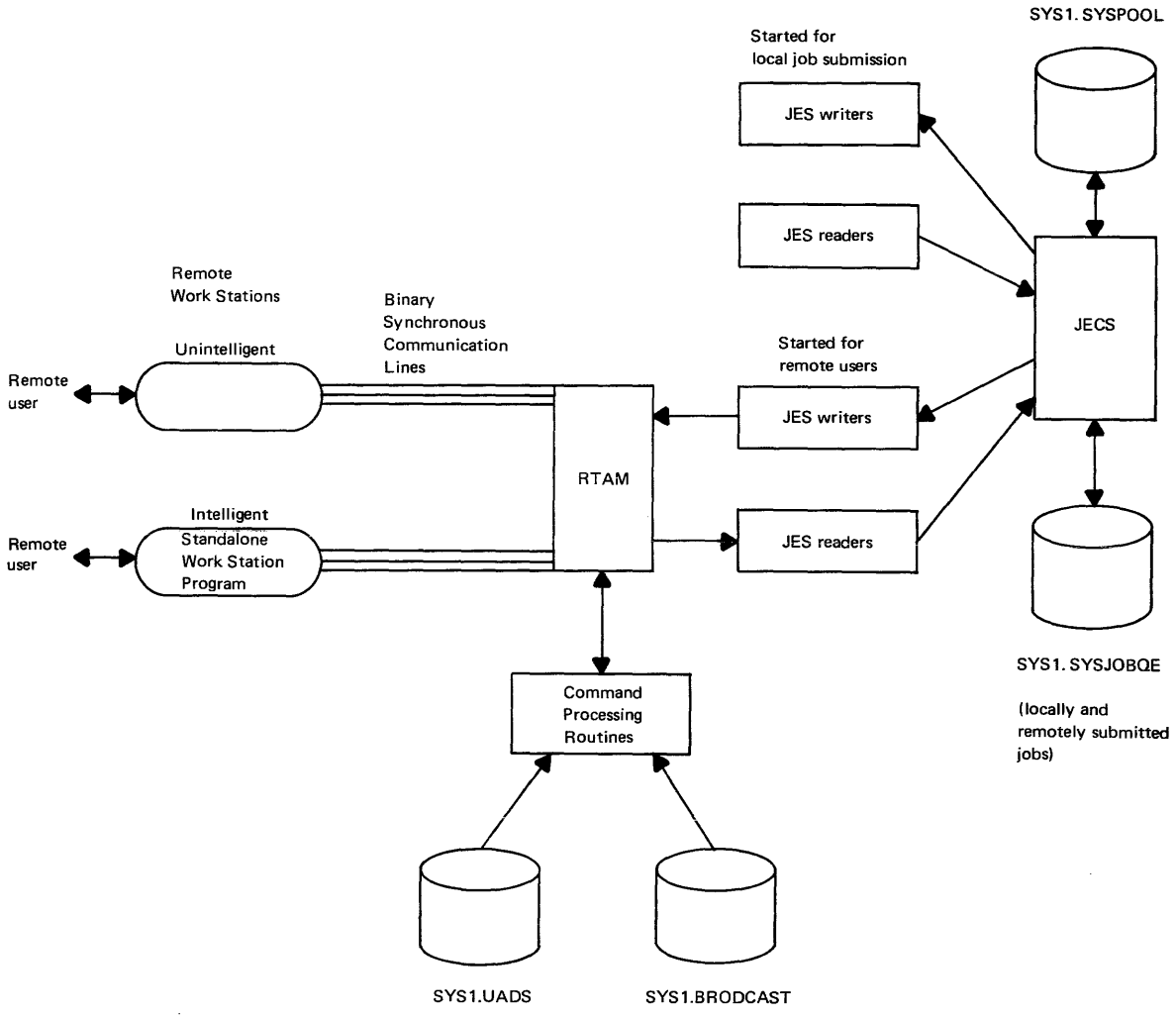


Figure 90.20.3. RES interface with JES

RES Data Sets

The user attributes data set, SYS1.UADS, is a new partitioned data set that is required when RES is included in the VS1 control program. It is used to control remote user access to the central system via RES. SYS1.UADS contains one or more members for each user authorized to access the central system. Members contain control information, identifying attributes, and SMF accounting data for each user (such as user-id, passwords, account numbers, and log-on procedures).

When a remote user attempts to log on the system, SYS1.UADS is accessed if verification of the password or the log-on procedure is required. If an entry does not exist for the user, or if an invalid parameter is supplied (such as user-id or password), the log-on request is denied. The SYS1.UADS data set must be created and maintained using the IBM-supplied ACCOUNT utility which provides the capability of adding or deleting users, changing fields in existing user members, and listing user members.

The SYS1.BRODCAST data set is also required when RES is used. It is used to hold messages that have been issued by remote users and the central operator using the SEND command, and that have not yet been sent.

SYS1.BRODCAST is divided into a notices section and a mail section. The notices section contains messages that are available to all remote users and the central operator. The mail section contains messages issued only for specific users. The mail section can be accessed by the central operator as well.

RES Commands

Five new commands are included in the set of VS1 operator commands, and new parameters have been added to some existing commands to control RES functions. Table 90.20.1 lists VS1 commands that can be issued by a remote work station user in addition to the new commands for RES. Commands not shown are rejected as invalid with a diagnostic message if issued from a remote work station. A remote user can issue commands to control only his jobs and output data. The central operator can use all VS1 operator commands and has access to all remotely submitted jobs.

Table 90.20.1. OS/VS1 operator commands that can be issued from an RES remote work station

CANCEL	LOG	RELEASE	STOP
DISPLAY	MODIFY	REPLY	STOPMN
HOLD	MONITOR	RESET	WRITER

A user at a remote work station issues a LOGON command to establish connection with the central system. During the log-on procedure, user identification, terminal identification, and password (if any) are verified using information contained in SYS1.UADS. The JES readers and writers identified in the log-on procedure specified in the LOGON command are started. (A remote user cannot start JES readers and writers directly but the central operator can.) If they were requested in the LOGON command, mail and notices contained in SYS1.BRODCAST are retrieved and sent to the user. If the LOGON command is valid, the user can begin transmitting jobs and messages.

The LOGOFF command is issued by a remote user to indicate that communication with the central system is finished. This command can be entered via an input stream between two jobs. The central operator can also issue the LOGOFF command to terminate operations at a remote work station. Log-off processing includes the stopping of readers and writers started for the remote user.

The ROUTE command is provided to allow a remote user to route SYSOUT data sets associated with his jobs to a particular work station or to a queue (class or hold queue) different from the one they are presently in. This command can be used to override the destination indicated in the DEST parameter on the SYSOUT DD statement submitted, or to specify the destination if the SYSOUT DD statement has a HOLD parameter to indicate the job is to be held until a ROUTE command is issued. The ROUTE command can be issued any time after the job is submitted and before the SYSOUT data set to which it refers is processed.

Messages can be transmitted among remote users, and between the operator and remote users via the SEND command. A given message can be sent to one or more remote users or to all of them. The central operator can request that a message be saved in the SYS1.BRODCAST data set instead of being transmitted immediately. The SEND command can also

be used to delete previously saved messages from SYS1.BRODCAST. The LISTBC command enables a remote user to list all notices and only mail belonging to him. The central operator can list mail belonging to any user as well as all notices.

RES Initialization

During system initialization, the RTAM module is brought into the pageable supervisor area after JES routines have been loaded. The user-specified maximum number of readers and writers that JES can support must include those required for RES work stations. In order to initiate RES processing, the central operator must enter a command to start RTAM. This causes RTAM to be activated and the lines indicated in the RTAM procedure are enabled. The central operator can issue the MODIFY command any time thereafter to enable additional lines, disable enabled lines, or reenable lines.

As part of the RTAM initialization procedure, work stations identified as belonging to the permanent log-on group are logged on. The optional permanent log-on facility allows nonintelligent terminals, such as 2770 and 2780 terminals that are connected to leased lines, to be automatically logged on whenever RTAM is started. These terminals remain logged on until a LOGOFF is issued for them.

Once RTAM is started, any terminals attached to the enabled lines that are not part of the permanent log-on group must be connected to the system via execution of the log-on procedure. During this procedure, JES readers and writers are started for the remote terminal, as indicated in the log-on procedure specified. Once operations at a work station connected to a switched line are finished, the user can log off and, thereby, release the line and make it available for use by another remote work station connected to it.

Virtual Storage Requirements

Assuming one terminal on one line, one reader, one punch, and one printer, the minimum virtual storage requirement for RES (RTAM and JES readers and writers) is approximately 18K without MULTI-LEAVING and 20K with MULTI-LEAVING. Another 6K is required, whether or not MULTI-LEAVING is included, for each additional line supported.

RES Advantages Over RJE

As a replacement for RJE, RES supports facilities equivalent to those of RJE and offers the following advantages:

- RES is fully integrated within VS1 and uses the normal system job scheduling facilities (such as JES readers and writers, scheduling routines, and the job queue manager). Only the RTAM module is optional.
- RES is designed to operate in a paging environment and can provide better performance than RJE partly because of MULTI-LEAVING support.
- RES requires less real storage for its operation.
- RES SYSOUT classes are independent of the central SYSOUT classes and more flexible SYSOUT routing is provided by RES (a unique set of queues is associated with each remote RES user that represents the SYSOUT data sets only for that remote user).

- The RES remote user command language is a compatible subset of the OS/VS1 central operator command language (the job entry control language of RJE is not). Jobs can be submitted locally or remotely using the same job control statements and commands.

- System/3 is supported as a remote work station.

CONVERSATIONAL REMOTE JOB ENTRY

The facilities offered by CRJE are the same in MFT and VS1. CRJE can operate in paged mode in a minimum partition of 128K. However, a minimum of 60K of real storage is fixed by CRJE. Therefore, a system with more than 160K of real storage is required to operate CRJE.

90:25 TASK MANAGEMENT

VS1 task management routines have been modified as required to operate in a paging environment, interface with other modified control program routines, and support EC instead of BC mode of system operation (different PSW format, interruption codes in permanently assigned locations above 127, for example).

VS1 and MFT task management routines are functionally identical for the most part. There are no functional changes to the contents supervisor, the overlay supervisor, the timer supervisor, or checkpoint/restart and warm start routines. (Note that checkpoint records are always 2K in size in VS1, and that MSGLEVEL=1 is no longer a required parameter on the JOB statement.) The timer supervisor supports timing facilities equivalent to those of MFT using the interval timer at location 80 and the time of day clock. It does not support the CPU timer and the clock comparator. The following identifies the significant functional differences between VS1 and MFT task management routines.

INTERRUPTION SUPERVISOR

Interruption handling is essentially the same in VS1 and MFT; however, a few additional interruptions are recognized and the SVC transient area is 2K instead of 1K. Specifically, segment and page translation exception, translation specification exception, monitor call, program event recording, and SET SYSTEM MASK (SSM) instruction interruptions are handled.

The SPIE facility has been expanded to allow the user to gain control after a segment translation exception (invalid bit is on in the addressed segment table entry), which is treated as an addressing exception. Authorized routines can gain control after a page fault (page translation exception caused by invalid bit on in the page table entry for the referenced virtual storage page), which normally is handled by page management. An authorized routine gains control after both disabled and enabled page faults and the system lock (described under "Task Supervisor") is not turned on. Therefore, the SPIE facility should be used carefully. A routine is considered authorized if it has the characteristics of a subsystem as defined in VS1: operates in supervisor mode with protect key 0 and is identified as a subsystem in the required control block. The data presented to a user-written SPIE routine has the same format in VS1 as in MFT so that SPIE routines that operate in BC mode will operate in EC mode without modification.

MONITOR CALL instructions are contained in various portions of the control program in order to alert the control program to the occurrence of certain events. For example, IOS uses the monitoring facility to

collect statistics about paging operations that are presented to SMF and to monitor the I/O events requested via the generalized trace facility (GTF). When appropriate, GTF is given control after a monitor call interruption occurs. When program event recording is operative, the dynamic support system (DSS) is entered after a PER interruption. (GTF and DSS are discussed in Section 90:40.)

The interruption supervisor also recognizes an SSM special operation exception that occurs when an SSM instruction is executed. Control is given to a routine that analyzes the masking requests indicated, which are assumed to be in BC mode format. This routine then puts the system in the requested state. (The new supervisor lock, described below, is tested prior to altering the system mask, if necessary.) A new supervisor macro, MODESET, is implemented that is designed to be used in VS1 in place of the SSM instruction. MODESET can be used to request a system mask setting, storage protect key alteration, and the setting of problem program or supervisor state in the PSW.

TASK SUPERVISOR

Two lock fields are implemented in the task supervisor to ensure proper system operation when a disabled page fault occurs. In VS1, a page fault can occur during the execution of a routine that has disabled the CPU for interruptions (I/O and/or external). This is called a disabled page fault. A routine normally operates with the CPU disabled for interruptions because it is not reentrant and, therefore, should not be reentered before its completion, or because it modifies or references a serially reusable resource. The processing of a page fault, which requires I/O interruptions to be enabled to allow the I/O interruption for a completed page-in operation to be presented, can allow code that operates with the CPU disabled to be reentered, with improper processing the result.

To prevent this situation, two lock fields are implemented in VS1, a system lock and a supervisor lock, which can be set on (locked) or off (unlocked). When a disabled page fault occurs in an executing task, the appropriate lock, system or supervisor, as indicated by the task, is turned on. When the system lock is on, no task can be dispatched except one that is related to paging operations. When the supervisor lock is on, ready tasks that are to operate with the CPU enabled for interruptions can be dispatched but no code that operates with the CPU disabled can be executed except that which is related to paging and task dispatching operations. The lock used remains on until the disabled page fault is resolved. Code is included within the control program to recognize an attempt made by code to enter the disabled state by executing an SSM instruction or a MODESET macro, and to place such a task in the wait state, when necessary (the appropriate lock is turned on).

Certain resident control program routines (IOS, page supervisor, task dispatching routines, for example) are structured to avoid disabled page faults in VS1. User-written Type 1 and Type 2 SVC's that are to be added to a VS1 control program should avoid disabled page faults, if possible. If disabled page faults are incurred by a user-written Type 1 or 2 SVC routine, the system lock will be used.

The lock approach implemented in VS1 has the advantage of allowing routines to encounter disabled page faults when necessary, in order to avoid fixing a large number of pages, and when the supervisor lock is used, the approach taken also avoids delaying total system operation while a disabled page fault condition is handled.

VIRTUAL STORAGE SUPERVISOR

The virtual storage supervisor is responsible for allocating and deallocating virtual storage in response to user (GETMAIN and FREEMAIN) and system requests for storage. Except for V=R requests, real storage is not assigned to allocated virtual storage until the virtual storage is referenced during processing. When a FREEMAIN is issued for the last allocated area in a virtual storage page, the appropriate page table entry user bit is turned off. If a page frame is allocated to that virtual storage page, it is released and made available for reassignment. The virtual storage supervisor is functionally equivalent to the main storage supervisor in MFT except for the following modifications:

- Support of dynamically expandable SQA instead of a fixed SQA to minimize system terminations because of the lack of SQA space
- Implementation of a protected queue area (PQA) in problem program partitions to enhance system integrity
- Allocation of virtual storage to minimize or eliminate page faults during virtual storage supervisor execution. For example, the control blocks that describe the virtual storage available in a problem program partition are contained in the fixed PQA.
- Expansion of the GETMAIN macro to request allocation of virtual storage on a page boundary and to specify a subpool number. The FREEMAIN macro in VS1 can specify the number of the subpool to be freed.

PROGRAM FETCH

Load modules in VS1 have a starting virtual storage address of zero and are stored in partitioned data sets in the same format that is used in MFT. Hence, when a load module is fetched in VS1, it must be relocated to the beginning address of the virtual storage area to which it is assigned and virtual storage address constants must be modified, just as in MFT.

The standard program fetch routine in VS1 is identical to its MFT counterpart, and it uses the channel program translation and page fixing facilities of IOS. Load module record loading and virtual storage address constant relocation are performed serially (the text record is read in and then the address constants are modified), as in MFT.

The optional PCI fetch routine is modified for operation in a paging environment. PCI fetch does not use the channel program translation facility of IOS. It uses the new EXCPVR macro (discussed in Section 90:30) instead of EXCP. In VS1, PCI fetch requests the allocation and fixing of up to six page frames (12K) for the execution of each read operation (SIO). Text records are read into these page frames. During execution of the CCW chain, PCI chaining is suppressed if it is determined that execution of the next CCW list with a text CCW will cause the fixed real storage area associated with the I/O operation to be exceeded. The channel program then terminates and the page frames used during the operation are unfixed. PCI fetch performs address constant relocation during read operations (adds the relocation factor to virtual storage address constants contained in text records), just as in MFT.

When a program is loaded by PCI fetch, its pages are not automatically written on external page storage as part of the program loading procedure. Page-outs of one or more pages of a program that is being loaded (or that is loaded) occur for the first time when the real

storage any pages of the program occupy is required for allocation to other pages, and the page supervisor considers these pages to be eligible for replacement as per its page replacement algorithm. The change and reference bits for each page frame that contains program text are on as a result of the I/O operation that read in the text. Hence, before the page frames allocated to a program that is being loaded (or to a recently loaded program) can be reassigned, a page-out will be performed. The fact that the change bit is turned on by the fetch operation is what causes the first and only page-out of pages that do not modify themselves.

90:30 DATA MANAGEMENT

Data management components are altered where necessary to operate in a paging environment and to interface with JES and the modified VS1 input/output supervisor (IOS). The significant functional differences between data management in VS1 and MFT are changes in IOS to handle channel program translation and page fixing, and the availability of a new access method called virtual storage access method (VSAM).

OPEN, CLOSE, EOV, and DADSM routines for VS1 and MFT are functionally equivalent. VS1 supports all the access methods provided in MFT except QTAM. The same functions the access methods provide in MFT are also supported in VS1. Programs that use these access methods can be executed in VS1 either in paged or nonpaged mode with one exception. A program that is to use the chained scheduling facility of QSAM or BSAM must execute in nonpaged mode. If a job step with chained scheduling specified is initiated to execute in paged mode, regular scheduling is automatically substituted.

All the VS1 access methods except TCAM and VSAM interface with IOS via the EXCP macro and, therefore, use the channel program translation and page fixing facilities of IOS. TCAM can operate in a pageable partition but requires certain of its message control program elements (such as control blocks and the buffer pool) to be long-term fixed in real storage during the entire time TCAM is in operation. TCAM interfaces with IOS via the new EXCPVR macro and performs its own channel program translation. TCAM does not require long-term fixing of any portion of the message processing programs that it services.

For performance reasons, certain access methods have also been modified to reduce the total amount of code they contain that operates with the CPU disabled for interruptions or to prevent page faults in any such code. ISAM requires an additional 2K of virtual storage because of the inclusion of new required I/O appendages.

The access methods do not support a parameter that can be used to cause buffers to be aligned on page boundaries when buffers are allocated by the access method. If an Assembler Language programmer wishes to have buffers aligned on a page boundary and/or ensure that buffers are packaged such that they do not cross page boundaries, buffers must be defined and aligned by the programmer.

INPUT/OUTPUT SUPERVISOR

In VS1, IOS has the following additional functions:

- Translation of the virtual storage addresses contained in CCW lists. The CCW translation routine performs this function prior to the issuing of the SIO instruction for each I/O operation requested by a pageable routine via the EXCP macro. A new CCW list with translated addresses is built in SQA. This new list is used for the actual I/O operation. A CCW list with up to 240 CCW's can be translated.

- Construction of indirect data address lists (IDAL's), when necessary. If the buffer specified in a CCW crosses a virtual storage page boundary or if the buffer is larger than 2K, the appropriate IDAL's consisting of indirect data address words (IDAW's) are constructed in SQA also. (Checking to determine whether a buffer that crosses a virtual page boundary is assigned contiguous page frames is not performed.)

- Short-term fixing of the pages associated with an I/O operation to prevent the occurrence of page faults during I/O operations. Each time an I/O request (EXCP) is received, IOS ensures that pages it will reference to service the I/O request are short-term fixed. This includes pages that contain control blocks (IOB, DCB, DEB, ECB/DECB, and AVT), required IOS appendage routines, and buffers.

- Translation of the real storage address in the channel status word to a virtual storage address at the completion of the I/O operation. In addition, pages that were short-term fixed prior to the I/O operation are unfixed.

The same five I/O appendage interfaces that are provided in MFT are supported in VS1 and one new appendage interface is defined. There also are new returns from the SIO and the PCI appendages. The new page fix appendage is actually part of the SIO appendage and it is entered using a new entry point into this appendage. The page fix appendage is provided to enable an EXCP user to request short-term fixing of up to seven different virtual storage areas that will be referenced during an EXCP request but that are not automatically fixed by IOS. A user-written EXCP program with user-written I/O appendages that can incur page faults can use the new appendage to short-term fix the areas referenced by the I/O appendages. The new PGFX parameter for the EXCP data control block (DCB) is provided to indicate that the page fix appendage is to be used.

In addition to the EXCP macro, VS1 IOS supports a new macro, EXCPVR, that can be used to request an I/O operation. This macro can be issued only by the page supervisor or by subsystem routines, such as JES components and TCAM. A routine is identified as a subsystem via a bit in the TCB or the JSCB. A problem program can use the EXCPVR macro if it identifies itself as a subroutine or if the appropriate bit in the data extent block (DEB) for the data set is turned on by the user. When IOS receives an EXCPVR macro, it does not perform channel program translation, page fixing, or validity checking. It is assumed that, where necessary, these functions have been performed by the requester prior to issuing the EXCPVR macro.

When the EXCPVR macro is used instead of EXCP, the time required for IOS to initiate an I/O operation is reduced. The EXCPVR macro should be used carefully, however, because the I/O supervisor does not perform any of the storage protection functions it provides when the EXCP macro is issued (checking to determine whether all the control blocks, buffers, etc., associated with the I/O request belong to the requesting task). Hence, a task that uses EXCPVR could inadvertently store information outside its partition and impair the integrity of the system.

VIRTUAL STORAGE ACCESS METHOD

General Description

Virtual Storage Access Method (VSAM) is a new component of OS data management that is supported in VS1 and VS2. VSAM provides a data set organization and access method for direct access devices that is different from existing OS data set organizations and access methods for

direct access devices (SAM, ISAM, DAM, PAM). In a VS1 environment, VSAM supports 2314/2319, 3330-series, and 2305 (Models 1 and 2) devices and uses rotational position sensing when the feature is present.

VSAM for VS1 and VS2 uses System/370 instructions and is designed to operate efficiently in a paging environment. VSAM uses the EXCPVR macro for I/O requests. Hence, like VS1 and VS2, VSAM can operate only on System/370 models with dynamic address translation hardware and cannot run on System/360 models.

A subset of OS/VS VSAM is supported by DOS/VS. The VSAM Assembler Language macros used in OS/VS and DOS/VS are compatible, except for OPEN and CLOSE. In addition, a VSAM file contained on a DOS volume can be processed by OS (VS1 or VS2) programs. Similarly, a VSAM data set contained on an OS volume can be processed by DOS/VS programs as long as facilities are not used that DOS/VS VSAM does not support. This compatibility enables VSAM data sets or files to be processed by both OS/VS and DOS/VS, and aids in the transition from DOS/VS to OS/VS1 or OS/VS2.

VSAM supports both sequential and direct processing and is designed to supersede ISAM, although the two access methods can coexist in the same operating system. VSAM supports functions equivalent to those of ISAM and offers new features. VSAM also can provide better performance than ISAM, particularly when the number or level of additions in the data set is high. The new structure and features of VSAM make it more suited than ISAM to data base and online environments.

VSAM support consists of the following:

- Access method routines with which the user interfaces to process logical records in VSAM data sets. These routines are reentrant.
- VSAM catalog/DADSM routines that manage direct access volumes and space used by VSAM data sets and catalogs. VSAM data sets are cataloged in the new required VSAM catalog.
- The access method services multifunction service program, which provides required VSAM services, such as data set creation, reorganization, and printing, and VSAM catalog maintenance.
- ISAM interface routine that enables the transition from ISAM to VSAM to be made with little or no modification of ISAM programs. This routine is reentrant.

Data Set Organizations

VSAM supports two different data set organizations, key-sequenced organization and entry-sequenced organization, both of which allow sequential and direct processing, record addition without data set rewrite, and record deletion. Key-sequenced organization is logically comparable to ISAM organization in that logical records, either fixed or variable in length, are placed in the data set in ascending collating sequence by a key field value. Records added after the data set is created are inserted in sequence and existing logical records are moved when necessary. In VSAM organization, as in ISAM, each logical record in a key-sequenced data set must have an embedded, fixed-length key located in the same position within each logical record. A key-sequenced data set also has an index containing key values. The entire index is used to process records directly and a portion is used to process records sequentially.

An entry-sequenced VSAM data set, which has no ISAM counterpart, contains records sequenced in the order in which they were submitted for

inclusion in the data set. Records added to an existing entry-sequenced data set are placed at the end of the data set after the last record. Therefore, records are sequenced by their time of arrival rather than by any field in the logical record. In addition, there is no index for an entry-sequenced data set.

Key-Sequenced Data Set Organization

The physical structure of a key-sequenced VSAM data set is very different from that of an ISAM data set. The index and the logical records in key-sequenced organization are two distinct data sets with separate data set names, although a portion of the index may be placed within the logical record data set area to improve performance. A key-sequenced data set does not have a separate additions (overflow) area, as can be defined for an ISAM data set, and additions to a key-sequenced data set are always blocked.

Like an ISAM data set, a key-sequenced VSAM data set can be multi-extent and multivolume. Secondary space allocation can be specified when the key-sequenced data set is defined so that the data set can be extended when logical records are added, if necessary. (This facility is not supported in ISAM.) All extents of logical records must reside on direct access volumes of the same type, and a data set can consist of a maximum of 255 extents. The index data set, however, can be placed on a device type that is different from that of the logical record data set. Unlike ISAM data set volumes, all volumes of a key-sequenced data set that contain logical records need not always be mounted at OPEN time. VSAM data sets can be placed on disk volumes that contain data sets with other organizations.

Each extent of a key-sequenced data set that contains logical records is divided into a number of control areas. Each control area contains a number of control intervals that are on contiguous direct access tracks. A control interval is composed of one or more fixed-length physical disk records. Unlike physical records in an ISAM data set, the physical records in a key-sequenced data set can be 512, 1024, 2048, or 4096 bytes in size only, and they are written without a key (count and data disk record format). The access method chooses the physical record size based on the user-specified buffer size and the device characteristics. When buffer size is large enough, the physical record size chosen is that which makes best use of the track capacity of the direct access device used. A control interval can be a maximum of 65,536 (64K) bytes in size.

A control interval contains logical records in ascending key sequence, free space, and system control information about the logical records and free space, in that sequence. Logical records must have unique keys. A logical record and its control information (record definition field), although not contiguous within a control interval, are called a stored record. A logical record can span physical records within a control interval, but it cannot span control intervals. A complete control interval is the unit of data transfer between the VSAM data set and real storage. Hence, command-chained reads/writes are used when a control interval contains more than one physical disk record.

Figure 90.30.1 shows an example of a control area that consists of three control intervals. There are three physical records in each control interval. The number of control intervals in a control area is determined by the access method and is optimized, based on direct access device and index characteristics. The maximum size of a control area on disk is one cylinder, and a control area contains an integral number of control intervals. The size of a control interval can be specified by the user and is used as long as it is within the limits defined by VSAM; otherwise, a user-specified control interval size is ignored.

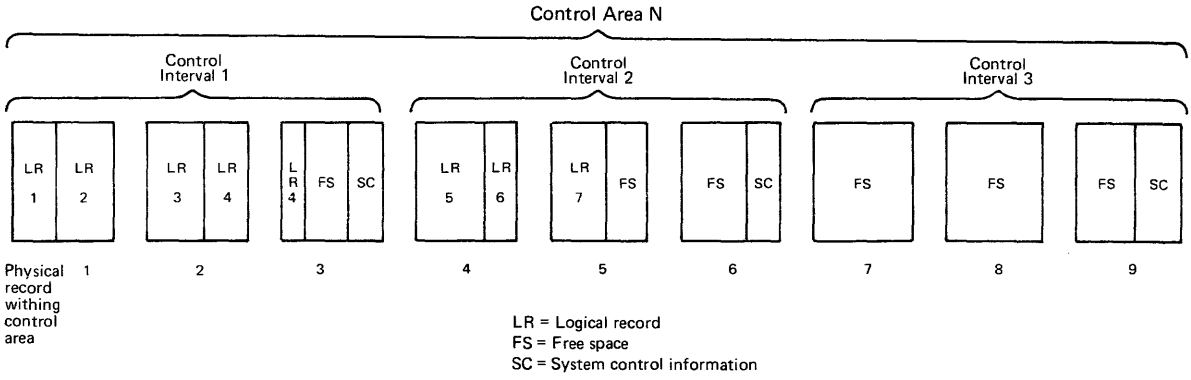


Figure 90.30.1. Organization of a control area for a VSAM key sequenced data set

A key-sequenced data set is divided into control areas and control intervals in order to distribute free space throughout the data set for the addition of logical records. When a key-sequenced data set is defined, the user can specify the percentage of unused control intervals that are to be left in each control area, and the percentage of free space to be left at the end of each control interval. For example, if 30 percent free control intervals in control areas and 20 percent free space in control intervals are specified, 70 percent of the total number of control intervals in each control area will be used for data when the data set is created. Each of the control intervals actually used for data will be 80 percent filled at load time. The unused space in control intervals and the unused control intervals in each control area are available for making additions.

The use of control intervals also reduces the amount of record processing that must be done to add a record and to retrieve an addition compared to what can be required in ISAM, since there are no overflow chains in VSAM organization. When a record must be added to a control interval, records are shifted to the right within the control interval to make room for the new record (if the record does not belong at the end of the control interval). As long as there is enough free space in the control interval, no other control interval is involved in the addition process.

If a control interval does not contain enough space to add another logical record, control interval splitting occurs. Some of the logical records and their control information are taken from the full control interval and moved to an empty control interval at the end of the same control area, if a control interval is available. The logical record is added to the appropriate control interval in key sequence.

When control interval splitting occurs, the physical sequence of control intervals within a control area no longer represents the correct sequence of logical records within the control area. Therefore, the index must be updated to reflect this condition. The only times the lowest level index must be updated are when control interval splitting occurs and when a record is added to the end of the data set. Hence, less index maintenance is required for a key-sequenced VSAM data set than for an ISAM data set.

If there is no free control interval within a control area when one is required, control area splitting occurs if there is free space at the end of the extent or if secondary allocation was specified at the time the data set was defined. A new control area is established and the

contents of approximately half of the control intervals in the full control area are moved to the new control area. The new logical record is inserted in the appropriate control area in key sequence. The time required to sequentially retrieve records is only slightly affected by control area splitting. Since the amount of space allocated to the data set is affected by control area splitting, the number of split control areas in a key-sequenced data set should be a factor that is considered when determining whether or not to reorganize the data set.

Logical records can be physically deleted from a key-sequenced data set (using the ERASE macro), and the length of a logical record can be increased or decreased. When space becomes available as a result of deleting or shortening a record, records within the control interval are shifted toward the beginning of the control interval to reclaim the free space and make it available for additions. The way in which free space can be distributed throughout a key-sequenced data set, support of space reclamation, and implementation of control interval and control area splitting are all factors that can minimize or possibly eliminate, in some cases, the need to reorganize a key-sequenced data set. This makes VSAM organization more suited than ISAM to an online environment.

Index Data Set for Key-Sequenced Organization

Like the index for an ISAM data set, the index for a key-sequenced VSAM data set contains key values and pointers. It is built when the key-sequenced data set is initially loaded. Unlike an ISAM index, a VSAM index also contains information regarding available space in the key-sequenced data set index.

The index for a key-sequenced VSAM data set also has a totally different structure from that used for an ISAM index. A VSAM index data set consists of two or more levels of index records structured as a balanced tree, and the highest index level contains only one index record (physical disk record). The one exception to this organization is discussed later. Index records are fixed length and of system-determined size. Each index record contains a number of index entries and a pointer to the next index record at the same index level. (The last index record in a level does not have such a pointer.)

The lowest level of the index is called the sequence set. All levels above the lowest are collectively referred to as the index set. The sequence set index level points to all the control intervals in the key-sequenced data set and contains the high compressed key value in each control interval. Since the sequence set index does not contain an entry for each logical record in the VSAM data set, it is a nondense index level.

Each index record in the sequence set contains a number of index entries that is equal to the number of control intervals in a control area. Hence, there is one sequence set index record per control area in the data set. An index entry in a sequence set index record consists of a key value, control information, and a pointer to the control interval that contains that key. The key in the index entry is the highest compressed key in the control interval.

When the logical record data set has few enough control intervals that one index record can contain all the required index entries, there is only one level of index and it consists of one sequence set index record.

When a key-sequenced data set is processed sequentially, the sequence set index level is used to indicate the order in which control intervals are to be accessed. To improve performance during sequential processing, the sequence set index level can be separated from the rest of the index data set (index set levels) and stored with the logical

records. When this option is chosen, the index records for a control area are placed on the first track(s) of the control area so that both index and logical records can be accessed without moving the disk arm (similar to the location of the track index within the prime area in ISAM).

When the sequence set index level is stored within the logical record area, sequence set records are also replicated. That is, each sequence set index record is allocated one track at the beginning of the control area. The index record is duplicated on the track as many times as it will fit. This technique significantly minimizes the rotational delay involved in arriving at the beginning of an index record. If there is only one control area in a cylinder, sequence set index records will be replicated beginning with track 0. If there are two control areas in a cylinder, initial tracks of the first area will contain replicated index records for the first area, while initial tracks of the second area will contain replicated index records for the second area.

Index set index records, like sequence set index records, contain blocked index entries. The index entries in each level of the index set point to index records of the next lower index level. An index entry within the index set contains a pointer to an index record, the highest key in that index record, and control information. Index set index levels can also be replicated. When this option is chosen, one track is required for each index record in the entire index set. An index record is duplicated on its assigned track as many times as it will fit. The index set may or may not be replicated when the index set and the sequence set are physically separate (sequence set stored with logical records). However, when the index set and the sequence set are stored together, both are replicated or neither is replicated.

The entire index (index and sequence sets) is used to process a key-sequenced data set directly by a user-specified key value. Each index level is inspected beginning with the highest level. One index block in each level must be inspected to obtain a pointer to the next lower level. An advantage of this structure over that of ISAM index structure is the fact that the time to locate any record directly is based on the number of levels in the index and on the location of the index records to be inspected (on the direct access device or in real storage). Therefore, the same time is required to locate an addition as an original record. In ISAM, additional rotation time is required to locate an addition that is not the first addition in the chain in the cylinder overflow area of a prime cylinder.

The index of a key-sequenced data set is designed to require as little direct access space as possible. In addition to being nondense, the index entries contain front and rear compressed keys. Compression is done to eliminate redundant characters in adjacent keys and thereby reduce the amount of key data that must be stored.

Since physical index records are written without a key, index entries are blocked within index records, and keys are compressed, an index record must be present in real storage in order for the user-supplied key value to be compared with the key values contained in an index record. As much of the total index set as possible, up to the entire index set, can be resident in virtual storage if enough buffer storage is specified by the user. Note that the access method does not preload index record buffer(s) with as many index records as will fit. Index records are allocated space in a buffer and loaded when required.

The index records that are resident in virtual storage are pageable; however, heavy referencing of an index record can tend to cause the page containing the index record to remain in real storage. (Index records cannot be fixed in real storage.) If an index entry that is not resident in virtual storage is required, and there is not enough room in

the buffer area provided to add the index record, the access method deletes an existing index record to make room. In general, an index record is selected that has been in the buffer the longest time and that belongs to the lowest level index represented in the buffer.

The index entries in an index record are not inspected sequentially. Entries are divided into sections (zones) for the purpose of searching. This reduces the time required to locate the desired entry. The structure of an index for a VSAM data set is shown in Figure 90.30.2.

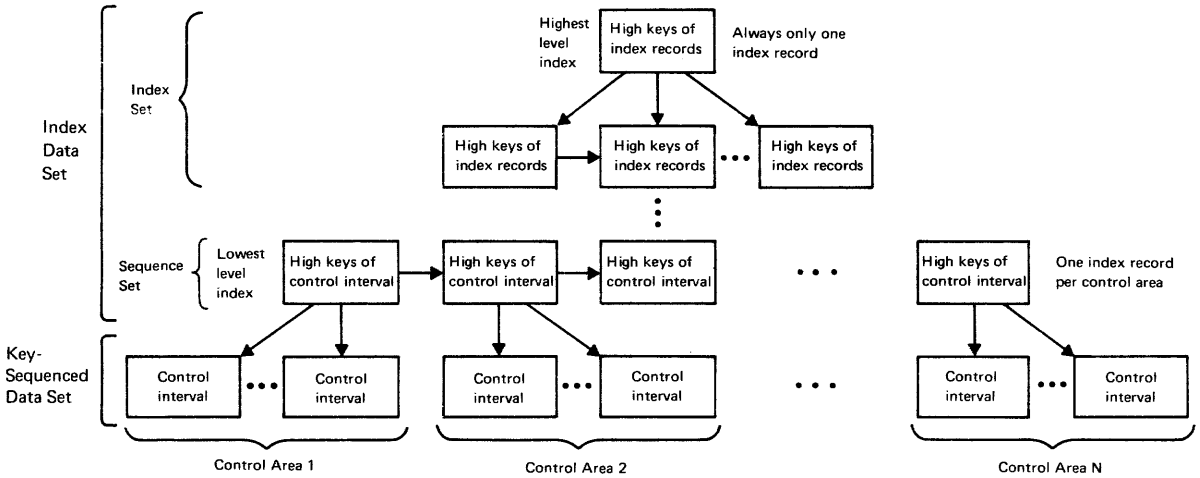


Figure 90.30.2. Structure of the index for a VSAM key-sequenced data set

Key-Sequenced Data Set Processing

The records in a key-sequenced data set can be processed sequentially or directly by key, using the index, or by relative byte address, not using the index. In the latter case, the volume containing the index need not be mounted unless it also contains logical records that are to be processed.

The data in a VSAM data set is considered to be mapped into a byte space which can be over 4 billion bytes in size. The relative byte address (RBA) of a logical record or an index entry is the byte displacement of the logical record or index entry relative to the beginning of the data set. The RBA of a record or index entry, therefore, is independent of the physical characteristics of the direct access device type on which the logical record or index entry resides, the number of extents in the data set, the size of a control interval, etc. All pointers to data that are contained in the index and in control intervals are in terms of relative byte address instead of the record address (CCHHR) that is used in ISAM pointer fields.

In order to locate a desired index or logical record, the access method calculates the disk address of the physical record using the RBA of the record. Hence, a key-sequenced data set is device independent. It can be moved from one direct access device type to another and its index data set need not be re-created. The RBA of a logical record in an existing key-sequenced data set can change only when a record is inserted or deleted, or if the size of a record is altered. A user-written routine should be included to record changes in RBA's when RBA is used for update. This routine is entered from VSAM when appropriate. Hence, programs that process a key-sequenced data set by RBA need not be modified if direct access device type is changed. Processing a VSAM

data set by RBA is called addressed accessing. When addressed sequential retrieval is used, records are retrieved in ascending RBA sequence. Thus, logical records will not be presented in key sequence if there have been any control interval or control area splits.

Entry-Sequenced Organization and Processing

An entry-sequenced data set is physically structured just like a key-sequenced data set except that (1) a control area always contains a minimum of two control intervals, (2) no index is provided, and (3) free space cannot be left within control areas or intervals when the data set is defined. Records can be retrieved directly by RBA or sequentially. Additions are placed in any available space left at the end of the entry-sequenced data set. This free area is also used if the size of an existing record is to be changed. The existing record must be marked deleted by the user with an installation-determined deletion identification, and the lengthened or shortened record must be written at the end of the data set. Space made available by marking a record deleted (because its size is changed or it is no longer required) is not reclaimed, and the ERASE macro is not effective for entry-sequenced data sets. The space occupied by a deleted record can be reused only by storing a new record of the same size in this space.

The only time a change is made in the RBA of a logical record in an entry-sequenced data set is when the size of the logical record is changed. Other records are not affected since the changed record is moved to the end of the data set. Hence, a program can maintain a table of RBA values for the logical records in an entry-sequenced data set that is used for direct record retrieval. The table must be updated only when records are added to the data set and when a record size is increased or decreased. An entry-sequenced data set can also be moved from one direct access device type to another and programs need not be modified because the RBA's of the logical records do not change.

An entry-sequenced data set can also be used like a BDAM data set. Instead of using a table of RBA and control field values, a randomizing routine can be used to associate the control field of a logical record with an RBA. The entry-sequenced data set must be preformatted with dummy records before the logical records are placed in the data set.

Processing Summary

Table 90.30.1 summarizes the types of access supported for key-sequenced and entry-sequenced data sets. All requests are made via GET and PUT macros. VSAM supports processing capabilities that are not provided by ISAM as follows:

- A group of additions that are in ascending sequence can be mass inserted in a key-sequenced data set using sequential instead of direct processing. Mass insertion should be used when the records to be added will be placed between two existing logical records or after the last record. The access method takes advantage of the fact that the additions are in sequence by not writing a control interval (and its sequence index record, if control interval splitting occurs) until the control interval has been packed with all the additions that will fit. The time required to make the additions and update the index is significantly reduced, and the index need not be searched to determine where each new logical record is to be placed.

- Records can be retrieved directly from a key-sequenced data set using a skip sequential technique. When a relatively small number of transactions that are in sequence are to be processed, skip sequential processing can be used to directly retrieve the records by key. Since the keys presented are in sequence, the access method

uses only the sequence set index level to locate the desired records. Skip sequential processing can be used to avoid retrieving the entire data set sequentially to process a relatively small percentage of the total number of records, or to avoid using direct retrieval of the desired records, which causes the entire index to be searched for each record.

- Both sequential and direct processing can be performed on a key-sequenced or an entry-sequenced data set using one OPEN and one access control block (ACB). The ACB is the equivalent of a DCB for VSAM. Closing and reopening of the data, as is required for an ISAM data set, is not necessary.

Table 90.30.1. The types of access supported for VSAM data set organizations. An entry indicates whether the function is supported using sequential or direct processing, whether or not a key or RBA is required, and whether or not keys or RBA's must be presented in sequence.

	Key-Sequenced Data Sets			
			Entry-Sequenced	
Types of Access	Keyed Sequential	Keyed Direct	Data Sets Addressed Sequential	Addressed Direct
Retrieval only	X No key presented	X Keys not in sequence	X No RBA presented	X RBA's not in sequence
Skip sequential retrieval, update and addition		X Keys in sequence		
Retrieve and update, including changing record size	X No key presented	X Keys not in sequence	X No RBA presented	X RBA's in sequence
Add Mass insertion Direct insertion	X Keys in sequence	 X Keys not in sequence	 X No RBA presented	
Delete	X Keys in sequence	X Keys not in sequence	(User must flag records)	X RBA's not in sequence

- Several parts of a key-sequenced or an entry-sequenced data set can be processed concurrently by a program or its subtasks using the same ACB. This facility is called multiple-request processing. Several requests for the same data set can be grouped and issued as one request with a single macro. Sequential-processing requests and direct-processing requests can be mixed within the same multiple-request group. A multiple-request can specify synchronous or asynchronous processing. If synchronous processing is indicated, one request in the group is processed at a time and control is

returned to the user (next instruction after the request) after the access method has processed all requests in the group. If asynchronous processing is specified, control is returned to the user as soon as the multiple-request is accepted. Requests in the group are processed one at a time and the programmer must check for the completion of each individual request. Several asynchronous multiple-request processing requests can be active concurrently for the same data set. The access method processes each multiple-request independently and asynchronously from all other outstanding multiple-requests. Concurrently executing requests from different multiple-requests can access the same logical record simultaneously unless exclusive control has been specified. Within a partition, exclusive control for update and insert requests is supported.

- Records can be retrieved directly from a key-sequenced data set by RBA, generic key, or key greater than supplied key, as well as by equal key. ISAM permits positioning by record ID, by generic key, or by key greater than supplied key but the record must be retrieved in sequential mode via a separate operation.

VSAM Catalogs

Unlike ISAM data sets, all VSAM data sets (index as well as those with logical records) must be cataloged in a VSAM catalog, which is formatted as a key-sequenced VSAM data set. Information required to process a VSAM data set, such as its location and characteristics, is contained in the VSAM catalog.

There must be one VSAM system catalog for a VS1 operating system and, optionally, one or more VSAM user catalogs can be defined. Each catalog is an individual data set. The VSAM system catalog data set is cataloged in the VS1 data set catalog (SYSCTLG), and each VSAM user catalog has an entry in the VSAM system catalog. Each VSAM data set is cataloged in the VSAM system catalog or a user catalog, but not both. All VSAM data sets on the same volume must be cataloged in the same VSAM catalog.

VSAM user catalogs can be used to reduce the size of the VSAM system catalog (to reduce catalog processing time), minimize the effect of a damaged catalog, and enable a VSAM data set to be portable from one system to another without having to use the access method services program to process VSAM catalogs.

The following information is recorded in the catalog entry for a VSAM data set:

- Device type and volume serial numbers of volumes containing the data set
- Location of the extents of the data set
- Attributes of the data set, such as control interval size, number of control intervals, etc.
- Statistics such as the number of insertions, the number of deletions, and the amount of remaining free space
- Password protection information
- An indication of the connection of a key-sequenced data set and its index
- Information that indicates whether a key-sequenced data set or its index has been processed individually (without reference to the other)

A VSAM catalog also contains information regarding the available space on volumes that contain VSAM data sets. Therefore, a volume containing a VSAM data set need not be mounted in order to determine whether or not it contains available space. VSAM catalog/DADSM routines, instead of OS catalog and DADSM routines, are used to process the catalog and to allocate space in VSAM catalog and data set volumes. Generation data groups of VSAM data sets cannot be defined in a VSAM catalog. In addition, temporary and concatenated VSAM data sets are not supported.

Access Method Services Program

The access method services general purpose, multifunction service program is provided to support functions required to create and maintain VSAM data sets. Facilities to convert ISAM and SAM data sets to VSAM organization are also included. The access method services program is invoked via a calling sequence and the functions desired are requested via a set of access method services commands. In VS1, the calling sequence and commands can be placed in the input stream or issued within a processing program.

The access method services program is used to:

- Define and allocate direct access space for all VSAM data sets and all VSAM catalogs. The DEFINE function must be used to describe a VSAM data set or catalog before any data is placed in the data set or the catalog. A key range can be specified for each volume in a key-sequenced data set.

- Create, reorganize, and back up VSAM data sets. Input to the COPY function can be an ISAM, SAM, or VSAM data set. The output can be a VSAM or SAM data set. When the input and the output organizations are different, conversion occurs. The COPY function, therefore, can be used to convert an ISAM data set to VSAM format, initially create a VSAM data set from sequenced records, merge new logical records into an existing VSAM data set, and reorganize a VSAM data set.

- Print all or some of the logical records of a SAM, ISAM, or VSAM data set. Three formats are supported: each byte printed as a single character, each byte printed as two hexadecimal digits, and a combination of the previous two (side by side).

- Maintain VSAM catalogs (alter, delete, or list catalog entries). Certain characteristics of a VSAM data set can be modified by altering the catalog entry for the data set.

- Perform processing required to make a VSAM data set portable from one System/370 to another if a user VSAM catalog is not available. This involves extracting required information about the VSAM data set from the VSAM system catalog of one operating system and inserting this data in a VSAM catalog of another operating system.

- Verify the accessibility of an existing VSAM data set. This function involves checking for a valid end-of-file indication and reestablishing the EOF record when necessary.

Since VSAM data sets must be cataloged, and the access method services program must be used to define and allocate space for VSAM data sets, a minimum number of job control parameters for DD statements are used by VSAM. Three new DD statement parameters are defined for VSAM: JOBCAT and STEPCAT for specifying VSAM catalogs, and AMP for overriding parameters specified in the processing program.

Password Protection

An expanded password protection facility is supported for VSAM. Optionally, passwords can be defined for logical record data sets, index data sets, and VSAM catalogs. Passwords are kept in VSAM catalog entries. The operator must supply the correct password in order for a data set to be opened. Up to seven retries can be made.

Multiple levels of protection are provided:

- Master access, which allows access to a data set, its index, and its catalog entry. Any operation (read, add, update, delete) can be performed.
- Control interval access, which allows the user to read and write entire control intervals instead of logical records. This facility is not provided for general use and should be reserved for system programmer use only.
- Update access, which allows logical records to be retrieved, updated, deleted, or added. Limited modification of the catalog entries for the data set is permitted, but an entry cannot be deleted.
- Read access, which allows access to a data set for read operations only. Read access to the catalog entries of the data set is permitted also. No writing is allowed.

Authorization to process a VSAM data set can be supplemented by a user-written security authorization routine. If supplied, such a routine is entered during OPEN processing after password verification has been performed, unless the master-access password was specified. A user-security authorization record can also be added to the catalog entry for the data set. This record can supply data to the user-written security authorization routine during its processing.

ISAM Interface Routine

The ISAM interface routine is provided as an aid in converting from ISAM organization to VSAM organization. It enables existing programs that process ISAM data sets to be used to process key-sequenced VSAM data sets without modification of ISAM macros. The VSAM data sets can be newly created or those that have been converted from ISAM format to VSAM key-sequenced format. The ISAM interface routine permits VSAM key-sequenced data sets to be processed by both ISAM programs and VSAM programs. This allows existing ISAM application programs to be used and additional applications that take advantage of new VSAM facilities to process the same VSAM data sets.

The ISAM interface routine operates in conjunction with VSAM access method routines. The interface routine intercepts ISAM requests and converts them to equivalent VSAM requests. Hence, only functions of ISAM that are equivalent to those of VSAM are supported by the ISAM interface routine. There are a few ISAM facilities that the ISAM interface routine does not support. These are discussed in OS/VS Virtual Storage Access Method Planning Guide, GC26-3799. Similarly, if VSAM facilities that are not supported by ISAM are to be used, an existing ISAM program must be modified to define a VSAM data set and to use VSAM macros. Assembler Language macros for ISAM and VSAM are not compatible.

When the ISAM interface routine is used by an ISAM program, existing job control for the ISAM data must be modified as appropriate. The ISAM interface routine and the access method services program simplify the

amount of effort required to replace ISAM data set organization with VSAM organization within an installation.

Summary

Highlights of VSAM when it is compared with ISAM are as follows.

VSAM provides new features:

- Two data organizations are supported, one with records in key sequence and one with records in time-of-arrival sequence.
- Data sets are device-type independent.
- Direct access space utilization is maximized by device type by using spanned blocked logical records within a control interval.
- Additions and index entries are blocked, which also reduces disk space requirements.
- Secondary space allocation is supported so that an existing data set can be extended.
- Free space for additions can be allocated at more frequent intervals throughout the allocated extents when the data set is created.
- Free space reclamation capabilities are expanded considerably, which can eliminate or significantly increase the time between data set reorganizations.
- Password protection is extended to provide more levels of protection, and user-written security protection routines are supported.
- Disk volumes containing VSAM data sets are portable between DOS/VS and OS/VS when VSAM features supported by both OS and DOS are used.

VSAM provides performance enhancements:

- Mass insertion processing reduces the time required to insert a group of new sequenced records between two existing logical records or at the end of the data set.
- Skip sequential processing reduces the time required to sequentially process a low volume of transactions.
- Total index size is reduced by compressing keys and blocking index entries. This minimizes index search time.
- Overflow chains are eliminated, which reduces the time required to make an addition.
- The same time is required to retrieve an added record as an original record.
- Index set and sequence set index records can be replicated to significantly reduce rotational delay when accessing index records on disk.
- Index set records, up to a maximum of all index set records, can be resident in virtual storage.

Table 90.30.2 compares the features of VSAM and ISAM as supported in OS/VS1 and OS/VS2.

Table 90.30.2. Comparison table of VSAM and ISAM facilities for OS

Characteristic	VSAM - OS	ISAM - OS
1. Supporting OS environments	VS1 and VS2	PCP, MFT, MVT, VS1 and VS2
2. Direct access devices supported	2314/2319, 3330-series, 2305 Models 1 and 2	Same as VSAM plus 2301, 2302, 2303, 2311, and 2321
a. RPS supported	Yes	Yes
b. Track overflow supported	No	No
3. Types of organization		
a. Key-sequenced	Yes Records are maintained in ascending sequence by key. An index is provided. The logical records and the index are two separate data sets. The key-sequenced data set contains logical records, distributed free space for additions (as an option), and, optionally, the sequence set index level.	Yes Records are maintained in ascending sequence by key. An index is provided that is part of the ISAM data set. The prime area contains logical records, the track index, and optionally overflow tracks in each cylinder for additions. A separate additions area can exist also. The cylinder and master index levels are a separate extent.
b. Entry-sequenced	Yes Records are sequenced by the order in which they are placed in the data set. Records are added to the end of an existing data set. No index is provided.	Not supported
4. Multiple extents and volumes for a data set	Yes	Yes
a. Secondary space allocation indicated at creation	Yes	No The space originally specified cannot be extended
b. Volumes of the same device type required	Yes for logical record extents. The index set can be on a device type that is different from that which contains the key-sequenced logical records.	Yes for all the volumes containing prime and separate overflow area extents. Index levels can be on a device type that is different from that which contains prime and overflow areas.
c. All volumes must be online at OPEN regardless of the type of processing	No	Yes
d. Free space available within the logical record area	Yes (for key-sequenced data sets) within control intervals and control areas. Free space is distributed within the tracks of a cylinder.	Yes, optionally, at the end of each prime cylinder. Free space on tracks within the prime cylinders can be created only by including deleted records when the data set is created.
e. Data set is device independent	Yes RBA pointers are used in the control interval and in the index	No Record address ID (CCHHR) is used in index pointers
5. Key-sequenced organization data set characteristics		
a. Fixed and variable length logical records	Yes Spanned blocked record format is used within a control interval. Original records and additions are blocked.	Yes Fixed or variable, blocked or unblocked record formats are used for prime records. Records in an overflow area are always unblocked.
b. Key field is written on disk	No Records are written in count and data format.	Yes Records are written in count, key, and data format.

Table 90.30.2. Comparison table of VSAM and ISAM facilities for OS (continued)

Characteristic	VSAM - OS	ISAM - OS
c. Key field must be embedded within each logical record	Yes	Yes, except for unblocked fixed length records
d. Key must be fixed length	Yes	Yes
e. Logical records with duplicate keys permitted	No	No
f. Physical record sizes supported	512, 1024, 2048, and 4096 bytes only	Block size specified by the user up to a maximum of the track size.
6. Index structure		
a. Number of levels	Two to N based on the number of index entries required and their size. Index is a balanced tree with one index record in the highest level index.	Track and cylinder index levels are required. Up to three master index levels are optional.
b. Nondense index	Yes	Yes
c. Key field written	No Index records are written in count and data disk record format.	Yes Index records are written in count, key, and data disk record format.
d. Index records are blocked	Yes	No
e. Index record size	Fixed length and determined by system.	Data field is always 10 bytes. Key field is key size.
f. Keys are compressed	Yes Both front and rear compression is performed to eliminate redundant characters.	No Full key is always written
g. Index record replicated on track to reduce rotational delay	Yes, as an option.	No
h. Sequence set index level adjacent to logical records	Optional If chosen, sequence set index records are replicated at the beginning of each control interval area.	Standard Track index is always on the first track(s) of prime cylinders.
i. Index resident in virtual storage	Standard As many index records as will fit in the user-specified buffer can be resident, up to a maximum of all index set records.	Optional Only the highest level can be made resident. Residence of part of an index is not supported.
j. Multiple indexes for the same key-sequenced data set	No	No
7. Types of processing supported for key-sequenced data sets		
a. Sequential retrieval and update without presenting key	Yes Each logical record is presented in key sequence. The sequence set index level is used.	Yes Each logical record is presented in key sequence. The track index is used.
b. Skip sequential retrieval and update (by keys specified in sequence)	Yes Only the sequence set index level is used.	No

Table 90.30.2. Comparison table of VSAM and ISAM facilities for OS (continued)

Characteristic	VSAM - OS	ISAM - OS
c. Sequential retrieval and update by record address	Yes, via presenting RBA's in sequence	Positioning via a SETL macro using record ID (CCHHR) is supported. Record must be retrieved sequentially after positioning.
d. Sequential updating by sequenced keys without retrieving records	No	Yes
e. Direct retrieval and update by generic key, equal key, or key-greater-than the specified key	Yes	Yes for equal key. Generic key and key greater than specified key can be used in a SETL macro for positioning. The record must be retrieved separately using sequential mode.
f. Direct retrieval and update by record address	Yes, via RBA	Yes, via record ID (CCHHR)
g. Additions by direct processing	Yes	Yes
h. Additions by mass insertion using sequential processing and key sequenced additions	Yes	No
i. Concurrent sequential and direct processing of the same data set with a single OPEN	Yes	No The data set must be closed and reopened to change modes. Alternately two DCB's, one for sequential and one for direct processing, can be used.
j. Deletions physically removed	Yes Records are shifted and free space is reclaimed.	Limited Records are flagged when deleted. Deletions are physically removed only if they are forced off a prime track or when a full track of variable length records is reorganized for an addition. A record that is marked deleted can be replaced with a record of the exact same size.
k. Logical records can be lengthened or shortened	Yes, and space is reclaimed for a shortened record.	Yes
l. Multiple-request processing is supported within a single program or a program and its subtasks.	Yes, with one ACB.	Yes, using multiple DCB's.
m. Write check after a write	Optional	Optional
n. Locate and move mode processing	Locate mode for read-only operations and move mode supported	Yes
o. OPEN validation of end-of-data indication	Yes Abnormal termination never occurs during OPEN processing.	Yes Abnormal termination can occur during OPEN processing.
8. Checkpoint/restart facilities	Yes, same as for ISAM	Yes
9. Password protection	Yes Levels supported for the user are: • Master access - allows access to the data set, its index data set, and its catalog entry for all operations	Yes Two levels of protection are provided. If the current password is presented, the data set can be opened for read only or for read and write processing.

Table 90.30.2. Comparison table of VSAM and ISAM facilities for OS (continued)

Characteristic	VSAM - OS	ISAM - OS
	• Control interval access - allows read/write of entire control interval instead of individual logical records. • Update access - allows access to the data set and its index for retrieval, updating, deletions, and additions. Limited modification of catalog entries for the data set is permitted but an entry cannot be deleted. • Read access - allows retrieval of data records and catalog entries (no writing of any kind).	
a. User written authorization routines supported	Yes	No
10. Data set sharing		
a. Within a partition	Yes, with exclusive control support	Yes, with exclusive control support
b. Across partitions (DISP=SHR)	Yes, without exclusive control support	Yes, with exclusive control support
c. Across systems	Yes, exclusive control can be achieved using the RESERVE macro	Same as VSAM
11. Data set cataloging	Required The VSAM system catalog or a VSAM user catalog must be used.	Optional The OS data set catalog is used. There is no special catalog for ISAM data sets.
12. Languages supporting VSAM	Assembler COBOL (via ISAM interface routine) PL/I (via ISAM interface routine)	Assembler COBOL PL/I RPG
13. VSAM data set direct input to sort/merge	No	No
14. Utility program functions	Access method services program can perform the following: • Define direct access space for a VSAM data set • List, alter, or delete an existing VSAM catalog entry • Create new and reorganize existing VSAM data sets • Copy a VSAM, ISAM, or SAM disk data set to a new SAM data set or into an existing VSAM data set • List some or all of the records in a VSAM, an ISAM, or a SAM data set • Perform functions required to make a VSAM data set portable from one system to another • Verify and reestablish, if necessary, the end-of-file marker in one VSAM data set	IEBISAM utility can perform the following: • Copy an ISAM data set from one disk volume to another, dropping deletions and merging additions into the prime area • Unload an ISAM data set onto a tape or a disk volume, dropping deletions and creating a backup sequential data set suitable for input to the load operation to re-create the ISAM data set • Load a previously unloaded ISAM data set from tape or disk onto a disk volume merging additions into the prime area • Retrieve and print the records of an ISAM data set, except deletions, or create a sequentially organized data set from active records

90:35 PAGE MANAGEMENT

GENERAL FUNCTIONS

Page management consists of a set of routines that manage real storage and external page storage. Page management implements demand paging and provides the program support required by dynamic address translation hardware for implementation of a virtual storage environment. The following routines are part of the page management function and are contained in the resident nucleus:

- Page exception handler
- Service interface routine
- Task switch analysis routine
- Real storage management routines
- External page storage management routines

The page exception handler and service interface routines channel requests to the task switch analysis routine, which processes certain types of requests and passes others to real storage management and external page storage management routines for servicing. The last two routines are referred to as the page supervisor, and operate as a task. The page supervisor has the highest priority of any task in the system.

The page exception handler (PEH) is entered after an implicit request for a page management service occurs (page translation exception). The PEH constructs a control block to describe the request and passes it to the service interface routine as an explicit request.

The service interface routine receives all explicit requests for page management services. The following services can be requested via page management macros:

- Make one or more virtual storage pages addressable and mark them fixed (PGFIX macro). Available page frames are allocated to the virtual storage pages and, if necessary, page-in operations are scheduled to cause the contents of the virtual storage pages to be loaded. A release parameter can be specified to indicate that a page-in is not required, such as when page frames are allocated for buffer space. Pages marked fixed cannot be paged out until a PGFREE macro is issued. PGFIX requests can also request that the real addresses of the page frames assigned be made available to the requester.

- Make one or more virtual storage pages addressable (PGLOAD macro). The service performed is like that for PGFIX except that the page frames allocated are not fixed. The PGLOAD macro provides a page-ahead facility.

- Mark the page frames allocated to the virtual storage pages indicated unfixed (PGFREE macro). A release parameter can also be specified to indicate that the contents of the unfixed pages are no longer required so that a page-out is avoided.

- Deallocate the page frames allocated to the virtual storage pages indicated (PGRLSE macro). The page frames are made available for allocation without a page-out. The virtual storage pages specified are marked invalid in the appropriate page table entries.

Page management services are implemented primarily for use by control program routines. The PGRLSE macro is the only page management macro that can be issued by a problem program. The other page management macros can be issued by a task if the task operates in supervisor state or has a protect key of zero.

REAL STORAGE MANAGEMENT

Real storage management routines process requests for the allocation and deallocation of real storage (page frames). The technique implemented is designed to keep real storage allocated to the pages that are deemed to be the most active at any time. Real storage management also monitors the availability of real storage and, when it is about to become totally allocated such that thrashing will occur, takes steps to prevent this condition.

The status of all real storage in the system is reflected in the real storage page table (RSPT), which is located at the end of the resident control program in the nonpageable area of real storage. The RSPT contains one 16-byte entry for each 2K page frame in the system. The entries in the RSPT are arranged in several page status queues. That is, entries are connected by pointers to form various queues. The RSPT entries are initialized at IPL and thereafter always reflect the current status of each page frame.

An RSPT entry contains identification of the task to which it belongs, the number of the virtual storage page to which it is assigned, flags to indicate its status (short- or long-term fixed, being paged in or out, allocated to a nonpageable job step, etc.), and queue pointers to indicate the page status queue of which it is a part.

Logically, the following page status queues are maintained:

- Available page queue that indicates the page frames that are available for allocation when page faults and page load/fix requests occur. When page frames are released, such as at end of job step, they are placed in this queue. Allocated page frames that become inactive can be placed on this queue. An available page count (APC) is maintained that always reflects the number of page frames in this queue.

- In-use queues that reflect the allocated page frames that are not fixed. As page frames in the in-use queues become inactive, they are subject to being placed in the available page queue.

- Logical fix queue that indicates the page frames that are currently in long- or short-term fixed status (SQA, fixed PQA, nonpaged job step pages, nucleus pages, I/O buffer pages, etc.). RSPT entries for fixed pages are not actually connected to form a queue.

- Malfunctioning page queue that contains the page frames that cannot be assigned because the MCH routine indicated they are malfunctioning (see discussion in Section 90:40)

The dynamic storage allocation routine is responsible for servicing real storage allocation requests. The allocation technique implemented attempts to (1) minimize paging requirements associated with the real storage allocation process itself, (2) minimize task wait time associated with real storage allocation, and (3) keep real storage assigned to the most active pages to reduce paging activity for executing tasks.

Real storage is allocated from the available page queue, which contains unassigned page frames. Frequently referenced page frames are normally not taken from one task to be allocated to another. If a situation arises in which there are no unassigned page frames available for allocation to a task, the real storage release routine is entered to make real storage available. If there are not enough allocated page frames that were not recently referenced to satisfy the request, a deactivation procedure is entered to make real storage available.

Tasks execute on a priority basis and, therefore, requests for page frames are received and serviced on a priority basis. However, page management does not ever attempt to ensure that a given number of page frames are allocated to each task (page frames are allocated to the currently most active pages without regard for the task to which they belong). Unauthorized pageable problem programs do not have any control over when or how many page frames are allocated to their pages.

Real Storage Allocation Procedure

The following is done to service a real storage allocation request (refer to Figure 90.35.1). The real storage reclamation routine, a subroutine of the dynamic storage allocation routine, determines whether a page-in can be avoided because the contents of the referenced virtual storage page are still in real storage. This condition exists when the page frame last assigned to the virtual storage page has not yet been reassigned. It can also occur when a page-in that was initiated by a previous request for the same page makes the page available after the second page fault occurs. If the RSPT entry for a desired page frame is still in the available page queue, an in-use queue, the page-out queue, or the logical fix queue, page reclamation is possible and the page frame is reassigned without a page-in.

If reclamation is not possible, the dynamic storage allocation routine attempts to allocate the requested number of page frames from the available page queue. If the number of page frames requested can be allocated from this queue, their RSPT entries are removed from the queue and the available page count is decremented. If a page-in is required for a page (user bit in the page table entry is on), the RSPT entry of the page frame assigned is placed in the appropriate page-in device queue. Otherwise, the RSPT entries are placed in an in-use queue or in fixed status and the allocated page frames are initialized to zero (for data security protection). The appropriate page table entries are updated to reflect the allocation of real storage.

If the allocation request does not indicate long-term fixing, page frames are allocated from the beginning of the available page queue. If the request does indicate long-term fixing, an optimization routine is entered to select a page frame that will least fragment real storage. This is done to leave as much contiguous real storage available as possible for allocation to nonpageable job steps. A page frame close to the end of the resident control program or the V=R line that is not currently fixed, or conditionally allocated to a nonpageable job step, or in a page-in operation is chosen as the optimum page frame. If the optimum page frame chosen is currently allocated to a virtual storage page, an available page frame is obtained and the contents of the selected optimum page frame are moved to it.

A request for SQA has the highest real storage allocation priority. If an SQA request cannot be satisfied, the requesting task is terminated or system processing terminates, depending on the reasons for the SQA request.

If the available page queue does not contain enough page frames to service a request, or if the available page count (APC) reaches or falls below an APC low threshold value as a result of page frame allocation, the dynamic storage allocation routine gives control to the real storage release routine. The APC low threshold is used to indicate the point at which the available page queue should be replenished with least-recently referenced page frames from the in-use queues. The real storage release routine performs the replenishment function.

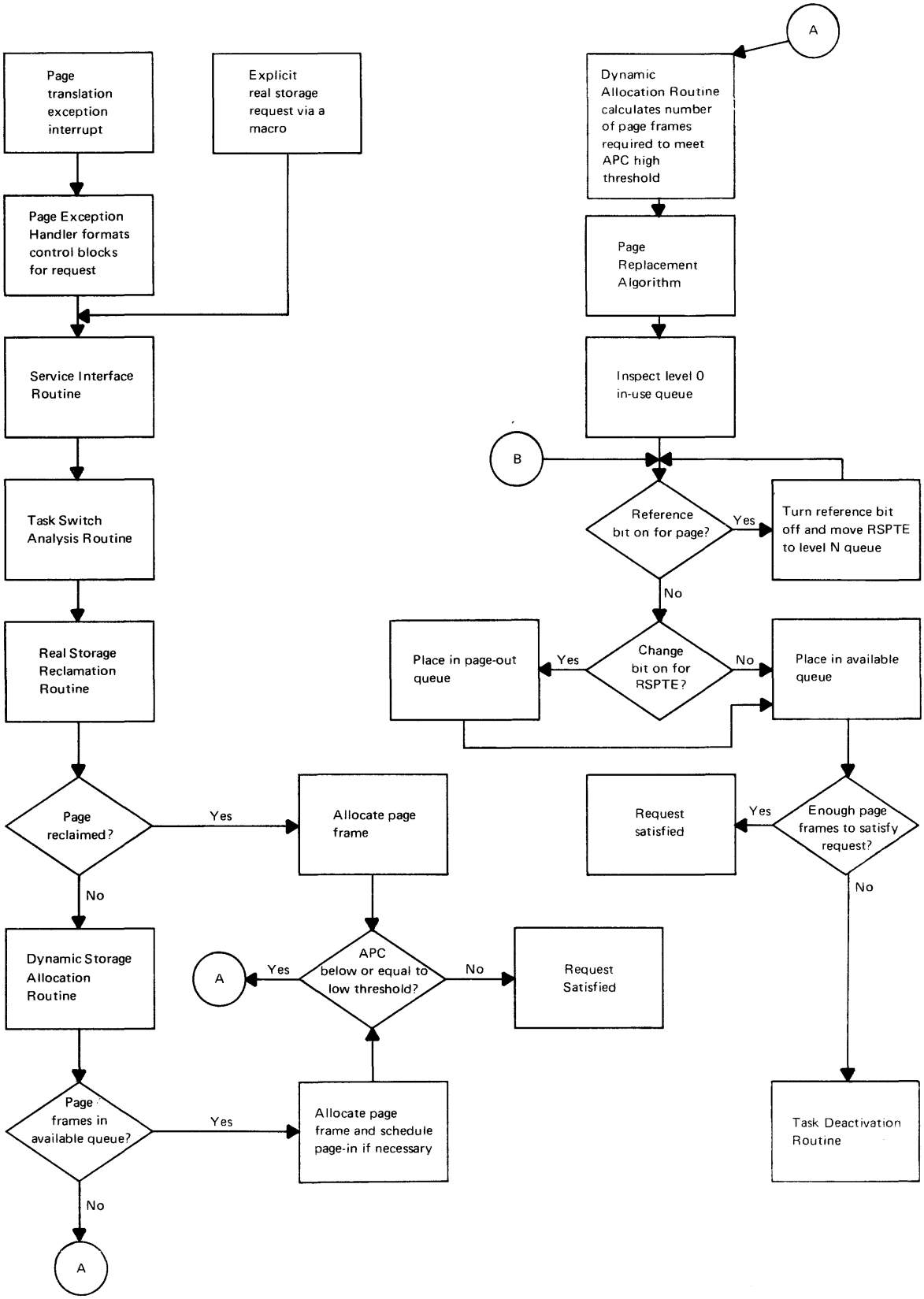


Figure 90.35.1. Flow of the real storage allocation procedure

The aim of the real storage release routine is to keep enough page frames in the available page queue to enable the dynamic storage allocation routine to allocate real storage without the necessity of a page-out operation. This routine calculates the number of page frames that should be placed in the available page queue in order to satisfy a request and raise the APC to a satisfactory level (a high threshold value for the APC). A request for the number of page frames calculated is passed to the page replacement algorithm.

The function of the page replacement algorithm is to replenish the available page queue by enqueuing on it least-recently referenced page frames taken from the in-use queues. If a page-out operation is required (change bit for the page frame is on), the RSPT entry is routed to the appropriate page I/O device queue. The technique used to determine which page frames to remove from the in-use queues is designed to ensure that the most recently referenced pages remain in real storage.

In order to determine the activity of pages, a series of in-use queues that contain RSPT entries are maintained. The number of active in-use queues at any given time is determined by the number of active partitions and system functions. Each in-use queue is assigned a reference level sequence number 0 to N. The reference level 0 in-use queue contains entries for the page frames referenced longest ago and, hence, tends to identify the least active pages. The reference level N in-use queue contains entries for the page frames most recently allocated and referenced and tends to identify the most active pages. The reference level N-1 queue indicates the next most active pages, etc.

An RSPT entry is placed in the level N in-use queue when it is assigned to a virtual storage page. If the RSPT is taken from the available page queue, its reference bit is turned on. If a page-in operation was required prior to placing the RSPT in the level N queue, the reference bit is already on as a result of the I/O operation. Page management ensures that the reference bit is turned on when a page frame is allocated so the RSPT entry will cycle through the in-use queues at least twice before it becomes eligible for placement on the available page queue. This technique allows a task to use a page frame it has been assigned before the page frame becomes eligible for assignment to another page.

The activity (frequency of reference) of a page frame is determined by inspecting its reference bit setting at certain intervals. The frequency of reference of page frames in the in-use queues is measured at problem program task switch time by the page replacement algorithm. Activity measurement is not performed every time a task switch takes place. The frequency of measurement is based on the number of active initiators. As the number of active initiators increases, the frequency of measurement decreases. As the number of active initiators decreases, the frequency of measurement increases. Hence, the number of problem program task switches that occur between measurements can vary.

At measurement time, all the RSPT entries from the reference level 1 queue, if any, are moved to the end of the level 0 queue. Then, the RSPT entries in all reference level queues except the level 0 queue are shifted to the next lowest reference level queue (all level 2 entries are placed in the level 1 queue, all level 3 entries are placed in the level 2 queue, all level N entries are placed in the level N-1 queue). Reference bits are not reset. Once the level shifting is complete, the reference bit of each RSPT entry in the reference level 0 queue, up to a maximum of 20 entries, is inspected from top to bottom. Only 20 entries are inspected in order to limit the time required to process the level 0 queue. If the reference bit in the page frame associated with an RSPT entry is on, indicating the page was referenced during processing that occurred since frequency of reference was last measured, the RSPT entry

is placed at the end of the level N queue and its associated reference bit is turned off. Level 0 then contains entries only for pages that have not been referenced since the last measurement (if it contains fewer than 21 entries).

Using this technique, page frame entries move toward the level 0 queue. If the page to which an RSPT entry is assigned has not been referenced during the period of time it takes the entry to get to the level 0 queue, the entry is considered to be assigned to an inactive page and is subject to being placed in the available page queue. The available page queue is not replenished at measurement time. (Figure 90.35.2 illustrates page activity measurement processing.)

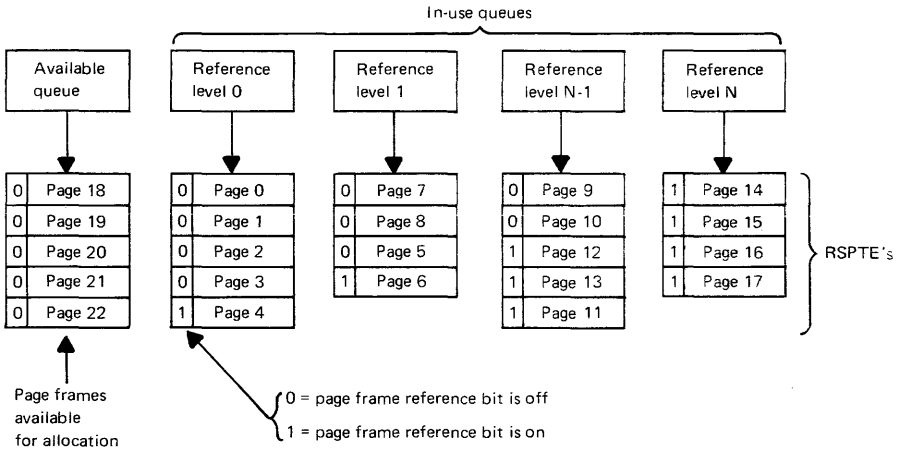
When the page replacement algorithm receives a replenish request, it attempts to satisfy the request by placing the indicated number of page frames in the available page queue. Initially, only page frames contained in the reference level 0 queue are eligible to satisfy a replenish request. To determine the activity of the pages represented in the level 0 queue, the page replacement algorithm inspects the reference bits associated with the RSPT entries in this queue, starting with the first RSPT entry in the queue. If the reference bit is on for a page frame, its RSPT entry is placed in the reference level N queue and the reference bit is turned off. If the reference bit is off, the change bit determines where the entry is placed. If the change bit is off, the entry is placed in the available page queue. If the change bit is on, the entry is placed in the appropriate page I/O device queue so that the contents of its associated page frame can be written out before the entry is placed in the available page queue. Inspection of the level 0 queue RSPT entries continues until enough unreferenced page frames to satisfy the request are selected or until the entire queue has been searched.

The page replacement algorithm returns control to the real storage release routine and, if the replenish request could not be satisfied, indicates how many of the page frames requested could not be released (placed in the available page queue or a page I/O device queue). This data is returned to the dynamic storage allocation routine. If the request was satisfied, the next queued real storage request is initiated. If the request was not satisfied, which signifies that real storage could not be allocated unless it was taken away from other active pages, the task deactivation/reactivation module is entered.

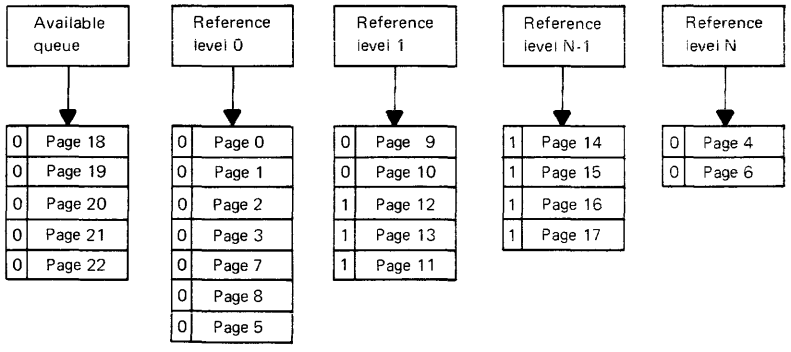
The primary function of the deactivation/reactivation module is to adjust the paging activity of the system to the availability of real storage so that throughput is optimized. When real storage is totally allocated and a real storage request must be satisfied, the deactivation routine attempts to select active page frames to satisfy the request such that paging activity is reduced. Similarly, the reactivation routine does not attempt to reactivate a deactivated partition until it determines that real storage is not being fully utilized so that reactivating a partition will not cause excessive paging to recur.

The task deactivation routine attempts to suspend the processing of a pageable problem program task (mark it nondispatchable) so that the real storage currently allocated to the task can be released and made available for allocation. Only dispatchable pageable partitions that are not in a disabled state and that do not have system-must-complete-ENQ's outstanding are eligible for deactivation. A partition is not deactivated if it currently is the only dispatchable partition and P0 is never deactivated.

Status of page queues and page frames at activity measurement time



Status of page queues and page frames after activity measurement time



Measurement Steps

1. Append RSPT entries (RSPTE's) from the level 1 queue to the level 0 queue (RSPTE's for pages 7, 8, 5, 6).

2. Shift RSPTE's on all reference level queues one level to the left, except for RSPTE's on the level 0 queue, placing shifted RSPTE's at the end of the new queue.

3. Move all RSPTE's from the level 0 queue to the level N queue that have their reference bit on in the associated page frame, and reset the reference bit to zero (RSPTE's for pages 4 and 6).

Figure 90.35.2. Example of page activity measurement

The lowest priority eligible partition is selected for deactivation first. After a partition has been suspended, the task deactivation routine makes the page frames assigned to the deactivated partition available by queuing their RSPT entries in the level 0 reference queue and turning off their reference bits. Control is returned to the dynamic storage allocation routine which attempts to satisfy the allocation request. If deactivation of the partition did not release enough page frames to satisfy the request, the next lowest priority partition is deactivated if it is eligible.

If no partitions are eligible for deactivation or if deactivation fails to make the required number of page frames available, a determination of whether any deferred V=R allocation requests are pending is made. If such a request is pending, it is overridden, and any page frames conditionally assigned to the V=R request are made available. The operator is notified and can request cancelation of the nonpageable job. If this procedure does not make enough page frames available, the in-use queues are inspected in lowest to highest level sequence and page frames are taken from the lowest level in-use queues first. This process continues until the request is satisfied.

The determination of whether or not partition reactivation can take place is made when the system is about to enter the wait state, a situation that could indicate a reduction in paging activity. At that time, the reactivation routine checks to see if there is paging I/O currently in progress (which could indicate that high paging activity has caused the system wait), there is pending I/O activity, or a significant portion of real storage is fixed (which temporarily makes less real storage available for paging). Reactivation does not occur if any one of these conditions exists or if there is not enough real storage available to reactivate the task. Reactivation also occurs if no paging activity has occured for the last six seconds and enough page frames are available, or if a partition runs out of work and enough page frames are available. When partition reactivation is performed, the highest priority deactivated partition is reactivated first. Only one deactivated partition is reactivated at this time.

Allocation of a V=R Area to a Nonpageable Job Step

Real storage allocation requests for nonpageable job steps have the lowest allocation priority if they must be enqueued because they cannot be satisfied immediately. Real storage allocated to a nonpageable job step must be contiguous and if a nonpageable job step is being restarted, the same page frames previously allocated must become available before the allocation request can be satisfied.

Prior to passing a V=R storage allocation request to the V=R allocation routine, the task switch analysis routine determines whether the request is too large to be satisfied. If allocation of the number of page frames indicated in a V=R request would cause the number of page frames available for paging operations to be reduced below the minimum requirement, a message is given to the operator indicating that the request cannot be satisfied.

The V=R allocation routine inspects the RSPT for a contiguous area within the V=R area that is large enough to satisfy the request (areas between long-term fixed pages). If long-term fixed pages have fragmented real storage within the V=R area to the extent that there is not enough contiguous real storage to satisfy the request, the operator is informed and can reply with a cancel or a retry request.

If the required contiguous space is conditionally available (the area contains only page frames that are available, that are allocated but not fixed, or that are short-term fixed), the RSPT entries for the area are

considered conditionally available and are marked for interception and allocation to the current V=R request. Available page frames in the area selected are allocated to the V=R request immediately and the residual allocation count, which indicates the number of additional page frames required to satisfy the request, is updated. As RSPT entries that are flagged as conditionally assigned are intercepted (when a PGRLSE is processed, when the available page queue is replenished, for example) and allocated to the V=R request, the residual count is decremented. When the count reaches zero, the V=R allocation request is satisfied.

Real Storage Release

The task switch analysis routine processes PGRLSE requests. The entries for the page frames allocated to the virtual storage pages being released are taken from the queue on which they reside (in-use or page-out) and placed in the available page queue. The appropriate page table entries are invalidated and the user bit in these entries is turned off. A page-out is not required. In addition, the storage protect key is stored in the associated page table entry.

EXTERNAL PAGE STORAGE MANAGEMENT

External page storage management routines initiate I/O operations on paging devices in response to page-in and page-out requests. They also perform required processing after paging I/O operations terminate. Two sets of queues are used to maintain control over paging operations: the page I/O device queues and the page I/O in-progress queue.

The page I/O device queues are constructed by real storage management routines and they contain page-in and page-out requests. There is a page-in and a page-out queue for each direct access device that contains a page data set. First-in, first-out queuing is used within the page-in queue and the page-out queue for a device. The page-in queue for a paging device has priority over the page-out queue for the device.

All page-in queues have priority over all page-out queues. The page-in queues for paging devices are arranged in priority sequence according to the virtual storage addresses mapped on the paging devices, and are followed by page-out queues arranged in the same priority sequence. That is, the page-in queue for the paging device whose page data set is assigned to the highest addressed paged virtual storage has the highest initiation priority. The page-in queue for the paging device assigned to the lowest addressed paged virtual storage (immediately above the V=R line) has the lowest initiation priority within the page-in queues. The page-out queue for the paging device that is to contain the highest addressed paged virtual storage is next in the total paging queue, etc. When different direct access device types are allocated as paging devices, the highest addressed virtual storage is associated with the fastest direct access device types. Hence, the faster paging devices are associated with the pageable supervisor area and the higher priority partitions, and they have priority over the slower paging devices for the initiation of paging requests.

The page I/O in-progress queue indicates the page-in and page-out requests that have channel programs constructed. Requests are moved from the page I/O device queues to this queue as they are selected and initiated.

The page I/O processor routine initiates paging I/O operations in the sequence indicated by the page I/O device queues. Page-in requests for a given page device with a movable arm have priority over page-out

requests unless the requests can be merged into a single channel program because they refer to the same cylinder.

A nonstandard interface to IOS for page I/O processing is required because of the specialized organization used for the page file. Therefore, the page I/O processor uses a tailored EXCPVR level to initiate paging I/O operations. It constructs its own CCW lists and performs channel program translation using the page device descriptor tables, which are located at the end of the resident control program in the nonpageable area of real storage. Channel programs are constructed such that the time taken for paging I/O operations is minimized. Rotational position sensing is supported when present for the paging device.

The number of requests in a paging channel program varies by paging device type. Up to three page requests are combined in 2314/2319 channel programs to ensure that the channel is not busy for too long when servicing a paging I/O request to this device type. Up to five page requests are chained together in 3330-series channel programs. Up to six page requests are placed in a 2305 Model 2 channel program. Three nondedicated exposures per 2305 volume are used for paging so that three paging channel programs can be active concurrently. For performance reasons, page-out write operations are not verified by reexecution of the CCW list.

When a page-in operation completes successfully, its associated RSPT entry is placed in the level N queue or in fixed status. The reference bit for the real storage page frame is left on but the change bit is turned off. The invalid bit in the page table entry for the virtual storage page whose contents were paged in is turned off. If a permanent I/O error occurs during a page-in I/O operation, the task that required the page-in is abnormally terminated.

When a page-out operation completes successfully, its associated RSPT entry is placed in the available page queue, unless it was reclaimed during the page-out operation or flagged as part of a deferred V=R allocation request. The change bit associated with the page frame is reset. An unsuccessful page-out operation causes the affected page to be long-term fixed so that no further attempts are made to write out the page. (Since virtual storage and external page storage are mapped on a one-to-one basis, no other slots on external page storage are available for allocation to the page.) The invalid bit in the page table entry for the affected virtual storage page is turned off so that address translation can be performed.

Requesting tasks that were placed in the wait state awaiting completion of a page-in operation are made ready after the successful completion of a paging operation.

90:40 RECOVERY MANAGEMENT

RECOVERY MANAGEMENT SUPPORT

The routines included in recovery management support are machine check handler (MCH), channel check handler (CCH), alternate path retry (APR), and dynamic device reconfiguration (DDR). MCH and CCH are standard. APR and DDR routines are optional.

The facilities provided by the MCH and CCH routines are functionally equivalent to those supported by OS MFT RMS routines for System/370 models except for a few new features. MCH routines are structured such that a VS1 control program generated for one System/370 model can be executed on other System/370 models. When MCH recognizes that it is

operating on a model other than the one for which it was generated, error conditions that require processing by model-dependent routines are handled by model-independent routines.

Extensions to recovery processing after real storage errors occur have been made also. When an uncorrectable real storage failure occurs after the IPL procedure has been completed, the MCH routine attempts to isolate the page frame involved and place it in the malfunctioning page queue so that it is not allocated by real storage management. An attempt to recover the contents of the damaged page frame is made. If the page was unchanged prior to the uncorrectable storage error, it is allocated another page frame and paged in again. If the page was changed and it belongs to a user task, the task is abnormally terminated. If the page is changed and it belongs to the system or a system task, recovery procedures are invoked.

CCH, APR, and DDR routines are alike in VS1 and MFT. DDR does not support the swapping of direct access volumes contained on paging devices or spooling (JES) devices in VS1.

OLTEP

OLTEP is a standard feature of VS1 and it supports the same functions provided by MFT OLTEP. OLTEP must operate in nonpaged mode in VS1 to control the execution of OLT's. OLTEP can execute in a paged mode, however, when it is controlling execution of the logout analysis program for a Model 158, 155 II, 168, or 165 II. A pageable partition of 192K minimum is required to execute a logout analysis program under OLTEP.

OLTEP can be invoked via job control or an operator command. It requires a minimum V=R area of 36K when OLT's no larger than 4K are executed. If additional real storage is allocated above the requirement for OLTEP and the particular OLT, it is used to increase OLTEP performance.

In the first release of VS1, OLTEP cannot be executed in systems with less than 160K of real storage. Thereafter, OLTEP can operate in a 144K system but not concurrently with another problem program.

PROBLEM DETERMINATION FACILITIES

Service Aids

The service aids in VS1 are designed to help diagnose a control or problem program failure by gathering information about the cause of the failure, formatting and printing the information in a readily usable form, and aiding in the development and application of an immediate fix for a given problem.

The following service aids are provided, all of which can operate in a pageable partition under VS1 control except IMCOSJQD (which is a standalone program):

- HMAPTFLE is used to apply PTF's to a system. This aid also produces the job control required to apply the fix. Independent component releases of VS1 are supported (not supported in MFT).
- HMBLIST replaces the IMAPTFLS and IMDMDMAP service aids of MFT and produces formatted listings that can be used for system serviceability and diagnostic purposes. It can print the following:

  Formatted load module listings
  Formatted object module listings

Load module map and cross-reference listings
Map and cross-reference listings of the system nucleus
Listings of the data stored in the CSECT Identification records of load modules
Load module map and cross-reference listings showing relocated addresses
Load module summary data including entry point addresses, module attributes, and the contents of the module's system status index
Program modifications to a load module library

- HMASPZAP provides the capability of inspecting and modifying any load module in a partitioned data set (PDS) or any specific data record on a direct access device. It also can be used to dump an entire data set, a specific member in a PDS, or any portion of a data set on a direct access device.

- IMCOSJQD can be used to print the contents of SYS1.SYSJOBQE and scheduler work area data sets (SWADS).

- HMDSADMP is a macro instruction that enables a user to generate a standalone, high-speed or low-speed real storage dump program. The high-speed version writes the contents of the control registers, real storage (including the seven-bit protect key), and, optionally, the page file to tape in large blocks (to be printed by HMDPRDMP), while the low-speed version prints the contents of the control registers and real storage or writes them to tape in unblocked printable format so it can be printed by IEBGENER or HMDPRDMP. The store status function must be performed by the operator prior to loading a standalone dump program.

- HMDPRDMP formats and prints a dump tape produced by a high-speed or low-speed version of HMDSADMP and the trace data gathered by the generalized trace function of GTF. It also can be used to print selected pages from the page file. The VS1 HMDPRDMP service aid formats a dump produced using a VS1 HMDSADMP dump routine only. It will not format a dump produced using a VS2 dump routine.

- IFCDIP00 initializes, reinitializes, and reallocates the SYS1.LOGREC data set, as in MFT.

- IFCEREP0 formats and prints records contained in SYS1.LOGREC and creates a history tape, if desired, as in MFT.

- The Generalized Trace Facility (GTF) supports the same functions in VS1 as those supported by GTF operating under OS MFT or MVT. The full function offered by GTF can be used only in systems with 160K or more of real storage. When executing under VS1 control, GTF uses the hardware monitoring facility and supports tracing of page fault interruptions.

  The generalized trace function of GTF is initiated via a START command. It is a system task and can be executed in a system task or a problem program partition of 64K minimum. Parameters (events to be traced, definition of trace output data set, for example) can be supplied to GTF via the START command or a SYS1.PARMLIB member. During its execution, the trace function requires a minimum of 22K of fixed real storage when trace data is contained in real storage and a minimum of 36K of fixed real storage when the data is written in a trace data set. If additional trace buffers are defined, more real storage is fixed.

  The trace EDIT function of GTF is a part of the HMDPRDMP service aid and is invoked as a problem program via job control. A minimum 64K pageable partition is required for its execution. The trace EDIT function of VS1 will format only the trace data produced in a VS1

environment. It will not format data traced using GTF in VS2, MFT, or MVT environments. However, MFT programs that use the GTRACE macro can be executed under VS1 control without modification. If user-written EDIT exit routines are being used in MFT, they may require modification for operation in a VS1 environment because of differences in the format of trace data for system events.

While GTF and the current MFT resident trace facility coexist in a VS1 control program, only one can be active at a time. GTF disables the trace facility whenever it activates its own tracing function and reenables the trace facility whenever GTF tracing is suspended.

The storage dump facilities available in MVT are also provided in VS1. Real storage and/or the contents of selected areas of virtual storage can be dumped in VS1.

Dynamic Support System (DSS)

The dynamic support system is a general purpose debugging tool that is designed to help locate and temporarily repair a failure in most components of the VS1 control program. DSS uses program event recording hardware in its interface with the operational VS1 operating system. DSS is designed to be used by authorized personnel, such as an IBM FE Programming Systems Representative.

The DSS user can interface with DSS only via a required primary console device type (3210 Model 1, 3215 Model 1, 3066, Model 158 display console) and communicates requests using a DSS language that consists of several commands. Secondary input can be entered via card readers and tape units. The SYS1.DSSVM data set is used to contain such things as DSS language processing routines, the paging data set for DSS, space for the DSS internal dump, and a nucleus swap area.

The DSS user can:

- Display any portion of real storage or virtual storage and any register or system control block during system operation under DSS. Any of the preceding can also be altered except DSS, IPL, NIP, the resident portions of MCH, and interruption handlers.
- Monitor hardware events recognized by the PER feature and certain program events that are detected using the monitoring feature
- Stop the operation of the system at a given point, perform maintenance procedures, and then continue system operation
- Save data (register or real storage contents etc.) accessed during DSS activation on sequential devices for later use

Unauthorized use of DSS must be prevented by installation designed procedures. The primary protection that DSS offers is the fact that only the primary system console device can be used for DSS operations.

90:45 LANGUAGE TRANSLATORS, SERVICE PROGRAMS, AND EMULATORS

SYSTEM ASSEMBLER

The System Assembler is a standard component of VS1 and is the same assembler provided in VS2. It is the only language translator that is a standard component of VS1. Program product and Type I language translators that are to be used with VS1 must be obtained and added to the VS1 system after the VS1 control program desired has been generated.

The System Assembler offers the same functions as OS Assembler F and many enhancements, including improved diagnostics and extended language capabilities. The System Assembler is compatible with OS Assemblers E and F, with a few minor exceptions (see OS/VS System Assembler Language, GC33-4010). Except for its support of certain new System/370 instructions, the System Assembler is a compatible subset of Assembler H.

The System Assembler supports all the new standard and optional System/370 instructions. It is the only OS Assembler that supports the following System/370 instructions:

LOAD REAL ADDRESS	STORE CLOCK COMPARATOR
PURGE TLB	STORE CPU TIMER
RESET REFERENCE BIT	STORE THEN AND SYSTEM MASK
SET CLOCK COMPARATOR	STORE THEN OR SYSTEM MASK
SET CPU TIMER	

The System Assembler is packaged to cause fewer page faults in a paging environment than does Assembler F. The System Assembler can operate in a partition of 64K, however, for more efficient operation, a partition 128K or larger in size is required.

The System Assembler is reentrant. Therefore, it can be made resident in the pageable supervisor area and shared by concurrently executing tasks.

LINKAGE EDITOR

The VS1 Linkage Editor program is a standard component of VS1 (and VS2). It also can operate under OS MFT and MVT. A minimum pageable partition of 64K is required for its operation under VS1; however, a 192K partition is recommended for better performance.

The VS1 Linkage Editor supports the same facilities as OS Linkage Editor F; however, it is designed to operate in a paging environment and it also supports two new features that can be used to reduce the paging and real storage requirements of programs.

One new function supported is control statements to indicate the order in which control sections (CSECTS) and common areas appear in a program (load module). By the reordering of control sections, existing OS MFT programs can be restructured (without a rewrite) for more efficient operation in a paging environment, if desired.

The other new feature of the Linkage Editor is the ability to specify which control sections and common areas of a load module are to be aligned on a page boundary in virtual storage. This new facility, like CSECT reordering, can be used to minimize page faults.

The VS1 Linkage Editor accepts as input all load modules produced by OS Linkage Editors E and F and the object modules that are produced by all OS language translators. Existing job control statements and Linkage Editor E and F control statements are accepted without modification except for the SIZE option.

UTILITIES

The same utilities that are provided in MFT are available in VS1. The IEBCOPY system utility is enhanced to allow a partitioned data set to be unloaded to a removable volume (tape or disk) and later reloaded to the same or a different type disk volume. This utility is to be used during a system generation to place distribution libraries supplied with

the VS1 starter system on direct access volumes. The starter system, therefore, is independent of the direct access devices that will be used during a system generation.

The IEHDASDR utility is modified to place a user-written or user-supplied IPL program on track 0 of an IPL volume, after the required IPL records and volume label(s). This function can be used to place an HMDSADMP dump program on disk so that it need not be IPLed from cards or tape. The disk volumes used to contain any user-written or user-supplied IPL program must have a track size that is large enough to contain the entire IPL program and the IPL records. (The IPL program must be totally contained on track 0.)

INTEGRATED EMULATORS

The functions supported by the integrated emulator programs that operate under VS1 are identical to the functions supported by these emulators when they operate under MFT. These functions are discussed in appropriate system library publications and in Section 40 of the following System/370 guides:

- A Guide to the IBM System/370 Model 135
- A Guide to the IBM System/370 Model 145
- A Guide to the IBM System/370 Model 155, GC20-1729
- A Guide to the IBM System/370 Model 165, GC20-1730

All the integrated emulator programs for VS1 are pageable. The DOS emulator can emulate DOS Version 3 or 4 but does not emulate DOS/Virtual Storage. An emulator program generated to operate on a Model 135, 145, 155, or 165 under OS MFT control can operate on a Model 135, 145, 158/155 II, or 168/165 II, respectively, under VS1 control. Emulator regeneration is not required.

90:50 OS MFT TO OS/VS1 TRANSITION

VS1 is designed to be upward compatible with MFT as of Release 20.1 and, therefore, migration from MFT to VS1 should involve minimal conversion effort. Some additional education of installation personnel is required. For the most part, this involves their becoming knowledgeable about the additional facilities and new environment offered by VS1. System programmers must become acquainted with new interfaces to VS1 (SMF exits and JES reader and writer procedures, for example). Operators must learn how to use the new operater command (WRITER), the new RES commands (if RES is used), and how to respond to new system messages, such as those related to paging and spool devices. Application programmers should learn how to use program structuring techniques that are designed to minimize page faults. System designers must become familar with the factors that affect system performance in a VS1 environment so that the system can be designed and operate in a manner that will achieve the results desired.

Once the VS1 environment to be supported has been determined, a system generation must be performed. A VS1 system control program is generated via a two-stage procedure, in function, much like that required to generate an MFT control program. The system generation macros used to describe the desired control program are identical for MFT and VS1 for like functions. Some of the macros and parameters used in MFT are not required in VS1 while new macros are provided to describe additional or different functions of VS1 (JES and page devices, for example). As in MFT, a complete, nucleus-only, or I/O-device-only generation can be performed. All OS program products and Type I and Type II components that are to be used with the generated VS1 SCP must be added to the VS1 operating system after its generation. Processor

generations for language translators cannot be performed using a VS1 system and must be done using OS MFT or MVT.

The VS1 starter system operates on any System/370 model with the minimum real storage size stated in Section 90:05 that has dynamic address translation, one nine-track tape unit, one SYSIN device, one SYSOUT punch device, one SYSOUT print device, and three 2314/2319 or 3330-series direct access devices. The VS1 starter system can be used to generate only a VS1 control program and is required only for the first generation. Thereafter, an existing VS1 control program can be used. The generated VS1 system can operate on any System/370 Model 135, 145, 158, 155 II, 168, or 165 II that has the hardware features and I/O devices required by the control program. The SECMODS parameter should be specified on the CENPROCS macro at system generation to cause inclusion in the generated VS1 operating system of the model-dependent EREP for the secondary models on which the VS1 control program is to be run, if any.

A new feature of the VS1 generation process is the installation verification procedure (IVP), which is designated to be performed after the VS1 control program is generated. The IVP involves executing an IBM-supplied job stream (maintained in the SYS1.SAMPLIB data set) under control of the generated VS1 operating system. The function of the IVP is to exercise the generated SCP system components to the degree that general operation of the VS1 operating system and support of the system hardware configuration specified is assured.

Existing user-written programs that operate under MFT on a System/370 model must be modified for correct operation under VS1 if they do any of the following. Otherwise, existing user-written executable programs (load modules) can be used without change.

- Reference permanently assigned locations in lower real storage whose contents vary depending on whether BC or EC mode is specified

- Issue the LPSW instruction or directly reference fields in old or new PSW locations whose function or location is affected by which mode, BC or EC, is specified (such as the system mask field and the interruption code field). The MODESET macro should be used to selectively enable or disable the system for interruptions.

- Access SYSIN or SYSOUT data sets using the EXCP macro. BSAM or QSAM must be used. In addition, DSCB's and user labels are not supported for SYSIN and SYSOUT spool data sets.

- Use the trace EDIT exit of GTF, if fields are accessed whose location varies between MFT and VS1

- Depend on a nonstandard interface to the MFT control program. These programs may require modification, based on the specific dependency. (Note that HASP II for MFT depends on interfaces that are changed in VS1 and, thus, MFT HASP II cannot be included in a VS1 operating system. Conversion from an MFT HASP II environment to an OS/VS1 JES environment will require some additional conversion effort, particularly if user modifications to HASP II have been made.)

- Use QTAM to support teleprocessing operations. These programs must be altered to use TCAM since QTAM is not supported in VS1. Minimal effort is required for this modification. (See OS TCAM Programmer's Guide and Reference Manual, GC30-2024, for a discussion of running QTAM application programs under TCAM.)

- Modify an active channel program with data being read (channel program contains self-modifying CCW's) or by executing instructions, if the program is to be run in a pageable partition. Program

modification is not required if such programs operate in nonpaged mode. This situation can apply to programs that use the EXCP macro instead of an access method. Such programs do not operate correctly because the modification affects the virtual channel program rather than the translated channel program that is actually controlling the I/O operation. (See OS/VS Data Management for System Programmers, GC28-0631, for a discussion of how to modify an EXCP program that contains dynamically modified channel programs so that the program can operate in paged mode.)

- Use the EXCP macro and user-written I/O appendages that can encounter a disabled page fault, if the program is to operate in paged mode. These programs do not require modification in order to run in nonpaged mode. These programs can operate in paged mode if they are altered to use the new page fix appendage in order to fix the required pages and avoid disabled page faults.

In addition, the following must be done, if applicable to the existing MFT installation:

- Programs that issue the SET STORAGE KEY (SSK) or the INSERT STORAGE KEY (ISK) instruction should be inspected to determine whether implementation of a seven-bit protect key instead of a five-bit protect key affects the processing being performed. If the INSERT STORAGE KEY instruction is used, it should be used with the understanding that it causes the reference and change bits in the storage protect key to be set also. Alteration of these bits, particularly the change bit, can impair system integrity. Note also that these instructions use real and not virtual storage addresses.

- PL/I F programs compiled using an OS release prior to 20 that use the teleprocessing facilities of this language translator must be recompiled and relink-edited.

- TCAM message control programs and message processing programs must be reasssembled and relink-edited in order to include the coding required for them to operate in a virtual storage environment. Modification of the source statements is not required.

- User-written SMF exit routines should be inspected to determine whether they are affected by SMF record changes.

The job control statements for existing user-written programs do not require alteration except for those that must operate in nonpaged mode. The ADDRSPC=REAL parameter must be added to the appropriate JOB or EXEC statements. If I/O device type changes are made and/or if unsupported device types, such as those listed in Section 90:05, are currently being used in an MFT environment, program and/or job control changes may be required to specify the supported I/O device that is used in a VS1 environment.

Existing data sets can be used without alteration, assuming that device type or access method changes are not made. If VSAM is to be used to replace ISAM, the affected data sets must be converted from ISAM format to VSAM format, as discussed in Section 90:30, and appropriate changes to existing ISAM job control statements must be made.

VS1 does not support System/370 models that are part of an ASP multiprocessing configuration. However, a system under VS1 control and a system under MFT or MVT control can share direct access devices using shared DASD support.

If desired, the structure of existing user-written MFT programs can be modified to minimize the occurrence of page faults and the use of real storage (as discussed in Section 15:15 or 30:15 of the base

publication of which this supplement is a part). Such modification may improve system performance but is not required to enable existing programs (load modules) to operate correctly in a VS1 environment.

For transition from a System/360 MFT environment to a System/370 VS1 environment, the considerations discussed in Section 60 of one of the following publications apply in addition to the preceding discussion:

A Guide to the IBM System/370 Model 135
A Guide to the IBM System/370 Model 145
A Guide to the IBM System/370 Model 155
A Guide to the IBM System/370 Model 165

90:55 SUMMARY OF ADVANTAGES

As a growth system for OS MFT users, OS/VS1 offers many new facilities. Some are changes in the internal structure and organization of the operating system control program to make its operation more efficient. Some new facilities improve operational aspects by simplifying the job of the operator and by reducing causes of total system termination. Others provide functions not available to MFT users. VS1 can be more responsive to a dynamically changing daily workload than MFT, and it supports an environment in which design changes can be made more easily to accommodate maintenance changes and the addition of new functions or applications.

While OS/VS1 supports many new features, including functions exclusive to System/370 (not provided in System/360), such as EC mode and dynamic address translation, VS1 remains upward compatible with MFT. Required control program modifications to handle new features are transparent to the user so that operators and programmers interface with VS1 using basically the same operator commands, job control statements, data sets, and programs they use in an MFT environment.

The single most important new feature of VS1 is its support of a virtual storage environment. The general advantages that can result from using a virtual storage operating system are discussed in the System/370 guide base publication of which this supplement is a part (either in Section 15:05 or 30:05). In addition to these, VS1 offers other specific advantages over MFT, several of which also result from the implementation of virtual storage. These are summarized below.

Improved Job Scheduling

- Small partition scheduling and transient readers and writers are eliminated.
- Job queue contention is reduced (by implementation of SWADS) and a given size job queue can contain more jobs since scheduler control blocks are maintained in SWADS.
- Dedicated work files for initiators are supported and can be used to eliminate allocation and deallocation time for temporary disk data sets.
- A job can be placed in the hold queue during initiation when data sets it requires are not available.
- An initiator can handle up to 15 job classes instead of a maximum of three.
- More readers and writers can be active concurrently (the limitation of three readers and 36 writers is removed).

- All partitions can be of equal size and large enough to contain the largest existing application to enable an application to execute in any available partition when priority is not important. Job class need not be related to partition size. In addition, job priority need not necessarily be associated with partition size so that priority can be assigned on the basis of job characteristics rather than real storage requirements.

- A larger number of partitions can be defined to handle periods during the day when the job queue contains many jobs with relatively small virtual storage requirements, if this situation exists. As long as enough resources are present (I/O devices, available compute time, and real storage), a higher level of multiprogramming can automatically occur during these periods, through proper use of job classes, to cause all available partitions to be used. The system can automatically adjust to the change in the workload without the operator having to intervene to change partition sizes.

Operational Enhancements

- Significant new operator control over system output (SYSOUT data sets) processing is provided by the WRITER command.

- The operator is relieved of most real storage management functions (such as changing partition sizes and altering partition types for the purpose of managing real storage).

- The JES configuration can be modified at IPL (via prior modification of the JESPARMS member) and does not require another system generation.

- The operator need not keep track of reader and writer partitions.

- A reader handling a card SYSIN device remains active when end of file occurs on the card reader (so the operator need not restart the reader from the console each time the card reader runs out of cards).

- High-priority jobs can be handled more easily. A high-priority partition can be established in virtual storage that is used only for these jobs. While this partition requires dedicated virtual storage (and, therefore, external page storage), real storage (except that required for fixed PQA) is required only when a job step is active in the high-priority partition.

- Remote users entering jobs via RES use standard OS commands instead of a special job entry control language (as is used in RJE) and a job can be submitted remotely or locally without changing job control or operator commands.

- The VS1 starter system is independent of the direct access device types to be used during a system generation.

Improved System Integrity and Availability

- More control blocks (specifically, those in a problem program partition) are protected from accidental or intentional modification by a problem program.

- Loss of an ABEND dump because of the lack of available storage in a partition can be eliminated. (Partitions can be made large enough to ensure the availability of enough virtual storage to perform ABEND dump processing.)

- Total system terminations that result from a lack of available SQA space are minimized because SQA is now dynamically expandable and two page frames are held in reserve for allocation to SQA and PQA.

Improved Utilization of Real Storage

- Inefficient use of real storage caused by unused storage within defined partition sizes and/or residence of inactive portions of the program is minimized. Unused virtual storage in a pageable partition does not have real storage assigned, and real storage allocated to inactive pages of a program is released and allocated to active pages when necessary.

- JES and RES are pageable so that during any time interval, they use only the amount of real storage required to handle the current activity. The operator need not perform any function to make real storage assigned to inactive readers or writers available for allocation to other programs.

- The amount of real storage used by reentrant routines (such as SVC's and access methods) made resident in the pageable supervisor area is automatically increased and decreased based on the activity of these routines. The most active modules at any given time will tend to remain resident in real storage without the necessity of preplanning on the part of system designers.

- The amount of storage allocated to SQA dynamically expands and contracts as required. SQA size cannot be varied during processing in MFT.

- Dynamic real storage management is provided for all programs that operate in paged mode in a VS1 environment, regardless of the language in which they are written. Dynamic serial program structure implemented via the use of LINK, LOAD, and XCTL macros and dynamic storage allocation supported via GETMAIN and FREEMAIN macros, all of which are supported by the Assembler Language in MFT, are not supported by all high-level languages.

- The practice of leaving unused real storage between the end of the resident control program and the lowest priority partition in order to leave room for control program expansion can be avoided.

Performance Enhancements

- Job scheduling improvements (as listed previously) are provided

- Improved utilization of real storage (as listed previously) may enable a higher level of multiprogramming to be supported in a given amount of real storage in some environments.

- JES is implemented to provide more efficient data spooling operations. Unit record devices can be operated near rated speeds and intermediate disk storage is allocated and used more efficiently. Less real storage is required for multiple readers and writers.

- Contention for the SVC transient area (and the resulting serialization of processing that can occur) can be minimized by making the most frequently used SVC routines resident in the pageable supervisor area. Task wait time spent waiting to use the SVC transient area is eliminated for these routines.

- SVC area size is increased to 2K (page size) and Type 4 SVC routines are loaded in 2K multiples, instead of 1K, to reduce the time required to load Type 4 SVC routines.

- Improved processing of certain operator commands is provided via use of the pageable 2K SVC transient area.

- Since real storage management is provided by the VS1 control program, problem programmers need not use LOAD, LINK, XCTL, GETMAIN, and FREEMAIN macros in new applications to efficiently manage real storage for partitions and can avoid the control program execution time required to service these requests.

New Features

- VSAM, a new access method designed to provide better performance and more function than ISAM, is provided.

- Expanded system debugging capability is provided by the dynamic support system.

The new facilities of OS/VS1 make it a desirable growth operating system for any MFT user. However, many of the new features of VS1 make it more suited to an online environment than MFT, as follows:

- Reduction of real storage restraints made possible by the implementation of virtual storage can be a significant advantage when designing, coding, and testing online applications that are typically larger and more complex than most batched jobs.

- New functions may be added to existing online applications more easily because the design of a program can be straightforward and need not involve the use of a complex dynamic or planned overlay structure.

- Dynamic storage management is provided automatically by the system, and real storage can be more efficiently used. Storage management no longer need be the major effort in online application design, as it often is in MFT.

- More freedom in program design and better utilization of real storage may enable lower cost entry into online applications processing.

- VSAM is designed to be more suitable than ISAM for an online or a data base environment.

- A system operating with VS1 should be less susceptible to the total termination of operations because of certain improvements made in the VS1 control program. System integrity enhancements have also been made.

- A system with a large online application need not be backed up with a system having the identical amount of real storage. A smaller amount can be used, assuming it provides acceptable performance.

GC20-1752-0

OS/Virtual Storage 1 Features Supplement Printed in U.S.A. GC20-1752-0

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)