GC20-1753-1
File No. S/370-20

Systems

OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement

Release 1.7

This supplement discusses OS/Virtual Storage 2 (OS/VS2) SVS features and organization. Only concepts and functions of OS/VS2 SVS that are new to and significantly different from those of OS MVT are presented in detail. Transition from OS MVT to OS/VS2 SVS is discussed also.

This supplement is an optional section 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/VS2 discussion presented: *A Guide to the IBM System/370 Model 145* (GC20-1734), *A Guide to the IBM System/370 Model 158 for System/370 Model 155 Users* (GC20-1754), *A Guide to the IBM System/370 Model 158 for System/360 Users* (GC20-1781), *A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users* (GC20-1755), or *A Guide to the IBM System/370 Model 168 for System/360 Users* (GC20-1787).

Readers who possess more than one of the above base publications need add this supplement to only one of the documents, as the OS/VS2 information presented applies to System/370 Models 145, 158, and 168 unless otherwise indicated in the text.

The contents of this supplement are designed to acquaint the OS MVT knowledgeable reader with the new facilities and the advantages of OS/VS2 SVS.

IBM

Front Cover (both sides) of GC20-1753-1
Revised September 13, 1976
By TNL GN20-3589

PREFACE

This supplement is stocked in the IBM Distribution Center, Mechanicsburg, as a separate form-numbered item and is not 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 MVT and who require information about OS/VS2 SVS as of Release 1.7 should obtain this supplement and insert it as Section 100 of one of the appropriate base publications listed below.

Base publications for the OS/VS2 SVS supplement are:

- A Guide to the IBM System/370 Model 145 (GC20-1734-2 or later editions)
- A Guide to the IBM System/370 Model 158 for System/370 Model 155 Users (GC20-1754)
- A Guide to the IBM System/370 Model 158 for System/360 Users (GC20-1781)
- A Guide to the IBM System/370 Model 168 for System/370 Model 165 Users (GC20-1755)
- A Guide to the IBM System/370 Model 168 for System/360 Users (GC20-1782)

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/VS2 SVS 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.

Second Edition (March 1974)

This is a major revision obsoleting GC20-1753-0. The text and illustrations have been updated to reflect changes to OS/VS2 SVS since its announcement. These changes are indicated by a vertical line to the left of the change.

This edition applies to Release 1.7 of OS/VS2.

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 systems library publications for OS/VS2 Release 1.7. 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.

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments concerning this publication to: IBM Corporation, Technical Publications/Systems, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. Comments become the property of IBM.

CONTENTS (Section 100)

FIGURES (Section 100)

TABLES (Section 100)

SECTION 100: OS/VIRTUAL STORAGE 2 SINGLE VIRTUAL STORAGE (SVS) FEATURES

100:05 FUNCTIONS AND HARDWARE SUPPORTED

OS/VS2 Single Virtual Storage (SVS) is a growth operating system for OS MVT, large OS MFT, and OS/VS1 installations. OS/VS2 SVS includes features equivalent to and compatible with those of OS MVT and offers major new functions and feature enhancements. The most significant new items of OS/VS2 SVS are:

- Support of one virtual storage of 16,777,216 bytes using dynamic address translation hardware
- Enhancements to job scheduling and the time sharing option, and new functions designed to increase system performance
- An additional access method, called Virtual Storage Access Method (VSAM), that is designed to provide more functions and to be more suitable to online and data base environments than ISAM
- Operational enhancements
- Additional system integrity, reliability, and data security features

OS/VS2 SVS supports one regionalized virtual storage of 16,777,216 bytes with segments of 64K and pages of 4K. OS/VS2 SVS support consists of Releases 1, 1.6, and 1.7 of OS/VS2. Support of multiple virtual storages (MVS) and loosely and tightly coupled multiprocessing configurations is provided in OS/VS2 MVS, which consists of OS/VS2 Releases 2 and up. OS/VS2 MVS is not discussed in this publication.

The organization of virtual storage in OS/VS2 SVS is very similar to that of main storage in MVT. The management of virtual, real, and external page storage and the paging activity of the system are handled entirely by the OS/VS2 control program and are transparent to the programmer.

OS MVT is upward compatible with OS/VS2 SVS to the extent that moving from MVT to OS/VS2 SVS resembles moving from one release of MVT to another that contains significant new features. (See Section 100:50 for a discussion of MVT to OS/VS2 SVS transition.) OS/VS1 (except for the JES and RES facilities) and MFT are upward compatible with OS/VS2 SVS in the same way that OS MFT is upward compatible with OS MVT.

OS/VS2 SVS, classified as system control programming (SCP) and referred to hereafter as VS2 as well as OS/VS2 with or without the SVS or Release 1 designation, supports System/370 Models 145, 155 II, 158, 165 II, and 168 operating in EC and translation modes. VS2 does not support System/370 models operating in EC mode without dynamic address translation specified, System/370 models operating in BC mode, or any System/360 models.

The following minimum system configuration and hardware features are used by VS2 SVS:

- 512K of real storage. (This supports concurrent batched job and TSO operations or dedicated TSO. A minimum batched job system with one reader and one writer operating concurrently with batch job processing can operate in 384K.)

- Byte multiplexer channel with associated I/O devices, including one reader, one punch, one printer, and one console
- One selector or block multiplexer channel with associated I/O devices, including three 3330-series (any models) or four 2314/2319 direct access devices. At least one tape unit (nine-track) and one additional 3330-series drive are required for system generation.
- Dynamic address translation and channel indirect data addressing
- Floating-point arithmetic
- Store and fetch protection
- Time of day clock, CPU timer, and clock comparator
- Monitoring facility
- Program event recording

Tables 100.05.1 and 100.05.2 list the standard and optional features of OS/VS2 SVS, and Table 100.05.3 lists the I/O devices and terminals supported. Just as for an OS MVT operating system, the desired installation-tailored OS/VS2 SVS control program must be generated, at which time user-selected optional features are included in the resulting system. More features are standard in VS2 SVS than in MVT. This can reduce the number of options from which a choice must be made and, thereby, reduce system generation preparation and execution time.

HASP II Version 4 is specifically designed to interface with VS2. Previous versions of HASP II will not operate with VS2, and HASP II Version 4 will not operate with other OS systems (PCP, MFT, MVT, or VS1). Although HASP II Version 4 is not classified as SCP, it has Class A programming service. Optionally, HASP II Version 4 can be used in a VS2 environment to provide data transcription (spooling) and job management functions as well as a remote job entry facility, which is not supported by VS2. When used, HASP II Version 4 supplants VS2 reader interpreter, output writer, and job scheduling facilities.

VS2 SVS support is based on that provided in MVT as of Release 21.8. However, the following MVT features are not available in VS2 SVS:

- Rollout/rollin (function not required in a virtual storage environment)
- SVC and I/O transient areas (all type 3 and 4 SVC's and ERP's are resident in virtual storage)
- Automatic SYSIN batch (ASB) reader and direct SYSOUT (DSO) writers (the functions provided by these facilities are available in HASP II Version 4)
- Scatter loading (function provided automatically)
- Storage hierarchies (2361 Core Storage cannot be attached to any System/370 model)
- Multiprocessing (that is, shared storage multiprocessing support like that provided for System/360 Model 65 multiprocessing configurations)
- TESTRAN
- QTAM (function provided by TCAM), Graphic Job Processor (GJP), and Satellite Graphic Job Processor (SGJP)

- RJE (function provided by HASP II Version 4)
- CRJE (function provided by TSO)
- SER0 and SER1 (replaced by MCH and CCH)
- IEBUPDAT utility (replaced by IEBUPDTE)

This page intentionally blank

- IEHIOSUP (function no longer required because of the new location of SVC routines in VS2)
- IMCJQDMP (replaced by IMCOSJQD)

Table 100.05.1. Standard features of OS/VS2 Release 1 (automatically included during system generation)

- One virtual storage of 16,777,216 bytes with 64K segments and 4K pages*
- Demand paging for allocation of real storage*
- Execution of programs in paged mode* and nonpaged (virtual equals real) mode
- Capability of starting up to 63 initiators*
- I/O load balancing*
- Fetch* and store protection
- Direct access volume serial number verification
- DEB validity checking
- Authorized Program Facility (APF)*
- Multitasking
- PCI fetch
- Advanced overlay supervisor
- Timing facilities
- Extended SVC routing
- Pageable SVC's and ERP's (in the link pack area)
- Pageable modules from SYS1.LPALIB (in the link pack area)
- Pageable BLDL table (in the link pack area)
- Quickcells for faster allocation of certain virtual storage areas*
- Operator communication at IPL
- Multiple Console Support (MCS)
- Hardcopy log
- System log
- Missing interruption checker
- Checkpoint/restart and warm start
- Access methods: QSAM, BSAM, BDAM, BPAM
- System Management Facilities (SMF)
- Error Statistics by Volume (ESV)
- Recovery management: MCH, CCH, APR (when alternate channel paths are specified), DDR
- Dynamic Support System (DSS)*
- Online Test Executive Program (OLTEP)
- Emulator interface (SVC 88)
- System Assembler and Linkage Editor/Loader
- System utilities: IEHDASDR, IEHLIST, IEHMOVE, IEHPROGM, IEHINITT, IEHATLAS, IFHSTATR
- Data set utilities: IEBCOPY, IEBGENER, IEBUPDTE, IEBEDIT, IEBPTPCH, IEBTCRIN, IEBCOMPR, IEBISAM, IEBDG, Access Method Services for VSAM*
- Service aids: AMPTFLE, AMBLIST, AMASPZAP, AMDSADMP, AMDPRDMP, IMCOSJQD, IFCDIP00, IFCEREP0, Generalized Trace Facility (GTF), System Modification Program
- Independent utilities: IBCDMPRS, IBCDASDI, ICAPRTBL

*Facility not available in MVT

Table 100.05.2. Optional features of OS/VS2 Release 1 (must be requested during system generation or added afterward)

- Fixed BLDL table
- Zeroing of a page frame the first time it is allocated to a virtual storage page*
- Automatic Priority Group (APG)*
- Tracing facilities (system trace table and tracing routines)
- Track stacking**
- Automatic Volume Recognition (AVR)
- Device Independent Display Operator Console Support (DIDOCS)
- Expanded device status testing during IPL (DEVSTAT option)
- Time slicing
- Time Sharing Option (TSO)
- Access methods: ISAM, VSAM*, BTAM, TCAM Level 5
- Graphic Programming Services
- Error Volume Analysis (EVA)
- Shared Direct Access Storage Devices (DASD): 2314/2319, 3330-series (all models), and 2305 Models 1 and 2
- Integrated emulators
  1401/1440/1460 for Models 158, 155 II, and 145
  1410/7010 for Models 158, 155 II, and 145
  7070/7074 for Models 168, 165 II, 158, and 155 II
  7080 for Models 168 and 165 II
  709/7090/7094/7094II for Models 168 and 165 II
  DOS Emulator for Models 158, 155 II, and 145
- Reduced error recovery for magnetic tape
- Reliability Data Extractor (RDE)
- Support of the power warning feature for Models 158 and 168

*Facility not available in MVT

**Note that track stacking should not be included in a VS2 Release 1.6 or 1.7 system that contains VSAM when user catalogs are to be utilized as a problem occurs when a user catalog specified in a JOBCAT or STEPCAT statement is opened.

Table 100.05.3. I/O devices, consoles, and terminals supported by OS/VS2 Release 1

<u>Readers and Punches</u>

2501 Card Reader, Models B1 and B2
2520 Card Read Punch, Models B1, B2, B3
2540 Card Read Punch
3505 Card Reader, Models B1 and B2
3525 Card Punch, Models P1, P2, P3

<u>Printers</u>

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

<u>Magnetic and Paper Tape</u>

2401 and 2420 Magnetic Tape Units (and 2816 Switching Units)
3410/3411 Magnetic Tape Units
3420 Magnetic Tape Units, all models
2495 Tape Cartridge Reader
2671 Paper Tape Reader

Table 100.05.3. (continued)

<u>Direct Access Storage</u> (All are supported for system residence, as paging devices, and for SYSIN and SYSOUT data sets, as well as for disk data sets. All channel switching and string switching features for the devices below are supported for alternate channel paths and alternate control unit paths. The rotational position sensing and 32 Drive Expansion features are also supported.) 2314 Direct Access Storage Facility, Models 1, A, and B, and 2844 Auxiliary Storage Control 2319 Disk Storage, A and B models 3330-Series Disk Storage, all models 2305 Fixed Head Storage, Models 1 and 2 <u>Optical and Magnetic Character Readers</u> 1287 Optical Reader 1288 Optical Page Reader 1419 Magnetic Character Reader (Dual Address Adapter and Extended Capability feature required) <u>Display Units</u> (locally attached) 2250 Display Unit 2260 Display Station 3270 Information Display System <u>Consoles</u> 3210 and 3215 Console Printer-Keyboards Display console for Model 158 3066 System Consoles for Models 168 and 165 II 2150 Console with 1052 Printer-Keyboard Model 7 2260 Display Station, Model 1 and 2250 Display Unit, Models 1 and 3 2740 Communication Terminal 3277 Display Station 3213 Printer (for hard-copy output for Model 158 display console only) Composite console (card reader and printer) <u>Terminals (Start/Stop)</u>* 1030 Data Collection System 1050 and 1060 Data Communication Systems 2260, 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 (as a 2740 Communication Terminal with checking)
*Terminals that 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 on programs may have on such terminals.

Table 100.05.3. (continued)

<u>Terminals (Binary Synchronous)</u>
2770 Data Communication System
2780 Data Transmission Terminal
2790 Data Communication System
2792, Models 8 and 11 General Banking Stations
3270 Information Display System
3670 Brokerage Communication System
3735 Programmable Buffered Terminal
3740 Data Entry System
3780 Data Communications Terminal
1130 System (as a processor station)
1800 System (as a processor station - BTAM only)
System/3 (as a processor station)
System/360 Models 20 and up (as a processor station)
System/370 models (as a processor station)
<u>Transmission Control Units</u>
2701 Data Adapter Unit and 2702, 2703 Transmission Controls
2715 Transmission Control Unit
3704, 3705 Communications Controllers (NCP/VS support)
7770-3 Audio Response Unit (TCAM support only, including 2721 Portable Audio Terminal and 2730 Transaction Validation Terminal Model 1 support)

The following I/O devices, some of which are supported by MVT, are not supported by VS2 Release 1:

1017/1018 Paper Tape Reader/Punch
1255 and 1259 Magnetic Character Readers
1442 Card Read Punch, Model N1
1442 Punch, Model N2
2245 Printer
2301 and 2303 Drum Storage
2311 Disk Storage Drive
2321 Data Cell Drive
2402, 2403, 2404 Magnetic Tape Units
2415 Magnetic Tape Unit and Control
2596 Card Read Punch
3881 Optical Mark Reader

100:10 ORGANIZATION AND INITIALIZATION OF STORAGE

VIRTUAL STORAGE ORGANIZATION

The size of virtual storage in a VS2 environment is always 16,777,216 bytes. The organization of virtual storage is reflected in tables and control blocks, similar to those used in MVT, that are established at system initialization and maintained throughout system operation by the control program. Virtual storage is organized, allocated, and freed in VS2 in much the same way as main storage is in MVT. However, in VS2 the virtual storage allocated to pageable programs does not require the allocation of real storage or external page storage until the virtual storage is actually referenced by an executing program.

Virtual storage in VS2, like main storage in MVT, is divided into two main areas: a nondynamic area in lowest- and highest-addressed virtual storage (corresponding to the fixed area in MVT) and a dynamic area between the two nondynamic areas, as shown in Figure 100.10.1. The nondynamic virtual storage area in lowest-addressed virtual storage is nonpageable. The virtual storage in this 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 so that virtual and real storage addresses are equal. The nondynamic area in lowest-addressed virtual storage contains the resident (nonpageable) control program and is a multiple of 4K in size.

"This page intentionally blank."

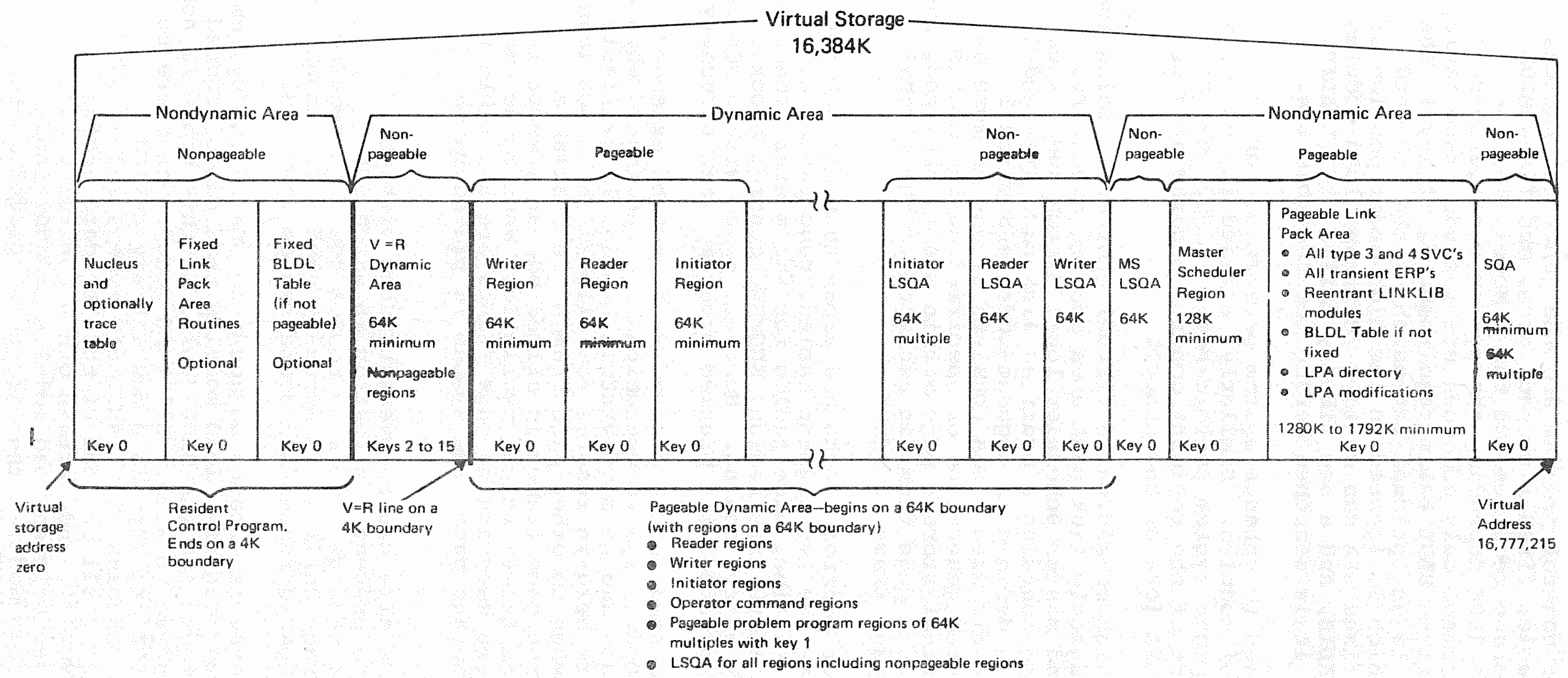


Figure 100.10.1. Virtual storage organization in OS/VS2 Release 1

Included in the resident control program are the generated nucleus and, optionally, a trace table, fixed link pack area, and fixed BLDL table. Typical nucleus storage requirements for a Model 145, for example, range from 138K to 150K depending the system configuration. (This is the nucleus requirement only and does not include trace table, fixed link pack area, and fixed BLDL table requirements, if any.) The fixed link pack area (LPA) is an optional extension of the standard pageable link pack area, which is located in the high-order portion of the nondynamic area. The fixed LPA can contain reentrant load modules from SYS1.SVCLIB, SYS1.LINKLIB, and a new library called SYS1.LPALIB. If a fixed LPA is present, it is searched before the pageable LPA.

A fixed LPA can be defined to enhance system performance or to satisfy time dependencies of modules. Similarly, a fixed instead of a pageable BLDL table may improve system performance. In VS2, the BLDL table is either a fixed part of the resident control program area or is contained in the pageable LPA (one or the other).

The nondynamic area in highest-addressed virtual storage contains the system queue area (SQA), pageable link pack area and its directory, master scheduler region, and master scheduler local system queue area (LSQA). Optionally, it also contains a modified link pack area and a pageable BLDL table. SQA is defined in highest-addressed virtual storage in the upper nondynamic area. SQA consists of one or more 64K segments, as defined at system generation or specified during system initialization. When a virtual storage page within SQA is obtained, the control program ensures that a page frame is assigned to the virtual storage page and that the page frame is fixed.

SQA is used primarily for control blocks and areas that are not job or job-step related, such as the two segment tables, page tables for virtual storage below the V=R line, page tables for the master scheduler region and virtual storage above this region, ENQ/DEQ control blocks, and CDE (contents directory entry) chains. However, space within SQA is allocated for functions associated with problem programs when necessary.

The virtual storage allocated to SQA during IPL cannot be extended during system operation. Hence, system operations are terminated when a request is made for SQA space and no more virtual storage is available in SQA. Two page frames are held in reserve for allocation to SQA (and LSQA, as discussed later) when no other page frames are available. Whenever one of these reserve page frames is allocated, an attempt is made to replace it when page frames become available. When these two reserve page frames have been used and no page frames are available to assign to an SQA virtual storage page, the system is placed in the enabled wait state.

In a VS2 environment, SQA size can be overestimated without performance loss in order to prevent a situation in which SQA is depleted, since the virtual storage allocated to SQA does not require the allocation of real storage until it is actually used. The amount of real storage allocated to SQA increases and decreases as required.

The pageable link pack area is created adjacent to SQA during system initialization. It contains reentrant load modules that can be shared by concurrently executing tasks and the BLDL table, if this table is not made fixed in the lower nondynamic area. All load modules that are to be placed in the pageable LPA are kept in a new library called SYS1.LPALIB. This library contains all transient (type 3 and 4) SVC routines, all transient ERP's, all but one of the standard access methods (BPAM is in SYS1.SVCLIB), and certain other control program routines. Any reentrant user-written load modules or additional reentrant control program load modules that are to be placed in the pageable link pack area must reside in SYS1.LPALIB also. This library can be placed on a disk volume that is demountable. The minimum size of the pageable link pack area varies from 20 segments (1280K) to 28 segments (1792K) depending on the options included in the system.

The following reentrant modules are contained in SYS1.LPALIB and are always loaded into the link pack area (pageable or fixed portion):

- Master scheduler
- Reader interpreter
- Initiator/allocation
- Terminator
- Output writer
- Queue manager
- SMF
- System log
- System task control
- System restart
- Asynchronous overlay supervisor
- GETPOOL
- BSAM, QSAM, BDAM
- All type 3 and 4 SVC's
- Certain TSO modules (when TSO is included in the VS2 system)

The following reentrant modules can be made resident in the link pack area:

- Loader
- All graphics modules and their OPEN/CLOSE routines
- BISAM, QISAM, BTAM, TCAM, VSAM, and OPEN/CLOSE routines for these access methods
- ERP's for BTAM and QTAM

A detailed list of the reentrant modules that can be placed in the link pack area, which also includes their storage requirements, is contained in <u>OS/VS2 Storage Estimates</u> (GC28-0604).

Each load module present in SYS1.LPALIB becomes part of the LPA. A module contained in SYS1.LPALIB is placed in either the pageable or the fixed portion of the LPA. Hence, all transient SVC routines that are not made part of the fixed control program are made resident in virtual storage, and the transient SVC areas used in MVT are not implemented in VS2. The IOS transient area of MVT is also eliminated in VS2 because all ERP's are resident in the LPA in VS2.

Implementation of a link pack area that is pageable and that contains the load modules described offers the following advantages:

- Contention for transient areas, which can occur frequently in MVT, is eliminated without having to specifically reserve additional real storage. Each SVC and ERP routine is allocated its own virtual storage in the LPA. When required, an SVC or an ERP module not currently present in real storage is automatically paged into real storage without the necessity of waiting for a transient area to become available. Also eliminated is CPU time for address constant relocation that the program fetch routine performs when a module is loaded, since these routines are paged in.

- The most frequently used LPA modules in any given time period will tend to remain in real storage because page management is designed to keep the most active pages resident. This eliminates the problem of using the resident reentrant modules option of MVT efficiently, since the VS2 control program is designed to keep the most active modules resident when required without the necessity of measurement and preplanning activities on the part of the system designer. When required, modules that are known to be very heavily used can still be made fixed in real storage by using the fixed LPA option of VS2.

- Less control program time is required to load type 3 and 4 SVC routines into real storage in VS2 than into a transient area in MVT, since transient SVC routines are paged in rather than fetched. Further, routines in the pageable LPA that do not modify themselves (refreshable modules) need not be paged out when they become inactive, and paging I/O time, which is not required in MVT, is not incurred in VS2 for this purpose.

- It becomes more practical to have more user-written code that can be shared resident in the VS2 pageable link pack area than in the MVT link pack area.

The pageable LPA also contains a pageable LPA directory that is created during system initialization. The LPA directory is used to determine where a module resides in LPA virtual storage. A specialized routine is performed (hashing technique using module name) which determines the location of a module name within the LPA directory. This avoids a sequential search and reduces directory search time.

A fixed LPA is also supported. Reentrant modules that are to be placed in the fixed rather than the pageable portion of the LPA are specified in a new parameter list (IEAFIX) that is contained in SYS1.PARMLIB. An IBM-supplied standard list is used unless the operator specifies a user-supplied alternate list during IPL via the new FIX parameter or cancels the fixed LPA facility. Modules placed in the fixed LPA should be those that are heavily enough used that making them fixed instead of pageable results in a significant increase in performance.

The master scheduler region is established in virtual storage below the pageable LPA. It is a minimum of 128K and pageable. Adjacent to this region is the master scheduler LSQA, which is 64K. In VS2, control blocks (such as TCB's), queues, etc., related to a job or a job step are kept in a local system queue area instead of within the region associated with the task. In MVT, LSQA is used only for TSO regions. In VS2, the master scheduler region, each reader region, each writer region, and each initiator started has an LSQA associated with it. LSQA is obtained for an initiator when it is started and released when the initiator is stopped. LSQA also contains the program fetch work area and the CLOSE routines work area that are part of a problem program region in MVT. In VS2, the page tables and external page tables associated with a region are kept in the LSQA for the region.

LSQA is allocated from highest-addressed available virtual storage within the pageable dynamic area. An LSQA must be a multiple of 64K in size. Space within LSQA is obtained on a virtual storage page basis. Whenever a virtual storage page within LSQA is allocated, a page frame is allocated to it and fixed. If a task requires virtual space in LSQA and none is available, or if no real storage is available for allocation to an LSQA virtual storage page, the requesting task is abnormally terminated. If there are two page frames reserved for allocation to SQA, one of them can be allocated to LSQA when no other page frames are available. LSQA will not be allocated the last reserved page frame. The control block space required for the task termination procedure is obtained from SQA. The allocated SQA and LSQA pages for a region are dumped when the SNAP macro includes a request to dump the nucleus.

The advantage of using separate LSQA for each initiator is that the possibility of using all available SQA is significantly reduced, since the control blocks for each individual region are isolated from one another. If a problem program region runs out of LSQA, only that job step need be terminated and system operation continues. In addition, the allocation of real storage for control blocks is more efficient in VS2 than in MVT because, in VS2, only the SQA and LSQA pages actually allocated have real storage assigned. (Note that a minimum of six pages of SQA and three pages of master scheduler LSQA are always fixed in real storage in VS2.) In MVT, when SQA is expanded to meet a large control block space requirement, the expanded SQA area remains allocated even though this larger amount of space may not be required at a later time.

The dynamic area of virtual storage consists of a nonpageable (V=R) area and a pageable area, which are divided by the V=R line. The location of the V=R line is established during system initialization. The address of the V=R line in virtual storage must be a multiple of 4K and can be equal to or less than the address of the end of real storage minus 64K. A minimum V=R dynamic area of 64K is required (to enable OLTEP, a standard facility of VS2, to be executed). A larger V=R dynamic area can be specified at system generation, and this size can be overridden during IPL. The maximum size of the V=R dynamic area cannot exceed real storage size minus 64K bytes.

The V=R dynamic area is used for the execution of nonpageable job steps. A request for a nonpageable region is made using the new ADDRSPC parameter for the JOB and EXEC job control statements. When ADDRSPC=REAL is specified, the REGION parameter indicates the amount of virtual and real storage that is to be allocated to the job step. The virtual and the real storage addresses in a nonpageable region are the same. A job can contain both pageable (ADDRSPC=VIRT) and nonpageable job steps. The default for the ADDRSPC parameter is VIRT.

A nonpageable region is allocated on a 4K boundary. It must be a multiple of 4K and a minimum of 12K in size, plus track stacking requirements, if any. The initiator adds the track stacking requirement to the REGION request. REGION requests for nonpageable regions are rounded to the next 4K boundary when necessary. The maximum size of a nonpageable region is determined by the size of the user-defined V=R dynamic area.

When a nonpageable job step is initiated, enough contiguous virtual and real storage must be available within the V=R dynamic area at that time to satisfy the REGION parameter request. If there is not enough contiguous real storage available because long-term fixed pages are allocated in the V=R dynamic area, scheduling of the nonpageable job step is terminated. Otherwise, the nonpageable job step waits for the required resources to become available. More than one nonpageable job step can be active concurrently (up to a maximum of 14 if storage protect keys 2 to 15 are available), subject to the availability of the contiguous virtual and real storage areas required within the V=R dynamic area.

Jobs that contain one or more nonpageable job steps are initiated using a pageable region in the pageable dynamic area. That is, the initiator uses a pageable region for a work area even though the nonpageable job steps it schedules operate in a nonpageable region in the V=R dynamic area. Nonpageable job steps are terminated using the nonpageable region in which the job step executed.

Nonpageable job steps operate with translation mode specified. This is done because they reference virtual storage addresses, such as those in the pageable LPA, contained outside their nonpageable region. Page tables are established for a nonpageable region so that the real storage address that results from any translation of an address in the program in the nonpageable region is equal to the virtual storage address. Channel program translation is not performed on CCW's contained within a nonpageable region. Checkpoint/restart routines ensure that a nonpageable job step is restarted in the same area within the V=R dynamic area that was used for the checkpoints.

Whenever a LOAD macro is issued by a nonpageable job step, the control program causes the specified routine to be loaded into real storage (if it is not already present), and the module is long-term fixed because a nonpageable job step cannot encounter a page fault. The module remains fixed until a DELETE macro is issued. Since access methods are loaded using a LOAD macro, any access methods used by a nonpageable job step will be long-term fixed during execution of the nonpageable job step and will reduce the amount of real storage available for paging. In addition, when a nonpageable job step issues a LOAD macro, the control program checks to see whether fixing the specified module will cause the limit for fixed real storage to be exceeded. If it will, the nonpageable job step is placed in the wait state until such time as the specified module can be fixed without exceeding the fixed real storage limit.

OLTEP is the only VS2 SCP component that is not part of the resident (fixed) control program and that must operate in nonpaged mode. In addition, in VS2, 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)
- 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 100:25 discusses disabled page faults.)

Existing user-written programs that are operating under MVT control and that must operate in nonpaged mode under VS2 control need not be modified to enable them to run in nonpaged mode. Section 100:50 discusses the job control statement changes that may be required.

The pageable dynamic area consists of all the virtual storage between the V=R line and the nondynamic area in highest-addressed virtual storage. The pageable dynamic area begins on a 64K boundary. Therefore, if the V=R line is not on a 64K boundary, there is unused virtual storage between the V=R line and the beginning of the pageable dynamic area. Reader interpreter regions, initiator regions, output writer regions, pageable problem program regions, operator command regions, and LSQA required for these regions are allocated from the pageable dynamic area. Reader, initiator, and writer regions are a minimum of 64K bytes in size. They are allocated from lowest addressed available virtual storage in the pageable dynamic area. Each reader, each initiator, and each writer is assigned its own LSQA of 64K, which is allocated from highest-addressed available virtual storage in the pageable dynamic area. The START command requires a region of 64K until the started task is initiated. The MOUNT command requires a 64K region until the mounting function has been performed.

Regions for pageable job steps are allocated from the lowest-addressed available virtual storage in the pageable dynamic area that is large enough to satisfy the region space request. If a request for a region larger than the entire pageable dynamic area is received, the job is canceled and the operator is notified. A pageable region is allocated contiguous virtual storage and must be a multiple of 64K in size. It is allocated on a 64K virtual storage boundary. Storage requests on job control statements for pageable regions are rounded up to the next 64K multiple when necessary, which can permit existing job control statements to be used. A pageable region uses the LSQA assigned to the initiator that schedules job steps in the region.

Pageable job steps (as well as nonpageable job steps) are initiated by an initiator operating in a pageable region in the lowest-addressed available virtual storage in the pageable dynamic area. Pageable job steps operate with instruction address translation performed by DAT hardware and channel program translation performed by the control program.

In VS2, up to 63 initiators can be started. The MVT limit of 15 can be extended because of the new method of storage protection implemented in VS2. Protection is accomplished using store and fetch protection hardware and two segment tables instead of one, as follows.

Storage protect keys are associated with virtual storage areas in VS2. When a 4K page frame is allocated to a virtual storage page, its two storage protect keys (one for each 2K block of real storage in the page frame) are set equal to the protect key value associated with that virtual storage page. In VS2, a zero storage protect key value is assigned to all control program areas (resident control program, SQA, LPA, master scheduler region, time sharing control region, reader regions, initiator regions, writer regions, and LSQA). Normally, all pageable background regions have the same protect key value, specifically, protect key value 1. All TSO foreground regions, which must be pageable, are assigned protect key value 1 also. Each nonpageable region is assigned a unique protect key value within the range of 2 to 15. A unique key value should be assigned to a TCAM region to ensure system integrity and security (this is a user responsibility).

If a nonzero protect key is not available for allocation to a job when the initiator attempts to obtain a key, the job is placed in the hold queue and a message is issued to the operator. (If any step within a job is to operate in nonpageable mode, a nonzero protect key is obtained when the job is initiated for assignment to the nonpageable job steps.) If a task requiring a unique protect key is initiated with a START command and no key is available, the START command is rejected. The fetch protect bits in the storage protect keys associated with each type of region are turned on for all assigned nonzero key areas within the region. The fetch protect bit is off in the keys for areas with a protect key of zero.

The program properties table (module IEFSDPPT) can be used to cause assignment of a unique storage protect key to the job steps of a job that is to operate in a pageable region. The names of programs that are to be assigned a unique key must be defined in the program properties table and identified as requiring a unique protect key. (The program properties table can also contain the names of programs that cannot be canceled.) The first step of a job must specify one of these names in order to have a unique protect key assigned to the job. This facility can be used to ensure that a unique key is assigned to a TCAM message control program region. A TCAM program name must be defined in the program properties table. This name must be the program name assigned to the first step of any job that initiates a TCAM message control program in order to acquire a unique key for the TCAM region. The TCAM job step need not be the first step of the job.

There is a system segment table and a user segment table. Both segment tables define the same 16-million-byte virtual storage and both address the same set of page tables. The system segment table has the invalid bit off in all segment entries that define an allocated virtual storage segment. Invalid bits are on for all unallocated virtual storage segments. When a task with a protect key value of zero or a nonpageable job step task is dispatched, the segment table origin control register points to the system segment table. Therefore, a task with a zero protect key has read/write access to all allocated virtual storage. A nonpageable job step task can modify only those areas that are assigned its unique four-bit protect key, (that is, areas within its own region) and can access only its region and all key zero areas (including those in pageable and other nonpageable regions).

The segment table origin control register points to the user segment table whenever a pageable job step task is dispatched. The user segment table has the invalid bit off only for virtual storage segments allocated to the pageable region of the task being dispatched and to the segments allocated to shared control program routines (nucleus, SQA, LSQA, LPA, and the time sharing control region, if TSO is active). The segment table entries for other pageable and nonpageable regions are all marked invalid. Thus, a pageable job step task has read/write access only to its own region (a segment translation exception occurs if another region is addressed) and read-only access to shared control program areas. Storage protection within a region is the same as in MVT. The protection scheme implemented in VS2 provides both fetch and store protection among all regions (except for the fact that nonpageable regions can access key zero areas in both nonpageable and pageable regions). MVT does not support fetch protection.

The organization of space within a region is basically the same in VS2 and MVT. However, subpools are allocated in 4K multiples in VS2, instead of in 2K multiples as in MVT. This increase may have no effect on the region size required by existing MVT programs, since work areas within the region have been restructured and LSQA contains certain areas that are contained in the region in MVT.

REAL STORAGE ORGANIZATION

Real storage in VS2 is also divided into a nondynamic and a dynamic area, as shown in Figure 100.10.2. VS2 Release 1 supports a maximum of 8192K bytes of real storage. The nondynamic area in lowest-addressed real storage is allocated to the lower nondynamic area of virtual storage on a V=R basis. It contains the nonpaged resident control program (nucleus and, optionally, the fixed LPA, fixed BLDL table, and trace table). 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 addresses other virtual storage areas, such as SQA, at various times during its execution and address errors would occur at these times if translation were not operative.

Page frames in the dynamic real storage area are allocated to both pageable and fixed virtual storage pages. An attempt is made to allocate page frames that are located above the address of the V=R line to LSQA, SQA, and other long-term fixed pages. This is done to maintain the capability of executing nonpageable job steps. However, if real storage above the address of the V=R line is not available, an available page frame contained within real storage below the address of the V=R line is allocated to a fixed page. Real storage in the dynamic area is shared by all routines contained in the pageable dynamic area and the upper nondynamic area of virtual storage.

A minimum number of page frames must always be available for paging, and, therefore, cannot be long- or short-term fixed at any given time. The minimum number of page frames that must be available for paging can be specified by the operator during IPL using the PAL parameter (NFX subparameter). This value cannot be specified during system generation. The NFX subparameter can specify a minimum value of 8 and a maximum value of 9999. The default NFX value, which is used if the operator does not supply the NFX subparameter, is 25 percent of the number of page frames between the end of the fixed resident control program in lower real storage and the end of real storage. (There must always be a minimum of 128K of real storage between the resident control program and the end of real storage.) If a fix request is made and the control program determines that honoring the request would cause the fixed real storage limit to be exceeded, the request is not satisfied.

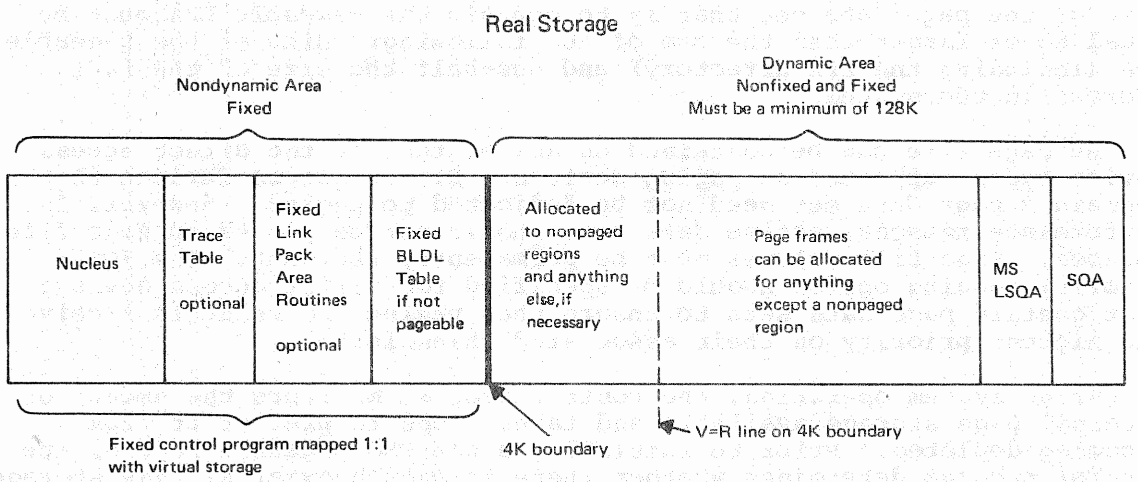

Real Storage
Nondynamic Area
Fixed
Dynamic Area
Nonfixed and Fixed
Must be a minimum of 128K
Nucleus
Trace Table optional
Fixed Link Pack Area Routines optional
Fixed BLDL Table if not pageable
Allocated to nonpaged regions and anything else, if necessary
Page frames can be allocated for anything except a nonpaged region
MS LSQA
SQA
Fixed control program mapped 1:1 with virtual storage
4K boundary
V=R line on 4K boundary

Figure 100.10.2. Real storage organization in OS/VS2 Release 1

The implementation of virtual storage in VS2 eliminates the kind of real storage fragmentation problems encountered in an MVT environment, since it is virtual rather than real storage that is divided into regions. (Real storage fragmentation in VS2 can be caused, however, by the location of long- and short-term fixed pages, which can delay or prevent execution of nonpageable programs.) Fragmentation of virtual storage that may occur in VS2 will not delay system operations until the amount of virtual storage used begins approaching 16 million bytes. In addition, virtual storage that is unused because of fragmentation or that has not been allocated does not cause external page storage to be used inefficiently, since unallocated virtual storage does not have external page storage assigned to it.

As a result of the organization of virtual and real storage in VS2, it should be possible in some environments to start more initiators and to maintain a higher level of multiprogramming (assuming other required hardware resources are available) in a VS2 environment than in a comparable MVT environment.

EXTERNAL PAGE STORAGE ORGANIZATION

External page storage is used to contain the pageable portion of the contents of virtual storage. Each virtual storage page actually used can be written on external page storage during processing, except for the virtual storage pages allocated to the nucleus, SQA, LSQA for non-TSO regions, and the V=R dynamic area. While the control program assumes a virtual storage 16 million bytes in size, the amount of virtual storage that can actually be used is determined by the amount of external page storage provided. TSO swap data sets are part of external page storage in VS2 instead of being separate data sets as in MVT.

The direct access devices supported as paging devices are the 2314/2319, 3330-series (all models), and 2305 Models 1 and 2 (all the direct access devices supported by OS/VS2 Release 1). Paging devices are specified at system generation and/or during IPL. 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 16 page data sets, each of which must be a single extent only and totally contained on one direct access volume. The paging volume that is to contain the contents of the pageable link pack area can be specified at system generation or during IPL. If no such specification is made, the operator is requested to specify a volume using the PAGE parameter. The size of the page data set that is to contain the pageable LPA must be equal to or larger than the sum of the following: size of the pageable LPA (including the LPA directory) and one-half the size of the real storage in the system.

The page file can be contained on any mixture of the direct access device types supported as paging devices. Direct access devices that contain a page data set need not be dedicated to paging. However, for performance reasons, active data sets should not be placed on page file volumes. Page file 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.

During system operation, the control program monitors the amount of external page storage available and takes steps to prevent it from becoming depleted. Prior to initiating a non-TSO pageable region, the control program determines whether there is enough external page storage available to contain the entire contents of the requested region size (one slot available for each virtual storage page in the region). The operator is informed if not enough external page storage is available. The operator can cancel the job or allow the initiator to wait until the required amount of external page storage becomes available. A TSO user is not logged on unless external page storage equal to a user-specified percentage of the TSO region size is available. A message is issued to indicate that the TSO user has been rejected.

An option of dividing paging devices into those that are primary devices and those that are secondary is provided. At least one paging device must be identified as a primary device to contain the contents of the pageable LPA and its directory. When such a division is made, space on the primary paging devices is used for all paging operations until the amount of space remaining on these primary devices falls below a threshold amount. At this time, a migration procedure is entered to make more space available on the primary devices by moving pages from primary paging devices to secondary paging devices.

A region migration selection routine is contained in the task dispatcher. The function of this routine is to identify the pageable region with the lowest dispatching priority so that this region can be migrated. Page management performs the migration procedure, which, for a non-TSO (batch) region, consists of moving all the pages of the selected region that are currently contained on a primary paging device to a secondary device. Once a batch region has been migrated, all paging operations for the region are performed using a secondary paging device until the region terminates.

If a TSO region is selected for migration, the currently executing user is migrated until all the pages on primary paging devices belonging to the TSO user are moved to a secondary device or until a swap-out occurs for the TSO user. Thus, the process of migrating a TSO user can last only as long as one time slice. That portion of the TSO user's region that has been migrated remains on secondary external page storage until the user logs off. Future swap-outs for that TSO user are performed using primary or secondary devices, as indicated in the swap request. If the page device configuration includes a mixture of direct access device types and the slower-speed paging devices are designated as secondary, use of the migration procedure ensures that the faster paging devices are allocated to the higher-priority tasks.

Slot records contained in the page file are 4K bytes in size, and page file tracks are formatted using the track overflow feature. Alternate tracks that have been assigned to defective tracks within page file extents are not used. The space within a page data set on a paging device is divided into a number of groups. There is a given number of tracks within a group, depending on the direct access device type. There is also a given number of slots within each group based on device type. The address of any slot is composed of a paging device number, a group number (within the page data set), and a slot number (within the group).

Regardless of the direct access device type used, page data set tracks are formatted with a space record written after each 4K slot. The size of the space record varies depending on the paging device type. The space records are added to increase paging performance by allowing time for electronic head switching while accessing multiple slots contained within the same cylinder using a command-chained channel program. The organization and capacity of the devices supported for paging in VS2 are shown in Table 100.10.1.

Table 100.10.1. Organization and capacity of paging devices in OS/VS2 Release 1

Device	2305 Model 1	2305 Model 2	3330-series Model 11	3330-series Models 1 and 2	2314 2319
Tracks per group	1	4	1	1	5
Slots per group	3	13	3	3	8
Groups per cylinder	8	2	19	19	4
Capacity per cylinder in bytes	93,304	106,496	233,492	233,472	131,072
Maximum capacity per device in bytes (VTOC on tracks 1 and 2 of cylinder 0)	4,694,016	10,170,368	188,399,616	94,076,928	26,181,632
Maximum number of pages per device	1,146	2,483	45,996	22,968	6,329
Maximum number of groups per device	382	191	15,332	7,656	799

SYSTEM INITIALIZATION

System Parameter Specification

During the initialization of a VS2 operating system, more system parameters can be specified or varied than during an MVT initialization. In addition, a new method by which the operator can supply system parameters during initialization is implemented that reduces the amount of data an operator must enter to provide system parameters. The contents of the SYS1.PARMLIB data set has been modified in VS2.

The following specifications can be supplied during a VS2 initialization in addition to those that can be supplied during an MVT initialization:

- Number of segments in SQA in addition to the one required segment (overrides SQA size specified at system generation) - SQA parameter. If an existing pageable LPA is to be used for this IPL and the size of SQA is increased, the operator is asked to cancel the increase or specify that the existing pageable LPA is not to be used.

- Quickcell area definitions for LSQA and/or SQA (overrides specifications indicated at system generation) - LSQACEL and SQACEL parameters

- Number of master scheduler region segments above the minimum of two (in the majority of systems, the two required segments are sufficient) - MPA parameter

- Indication that an existing pageable LPA is not to be used - CLPA parameter. A new pageable LPA is created, and if an existing LPA is found, it is logically deleted from the page file.

- Amount of virtual storage to be added to the minimum V=R dynamic area of 64K (overrides the system generation specification) - REAL parameter. If NIP determines that the system does not have enough real storage to support the amount of V=R space indicated at system generation or at IPL, the operator is asked to respecify the V=R dynamic area amount.

- BLDL table list that is to be used and whether it is fixed or pageable (overrides the standard IEABLD00 list) - BLDLF or BLDL parameter. The option to have a BLDL table can be canceled.

- Fixed LPA lists (IEAFIXxx) that are to be used (overrides the standard IEAFIX00 list) - FIX parameter. The option to have a fixed LPA can be canceled.

- Modified LPA lists (IEALPAxx) that are to be used (creates a modified LPA) - MLPA parameter

- Page file (external page storage) parameters that specify paging devices or volumes, primary and secondary designations, amount of page file space, and formatting requirements (overrides the system generation specification) - PAGE parameter. The specifications in the PAGE parameter do not totally replace the specifications in SYS1.PARMLIB. If a unit is specified in the PAGE parameter that is not present in SYS1.PARMLIB, the unit is added to the set of paging devices defined in SYS1.PARMLIB. If the unit specified in the page parameter is also defined in SYS1.PARMLIB, the PAGE specifications override the existing specifications. A paging device defined in SYS1.PARMLIB can be deleted using the latter technique. If no paging devices are specified in SYS1.PARMLIB and the operator does not enter a PAGE parameter, a message is issued to request specification of the paging devices.

- Number of channel programs that are to be available for paging I/O operations in addition to the minimum requirement - CPQE parameter. A minimum of 10 is required for one page data set and a minimum of 15 for two or more page data sets. When TSO is present in the system, 80 channel programs in addition to the minimum are made available.

- System parameter lists (IEASYSxx) that are to be used in addition to the standard list (IEASYS00) - SYSP parameter. The standard list cannot be canceled.

- Number of entries in the trace table (nonzero value overrides the value specified at system generation) - TRACE parameter. A zero value cancels the tracing facility.

- Automatic priority group parameters for the dynamic dispatching function (each parameter stated overrides the corresponding parameter stated at system generation) - APG parameter

- Two parameters associated with the time sharing option. The AUXLIST parameter is used to cause certain TSO allocations to be displayed during IPL. The TSOAUX parameter is used to alter one of these specifications.

- Minimum number of page frames that cannot be fixed at any time, certain values used by the page replacement algorithm, and certain threshold values used by the system to determine when task deactivation should occur - PAL parameter. The values specified override system-supplied defaults that cannot be user specified during system generation.

Instead of specifying during processing a tape that is to contain the SYS1.DUMP data set, the operator can specify during system initialization via the DUMP parameter either a tape or a disk unit that is to be used for dumps.

The operator can request that the following be written on the console during system initialization:

- Certain external page storage specifications for TSO

- Page file definitions that exist in SYS1.PARMLIB

- Current page fix limit, page replacement values, and task disable values (that can be modified via the PAL parameter)

- BLDL lists (IEABLD modules)

- Modified LPA lists (IEALPA modules)

- Fixed LPA lists (IEAFIX modules)

- System parameter lists (IEASYS modules)

In VS2, system parameters defined during system generation that can be altered during system initialization are contained in a new member of SYS1.PARMLIB called IEASYS00. SYS1.PARMLIB can contain multiple IEASYSxx members that define different combinations of system parameters, just as this library can contain multiple members that define different BLDL table lists.

In response to the "SPECIFY SYSTEM PARAMETERS" message, the operator can indicate that the parameters specified at system generation are to be used, by pressing the END key, as for MVT. Parameters contained in IEASYS00 (those specified at system generation plus any subsequent modifications) are then used for this IPL. Alternatively, the operator can enter system parameters and/or the new SYSP parameter to specify one or more IEASYSxx members. The system parameters found in the IEASYSxx members indicated are merged with those in the default system parameter list member (IEASYS00) in ascending priority sequence so that parameters in each IEASYSxx member indicated can be overridden by parameters in successive IEASYSxx members specified by the operator. Merging means that the operator-supplied parameter replaces the corresponding parameter as it exists in SYS1.PARMLIB.

Another new parameter that can be included in a system parameter list member (IEASYSxx) is OPI=YES/NO, which indicates whether the operator can override an individual system parameter during system initialization. Any system parameter can be modified by the operator unless OPI=NO has been specified for it. The OPI parameter is effective during the merge of system parameters. The operator can also indicate that one or more parameters in SYS1.PARMLIB are to be canceled for this IPL.

For example, assume the operator enters SYSP=(01,02), BLDL=02, TRACE=, during IPL and the system parameter members contain the following:

```
IEASYS00:    BLDL=00, SQA=2, TRACE=50, TMSL (10,20,OPI=NO)
IEASYS01:    MLPA=(00,01), BLDL=01
IEASYS02:    MLPA=02, TRACE=10, TMSL (5,20)
```

Tracing is canceled for this IPL, and the effective system parameters are:

```
SQA=2, MLPA=02, BLDL=02, TMSL=(10,20)
```

This new approach also allows the operator to press the END key and have the system use an altered standard parameter list since the IEASYS00 member can be modified after system generation. In MVT, the system parameter used when the operator does not enter a different parameter is the system-generation-defined parameter. Redefinition of standard parameters for an MVT system requires another system generation.

The contents of SYS1.PARMLIB are different in MVT and VS2. SYS1.PARMLIB does not contain the resident error recovery procedures list (IEAIGE00), resident access method list (IEAIGG00), or resident SVC list (IEARSV00), which are included in the MVT SYS1.PARMLIB data set. In VS2, SYS1.PARMLIB contains the resident BLDL list (IEABLD00) and link library list (LNKLST00) that are present in MVT as well as several new IBM-supplied standard lists. New user-supplied lists can be added to the VS2 SYS1.PARMLIB data set using the IEBUPDTE utility program.

In addition to the system parameter lists already described (IBM-supplied IEASYS00 list and any user-supplied IEASYSxx lists), the VS2 SYS1.PARMLIB data set contains the following new IBM-supplied standard lists and, optionally, new user-supplied lists:

- IEAFIX00 - This IBM-supplied standard list defines the reentrant modules in SYS1.LINKIB, SYS1.LPALIB, and SYS1.SVCLIB that are to be placed in the fixed (nonpageable) LPA. It contains the names of two modules that handle direct access volume serial number verification for page data set volumes. Optionally, user-supplied alternate (IEAFIXxx) lists can be added to SYS1.PARMLIB that specify reentrant modules from SYS1.LINKLIB, SYS1.LPALIB, SYS1.SVCLIB, and other libraries that are to be placed in the fixed LPA. The standard list (IEAFIX00) is used unless the operator specifies an alternate list during system initialization via the FIX parameter or cancels this facility.

- IEAPAK00 - This IBM-supplied standard list is used during the creation of a pageable LPA when the system is initialized. It identifies the standard modules in SYS1.LPALIB that can be packed together within a 4K page. Each entry in this list identifies a group of two or more modules whose total storage requirement does not exceed 4K bytes. Modules that are most likely to be used together are placed in the same group. This packing list is designed to reduce the amount of virtual storage required for the pageable LPA and reduce paging activity for the modules that are resident in the pageable LPA. User-supplied alternate packing lists are not supported. However, user-defined groups can be added to the IEAPAK00 list. Aliases for module names cannot be specified in this list.

- IEALOD00 - This LPA directory load list indicates frequently used modules in the pageable LPA whose contents supervision directory control block data is to be fixed in real storage during system initialization. Implementation of this load list eliminates a search of the pageable LPA directory for the specified modules, which eliminates the page-in operation that is required if the pageable LPA directory is not present in real storage at the time a module in the pageable LPA is referenced. User-supplied alternate lists are not supported. Modules can be added to the IEALOD00 list.

- IEALPAxx - This optional user-supplied list is used to create the modified LPA. An IEALPAxx list specifies the names of reentrant modules in SYS1.LINKLIB, SYS1.SVCLIB, SYS1.LPALIB, and any user libraries concatenated to SYS1.LINKLIB that are to be made resident in the pageable LPA on a temporary basis, that is, for the duration of this IPL. The specific IEALPAxx list to be used is indicated by the operator during system initialization via the MLPA parameter.

- IEABLDxx - This optional user-supplied alternate list specifies the names of modules in SYS1.LINKLIB, or any data set concatenated to SYS1.LINKLIB, whose directory entries are to be made part of the pageable or fixed BLDL table. If the operator does not specify an IEABLDxx list during system initialization via the BLDL or BLDLF parameter and does not cancel this facility, the IBM-supplied standard list (IEABLD00) is used.

The format of the VS2 SYS1.PARMLIB parameter lists is different from the format used in MVT. The new format affords better direct access space utilization within parameter records and allows more flexibility in parameter definition. In cases in which a parameter list applies to both MVT and VS2 (such as the BLDL table list), both parameter list formats are accepted by VS2 initialization routines. However, the VS2 format for the lists that are common to VS2 and MVT is not valid input to MVT initialization routines.

Initialization of Storage

At the completion of system initialization processing, EC and translation modes are operative. During system initialization, virtual, real, and external page storage are initialized as follows.

Control blocks and tables similar to those used in MVT are built to define a 16-million-byte virtual storage with areas as shown in Figure 100.10.1. Control program modules that are to be made resident in the pageable LPA are allocated virtual storage, fetched into real storage from the appropriate load module libraries, and paged out to the user-specified or default page data set for the pageable LPA. Modules that are part of the modified LPA can be placed on any paging device. The LPA directory is paged out as well. This is the only time the control program forces the page-out of a load module while it is being fetched. When a program is loaded at any other time, it is paged out under the same page replacement algorithm rules that govern page-outs of executing programs. (See the discussion of program fetch in Section 100:25.) The segment tables are initialized to reflect the virtual storage allocated, and page tables are constructed as required for allocated virtual storage segments.

Load modules that are to be made resident in the pageable LPA are fetched from SYS1.LPALIB and placed in the pageable LPA using the standard packing list (IEAPAK00). Modules that are not included in the packing list, such as those larger than 4K in size and those that have been link-edited using the page alignment option, will begin on a page boundary. A pageable LPA directory is built to contain the beginning virtual storage address of each load module. The load list of frequently used routines is also built and placed in real storage, using the IEALOD00 list.

During system initialization, real storage is loaded with the fixed portion of the control program. Load modules from SYS1.SVCLIB, SYS1.LINKLIB, and SYS1.LPALIB that are placed in the fixed LPA are packed in real storage without respect to page boundaries since they will not be paged.

The external page storage configuration is established during system initialization and automatically formatted with slot records, if necessary. If a specified page data set is found to be unavailable or not usable, the operator can respecify a volume or cancel the definition. Once external page storage has been formatted, it need not be reformatted unless a new unformatted volume is used or unless the amount of space allocated to a previously formatted volume is extended. The operator can specifically request formatting. At the completion of system initialization, external page storage contains the contents of the pageable link pack area and its directory. The volumes containing external page storage and the paging devices on which these volumes reside cannot be changed without a re-IPL.

<u>IPL Without Creation of a Pageable Link Pack Area</u>

In an MVT environment, an IPL can be performed with or without formatting the job queue, and the time required to initialize the system is reduced if job queue formatting is not performed. In a VS2 environment, an IPL can be performed with or without formatting the job queue and with or without creating the pageable link pack area in external page storage.

When previously formatted external page storage with an existing usable LPA is available, IPL's can be performed without the necessity of re-creating the pageable LPA in external page storage during each IPL. However, the fixed LPA, the modified pageable LPA, if specified, and the BLDL table (whether fixed or pageable) are re-created during every IPL. A record describing the pageable LPA is created and placed in external page storage whenever the LPA is created in external page storage during system initialization. This record is used during IPL's when LPA creation is not performed. System initialization is performed without LPA creation whenever a previously created LPA is found in external page storage unless the operator specifically requests creation of the LPA (via the CLPA parameter), the record that describes the existing LPA cannot be read, the operator specified the paging volume that is to contain the LPA and it is different from the one that currently contains the CPA, or the size of SQA has been increased by the operator so that it overlaps the high end of the previously defined pageable LPA location.

Temporary modifications to the contents of the pageable LPA can be made during a system initialization that uses an existing LPA in external page storage. Modules can be added or replaced but not deleted. These modifications are effective only for the duration of one IPL and, hence, must be made after each IPL. (SQA size cannot be increased after an IPL for which an existing LPA is to be used.) LPA modifications are specified in the IEALPAxx list in SYS1.PARMLIB. The sequence of searching for a module in the link pack area is the (1) fixed LPA, if any, (2) modified LPA, if any, and (3) pageable LPA. LPA modification is useful, for example, for making effective APAR corrections, SUPERZAP changes, and other modifications, such as the addition of untested user modules, without having to re-create the pageable LPA by refetching all the required modules.

Device Availability Testing

In VS2, device availability testing during system initialization has been improved, and a new DEVSTAT option replaces the non-Type I Smart-NIP option of MVT. In an MVT system without Smart-NIP, only direct access device availability is tested during system initialization. When Smart-NIP is present in MVT, all system-generation-specified device types are tested, and direct access devices without an available path or in unit check status are marked offline. In a VS2 system without the DEVSTAT option, all device types are tested during system initialization, and more precise tape testing is performed than for an MVT system with Smart-NIP. Unexpected and unusual I/O error conditions encountered during availability testing of direct access devices results in the printing of an interpretive I/O error message, and the device is placed in offline status. System operation continues in this case, whereas in MVT such an error results in a system wait state. An online but unlabeled disk volume or an online nondirect access device with a unit address designated for a direct access device type at system generation can cause such errors during availability testing.

The DEVSTAT option essentially provides the same capability for direct access devices as the Smart-Nip option. When it is included, direct access devices with removable volumes and pluggable addresses (2314/2319/3330-series) are set to offline status when a not-operational, unit-check-intervention-required, or other I/O error condition is detected. Without the DEVSTAT option, not-ready status is set for a unit-check-intervention-required condition and offline status is set for a not-operational or other I/O error condition on a 2314/2319/3330-series direct access device.

Missing Interruption Checker

Another facility that is automatically initialized during the system initialization procedure is a routine that checks for missing channel and I/O device end interruptions. The missing-interruption-checker routine, which is standard in VS2, is available only as a program temporary fix (PTF) in MVT. The same functions are provided by this routine in MVT and VS2. This checking feature is designed to lessen the impact on system operation of missing I/O interruptions that result from a hardware malfunction. When the control program expects an I/O interruption that fails to occur, a task, or in some cases the system, enters the wait state. A missing channel or device end interruption can cause a job to be canceled because the allowable wait time for the job is exceeded.

The CPU model- and channel-independent missing-interruption-checker routine operates as a pageable subtask of the master scheduler during system operation. As soon as the master scheduler initialization procedure is complete, the checking module is attached in the master scheduler region. The module performs a polling function on all active nonteleprocessing I/O devices to ensure that device and channel end signals are received within a reasonable amount of time. The operator is notified if an I/O interruption is not received within a time interval that is established when the I/O operation is initiated. The operator is also notified if a MOUNT command is not satisfied in the time interval. The IBM-supplied time interval of three minutes can be changed by the user via modification of the interruption checker module if this interval is not acceptable.

System operation continues after the operator is notified of a missing interruption. The condition may be correctable by the operator (such as when a MOUNT was not satisfied) or a hardware malfunction may have occurred that requires cancelation of the affected job.

100:15 MAJOR COMPONENTS

The major control and processing program components of VS2 Release 1 are shown in Table 100.15.1. Except for the integrated emulator programs, components identified as SCP are distributed as part of VS2. Integrated emulators are distributed separately. Type I programs and program products are not distributed as part of VS2 and must be obtained separately as desired.

The division of control program routines in VS2 and MVT is similar. Both have job, task, data, and recovery management functions. However, VS2 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 virtual storage supervisor of task management.

The new features of VS2 and the most significant functional differences between VS2 and MVT components are presented in the discussions that follow. VS2 uses the same system libraries and data sets as are used in MVT. VS2 also uses two new required libraries and one new required data set: SYS1.LPALIB (already discussed), SYS1.DSSVM, which is described in Section 100:40, and SYS1.PAGE (page file data set). Note that if the Shared DASD option is included in a VS2 system, SYS1.LPALIB and SYS1.PAGE data sets cannot be shared. In addition, in VS2, SYS1.SVCLIB contains only BPAM, transient MCH and CCH modules, TCAM I/O appendage routines, and SVC routines required by NIP. Other modules present in SYS1.SVCLIB in MVT (such as SVC and I/O error routines) and many modules that in MVT are in SYS1.LINKLIB (access methods, etc.) are contained in SYS1.LPALIB in VS2.

VS2 supports all the primary operator console devices required for Models 145, 158, 155 II, 168, and 165 II. The DIDOCS option is required to support display mode operations on the Model 158 display console and the display console contained in the 3066 standalone console unit for Models 168 and 165 II. The 3213 Printer is supported only as a hard-copy output device for the Model 158 display console. It is not supported for input operations. Functionally, the same console support is provided by MCS and DIDOCS in VS2 and MVT.

Table 100.15.1. OS/VS2 Release 1 control and processing program components

OS/VS2 RELEASE 1	
CONTROL PROGRAM COMPONENTS (SCP)	
<u>Job Management</u>	<u>Task Management</u>
• Master scheduler • Reader interpreters and output writers • Job queue management • Job scheduler Initiator Allocation Terminator • System Management Facilities (SMF) • Time sharing option	• Interruption supervisor • Task supervisor • Virtual storage supervisor • Contents supervisor • Overlay supervisor • Time supervisor <u>Page Management</u> • Real storage administration • External page storage administration • Page administration

Table 100.15.1. (continued)

<u>Data Management</u> • Input/output supervisor • Access methods QSAM, BSAM, QISAM, BISAM, BDAM BPAM, BTAM, TCAM, GAM, VSAM • Catalog management • Direct Access Device Space Management (DADSM) • OPEN/CLOSE/EOV • 3704/3705 System Support Programs • 3540 Diskette Copy Programming	<u>Recovery Management</u> • 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	
<u>Language Translators</u> • 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 and BASIC (PP) • VS BASIC • System/7 FORTRAN IV System/370 Host Compiler and Library (PP) • TSO Programs (PP) COBOL Interactive Debug FORTRAN Interactive Debug Assembler, COBOL, and FORTRAN Prompters Data Utilities ITF BASIC AND PL/I VS BASIC	<u>Service Programs</u> • Linkage Editor (SCP) • Loader (SCP) • Utilities System and data set utilities (SCP) Data set utilities with ASCII (PP) • Basic Unformatted Read System (PP) • OS Sort/Merge 5734-SM1 (PP) • OS/VS Sort/Merge 5740-SM1 (PP) <u>Integrated Emulators</u> • DOS Emulator (SCP) • 1401/1440/1460 (SCP) • 1410/7010 (SCP) • 7070/7074 (SCP) • 7080 • 709/7090/7094/7094II (SCP) <u>General</u> • Application-oriented program products (some operate in paged mode and some operate in nonpaged mode).
PROBLEM PROGRAMS - TYPE I AND USER-WRITTEN	
<u>Language Translators</u> • COBOL F (360S-CB-524) • COBOL F Library (360S-LM-525) • COBOL F to ANS COBOL LCP (360C-CV-713) • ANS COBOL Version 2 (360S-CB-545) • ANS COBOL Version 2 Library (360S-LM-546) • FORTRAN G (360S-FO-520) • FORTRAN H (360S-FO-500) • FORTRAN Library (360-LM-501) • FORTRAN Syntax Checker (360-FO-550) • PL/I F (360S-NL-511) • PL/I F Library (360S-LM-512) • PL/I Syntax Checker (360-PL-552)	<u>Service Programs</u> • Sort/Merge (360S-SM-023) <u>General</u> • User-written programs compiled using the Type I language translators listed • User-written programs compiled using program product language translators

100:20 JOB MANAGEMENT

VS2 and MVT job management functions are logically the same and, externally, the VS2 job management interface with the operator is compatible with that of MVT. The internal organization of job management in VS2 and MVT differs somewhat, however. VS2 job management has been modified to operate in a paging environment, and it is designed to offer improvements in performance, reduced real storage requirements, greater reliability, and new functions. Some of the significant new items of VS2 job management are:

- Support of up to 63 initiators
- A new algorithm for allocating I/O devices that is designed to reduce I/O contention within the system (I/O load balancing) and improve performance
- Improved system recovery after error conditions in scheduler tasks (STAE/STAI/STAR processing)

MASTER SCHEDULER

The master scheduler in VS2 is reentrant and pageable. Therefore, it resides in the pageable link pack area. The master scheduler region, which is a minimum of 128K, and master scheduler LSQA of 64K are used for the execution of subtasks of the master scheduler. In VS2, all command processing tasks except START and MOUNT are attached as subtasks of the master scheduler and operate in the master scheduler region. (These commands are CANCEL, DISPLAY, DISPLAY USERS, HOLD, HOLD Q, RELEASE, RELEASE Q, RESET, and SEND.) This reduces the serialization of command processing that can occur in MVT. SMF, the missing interruption checker routine, and system log tasks also operate in the master scheduler region as job step subtasks of the master scheduler. More than two segments can be allocated to the master scheduler region to handle an environment in which concurrent execution of many operator command tasks will occur frequently.

Functionally, the master scheduler in VS2 and MVT are the same. All MVT operator commands and parameters and their formats are accepted in VS2 except those associated with MVT features that are not supported in VS2. No new commands have been added to VS2. The functions of the following commands have been modified:

- CANCEL - A job waiting for a data set, virtual storage, or external page storage can be canceled.
- DISPLAY ACTIVE and MONITOR ACTIVE - For an active job step, the following is given: virtual rather than real storage utilization, the number of LSQA pages allocated, and an indication of whether the job step is operating in paged or nonpaged mode. The number of tasks that a DISPLAY ACTIVE command can handle is increased to 255.
- DUMP - The contents of virtual instead of real storage are dumped, and the ALL parameter is not supported.
- MODE - A simplified format is used that applies to all System/370 models.
- SET - A new parameter, GMT, is added to indicate that the time of day specified in the CLOCK parameter is Greenwich Mean Time instead of local time.
- START - This command can specify up to 15 job classes when an initiator is started, and LSQA allocation can be specified.

Extended environmental recovery is included in subcomponents of the master scheduler to provide increased availability of these critical routines. The master scheduler task, command processing tasks, SMF, and system log task use the facilities of STAE (specify task abnormal exit), STAI (system task ABEND intercept), and STAR (system task ABEND recovery) to effect recovery. More STAE exits are used in VS2 than in MVT in an attempt to reduce abnormal terminations of master scheduler components. The SYS1.DUMP data set is used by STAE exit routines to record diagnostic data pertaining to an error. Recovery in VS2 is also improved by the fact that SMF and the system log task operate as job step subtasks of the master scheduler. The impact of a failure in one of these tasks is confined to the task itself.

READER INTERPRETERS AND OUTPUT WRITERS

The reader interpreter routine and the output writer routine in VS2 are reentrant and pageable. They operate in the pageable link pack area. Each reader and each writer started has LSQA and a pageable region associated with it. The reader/writer region is used for work areas and buffers.

Functionally, the reader interpreter and the output writer in VS2 are the same as their MVT counterparts. Automatic SYSIN batch readers and direct SYSOUT writers are not supported. The fact that VS2 readers and writers are reentrant and pageable offers the advantage of imprcved real storage utilization without operator intervention. Real storage is automatically used only by active readers and writers.

The VS2 interpreter routine is pageable and reentrant and it operates in the pageable link pack area. It accepts all job control statements supported in MVT. New parameters have been added to the job control language for VSAM (see Section 100:30). The only other new job control parameter in VS2 is the ADDRSPC parameter, previously discussed. Processing of the REGION parameter is modified because of the elimination of hierarchy support. The region size allocated when hierarchy parameters are encountered is the sum of that requested for hierarchy 0 and 1, rounded to the next 64K multiple for pageable job steps, or to the next 4K multiple for nonpageable job steps.

The ROLL parameter is ignored. The region space requested by programs that use the rollout/rollin facility to obtain more region space during execution in MVT may have to be increased when these programs execute in a VS2 environment.

JOB SCHEDULER

The components of the job scheduler (initiator, allocation, terminator) are modified to operate in a paging environment and to provide functions not available in MVT. The total real storage requirement for these routines is reduced because they now are reentrant (except for allocation) and they operate in the pageable LPA. The job queue (SYS1.SYSJOBQE) in VS2 is identical in contents and format to the MVT job queue except that the ASB and the RJE queues are omitted.

Initiator

The VS2 initiator is reentrant and pageable. It operates in the pageable link pack area to perform its scheduling function and uses a pageable region for a work area to schedule both pageable and nonpageable job steps. Each initiator has LSQA allocated to it that is used by the job steps it initiates. A VS2 initiator is functionally the same as an MVT initiator except that, in VS2, an initiator can have up to 15 job classes assigned to it instead of 8.

The operator has more opportunity in VS2 than in MVT to cancel a job (either pageable or nonpageable) while it is being scheduled because required resources are unavailable. Specifically, in VS2 the operator can cancel a pageable job after receiving a message indicating that the job is waiting for data sets, region space (virtual storage), or external page storage. (A check is made to determine whether enough external page storage is available to contain the entire contents of the region size requested.) A nonpageable job step can be canceled if it is waiting for data sets or for enough page frames for initiation. In addition, if one or more data sets required by a job are found to be permanently unavailable, the job is canceled automatically by the system.

Allocation

The allocation routine operates as a subroutine of the initiator to allocate I/O devices to job steps, as in MVT. The VS2 allocation routine is pageable and serially reusable. It operates in the pageable link pack area.

The channel load balancing algorithm for nonspecific device requests used by the MVT allocation routine is replaced in VS2 by a new I/O load balancing algorithm that is designed to minimize contention among I/O devices. In MVT, the load on a direct access device is assumed to be directly proportional to the number of data sets allocated to the device. However, because data sets have different activity levels, experience has shown that a count of the number of data sets present does not accurately indicate the load on the device.

In VS2, a new algorithm is used for determining the activity on a tape or a direct access device. The new I/O load balancing algorithm is called by the allocation routine to allocate devices for new tape and disk data sets that do not have specific volume serial numbers indicated in their DD statements (nonspecific device requests). The SEP parameter on DD statements is not effective for new nonspecific tape and disk device requests since the load balancing algorithm is designed to balance the load across the entire configuration. (The algorithm used for allocating a device to an old data set without a specific device request that has not been premounted is the same as that used in MVT.)

The utilization of a tape or a direct access device is determined in VS2 by counting the number of I/O requests (EXCP macros and PCI interruptions) for the device in a given interval. The length of the time interval varies by System/370 model. An exit is taken during I/O supervisor processing in order to accumulate these counts (EXCP rate per device). When I/O devices must be selected for new tape and disk nonspecific device requests, current I/O device and channel utilization is calculated, taking into account the potential load that will be added by the allocation of specifically requested tape and disk devices for the job step. Channel utilization is determined by taking into account EXCP rate, number of allocated data sets, and average EXCP rate per data set for the channel. Device utilization is based on the amount of channel time used (taking into account the average data transfer rate of the device) and, for disk, the number of standalone seeks issued (taking into account average seek time for the device). The device determined to be the best candidate for allocation to a given data set is then selected. If the volume mounted on a selected direct access device does not have enough available tracks to satisfy the space request, the next best candidate is selected. The new I/O load balancing algorithm is also invoked when a nonspecific device request is made by a TSO user.

In order to make most effective use of the new algorithm, the following should be done:

- Public devices should be distributed evenly across channels.
- Public devices should be distributed evenly across control units on the same channel.
- DD statements should be sequenced in the expected order of data set activity (more active before less active).
- Nonspecific volume requests should be made whenever possible.

Terminator

The terminator is pageable and reentrant. It operates in the pageable link pack area. The terminator uses a pageable region as a work area to terminate pageable steps and a nonpageable region to terminate nonpageable job steps. No other functions different from those of MVT are supported by VS2 terminators (except those related to supporting a paging environment).

System Management Facilities (SMF)

SMF is a standard feature of VS2. SMF provides all the same functions in VS2 as it does in MVT and is expanded to include new exits and new accounting data. SMF records (SYS1.MANX and SYS1.MANY data sets) can be written only on disk in VS2. They cannot be written on tape, as in MVT.

The following are the major differences between SMF support in VS2 and MVT:

- Two new exits have been added to SMF in VS2. One new exit (IEFU83) is taken each time an SMF logical record is ready to be written (after formatting has been performed). The return from this exit can indicate the record is not to be written. The other new exit (IEFUJP) is taken at job purge time only when HASP II Version 4 is included in the VS2 system. One SMF exit that is supported in MVT (IEFUSO) is not supported in VS2.
- A new record type 26 (job purge record) is produced for VS2 systems in which HASP II Version 4 is included, as are the new HASP subsystem records 43H, 45H, 47H, and 48H.
- New record types (62, 64, 68, and 69) have been added to VS2 for VSAM accounting data
- Record type 38, produced in MVT, is not produced in VS2. Record types 30 and 41 in VS2 contain the data that is placed in record type 38 in MVT.

The SMF record types and formats produced by SMF routines in VS2 are compatible with those produced in MVT for the most part. However, additional accounting information is supplied in VS2, certain fields have a different meaning, and minor changes to existing fields have been made. For example, in the step termination record the storage-requested and storage-used fields reflect the virtual storage used. If the job step was executed in nonpaged mode, these fields also reflect the real storage used. SMF records that are modified in VS2 are the IPL (type 0), system statistics (type 1), step termination (type 4), job termination (type 5), output writer (type 6), end-of-day (type 12), start time sharing (type 30), and modify time sharing (type 41).

The page supervisor provides the following new data to SMF:

- Number of page-ins per job step (including user and system page-ins) and 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 for the entire system

- Number of reclaimed pages for the entire system

- Number of swaps (swap-ins and swap-outs) that occurred for all TSO job steps

- Total number of pages swapped in for each TSO job step and for all TSO job steps because of time-slice-begins

- Total number of pages swapped out for each TSO job step and for all TSO job steps because of time-slice-ends

- Total number of page migrations from a primary paging device to a secondary paging device for the entire system

- Total number of pages involved in migration from a primary to a secondary paging device per job step and for the entire system

Note that the way in which job step time (CPU utilization) is accumulated for batch (non-TSO) jobs is modified in Release 1.6 of VS2 in order to give more consistent times for the some job step. CPU utilization is accumulated for a job step during the time (1) any user task associated with the job step is executing, (2) any SVC routine (other the those for EXCP and EXCPVR requests) is executing to service a job step request, (3) a user-written SPIE routine is executing, and (4) the MCH routine is processing a machine check that occurs while the job step has CPU control. CPU utilization is not accumulated for a job step during the processing of external interruptions, EXCP and EXCPVR requests, page faults (unless processed by a user-written SPIE routine), monitor call interruptions, and I/O interruptions. The CPU time used by the task dispatcher is not included in job step times either. Variances in job step times can still be caused by hardware factors such as the amount of channel interference or cycle stealing, the number of times virtual storage address translations are found in the TLB, and for Models 155 II to 168, the hit ratio for the high-speed buffer.

HASP II VERSION 4

HASP II Version 4 provides the same spooling and job scheduling functions as the MVT version. In addition, it has been significantly enhanced. In VS2, HASP operates as a system task in a pageable region. Approximately 25% of the total storage required by the generated HASP system is permanently fixed in the HASP region. The other 75% is pageable. The minimum fixed requirement is 12K bytes. HASP II Version 4 code is reentrant.

The following unit record and direct access devices are supported by HASP II Version 4:

- 2501, 2540, and 3505 card readers (80-column cards only) up to a maximum of 99 readers in any combination of types. Column binary reading is supported.

- 2520, 2540, and 3525 card punches up to a maximum of 99 punches in any combination of types

- 1403 and 3211 printers up to a maximum of 99 printers in any combination of types
- 2314/2319, 3330-series (all models), and 2305 Model 1 and 2 direct access devices up to a maximum of 36 devices in any combination of types

The remote job entry facility of HASP II Version 4 supports up to 99 remote work stations on leased (point-to-point) or dialed lines. The following work stations are supported:

- 2770 Data Communication System
- 2780 Data Transmission Terminal
- 3780 Data Communications Terminal
- System/360 Model 20 with BSC adapter (as a multileaving work station)
- System/360 Models 22, 25, 30, 40, 50, 65, 75, 85, and 195 with a BSC adapter (as a multileaving work station)
- 1130 Computing System with a BSC adapter (as a multileaving work station)
- System/3 Model 10 with BSC adapter (as a multileaving work station)

The following list highlights the enhancements that are provided in HASP II Version 4:

- The interface between HASP II and VS2 is improved. Communication between HASP II and VS2 is via a formal interface which, for example, permits a consolidation of HASP II and VS2 console support that provides more operator facilities. The HASP initialization SVC is now unique to HASP.
- The TSO interface is improved. TSO-user control of submitted jobs, including display status and cancel facilities, is now provided.
- SMF recording of HASP accounting information is provided
- Orientation of the print-punch processor is now on a data set basis. For example, multiple copies of output may be requested on a data set basis via a HASP control card, independent destination of output data sets is provided, and automatic control of FCB and UCS loading is provided on a data set basis.
- Restrictions on the number of output data sets permitted for a job are eliminated
- A four-digit job number is now standard
- Operator action requests are saved for display by operator command. Automatic deletion of operator requests from graphic consoles is provided.
- More flexible formatting and redirection of HASP command responses are now provided
- Normal WTO macros are now time stamped and job numbered through the HASP-VS2 console interface
- A more efficient psuedo device selection algorithm is implemented that reduces the requirement for psuedo devices

The following HASP functions have been withdrawn from HASP II Version 4:

- Direct control of local consoles by HASP. This function is replaced by improved support of VS2 consoles
- Execution task monitor functions. These functions are performed by the automatic priority group facility of VS2.
- Remote job entry support of STR (synchronous transmit receive) devices
- Direct tape SYSIN to HASP, which can be accomplished using the HASP internal reader facility

TIME SHARING OPTION

General Description

The time sharing option (TSO) of VS2 provides the same facilities as are offered by TSO in an MVT environment. TSO in VS2 also offers functional enhancements. Hence, TSO operations that are currently performed using MVT can be performed using VS2 with little or no conversion effort. (TSO system parameters may require modification.) A significant advantage that TSO can offer in a VS2 environment is increased performance potential using the same amount of real storage used in an MVT environment. Performance gains may be realized through better utilization of real storage, as is provided in a paging environment, and through enhancements in the way region swapping is handled.

Dedicated TSO operations (no background regions) or concurrent operation of background and foreground (TSO) regions is supported in a minimum of 512K of real storage. One of the new features of TSO in VS2 is support of up to 42 foreground regions, instead of a maximum of 14 as in MVT. A foreground region can be a minimum of 64K in size.

The restrictions on foreground regions are the same in VS2 and MVT. In VS2, the maximum virtual storage size of a foreground region is 896K. Foreground and background regions share the one 16-million-byte virtual storage supported in VS2. In addition, in VS2 a nonpageable job step cannot be executed in a foreground region. Just as is true for background regions in VS2, each foreground region has a separate LSQA associated with it. However, LSQA for a TSO region can be a maximum of 64K in size, of which three or four pages are always fixed. LSQA is not part of a foreground region, as it is in MVT. All foreground regions are assigned protect key 1.

In VS2, multiple users can share concurrently a foreground region and its LSQA, on a time-shared basis, just as in MVT. Prior to the initiation of a time slice for a user, a swap-in is performed (the contents of the user's program are brought into real storage from direct access storage). When the time slice has expired or a long wait condition is encountered, a swap-out occurs (the active contents of the foreground region and LSQA are written out to direct access storage). Another user assigned to the region is then swapped in and given control. In VS2, however, swapping is performed using paging devices, instead of a separate set of swapping devices, and a user's program is paged during its time slice. That is, foreground regions are paged as well as time-shared. After a swap-in occurs, real storage is assigned to the TSO user's program on a demand paged basis.

TCAM Level 5 supports TSO terminals attached to 3704/3705 communications controllers in network control program mode.

Allocation of External Page Storage

External page storage is used for both paging and swapping operations in order to provide more efficient use of auxiliary direct access storage than the use of separate direct access storage for paging and swapping. The allocation scheme used in VS2 permits TSO users to have less external page storage reserved for their regions than is required for background regions because it is assumed that most TSO users will not always require the entire amount of virtual space allocated to the foreground region they use for program execution.

New parameters (TSOAUX, TSOMAX, and BACKUP) have been added to the START TSO and MODIFY TSO commands that enable the operator to indicate how external page storage is to be shared by foreground and background regions. A certain amount of external page storage is required to back up (contain the contents of) the pageable dynamic area of virtual storage (that area between the V=R line and the master scheduler LSQA, as shown later in Figure 100.20.2).

The TSOAUX parameter, which also can be specified at system generation or system initialization, can be used to indicate what percentage of the amount of external page storage required to back up the pageable dynamic area is to be reserved for TSO use only. In effect, this parameter also defines the maximum amount of the external page storage that is available to back up the pageable dynamic area that background regions can use and, thus, limits background region use of virtual storage in the pageable dynamic area. There must be at least 500 slots available for background pageable regions. If the TSOAUX percentage specified would make fewer than 500 slots available, the percentage is reduced. If fewer than 500 slots are contained in the defined external page storage, the TSOAUX parameter is ignored and the operator is notified.

The TSOMAX parameter can be used to specify the maximum percentage of the pageable dynamic area external page storage requirement that TSO regions can use, which also indicates the minimum percentage of external page storage (and, therefore, pageable dynamic area) available for background regions. Any external page storage provided in the page file in excess of the amount required to back up the pageable dynamic area is automatically reserved for TSO use only. The external page storage reserved for TSO is used only for TSO foreground regions. External page storage required to back up the TCAM and the time sharing control regions is taken from that available to background regions. Once the TSOMAX percentage has been allocated to foreground regions, no more LOGON commands are accepted until a LOGOFF command is received and processed. The TSOAUX and TSOMAX parameters guarantee that a minimum amount of external page storage is exclusively available for background region use and another minimum amount is exclusively available for TSO region use. If these parameters are not user-specified, TSOAUX defaults to zero and TSOMAX defaults to 100 percent.

The effect of these parameters is shown in Figure 100.20.1. The total amount of external page storage available to be shared by TSO and background regions is the total amount of external page storage in the page file less the requirement for the nondynamic area in highest-addressed virtual storage. The amount of external page storage required for this nondynamic area is 2 (pageable LPA size + LPA directory size) + master scheduler region size + pageable BLDL size + modified LPA size. The pageable LPA and its directory are backed up by external page storage equal to twice their size because they can contain routines that are reentrant but that modify themselves (are not refreshable). Modified pages are not written in the external page storage that contains the pageable LPA because this would necessitate re-creation of the pageable LPA at the next IPL.

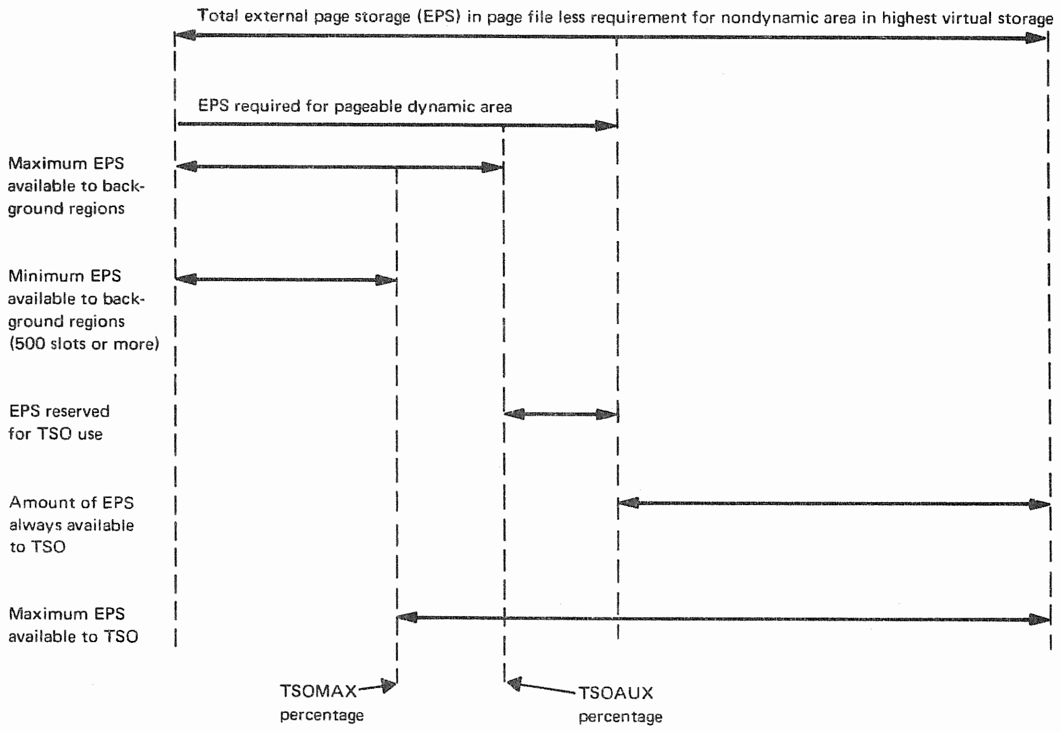

Total external page storage (EPS) in page file less requirement for nondynamic area in highest virtual storage
EPS required for pageable dynamic area
Maximum EPS available to back-ground regions
Minimum EPS available to back-ground regions (500 slots or more)
EPS reserved for TSO use
Amount of EPS always available to TSO
Maximum EPS available to TSO
TSOMAX percentage
TSOAUX percentage

Figure 100.20.1. Division of external page storage when TSO is used

The new BACKUP parameter specifies the percentage of the foreground region size for which slots must be available at each LOGON. Whenever a TSO user attempts to log on to a foreground region, external page storage is inspected to determine whether enough slots are available to contain the BACKUP parameter percentage of the foreground region size. If the required percentage of slots is not available, the TSO user is not logged on. (Background regions are always backed up 100 percent.) If the BACKUP parameter specifies less than 100 percent, it does not mean that TSO users cannot use more than this percentage of their region size. However, the total number of pages actually used by all active TSO regions at any time cannot exceed the total number of slots reserved for allocation to foreground regions.

The new AUXLIST parameter can also be specified via START TSO and MODIFY TSO commands. This parameter indicates what information concerning the use and availability of external page storage is to be listed. The AUXLIST parameter can also be specified during IPL. This parameter causes the following to be listed:

- Total amount of external page storage available
- External page storage available for background regions
- External page storage available for TSO regions
- Current value of TSOAUX

Swapping Procedure

In VS2, swapping is performed somewhat differently than it is in MVT in order to make better use of real storage and to save swapping I/O time. Swapping is also referred to as block paging in VS2. Swapping I/O operations are scheduled and initiated by the page supervisor in response to swapping requests made by the TSO supervisor using the new BLKPAGE macro. There is no code in the VS2 TSO supervisor that handles the actual swapping operation. In MVT, most of a TSO user's region is swapped in and out during each time slice. In VS2, all or a portion of the working set for a region is swapped out. The working set is determined by the addressing pattern of the program as indicated by the reference and change bits at the time the region is to be swapped out.

The working set for a TSO region consists of the LSQA pages for the region and all the pages in real storage at time-slice-end time that have been referenced or changed during the time slice (page frames with a 1,0; 1,1; or 0,1 reference and change bit setting). Page-ins in progress, queued page-in requests, and queued page-out requests for the TSO region are not part of the working set and are purged at swap-out time. The determination of whether to swap out the entire or only a portion of the working set is made based on the type of direct access device, movable head or fixed head, that was used for the previous swap-out. If a movable head device was used for the previous swap-out, the entire working set is swapped out. The slots containing the referenced but unchanged pages that are on the movable head device that was used for the previous swap-out are made available for reassignment. If a fixed head device was used for the previous swap-out, only the changed pages within the working set are swapped out. The referenced but unchanged pages in the working set are left on the fixed head device previously used for the swap-out.

SWAP is another new system parameter for the START TSO command that can be used optionally to request parallel swapping. In VS2, parallel swapping of a foreground region can be done using two, three, or four paging devices. In MVT, only two devices are supported for parallel swapping operations. The paging devices indicated in the SWAP parameter are not dedicated to parallel swapping and are used for normal paging operations as well. If a paging device that is used for parallel swapping becomes full, a paging device is selected using the normal algorithm for paging device selection. If the SWAP parameter is not specified or if the NOSWAP parameter is issued to override a previously specified SWAP parameter, swapping is done serially on one device at a time. If secondary paging devices are specified, the one with the most available slots is selected. If no secondary paging devices are specified or if they are all full, the primary paging device with the most available slots is selected. If a swap-out is to be performed and not enough slots are available to contain the user's working set, the user is logged off and the operator is informed that the page data sets are full.

Whenever a swap-in is to be performed, the paging supervisor schedules LSQA pages to be read into real storage first. If the working set consists of more than 16 pages, the TSO user need not wait for the swapping in of the total working set and is given control to begin executing after the first 16 pages are read into real storage. If the working set consists of fewer than 17 pages, it is completely swapped in before the TSO user gains control to begin executing. The entire working set is always brought into real storage during a swap-in.

In VS2, the region quiescing functions performed before a swap-out occurs include unfixing pages that are currently long- or short-term fixed and modifying the page tables as required. Similarly, during a restore operation after a swap-in, pages must be refixed and page tables must be updated. A TSO user cannot assume that long-term fixed pages will always have the same page frames allocated since the fixed page frames allocated can change from one time slice to the next.

TSO Supervisor

Control of time sharing operations in VS2 is provided by the TSO supervisor, as in MVT. The TSO supervisor operates in the time sharing control (TSC) region with a protect key of zero. The TSC region contains the time sharing control task, time sharing driver, region control task, several resident SVC routines, pageable TSO link pack area, and various control blocks. The TSC region operates in paged mode but contains a certain number of long-term fixed pages. The TSC region and all foreground regions operate with dynamic address translation operative.

The TSC region in VS2 has its own LSQA, which contains a minimum of two fixed pages. This LSQA also contains the TSO LPA directory of modules in the pageable TSO LPA. Frequently used TSO commands or service routines can be made resident in the TSO LPA or the system LPA. The new LPAR parameter can be specified via a START TSO command to indicate the modules that must be contained in the TSO LPA. If the modules indicated are not found in the TSO LPA or SYS1.LINKLIB, the START command is rejected. The LPAF parameter can also be specified via a START TSO command that specifies the LPAR parameter. It indicates the modules that are to be fixed in the TSO LPA for as long as TSO is active.

The functions performed by the TSO supervisor are the same in VS2 and MVT, except that in VS2 the swapping function is handled by page management and operations related to a paging environment must be performed. The same IBM-supplied time sharing driver is provided for both operating systems; however, changes have been made to the TSEVENT Driver Entry Code. Dynamic allocation routines have been repackaged to operate more efficiently in a virtual storage environment. Figure 100.20.2 shows the layout of virtual storage in VS2 when TSO is used.

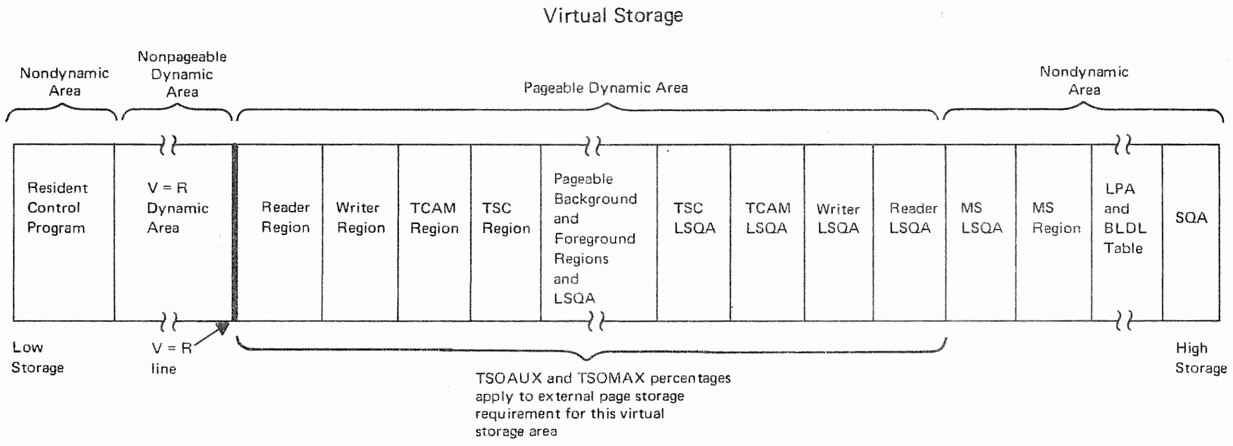


Figure 100.20.2. Virtual storage organization when TSO is used

TSO Commands

The same commands are supported in VS2 TSO as in MVT TSO. However, several TSO commands have been enhanced in VS2. The following new functions are available via enhanced TSO commands in VS2:

- The ALLOCATE command supports specification of space in tracks (TRACK operand) or cylinders (CYLINDER operand), a request to release unused space in a data set when it is closed (RELEASE operand), prompting of the user to free and allocate a file or terminate the command when the specified filename is in use, and the DUMMY operand to allocate a dummy data set.

- The ADD and CHANGE subcommands of the ACCOUNT command provide the ability (via the USERDATA operand) to update installation data fields in the user attribute data set.

- The SEND command can be specified as a subcommand of the EDIT command. The SEND command or subcommand allows a user to execute programs that require subroutines from private libraries (specified via the LIB operand). The RUN command also provides this capability via the LIB operand.

- The OPERATOR command supports (1) an option on the SEND subcommand to have notices placed in the SYS1.BRODCAST data set without being sent to the currently logged-on users, (2) the identifying keyword OPER appended to SEND messages sent by the operator to a terminal user, and (3) the ability to send a message from the SYS1.BRODCAST data set to a user terminal.

- The PROFILE command can be used to list the profile of a user (via the LIST operand).

- The SEND command allows a user to wait for each specified logged-on user to receive the message sent (WAIT operand), allows a user to indicate that a message is not to be sent to logged-on users whose terminals are busy (NOWAIT operand), and permits a message to be sent to the issuer of the SEND command (USER * operand).

System Parameter Keywords

The operator keywords for TSO system parameters are the same in VS2 and MVT except for the following:

- New keywords have been added to the START command - TSOAUX, TSOMAX, BACKUP, AUXLIST, LPAR, LPAF, DUMP (replaces DUMP=DUMP), and NODUMP (replaces DUMP=NODUMP).

- New keywords have been added to the MODIFY command - TSOAUX, TSOMAX, AUXLIST, and BACKUP.

- Certain parameters have been changed. REGSIZE can specify a maximum of 896K and the LSQA parameter is deleted. REGNMAX can specify a maximum of 42. The LPA, DUMP, and NODUMP parameters can be specified on START commands as well as in SYS1.PARMLIB. LIST is no longer a positional parameter.

- Certain parameters are deleted - MAP, FORM, DUMP=DUMP, and DUMP=NODUMP.

All TSO operator keywords in VS2 also have a unique abbreviation, which never contains more than six characters. These abbreviations are not supported in MVT.

Performance in VS2

Performance gains over what is achieved using TSO in an MVT environment can sometimes be obtained in the same amount of real storage in a VS2 environment because more foreground regions can operate concurrently. For example, say an MVT configuration supports a single TSO region with some number of users. Using VS2 in the same configuration, two TSO regions may be able to operate concurrently with half the number of users assigned to each. Better response time might be realized because certain operations of the two TSO regions can be overlapped and fewer users must be swapped in and out per region. Alternatively, some percentage of additional users might be supported in the two TSO regions in VS2 and the response time achieved in MVT could be maintained.

These gains can result primarily from better real storage utilization in a VS2 environment because of demand paging. In MVT, the total amount of real storage dedicated to a TSO region is usually not used 100 percent of the time because of the various sizes of the programs that are executed in the TSO region.

The implementation of virtual storage in VS2 may enable TSO to be added to a VS2 system configuration for a lower real storage cost than adding it to an MVT installation. This cost is even less if TCAM is already in use. TSO can be installed to complement the advantages of virtual storage in that it may be of benefit to programmer productivity when it is used for online program development.

100:25 TASK MANAGEMENT

The VS2 task management routines offer new functions and are designed 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). No significant new functions are provided by the overlay supervisor, checkpoint/restart routines, step restart routines, or warm start routines. (Checkpoint records are always 2K in size in VS2.) Other supervisor routines have been altered to provide new functions, such as a new method of dispatching tasks assigned to the automatic priority group, fetch protection, and support of the CPU timer and the clock comparator. The following identifies the significant functional differences between VS2 and MVT task management routines.

INTERRUPTION SUPERVISOR

Interruption handling is essentially the same in VS2 and MVT; however, additional interruptions are recognized in VS2. Specifically, segment and page translation exception, translation specification exception, monitor call, program event recording, SET SYSTEM MASK (SSM) instruction, clock comparator, and CPU timer interruptions are handled. The CPU is disabled for interruptions from the interval timer at location 80.

The SPIE facility is expanded to allow problem programs to gain control after certain types of translation errors cause an interruption. If a segment translation exception occurs during the execution of a problem program task because the segment entry referenced has its invalid bit on, logically, a storage protection violation has occurred. Similarly, if a page translation interruption occurs because of an error (entry outside the page table is referenced, for example), a protection violation has occurred. The program interruption handler changes the interruption code to that for a protection error in these cases. A store or fetch protection exception can also occur because of a mismatch between the protect key in the current PSW and that in the storage block a task attempts to access. These protection violation error conditions can be handled by a user-written protection error handling routine indicated via the SPIE macro.

When a page fault occurs (invalid bit is on in the page table entry for the referenced virtual storage page), normally, the page supervisor gains control to allocate real storage. However, an authorized program (as determined by the authorized program facility) can indicate in the SPIE macro that its SPIE exit routine is to be entered after a page fault. The authorized program will receive control after both enabled and disabled page faults, and the supervisor lock (described under "Task Supervisor") is not turned on. Therefore, this facility should be used carefully. The data presented to a user-written SPIE routine has the same format in VS2 as in MVT so that SPIE routines that operate in BC mode will operate in EC mode without modification. (Note that the BC mode interface for STAE and STAI exit routines is also preserved and is extended to include status information unique to EC mode.)

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 (PER) is enabled, the dynamic support system (DSS) is entered after a PER interruption. (GTF and DSS are discussed in Section 100:40.)

The interruption supervisor also recognizes an SSM special operation exception that occurs when an SSM instruction is executed. The mask used in this instruction is assumed to be in BC mode format. Control is given to a routine that analyzes the masking requests indicated and puts the system in the requested state using the new MODESET macro. The new supervisor lock (described below) is tested, if necessary. The MODESET macro is implemented in VS2 to be used in place of the SSM instruction. MODESET can be used to request setting of the system mask, alteration of a storage protect key, and the setting of problem program or supervisor state in the PSW. This macro can be issued only by a problem program that is authorized via the authorized program facility. User-written programs that operate in supervisor state or with protect key zero can issue MODESET as well.

TASK SUPERVISOR

Automatic Priority Group

The most significant new feature of the task supervisor is a new method of dispatching tasks that are designated as part of an automatic priority group. This new dispatching method is sometimes called dynamic dispatching or heuristic dispatching, a facility that is not provided in MVT. The VS2 task dispatcher is designed to dispatch a user-designated group of tasks on the basis of their operational characteristics relative to one another, either more CPU oriented or more I/O oriented. The CPU and I/O characteristics of this group of tasks are constantly monitored during their execution, and changes are dynamically taken into account in the dispatching process. Paging I/O is not considered to be part of the I/O requirement of a task. The dynamic dispatcher is designed to improve system performance in a multiprogramming environment by more readily adapting task dispatching to the changing CPU and I/O usage requirements of a group of programs.

Tasks to be dispatched on the basis of the dynamic dispatching algorithm become a part of the automatic priority group (APG). At system generation or system initialization, a single job priority level (0 to 13) can be specified to identify the APG. The priority level selected for the APG cannot be the same as any priority level assigned to a time-sliced group. Tasks that are not part of the APG are dispatched as in MVT, on a priority basis using their system or user-assigned dispatching priority.

For dispatching purposes, the APG tasks are treated as a logical subset of all the existing (system and user) tasks in the system. As shown in Figure 100.25.1, tasks are logically connected in high to low dispatching priority sequence, with APG tasks logically divided into an I/O-oriented subgroup and a CPU-oriented subgroup. The I/O subgroup is positioned within the APG to have higher priority than the CPU subgroup. When the dispatcher is ready to give CPU control to a task, the task queue shown in Figure 100.25.1 is searched from left to right.

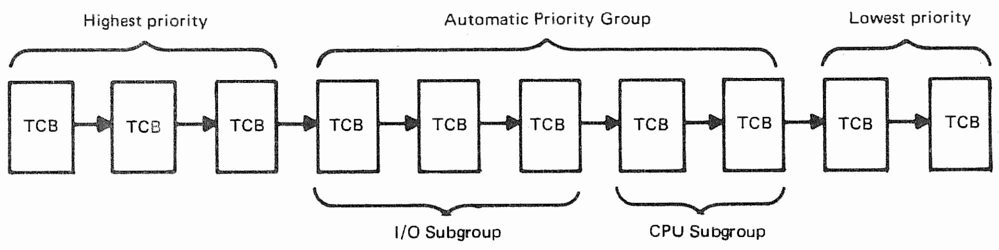


Figure 100.25.1. Task queue containing an automatic priority group

The operating characteristic of each task in the APG is determined by constantly monitoring its use of CPU time. Each time an APG task is dispatched, a time interval is established for the task. The same interval is used for each task during a period of time called the statistics interval (from 1 to 9999 ms). The statistics interval value is user specified at system generation and can be overridden during IPL via the APG parameter. If the entire interval is used (task processing continues until the interval elapses), the task is assumed to be more CPU oriented and is associated with the CPU subgroup. Tasks are positioned in the CPU subgroup so that they are dispatched in a cyclic manner. This is done to ensure that available CPU time is distributed evenly among them and that no task is kept at the end of the CPU-oriented subgroup indefinitely.

If a task does not use its entire time interval, it is assumed to be more I/O oriented and is associated with the I/O subgroup. I/O-oriented tasks are positioned within their subgroup according to the amount of the time interval they used. The smaller the portion of the interval used, the higher a task is placed within the I/O subgroup. When a task enters the APG group of tasks as a result of the issuing of a CHAP or ATTACH macro, the task is placed at the beginning of the I/O subgroup. The task switching rules for APG tasks are summarized in Table 100.25.1.

The dynamic dispatcher is designed to be self-adjusting to ensure that it is accurately differentiating between CPU- and I/O-oriented tasks. At the end of each statistics interval of time, the effectiveness of the time interval currently being used is determined. If the time interval does not adequately distinguish between I/O-oriented and CPU-oriented tasks, the time interval value is increased or decreased (as required) by the user-specified incremental value amount (from 1 to 99 ms). A lower limit (from 1 to 998 ms) and an upper limit (from 2 to 999 ms) for the adjusted time interval are also user specified, as is a ratio value that is used to determine whether the current time interval is effective. These values are specified at system generation and can be overridden during IPL via the APG parameter. The initial time interval value used is the sum of the upper limit and the lower limit divided by 2.

Table 100.25.1. Task switching rules for APG tasks

Reason for loss of control	Current classification	New classification	Actions taken
Task with a higher dispatching priority than that of the APG group becomes ready	I/O oriented	I/O oriented	• Save unused portion of the time interval for use next time this APG task is dispatched • Give CPU control to the higher priority non-APG ready task
Task in the I/O-oriented subgroup or with a higher dispatching priority than that of the APG group becomes ready	CPU oriented	CPU oriented	Same as above
Time interval ends	I/O oriented	CPU oriented	• Establish the full time interval as that to be used next time this APG task is dispatched • Mark this APG task CPU oriented and placed at the beginning of the CPU-oriented subgroup • Search task queue beginning with the first task in the APG group and give CPU control to first ready task found
Time interval ends	CPU oriented	CPU oriented	• Establish the full time interval as the interval to be used next time this APG task is dispatched • Place this APG task at the end of the CPU-oriented subgroup • Search task queue beginning with the first task in the CPU-oriented subgroup of APG tasks and give CPU control to first ready task found
APG task enters wait state voluntarily	I/O oriented	I/O oriented	• Establish the full time interval as that to be used next time this APG task is dispatched • Place this APG task in the I/O-oriented subgroup based on

Table 100.25.1. (continued)

Reason for loss of control	Current classification	New classification	Actions taken
			the percentage of the time interval it used • Search task queue beginning with the first task in the APG group and give CPU control to first ready task found
APG task enters wait state voluntarily	CPU oriented	I/O oriented	• Establish the full time interval as the interval to be used next time this APG task is dispatched • Mark this APG task I/O oriented and place it at the end of the I/O-oriented subgroup • Search task queue beginning with the first task in the CPU-oriented subgroup of APG tasks and give CPU control to first ready task found

The dynamic task dispatching capability in VS2 enables job priority assignments to be more truly related to priority than to system performance. In MVT, job priority assignment is frequently based on CPU and I/O usage to maximize resource utilization and increase system performance. In VS2, jobs that must have a high dispatching priority because a certain response or fast turnaround is required can be assigned a priority higher than that used for APG jobs. Jobs without any special priority requirements can be assigned the APG priority. The dispatcher will attempt to balance CPU usage among these jobs in such a way that CPU and channel resources are efficiently used. Jobs that actually have a low completion priority can be assigned a job priority lower than that of APG jobs.

The task dispatcher has also been modified to handle the following new functions which are discussed, as appropriate, in various subsections of this supplement:

- Support of a new protection scheme, which includes fetch protection as well as store protection

- Selection of a task for deactivation to prevent system thrashing and selection of a task for reactivation

- Selection of a task for migration from a primary to a secondary paging device

- Support of a new supervisor lock in conjunction with the handling of disabled page faults

In addition, the task dispatcher is modified to use the time of day clock instead of the interval timer to determine system wait time.

Authorized Program Facility

The authorized program facility (APF) is a new system integrity feature that is standard in VS2. It is designed to prevent unauthorized programs from performing functions that are designated as restricted.

Programs that are to be authorized via APF must reside in a secure library, that is, SYS1.LPALIB, SYS1.LINKLIB, or SYS1.SVCLIB, which must be password protected in order for APF to effective. Authorized programs are identified at link-edit time via a new PARM field parameter on the EXEC statement or via a linkage editor control statement. The linkage editor places the authorization code (0 or 1) in the directory entry for the program (load module). Code 0 means the program is not authorized to perform a restricted function. Code 1 designates the program as authorized to perform all restricted functions. Critical system functions that are to have access restricted are identified with function code 1, designating them as requiring authorization.

When a job step is initiated, the authorization code of the program fetched from a secure library is placed in the job step control block (JSCB), if a code is present. The JSCB of a program fetched from a nonsecure library indicates the program is not authorized. The TESTAUTH SVC routine is provided to test the JSCB for APF authorization. Whenever any tasks of a job step attempt to use a function marked restricted, the system tests the authorization of the requesting program. If a problem program is not properly authorized, or is authorized but was not contained in SYS1.LPALIB, SYS1.LINKLIB, or SYS1.SVCLIB, the problem program is abnormally terminated.

A user-written routine can be restricted to access only by APF authorized routines in one of two ways. The restricted routine can include a procedure to test for APF authorization in the JFCB using the TESTAUTH macro, or the routine can be assigned the authorized attribute of 1 during link-editing and placed in a secure library, which includes libraries concatenated to SYS1.LINKLIB. Note that a task that is assigned protect key zero or that operates in supervisor state is also authorized to access restricted functions.

User-written programs that use the following macros and programs must be authorized via APF: CVOL, DASDR, MODESET, PGFREE, PGFIX, PGLOAD, and EXCPVR macros and IEHDASDR, IEHATLAS, IEHPROGM, AMASPZAP, OLTEP, and DDR programs.

Supervisor Lock

A supervisor lock is implemented in the task supervisor to ensure proper system operation when a disabled page fault occurs. In VS2, 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 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 page-in completion I/O interruption to be presented) can cause code that operates with the CPU disabled to be reentered, with improper processing the result.

To prevent this situation, a supervisor lock is implemented in VS2 that can be set on (locked) or off (unlocked). Supervisor code is included to set and test the lock as required. When a disabled page fault occurs in an executing task, the supervisor lock is turned on and identified as belonging to the task that caused the disabled page fault. As long as the supervisor lock is on, no code that operates with the CPU disabled for interruptions can be executed except code that is related to paging or basic dispatching operations. However, CPU control is given to another ready task that is to operate with the CPU enabled. The lock remains on until the disabled page fault is resolved, and only tasks that execute with the CPU enabled and paging tasks can execute until the lock is turned off. Code is included within the control program to recognize an attempt made by a task to disable the CPU for interruptions by executing an SSM instruction or a MODESET macro and to place such a task in the wait state when the supervisor lock is on.

Certain resident control program routines (IOS, page supervisor, task dispatching routines, for example) are structured to avoid disabled page faults in VS2. Most type 1 SVC routines that potentially could cause a disabled page fault have been converted to type 2 SVC routines. Type 2 SVC routines have been modified to address all the pages they require before they disable the CPU for interruptions. This causes any nonresident pages to be loaded before processing begins with the CPU in a disabled state. In VS2, all type 1 SVC routines receive control with the CPU disabled for interruptions, and type 2, 3, and 4 SVC routines are given control with the CPU enabled or disabled, depending on the indication in the SVC table. (In MVT, all type 1 and 2 SVC's are entered with the CPU disabled.) User-written type 1 and type 2 SVC's that are to be added to a VS2 control program should also avoid disabled page faults.

The lock approach implemented in VS2 has the advantage of allowing routines to encounter disabled page faults when required, in order to avoid fixing a large number of pages. The approach used also avoids delaying total system operation while a disabled page fault condition is handled.

DEB Validity Checking

A more comprehensive method of ensuring that a task cannot access a data set associated with another task is provided in VS2 via implementation of an expanded DEB (data extent block) validity checking scheme. A new DEBCHK macro and SVC routine are provided to support DEB validity checking. The DEBCHK macro is designed to be used by control program routines that modify a DEB or that use or modify a control block that is located via accessing the DEB. The DEBCHK macro can also be used by system programmers (most options of this macro require the issuing task to be operating in supervisor state).

Routines that currently perform DEB validity checking in MVT, such as OPEN and CLOSE, are modified to use the DEBCHK macro for DEB processing. In VS2, however, the I/O supervisor (IOS) is also modified to issue the DEBCHK macro each time a DEB is passed to it via an EXCP macro to determine whether the DEB is associated with the task that issued the EXCP. (The DEB validity check routine ensures that the specified DEB is in the DEB table for the task.) A user task is abnormally terminated if the DEB validity checking routine finds the DEB to be invalid in any way.

VIRTUAL STORAGE SUPERVISOR

The virtual storage supervisor is responsible for allocating and deallocating virtual storage in response to user (GETMAIN and FREEMAIN) requests for storage and system requests for storage other than for LSQA and SQA. Except for V=R requests, real storage is not assigned to allocated virtual storage until the virtual storage is referenced during processing. If a task exhausts the virtual storage available in its region, it is abnormally terminated. The virtual storage supervisor is functionally equivalent to the main storage supervisor in MVT except for the following modifications:

- Use of LSQA to store region-related control information
- Allocation of 4K instead of 2K areas to subpools in a region
- An interface with the page table create and destroy routine that is part of page management. When a virtual storage area (region, LSQA, etc.) is allocated, this routine is called by the virtual storage supervisor to create and initialize the required page tables and external page tables and to modify the two segment tables as required. When a virtual storage area is freed, this routine destroys the associated page tables and external page tables and invalidates the appropriate entries in the segment tables.
- Expansion of the GETMAIN macro to request allocation of virtual storage on a page boundary. Also, when a request for storage contains hierarchy parameters, the storage allocated is the sum of that requested in hierarchy 0 and hierarchy 1, rounded up as appropriate. GETMAIN requests in VS2 are satisfied on a best-fit rather than a first-fit basis, as in MVT. This is done to pack allocated virtual storage within the fewest number of virtual storage pages.
- Implementation of a new quickcell facility for handling certain allocation requests for virtual storage in SQA and LSQA. This facility is designed to reduce the amount of time required to service a GETMAIN request for a relatively small amount of space (8 to 256 bytes) in SQA and LSQA that will be allocated for a short duration. Since these types of requests are grouped together, storage fragmentation within LSQA and SQA is reduced also.

  A quickcell area no larger than 4096 bytes is established in SQA and in each LSQA whenever these areas are created. An individual quickcell can be a multiple of 8 bytes in size up to a maximum of 256 bytes. A maximum of eight quickcells can be specified for a given quickcell length. The number of quickcells to be allocated for each size and the size of the total quickcell area in SQA and LSQA are specified during system generation, and these values can be overridden during IPL. The quickcell area is allocated real storage when it is created, and this real storage remains allocated for as long as the SQA or the LSQA exists.

  Allocation requests for space in the quickcell area are made via a special branch. Requests for space of 8 to 256 bytes (in subpool 245 in SQA and in subpools 235 and 255 in LSQA) are satisfied from the quickcell area. If a request cannot be satisfied, normal GETMAIN logic is used.

CONTENTS SUPERVISOR

Contents supervision in VS2 is functionally equivalent to that provided in MVT except that scatter loading and hierarchy support, facilities that are not required in VS2, are not supported. Attributes associated with these functions are ignored. In addition, contents supervision has been modified as appropriate to support APF, use a new LPA directory search technique, and check for LOAD macro requests issued by nonpageable job steps, all of which have been discussed previously.

Program Fetch

In VS2, load modules have a zero starting address and are stored in partitioned data sets in the same format that is used in MVT. Hence, when a load module is fetched in VS2, 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 MVT.

The PCI fetch routine used in MVT is modified for operation in a VS2 environment. Support of storage hierarchies and scatter loading is removed and the fetch routine is altered to operate in a paging environment. One new function PCI fetch provides is the loading of load module control sections on page boundaries (see the linkage editor discussion in Section 100:45).

PCI fetch uses the new EXCPVR macro (discussed in Section 100:30) instead of EXCP. In VS2, PCI fetch requests the allocation and fixing of up to five page frames for the execution of each read operation (START I/O instruction) when the size of the load module is greater than 16K. 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 actually used during the read 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 MVT.

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 occupied by any recently loaded pages is required for allocation to other pages, and the page supervisor considers these pages to be eligible 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 that was recently loaded) 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 (refreshable pages).

Note that in OS there is a distinction between a refreshable module and a reentrant module. A refreshable module is one that is never modified, and such a module can be used by concurrently executing tasks. A reentrant module is one that can be used by concurrently executing tasks but that may modify itself during execution. A module can modify itself and still be sharable if the module prevents task switching during the time it is in a changed state (disables the CPU for I/O and external interruptions, makes a change, changes altered data to its original value, and reenables the CPU for I/O and external interruptions). Refreshable modules will be paged out only once, since their change bits will never be turned on during execution of the module. Pages of reentrant modules that are changed during their execution will have their change bit turned on. These pages will be paged out if they become inactive and their page frames are needed and taken for reassignment to other pages.

TIMER SUPERVISOR

The timer supervisor in VS2 uses the time of day clock and the new CPU timer and clock comparator to provide timing facilities. The interval timer at location 80, which is supported by MVT, is not used.

Timing facilities identical to those provided in MVT are supported in VS2. However, in VS2 the STIMER macro has been expanded to allow an interval of time to be specified in terms of microseconds (the resolution of the CPU timer), the TTIMER macro is expanded to request the amount of time remaining in an interval in terms of microseconds, Greenwich Mean Time is used in the time of day clock, and the internal logic of the timer handling routines has been altered to give better performance.

In VS2, the time in the time of day clock is Greenwich Mean Time (GMT) with a base of January 1, 1900 instead of local time. During system generation, the time zone differential, east or west of GMT, can be specified so that the system communicates with the operator using local time instead of GMT. The time zone differential can be modified after system generation by changing the appropriate member of SYS1.PARMLIB. When the operator enters the time of day, absence of the new GMT subparameter for the CLOCK parameter indicates that the time specified is local. This causes the control program to convert the local time to GMT, using the time zone differential, before placing the time in the time of day clock.

Reductions in the amount of code required to handle timing queues are made possible by the common doubleword format of the clock comparator, CPU timer, and time of day clock. The higher resolution of the new hardware clocks (one microsecond) and the modified timer routines are designed to provide timing facilities of greater accuracy.

100:30 DATA MANAGEMENT

Data management components are altered where necessary to operate in a paging environment and to interface with the modified VS2 input/output supervisor (IOS). The significant functional differences between data management in VS2 and MVT exist in IOS. OPEN, CLOSE, EOV, and DADSM routines for VS2 and MVT are functionally equivalent. All the access methods provided in MVT are supported in VS2, except QTAM. All the functions these access methods provide in MVT are also supported in VS2. Programs that use these access methods can be executed in VS2 in either 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. VS2 also provides a new access method called VSAM.

All the VS2 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 region 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 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. A system with a minimum of 512K of real storage is required for TCAM operations.

For performance reasons, certain access methods have 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 in VS2 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 or have buffers packed within pages so they do not cross page boundaries, buffers must be defined and aligned by the programmer.

INPUT/OUTPUT SUPERVISOR

In VS2, IOS provides 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 START I/O instruction (or enqueuing of the request) 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. The only restriction on the size of the CCW list translated is the availability of SQA space.

- Construction of indirect data address lists (IDAL), when necessary. If the buffer specified in a CCW crosses a virtual storage page boundary or if the buffer is larger than 4K, the appropriate IDAL's consisting of indirect data address words (IDAW's) are constructed in SQA also. (IOS does not determine whether buffers that cross virtual page boundaries have actually been allocated contiguous page frames.)

- Short-term fixing of the pages associated with an I/O operation to prevent the occurrence of page faults during the operation. 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 for the duration of the I/O operation. This includes pages that contain control blocks (IOB, DCB, DEB, ECB/DECB, and AVT), I/O appendages, 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 before the I/O operation are unfixed.

The same five I/O appendage interfaces that are provided in MVT are supported in VS2, 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 the 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 this new appendage to short-term fix the areas referenced by the I/O appendage. The new PGFX parameter for the EXCP DCB is provided to indicate that the page fix appendage is to be used.

In addition to the EXCP macro, VS2 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 and by routines that have the required authorization. A routine is authorized to issue EXCPVR if it has a zero protect key, operates in supervisor state, or has APF authorization. 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 region and impair the integrity of the system.

VIRTUAL STORAGE ACCESS METHOD

General Description

Virtual Storage Access Method (VSAM) is a new component of OS/VS 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 VS2 Release 1 environment, VSAM supports 2314/2319, 3330-series (all models), and 2305 (Models 1 and 2) direct access devices. Rotational position sensing is used when the feature is present.

VSAM uses System/370 instructions not available in System/360 and is designed to operate efficiently in a paging environment. 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. VSAM uses the EXCPVR macro for I/O requests.

VSAM is also 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/VS volume can be processed by OS (VS1 or VS2) programs. Similarly, a VSAM data set contained on an OS/VS volume can be processed by DOS/VS programs. 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/VS.

VSAM supports both sequential and direct processing and is designed to supersede ISAM, although the two access methods can coexist in the same OS/VS2 operating system. VSAM supports functions equivalent to those of ISAM and offers several additional 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 to data base and online environments than other OS/VS2 access methods.

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 master catalog or, optionally, a VSAM user 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.
- The ISAM interface routine, which enables the transition from ISAM to VSAM to be made with little or no modification of ISAM programs. This routine is reentrant.

This discussion describes Release 1 of VSAM as implemented in Release 1.7 of OS/VS2. VSAM Release 2 is not supported in VS2 Release 1.7.

General Description of VSAM Data Set Organizations

VSAM supports two different data set organizations, key-sequenced and entry-sequenced, each of which allows both sequential and direct processing, record addition without data set rewrite, and record deletion. The primary difference between these two organizations is the sequence in which logical records are stored.

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, which is called the primary key. Records added after the key-sequenced data set is created are inserted in primary key sequence and existing logical records are moved when necessary. In VSAM key-sequenced data set organization, as in ISAM, each logical record must have a unique, embedded, fixed-length primary key located in the same position within each logical record.

A key-sequenced data set always has a primary index containing primary key values. The entire primary index is used to process records directly and a portion is used to process records sequentially. Alternate indexes for key-sequenced data sets are not supported in VSAM Release 1.

An entry-sequenced VSAM data set, which has no ISAM counterpart, contains either fixed- or variable-length records sequenced in the order in which they were submitted for inclusion in the data set (like a SAM 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. A primary index is never created for an entry-sequenced data set. Alternate indexes for entry-sequenced data sets are not supported in VSAM Release 1.

Physical Structure of VSAM Data Sets

The way in which the extents of the logical records of a VSAM key-sequenced or entry-sequenced data set are physically stored on direct access volumes is quite different from the way in which ISAM logical record extents are stored.

Each extent of a VSAM 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 tracks on the direct access device. 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 VSAM data set can be 512, 1024, 2048, or (except on 2314/2319 devices) 4096 bytes in size only, and they are written without a key (count and data disk record format). VSAM 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 32,768 (32K) bytes in size and contains an integral number of physical records. If a control interval is greater than 8192 bytes, it must be a multiple of 2048 bytes in size.

A control interval contains logical records, free space (for key-sequenced data sets only), system control information about the logical records (record definition fields), and system control information about the free space (control interval definition field), in that sequence.

There is one control interval definition field per control interval and usually multiple record definition fields, depending on the number of logical records in the control interval.

A logical record and its control information (record definition field), although not contiguous within a control interval, are called a stored record. A complete control interval is the unit of data transfer between a VSAM data set and real storage. Hence, command-chained reads/writes are used when a control interval contains more than one physical disk record.

A logical record in a VSAM data set can span physical records within a control interval. A logical record cannot span two or more control intervals within the same control area in VSAM Release 1. Therefore, the maximum logical record size is 32K bytes.

Figure 100.30.1 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 VSAM and for a key-sequenced data set is chosen taking into account the amount of space allocated to the data set, index and data control interval size, and buffer space available to the data set. The maximum size of a control area on disk is one cylinder. 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.

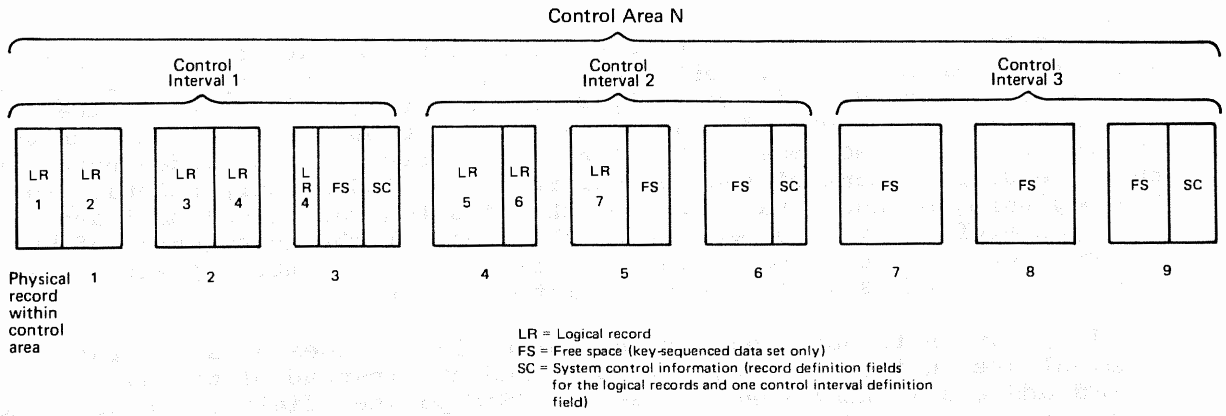

Control Area N
Control Interval 1
Control Interval 2
Control Interval 3
LR 1
LR 2
LR 3
LR 4
LR 4
FS
SC
LR 5
LR 6
LR 7
FS
FS
SC
FS
FS
FS
SC
Physical record within control area
1
2
3
4
5
6
7
8
9
LR = Logical record
FS = Free space (key-sequenced data set only)
SC = System control information (record definition fields for the logical records and one control interval definition field)

Figure 100.30.1. Organization of a control area for a VSAM data set

When a VSAM data set is loaded, VSAM does or does not preformat control areas, depending on the attribute specified when the data set is defined, RECOVERY or SPEED, respectively. When RECOVERY (the default) is specified, during loading VSAM preformats each control area immediately before loading any records into it. Preformatting for a key-sequenced data set consists of putting the appropriate control information in each control interval and an end-of-file indication in the first control interval in the next control area after the control area just preformatted. All zeros in the control interval definition field indicates end of file or end of key range for a key-sequenced data set. For an entry-sequenced data set, control information and an end-of-file indication is placed in each control interval of the control area during preformatting.

The RECOVERY option ensures that if an error that prevents further processing occurs while a control area is being loaded, the previously loaded control areas are not lost. Loading can resume from the first or only end of file indicator. Preformatting is always done when records are added to an existing VSAM data set.

When SPEED is specified, records are loaded without preformatting each control area before loading and the end-of-file indicator is not written until the data set is closed. When this option is chosen, loading proceeds more rapidly, but if an error that prevents further processing occurs, all the records loaded up to that point may be lost and loading would have to resume at the beginning of the data set.

Like an ISAM data set, a VSAM data set can be multi-extent and multivolume. Secondary space allocation can be specified when a key-sequenced or entry-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). A VSAM data set can have a maximum of 126 extents of logical records.

VSAM data sets can be placed on disk volumes that contain data sets with other organizations. Space on a volume that is defined for exclusive use by VSAM is called a <u>data space</u>. A VSAM data space can consist of a maximum of 16 extents on a volume.

Before a VSAM data set or catalog can be loaded, its attributes and space requirements must be defined using the DEFINE function of the access method services program. In order to delete a VSAM data set or uncatalog and make the space available for reassignment, the DELETE function of the access method services program must be used. VSAM space cannot be deleted using the OS/VS job control DISP parameter or an OS/VS data set utility.

The data in a VSAM data set is considered to be mapped into a byte space that can be over 4.2 billion bytes in size. The physical location of a logical record or index entry within a data set is given in the form of a relative byte address rather than a CCHHR disk record address. 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 logical record or index entry, therefore, is independent of the physical characteristics of the direct access device type on which it resides, the number of extents in the data set, the size of a control interval, etc.

All pointers to data that are contained in an index or a control interval are in terms of relative byte address instead of the disk record address (CCHHR) that is used in ISAM pointer fields. In order to locate a desired index or logical record, the VSAM access method calculates the disk address of the physical record, using the RBA of the record and the physical characteristics of the data set. As a result, VSAM data sets are device type independent. A key-sequenced data set can be moved from one device type to another and its index data set need not be recreated.

The logical records of a VSAM data set can be processed by keyed and/or addressed access depending on the organization. For keyed access, logical records are processed in a sequential, skip-sequential (key-sequenced organization only), or direct fashion by a key field, which must be contained in each logical record. For addressed access, records are processed in a sequential or direct fashion by RBA. With keyed or addressed processing, a VSAM data set is processed by the user on the basis of logical records and VSAM always manages the I/O buffers.

Access to a VSAM data set by control interval is also supported, primarily for use by system programmers. This type of access allows the user to read and write a VSAM data set on a control interval basis. That is, each read or write accesses an entire control interval of data. A data set that is opened for control interval access can be processed by keyed and addressed access at the same time (assuming keyed or addressed is supported for its organization) as long as VSAM manages the I/O buffers.

VSAM Macros

The macros provided to define and process VSAM data sets are divided into control block macros and request macros. The control block macros are used to define, modify, display, and test the contents of VSAM control blocks and lists. The request macros are used to specify the processing action (read, write, etc.) to be taken on data and index records.

The following are the VSAM control block macros:

- ACB (generate an access control block). The ACB macro is the VSAM counterpart of the DCB macro that is used for other OS/VS data set organizations. It causes an access control block to be generated during program assembly. One ACB (or GENCB) macro must be specified in a program for each VSAM data set that is to be processed by the program. More than one ACB can be specified in a program for the same VSAM data set. In this case, the ACB's are connected to the same VSAM control block structure and the same set of I/O buffers is used for all requests issued to the data set. The ACB for a VSAM data set must be opened before any processing of the data set can occur.

  The ACB specifies the following for a VSAM data set: name of the DD statement for the data set, address of a list of exit routine addresses for user-written exit routines, buffer space requirements, the password required for the type of processing to be done, all processing options to be used with the data set (keyed, addressed, and/or control interval, sequential, skip-sequential, and/or direct, etc.), and the number of requests that can be outstanding concurrently for the data set using this ACB.

- EXLST (generate an exit list). The EXLST macro is used to define a list of the addresses of the user-written exit routines that are to be entered when certain conditions occur during the processing of a VSAM data set. The EXLST macro causes an exit list to be generated during program assembly.

  Exit to a user-written routine can be taken when end of data set is reached (EODAD exit), a logical error occurs (LERAD exit), an uncorrectable physical I/O error occurs (SYNAD exit), or to perform a journaling operation (JRNAD exit). Each exit routine can be marked active or inactive. An exit routine that is inactive is not entered when its associated condition occurs. The exits to be used during the processing of a given VSAM data set are specified in its ACB (the address of an EXLST macro can be given). More than one ACB can specify the same EXLST macro.

  The journaling exit is taken by VSAM at the following times: whenever a GET, PUT, or ERASE macro is issued to the VSAM data set; each time data is shifted within a control interval or moved to another control interval (key-sequenced data sets only); and each time a physical I/O error occurs.

  A user-written journaling routine can be used, therefore, to keep track of any RBA changes for the logical records of a key-sequenced data set, if it is to be processed by RBA, and/or to record the VSAM transactions that are processed against a VSAM data set (for recovery and reconstruction purposes, for example).

- RPL (generate a request parameter list). An RPL macro is used to generate a request parameter list during program assembly. This list defines a request for processing. Certain request macros (GET, PUT, ERASE, POINT, CHECK, and ENDREQ) must specify the address of a request parameter list to indicate the processing to be performed. The same RPL can be specified in more than one type of request macro.

  An RPL macro specifies the following: the ACB of the data set with which it is to be used (multiple RPL macros can specify the same ACB); the size and address of a work area if logical records are not to be processed in an I/O buffer; the search argument to be used during direct retrieval, skip-sequential retrieval, and positioning (full key, generic key, RBA, or relative record number); address of an ECB if this is an asynchronous request (optional parameter); the type of processing for this request, such as keyed or addressed, sequential or direct, synchronous or asynchronous request, etc.

  When a synchronous request is specified in the RPL indicated by a GET or PUT macro, control is not returned to the instruction after the GET/PUT macro until processing of the request is completed. The logical record is then available for processing. When an asynchronous request is specified, control returns to the instruction after the GET/PUT macro as soon as the request has been scheduled. The user must then test for completion of the I/O operation (usually using a CHECK macro). Asynchronous processing of a request permits the overlap of I/O operations with program execution and is particularly useful with skip-sequential and direct processing. Up to 255 asynchronous requests (RPL's) can be outstanding concurrently for the same VSAM data set.

  Two or more RPL's can be chained together via a pointer field in the RPL itself. A chained parameter list can be used to read or write several records (one for each RPL in the chain) using one GET or PUT macro instead of multiple macros. Chained parameter lists can be used only to retrieve several existing records or to add several new records. It cannot be used to retrieve-for-update, update, or delete existing records.

- GENCB (generate a control block or list). The GENCB macro can be used to generate an ACB, EXLST, or RPL during program execution instead of program assembly. The GENCB macro can be used to eliminate changing these control macros and reassembling VSAM programs when control block formats change in new versions of VSAM.

  The same parameters can be specified in a GENCB macro as in ACB, EXLST, and RPL macros. However, a GENCB macro can specify that multiple copies of the control block are to be generated and parameter values can be specified in more ways (such as in general registers).

- MODCB (modify the contents of a control block or list). The MODCB macro is used to change, during program execution, the contents of an unopened ACB, an EXLST, or an inactive RPL (one not currently involved in a processing operation).

- SHOWCB (display the contents of a control block or list). The SHOWCB macro is used to place the contents of user-specified fields of an ACB, EXLST, or RPL in a user-specified work area.

- TESTCB (test the contents of a control block or list). The TESTCB macro is used to have VSAM compare a user-specified value with a field in an ACB, EXLST, or RPL. The condition code in the PSW is set to indicate the results of the comparison.

The following request macros are used to process VSAM data sets:

- OPEN - A VSAM data set must be opened before it can be processed by other request macros. The OPEN macro provides the same types of processing functions for VSAM data sets as for other types of data sets. OPEN causes the volumes of the VSAM data set to be mounted if necessary, constructs the control blocks required (in addition to those already created by EXLST, ACB, and GENCB macros) for the type of processing to be done, overrides information in the ACB and EXLST with any parameters specified in the DD statement for the data set, causes the loading into virtual storage of any VSAM routines required (in addition to the resident VSAM routines) for the processing specified, and verifies that the password given is correct. Any parameter not specified via job control or the ACB is taken from the catalog entry for the data set.

  Both sequential and direct processing can be performed on a VSAM data set using one OPEN macro and one ACB. Closing and reopening of the data set to switch modes, as is required for an ISAM data set, is not necessary.

- GET - This macro is used for retrieval only and for retrieval and update (GET for update) operations. The RPL specified by a GET macro indicates whether the request is for a retrieval only or a retrieve and update operation. A record that was retrieved by a GET for update request need not be written back if it is not to be changed.

  Locate mode (logical record made available in the input buffer) can be specified for retrieval only (GET) and retrieve for update without record length change (GET for update) operations. In the latter case, however, the updated record must be placed in a work area before it is rewritten. Move mode (logical record made available in a work area) is supported for all read and write requests and is required for all write (PUT and ERASE) operations.

- PUT - This macro is used to write a new record in a data set during its creation or insert a new record in an existing data set. A PUT for update is used to change the contents of an existing record (update it or mark it deleted with a user-defined deletion indication). A PUT for update request must be preceded by a GET for update request. Write verification (automatic reading by VSAM after each write operation) is optional.

- ERASE - This macro is used to delete a logical record from a key-sequenced data set. The record is physically removed from the data set. An ERASE macro must be preceded by a GET for update macro.

- POINT - This macro is used to position VSAM to a particular logical record in the data set from which processing is to continue. Positioning can be in a forward or backward direction and a key value or RBA can be used to identify the logical record at which positioning is set.

- CHECK - This macro is used to cause VSAM to determine whether processing of a specific asynchronous request has been completed and to suspend program execution until processing is completed for an incomplete request. CHECK also causes the appropriate active user-written exit routine to be entered, if necessary, at the completion of the request.

  A test for the completion of an asynchronous request can also be made by specifying an ECB in the RPL for the request and testing the completion bit. Completion can be tested using the TESTCB macro (IO=COMPLETE operand) as well. These two completion tests can be used to delay issuing the CHECK macro until the operation is completed so that processing is not suspended by the CHECK macro.

- ENDREQ - This macro is used to terminate the processing of an asynchronous request whose completion is no longer required or to free VSAM from keeping track of a position in a data set. VSAM can maintain knowledge of the same number of positions as the number of requests that can be outstanding concurrently (specified in the ACB or GENCB macro).

- CLOSE - The CLOSE macro provides the same types of processing functions for VSAM data sets as for other types of data sets. It causes VSAM to write any unwritten data or index records remaining in the output buffers if their contents have changed, update the catalog entry for the data set, if necessary (if the location of the end-of-file indicator has changed, for example), and write SMF records if SMF is being used. The access method control block(s) for the data set (such as the ACB's) are restored to what they were before the data set was opened and virtual storage that was obtained during OPEN processing for additional VSAM control blocks and VSAM routines is released.

  Once a VSAM data set has been closed, it must be reopened before any additional processing can be performed on it. A CLOSE macro with TYPE=T (temporary CLOSE) can be issued to cause VSAM to complete any outstanding I/O operations, update the catalog if necessary, and write any required SMF records. Processing can continue after a temporary CLOSE without the issuing of an OPEN macro.

  When a job step in a region terminates, either normally or abnormally, all open VSAM data sets for the job step are automatically closed.

Figure 100.30.2 illustrates the relationships among the most frequently used control macros and the request macros.

Note that several parts of a VSAM data set can be accessed concurrently via sequential and direct processing by a program or its subtasks using the same ACB without the necessity of closing and reopening the data set. Each request is processed independently and asynchronously with respect to all other outstanding requests. This is called concurrent request processing and is made possible by the fact that VSAM can keep account of multiple positions in the data set at one time. The number of concurrent requests that can be outstanding is specified in the ACB but is extended by VSAM during processing if necessary.

Concurrently outstanding requests for a data set can be any combination of sequential and direct processing requests. Each outstanding request can specify one RPL or a list of RPL's (chained RPL's) and synchronous or asynchronous processing. When a request consists of a list of RPL's, the first RPL in the list determines whether synchronous or asynchronous processing is performed for the request. When synchronous processing is requested in the first RPL, control is not returned to the user until all requests in the list have been processed. When asynchronous processing is specified in the first RPL, control is returned to the user as soon as the chained request is accepted by VSAM, and the processing status of the list must be checked by the user by issuing a CHECK macro for each RPL in the list.

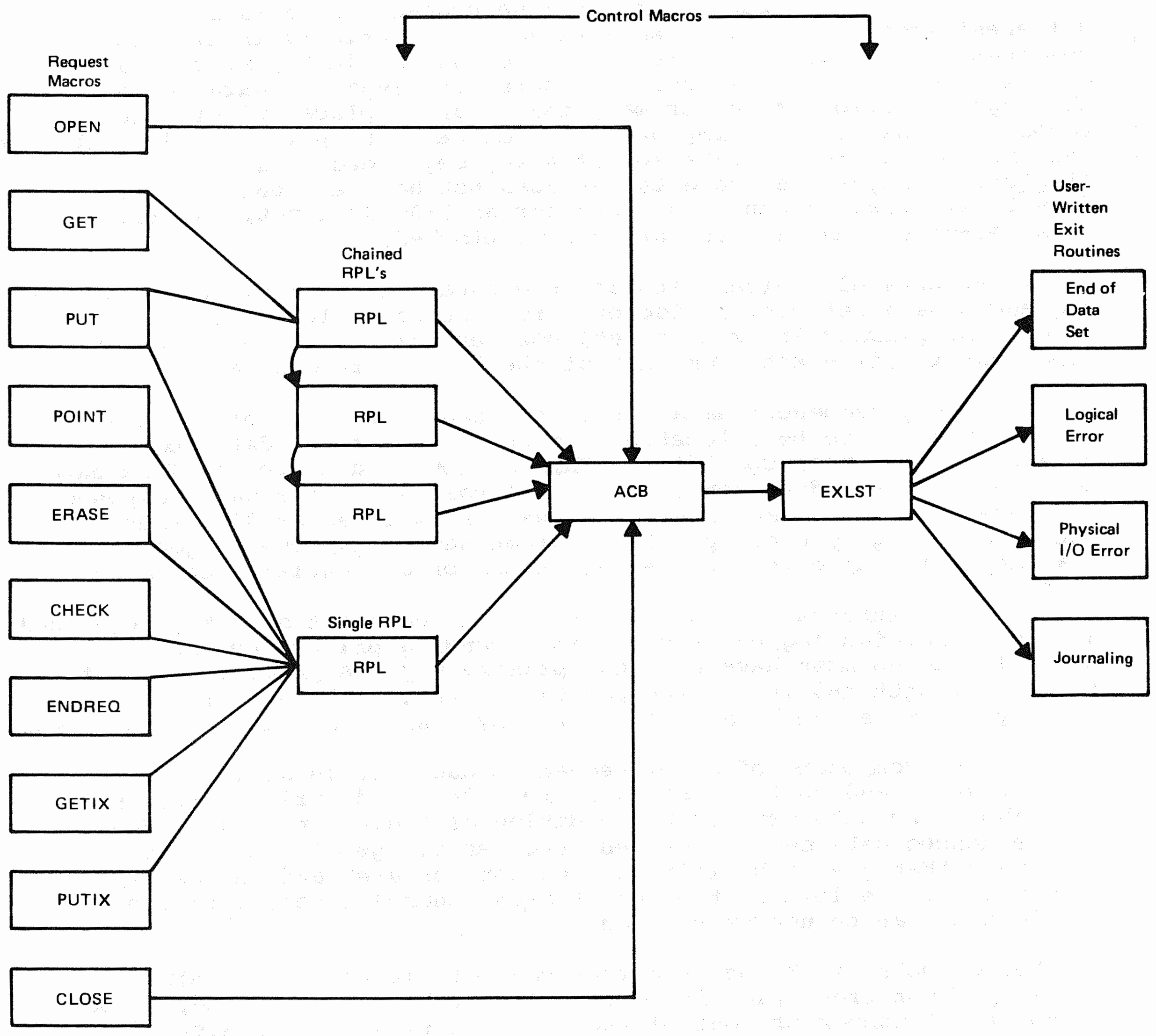


Figure 100.30.2. Relationships among VSAM control and request macros

Key-Sequenced Data Set Organization and Processing

The logical organization of a key-sequenced VSAM data set is very different from that of an ISAM data set. The primary index (index component) and logical records (data component) in key-sequenced organization are two distinct data sets with separate data set names, although a portion of the primary index can be placed within the logical record data set area to improve performance. The primary index data set and the logical record data set of a key-sequenced data set form a cluster. 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.

All extents of logical records (the data component extents) in a key-sequenced data set must reside on direct access volumes of the same type. The primary index data set, however, can be placed on a device type that is different from that of the logical record data set.

When a key-sequenced data set is created, the range of primary key values that are to be allocated to each volume of the data component data set can be user-specified. This cannot be done for an ISAM data set. Unlike ISAM data set volumes, all volumes of the data component of a key-sequenced data set need not always be mounted at OPEN time. Subset mounting by user-specified volume serial numbers in job control statements is supported for certain types of sequential processing.

A control interval in the data component data set of a key-sequenced data set contains logical records in ascending primary key sequence. Logical records must have a unique primary key. A primary key must be fixed in length and in a fixed position within each logical record. Primary key size can be a minimum of 1 byte and a maximum of 255 bytes.

The data component of 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 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 during data set loading can be user-specified.

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 in the data component when the key-sequenced 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 additions.

The use of free space requires less record processing to add a record and to retrieve an addition than would be needed in ISAM, since there are no overflow chains in key-sequenced 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 another control interval is available. The logical record is then added to the appropriate control interval in primary 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 primary index must be updated to reflect this condition. The only times the lowest level of the primary 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 primary 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 then inserted in the appropriate control area in primary 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 in a variable-length key-sequenced data set 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 and record extensions.

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 design makes VSAM key-sequenced organization more suited than ISAM to an online environment.

<u>Logical organization of the primary index data set for key-sequenced organization</u>. Like the index for an ISAM data set, the primary 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 primary index also contains information regarding available space in the primary index data set.

The primary index for a key-sequenced VSAM data set also has a totally different logical structure from that used for an ISAM index. A key-sequenced primary 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. Primary index records are fixed in length and of system-determined size. Each physical index record contains a number of index entries and a pointer to the next physical index record at the same index level. (The last index record in a level does not have such a pointer.) Index entries contain primary keys in ascending collating sequence.

The lowest level of the primary index is called the <u>sequence set</u>. All levels above the lowest are collectively referred to as the <u>index set</u>. The sequence set index level points to all the control intervals in the key-sequenced data set and contains the high compressed primary key value in each control interval. Since the sequence set does not contain an entry for each logical record in the key-sequenced data set, it is a nondense index level. The structure of the primary index for a VSAM key-sequenced data set is shown in Figure 100.30.3.

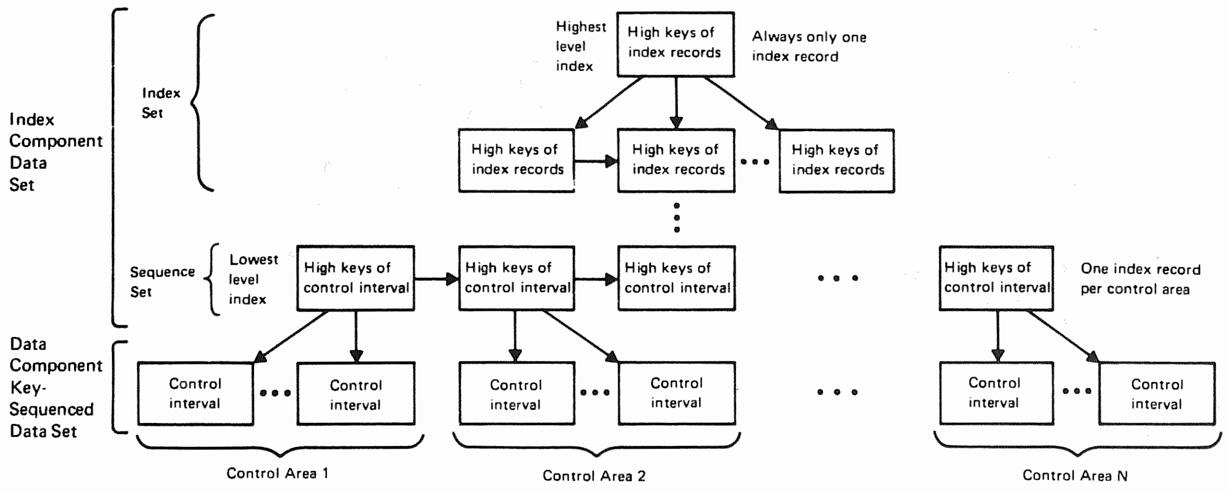

Index Component Data Set
Index Set
Highest level index
High keys of index records
Always only one index record
High keys of index records
High keys of index records
High keys of index records
Sequence Set
Lowest level index
High keys of control interval
High keys of control interval
High keys of control interval
High keys of control interval
One index record per control area
Data Component Key-Sequenced Data Set
Control interval
Control interval
Control interval
Control interval
Control interval
Control interval
Control Area 1
Control Area 2
Control Area N

Figure 100.30.3. Structure of the primary index for a VSAM key-sequenced data set

Each physical 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 primary key value, control information, and a pointer to the control interval in the data component data set that contains the record with that primary key value. The key in the index entry is the highest compressed primary key in the indicated control interval.

When the logical record data set has few enough control intervals that one physical index record can contain all the required index entries, there is only one level of primary 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 primary index data set (index set levels) and stored with the logical records in the data component data set. 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 an ISAM data set).

When the sequence set index level is stored within the data component data set, 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 control area, while initial tracks of the second area will contain replicated index records for the second control 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 primary 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 of the primary index 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 primary index (index and sequence sets) is used to process a key-sequenced data set directly by user-specified primary key values. 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 needed to locate any record directly is based on the number of levels in the primary index and on the current location of the index records to be inspected (on the direct access device or in real storage). Therefore, the same amount of 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 primary 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 in the index 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 (this comparison cannot be done on disk as for ISAM organization). 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 VSAM does not preload index record buffer(s) with as many primary index records as will fit. Index records are allocated space in a buffer and loaded when required.

The primary 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 record 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 compressed index entries in an index record cannot be inspected using a binary search techinque; however, the entries are not inspected sequentially. Index entries are divided into sections for the purpose of searching. The number of sections in an index record is approximately equal to the square root of the number of index entries in the index record.

A primary index search is begun by comparing the search key with the highest key in the first section of the index record. If the search key is less than the highest key, the search continues with the first key in the first section. An equal or the first greater than comparison terminates the search operation. If the search key is higher than the highest key in the first section, it is then compared with the highest key in the second section, etc.

Using this technique, the average number of index entries inspected to locate the desired entry is approximately equal to the square root of the number of entries in the index record. On the average, half of the number of entries in an index record would have to be searched if a linear search technique were used.

Physical structure of the primary index of a key-sequenced data set. Primary index records are stored in control intervals as are the logical records in the data component of a key-sequenced data set. However, there is only one physical index record written in a control interval, control intervals are not grouped into control areas, and no free space is left within a control interval between a logical record (index entry) and its control information.

The physical index records associated with each index level of the primary index are not necessarily stored together in contiguous control intervals (except when the sequence set level is stored separately from the index set levels). When a primary index is created or a new index record is added to an existing primary index, the new index record is placed in the next available control interval after the last existing index record. The level to which each index record belongs is indicated in the control information (header field) in the index record.

In addition to header information and variable-length index entries, a sequence set index record (but not an index set record) can contain a set of free control interval entries. These entries indicate the location of available control intervals in the data component that are within the control area governed by the sequence set index record.

Key-sequenced data set processing. The records in a key-sequenced data set can be processed sequentially, skip-sequentially, or directly using the primary key. Such processing is called keyed sequential, keyed skip-sequential, or keyed direct processing, respectively. All volumes of a key-sequenced data set must be mounted at OPEN for keyed processing.

Records can also be processed sequentially or directly by relative byte address. Such processing is called addressed sequential or addressed direct processing, respectively. Control interval access is supported as well. When addressed sequential processing is used, all volumes of the data set need not be mounted at OPEN time. As many volumes as there are available direct access devices can be mounted at OPEN and the mounting of additional volumes will be requested as they are required, as is done for SAM data sets.

The RBA of a logical record in an existing key-sequenced data set can change only when a record is inserted or deleted, or the size of a variable-length 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 via the journaling exit when appropriate. Programs that process a key-sequenced data set by RBA need not be modified if direct access device type is changed.

Keyed sequential processing of a key-sequenced data set is like sequential processing of an ISAM data set. It is used to load a key-sequenced data set and to retrieve, update, delete, and add logical records to an existing key-sequenced data set. When keyed sequential processing is used, records can be processed in ascending sequence by primary key, using GET and PUT macros. This is called forward processing. Processing records in descending key sequence (backward processing) is not supported in VSAM Release 1. The ERASE macro (not supported by ISAM) can be used to physically delete records.

Key values need not be user supplied for keyed sequential processing, since VSAM automatically obtains the next logical record in sequence. The POINT macro can be issued at any time during processing to position VSAM at a specific logical record from which sequential processing is to proceed. Positioning can be in a forward or backward direction. Only the sequence set of the primary index is referenced for keyed sequential processing by primary key and only for control interval sequencing.

A mass sequential insertion technique, not supported in ISAM, is automatically used by VSAM when additions are sequenced and made using keyed sequential processing. When VSAM determines that two or more logical records are to be inserted between two existing logical records, the control interval (and its sequence set index record if control interval splitting occurs) is not written until the control interval has been packed with all the additions that will fit. Mass sequential insertion is also used by VSAM to add logical records after the last existing record (extend a key-sequenced data set).

The time required to make additions and update the primary index is reduced by using the mass sequential insertion facility of keyed sequential processing. If additions are not sorted and keyed direct processing is used to add the records, the entire primary index must be searched to determine where each logical record is to be placed.

Keyed skip-sequential processing, which is not supported by ISAM, is a variation of direct processing. It can be used for retrieval, update, add, and delete operations (GET, PUT, and ERASE macros). Keys of the logical records to be processed must be presented by the user in ascending sequence. Only the sequence set of the primary index is used for skip-sequential processing using the primary key.

When a relatively small number of transactions that are in primary key sequence are to be processed, skip-sequential processing can be used to retrieve the records directly by key. Since the primary keys presented are in sequence, the access method uses only the sequence set index level of the primary index 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 primary index to be searched for each record.

Keyed direct processing of a key-sequenced data set is like direct processing of an ISAM data set. It can be used to retrieve, update, delete, and add logical records. A primary key value must be presented by the user for each logical record that is to be processed. For a retrieval operation, the key can be the exact key of the desired record, a generic key, or a key that is less than or equal to the key of the desired record. In ISAM, direct retrieval by exact key value only is supported. Positioning by generic key or key less than or equal to the desired record is supported but the record must be retrieved sequentially via a separate operation. The entire primary index is searched to locate the requested logical record during keyed direct processing.

Addressed sequential can be used to process the logical records of a key-sequenced data set in ascending RBA sequence. It can be used to retrieve, update, or delete logical records (GET, PUT for update, and ERASE macros). Addressed sequential cannot be used to add logical records to a key-sequenced data set or to change the length of existing variable-length records.

The user need not supply RBA's during addressed sequential processing. VSAM automatically retrieves records in RBA sequence. Logical records will not be presented in primary key sequence if there have been any control interval or control area splits. Positioning to a given RBA can be accomplished using the POINT macro, as for keyed sequential processing.

Addressed direct processing enables the logical records of a key-sequenced data set to be processed directly by user specified RBA's. As for addressed sequential processing, only retrieval, update, and delete operations can be performed. Additions and record length changes can not be made using addressed direct processing.

Sequential and direct processing of a key-sequenced data set by control intervals is also supported. Skip-sequential processing by control intervals is not supported. For sequential access, records are processed in ascending sequence by control interval. Each GET causes the next control retrieval in sequence to be presented. For direct access, the RBA of each desired control interval must be supplied by the user. Requests can be synchronous or asynchronous and control intervals can be processed in the I/O buffer (except when chained RPL's are used) or in a work area.

The GET, PUT for update, POINT, CHECK, and ENDREQ macros can be used with control interval processing. When updating using control interval access, a control interval can be rewritten without first having been retrieved. The ERASE macro cannot be used nor can PUT macros be issued to load or extend a key-sequenced data set when control interval processing is utilized.

Processing of the primary index data set for a key-sequenced data set. The primary index component of a key-sequenced data set can be processed using GET and PUT macros. The index component data set must be opened alone. The primary index can then be processed like an entry-sequenced data set. It can be accessed using addressed or control interval processing.

Processing summary. Table 100.30.1 summarizes the primary types of processing supported for key-sequenced VSAM data sets (control interval processing is not included in the table).

Table 100.30.1. Types of processing supported for VSAM key-sequenced data sets. (An entry indicates whether the access type is supported, a key or RBA is required, and keys or RBA's must be presented in sequence.)

Type of Access	Keyed Sequential (forward processing only)	Keyed Skip-Sequential (forward processing only)	Keyed Direct	Addressed Sequential (forward processing only)	Addressed Direct
Retrieval only (GET without update)	No keys required	Keys in ascending sequence	Keys not in sequence	No RBA's required	RBA's not in sequence
Retrieval and update, including changing record size (GET and PUT for update)	No keys required	Keys in ascending sequence	Keys not in sequence	Retrieval and update only. No RBA's required.	Retrieval and update only. RBA's not in sequence.
Add (PUT without update)	No keys required	Keys in ascending sequence	Keys not in sequence		
Delete (ERASE)	No Keys required	Keys in ascending sequence	Keys not in sequence	No RBA's required	RBA's not in sequence

Entry-Sequenced Data Set Organization and Processing

The logical records in an entry-sequenced data set are ordered by the sequence in which they are presented for entry into the data set. Free space cannot be left within the control intervals and control areas of an entry-sequenced data set when it is defined. Additions to an existing entry-sequenced data set are placed in any available space left at the end of the data set. Extents can be added to an existing entry-sequenced data set if secondary allocation was specified when the data set was defined. All logical record extents of an entry-sequenced data set must be placed on volumes of the same direct access type. Although an entry-sequenced data set consists only of a data component and cannot have an index, it is still referred to as a cluster.

The ERASE macro is not supported for entry-sequenced data sets. A record that is to be deleted must be marked deleted with an installation-defined identification. Space made available by marking a record deleted is not reclaimed. The space occupied by a record marked deleted can be reused only by storing a new record of the same size in the space.

Available space at the end of the data set is also used when the size of an existing record in a variable-length entry-sequenced data set is to be changed. The existing record must be marked deleted by the user with an installation-defined deletion identification, and the lengthened or shortened record must be written at the end of the data set.

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 by the user. Other records are not affected since the changed record is moved to the end of the data set. 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.

Entry-sequenced data set processing. Addressed sequential, addressed direct, and control interval processing are supported for entry-sequenced data sets. When addressed sequential is used, records can be processed in ascending RBA sequence, using GET and PUT macros. The POINT macro can be used for forward or backward positioning to a specific RBA. For addressed sequential processing, no RBA is given by the user. VSAM automatically presents records in RBA sequence.

When addressed sequential is used to process records in ascending RBA sequence, existing records can be retrieved, updated (but not changed in size), and marked deleted, and new records can be added. Record size changes can be accomplished by the procedure described previously.

Addressed direct processing by user-supplied RBA's can be used to retrieve records, update their contents (but not change their size), and mark records deleted. New records cannot be added and record size changes cannot be made during addressed direct processing.

An entry-sequenced data set can be processed by control interval using addressed sequential or addressed direct (by RBA) access. The control intervals in an existing entry-sequenced data set can be retrieved and updated (but new control intervals cannot be added) using sequential or direct access and a new entry-sequenced data set can be created using sequential control interval processing. GET, PUT, POINT, CHECK, and ENDREQ macros can be used. If updating is to be performed, a work area must be used.

An entry-sequenced data set can also be used like a BDAM data set. A randomizing routine can be used to associate the control field of a logical record with an RBA. The randomizing routine must include a technique for assigning an alternate RBA to synonyms (records whose control field converts to the same RBA as an existing record in the data set). The entry-sequenced data set must be preformatted with dummy records before the logical records are placed in the data set.

Table 100.30.2 summarizes the primary types of processing supported for VSAM entry-sequenced data sets. Access by control interval is not included in the table.

Table 100.30.2. Types of processing supported for VSAM entry-sequenced data sets. (An entry indicates whether the access type is supported, an RBA is required, and RBA's must be presented in sequence).

Type of Access	Addressed Sequential (forward processing only)	Addressed Direct
Retrieval only (GET without update)	No RBA required	RBA's not in sequence
Retrieval and update without record size changes (GET and PUT for update)	No RBA required	RBA's not in sequence
Delete	Records marked deleted by user identification. No RBA's required.	Records marked deleted by user. RBA's not in sequence.

VSAM Catalogs

Unlike ISAM data sets, all VSAM data sets must be cataloged in a VSAM catalog. Information required to process a VSAM data set, such as its location and physical characteristics, is contained in the VSAM catalog. Non-VSAM data sets that are not part of a generation data group can also be cataloged in a VSAM catalog.

There must be one VSAM master catalog for a VS2 operating system that contains VSAM. Optionally, one or more VSAM user catalogs can be defined. Each catalog is an individual data set. The VSAM master catalog data set is cataloged in the VS2 data set catalog (SYSCTLG), and each VSAM user catalog has an entry in the VSAM master catalog. Each VSAM data set is cataloged in the VSAM master catalog or a user catalog, but not both. All VSAM data sets on the same volume must be cataloged in the same VSAM catalog. Duplicate data set names in the same VSAM catalog are not permitted but a given data set name can appear in more than one VSAM catalog.

VSAM user catalogs can be used to reduce the size of the VSAM master 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 master catalogs.

The following information is recorded in the catalog record 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, physical record size, number of control intervals in a control area, location of the primary key field for a key-sequenced data set, 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 between a key-sequenced data set and its index
- Information that indicates whether a key-sequenced data set or its primary index has been processed individually (without reference to the other)

A VSAM catalog also contains information regarding the location of data spaces and 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 it contains available space. VSAM catalog/DADSM routines, instead of OS/VS catalog and DADSM routines, are used to process the catalog and to allocate space on 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.

Several types of entries are used in a VSAM catalog to describe the various objects the catalog defines (data sets, available space, etc.). The entry types are cluster, data component, primary index component, user catalog, non-VSAM space or volume, and alias. A given data set may require more than one entry type for its description. A key-sequenced data set, for example, requires a cluster, primary index component, and data component entry.

A VSAM catalog is logically structured as a key-sequenced data set that contains 44-byte keys and variable-length records. The data component is physically divided into two address range areas. One area is the high-address range and the other is the low-address range. The index component is physically embedded between the two address range areas.

A VSAM catalog can be accessed as a catalog using access method services commands and the SHOWCB and TESTCB macros. A VSAM catalog can also be opened and processed as a key-sequenced data set. Keyed, addressed, and control interval processing are permitted.

Access Method Services Program

The access method services general purpose, multifunction service program is provided to support functions required to create, maintain, and back up 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 VS2, 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. The DEFINE function is also used to define data spaces and to catalog non-VSAM data sets in a VSAM catalog.
- Create, reorganize, and back up VSAM data sets. Input to the REPRO function can be an ISAM, SAM, or VSAM (key-sequenced or entry-sequenced) data set. The output can be a VSAM (key-sequenced or entry-sequenced) or SAM data set. When the input and the output organizations are different, conversion occurs. The REPRO function, therefore, can be used to convert an ISAM data set to VSAM key-sequenced 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 or a VSAM catalog. 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.

- Delete data sets, data spaces, indexes, and catalogs and make the space available for reallocation. The freed space is overwritten with binary zeros if the erase option is specified.

- Perform processing required to make a VSAM data set portable from one System/370 to another if a VSAM user catalog is not available. The EXPORT command is used to copy a VSAM data set (any organization) to a tape or disk volume as a sequentially organized data set. Required information is extracted from the catalog entry for the data set and written on the transporting volume as well. The IMPORT command is used to create a VSAM data set and its catalog entry from the data set created by an EXPORT command. EXPORT and IMPORT are also used to disconnect a VSAM user catalog from one VSAM master catalog and catalog it in another VSAM master catalog. In this case, the volume containing the VSAM user catalog is transported from one system to another without copying.

- Create backup copies from VSAM data sets. The EXPORT command is used to create the backup copy (as for exportation) and the IMPORT command is used to load the backup copy into the system if necessary.

- Verify the accessibility of an existing VSAM data set (using the VERIFY command). This function involves checking for valid end-of-file or end-of-key range information in the catalog entry for a VSAM data set. If the catalog information does not agree with the actual end-of-file or end-of-key range in the data set, the catalog information is updated.

- Convert entries in the OS or OS/VS system catalog to entries in an existing VSAM master or user catalog, using the CNVTCAT command.

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 keywords are defined for VSAM. The DD names JOBCAT and STEPCAT are provided for specifying the VSAM user catalogs available to a job or job step, respectively. The AMP parameter is provided for overriding ACB, EXLST, and GENCB parameters that are specified in the processing program, supplying missing ACB or GENCB macro operands, indicating checkpoint/restart options, specifying ISAM interface options, requesting storage dumps of VSAM access method control blocks, and indicating that the DD statement defines a VSAM data set under certain conditions (DUMMY specified in the DD statement, for example).

Password Protection

An expanded password protection facility is supported for VSAM. Optionally, passwords can be defined for clusters, cluster components (data component and index component), and VSAM catalogs. Passwords are kept in VSAM catalog entries. The password can be supplied by the programmer via the ACB. If password protection is indicated for a VSAM data set and the ACB does not specify a password or specifies it incorrectly, the operator must supply the correct password in order for the data set to be opened. Up to seven retries can be made.

Multiple levels of protection are provided:

- Full access, which allows access to a data set, its index, and its catalog entry. Any operation (read, add, update, delete) can be performed on the data set and its catalog entry.

- Control interval access, which allows the user to read and write entire control intervals using the control interval interface. All read, write, and update operations can be performed at the logical record level as well. 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 (definition of new objects and alteration of existing objects), 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.

A password can be defined for a given VSAM data set for each level of protection: master password, control interval access password, read-write-add-delete password, and read-only password. When multiple passwords are defined for a data set, the password given when the data set is opened establishes the level of protection to be in effect for this OPEN.

Authorization to process a VSAM data set can be supplemented by a user-written security authorization routine. If supplied, such a routine must reside in SYS1.LINKLIB. It is entered during OPEN processing after password verification has been performed by VSAM, unless the master-access password was specified. A user security authorization record of up to 255 bytes maximum 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.

<u>Data Set Sharing</u>

A VSAM data set can be accessed concurrently by two or more subtasks within the same region and two or more job steps (regions) when DISP=SHR is specified for the VSAM data set by each job step. Both types of sharing can be used for a VSAM data set at the same time. The type of data set sharing permitted for two or more regions is controlled by using the SHAREOPTIONS parameter of the DEFINE command when the VSAM data set is defined. The following types of cross-region sharing options are supported:

- The data set can be opened by one user for output processing (to update or add records) <u>or</u> the data set can be opened by multiple users for read operations only. Full read and write integrity is provided by this option (SHAREOPTIONS 1).

- The data set can be opened by one user for updating or record addition (output operations) <u>and</u> by multiple users for read-only processing. Since only one user can perform write operations, write integrity is provided by this option. However, read integrity must be provided by each user since users can read a record that is in the process of being updated (SHAREOPTIONS 2).

- The data set can be opened by any number of users for both read and write operations. Data set integrity must be maintained by the user. No integrity (read or write) is provided by VSAM (SHAREOPTIONS 3).

- The data set can be opened by any number of users for both read and write operations. The ENQ and DEQ macros must be issued by users to maintain data integrity. For direct processing operations, VSAM provides a new buffer for each request (SHAREOPTIONS 4).

Data set sharing by subtasks within the same region can be accomplished using one DD statement for the VSAM data set or multiple DD statements. When a single DD statement is used, multiple subtasks in the same region can perform read and update operations on the VSAM data set. VSAM uses the exclusive control facility to maintain integrity during update operations. The SHR disposition parameter need not be specified in order to share a VSAM data set when one DD statement is used. However, if DISP=SHR is specified when one DD statement is used, both subtask sharing and cross-region sharing (as described above) can be used concurrently.

When multiple DD statements are used, multiple subtasks within a region can share a VSAM data set using the same options as are supported for cross-region sharing. The DISP=SHR parameter must be specified on the DD statements.

VSAM data sets can also be shared across systems as follows:

- The data set can be opened by any number of users for both read and write operations. VSAM provides a new buffer for each direct processing request and RESERVE and RELEASE macros must be issued by users to maintain data set integrity. All job steps that are accessing a VSAM data set concurrently must specify DISP=SHR if data set integrity is to be maintained.

- The data set can be opened by any number of users for both read and write operations. Data set integrity is a user responsibility as VSAM does not provide any.

Note the following restriction when DISP=SHR is specified for cross-region or cross-system sharing, VSAM is providing a new buffer for each direct processing request, and users are issuing ENQ/DEQ or RESERVE/RELEASE macros to ensure data set integrity (SHAREOPTION 4 is specified). VSAM will not allow a control area split for this sharing of a key-sequenced data set. VSAM indicates no-space-available if an attempt is made to add or change the size of a record and a control area split is required to perform the operation.

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 the 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 capability 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 can be used in Assembler, COBOL, and PL/I programs. The PL/I Optimizing and PL/I Checkout compilers and FULL ANS COBOL support VSAM organization directly, that is, without use of the ISAM interface routine.

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 detail in _OS/VS Virtual Storage Access Method Programmer's Guide_ (GC28-3838). 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 ascending key sequence and one with record in time-of-arrival sequence.
- Data sets are direct access 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 and disk space requirements are therefore reduced.
- 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 a key-sequenced data set is created.
- Free space reclamation capabilities are expanded considerably. This factor can eliminate or significantly increase the time between key-sequenced data set reorganizations.
- Subset mounting by volume serial number is supported for sequential processing.
- 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.

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. Index search time is thus minimized.
- Overflow chains are eliminated and the time required to make an addition to a key-sequenced data set is therefore reduced.
- The same time is required to retrieve an added record as an original record in key-sequenced organization.
- 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 100.30.3 compares the features of VSAM and ISAM as supported in OS/VS2 Release 1.7.

Table 100.30.3. Comparison table of VSAM and ISAM facilities for OS/VS2 Release 1

Characteristic	VSAM - OV/VS2	ISAM - OS/VS2
1. Supporting OS environments	VS1 and VS2 (VS2 Releases 1, 1.6, and 1.7 support VSAM Release 1 only)	VS1 and VS2
2. Direct access devices supported	2314/2319, 3330-series (all models), and 2305 Models 1 and 2	Same as VSAM
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 a primary key. A primary index is always provided. The logical records and the primary 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 of the primary index.	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 in the order in which they are placed in the data set. New records are added to the end of an existing data set.	Not supported
4. Multiple extents and volumes for a data set	Yes	Yes
a. Secondary space allocation indicated at creation	Yes	No
b. Volumes of the same device type required	Yes for logical record extents. The primary index data set can be on a device type that is different from that which contains they key-sequenced logical records.	Yes for all the volumes containing prime and separate overflow area extends. 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 Subset mounting by volume serial number is supported for sequential processing by RBA.	Yes
d. Free space available within the logical record area	Yes (for key-sequenced data sets only), 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 direct access device independent	Yes RBA pointers are used in the control interval and in all indexes	No Record address ID (CCHHR) is used in index pointers

Table 90.30.3. Comparison table of VSAM and ISAM facilities for OS/VS2 Release 1 (continued)

Characteristic	VSAM - OS/VS2	ISAM - OS/VS2
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 records formats are used for prime records. Records in an overflow area are always unlocked.
b. Key field is written on disk	No Physical disk records are written in count and data format.	Yes Physical disk records are written in count, key, and data format.
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	
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.
g. Allocation of logical records to volumes by key range	Yes	No
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. Logical index records are blocked	Yes	No
e. Physical index record size	Fixed length and determined by system	Data field is always ten bytes. Key field is key size.
f. Keys are compressed in the index component	Yes Front and rear compression eliminates 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 placed adjacent to logical records for key-sequenced organization	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

Page of GC20-1753-1
Added July 18, 1975
By TNL GN20-3481

Table 90.30.3. Comparison table of VSAM and ISAM facilities for OS/VS2 Release 1 (continued)

Characteristic	VSAM - OS/VS2	ISAM - OS/VS2
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 or entry-sequenced data set.	Yes	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 ascending primary key sequence. The sequence set index level of the primary index is used.	Yes Each logical record is presented in ascending key sequence. The track index is used. Processing in descending key sequence is not supported.
b. Skip-sequential retrieval, addition, and update (by keys specified in ascending sequence)	Yes The sequence set index of the primary index is used.	No
c. Sequential retrieval and update by record address	Yes, via RBA (addressed sequential or addressed direct)	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 A record must be retrieved to be updated.	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. Alternatively 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.

Page of GC20-1753-1
Added July 18, 1975
By TNL GN20-3481

Table 90.30.3. Comparison table of VSAM and ISAM facilities for OS/VS2 Release 1 (continued)

Characteristic	VSAM - OS/VS2	ISAM - OS/VS2
k. Variable-length logical records can be lengthened or shortened	Yes, and space is reclaimed for a shortened record.	Yes (with space reclamation as indicated above)
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 • Control interval access allows read/write of entire control interval as well as of individual logical records. • Update access - allows access to the data set and its index, for retrieval, updating, deletions, and additions. Limited modification of the catalog entries for the data set is permitted but an entry cannot be deleted. • Read access - allows retrieval of data records only (no writing)	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.
a. User-written authorization routines supported	Yes	No
10. Data set sharing		
a. Within a region	Yes	Yes
b. Across region(DISP=SHR)	Yes	Yes
c. Across systems	Yes	Yes
11. Data set cataloging	Required The VSAM master catalog or a VSAM user catalog must be used.	Optional The OS/VS data set catalog (SYSCTG) is used. There is no special catalog for ISAM data sets.
12. Languages supporting (for VSAM directly and via ISAM interface)	Assembler COBOL PL/I	Assembler COBOL PL/I RPG
13. VSAM data set direct input to sort/merge	Yes, OS/VS Sort/Merge only	No

Table 90.30.3. Comparison table of VSAM and ISAM facilities for OS/VS2 Release 1 (continued)

Characteristic	VSAM - OS/VS2	ISAM - OS/VS2
14. Utility program functions	Access method services program can perform the following: • Define and delete 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, ISAM, or SAM data set • Perform functions required to make a VSAM data set or catalog 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

100:35 PAGE MANAGEMENT

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

- Interface control
- Real storage administration
- External page storage administration
- Page administration

The interface control routine is primarily responsible for receiving requests for page management services and for controlling the flow of requests to the other page supervisor routines, which actually perform the required services.

REAL STORAGE ADMINISTRATION

The routines that are part of real storage administration perform all real storage allocation and deallocation. Requests for services associated with real storage can be implicit, such as after a page fault occurs, or explicit, such as those requested via page supervisor macros.

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. Pages are marked short- or long-term fixed as indicated in the PGFIX macro. 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. A suspend parameter can be used to indicate that the request can be queued if no real storage is currently available. Pages marked fixed cannot be paged out until a PGFREE macro is issued.

- 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 function.

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

- Deallocate the page frames and the slots 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. This macro is issued to free the real storage and external page storage associated with a virtual storage page when its contents are no longer required. (PGRLSE does not cause the virtual storage to be deallocated.) For example, during job step termination, PGRLSE is issued to release the page frames and slots associated with a pageable problem program region.

Page management services are implemented primarily for use by control program routines; however, system programmers can use them, if necessary. The PGRLSE macro is the only page management macro that can be issued by an unauthorized problem program. The other macros can be issued by a problem program if authorized via APF. User-written routines that operate in supervisor state or with a protect key of zero can also use all the page management services macros.

The real storage allocation routine processes requests for the allocation of page frames to satisfy page faults and explicit requests via macros. The allocation technique implemented attempts to (1) minimize paging requirements asssociated 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 active pages in the system to reduce paging activity for executing tasks.

The status of all real storage in the system is reflected in the page frame table (PFT), which contains one 16-byte entry for each 4K page frame in the system. The page frame table entries (PTFE's) are connected by pointers to form six page frame status queues and one SQA/LSQA reserve queue. The latter queue contains the page frames that are reserved for allocation to SQA and LSQA virtual storage pages. The PFTE's are initialized at IPL and thereafter always reflect the current status of each page frame. The page frame table is contained in the nucleus.

A PFTE contains a pointer to the TCB of the task to which the page frame is assigned, the number of the virtual storage page to which the page frame is assigned (if any), a fix counter, queue pointers to indicate the page frame status queue of which the PFTE is a part (if any), and status flags. Certain status flags indicate whether the page frame contains a long-term fixed page, a page that is being paged in or out, or an LSQA or SQA page. Other status flags identify the page frame as not allocatable (because of a permanant malfunction), allocated to a nonpageable region, or required for allocation to a nonpageable region.

The following page frame status queues are maintained:

- Available queue, which 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. Whenever a TSO region is swapped out, the PFTE's for the page frames allocated to the region are placed at the beginning of the available queue. Allocated page frames that become inactive can be placed on this queue. An available page frame count (APC) is maintained that always reflects the number of page frames in this queue. The available queue has a low threshold value and a replenish count, which are used in determining when to replenish the available queue and by how much, respectively. A status flag identifies a PFTE as belonging to the available queue.

- Hold queue, which indicates the page frames most recently allocated. These page frames are not immediate candidates for reallocation. As soon as a page frame is allocated, it is placed in the hold queue to enable it to be used by the task to which it is allocated before it is put on an active queue and, thereby, made available for reassignment. A status flag identifies a PFTE as belonging to the hold queue.

- Four active queues, which contain all the PFTE's that are not long-term fixed, unallocatable, or in the available queue or the hold queue. These queues reflect the currently allocated page frames that are not long-term fixed. As page frames in these queues become inactive, they are subject to being placed on the available queue, as per the page replacement algorithm.

Real storage is allocated from the available queue, which contains unassigned page frames. Frequently referenced page frames are normally not taken from one task to be allocated to another; however, this can occur if a situation arises in which there are no unassigned or inactive page frames available for allocation to a task.

Tasks execute on a priority basis and, therefore, requests for page frames are received and allocated on a priority basis. However, except for a swap-in operation for a TSO user, page management does not ever attempt to ensure that a given number of page frames are allocated to each task (page frames are assigned 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.

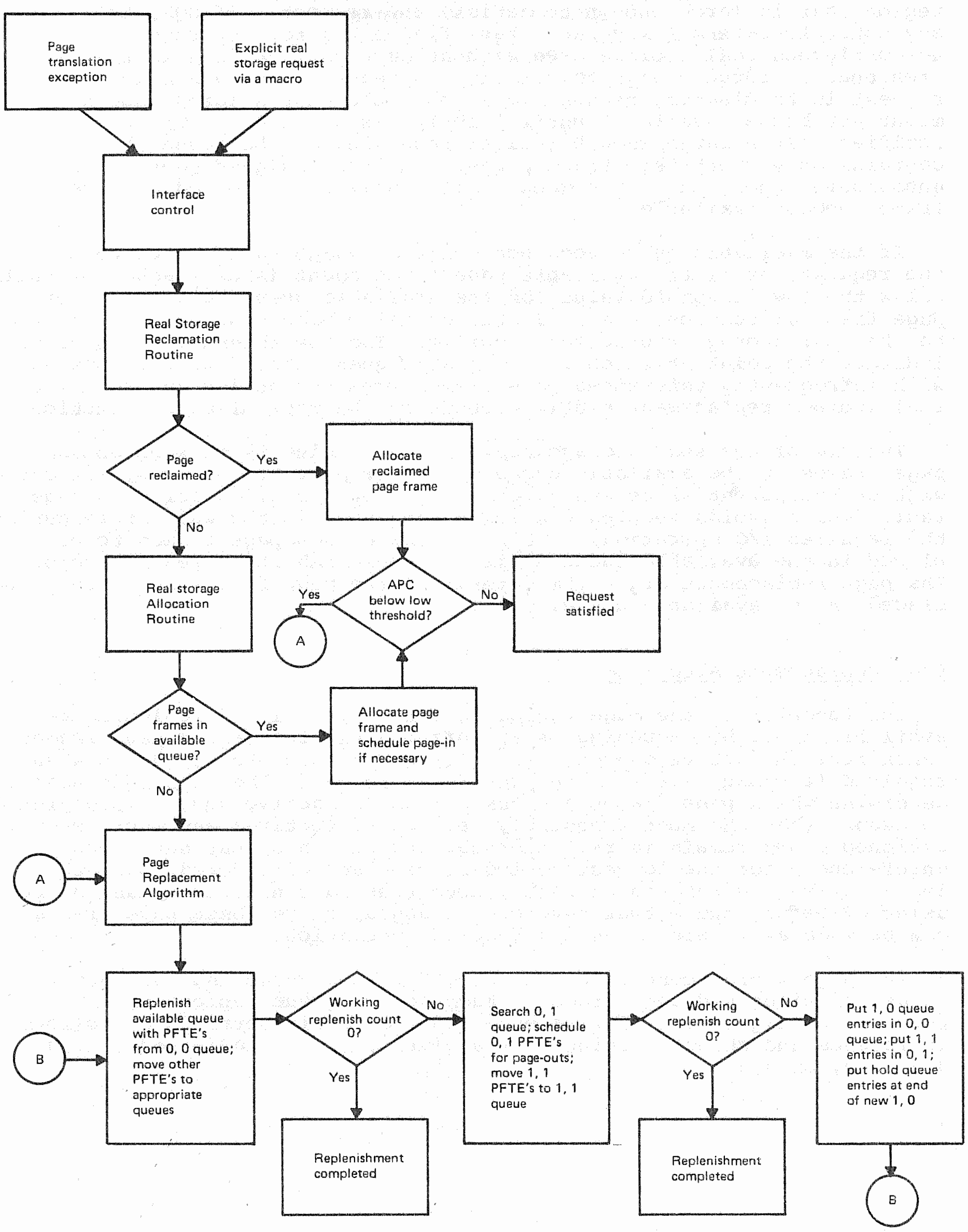
At regular intervals, the page supervisor inspects the paging activity of the system. If it is deemed to be too high, a deactivation procedure is entered to make real storage available. This is done to prevent the occurrence of thrashing.

Real Storage Allocation

The following is done to service a real storage allocation request (refer to Figure 100.35.1). Before allocating a page frame, page reclamation is attempted. If the contents of the referenced virtual storage page are still in real storage, a page-in operation can be avoided. Page reclamation is possible (1) if the page frame last assigned to the virtual storage page has not yet been reassigned (is in the available queue), (2) when the page frame containing the desired page is still waiting to be paged-out, and (3) when a page frame is in the process of receiving the desired page in response to another request. Reclamation is not attempted when a TSO region is swapped in.

If reclamation is not possible, the real storage allocation routine attempts to allocate a page frame from the available queue. If a page frame can be allocated from this queue, its PFTE is removed and the available page frame count is decremented. If a page-in is required, a request is placed in the appropriate page-in device queue, and the task requiring the page remains in the wait state. If a page-in is not required because this is the first time a page frame has been assigned to this page, the PFTE is placed at the end of the hold queue and the allocated page frame is initialized to zero (for data security protection) if the optional page frame clearing facility was selected at system generation. Page frame clearing can be eliminated to reduce the amount of CPU time used by the page manager. The appropriate page table entry is updated to reflect the allocation of real storage. The same procedure is used to service a request for more than one page frame.

If the page frame allocation request indicates long-term fixing (for SQA or LSQA, for example), a page frame in real storage located above the V=R line address is selected if possible. If a request is received to long-term fix a page that is already present in real storage and the page resides in the V=R area, it is moved outside the V=R area if possible. If no page frame outside the V=R area is available, any available page frame within the V=R area is allocated.



Page translation exception
Explicit real storage request via a macro
Interface control
Real Storage Reclamation Routine
Page reclaimed?
Yes
No
Allocate reclaimed page frame
APC below low threshold?
Yes
No
A
Request satisfied
Real storage Allocation Routine
Page frames in available queue?
Yes
No
Allocate page frame and schedule page-in if necessary
A
Page Replacement Algorithm
B
Replenish available queue with PFTE's from 0, 0 queue; move other PFTE's to appropriate queues
Working replenish count 0?
No
Yes
Replenishment completed
Search 0, 1 queue; schedule 0, 1 PFTE's for page-outs; move 1, 1 PFTE's to 1, 1 queue
Working replenish count 0?
No
Yes
Replenishment completed
Put 1, 0 queue entries in 0, 0 queue; put 1, 1 entries in 0, 1; put hold queue entries at end of new 1, 0
B

Figure 100.35.1. Flow of the real storage allocation procedure

The V=R real storage allocation routine is entered after it has been determined that the V=R dynamic area of virtual storage contains a virtual storage area large enough to satisfy the region size request. The V=R allocation routine attempts to locate a contiguous real storage region that is large enough to satisfy the request. If SQA, LSQA, or any other long-term fixed pages have fragmented real storage such that no contiguous real storage area without such pages exists in the V=R area that is large enough to satisfy the request, the V=R allocation request is terminated, because it is possible that a large enough area might not become available until a re-IPL is performed. The operator is notified. If a large enough real storage area can be found that contains only short-term fixed pages, nonfixed assigned pages, and unassigned pages, the V=R request waits until the required assigned page frames become available.

If the available queue does not contain enough page frames to service the request, or if the available page frame count (APC) reaches or falls below the low threshold value for the available queue as a result of page frame allocation, the real storage allocation routine gives control to the real storage replacement routine. The low threshold is used to indicate the point at which the available queue should be replenished with infrequently referenced page frames from the active queues. The real storage replacement routine schedules the replenishment function.

The aim of the real storage replacement routine is to keep enough page frames in the available queue to enable page frames to be allocated without the potential necessity of a page-out operation for each page fault (which avoids keeping the requesting task in the wait state during the required I/O operation). This routine causes page frames to be placed in the available queue until the replenish count reaches zero. The page replacement algorithm determines the page frames that are to be placed in the available queue.

Page Replacement Algorithm

The function of the page replacement algorithm is to replenish the available queue by enqueuing on it infrequently referenced page frames taken from the active queues. Page-out operations are scheduled when required (a change bit for the page frame is on). The technique used to determine which page frames are taken from the active queues is designed to ensure that the most frequently referenced (active) and most recently assigned pages remain in real storage. Page frames that have been unreferenced for the longest period of time are considered to be the least active. Unreferenced page frames that have not been changed are selected before those that have been changed, since these page frames can be made available without a page-out operation.

The page replacement algorithm uses one hold queue and four active queues of allocated page frames. Each active queue represents a possible configuration of reference and change bit settings as follows (reference and change settings for a given PFTE are indicated as 0,0; 0,1; 1,0; or 1,1):

Reference Bit	Change Bit	Active Queue Contents
0	0	PFTE's for page frames that were unreferenced and unchanged since the last inspection by the replacement algorithm. Only 0,0; 1,0; or 1,1 PFTE's can be in this queue.
0	1	PFTE's for page frames that were unreferenced since the last inspection by the replacement algorithm but that were changed at some previous time. Only 0,1 and 1,1 PFTE's can be in this queue.
1	0	PFTE's for page frames that were referenced since the last inspection by the replacement algorithm but that were not changed. Only 0,0; 1,0; or 1,1 PFTE's can be in this queue.
1	1	PFTE's for page frames that were referenced and changed since the last inspection by the replacement algorithm. Only 0,1 or 1,1 PFTE's can be in this queue.

A replenish count, as well as a low threshold value and an APC, is associated with the available queue. The low threshold value (LTH) and the replenish count (REPCT) can be supplied by the operator during IPL via the PAL parameter. They cannot be specified during system generation. LTH can be a value from 1 to 99 and defaults to 5. REPCT can be a value from 1 to 99 and defaults to 3. The system-supplied defaults are used if the operator does not provide these values. The page replacement algorithm is entered whenever the APC falls below the low threshold value for the available queue. The page relacement algorithm attempts to take the number of page frames specified by the replenish count from the unreferenced, unchanged (0,0) queue, and place them in the available queue. As the 0,0 active queue is inspected, page frames whose reference and change recording bits have changed since the last inspection are moved to their appropriate queues. After a PFTE is moved from one active queue to another, the reference bit for the page frame associated with the PFTE is set to zero. The change bit for a PFTE is set to zero only when the entry is placed in the available queue after a page-out operation and at the completion of a page-in. The reference bit also is set to zero after a page-in.

The following defines the activity of the page replacement algorithm during an inspection sequence.

- A working replenish count is set equal to the replenish count value. The 0,0 active queue is searched serially from top to bottom. PFTE's with a 0,0 reference and change bit setting are placed in the available queue, the working replenish counter is decremented, and the APC count is incremented. PFTE's with reference and change settings other than 0,0 are moved to the appropriate active queue. Their reference bit is set to zero, and they are placed at the end of the queue they are assigned. Hence, active queues are maintained in FIFO sequence to preserve a record of comparative length of time in the queue among PFTE's in the same active queue. (The hold queue is maintained in FIFO sequence also.) The 0,0 active queue is searched until enough 0,0 PFTE's are found to raise the APC to the high threshold value of the available queue.

- If enough 0,0 PFTE's can be placed in the available queue from the 0,0 active queue to reduce the working replenish count to zero, page replacement processing terminates. If enough 0,0 PFTE's are not found in the 0,0 active queue, the 0,1 active queue is inspected next from top to bottom. PFTE's with a 0,1 reference and change bit setting are selected and scheduled for a page-out operation. PFTE's with a 1,1 setting are moved to the 1,1 active queue and their reference bits are turned off. The working replenish count is reduced when a 0,1 PFTE is selected. However, 0,1 PFTE's that are selected do not cause the APC to be incremented, since the pages these PFTE's represent can be reclaimed before the page-out occurs. Once a page-out is completed, the associated PFTE is placed in the available queue and the APC is incremented.

- Since the 0,0 and 0,1 queues have been depleted by the previous two searches, they must be replenished. All the PFTE's on the 1,0 active queue are moved to the 0,0 queue, and the entire contents of the 1,1 queue are moved to the 0,1 queue. All the PFTE's in the hold queue are moved to the 1,0 queue. The reference bits of PFTE's are not reset to zero when the queues are switched. The 0,0 queue is then searched as before for 0,0 entries, and entries with other settings are moved to the appropriate queues.

Figure 100.35.2 illustrates the operation of the page replacement algorithm. The page replacement algorithm used in VS2 is sensitive to the paging rate of the system. The higher the paging rate, the more frequently the algorithm will be entered to reexamine the status of the PFTE queues and reclassify them accordingly. The page replacement algorithm also selects unreferenced, unchanged pages before unreferenced, changed pages when replenishing the available queue. Therefore, inactive refreshable pages will be made available for allocation before inactive nonrefreshable pages.

The page supervisor monitors the paging activity of the system. When it becomes too high, as determined by user- or system-specified values, task deactivation occurs to prevent a system thrashing condition. (The page supervisor does not monitor the paging activity of individual programs, only of the entire system.)

Note that the PURGE TLB instruction is issued to invalidate all the entries in the translation lookaside buffer when task switching from one region to another occurs, at swap-out time for a TSO region, at task or job step termination time, and when the invalid bit is turned on in an entry in an active segment table or page table.

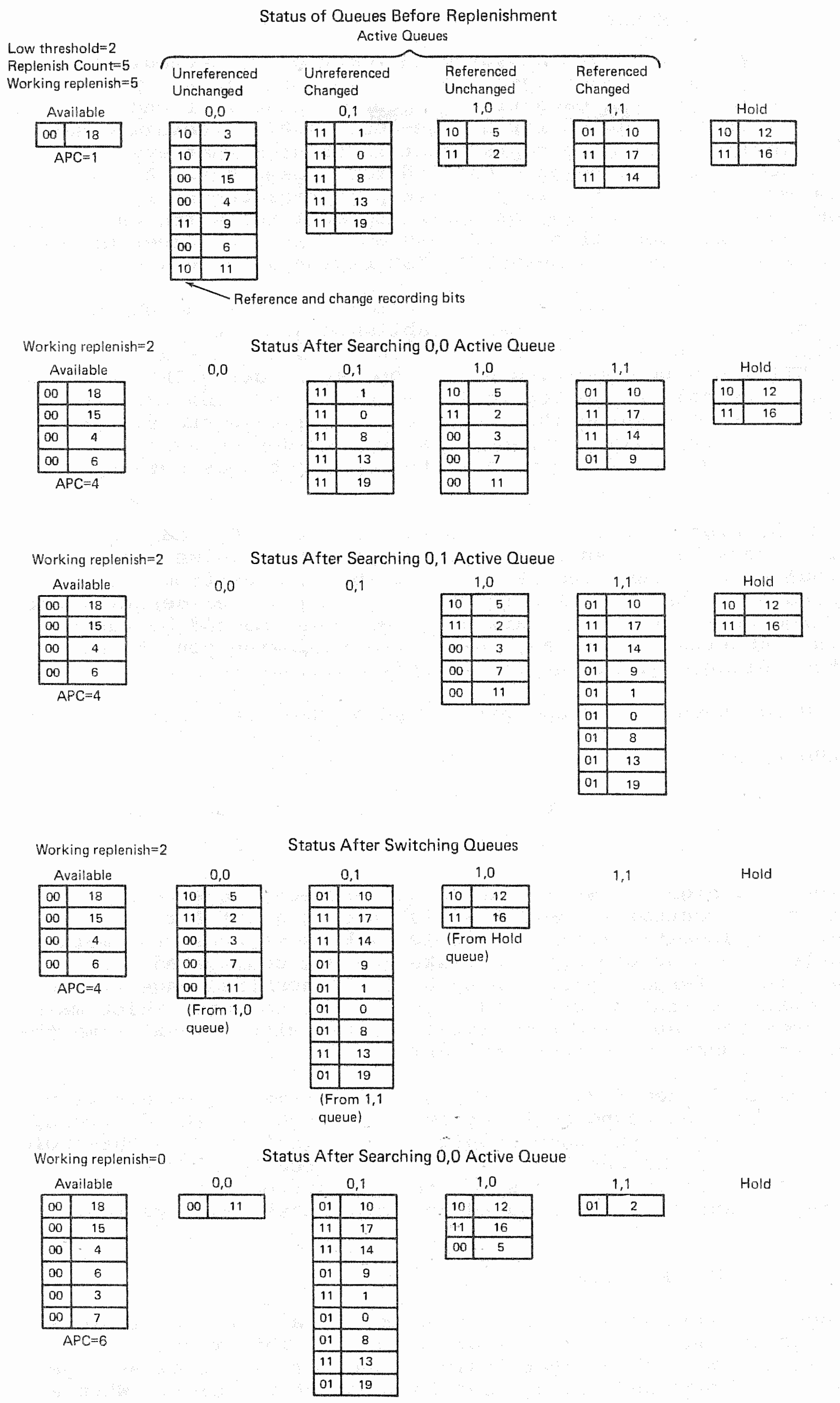


Figure 100.35.2. Operation of the page replacement algorithm

Task Deactivation and Reactivation

Paging activity for the entire system is measured by accumulating statistics regarding reclaimed pages and page-in operations. The reclaim count is incremented each time a required page is found to be in real storage and also scheduled for a page-out. A high reclaim count indicates the incidence of page replacement is so high that even frequently needed pages are being scheduled for a page-out. A low reclaim count may indicate that very little page replacement is occurring and that the system may be operating well below its capacity. The page-in count includes all page-ins and one-half the number of pages brought into real storage as a result of TSO region swap-ins.

Periodically, these two counts are inspected to determine whether they exceed the high threshold values established at system initialization. The interval of real time in seconds between inspections (MTIM) can be specified by the operator during IPL as can a high and a low threshold value for reclaimed pages (HRC and LRC) and a high and a low threshold value (HRD and LRD) for page-in and swap-in operations. These five values cannot be specified during system generation. System defaults are used if the operator does not supply these values.

MTIM can be a value from 1 to 9 and defaults to 1. HRC can be a value from 0 to 9999 and defaults to 20. LRC can be a value from 0 to 9999 and defaults to 5. HRD and LRD each can be a value from 0 to 9999. HRD defaults to 0 and LRD defaults to 9999. If the system defaults for the high and low threshold values are not used, care should be taken when selecting substitutes. If any one of the following conditions exist, the task disable algorithm is effectively disabled:

- LRC or LRD (or both) are 0 and either HRC or HRD (or both) are 9999
- LRC or LRD is 9999
- HRC is 0
- HRD is 1

When established high threshold values are exceeded, the task dispatcher is given control to select a task (not region) for deactivation. The lowest-priority pageable task is selected and marked nondispatchable. Nonpageable and TSO tasks are not considered for deactivation. The reference bits for each of the nonfixed page frames currently allocated to the deactivated task are set to zero, which makes these page frames available and most likely to be selected next time the page replacement algorithm inspects the PFTE queues.

When any task is in deactivation status at the time paging activity counts are inspected and found to be below their high threshold values, a check is made to see if the same counts are also below a low threshold value. When they are, the task dispatcher is entered to select a deactivated task for reactivation, since this condition indicates that paging activity is sufficiently reduced to permit task reactivation.

EXTERNAL PAGE STORAGE ADMINISTRATION

External page storage administration manages the allocation of slots in external page storage to virtual storage pages. This routine maintains external page tables that indicate whether a virtual storage page has a slot assigned and, if so, the location of the slot. When a page-in operation is required, these tables are inspected to determine the address of the assigned slot.

There is one external page table associated with and adjacent to the page table for each allocated segment of virtual storage that is pageable and for segments allocated to TSO LSQA's. These tables are contained in SQA or the appropriate LSQA. There are 16 entries, eight bytes in size, in each external page table (one for each virtual storage page in the segment the associated page table describes). In addition to indicating the location of an assigned slot, an external page table entry contains the storage protect key associated with its corresponding virtual storage page.

Bit maps, one per paging device, are used to indicate whether a slot in defined external page storage is assigned or available. When a slot must be allocated for a page-out operation, these maps are inspected during the selection of a slot. Bit maps are contained in the nucleus.

When a slot must be assigned, the external page storage algorithm attempts to select a device that will balance the usage of paging devices. When primary and secondary paging devices are defined, slot assignment is balanced among primary paging devices and secondary devices are not selected until the threshold value for available primary external page storage is reached. When there is more than one primary paging device, the one selected is the one with the most available slots. If the selected device has a movable arm, the algorithm attempts to select a slot that will minimize seek time. This is accomplished by selecting an available slot within or as close as possible to the cylinder at which the arm is currently positioned. If the selected paging device is a fixed head type, the algorithm attempts to select a slot that will minimize rotational delay by allocating the next available slot that will pass under the read/write heads.

A virtual storage page does not have a slot permanently assigned to it, except for those allocated to the pageable link pack area and its directory. Slot assignment is performed for a page every time a page-out operation is required for the page. A slot is selected that will minimize paging I/O time. Hence, each time a page is written out, a new slot may be assigned and, if so, the slot previously assigned is freed.

PAGE ADMINISTRATION

Routines in page administration are primarily responsible for initiating I/O operations (page-ins and page-outs) on paging devices and for performing the required processing when these I/O operations terminate. Page administration performs the following:

- Page I/O initiation. Channel programs for all paging operations are built according to a slot sorting algorithm (discussed below) that is designed to optimize paging operations. The EXCPVR macro is issued to initiate paging I/O requests and a special interface to IOS is used.

- Completion processing for paging operations. For example, when a page-in is completed, the reference and change bits for the affected page frame are reset, the PFTE for the page frame is placed in the hold queue, the appropriate page table entry is validated, and the task that was awaiting completion of the page-in operation is placed in the ready state. When a page-out completes, the affected page frame may be reclaimed, allocated to a nonpageable region, allocated as a reserved page frame for SQA and LSQA, or placed in the available queue.

- Determination of the location (virtual storage address) of the page table entry and the external page table entry for a virtual storage address when they are required (such as after a page fault occurs)

- Creation and destruction of page tables when these functions are requested by virtual storage management (discussed in Section 100:25)

- Release of page frames and slots currently assigned to an area of virtual storage. (PGRLSE macro processing is part of page administration.)

- Swapping of TSO region contents in and out of external page storage (described under "Time Sharing Option" in Section 100:20)

The slot sorting algorithm constructs and initiates channel programs for paging operations. The number of channel programs made available for exclusive use in paging operations is determined during IPL. These channel programs are located near the end of the resident nucleus. When a paging operation is required, the slot sorting algorithm places the required device-dependent information into an available channel program based on the address of the slot assigned by the external page storage algorithm. The channel programs built for paging operations contain real addresses. Rotational position sensing is utilized when the feature is present for the paging device.

When construction of a paging channel program has been completed, the slot sorting algorithm determines whether the paging operation is for a movable or fixed head device. If it is for a movable head device, the slot sorting algorithm determines whether the addressed paging device is busy. If not, the paging channel program is initiated on the device, using the new EXCPVR macro. If the paging device is busy, the slot sorting algorithm determines whether a paging operation is taking place in the cylinder addressed by the new paging channel program. If so, the new paging channel program is chained to the operational paging channel program. If not, the new paging channel program is placed in the appropriate slot queue for the addressed paging device without also being initiated.

A slot queue is a queue of channel programs that are to perform operations on slots that have the same slot number (but not group number) within the same paging device. The number of slot queues per paging device is equal to the number of slots per group for that direct access device type (three slot queues for a 3330-series device, eight slot queues for a 2314/2319 device, etc.). For a paging device with a movable arm, channel programs are placed in a slot queue in low-to-high cylinder address sequence. The channel programs for the same cylinder are then arranged in task priority sequence, with read requests having priority over write requests when task priorities are equal. Read requests or write requests with the same priority are queued in FIFO sequence.

When a paging channel program completes for a movable head paging device, a channel end appendage is entered. This routine posts the I/O operation complete, releases the paging channel programs, and attempts to start a paging channel program on the next highest addressed cylinder. The slot sorting algorithm is called to inspect the slot queues for the movable head paging device and construct a channel program chain for the next cylinder. The channel program chain is constructed such that rotational delay is minimized. Once a paging channel program has been constructed, it is initiated by the channel end appendage routine. Note that the paging channel program chain that is constructed in Release 1.6, for either a movable head or a fixed head paging device, does not contain PCI flags as does the paging channel program chain that is constructed in Release 1.

When a paging channel program is constructed for a fixed head paging device, the slot sorting algorithm initiates the channel program via the EXCPVR macro if the addressed paging device is not busy. If the fixed head paging device is operational, the paging channel program is chained to the operational channel program. A fixed head paging device is considered to have only one cylinder. Hence, a paging operation on a fixed head paging device terminates only when there is no paging to be done in the system.

Slot queues are also maintained for fixed head paging devices. Channel programs are placed in a slot queue for a fixed head paging device in task priority sequence, with read requests having priority over write requests when task priorities are equal. Read requests or write requests with the same priority are queued in FIFO sequence. When a paging operation terminates on a fixed head device, a channel end appendage is entered to post the I/O operation complete, release the paging channel programs, and start another paging I/O operation. The slot sorting algorithm is called to construct a channel program chain that consists of the paging requests in the slot queues for the fixed head paging device. This channel program chain is then started by the channel end appendage routine.

Page-out write operations are not verified for performance reasons. When a permanent write error occurs during a page-out, the slot involved is marked assigned so that it will not be reassigned. Another available slot is selected and the page is written in the new slot. The operator is informed that an error occurred on a paging device. When a permanent read error occurs during a page-in operation, the task associated with the page is abnormally terminated, the slot is marked assigned to prevent further allocation of the slot, and the operator is informed of the error.

100:40 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). All are standard in VS2. (APR is included automatically only when an alternate path to a device is specified.) The facilities provided by these routines are functionally equivalent to those supported by OS MFT and MVT RMS routines for System/370 models, with a few exceptions, as follows.

MCH routines are structured such that a VS2 control program generated for one System/370 model can be executed on other System/370 models supported by VS2. 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 a real storage error occurs have been made as well. When an uncorrectable real storage failure occurs after the IPL procedure has been completed, MCH determines whether the page frame in error is assigned to the resident nucleus, SQA, the paging channel program area, or LSQA. If so, system operation is terminated. Otherwise, if the page frame is allocated to a virtual storage page in the pageable dynamic area, MCH attempts to isolate the page frame involved so that it will not be allocated by real storage management, and an attempt to recover the contents of the damaged page frame is made. If the page was unchanged before the uncorrectable storage error occurred, it is assigned 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 was changed and it belongs to a system task, MCH determines whether the task is critical or noncritical. The system is placed in a wait state if the error is associated with a critical task. If the affected task is noncritical, it is marked nondispatchable and system operation continues.

In all cases, MCH determines whether the real storage error was intermittent or permanent. If the error is permanent, the PFTE for the affected page fram is marked not allocatable and removed from the queue of page frames that can be assigned to virtual storage pages so that it will not be reassigned. This action is not taken if the error was intermittent.

The resident portion of MCH is contained in the control program nucleus. The transient modules of MCH are contained in SYS1.SVCLIB. A central CCH routine is also part of the nucleus. The five channel-dependent CCH routines (for 2860, 2870, 2880, Model 158/155 II, and Model 145 channels) are contained in SYS1.LINKLIB. During IPL, the required channel-dependent module or modules, as determined by the STORE CHANNEL ID instruction, are fetched and made resident in the nucleus.

APR and DDR processing is equivalent in function in VS2 and MVT, except that DDR does not support the swapping of a system residence or a page data set volume in a VS2 environment.

Power Warning feature support (also available in Releases 21.6, 21.7, and 21.8 of MVT) is a system generation option that is provided for Model 158 and 168 users that have the optional Power Warning feature and uninterruptible power supplies installed for their systems. When this support is included in a generated VS2 system and a warning machine check interruption occurs as a result of a power disturbance, actions are taken to prevent a system termination, if possible, or to save the contents of real storage on disk before a termination occurs so that system operations can be restarted when normal power is restored.

Power Warning feature support requires a system to have uninterruptible power supplies for only a critical subset of the hardware configuration, that is, the central processing unit, all channels, two 3330-series drives, and the critical paths to these disk drives. When a system is fully protected by uninterruptible power supplies, system termination can be avoided after a power disturbance of short enough duration, that is, a disturbance of shorter duration than the interval of time during which the system can be powered by the reserve power supply.

When Power Warning feature support is present in the generated VS2 control program, two warn data sets, SYS1.WARNA and SYS1.WARNB, must be allocated on two separate 3330-series volumes of the same model (1, 2, or 11). One of these can be the system residence volume in VS2. Each data set must consist of one contiguous extent of cylinders that is large enough to contain the entire contents of real storage. When required, the contents of real storage are dumped to SYS1.WARNA, which is the primary warn data set. The alternate warn data set, SYS1.WARNB, is used when the primary data set is not accessible.

The two warn data sets must be placed on two 3330-series drives that have an uninterruptible power supply. The 3330-series drives that have an uninterruptible supply are identified by the user during system generation. During system initialization, the initialization routines ensure that the two warn data sets are mounted on 3330-series drives that have uninterruptible supplies and that a channel path to these devices exists. The DDR routine ensures that a volume with a warn data set is swapped only to another 3330-series device that has an uninterruptible power supply. Both warn data sets are always formatted during system initialization for data security reasons. The operator is notified if a warn data set contains dump data that was not used during a system restart. The operator must indicate that this dump data is to be restored or give control to a system routine that erases and reformats the warn data set. A user-written routine can be included in VS2 that will be executed before the warn data sets are erased and reformatted.

When a warning machine check interruption occurs, one of the following steps is taken, depending on the user specification at system generation for Power Warning feature support:

- A real storage dump routine is entered immediately after the warning interruption. This routine dumps the contents of real storage to disk and terminates system operations. This option should be chosen when an I/O device critical to the operation of the system does not have an uninterruptible power supply.
- A timing routine is entered after the warning interruption occurs to determine whether the disturbance is transient before any other action is taken.

When the second option listed above is chosen, a system-supplied timing routine executes for an interval of time called the time delay interval. This interval is user specified at system generation. The length of the time delay interval specified should be limited by the amount of reserve power provided by the installed uninterruptible power supplies. During the time delay interval, the timing routine constantly enables the system for warning and all exigent machine check interruptions. The power disturbance is determined to be transient or nontransient, depending on the interruptions received during the time delay interval after each enabling of the system.

If a warning interruption occurs after an enabling, the timing routine continues the enabling procedure. The disturbance is considered to be transient if a warning interruption is not received after an enabling, which indicates utility power has been restored. If a warning interruption is still occurring at the end of the time delay interval, the disturbance is considered to be nontransient. The disturbance is also considered to be nontransient if an exigent machine check interruption occurs during the time delay interval. Operation of the timing routine terminates immediately after the occurrence of an exigent machine check interruption even though the time delay interval has not expired.

When a disturbance is determined to be transient, the warning condition is treated as a repressible machine check condition. MCH logs the error and returns control to the VS2 supervisor so that system operations can continue. When a power disturbance is found to be non-transient, control is passed to a user-written routine, if one is present, or to the real storage dump routine which will write the entire contents of real storage in a warn data set. System operation is terminated after the dump is taken. If a user-written routine is executed after a nontransient power disturbance occurs, the routine can determine whether system termination is required. The user-written routine can return control to the VS2 supervisor so that system operations continue or to the dump routine as required.

When a power disturbance causes a system termination, system operations can be restarted using the contents of the warn data set after normal power is restored. The routine that restores the contents of real storage is contained in SYS1.LINKLIB. When the operator indicates that restoration of real storage is to occur during IPL, the restore routine is loaded into the real storage. It reads the contents of the warn data set and places the data in the real storage locations from which it was dumped. The system is placed in a disabled wait state after real storage has been successfully restored. System recovery and restart procedures can then be performed.

OLTEP

OLTEP is a standard feature of VS2, and it supports the same functions as OS MVT OLTEP. When controlling the execution of OLT's, OLTEP must operate in a nonpageable region in VS2. A minimum of 64K is required for the nonpageable region when 4K OLT's are executed. OLTEP can execute in a pageable region, however, when it is controlling execution of the logout analysis program for a Model 158, 155 II, 168, or 165 II. A pageable region of 192K minimum is required to execute a logout analysis program under OLTEP.

PROBLEM DETERMINATION FACILITIES

Service Aids

The service aids in VS2 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 region under VS2 control, except IMCOSJQD:

- AMAPTFLE 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 VS2 are supported (not supported in MVT). AMAPTFLE is functionally superseded by the system modification program (SMP), which is also provided for MVT (Releases 21.0, 21.6, 21.7, and 21.8). SMP provides an improved and more comprehensive method of applying system modifications (PTF's), component releases, and user modifications to VS2 distribution libraries and system libraries.

- AMBLIST replaces the IMAPTFLS and IMDMDMAP service aids in MVT 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
  A map of the pageable link pack area (does not include modifications to the pageable link pack area)

- AMASPZAP 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 partitioned data set, or any portion of a data set on a direct access device. Use of this service aid is restricted via APF.

- IMCOSJQD can be used to print the contents of SYS1.SYSJOBQE. This standalone program replaces the IMCJQDMP program provided in MVT.

- AMDSADMP 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 AMDPRDMP). The low-speed version prints the contents of the control registers and real storage or writes them to tape in unblocked printable format so the tape can be printed by IEBGENER or AMDPRDMP. The store status function must be performed by the operator before loading a standalone dump program.

- AMDPRDMP formats and prints a dump tape produced by a high-speed or low-speed version of AMDSADMP, the trace data gathered by the generalized trace function of GTF, the SYS1.DUMP data set, and the TSO DUMP data set. It also can be used to print selected pages from the page file. The VS2 AMDPRDMP service aid formats dumps created using the VS2 AMDSADMP dump routine only. It will not format a dump created using a VS1 dump routine.

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

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

- Generalized Trace Facility

  The general functions of GTF, as implemented in VS2, are the same as those for GTF operating under OS MFT or MVT. When executing in VS2, GTF uses the hardware monitoring facility and supports tracing of page fault interruptions.

  The generalized trace function of GTF must be initiated as a system task via a START command. A virtual storage region of 64K minimum is required. A larger region may be required depending on the number and size of the trace buffers used and the options selected. Parameters (events to be traced, definition of trace output data set, etc.) can be supplied to GTF via the START command or a SYS1.PARMLIB member. During its execution in VS2, the trace function requires a certain minimum of fixed real storage when trace data is contained in real storage and a larger minimum 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 AMDPRDMP service aid and is invoked as a problem program via job control. A minimum 128K byte pageable region is required for its execution. The trace EDIT function of VS2 will format only the trace data produced in a VS2 environment. It will not format data traced using GTF in VS1, MFT, or MVT environments. However, MVT programs that use the GTRACE macro can be executed under VS2 control without modification. If user-written EDIT exit routines are being used in MVT, they may require modification for operation in a VS2 environment because of differences in the format of trace data for system events.

  While GTF and the current MVT resident trace facility coexist in a VS2 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 VS2. Real storage and/or the contents of selected areas of virtual storage can be dumped in VS2.

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 VS2 control program. DSS uses program event recording hardware in its interface with the operational VS2 operating system. DSS is designed to be used by authorized personnel, such as an IBM FE Programming Systems representative.

The DSS user interfaces with DSS only via a required primary console device type (3210, 3215, Model 158 display console, 3066) 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 be altered also, except DSS, IPL, and NIP code.

- 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 procedures. The primary protection that DSS offers is the fact that only the primary system console can be used for DSS operations.

100:45 LANGUAGE TRANSLATORS, SERVICE PROGRAMS, AND EMULATORS

SYSTEM ASSEMBLER

The System Assembler is a standard component of VS2 (and VS1). It is the only language translator that is a standard component of VS2. Program product and Type I language translators that are to be used with VS2 must be obtained and added to the VS2 system after the desired VS2 control program has been generated. The System Assembler offers the same functions as 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). The System Assembler is a compatible subset of Assembler H.

The System Assembler and the OS Assembler H (Release 5) program product are the only OS Assemblers that support the following System/370 instructions:

CLEAR I/O	SET PREFIX
COMPARE AND SWAP	SET PSW KEY
COMPARE DOUBLE AND SWAP	SIGNAL PROCESSOR
INSERT PSW KEY FROM ADDRESS	STORE CLOCK COMPARATOR
LOAD REAL ADDRESS	STORE CPU ADDRESS
PURGE TLB	STORE CPU TIMER
RESET REFERENCE BIT	STORE PREFIX
SET CLOCK COMPARATOR	STORE THEN AND SYSTEM MASK
SET CPU TIMER	STORE THEN OR SYSTEM MASK

The System Assembler is packaged to cause fewer page faults in a paging environment than does Assembler F, and its modules are reentrant. Therefore, the System Assembler can be placed in the pageable LPA and shared by concurrently executing tasks. The System Assembler can operate in a pageable region of 64K; however, for more efficient operation, a region 128K or larger in size is required.

LINKAGE EDITOR

The VS2 Linkage Editor program is a standard component of VS2 (and VS1). It also can operate under OS MFT and MVT. A pageable region a minimum of 64K in size is required for its operation; however, a region of 118K or more is recommended for better performance.

The VS2 Linkage Editor supports the same facilities as OS Linkage Editor F. In addition, it is designed to operate in a paging environment, to support the authorized program facility (as previously described), and to provide two new features that can be used to reduce the paging and real storage requirements of programs.

The new features provided for use in minimizing paging activity and real storage usage are CSECT reordering and CSECT alignment on a page boundary. Linkage editor control statements can be included 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 MVT programs can be restructured (without a rewrite) for more efficient operation in a paging environment, if necessary. Linkage editor control statements can also be included that specify which control sections and common areas of a load module are to be aligned on a page boundary in virtual storage.

The VS2 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 and certain linkage editor program names. VS2 does not recognize IEWL440, IEWL880, or IEWL128 as linkage editor program names on EXEC statements. Only IEWL and LINKEDIT can be used in VS2 as linkage editor program names.

UTILITIES

The same utilities that are provided in MVT are available in VS2. 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 starter system on direct access volumes. The VS2 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 IPL program on track zero of an IPL volume, after the required IPL records and volume label(s). This function can be used to place an AMDSADMP dump program on disk so that it need not be IPLed from cards or tape. The disk volumes used to contain any user-written 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 zero.)

SORT/MERGE PROGRAMS

The OS Sort/Merge (5734-SM1) and OS/VS Sort/Merge (5740-SM1) program products can be used in a VS2 environment. Both programs can operate in the minimum size virtual region (64K). While the OS Sort/Merge program can operate in nonvirtual storage OS environments, the OS/VS Sort/Merge program operates only under VS1 and VS2 control.

The OS/VS Sort/Merge supports the same functions, facilities, and options that the OS Sort/Merge supports. In addition, the OS/VS Sort/Merge supports features and I/O devices not supported by the OS Sort/Merge and can also provide better performance via a new disk sorting technique for fixed- and variable-length records.

The OS/VS Sort/Merge is compatible with the OS Sort/Merge. Therefore, sort/merge control statements, job control, and user-written exit routines that are used with the OS Sort/Merge can be used with the OS/VS Sort/Merge without modification, except when they apply to direct access devices that the OS/VS Sort/Merge does not support.

Differences between the OS/VS Sort/Merge and the OS Sort/Merge are the following:

- The OS/VS Sort/Merge supports as intermediate storage for sorting, 3330-series Model 11 and 3340 disk storage units, which are not supported by the OS Sort/Merge. The OS/VS Sort/Merge does not support, for intermediate storage, 2311 and 2301 direct access storage, which are supported by the OS Sort/Merge. The OS/VS Sort/Merge supports up to 100 direct access devices for intermediate storage when the new disk sorting technique is used. Both programs support 2314/2319, 3330-series Models 1 and 2, and all 3400-series tape units (from 3 to 32) for intermediate storage.

- A new sorting technique is implemented in the OS/VS Sort/Merge. It is used for fixed- or variable-length records when disk storage is provided for intermediate work storage. This new technique takes advantage of any existing sequencing within the input file that is to be sorted. When the new technique is used, the input file is never written to intermediate disk storage if it can be completely contained in virtual storage.

- The OS/VS Sort/Merge is also designed for more efficient operation in a virtual storage environment and is reentrant. These features can provide better performance, depending on the ratio of virtual storage allocated to the sort/merge and the amount of real storage dynamically available for allocation to the sort/merge.

- The OS/VS Sort/Merge supports key-sequenced and entry-sequenced VSAM data sets as well as QSAM data sets as input and output for both sorting and merging operations. Any I/O devices supported by QSAM or VSAM can be used as the input or output device type. The OS Sort/Merge program supports only QSAM organization for input and output data sets. VSAM input data sets cannot be concatenated as can QSAM data sets. The minimum amount of virtual storage required by the OS/VS Sort/Merge is greater than 64K when VSAM data sets are sorted. Checkpoints cannot be written when the output from a merge operation is a VSAM data set.

- The OS/VS Sort/Merge provides the capability of invoking a merge operation within a sorting operation for either a QSAM or VSAM input data set. A new exit is provided that enables a user-written routine to supply input records to the invoked merge operation.

- The OS/VS Sort/Merge can sort records based on a user-specified collating sequence, which is not supported by the OS Sort/Merge.

- The OS/VS Sort/Merge handles up to 64 binary or character control fields, which can contain a total of 4092 bytes. The OS Sort/Merge handles up to 64 binary or character control fields with a total of 256 bytes.

- The OS/VS Sort/Merge provides the capability of maintaining the input order of records with equal control fields. This capability is not supported by the OS Sort/Merge.

- When fixed-length records are being sorted, the OS/VS Sort/Merge automatically provides a formatted dump of the sort/merge program when a program failure occurs that terminates sorting operations. This capability is not provided by the OS Sort/Merge program.

INTEGRATED EMULATORS

The following emulator programs that operate under MVT control will also operate under VS2 SVS control:

- 1401/1440/1460 emulator Version 2 operating on a Model 145, 155 II, or 158

- 1410/7010 emulator operating on a Model 145, 155 II, or 158

- 7070/7074 emulator operating on a Model 155 II, 158, 165 II, or 168

- 7080 emulator operating on a Model 165 II or 168

- 709/7090/7094/7094II emulator operating on a Model 165 II or 168

- DOS emulator Version 3 Level 2 operating on a Model 145, 155 II, or 158 to emulate a DOS Version 3 or 4 system (Releases 25, 26, and 27) or DOS/VS system (Releases 28 to 32).

The functions supported by the integrated emulator programs listed above when they operate under VS2 are identical to the functions supported by these emulators when they operate under MVT. 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 145 (GC20-1734)

- A Guide to the IBM System/370 Model 155 (GC20-1729)

- A Guide to the IBM System/370 Model 165 (GC20-1730)

Version 3 of the 1401/1440/1460 emulator program runs under VS2 SVS (and VS1) but not under MVT (or MFT). Version 3 supports the same facilities as Version 2 and provides the following new features:

- Support of multiple-volume 1400 files. Version 2 supports only single-volume 1400 files.

- Carriage control commands for the 1403 printer are emulated directly on the System/370 printer device.

- Stacker selection for the 3505 reader and 3525 punch is supported.

- A 1401/1440/1460 simulator is supported. This simulator is provided for use on Model 165 II and 168 Systems for which a 1401/1440/1460 Compatibility hardware feature is not available. This simulator can also be used under VS2 on Models 145, 155 II, and 158. The 1401/1440/1460 simulator supports the same functions as Version 3 of the 1401/1440/1460 emulator and requires 10K more of virtual storage.

The additional virtual storage is required for programming that provides the functions of the 1401/1440/1460 Compatibility feature.

All the integrated emulator programs for VS2 are pageable. The emulator interface, SVC 88, is standard in VS2; however, the desired emulator programs must be ordered separately, as for MVT. Emulator programs generated to operate on a Model 145, 155, or 165 under OS MVT control will operate on a Model 145, 158/155 II, or 168/165 II, respectively, under VS2 control. Emulator regeneration is not required.

100:50 OS MVT TO OS/VS2 SVS TRANSITION

VS2 SVS is designed to be upward compatible with MVT, as of Release 21.8, and, therefore, migration from MVT to VS2 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 VS2. System programmers must become acquainted with new interfaces to VS2 (SMF exits, for example). Operators must learn how to respond to new system messages, such as those related to paging devices, and must become familiar with changes to the IPL procedure. Application programmers should learn how to use some program structuring techniques that are designed to improve system performance in a paging environment. System designers must learn about the new factors that affect system performance in a VS2 environment so that the system can be designed and operated in a manner that will achieve the results desired.

Once the VS2 SVS environment to be supported has been determined, a system generation must be performed. A VS2 system control program is generated via a two-stage procedure, which is, in function, much like that required to generate an MVT control program. The system generation macros used to describe the desired control program are identical for MVT and VS2 for like functions. Some of the macros and parameters used in MVT are not required in VS2, while new macros are provided to describe additional or different functions of VS2 (paging devices, automatic priority group, etc.).

As for MVT, a complete, nucleus-only, or I/O-device-only VS2 generation can be performed. Processor-only generations, which are supported in MVT, are not supported in VS2. All OS program products and Type I and Type II components that are to be used with the generated VS2 SCP must be added to the VS2 operating system after its generation. Since processor generations for Type I language translators cannot be performed using a VS2 system, they must be done using OS MFT or MVT.

The VS2 Release 1.7 starter system (available on tape only for 2314/2319 or 3330-series residence) operates on any System/370 model with 384K or more of real storage that has the dynamic address translation feature, one console, one nine-track tape unit, one SYSOUT device, and four 2314/2319 or three 3330-series direct access storage devices. The VS2 starter system can only be used to generate a VS2 control program and is required only for the first generation. Thereafter, an existing VS2 control program can be used. The Model 158 display console can be used only in printer-keyboard mode when the starter system is being used for the generation.

If the existing VS2 system is not to be modified, the system generation can be performed concurrently with other executing jobs. The generated VS2 system can operate on any System/370 Model 145, 155 II, 158, 165 II, or 168 that has the hardware features and I/O devices required by the control program. The SECMODS parameter should be specified in the CENPROCS macro at system generation to cause inclusion in the operating system of the model-dependent EREP for the secondary models on which the VS2 control program is to be run, if any.

A new feature of the VS2 generation process is the installation verification procedure (IVP) which is designed to be performed after the VS2 control program is generated. The IVP involves executing an IBM-supplied job stream (maintained in the SYS1.SAMPLIB data set) using the generated VS2 system. The function of the IVP is to exercise the generated SCP system components to the degree that general operation of the operating system and support of the system hardware configuration specified are assured.

Existing user-written programs that operate under MVT on a System/370 model must be modified for correct operation under VS2 if they do any of the following (otherwise, user-written existing executable programs, that is 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.

- Use the trace EDIT exit of GTF, if fields are accessed whose location varies between MVT and VS2

- Depend on a nonstandard interface to the MVT control program. These programs may require modification, based on the specific dependency.

- Use QTAM to support teleprocessing operations. These programs must be altered to use TCAM since QTAM is not supported in VS2. Minimal effort is required for this modification. (See <u>OS/VS TCAM Programmer's Guide</u>, GC30-2034, for a discussion of running QTAM application programs under TCAM.)

- Modify an active channel program by data being read (channel program contains self-modifying CCW's) or by executing instructions, if the program is to be run in a pageable region under VS2 control. Program modification is not required if such programs operate in a nonpageable region. This situation can apply only to user-written programs that use the EXCP macro instead of an access method. Such programs do not execute correctly because the modification affects the virtual channel program rather than the translated channel program that is actually controlling the I/O operation. (See <u>OS/VS Data Management for System Programmers</u>, GC28-0631, for a discussion of how to modify an EXCP program that contains dynamically modified channel programs so it can operate in paged mode.)

- Use the EXCP macro and contain user-written I/O appendages that can incur disabled page faults, if these programs are to be run in paged mode. Modification is not required to operate such programs in nonpaged mode. These programs operate in paged mode if they are altered to use the page fix appendage in order to fix the required pages.

In addition, the following must be done if applicable to the existing MVT 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 key instead of a five-bit key affects the processing being performed. If the SET 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 that are assembled using an OS MVT release prior to 20 and that use the teleprocessing facilities of this language translator must be reassembled and link-edited.
- TCAM message control programs must be reassembled 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.
- TSO system parameters must be modified as required to adhere to changes indicated in Section 100:20.

The job control statements for existing user-written problem programs do not require alteration except as follows:

- The ADDRSPC=REAL parameter must be added to the appropriate JOB or EXEC statements for programs that must operate in nonpaged mode if the IBM-supplied reader procedure is used. The REGION parameter for nonpaged job steps may have to be changed to request more space because of differences in subpool allocation within a region in VS2 and MVT. In addition, the default region size for a nonpaged job step in VS2 is 12K (plus track stacking), which is larger than the MVT region size default.
- The REGION parameter for job steps that use the rollout feature may have to be changed to specify a larger amount of storage, since rollout is not supported.
- EXEC statements that specify IEWL440, IEWL880, or IEWL128 as the linkage editor program name must be modified to specify IEWL or LINKEDIT. Alternatively, the three MVT linkage editor program names can be specified as aliases of IEWL or LINKEDIT to avoid job control changes.

If I/O device type changes are made and/or if unsupported device types, such as those listed in Section 100:05, are currently being used in an MVT environment, program and/or job control changes may be required to specify the supported I/O device that is used in a VS2 environment. Existing data sets can be used without alteration, assuming device type or access method changes are not made. If VSAM is to be used instead of ISAM, the affected data sets must be converted from ISAM format to VSAM format, as discussed in Section 100:30, and appropriate changes to existing ISAM job control statements must be made.

If desired, the structure of existing user-written MVT programs can be modified to minimize the occurrence of page faults and 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 VS2 environment.

As part of the transition process, MVT users with HASP II installed must perform a HASP II generation using Version 4. If the 7094 emulator is used in a VS2 environment and column binary cards are placed in the input stream, HASP II Version 4 must be used, because the VS2 reader interpreter does not support column binary reading of card SYSIN data sets.

VS2 can also be used in the main processor(s) and/or the support processor of an ASP Version 3.1 multiprocessing configuration and in a processor operating in local execution mode. VS2 and MVT (Releases 21.0, 21.6, 21.7, and 21.8) can be mixed in an ASP configuration (VS2 or MVT in the support processor and MVT or VS2 in the main processor). The ASP program operates in nonpaged mode under VS2 control.

For transition from a System/360 MVT environment to a System/370 VS2 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 145 (GC20-1734)
- A Guide to the IBM System/370 Model 155 (GC20-1729)
- A Guide to the IBM System/370 Model 165 (GC20-1730)

100:55 SUMMARY OF ADVANTAGES

As a growth system for OS MVT users, OS/VS2 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 MVT users. VS2 can be more responsive to a dynamically changing daily workload than MVT, and it supports an environment in which design changes can be made more easily to accommodate maintenance changes and the addition of new function or applications.

While OS/VS2 supports many new features, including functions exclusive to System/370 (not provided in System/360), such as EC mode and dynamic address translation, it remains upward compatible with MVT because existing standard interfaces have been preserved. Control program modifications that are required to handle new features are transparent to the user so that operators and programmers interface with VS2 using basically the same commands, job control statements, data sets, and programs they use in an MVT environment.

The single most significant new feature of VS2 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 Section 30:05). In addition to these, VS2 offers other specific advantages over MVT, several of which also result from the implementation of virtual storage. These advantages are summarized below.

Improved Job Scheduling

- Up to 63 initiators can be started, if enough resources are present, instead of a maximum of 15.
- Up to 42 foreground (TSO) regions can be started, instead of a maximum of 14.
- An initiator can have up to 15 job classes assigned instead of a maximum of 8.

- The operator can cancel a job during its initiation if it is waiting for data sets, region space, or external page storage to become available. The system automatically cancels a job that requires data sets that are permanently allocated to another job.

Operational Enhancements

- The operator is relieved of most real storage management functions (such as starting long-running jobs at certain times to avoid fragmenting real storage).

- High-priority jobs can be handled more easily. The operator can start an initiator that is to be used to initiate only high-priority jobs. Since real storage fragmentation is significantly reduced in VS2, there is more chance that the high-priority job can be initiated in a VS2 environment than in an MVT environment.

- System parameters can be modified at IPL by the operator more easily because of the addition of several new types of system parameter members to SYS1.PARMLIB. Default system parameters can be modified by changing the default SYS1.PARMLIB member and do not require a system generation.

- The VS2 starter system is independent of the direct access device types to be used during a system generation.

Improved System Integrity and Availability

- Fetch protection as well as store protection is provided for all regions.

- The new authorized program facility is supported to prevent unauthorized use of routines identified as having restricted access.

- Validity checking of data extent blocks (DEB's) is significantly expanded to prevent one job step from accessing data sets belonging to another job step (unless the data sets are to be shared).

- Total system terminations that result from the lack of available real storage for control blocks are minimized through enhanced SQA management and via the implementation of LSQA for all regions rather than TSO regions only.

- A module that checks for missing channel end and I/O device end interruptions during system operation is standard to prevent system waits, indefinite job step waits, and job step cancellations because of an uncompleted I/O operation.

- Additional error recovery procedures have been included in master scheduler tasks that are designed to prevent abnormal termination of these tasks when certain errors occur.

Improved Utilization of Real Storage

- Inefficient use of real storage caused by unused storage within the region size specified and/or residence of inactive pages of a program is eliminated. Unused virtual storage within a pageable region (either background or foreground) 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.

- Real storage fragmentation that occurs in MVT as regions of differing sizes are allocated and deallocated is eliminated because real storage is allocated on a demand basis, 4K at a time.

- Readers and writers are totally 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 for routines in the link pack area automatically increases and decreases 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 real storage allocated to system control blocks (LSQA and SQA) dynamically expands and contracts as required.

- Dynamic real storage management is provided for all programs that operate in paged mode in a VS2 environment, regardless of the language in which they are written. Dynamic serial program structure implemented via 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 MVT, are not supported by all high-level languages.

Performance Enhancements

- Improved utilization of real storage, as discussed previously, may enable a VS2 control program to operate in the same real storage configuration as an MVT control program and support (1) a higher level of multiprogramming, if more initiators can be kept active, (2) more TSO users, or (3) better response for TSO regions with the same number of TSO users.

- A new I/O load balancing algorithm is implemented to allocate I/O devices in such a way that I/O activity is more evenly distributed on available channels and contention among devices is reduced.

- A new task dispatching algorithm is provided that can increase system throughput by allocating CPU time to selected jobs (those in the APG) on the basis of their changing operational characteristics (more CPU-bound or more I/O-bound) rather than according to user-assigned priorities.

- The required link pack area is greatly expanded, and it includes most of the more frequently used control program routines. All transient SVC routines reside in the link pack area. SVC and I/O transient areas are not implemented in order to eliminate contention for these areas. Serialization of system processing that results from such contention in MVT is avoided. Less control program time is required to page-in transient SVC routines in VS2 than to fetch them in MVT.

- Serialization of command processing is minimized by executing command processing routines concurrently as subtasks of the master scheduler.

- The time required to process storage allocation (GETMAIN) requests for certain space in LSQA and SQA is reduced via implementation of quickcells in these areas.

- The time required to process timing queues during a task switch is reduced through implementation of new algorithms that support the CPU timer and clock comparator instead of the interval timer at location 80.

- Since real storage management is provided by VS2, problem programmers need not use LOAD, LINK, XCTL, GETMAIN, and FREEMAIN macros in new applications to efficiently manage real storage for their pageable regions, 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 online system debugging capability is provided by the dynamic support system.

The new facilities of OS/VS2 make it a desirable growth operating system for MVT users. However, many of the new features of VS2 make it more suited to an online environment than is MVT. Specifically:

- Reduction of real storage restraints made possible by the implementation of virtual storage can be an 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 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 MVT.

- 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 for an online or a data base environment than ISAM.

- A system operating with VS2 should be less susceptible to the total termination of operations because of certain improvements made in the VS2 control program. System integrity and protection 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.

INDEX (Section 100)

GC20-1753-1

OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement Printed in U.S.A. GC20-1753-1

IBM

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

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

READER'S COMMENT FORM

GC20-1753-1

OS/Virtual Storage 2

Release 1

Features Supplement

Release 1.7

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
FOLD ON TWO LINES, STAPLE AND MAIL.

GC20-1753-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

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

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Systems — Dept. 824

Fold Fold

IBM®

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

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

OS/Virtual Storage 2 Release 1 Features Supplement Release 1.7 Printed in U.S.A. GC20-1753-1

READER'S COMMENT FORM

OS/Virtual Storage 2 GC20-1753-1

Single Virtual Storage (SVS)

Features Supplement

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold fold

fold fold

• Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
FOLD ON TWO LINES, STAPLE AND MAIL.

GC20-1753-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold　　　　Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessay if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Systems – Dept. 824

Fold　　　　Fold

IBM

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

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

OS/Virtual Storage 2 Single Virtual Storage (SVS) Features Supplement Printed in U.S.A. GC20-1753-1