GC20-1752-3
File No. S/370-20

Systems

OS/Virtual Storage 1 Features Supplement

Release 6

This supplement discusses OS/Virtual Storage 1 (OS/VS1) features and organization as of Release 6. Only concepts and functions of OS/VS1 that are new to and significantly different from those of OS MFT are presented in detail. Transition from OS MFT to OS/VS1 is discussed also.

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/VS1 discussion presented: *A Guide to the IBM System/370 Model 135* (GC20-1738), *A Guide to the IBM System/370 Model 138* (GC20-1785), *A Guide to the IBM System/370 Model 145* (GC20-1734), *A Guide to the IBM System/370 Model 148* (GC20-1784), *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* (GC20-1787).

Readers who possess more than one of the above base publications need add this supplement to only one of the documents, as the OS/VS1 information presented applies to System/370 Models 135, 138, 145, 148, 158, and 168 unless otherwise indicated in the text.

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

IBM

PREFACE

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

Base publications for the OS/VS1 supplement are:

- A Guide to the IBM System/370 Model 135 (GC20-1738-4 or later editions)
- A Guide to the IBM System/370 Model 138 (GC20-1785)
- A Guide to the IBM System/370 Model 145 (GC20-1734-2 or later editions)
- A Guide to the IBM System/370 Model 148 (GC20-1784)
- 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 (GC20-1787)

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

Fourth Edition (January 1977)

This edition is a major revision obsoleting GC20-1752-2. The text and illustrations have been updated to reflect the new features of Release 6 of OS/VS1. Changes are indicated by a vertical line in the left margin.

CONTENTS (Section 90)

FIGURES (Section 90)

TABLES (Section 90)

SECTION 90: OS/VIRTUAL STORAGE 1 FEATURES

90:05 FUNCTIONS AND HARDWARE SUPPORTED

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

- Support of one virtual storage of up to 16,777,216 bytes using dynamic address translation hardware
- More efficient peripheral I/O operations processing provided by the job entry subsystem (JES), which replaces OS MFT input readers and output writers and incorporates many of the spooling features of MFT HASP II
- Improved remote job entry provided by the remote entry services (RES) facility, a logical and functional extension of JES. RES replaces OS RJE and provides functions similar to those available in HASP II RJE in addition to support of system network architecture (SNA) terminals.
- Job scheduling enhancements that include elimination of small partition scheduling, significant reduction in contention for the job queue, and a new I/O device allocation technique
- Performance enhancements, such as dynamic task dispatching
- An additional access method, called Virtual Storage Access Method (VSAM), that is designed to offer more functions and to be more suitable to online and data base environments than ISAM
- An additional access method, called Virtual Telecommunications Access Method (VTAM), that supports network control program mode for 3704 and 3705 Communications Controllers and provides facilities not available in BTAM or TCAM
- Operational enhancements, such as more automated system initialization and new and expanded operator commands
- Improvements in system integrity and data security protection, such as additional protection of control blocks within a problem program partition, fetch protection, and an authorized program facility

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

OS MFT is upward compatible with OS/VS1 to the extent that moving to OS/VS1 resembles moving from one release of MFT to another that contains significant new features. (See Section 90:60 for a discussion of MFT to OS/VS1 transition.) Except for JES and RES, OS/VS1 is upward compatible with OS/VS2 SVS in the same way that MFT is upward compatible with MVT.

DOS Version 3 and 4 and DOS/VS users can make the transition to OS/VS1 with the aid of the OS/DOS emulator, which operates under OS/VS1 control. Compatibility that exists between DOS and DOS/VS files and MFT data sets also exists between DOS and DOS/VS files and OS/VS1 data sets. Because of the compatibility between OS/VS1 and MFT, the effort required to convert from DOS or DOS/VS to OS/VS1 is similar to that required to convert from DOS to MFT.

OS/VS1 is an intermediate-level operating system, classified as system control programming (SCP) and hereafter referred to as VS1 or OS/VS1, that supports System/370 Models 135 (Models 0 and 3), 138, 145 (Models 0, 2, and 3), 148, 155 II, 158 (Models 1 and 3), 165 II, and 168 (Models 1 and 3). VS1 supports these models operating in EC and translation modes.

VS1 does not support System/370 models operating in EC mode without address translation operative, System/370 models operating in BC mode, or any System/360 models. VS1 does not support Model 158/168 Attached Processor Systems or Model 158/168 tightly coupled multiprocessing systems operating in multiprocessor mode but can be used when these systems operate in uniprocessor mode.

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

- 144K of real storage for a Model 135, 160K of real storage for a Model 145, and minimum real storage size for the other System/370 models supported by VS1 (configurations with less than 160K are subject to certain restrictions)

- Byte multiplexer channel with associated I/O devices, including one reader, one punch, one printer, and one console

- One selector or block multiplexer channel (or IFA for Models 135, 138, and 145 Model 0 or 2) with associated direct access devices that include at least three 2314/2319, two 3330-series (any model), two 3340/3344 (any model), or two 3350 disk drives. The preceding direct access devices and the 2305 Model 2 are supported for system residence.

- Dynamic address translation and channel indirect data addressing

- Storage protection

- Interval timer (at location 80) and time-of-day clock

- Monitoring facility

- Program event recording

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

- A VS1 system generation cannot be performed using the starter system (160K minimum is required).

- A maximum of 2048K of virtual storage is supported.

- The recommended maximum number of partitions is two. Only one partition can be defined if RES is included in the generated VS1 system.

- If RES is included in the system, it can support only one binary synchronous communications line. Support of SNA terminals requires a minimum of 256K.

- The Generalized Trace Facility (GTF) can operate but the external trace option cannot be used.

- The Conversational Remote Job Entry (CRJE) facility cannot be used.

- VTAM cannot be used (a minimum of 256K of real storage is required).

- The 3600 Finance Communication System is not supported (a minimum of 256K of real storage is required for support of this system).

The standard features of VS1 and a minimal I/O configuration are supported in a system with 160K of real storage. Inclusion of optional features in the generated VS1 control program, support of a larger than minimal I/O configuration, and improved system performance require a system with more than 160K of real storage. Note that a Model 145 with a GE storage size will have less than 160K of real storage available when the control storage requirement exceeds 32K.

The advantages and significant new functions offered by a VS1 virtual storage environment and optimal system performance can best be attained for a Model 135 if 240K or more is installed, and for a Model 145 if 256K or more is installed. The maximum amount of real storage supported by VS1 is 8192K bytes.

Table 90.05.1 lists the standard and optional features of OS/VS1 and Table 90.05.2 lists the I/O devices, consoles, and terminals supported. Just as for an OS MFT operating system, the desired installation-tailored OS/VS1 control program must be generated, at which time user-selected optional features are included in the resulting system. More features are standard in VS1 than in MFT. This enhancement can reduce the number of options that must be specified and, thereby, reduce system generation preparation and execution time.

OS/VS1 is a functional extension of OS MFT (as of Release 21.8). However, the following MFT features are not available in VS1:

- Storage hierarchies (2361 Core Storage cannot be attached to any System/370 model). Hierarchy parameters are processed at link-edit time but are ignored during program loading.

- TESTRAN

- QTAM (function provided by TCAM and VTAM), Graphic Job Processor (GJP), and Satellite Graphic Job Processor (SGJP)

- RJE (function provided by RES)

- IEBUPDAT utility (replaced by IEBUPDTE)

- IMAPTFLE (replaced by HMAPTFLE), IMDMDMAP (replaced by HMBLIST), IBCRCVRP (replaced by IEHATLAS), and IMASMP (replaced by HMASMP)

- IHGUAP utility

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

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

Table 90.05.1. Standard and optional features of OS/VS1. (Standard features are automatically included during system generation. Optional features must be requested during system generation or added after the generation is performed. An asterisk identifies features not available in MFT.)

Standard Features
• One virtual storage of up to 16,384K bytes with 64K segments and 2K pages* • Up to 52 partitions: any combination of 1 to 15 problem program and 1 to 52 system task* • Demand paging for allocation of real storage* • Execution of programs in paged* and nonpaged (V=R) modes • Paging algorithm tuning (PAGETUNE command)* • Independent job scheduling* • Job entry subsystem (JES)* • Remote entry services (RES)* • Automated system initialization* • Installation-specified selection parameters (ISSP)* • Multitasking (resident ATTACH) • Storage protection • Timing facilities (support of date and time of day, job step timing and limiting including the changeable wait limit capability, and interval timing) • Extended timer support (standard for all supported models except Models 135 Model 0 and 145 Models 0 and 2)* • System log (can be excluded) • Resident BLDL table (either pageable or fixed) • Resident access methods (both pageable and fixed) - included whether or not option is user-specified • Resident reentrant modules from SYS1.LINKLIB and SYS1.SVCLIB (both pageable and fixed) - included whether or not option is user-specified • Resident IDENTIFY, EXTRACT, SPIE • Extended fixed list* • Standard fetch and multiple wait • Fast multiple wait (EVENTS macro)* • Fast start function (STARTF command)* • Basic overlay supervisor • Operator communication at IPL • Missing interruption checker routine* • Authorized program facility (APF)* • Validity checking (for GETMAIN, FREEMAIN, POST, and WAIT macros) • Extended DEB validity checking* (can be excluded) • Access methods: BSAM, QSAM, BDAM, BPAM, VSAM* (VSAM can be excluded) • Direct access volume serial number identification (can be excluded) • Dynamic support system (DSS)* • Checkpoint/restart • Error statistics by volume (can be excluded) • Online test executive program (OLTEP) • Machine check handler (MCH) and channel check handler (CCH) • Alternate path retry (APR)- effective only if alternate channel paths are specified • Dynamic device reconfiguration (DDR) - can be excluded • Integrated emulator interface • System Assembler, Linkage Editor, and Loader • System utilities: IEHDASDR, IEHPROGM, IEHMOVE, IEHIOSUP, IFHSTATR, IEHATLAS, IEHLIST, IEHINITT, IEHUCAT • Data set utilities: IEBCOPY, IEBTCRIN, IEBPTPCH, IEBGENER, IEBCOMPR, IEBEDIT, IEBUPDTE, IEBISAM, IEBDG, IEBIMAGE* • Access Method Services (for VSAM)* • Account Facility Program (IKJRBBMP) for RES* • Independent utilities: IBCDMPRS, IBCDASDI, ICAPRTBL, IAPAP100 • Service aids: HMAPTFLE, HMASMP, HMASPZAP, HMDSADMP, HMBLIST, HMDPRDMP, IMCJOBQD, IFCDIP00, IFCEREP0, IMCOSJQD, Generalized Trace Facility (GTF)

Table 90.05.1 (continued)

- Industry Subsystem Support (for the 3600 Finance Communication System, 3650 Retail Store System, 3660 Supermarket System, and 3790 Communication System)*

Optional Features
• Resident transient SVC table • Resident type 3 and 4 SVC routines (both pageable and fixed) • Resident ERPs (fixed only) • User-written SVC routines resident in the generated nucleus • PCI fetch • TRACE function • Dynamic dispatching* • Fetch protection* • I/O load balancing* • Extended timer support (optional for Models 135 and 145 only)* • Advanced overlay supervisor • ASCII translation routine • Alternate channels (one for the 2305 and up to three for 2314/2319, 3330-series, 3340/3344, 3350, and 3330 virtual volumes) • Alternate and/or composite console • Multiple Console Support (MCS) • Device Independent Display Operator's Console Support (DIDOCS) • Remote terminal access method (RTAM) for RES* • Automatic volume recognition (AVR) • Time slicing • Job log facility* • System Management Facilities (SMF) • Access methods: VTAM*, ISAM, BTAM, TCAM • Shared Direct Access Storage Devices (DASD) for 2314/2319, 3330-series (all models), 3340/3344 (all models), 3350, 2305 Models 1 and 2, and 3330 virtual volumes • Graphic programming services (GPS) • Conversational remote job entry (CRJE) • Integrated emulators (independent component releases) 1401/1440/1460 emulator for Models 158, 155 II, 148, 145, 138, and 135 1410/7010 emulator for Models 158, 155 II, 148, and 145 7070/7074 emulator for Models 165 II, 158, and 155 II 7080 and 709/7090/7094/7094II emulators for Models 168 and 165 II DOS emulator for Models 158, 155 II, 148, 145, 138, and 135 • Programmed-function keyboard (PFK) command entry and/or light pen command entry support for graphic console devices • Reliability data extractor • Reduced error recovery for magnetic tape • Error volume analysis (EVA) • Teleprocessing Online Test Executive Program (TOLTEP) for terminals supported by VTAM* • Power Warning feature support for Models 158 and 168* • Automatic device status initialization (smart NIP routine) • Distributed Intelligence System (for System/7 processors)* • Extended Control Program Support feature support for Models 135-3, 138, 145-3, 148, and 158)* • VM/370 Handshaking* • 3850 Mass Storage System support* • Channel-to-channel communication*

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

<u>Readers and Punches</u>

1442 Card Read Punch, Models N1 and 2
2501 Card Reader, Models B1 and B2
2520 Card Read Punch, Models B1, B2, B3
2540 Card Read Punch
2596 Card Read Punch (96 column) as 1442 equivalent
3505 Card Reader (including optical mark reading)
3525 Card Punch (including card reading and card printing)

<u>Note</u>: Column binary reading and punching is supported for card devices with the card image feature.

<u>Printers</u>

1403 Printer, Models N1, 2, 3, 7
1443 Printer, Model N1
3203 Printer, Model 4
3211 Printer
3284 Printer, Models 1 and 2
3286 Printer, Models 1 and 2
3800 Printing Subsystem

<u>Diskette Devices</u>

3540 Diskette Input/Output Unit (as a SYSIN or SYSOUT device only and by the COPY/RESTORE utility program. The 3540 is not supported as an I/O device that can be accessed directly by problem programs using an access method.)

<u>Direct Access Storage</u>

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
3340 Direct Access Storage Facility, all models
3344 Direct Access Storage, all models
3350 Direct Access Storage, all models (in native and 3330 compatibility modes)
2305 Fixed Head Storage Facility, Models 1 and 2

<u>Note</u>: All the direct access devices listed above except the 2305 Model 1 are supported for system residence, as paging devices, for spool data sets, the SYS1.SYSJOBQE data set, the SYS1.SYSWADS data set, SWADS data sets, and other system (SYS1.) data sets. The 2305 Model 1 is supported only as an input/output device by the disk access methods, as a SYSIN/SYSOUT device, and by Shared DASD support. All the other direct access devices listed are also supported as SYSIN/SYSOUT data sets. Direct access devices attached to a byte multiplexer channel are not supported. All channel-switching features and string-switching features for the above direct access storage devices are supported for alternate channel paths and alternate control unit paths. The rotational position sensing, 32-Drive Expansion, and 3333/3340 Intermix features are also supported.

<u>Mass Storage</u>

3850 Mass Storage System

Table 90.05.2 (continued)

Magnetic and Paper Tape

- 2400-series and 3400-series magnetic tape units, all models, densities, and switching units (attachment via a byte multiplexer channel is not supported)
- 2495 Tape Cartridge Reader
- 2671 Paper Tape Reader

Optical and Magnetic Character Readers

- 1287 Optical Reader
- 1288 Optical Page Reader
- 1419 Magnetic Character Reader (Dual Address Adapter and Expanded Capability feature required)
- 3886 Optical Character Reader
- 3890 Document Processor

Display Units

- 2250 Display Unit (attachment via a byte multiplexer channel is not supported)
- 2260, 2265 Display Stations
- 3270 Information Display System

Primary Consoles

- 3210, 3215 Console Printer-Keyboards
- Display consoles for Models 138, 148, and 158
- 3066 System Console for Models 168 and 165 II
- Composite console (card reader and printer)

Alternate and Secondary Consoles

- 2150 Console with 1052 Printer-Keyboard Model 7
- Composite console (card reader and printer)
- 2250 and 2260 display units (locally attached)
- 2740 Communication Terminal
- 3277 Display Station

Hard-copy Consoles

- 3213 Printer (for Model 158 display console)
- 3286 Model 2 Printer (for the Model 138 and 148 display console)
- System log in SYS1.SPOOL data set
- Any console device with output capability (MCS support required)

Transmission Control Units

- 2701 Data Adapter Unit
- 2702 and 2703 Transmission Controls
- 2715 Transmission Control Unit
- 2772 Multi-Purpose Control Unit
- 2955 Data Adapter Unit
- 3704, 3705-I, and 3705-II Communications Controllers and channel-switching features
- 5098-N05 Sensor-Based Control Unit (required by Distributed Intelligence System support)
- 7770 Audio Response Unit

Table 90.05.2 (continued)

<u>Terminals</u> (<u>Start/Stop</u>)

1030 Data Communication System
1050 and 1060 Data Communication Systems
2721 Portable Audio Terminal
2740 Models 1 and 2, and 2741 Model 1 Communication Terminals
2760 Optical Image Unit
3767 Communication Terminal (as a 2740 Model 1 or 2, or a 2741)
83B3 AT&T Terminal
WU115A Teletype
TWX-33/35 AT&T Teletype Terminal
System/7 Sensor-Based Information System
Communicating Magnetic Card Selectric Typewriter

<u>Terminals</u> (<u>Binary</u> <u>Synchronous</u>)

2770 and 2790 Data Communication Systems and 2798 Guidance Display Unit
2780 Data Transmission Terminal
2972 Models 8 and 11 General Banking Stations
2980 General Banking Terminal System
3270 Information Display System (also locally attached)
3650 Retail Store System (as a System/3)
3660 Supermarket System (as a System/3)
3670 Brokerage Communication System
3735 Programmable Buffered Terminal
3740 Data Entry System
3770 Data Communication System (as a 2772/3780)
3780 Data Communications Terminal (as a 2772)
1130 System (as a processor station)
1800 System (as a processor station)
System/3 (as a processor station)
System/7 (as a System/3)
System/32 (as a System/3)
System/360 Models 20 and up (as a processor station)
System/370 models (as a processor station)

<u>Terminals</u> (<u>Synchronous</u> <u>Data</u> <u>Link</u> <u>Control</u>)

3270 Information Display System (SNA)
3600 Finance Communication System
3650 Supermarket System
3767 Communication Terminal
3770 Data Communication System
3790 Communication System
System/32 (as a 3770)

<u>Note:</u> 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 or programs may have on such terminals.

90:10 ORGANIZATION AND INITIALIZATION OF STORAGE

VIRTUAL STORAGE ORGANIZATION

The organization and allocation of virtual storage in VS1 is reflected in tables and control blocks that are established at system initialization and maintained throughout system operation, as is done for main storage in MFT. However, in VS1, virtual storage that is allocated to pageable programs does not require the allocation of real storage until the virtual storage is actually referenced by executing code. The size of the virtual storage supported can be specified at system generation and changed by the operator during system initialization. Virtual storage size can vary from a minimum of 1024K bytes to a maximum of 16,384K (16,777,216) bytes and must be a multiple of 64K.

The default virtual storage size assumed at system generation is 2048K unless certain options are included in the generated VS1 control program. In a system with only 144K of real storage, 2048K is the maximum amount of virtual storage that is supported. A minimum virtual storage size of 2048K is required if VSAM is to be used in a one-partition system. However, if the resident transient SVC routines or resident access methods option is included in addition to VSAM, the minimum virtual storage size for a one-partition system is 3072K. A minimum of 3072K of virtual storage is also required for a multipartition VS1 system in which VSAM is included, VTAM is included, or industry subsystem support is to be used. (Industry subsystem support requires the inclusion of VSAM and VTAM.)

Virtual storage in VS1 is divided into two main areas: a nonpageable area in lowest addressed virtual storage and a pageable area in highest addressed virtual storage, as shown in Figure 90.10.1. The two areas are divided by the virtual-equals-real (V=R) line. Storage protection, functionally equivalent to that implemented in MFT, is provided as a standard feature. Protect key values are assigned to virtual storage areas. Each time a page frame is allocated, its protect key is set equal to the protect key value assigned to the virtual storage page to which the page frame is allocated. Fetch protection for problem program partitions, not supported in MFT, is available in VS1 as an optional feature.

Nonpageable Area

The virtual storage in the nonpageable area is mapped on a virtual-equals-real 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 nonpageable area contains the resident control program, fixed system queue area (SQA), and V=R area. Included in the resident control program are the generated fixed nucleus, which is a minimum of 68K (for one partition, the minimum I/O configuration, and no options), and RMS routines (MCH and CCH).

The fixed nucleus in VS1 always contains the interruption supervisor, virtual storage supervisor, page manager, timer supervisor, nonpageable portion of the communications task, nonpageable portion of the master scheduler, I/O supervisor, BLDL routine of BPAM, and portions of the task supervisor and overlay supervisor. The two RMS routines require 6K of resident control program storage and 2K of pageable supervisor area when less than 192K of real storage is present, and 8K of resident control program storage when 192K or more is available. The resident control program (fixed nucleus and RMS routines) is assigned storage protect key 0 and is a multiple of 2K in size.

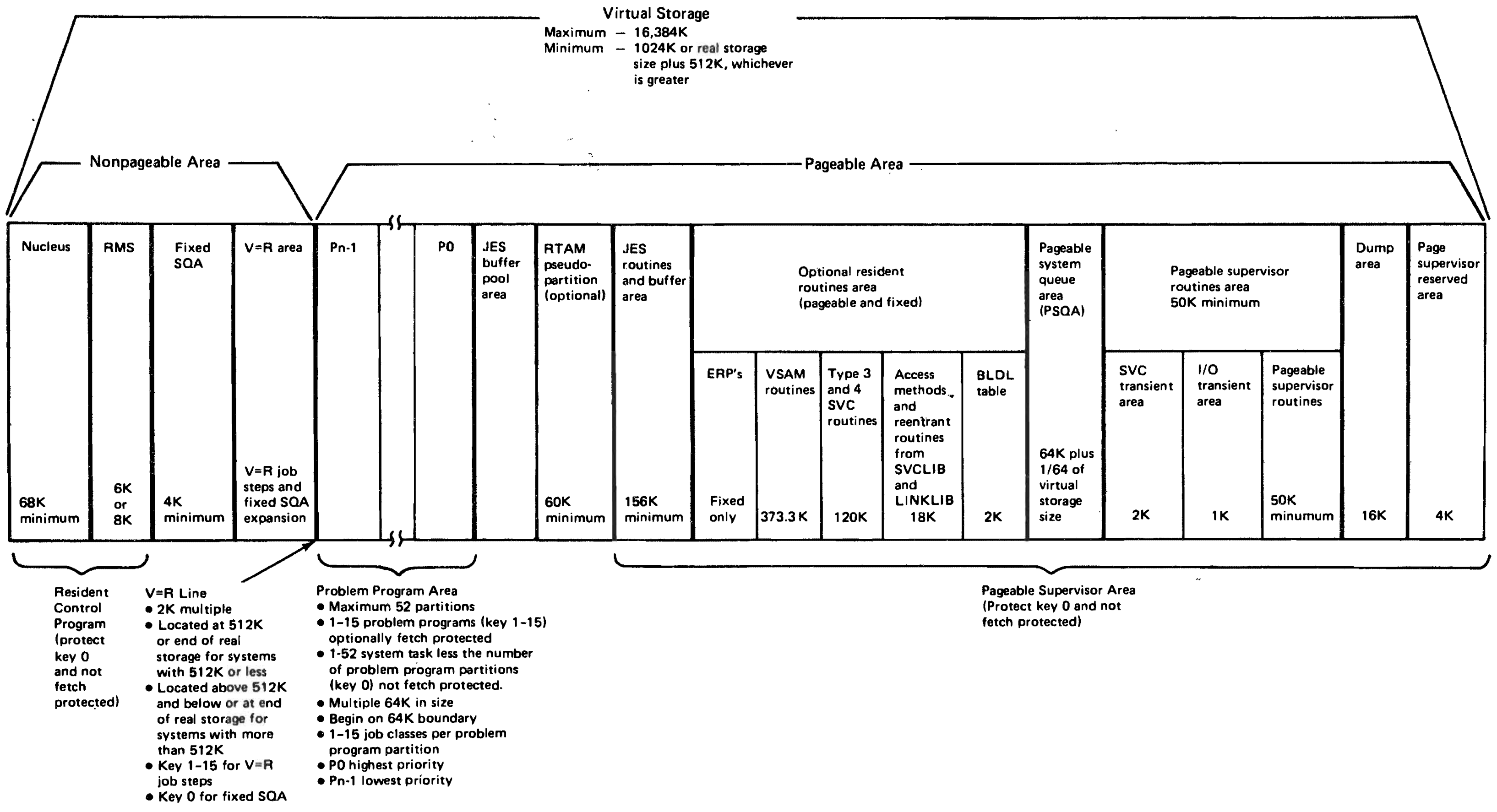

Virtual Storage
Maximum – 16,384K
Minimum – 1024K or real storage size plus 512K, whichever is greater
Nonpageable Area
Pageable Area
Nucleus
68K minimum
RMS
6K or 8K
Fixed SQA
4K minimum
V=R area
V=R job steps and fixed SQA expansion
Pn-1
P0
JES buffer pool area
RTAM pseudo-partition (optional)
60K minimum
JES routines and buffer area
156K minimum
Optional resident routines area (pageable and fixed)
ERP's
Fixed only
VSAM routines
373.3K
Type 3 and 4 SVC routines
120K
Access methods, and reentrant routines from SVCLIB and LINKLIB
18K
BLDL table
2K
Pageable system queue area (PSQA)
64K plus 1/64 of virtual storage size
Pageable supervisor routines area
50K minimum
SVC transient area
2K
I/O transient area
1K
Pageable supervisor routines
50K minumum
Dump area
16K
Page supervisor reserved area
4K
Resident Control Program (protect key 0 and not fetch protected)
V=R Line
• 2K multiple
• Located at 512K or end of real storage for systems with 512K or less
• Located above 512K and below or at end of real storage for systems with more than 512K
• Key 1-15 for V=R job steps
• Key 0 for fixed SQA
Problem Program Area
• Maximum 52 partitions
• 1-15 problem programs (key 1-15) optionally fetch protected
• 1-52 system task less the number of problem program partitions (key 0) not fetch protected.
• Multiple 64K in size
• Begin on 64K boundary
• 1-15 job classes per problem program partition
• P0 highest priority
• Pn-1 lowest priority
Pageable Supervisor Area (Protect key 0 and not fetch protected)

Figure 90.10.1. Virtual storage organization in OS/VS1

The standard extended fixed list facility of VS1, not supported in MFT, enables user-written CSECTs to be added to the generated fixed nucleus (SYS1.NUCLEUS data set) without the necessity of a nucleus generation. User-written CSECTs can be link-edited into the nucleus data set during or after a nucleus or system generation.

A CSECT named IEASPL11 that contains five V-type address constants for the default CSECTs IEAXYZ1 through IEAXYZ5 is IBM-supplied in the nucleus data set. If the user-written CSECTs that are to be added to the nucleus are assigned these names, reassembly of the IEASPL11 CSECT is not required. If other names are assigned to one or more user-written CSECTs, the IEASPL11 CSECT must be modified to include these user-assigned names, reassembled, and link-edited into the nucleus data set. The IEASPL11 CSECT is used during IPL to cause the user-written CSECTs it names via V-type address constants to be loaded into real storage as part of the fixed nucleus.

In VS1, when a trace table is defined, it is located adjacent to the resident control program and, optionally, its size can be varied. The trace table in VS1 is not included in the resident nucleus nor always fixed in size, as in MFT. The size of the trace table can be varied during VS1 system initialization only when the TRACE parameter is specified at system generation. The new TRACE system parameter can be entered by the operator or specified via the automated system initialization facility to override the number of 16-byte trace table entries specified at system generation.

The maximum size of the trace table is limited by the size of real storage and the resident control program. The trace table is built by the nucleus initialization program. The trace table and fixed SQA must fit in the real storage available during system initialization that is between the nucleus initialization program in upper real storage and the resident control program in lowest addressed real storage.

The trace table in VS1 contains SVC (SVC interruption), DSP (dispatching), I/O (I/O interruption), and SIO (START I/O instruction execution) entries, as does the MFT trace table. However, the VS1 trace table also contains program check interruption (PIO) and external (EXT) interruption entries, which are not present in the MFT trace table.

The fixed system queue area is adjacent to the trace table or to the resident control program if zero entries are specified for the trace table. Fixed SQA, with a protect key of zero, is used for control blocks and work areas that are not job or job step related or that must have virtual-equals-real storage addresses. The size of fixed SQA is specified at system generation and must be a minimum of 4K for a system with 144K of real storage. An additional 2K of fixed SQA should be allocated for each 64K of real storage present in excess of 144K up to a maximum of 64K of fixed SQA for systems with 512K or more.

The amount of fixed SQA specified is reserved in the nonpageable area of virtual storage during system initialization. However, during system operation, fixed SQA is dynamically expanded and contracted as needed, a virtual storage page (and page frame) at a time. When an allocated virtual storage page of expanded fixed SQA is no longer required, it and its allocated page frame are freed and become available for reassignment. The DISPLAY SQA command can be issued to cause the virtual storage addresses of the upper and lower boundaries and the amount of free space in fixed SQA to be displayed.

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

The size of the V=R area is variable by system. The V=R area consists of all the virtual storage from the end of defined fixed SQA to the location of the V=R line. The V=R line is established during system initialization when the amount of real storage present has been determined.

The default address of the V=R line in virtual storage is 512K or the address of the end of real storage, whichever is less. The operator can override the default location of the V=R line during system initialization only if the system has more than 512K of real storage. The address specified by the operator must be larger than 512K, less than or equal to the address of the end of real storage, and a multiple of 2K.

Note that since the nucleus must be totally contained in virtual (and real) storage below the V=R line, a VS1 nucleus cannot be larger than 512K, the default maximum value for the V=R line location. If a nucleus (IEANUC0X member) larger than 512K is required, the default 512K value in the IPL program can be modified by the operator during IPL (see OS/VS1 IPL and NIP Logic, SY24-5160).

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

The ADDRSPC and REGION defaults are specified in the PARM field of the EXEC statement in the reader procedure. The default for the ADDRSPC parameter is VIRT and the default size for a nonpageable job step is 50K bytes in the IBM-supplied reader procedure.

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

Jobs containing one or more steps that are to execute in nonpaged mode are scheduled (interpreted, initiated, terminated) using a pageable problem program partition in the pageable area that handles the job class indicated. That is, the scheduler operates paged in a pageable partition even though one or more of the job steps it schedules operate nonpaged in a contiguous area in the V=R area.

Although V=R job steps are not paged, they execute with translation mode operative, because they reference virtual storage addresses contained in the pageable supervisor area. Page tables associated with a nonpageable job step are established such that the real storage address that results from the translation of an address contained within the V=R area allocated to the job step is equal to the virtual storage address. Channel program address translation is not performed on CCW lists contained in a nonpaged program. A nonpaged job step must be restarted from a checkpoint in the same location in the V=R area that was used for the checkpoint.

Except for the resident control program in the nonpageable area of virtual storage (nucleus and RMS routines), none of the SCP components of VS1 must execute in nonpaged mode. However, any program that executes in VS1 must execute in nonpaged mode if it does one of the following:

- Contains a channel program that is modified while the channel program is active
- Is highly time-dependent (involves time-dependent I/O operations, for example, such as 1419, but not 3890, magnetic ink character reader programs)
- Must have all of its pages in real storage when it is executing (for performance reasons, for example)
- Uses the EXCP macro and executes user-written I/O appendages that can encounter a disabled page fault (Section 90:25 discusses disabled page faults)
- Uses the EXCP macro and has chained CCW lists with more than 239 CCWs

Existing user-written programs that are operating under MFT control and that must operate in nonpaged mode under VS1 control need not be modified in order to run in this mode. Existing optical character reader and certain types of card reader programs can be run in nonpaged mode under VS1 to maintain the performance achieved when they execute in an MFT environment.

A program that involves operations on a 1287 or a 1288 optical character reader will run slower in paged mode than in nonpaged mode. A program that accesses a card reader directly (does not use JES) and that uses the CNTRL macro, say for stacker selection, will probably run slower in paged mode than in nonpaged mode. The reduction in performance is the result of additional control program processing that is required to perform channel program translation and page fixing.

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

Pageable Area

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

The pageable supervisor area is located in highest addressed virtual storage and has a protect key of zero unless otherwise noted below. It contains the following in high-to-low virtual storage address sequence:

- An area of 4K reserved for the page supervisor
- A 16K-byte dump area that is used when an abnormal termination (ABEND) dump is to be taken for a pageable partition and the amount of virtual storage required for the dump is not currently available in the partition. This dump area consists of four pages of pageable partition queue area (PQA) and four pages of problem program virtual storage that is appended to the ABENDing partition. This dump area is used when required only to take an ABEND dump for an ABENDing job step task (not an ABENDing subtask). In addition, the dump area is not used for ABEND dumps of nonpageable partitions.

  Only one pageable partition can use the dump area at a time. If the dump area is being used by a pageable partition and another pageable partition requests access to the area, the requesting partition must wait until the dump area is released by the other partition. A queue of waiting partitions is maintained. The operator is notified when an ABENDing partition will have to wait for the dump area and must indicate whether the partition is to wait or the dump should be canceled. In the latter case, the partition is abnormally terminated without taking a dump.

  The dump area has a protect key of zero when it is not in use. When the dump area is being used, the four pages of problem program virtual storage have the protect key of the problem program partition that is being dumped. The fetch protect bit is on for these four pages if fetch protection is included in the system.
- The pageable supervisor routine area of approximately 50K minimum, which contains supervisor routines that can be paged instead of fixed with minimal effect on system performance. The size of this area varies depending on the system generation options selected (RTAM and VTAM, for example).

  This area always contains ATTACH, pageable portions of the communications task, DISABLE, enqueue/dequeue, EXTRACT, FIND/BLDL/CONVTTR, IDENTIFY, LINK/LOAD/XCTL/FINCH, pageable portions of the master scheduler, standard program fetch (unless PCI fetch is included), SEGLD/SEGWT, SPIE, SYNCH, and, if only one partition is defined, TIME. This area also contains the SVC transient area, which is 2K bytes (instead of 1K, as in MFT) and the I/O transient area of 1K.
- A pageable system queue area, which system tasks can use for their pageable virtual storage requirements. In general, any GETMAIN request issued by a system task that does not have its own pseudo partition is satisfied from pageable SQA. This area is implemented to reduce the fixed SQA space requirement. Pageable SQA also contains the resident job list.

  The default size of pageable SQA is determined by the size of the virtual storage supported and the maximum size of the resident job list. It is 64K bytes, plus 16 times the virtual storage size expressed in K bytes (or 1/64 of virtual storage size), plus maximum resident job list requirements. For example, the default pageable SQA size when virtual storage size is 2048K bytes is 96K bytes (64K bytes + (16 x 2048 bytes)) plus maximum job list size for this IPL as determined by the supplied job list definition parameters.

  Pageable SQA size can be specified by the operator during system initialization via the PSQA system parameter. The specified size overrides the default size unless it is less than the default size or not enough virtual storage is available to satisfy the request. An error message is issued in the latter situation. Pageable SQA cannot be extended during system operation. The system is abnormally terminated if pageable SQA space runs out during system operation.

- The resident supervisor area, which can contain the BLDL table, access method and other reentrant routines from SYS1.LINKLIB and SYS1.SVCLIB, type 3 and 4 SVC routines, and error recovery procedures (ERPs). The resident BLDL table feature is standard. The resident BLDL table is either pageable or fixed as indicated during system generation or initialization. The IBM-supplied BLDL table list in SYS1.PARMLIB is used unless one or two alternate user-defined lists are specified during system initialization via the BLDL or BLDLF parameter. The BLDL or BLDLF parameter can be specified via the automated system initialization facility or by the operator. The resident BLDL table feature can also be canceled for an IPL.

  The resident access methods, resident reentrant modules from SYS1.LINKLIB and SYS1.SVCLIB, and resident type 3 and 4 SVC routines options can be included during system generation. If the resident access methods or resident reentrant modules options are not specified during system generation, the RAM and RAMF parameters can still be specified during system initialization to indicate the access method and reentrant modules that are to be made resident, since the resident access methods and reentrant modules options are assumed whether or not they are specified during system generation.

  A standard list of pageable modules for each of these three resident module options is IBM-supplied. User-specified modules can be added to the three IBM-supplied lists of pageable modules during system generation (via the VIRTUAL parameter of the LINKLIB and SVCLIB macros). In addition, alternate lists can be defined by the user and added to SYS1.PARMLIB. Note that when VSAM is included in a generated system, approximately 382,800 bytes of VSAM modules are automatically made resident.

  Inclusion of these resident module options also provides the capability of making access methods, reentrant routines, and type 3 and 4 SVC routines resident in the resident supervisor routines area and fixed instead of pageable. For each option, some modules can be made fixed while others are pageable, all modules can be made pageable, or all modules can be made fixed. The modules that are to be placed in the IBM-supplied standard lists for fixed resident modules can be user-specified during system generation (via the RESIDNT parameter of the LINKLIB and SVCLIB macros). The three IBM-supplied SYS1.PARMLIB members for fixed lists are null.

  The IBM-supplied standard pageable list and fixed list are used for each of these three resident module options unless alternate user-defined pageable and fixed lists are specified or the options are canceled during system initialization via the RAM, RAMF, RSVC, and RSVCF parameters. These parameters can be specified via the automated system initialization facility or by the operator. Even when the resident access methods and reentrant routines option is canceled during system initialization, certain access method modules are made resident and pageable.

  If ERPs are made resident, they are always fixed. Resident ERPs cannot be made pageable. The IBM-supplied standard list, which does not contain any entries, is used unless an alternate user-defined list is specified or the option is canceled during system initialization using the RERP parameter. This parameter can be specified via the automated system initialization facility or by the operator. Any ERP routines that are specified in the RESIDNT and VIRTUAL parameters of the SVCLIB macro are placed in the IBM-supplied standard list during system generation.

The entries in the IBM-supplied standard lists for resident modules differ in VS1 and MFT. In VS1, approximately 141K bytes of virtual storage in the resident supervisor area are required to contain the BLDL entries and modules indicated in the standard lists (approximately 2048 bytes for the BLDL table, 129,768 bytes for type 3 and 4 SVC routines, and 18,856 bytes for QSAM, BSAM, and JAM access methods and certain dump formatting routines). Additional modules are made resident if TCAM or VTAM is included in the generated VS1 system. It may not be possible to use the IBM-supplied lists in a system that supports only 1024K bytes of virtual storage.

- The job entry subsystem area (pseudo partition), which contains JES routines and buffers. The JES area is a minimum of 156K and is pageable.
- The remote terminal access method (RTAM) area (pseudo partition), if RES is used. The RTAM area is a minimum of 58K in size. RTAM is the teleprocessing access method used by RES.
- The JES buffer pool area, which is used by the reader and writer tasks in the JES pseudo partition. The pageable supervisor area size is rounded to a 64K multiple. Any available virtual storage between the problem program area and the beginning of the VTAM pseudo partition is used for JES buffers.

The problem program area in VS1 virtual storage is divided into contiguous partitions, just as the problem program area is in MFT. A maximum of 52 partitions (P0 to P51) can be defined. A partition is identified either as a problem program partition or a system task partition. The maximum number of problem program partitions that can be defined is 15. The maximum number of system task partitions that can be defined is 52. P0, in the highest addressed portion of the problem program area, has the highest execution (task dispatching) priority and P51 has the lowest.

A partition in the problem program area of virtual storage must be a multiple of 64K in size (the segment size used) and begin on a 64K virtual storage boundary. All partitions must be a minimum of 128K in VS1 systems that include VSAM. In VS1 systems without VSAM, 64K is the minimum partition size. These two minimums assume no user-written routines have been added to the IBM-supplied routines that execute in a partition. Small partitions, as defined in OS MFT (those too small to contain the job scheduler), do not exist in OS/VS1, and job entry subsystem support eliminates the need for reader and writer partitions.

The resident ATTACH feature is standard in VS1 and provides the same multitasking capability for partitions as in MFT. In VS1, however, there is no limit on the total number of subtasks that can be created as there is in MFT (from 196 to 249 depending on the number of partitions defined and system options used in MFT).

Jobs are scheduled to execute in problem program partitions by job priority within job class, just as in MFT. However, a VS1 problem program partition can have up to 15 job classes assigned, instead of the MFT limit of 3. In addition, there are 36 job classes (A-Z, 0-9) for problem program partitions in VS1 instead of the 15 job classes (A-O) supported in MFT. A given job class can be assigned to more than one partition in VS1, as in MFT.

The maximum number of partitions to be supported, their size, their type, etc., are defined at system generation. These parameters can be altered by the operator during and after system initialization as long as the maximum number of partitions or the maximum amount of virtual storage specified at system generation or initialization is not exceeded.

Problem program partitions are assigned a nonzero storage protect key. They are used for the scheduling and execution of pageable job steps and direct system output (DSO) writers, and for the scheduling (but not execution) of nonpageable job steps. A pageable job step can be initiated in a problem program partition as long as the size of the program to be executed is at least 4K less than the partition size. (The 4K is the minimum required for system control blocks and tables.) Use of the scheduler work area facility of JES requires the partition to contain considerably more than 4K of virtual storage for non-problem program use. Paged job steps execute with storage addresses in instructions translated by DAT hardware and storage addresses in channel programs translated by the control program.

When a pageable job step executes in a problem program partition, virtual storage within the partition consists of:

- Fixed protected queue area (PQA) of 2K minimum with a protect key of zero
- Pageable protected queue area of 2K minimum with a protect key of zero
- Problem program area, which is all virtual storage in the partition not allocated to fixed or pageable PQA. This area has a nonzero protect key.

The fixed PQA contains partition-related control information that is not paged for system integrity and reliability reasons. For example, the fixed PQA contains the page tables required for the partition, request blocks (PRBs, SVRBs, IRBs, etc.), and the TCBs for any subtasks that are created during execution of a job step. (TCBs for system tasks and one TCB for each partition defined at system generation are contained in the resident nucleus.)

When a partition is defined, either during system initialization or when the operator enters the DEFINE command, at least one virtual storage page is allocated for fixed PQA from the highest addressed virtual storage in the partition, and a page frame is assigned to it and fixed.

The number of virtual storage pages required for fixed PQA at partition initialization time depends on the size of the virtual partition and whether PCI fetch is included in the control program. The amount of fixed PQA required at partition initialization in bytes is 500, plus 1600 if PCI fetch is present, plus 1 byte for each 1024 bytes of virtual storage in the partition rounded to the next multiple of 2048. (For the RTAM pseudo partition, 2K of fixed PQA is required. For the VTAM pseudo partition, 22K of fixed PQA is required.) If additional fixed PQA space is needed during problem program execution, it is taken, a virtual storage page at a time, from the highest available problem program virtual storage in the partition.

Pageable PQA contains control blocks that can be paged with minimal effect on performance. The amount of virtual storage initially allocated to fixed PQA is affected by the size of the scheduler work area for the partition (if any), the size of the data extent block (DEB) table, whether or not PCI fetch is included in the system, the maximum size of the task I/O table (TIOT), and certain JES data set (SYS1.SYSPOOL) characteristics.

No pageable PQA is allocated when a partition is initialized. When a pageable job step is initiated, one or more 2K pages of pageable PQA are allocated from the highest addressed available virtual storage in the partition. Pageable PQA can be expanded during problem program execution. Pages are allocated from highest available problem program virtual storage in the partition.

At partition initialization time, the problem program area contains all virtual storage in the partition below fixed PQA. When a pageable job step is initiated, virtual storage is allocated to it beginning with the lowest addressed virtual storage in the partition. Virtual storage above this is then available for allocation to the fixed and pageable PQAs and to the problem program as needed. The problem program area within a partition that is below the pageable PQA is organized the same way in VS1 as in MFT, as shown in Figure 90.10.2.

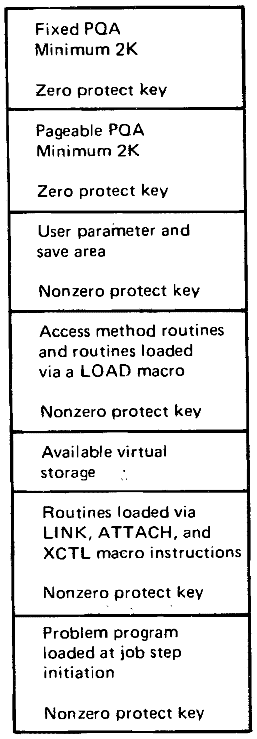

Fixed PQA
Minimum 2K
Zero protect key
Pageable PQA
Minimum 2K
Zero protect key
User parameter and save area
Nonzero protect key
Access method routines and routines loaded via a LOAD macro
Nonzero protect key
Available virtual storage
Routines loaded via LINK, ATTACH, and XCTL macro instructions
Nonzero protect key
Problem program loaded at job step initiation
Nonzero protect key

Figure 90.10.2. Problem program partition organization for a pageable job step in OS/VS1

When a nonpageable job step is initiated, fixed PQA for the job step is allocated in the problem program partition that is used to schedule the job step, as for a pageable job step. However, pageable PQA is allocated in highest addressed virtual and real storage in the V=R area in which the nonpageable job step executes. Therefore, the amount of storage to request in the REGION parameter for a nonpageable job step should include pageable PQA but not fixed PQA requirements. The amount of nonpaged storage required in addition to the fixed PQA requirement should be determined in the same way as for a pageable job step.

Optionally, fetch protection for problem program partitions, both pageable and nonpageable, and all subpools with a nonzero protect key can be included in a VS1 control program during system generation. Fetch protection is established during system initialization for each defined partition and whenever partition redefinition is performed using the DEFINE command. When present in a VS1 control program, fetch protection can be canceled during system initialization.

Fetch protection support causes the fetch protect bit to be turned on in all areas within a problem program partition that have a nonzero storage protect key. Areas within a problem program partition that have a zero protect key, such as fixed and pageable PQA, are not fetch-protected. This approach prevents each problem program partition from accessing and storing data in nonzero key areas of every other problem program partition. A fetch or storage protection interruption occurs if any such attempt is made. Key zero areas in a problem program partition can be accessed by any routine but can be modified only by routines with a zero protect key.

Fetch protection is not provided for the resident control program and fixed SQA in the nonpageable area, the pageable supervisor area (except for the VTAM pseudo partition, which is fetch-protected in its non-PQA areas), or system task partitions.

Problem program partition organization in VS1 offers integrity advantages over MFT partition organization in that all control blocks contained within a VS1 virtual partition are protected from modification by the problem program and fetch protection for nonzero key areas is available.

System task partitions are designated with the identification STP instead of with job classes. They have the same organization as problem program partitions and operate with the CPU in problem program state; however, system task partitions have protect key zero assigned and are not fetch-protected. Problem programs cannot be executed in a system task partition. The Generalized Trace Facility, the MOUNT command, START RDR/WTR commands, STOP RDR/WTR commands, and conversational remote job entry readers are system tasks that are authorized to operate in a system task partition. The missing interruption checker routine and SYS1.SYSJOBQE extension program can also be executed in a system task partition.

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

Unlike main storage areas in MFT, certain virtual storage areas in VS1 need not be contiguous with each other. There can be undefined virtual storage between P0, which must be a multiple of 64K and located on a 64K boundary, and the pageable supervisor area, which is allocated from the top of virtual storage down and need not be a multiple of 64K in size. There cannot be any undefined virtual storage between virtual partitions but there can be between the nonpageable area (V=R line, which is on a 2K boundary) and the pageable area (lowest priority pageable partition starting address), which is on a 64K boundary.

REAL STORAGE ORGANIZATION

Real storage is also divided into a nonpageable and a pageable area, as shown in Figure 90.10.3. The nonpageable area in lowest addressed real storage is allocated to the nonpageable area of virtual storage on a V=R basis. It contains the fixed resident control program (nucleus and RMS routines) and the fixed SQA defined at system generation or system initialization. With a few exceptions, resident control program routines operate with translation mode specified even though they are not paged. This approach is taken because the resident control program accesses virtual storage addresses at various times during its execution and address errors would occur at these times if translation was not operative.

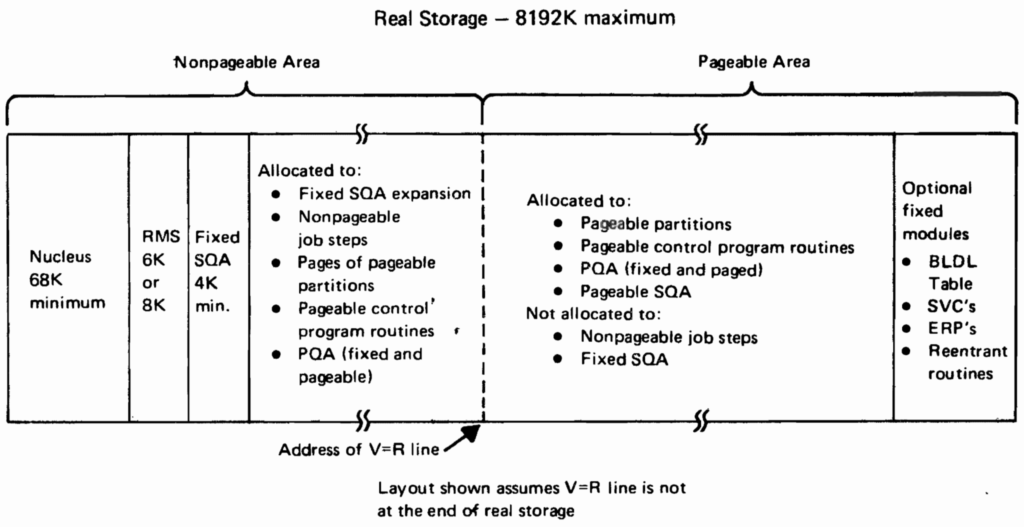


Figure 90.10.3. Real storage organization in OS/VS1

Page frames in the nonpageable area above the resident control program up to the V=R line are allocated to both pageable and nonpageable virtual storage pages. Page frames are allocated to fixed SQA and nonpageable job steps only from available real storage below the address of the V=R line. However, available real storage below the V=R line can also be allocated to pageable supervisor routines, pageable problem programs, pageable SQA, and fixed and pageable PQA, when necessary.

Page frames above the V=R line are allocated to pageable supervisor routines, pageable problem programs, fixed and pageable PQA, and pageable SQA, but cannot be allocated to fixed SQA or nonpageable problem programs. That portion of the pageable supervisor area (resident access methods, BLDL table, etc.) that is made fixed instead of pageable is loaded into highest addressed available real storage during system initialization and marked long-term fixed. (Real storage allocation is discussed in more detail in Section 90:35.)

EXTERNAL PAGE STORAGE ORGANIZATION

External page storage is used to contain the contents of the pageable virtual storage area, which consists of all virtual storage between the V=R line and the end of defined virtual storage. Only fixed PQA pages and any fixed control program routines in the pageable supervisor area, such as SVCs, access methods, etc., are not written in external page storage. The direct access devices supported as paging devices are the

2314/2319, 3330-series (all models), 3340/3344 (all models), 3350 (in native and compatibility modes), and 2305 Model 2.

The direct access storage allocated as external page storage is called the page file (SYS1.PAGE data sets). The page file configuration is defined during system generation. The device types specified during system generation are placed in a list in the NIP routine and the space allocation data is placed in a member of SYS1.PARMLIB called IEASYS00. The page file configuration that was specified at system generation can be modified by updating the IEASYS00 member or by using the PAGE system parameter, which can be specified via the automated system initialization facility or entered by the operator during system initialization. If the PAGE parameter is issued, it overrides (but does not change) the specifications in the IEASYS00 member for the IPL.

The page file configuration specified during system generation can be modified by the addition of volumes, deletion of volumes, and alteration of the space allocation for a given SYS1.PAGE data set. If a paging volume is added, its type must be the same as one of the direct access device types that were defined for paging at system generation. The PAGE parameter can also be used to cause page file volumes to be formatted during system initialization.

If a specified page data set volume is found not to be mounted during system initialization, it is bypassed and a message to the operator is issued that indicates the volume was not mounted. The operator can mount the volume if it is needed and then must enter the PAGE parameter to add the volume to the page file.

The page file can consist of up to eight SYS1.PAGE data sets, each of which must be a single extent only and totally contained on one direct access device. Any mixture of the direct access device types that are supported for paging can be used. While direct access devices that contain a page data set need not be dedicated to paging, this approach is recommended for performance reasons. Page file disk volumes must be permanently resident. The IOS priority queuing option need not be specified for direct access devices that contain page data sets, since paging I/O requests are queued on a priority basis regardless of the queuing option specified for the device for which they are queued.

A track in a VS1 page data set contains a number of 2K record areas called slots. Slots are written as unblocked records without a key. Regardless of the direct access device type used, page data set tracks are formatted with a dummy record written after each 2K slot. The dummy records are added to increase paging performance. They allow time for electronic head switching during accessing of multiple pages contained within the same cylinder, using a command-chained channel program. The track overflow feature is not used because, for a 2K record size, it yields no significant space benefit on the supported devices. The number of 2K slots available per device type is shown in Table 90.10.1.

The page file must be large enough to contain a number of 2K slots equal to or greater than the number of 2K virtual storage pages that exist between the V=R line and the end of virtual storage. External page storage is statically mapped on a one-to-one basis with virtual storage above the V=R line. That is, the contents of any given virtual storage page are always placed in the same slot and the first virtual storage page is associated with the first slot in external page storage, etc. (See Figure 90.10.4.)

Table 90.10.1. Number of 2K slots per paging device type

Device Type	Slots per Track	Slots per Cylinder	Slots per Pack
2314/2319	3	60	11,940*
3330-series Models 1 and 2 and 3350 in 3330 Model 1 compatibility mode	5	95	38,285*
3330-series Model 11 and 3350 in 3330 Model 11 compatibility mode	5	95	76,570*
3340 Model 35	3	36	12,528*
3340 Models 70 and 70F and 3344	3	36	25,056*
3350 (native mode)	7	210	116,550
2305 Model 2	6	--	4608

* Maximum number of 2K slots required is 8192 (for 16 million bytes of virtual storage) less the number of page frames in real storage below the address of the V=R line.

Virtual storage is mapped to the set of paging volumes that are defined and mounted during system initialization beginning with the highest addressed virtual storage. The page data sets defined in IEASYS00 are allocated first in the sequence in which they are listed (that is, as specified during system generation). Second, the devices specified via the automated system initialization facility (NIPxxxxx member) are allocated in the order in which they are listed in the member. Last, the devices specified by the operator during system initialization via the PAGE parameter are allocated in the order in which they are entered.

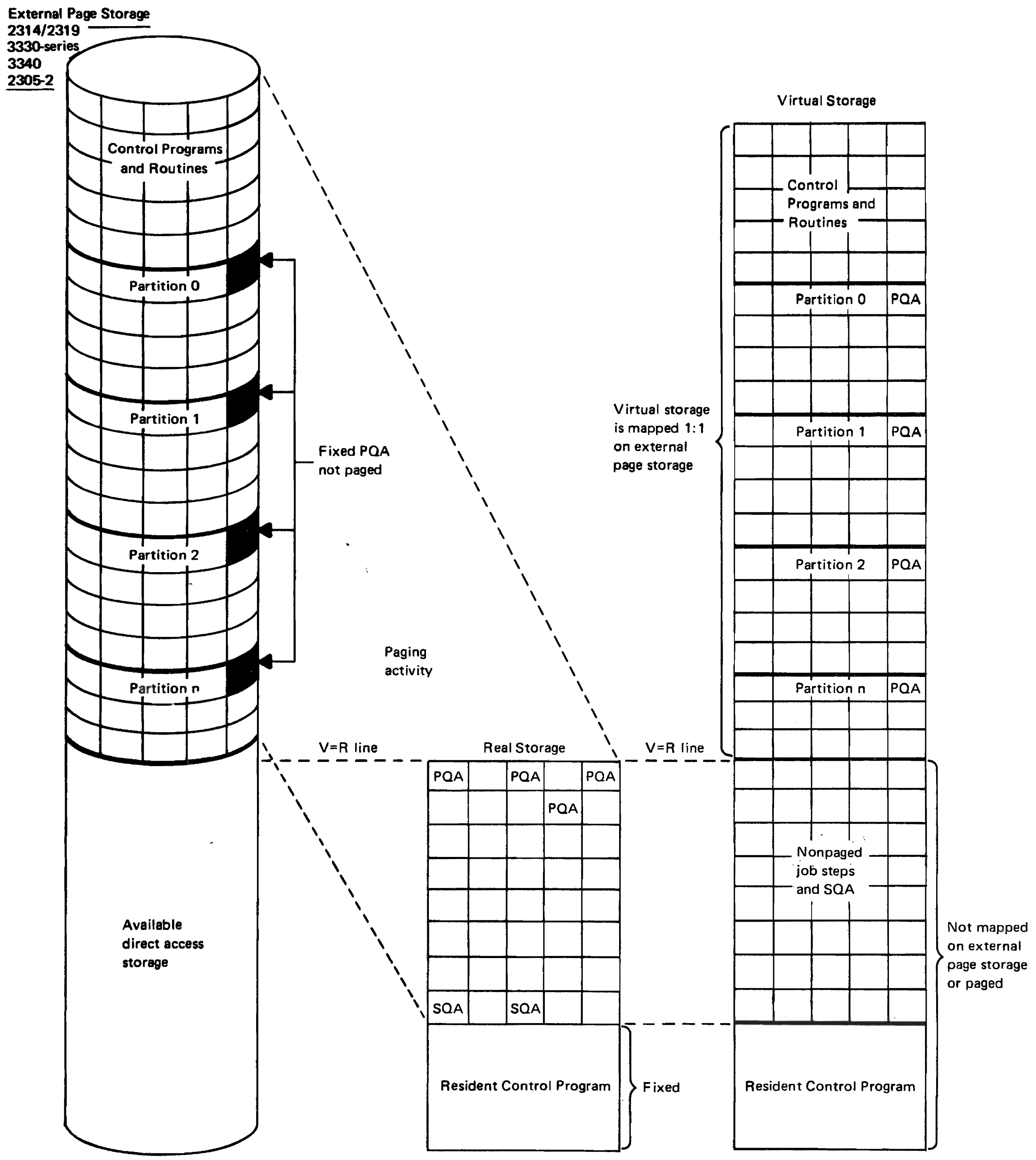


Figure 90.10.4. External page storage, real storage, and virtual storage relationship in OS/VS1

SYSTEM INITIALIZATION

At the completion of the VS1 system initialization procedure, EC and translation modes are operative. During system initialization, virtual, real, and external page storage are initialized as follows.

Virtual Storage

During system initialization, a virtual storage of the size specified at system generation is initialized, unless the size is smaller than the minimum virtual storage size required or the operator overrides the system generation specification. The operator can increase or decrease the amount specified at system generation but cannot specify less than 1024K or the size of real storage plus 512K, whichever is greater.

The initialization of virtual storage consists of building the control blocks and tables required to define the various areas of virtual storage that are shown in Figure 90.10.1. Virtual storage is mapped according to system generation parameters and any additional definitions specified in SYS1.PARMLIB or overrides indicated by the operator. Control program modules that are to be made resident in the pageable supervisor area of virtual storage and paged are allocated virtual storage and fetched from load module libraries. Resident modules that are to be fixed are placed in the high end of real storage and marked long-term fixed.

Resident modules that are to be paged are paged out to external page storage as a result of normal paging activity, if necessary. There is no routine in VS1 that forces the page out of pageable load modules that are fetched during system initialization or thereafter. If real storage becomes full during the loading of modules, pages are written out as per the page replacement algorithm.

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

Real Storage

The operator can set the end-of-real-storage address to a value lower than that of the actual end of real storage during IPL (see _OS/VS1 IPL and NIP Logic_, SY24-5160). The real storage size specified must be a multiple of 64K. Only real storage up to the limit specified will be zeroed during IPL and used by the VS1 operating system for the duration of this IPL.

At the completion of system initialization, real storage contains:

- The resident control program (in the nonpageable area in lowest addressed real storage) and any resident supervisor routines that are fixed (in the high end of real storage). The standard nucleus (member IEANUC01) or any one of eight alternate nucleus members can be selected during IPL.
- 4K or more of fixed SQA (adjacent to the resident control program)
- At least one page frame of fixed PQA for each partition defined and initialized
- A minimum of three fixed pages for JES

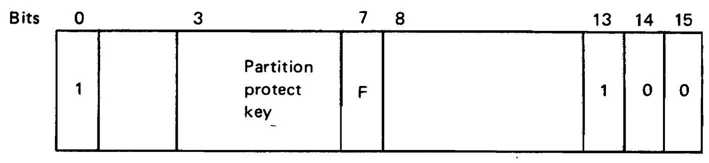
The minimum VS1 control program (one partition, one JES reader, one JES writer, and no fixed supervisor routines) requires 86K of fixed real storage (68K for the nucleus, 6K for RMS, 4K of SQA, 2K of fixed PQA, and 6K of fixed real storage for JES).

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

Control blocks and tables are initialized to reflect the size and organization of the virtual storage defined, as well as the real storage present and allocated. The operator is notified during system initialization if VS1 is being loaded into a system with more than 4096K of real storage and real storage above 4096K is not used.

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

A page table entry for a problem program partition is initialized as shown in Figure 90.10.5. Bit 0 is on to indicate that bits 3 to 6 contain the nonzero protect key of the partition. If fetch protection is included in the system, bit 7 (fetch protect bit) is on. The invalid bit is on and the user bit is off. When off, the user bit indicates that a page-in is not required after a page frame has been allocated to the virtual storage page. The user bit is turned on the first time real storage is allocated to this virtual storage page after job step initialization. At job step termination, the user bit is turned off.



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

Figure 90.10.5. Page table entry contents for an initialized, inactive problem program partition

The first time any problem program virtual storage page is referenced, a page frame is allocated without a page-in and cleared to zeros (for security protection). The protect key value in bits 3 to 7 of the page table entry is inserted in the protect key of the page frame allocated. The invalid bit is turned off and the user bit is turned on in the associated page table entry. The high-order bits of the address of the allocated page frame are placed in bits 0 to 12 of the page table entry. The page table entry for a virtual storage page with a page frame allocated is shown in Figure 90.10.6.

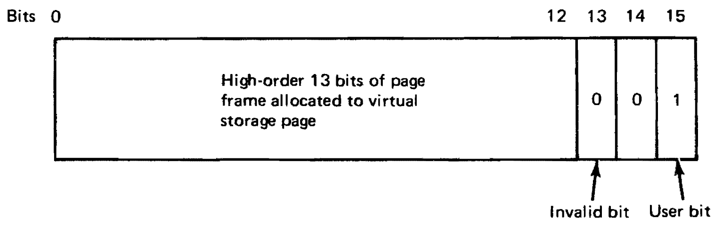

Bits 0
12 13 14 15
High-order 13 bits of page frame allocated to virtual storage page
0
0
1
Invalid bit
User bit

Figure 90.10.6. Page table entry contents for a problem program partition virtual storage page with a page frame allocated

External Page Storage

The location, in terms of unit address(es) or volume serial number(s), and the size of the page data sets in the page file that were specified at system generation are used unless overridden during system initialization. The operator is notified if the page file space allocated is not large enough to contain the virtual storage space above the current V=R line. The first time a given volume is used for a page data set, space is allocated and slots are formatted. Thereafter, the page data set can be used without reformatting as long as the same or a lesser amount of space is allocated. Formatting can be requested by the operator.

At the completion of system initialization, external page storage may contain some of the pageable control program load modules that are resident in virtual storage (if any were paged out during system initialization). The volumes containing external page storage and the paging devices on which these volumes reside cannot be changed without a re-IPL.

Missing Interruption Checker Routine

The missing interruption checker routine, not provided in MFT, is a standard feature of VS1. If this routine is to be used, the operator must initiate its execution in a partition via a START command. This routine is authorized to execute in a system task partition. The missing interruption checker routine checks for missing channel-end and I/O device-end interruptions during system operation.

This routine is designed to lessen the impact on system operation of missing I/O interruptions that are the result of a hardware malfunction, operational error, or environmental circumstance. 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 missing interruption checker is CPU model- and channel-independent. It 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 by alteration of the missing interruption checker module, which is contained in SYS1.LINKLIB.

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 cancellation of the affected job.

The operator can terminate operation of the missing interruption checker routine at any time during system operation by entering a STOP command.

Automatic Device Status Initialization

The smart NIP routine that provides automatic device status initialization can be included in a VS1 system during system generation, as in MFT. In VS1, however, the operator can include the smart NIP routine during system initialization by specifying the DEVSTAT system parameter. If the smart NIP routine is not requested, the availability of the direct access devices specified during system generation is tested. If the smart NIP routine is requested, all the devices of the type(s) specified in the DEVSTAT parameter or at system generation are tested for availability.

In VS1, NIP notifies the operator when an unexpected device status that is unacceptable is received several times for a device during the testing of device addresses that were specified during system generation. This facility is designed to notify the operator of a condition of which he would otherwise be unaware. It is included in VS1 whether or not smart NIP is present.

Automated System Initialization

During the initialization of a VS1 control program, many more parameters can be specified than during the initialization of an MFT control program. This capability is provided because VS1 supports many new features and is designed to offer the advantage of allowing an installation to change most parameters that were specified at system generation without the necessity of a regeneration.

The amount of operator action required to initialize a VS1 control program can be minimized by use of the standard automated system initialization (ASI) feature, which is not supported in MFT. ASI provides an automated way for the operator to indicate the parameters to be used for an IPL that allows system generation specifications to be changed without the necessity of entering a large amount of data via the operating system console device.

ASI enables an installation to establish several different sets of parameters that override various system generation specifications when necessary and permits the operator to select a set of parameters during system initialization with a minimum of keyboarding. In addition, the time required to initialize a VS1 system is significantly reduced when the parameters specified at system generation and via the ASI process are not be be changed.

When the ASI facility is used, the NIP and master scheduler initialization (MSI) routines establish parameters by processing a set of up to ten different types of SYS1.PARMLIB parameter specification members that contain the parameters to be used for this IPL. Each type of parameter specification member contains a particular group of parameters or commands: system parameters, JES reconfiguration parameters, DEFINE command parameters, SET parameters, permanently resident volume list parameters, automatic commands, SMF parameters, RTAM parameters, or mass storage volume control parameters for the 3850 Mass Storage System.

A default parameter specification member for five of the nine member types is IBM-supplied. One or more user-defined parameter specification members can be defined for each of the ten types of parameter groups and placed in SYS1.PARMLIB using the IEBUPDTE utility.

A parameter specification member contains 80-byte records, each of which specifies a parameter in the same format as that used when the parameter is entered manually during system initialization. The four-to-eight-character names assigned to user-defined parameter specification members must adhere to the following conventions: the first three characters must be the IBM-defined characters that identify the particular parameter-type group (NIP, JES, etc.) and the last five characters can be user-selected to identify different members for the same parameter-type group (NIP01, NIP02, for example).

In addition to parameter specification members, SYS1.PARMLIB must contain user-defined members whose contents are a list of the names of parameter specification members. One default member name list is IBM-supplied in a table in the master scheduler initialization routine. This table contains the names of the six default parameter specification members and a null entry for each of the other four types of parameter specification member types, as shown in Figure 90.10.7. The names assigned to the user-defined member name list members need not follow a particular naming convention. A user-defined member name list member can contain the names of from one to nine user-defined parameter specification members that are listed in any sequence.

When ASI is to be utilized, general operation of the system initialization procedure is as follows. In response to the NIP message SPECIFY SYSTEM AND/OR SET PARAMETERS (IEA101A), the operator indicates via the operating system console the user-defined member name list member that is to be used for this IPL or supplies the names of from one to ten parameter specification members in cards (one member name per card).

Alternatively, the member name list member that is to be used when an automated system initialization is performed can be specified at system generation via the AUTO parameter so that it need not be entered by the operator during system initialization in response to the IEA101A message. The operator can also enter parameters via the console that are to override corresponding parameters in the user-defined parameter specification members that are used for this IPL.

NIP reads the member specified or the cards supplied and any additional parameters the operator supplies. NIP then overrides corresponding entries in the default member name list with the member names read. If the user-supplied member name list does not contain a member name for a particular parameter-type group, the member name or null entry in the default member name list for that group is used. Thus, the operator can override some or all of the default parameter specification members for an IPL.

Once modification of the default member name list is completed, NIP processes the contents of the system parameters member if one was specified and operator-supplied system parameters, if any. The MSI routine then processes the parameters specified in the other parameter specification members that are named in the modified default member name list. The parameter specification member names contained in the modified default list that is used for the IPL are listed on the operator's console.

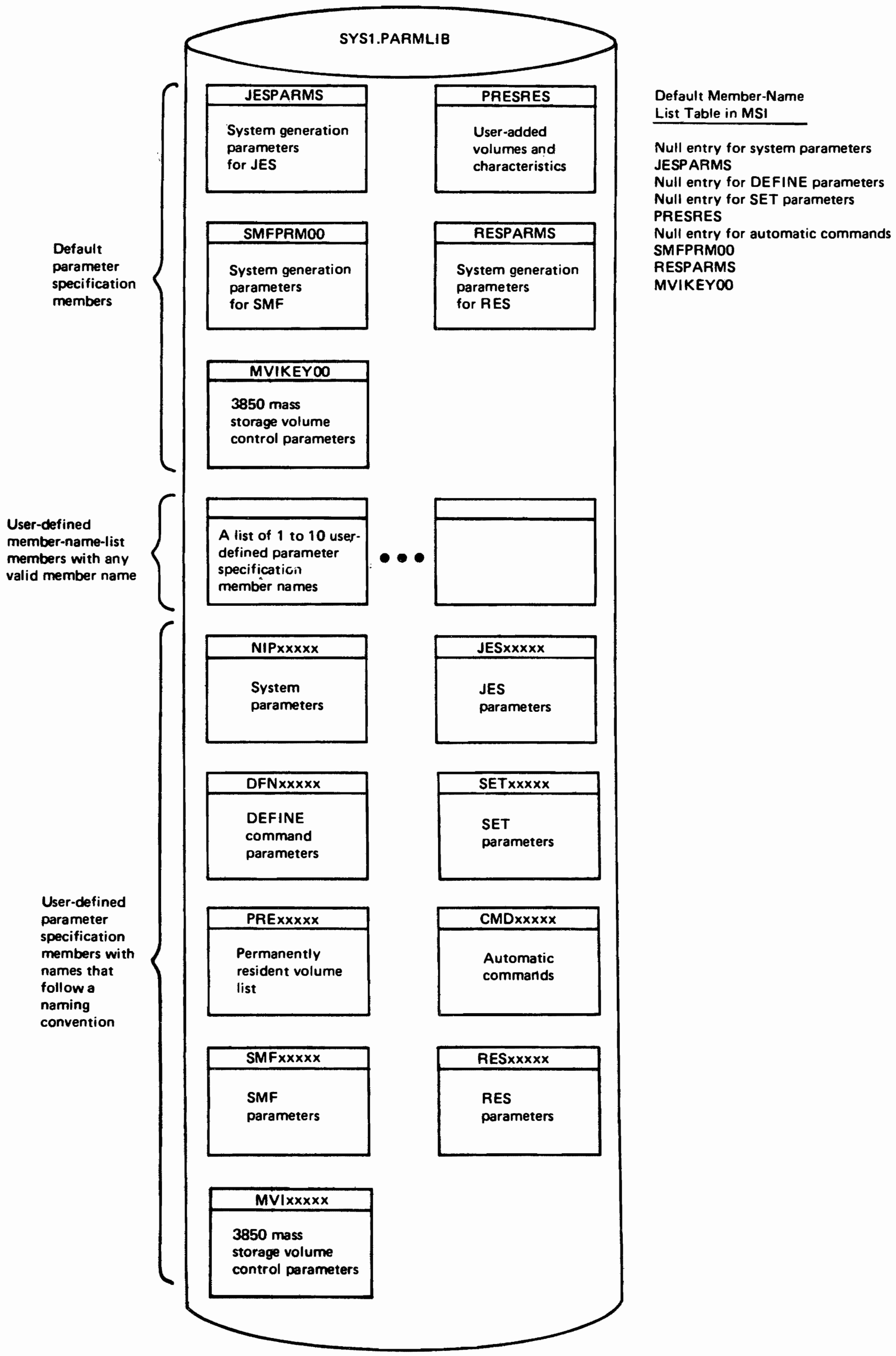

SYS1.PARMLIB
Default parameter specification members
JESPARMS
System generation parameters for JES
PRESRES
User-added volumes and characteristics
SMFPRM00
System generation parameters for SMF
RESPARMS
System generation parameters for RES
MVIKEY00
3850 mass storage volume control parameters
User-defined member-name-list members with any valid member name
A list of 1 to 10 user-defined parameter specification member names
User-defined parameter specification members with names that follow a naming convention
NIPxxxxx
System parameters
JESxxxxx
JES parameters
DFNxxxxx
DEFINE command parameters
SETxxxxx
SET parameters
PRExxxxx
Permanently resident volume list
CMDxxxxx
Automatic commands
SMFxxxxx
SMF parameters
RESxxxxx
RES parameters
MVIxxxxx
3850 mass storage volume control parameters
Default Member-Name List Table in MSI
Null entry for system parameters
JESPARMS
Null entry for DEFINE parameters
Null entry for SET parameters
PRESRES
Null entry for automatic commands
SMFPRM00
RESPARMS
MVIKEY00

Figure 90.10.7. IBM-supplied default parameter specification members and member name list for automated system initialization

Listed below are the types of parameter specification members. For each type, the member name that must be assigned to a user-defined parameter specification member is given, the name of the default member in the IBM-supplied member name list is given, the contents of the default and user-defined members are described, and the way in which the parameters specified in the member can be manually overridden by the operator, if any, is indicated. The action taken if no parameter specification member is specified for a given parameter-type group and when a specification error is encountered during the processing of a parameter specification member are also described.

- System parameters (user-defined member name is NIPxxxxx, default member name is all blanks to indicate there is no default member). A user-defined system parameters member can contain any of the system parameters that the operator can enter in response to the IEA101A message except the parameters D=member name, AUTO, and RDR.

  Hence, the following system parameters are valid (an asterisk identifies parameters that are not also MFT parameters): ALTSYS, BLDL, BLDLF*, dynamic dispatching parameters (DDDEL, DDG, DDMIN, DDRATIO, DDSTAT)*, DEVSTAT (smart-NIP option parameter)*, HARDCPY, JES job list parameters (JOBQINT, JOBQEXT, JOBQNXT)*, PAGE*, PSQA*, PURGE and NOPURGE (demounting options for the 3850 Mass Storage System)*, RAM, RAMF*, REACT (task reactivation parameter)*, RERP, RSVC, RSVCF*, SECURITY (fetch protection parameter)*, SQS, TRACE*, and VR (virtual=real area specification)*.

  The system parameters specified in the NIPxxxxx member override for this system initialization the corresponding parameter that was specified at system generation. The operator can override one or more of the system parameters in the specified NIPxxxxx member by including system parameters in the response to the IEA101A message.

  If a NIPxxxxx member is not specified for an IPL, the system parameters specified at a system generation are used unless the operator overrides them by supplying system parameters in his response to the IEA101A message.

  If NIP encounters a specification error during the processing of a NIPxxxxx member, the operator is notified and further processing of the NIPxxxxx member is terminated. The operator must reenter the erroneous system parameter, all unprocessed system parameters in the NIPxxxxx member (those following the incorrect system parameter), and any system parameters specified in response to the IEA101A message.

- JES reconfiguration parameters (user-defined member name is JESxxxxx, default member name is JESPARMS). A JESxxxxx member can contain any of the JES parameters that are contained in the JESPARMS member (ALCUNIT, BUFSIZE, JOBLOG, JOBQVOL, JOUTLIM, LPRPARM, LPUPARM, LRDPARM, NUMBUFS, PRLRECL, RDR, RPRPARM, RPUPARM, RRDPARM, SPOLCAP, SPOLVOL, STEPWTP, SWDSLMT, WTLRCDS, and WTR). JESPARMS contains all the JES parameters that are specified during system generation except the three resident job list parameters (JOBQINIT, JOBQEXT, and JOBQNXT). If a JESxxxxx member is not named in the user-specified member name list, the JESPARMS member is used.

  The MSI routine compares the JES parameters specified in JESPARMS with those specified at system generation. When a difference is encountered, the JES parameter in the JESPARMS member overrides the system generation specification. JES parameters in JESPARMS can be modified using the IEBUPDTE utility.

  If a JESxxxxx member is specified, the JESPARMS member is not used. Any parameter that is specified in the JESxxxxx member overrides the corresponding JES parameter that was specified at system generation. The name JESNULL can be specified in the supplied member name list when the JES parameters that were specified at system generation are not to be modified for an IPL. This causes all processing of the JESPARMS member to be bypassed.

If a specification error is encountered during the processing of a JESxxxxx or the JESPARMS member, the operator is notified. The specified JES member is ignored and all the system generation JES parameter specifications are used for the IPL.

- DEFINE parameters (user-defined member name is DFNxxxxx, default member name is all blanks to indicate there is no default member). A DFNxxxxx member can contain any of the parameters that can be specified in a DEFINE command except the PARM=membername parameter. The parameters in this member override for this system initialization the partition definitions that were specified at system generation.

  The partition definitions made effective by this member can be changed after system initialization via a DEFINE command that specifies new definitions or the PARM parameter, which specifies the name of a member in SYS1.PARMLIB that contains the new definitions. If a DFNxxxxx member is not specified in the member name list for a given system initialization, the master scheduler issues the CHANGE PARTITIONS? message (IEE8021D) and the operator can enter any required changes via the operating system console as usual.

  If a specification error is encountered during processing of the parameters in a DFNxxxxx member, the same error messages are issued as when the operator enters the definitions via the operating system console and the operator can take the same recovery actions. The DFNxxxxx member is not modified by any parameter specification corrections made by the operator. Any incorrect specifications in a DFNxxxxx member must be corrected using the IEBUPDTE utility.

- SET parameters (user-defined member name is SETxxxxx, default member name is all blanks to indicate there is no default member). A SETxxxxx member can contain the following SET parameters: PROC, Q, SPOOL, and SYSW. The parameters specified in this member override for this system initialization the corresponding SET parameters that were specified at system generation.

  The operator can override any parameters in a SETxxxxx member by including the SET parameters in his response to the IEA101A message. The SET parameters DATE, CLOCK, and GMT cannot be specified in a SETxxxxx member, but they can be entered by the operator in response to the IEA101A message. DATE and CLOCK can also be specified in a SET command that is issued after system initialization is completed.

  If a SETxxxxx member name is not specified in the member name list for a given system initialization, the system generation SET parameter values are used unless the operator overrides them by entering new specifications via the operating system console in response to the IEA101A message.

  If specification errors are encountered during processing of the parameters in the SETxxxxx member, the same error messages are issued as when the operator enters SET parameters via the operating system console and the same operator responses can be entered. Parameter corrections made by the operator do not cause the SETxxxxx member to be updated.

- Permanently resident volume list parameters (user-defined member name is PRExxxxx, default member name is PRESRES). The PRExxxxx member can contain a list of direct access volumes with mount and allocation characteristics specified in the same format as that used in the PRESRES member. When a PRExxxxx member is specified in the user-supplied member name list, it is used instead of PRESRES for this IPL. The name PRESNULL can be specified in the user-supplied member name list to cause processing of PRESRES to be bypassed.

- Automatic commands (user-defined member name is CMDxxxxx, default member name is all blanks to indicate there is no default member). The CMDxxxxx member can contain any commands that are to be executed automatically during system initialization. START, STARTF, LOGON, VARY, MONITOR, MODE, and DISPLAY commands are examples of commands that could be placed in a CMDxxxxx member. If a CMDxxxxx member is not specified in a member name list for a given system initialization, no automatic commands are issued, since automatic commands cannot be specified at system generation.

  If a specification error is encountered during the processing of a command in the CMDxxxxx member, the operator is notified and can correct the error by reentering the command via the console. The CMDxxxxx member is not updated.

- SMF parameters (user-defined member name is SMFxxxxx, default member name is SMFPRM00). The SMFxxxxx member can contain any of the SMF parameters that are contained in the default member SMFPRM00, which is IBM-supplied. If an SMFxxxxx member is specified, it is used instead of SMFPRM00. The SMFPRM00 member is used when SMFxxxxx is not specified in the user-supplied member name list. The operator can override the SMF parameters in the member used (either SMFPRM00 or SMFxxxxx) during SMF initialization only if the OPI=YES parameter was specified in the member used.

  If an SMFxxxxx member is not named in the user-supplied member name list and an SMFPRM00 member is not present in SYS1.PARMLIB, the operator is asked to supply the required SMF parameters via the operating system console during SMF initialization. If specification errors occur during processing of the parameters in the SMF parameters member, the operator is notified and can correct the error via the console whether or not OPI=YES was specified. The name SMFNULL can be placed in a member name list to specify the use of a null member instead of SMFPRM00 for an IPL.

- RTAM parameters (user-defined member name is RESxxxxx, default member name is RESPARMS). An RESxxxxx member and the RESPARMS member can contain one or more of the following RTAM parameters: APPLID, CPACT, CPACTDF, CTABLE, EXTRA, MSGFCTR, OMIT, PASSWD, PORTS, TPBUF, TPREAD, TPPRINT, TPPUNCH, and STBUFNO. The RTAM parameters in the specified RESxxxxx member or the RESPARMS member override for an IPL the RTAM values that were specified during RTAM generation.

  The RESPARMS member is used unless an RESxxxxx member is specified. The user is responsible for creating the RESPARMS member as well as any RESxxxxx members that are to be used. If an RESxxxxx member is not specified and an RESPARMS member does not exist, the RTAM values specified at RTAM generation are used for the IPL. If a specification error is encountered during processing of the specified RES parameters member, all values in the member are ignored and the RTAM values that were specified at RTAM generation are used.

- Mass storage volume control parameters (user-defined member name is MVIxxxxx, default member name is MVIKEY00). The MVIKEY00 or an MVIxxxxx member supplies the following three parameters to the Mass Storage System Communicator: names of the Mass Storage Volume Inventory and Mass Storage Volume Control Journal data sets, location of messages for the space manager, and time interval for checking staging drive groups.

Regardless of how the member name list is supplied, the operator can specify the NOLIST keyword in response to message IEA101A to prevent noncritical informational and nonerror messages from NIP and the MSI routine from being written to the operating system console during system initialization. For example, eliminated are printing of the ready message, any automatic commands, a list of the parameter specification members identified in the AUTO or RDR parameter for this IPL (IEA764I), and a list of the system parameters contained in the NIPxxxxx member used for this IPL (IEA765I).

When the AUTO parameter was not specified during system generation, the operator can perform one of the following in response to the message IEA101A during system initialization:

- Enter the AUTO parameter with or without NOLIST to specify the name of the member name list member that is to be used for this automated system initialization. In addition to AUTO, the operator can enter any of the system parameters and SET parameters that are valid for a response to the IEA101A message. These parameters override corresponding parameters in the system parameters and SET parameters members or the system generation specifications, as previously described.

- Enter the RDR parameter with or without NOLIST to indicate that an automated system initialization is to be performed and that the member name list is to be read from a card reader. As when the AUTO parameter is specified, the operator can also enter system parameters and SET parameters that are valid for the IEA101A message.

- Omit the AUTO and RDR parameters from the response. This causes the default member name list member to be used, and results in a manual rather than an automated system initialization. In this case, the operator must enter any system and SET parameters that are to override the system generation specifications in response to the IEA101A message.

If the AUTO parameter was specified at system generation, the operator can perform one of the following in response to the IEA101A message, which is issued after, instead of before, NIP has processed the system parameters in the NIPxxxxx member named in the member name list member specified during system generation, if any:

- Enter an EOB without entering any data to indicate that an automated system initialization is to be performed using the member specified in the AUTO parameter at system generation and that no parameters are to be changed by the operator for this IPL.

- Enter, with or without NOLIST, the AUTO parameter with a member name or the RDR parameter to indicate that an automated system initialization procedure is to be performed but that the list indicated in this response is to be used for this system initialization instead of the member specified at system generation. The system and SET parameters that are valid in a response to message IEA101A can be included to override corresponding parameters in the parameter members specified or the system generation specifications.

- Enter AUTO=, to indicate a manual instead of an automated system initialization is to be performed using the default member name list. The system and SET parameters that are valid for message IEA101A can be entered as well to override system generation specifications and those contained in the NIPxxxxx member that is specified in the member name list member identified by the AUTO system generation parameter.

Note that if NOLIST was specified with the AUTO parameter at system generation, the LIST parameter can be included in the response to message IEA101A to override the NOLIST specification.

If the virtual storage size specified at system generation is to be used and no initialization parameters are to be changed for an IPL, the system initialization process can be speeded up considerably by initiating the FASTNIP function. This procedure is accomplished by pressing the request key (enter key on the Model 138/148/158 display console operating in display mode or end key on the 3066 console) immediately after pressing the load button.

When FASTNIP is used, the page file messages (IEA757I, IEA758I, and IEA761I), SPECIFY VIRTUAL STORAGE SIZE message (IEA760A), SPECIFY SYS1.DUMP TAPE UNIT ADDRESS message (IEA135A), USE VS1 ASSIST message (IEA407A), and SPECIFY SYSTEM AND/OR SET PARAMETERS message (IEA101A) are not issued during system initialization. Only informational messages regarding parameters used, readers and writers started, automatic commands issued, etc., that are usually printed during system initialization are issued. FASTNIP can be used whether or not the VS1 system is set up to use the automated system initialization facility.

90:15 MAJOR COMPONENTS

The major control and problem program components of OS/VS1 are shown in Table 90.15.1. Except for the integrated emulator programs and industry subsystems, components identified as SCP are distributed with the VS1 starter system. Integrated emulators, Type I programs, and program products are not distributed as part of VS1 and must be obtained separately. Industry subsystem support for the 3600 Finance Communication System, 3650 Retail Store System, 3660 Supermarket System, and 3790 Communication System is provided in independent releases.

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

The new features of VS1 and the most significant differences between VS1 and MFT components are presented in the discussions that follow. VS1 uses the same system data sets and libraries that are used in MFT as well as the following additional libraries and data sets:

- SYS1.DSSVM - required if DSS is used
- SYS1.PAGE - required for the page file
- SYS1.SYSPOOL - required for JES spool data sets
- SWADS (scheduler work area data set) - one required for each initiator started unless a scheduler work area is assigned instead
- SYS1.SYSWADS - required for certain job scheduling data
- SYS1.ISPMAC - required for the assembly of exit definition macros if the installation-specified selection parameters facility is used
- SYS1.VTAMLIB, SYS1.VTAMLST, and SYS1.VTAMOBJ - required if VTAM is included in the generated system
- SYS1.INDMAC - required if support of an industry subsystem, such as the 3600 Finance Communication System, is included in the generated system
- SYS1.UADS and SYS1.BRODCAST - required if RES is used. The SYS1.RMTMAC and SYS1.RMTOBJ system data sets are required for an RTAM generation.
- SYS1.WARNA and SYS1.WARNB - required if Power Warning feature support for a Model 158 or 168 is included in the system

Note that if the Shared DASD option is included in the generated system, SYS1.PAGE, SYS1.SYSWADS, SWADS, SYS1.SYSPOOL, SYS1.UADS, SYS1.BRODCAST, SYS1.WARNA, and SYS1.WARNB data sets cannot be shared in addition to those that cannot be shared in an MFT environment (which are SYS1.SVCLIB, SYS1.NUCLEUS, SYS1.LOGREC, SYS1.SYSJOBQE, SYSCTLG, SYS1.ACCT, SYS1.MANX, SYS1.MANY, and PASSWORD data sets).

Table 90.15.1. OS/VS1 control and processing program components

CONTROL PROGRAM COMPONENTS (SCP)	
<u>Job Management</u>	<u>Task Management</u>
• Master scheduler and communications task • Job Entry Subsystem (JES) Job entry peripheral services Job entry central services • Job scheduler Initiator Interpreter Allocation Terminator Direct SYSOUT writers • System Management Facilities (SMF) • Remote Entry Services (RES) • Conversational Remote Job Entry (CRJE)	• Interruption supervisor • Task supervisor • Virtual storage supervisor • Contents supervisor • Timer supervisor • Overlay supervisor <u>Page Management</u> • Page exception handler • Page supervisor Real storage management External page storage management
<u>Data Management</u>	<u>Recovery Management</u>
• Input/output supervisor • Access methods QSAM, BSAM, QISAM, BISAM, VSAM, BDAM, BPAM, BTAM, TCAM, GAM, VTAM • Catalog management • Direct Access Device Space Management (DADSM) • OPEN/CLOSE/EOV • 3704/3705 System Support Programs • Industry Subsystems	• Machine Check Handler (MCH) • Channel Check Handler (CCH) • Alternate Path Retry (APR) • Dynamic Device Reconfiguration (DDR) • Online Test Executive Program (OLTEP) • Problem determination facilities Service Aids Dynamic Support System
PROBLEM PROGRAMS (SCP AND PP)	
<u>Language Translators</u>	<u>Service Programs</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 (PP) • ITF BASIC (PP) • System/7 FORTRAN IV System/370 Host Compiler and Library • VS BASIC (PP)	• Linkage Editor (SCP) • Loader (SCP) • Utilities System and data set (SCP) Data set with ASCII (PP) • Basic Unformatted Read System (PP) • OS Sort/Merge 5734-SM1 (PP) • OS/VS Sort/Merge 5740-SM1 (PP) • Subsystem Support Services (SCP) <u>Integrated Emulators</u> • 1401/1440/1460 (SCP) • 1410/7010 (SCP) • 7070/7074 (SCP) • 7080 (SCP) • 709/7090/7094/7094II (SCP) • DOS Emulator (SCP)
<u>General</u> • Application-oriented program products (some operate in paged mode and some in nonpaged mode)	

Table 90.15.1 (continued)

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

VS1 supports all the primary operator console devices required for Models 135 to 168. The DIDOCS option with 3270 support must be included in a VS1 control program to support display mode operations on the display console for Models 138, 148, and 158 or to support 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 and not for output operations. MCS and DIDOCS in VS1 support the same console devices and console functions as in MFT.

VS1 supports only the printer-keyboard and display modes of the display console for Models 138 and 148. The 115/125 Console-Display-Emulation mode of this display console is not supported by VS1. Printer-keyboard support is functionally equivalent to 3215 Console Printer-Keyboard support. Display mode support is equivalent to that supported for the Model 158 display console except that Model 138 and 148 display console users can program twelve function keys that are not available on the Model 158 display console. The 3286 Model 2 Printer is supported for hard-copy output in both printer-keyboard and display modes. This hard-copy support for the Model 138 and 148 display console is optional.

90:20 JOB MANAGEMENT

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

The organization and new features of VS1 job management are designed to provide the following advantages over MFT:

- More efficient handling of peripheral I/O operations (via JES)
- Enhanced support of remote job entry (via RES)
- More system configurability without regeneration
- Enhanced operator command processing
- Additional operator control via the new WRITER and PAGETUNE commands and additions to the support provided by several MFT commands
- Improved job scheduling via elimination of small partition scheduling and significant reductions in contention for the job queue (implementation of the resident job list, SYS1.SYSWADS data set, scheduler work area data sets, and scheduler work areas)

MASTER SCHEDULER AND COMMUNICATIONS TASKS

As in MFT, the master scheduler task and the communications task in VS1 handle initialization functions during system initialization and operator/system communication. Portions of these routines are contained in the fixed nucleus while other portions operate in the pageable supervisor area. The communications task is repackaged in VS1 to minimize the occurrence of page faults and to combine a transient routine that calls another transient routine into one 2K transient routine when possible.

The communications task is also modified in VS1 to dynamically obtain any required WTO/WTOR buffers when the number specified at system generation are all being used, instead of waiting until a buffer becomes available. In MFT, the communication task waits for a buffer unless it is servicing a request from certain high-priority system routines. In addition, when a WTO that specifies the ID of an inactive console is issued, the message is not saved but is routed to the master console (to avoid using up available pageable SQA space).

Most commands are processed in the 2K SVC transient area, which is pageable. Certain of these command processing routines are repackaged in VS1 to execute in 2K multiples (instead of 1K) to increase their performance. Some commands must operate in a partition (as in MFT) and a system task partition can be defined for this purpose, as indicated previously.

The master scheduler processes the following operator commands: CANCEL (a queued job), DEFINE, DISPLAY, DISPLAY ACTIVE, DISPLAY C, DISPLAY K, DISPLAY PFK, DISPLAY U, DUMP, HALT EOD, HOLD, LISTBC, LOGON, MONITOR ACTIVE, RELEASE, RESET, ROUTE, SEND, SETPRT, and SWITCH.

An internal command processing capability, not provided in MFT, is supported in VS1. This facility enables a program that is authorized via APF to use the MGCR macro or an SVC 34 instruction directly to issue a VS1 operator command (those handled by SVC 34) during its execution. The MGCR macro provides linkage to an SVC 34 instruction, which is used to issue an operator command via programming. Messages associated with the command are written to the operator (master) console device.

All MFT operator commands and parameters and their formats are accepted in VS1 except those associated with MFT features that are not supported in VS1. Modifications or extensions to the functions performed by the following commands, which are also supported in MFT, have been made in VS1:

- CANCEL - This command can be issued in VS1 to cancel an abnormal termination dump at any time during construction of the dump data. In MFT, a CANCEL command is rejected if it is received while dump data is being written to the SYSABEND data set. In addition, up to five jobs, five system tasks in device allocation status, or the processing of up to five SYSOUT data sets by writers can be canceled with a single CANCEL command in VS1, instead of only one as in MFT. The USER parameter is added to identify remote (RES) users.

- DEFINE - In VS1, the additional keyword PARM is supported for the DEFINE command. The PARM keyword can be used to specify the name of a member in SYS1.PARMLIB that contains the partition definitions to be made effective as a result of the DEFINE command. The same partition definition data that can be supplied via a DEFINE command can be placed in a SYS1.PARMLIB member. This facility relieves the operator of having to manually key in partition redefinitions in VS1. The DEFINE command cannot be entered via an input stream.

- DISPLAY - The class, priority, queue location, position on the queue, and (for an RES job) user identification for an active job are displayed when a DISPLAY job names command is issued (not given in MFT for active jobs). Up to five job names can be specified in one DISPLAY command in VS1 instead of a maximum of one as in MFT. The P parameter can be specified to cause profiles associated with the installation-specified selection parameters facility to be displayed (see interpreter routine discussion later in this subsection under "Job Scheduler"). The LIST parameter can be specified on a DISPLAY R command to cause the display of up to 72 characters of the text of all outstanding WTOR's.

  The parameters USER and RT are added to the DISPLAY command for RES users and the R parameter has the keywords USER and ALL. A DISPLAY USER command can be issued to cause a display of one of the following: the number of RES users logged on, the number of RES users logged on and their user identifications, or information about a specified RES user. This information is whether the user is logged on, the number of the binary synchronous line or the symbolic node name of the terminal assigned by VTAM at logon (for SNA terminals) that he is using, his routing mask, his limiting priority, his default logon procedure name, and the procedure actually used for this logon.

  A DISPLAY R USER command can be issued to cause the reply IDs of outstanding WTORs for a specified RES user to be displayed. The reply IDs of outstanding WTORs for all RES users and the central operator are displayed when a DISPLAY R ALL command is issued.

  The DISPLAY RT command can be issued by any user to display the following: number of active lines (ACT parameter), which include those that are started and logged on; number of inactive lines (INACT parameter), which include lines that are not started or started but not logged on; number of inactive and active lines (ALL parameter); the user identification of the RES user at the specified remote terminal (TERM parameter) as well as the number of readers, printers, and punches assigned to his work station.

The L, LB, or LS parameter can also be specified with the ALL, ACT, or INACT parameter. L requests the display for both binary synchronous lines and SNA ports. LB requests the display for binary synchronous lines only and LS for SNA ports only.

- HOLD and RELEASE - In MFT, the HOLD command can be issued to place one specific job, jobs in all input queues, or jobs in up to four specific input queues in held status. In VS1, the HOLD command is expanded to allow up five job names to be specified in one HOLD command instead of a maximum of one and to allow the following to be placed in held status: jobs in all input queues and all output queues for locally submitted jobs (ALL parameter), jobs in up to four or all output queues for locally submitted jobs (OUT parameter), and output for a job queued in up to four or all local output queues (job name with OUT parameter).

  The central operator can also issue a HOLD command to place a remotely submitted (RES) job that is in an input queue in held status or to place in held status the output from a remotely submitted job that is queued in up to four or all output queues for the remote user.

  Note that in VS1, input queues contain both locally submitted and remotely submitted (RES) jobs. However, there is one set of output queues for all locally submitted jobs and one set of output queues for <u>each</u> logged-on remote (RES) user. The RELEASE command is expanded to provide release functions equivalent to the hold functions.

  Input and output queues are held by class or by job name in VS1. The JBN parameter is specified on a HOLD command to hold a queue by job name. When a queue is held either by class or job name, all jobs that are placed in the queue after the HOLD has been issued are also held. If a queue is held by job name (HOLD queue command with JBN parameter specified), it must be released by job name (RELEASE queue command with the JBN parameter specified). A RELEASE command without the JBN parameter will not release a queue that has been held by job name.

  When a queue is held by job name, individual jobs can be released by issuing RELEASE job name commands. A RELEASE job name command, however, does not release a job that is in a queue that was held by class (HOLD queue command without the JBN parameter). Queues that are held by class must be released by class (RELEASE command without JBN parameter).

- MODIFY - Up to 15 job classes (A-Z, 0-9) are accepted. The TYPRUN=HOLD parameter can be specified on a MODIFY command for a specific reader to cause all jobs that are read by this reader after the MODIFY command is issued to be placed in held status by job name. Jobs held in this way can be released by issuing a RELEASE command with the JBN parameter (releases all such jobs) or a RELEASE job name command (releases only the specified job). The TYPRUN=NOHOLD parameter on a MODIFY command cancels this holding process for a reader.

  The MODIFY command in VS1 also provides the capability of changing the job classes that an active initiator is assigned to handle and of passing text from the operator's console to an executing problem program. The MODIFY command is used by the central operator in an RES environment to start lines, stop lines, stop and then restart a line, and alter remote workstation output characteristics.

- MODE - A simplified format is used that is applicable to all System/370 models. The operator can no longer establish threshold values for ECC errors and instruction retry errors. The MODE command cannot be entered via an input stream.

- MONITOR - Three new functions are added to this command in VS1. When MCS and one or more display consoles are present in a VS1 configuration, a MONITOR A command can be issued to cause periodic displays on a display console of the currently active jobs in the system. The CONTROL command is used to set a time interval. Each time the interval elapses, a new display of the currently active jobs replaces the old display. The format and content of this display are the same as the display created by a DISPLAY A command. A STOPMN A command terminates this periodic display.

  A MONITOR MSG command can be issued to cause message IEE087I COMMAND RECEIVED to be issued to the operator whenever VS1 receives a command from the operating system console or input stream. A STOPMN MSG command terminates acknowledgment by VS1 when a command is received.

  A MONITOR SESS command can be issued by any RES user to cause a message to be issued to him any time an RES user logs on or off. If the T parameter is specified as well, the current time is included in the message. This monitoring facility is terminated by issuing a STOPMN SESS command.

- REPLY - An abbreviated form that contains only the message identification and response (no command verb) can be used after initialization of the communications task (1, 44 instead of R 1, 44, for example). The abbreviated form cannot be used when a reply is entered via the input stream.

- RESET - Up to five job names can be specified in VS1 instead of a maximum of one, as in MFT.

- SET - In VS1, the SET command can be issued only after system initialization is completed to change the time differential between GMT and local time or the date. SET cannot be issued during VS1 system initialization as it can in MFT. The SET command contains only the CLOCK and DATE parameters in VS1. The parameters CLOCK, DATE, PROC, and Q, which are specified via the SET command during an MFT system initialization, can be specified during a VS1 system initialization as SET parameters in response to the IEA101A message.

  Two additional SET parameters, SPOOL and SYSW, are supported in VS1. The SPOOL parameter is used to change the SPOOL file configuration that was specified at system generation and/or to cause SPOOL data sets to be formatted. The SYSW parameter can be used to specify the location of the SYS1.SYSWADS data set.

- START - This command is modified such that when a reader or writer procedure is started, a partition cannot be specified. The procedure is initiated using the next available partition.

- START INIT - Three new keywords can be specified in the PARM parameter. SWA can be specified to indicate that a scheduler work area instead of a scheduler work area data set is to be used for this initiator. EXCPVR=NO can be specified when a scheduler work area data set is used by this initiator. This keyword indicates the EXCP macro is to be used for I/O operations to the scheduler work area data set. The RESV keyword can be specified to override the reserve number of scheduler work area data sets or scheduler work area records that is specified in the initiator procedure.

- STOP - Up to five job names, unit addresses, or identifiers can be specified in one STOP command, instead of a maximum of one as in MFT.

VS1 supports new commands associated with RES (discussed under "Remote Entry Services" later in this subsection). Other new operator commands, PAGETUNE, WRITER, STARTF, and SETPRT, are provided also. The PAGETUNE command can be entered only via the operator console (not via the input stream) or placed in a CMDxxxxx member in SYS1.PARMLIB. The PAGETUNE command allows the operator to alter operation of the paging algorithm in a multiprogramming environment by altering certain of the values used by page management routines (see discussion in Section 90:35 under "Real Storage Management").

The WRITER command enables the operator to communicate requests to a JES writer. The WRITER command gives the operator significantly more control over executing writers than is provided in MFT. Parameters for the WRITER command have one-character abbreviations.

Using the WRITER command, the operator can:

- Request up to 254 additional copies of output (printed, punched, or written to tape), on a data set or a job basis

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

- Request a backspace to the beginning of the data set and a reprint of the data set from the beginning

- Request a forward space to the end of the data set

- Request a backspace to the beginning of the job class being handled and a rewriting of the data sets

- Terminate the printing of a SYSOUT data set and place it in held status. Such data sets can be released using the RELEASE command or the new ROUTE command with the HOLD=NO parameter that is provided for RES support. Printing can be resumed at the beginning of the data set or at the beginning of the page that was being printed when termination occurred.

- Alter printer line spacing (single, double, triple) for the SYSOUT data set currently being processed

The STARTF command is provided to start JES readers and writers more quickly than does the START command. Readers and writers started with STARTF are also stopped more quickly although the existing STOP command is used as usual.

When a START command is issued, an available partition must be found in which to perform the start and a device must be allocated before the JEPS monitor task receives control. A job is also created.

The STARTF command does not require the use of a partition. STARTF code is executed under the control of SVC 34, JEPS monitor, and JEPS reader and writer routines. In addition, the STARTF command does not create a job. No job queue entry is made, no spool space is allocated, and no system messages are issued. Some operator messages and all user exits are bypassed. No allocation recovery exists for the STARTF command. The device to be used must be online and not allocated to another program or the STARTF command terminates.

The STARTF command does not use cataloged procedures. Optionally, a job name (not a procedure name) can be specified in the STARTF command.

If the name begins with RDR, a reader is started. A writer is started if the name begins with WTR. A unit address must be specified in the STARTF and its type must be consistent with the type implied by the job name, if one is specified.

If a job name is not specified in a STARTF command, the type of device specified indicates whether a reader (with default name RDRF) or writer (with default name WTRF) is started for a local reader or writer. For an RES reader or writer, the user identification is used as the default job name. The name specified in the STARTF command or the default name assigned is the name used in the STOP command for a reader or writer started via STARTF.

The following can also be specified in a STARTF command:

- The name of the job to be processed first (for a reader)
- The output classes to be processed (for a writer)
- Keyword options (USER, DCB=LRECL, DCB=BLKSIZE, DCB=BUFNO, DCB=OPTCD=U, PARM, UCS, and FCB). The PARM keyword specifies the parameters for the reader or writer. If this keyword is not specified, the default values specified in the JESPARM or JESxxxxx member (LRDPARM, LPRPARM, RPRPARM, LPUPARM, RPUPARM, and PRLRECL parameters) used during system initialization are used.

The SETPRT command is provided to support the 3800 Printing Subsystem. It is used to alter or list the operator control and hardware features in effect for a specific 3800 unit. Parameters to be changed can be specified in the SETPRT command or the member name of an IEAPRTxx member in SYS1.PARMLIB can be specified. In the latter case, the member is read and the unit control block (UCB) extension for the specified 3800 is updated.

The SETPRT macro can be used in a program to cause the loading of UCS and FCB buffers in a 3800 unit. The SETPRT macro is also used to initially set or dynamically change the following printer control information: forms bursting, character arrangements to be used, number of copies, starting copy number, vertical formatting of a page, flashing of forms, initialization of printer control information, modification of copy, and blocking or unblocking of data checks.

JOB ENTRY SUBSYSTEM

The job entry subsystem (JES) is a significant new feature of VS1. It replaces MFT readers, writers, job queue management, and HASP II. JES provides centralized management of system input data, system output data, and job queue processing. It handles local system input and system output streams, allocates and manages intermediate direct access storage for this data, interfaces with RES to handle remote input and output streams, and manages allocation and processing of the job queue.

JES is designed to maximize utilization of the unit record and direct access devices involved in peripheral I/O processing and to minimize contention for the job queue. It also supports a checkpoint/restart capability.

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

Spooling operations in JES are centralized such that all system input reading, system output writing, and spool volume processing are controlled by one set of modular routines. Centralization eliminates duplication of functions within the system and improves the performance of spooling operations.

JES reader code and writer code are reentrant, which reduces the amount of storage (both virtual and real) required to service multiple input and output streams. Because all JES routines are totally pageable, real storage is allocated only to active JES tasks and without operator intervention. (In MFT, for example, main storage allocated to an inactive reader or writer partition cannot be used by other active partitions unless the operator intervenes to redefine partition allocation or to change the type of the reader/writer partition.)

The total virtual storage requirement for the minimum JES configuration (one reader, one writer, one spool volume, and minimum buffering) handling one partition is 156K. Additional virtual storage is required to handle a larger spooling configuration and more or larger buffers. If user-written output writer, job separator, or SMF routines are included in the VS1 system, they execute in the pageable JES routines area (pseudo partition) and increase its size. Use of the installation-specified selection parameters facility also increases the size of the JES routines area.

All the JES modules are pageable, except for approximately 600 bytes of I/O appendage code, which must be fixed. The JES routines pseudo partition always requires one page of fixed PQA. This page is sufficient to support one active reader and one active writer. Additional pages are fixed and unfixed as required as additional readers and writers are started and stopped.

When JES modules are link-edited during system generation, they are ordered (using ORDER and ALIGN2 control statements) by use and function on 2K page boundaries to reduce the real storage requirements and paging activity of JES modules.

Centralization of control also enables JES spooling operations to be performed more efficiently. Buffer storage for all readers and writers is contained in one pool, buffer storage for all spool volumes is contained in one pool, and direct access spool space (the SYS1.SYSPOOL data set) for all system input and output data is shared. These pools are in the JES buffer pool area.

Buffer storage and spool space are managed (allocated, opened, closed, and deallocated) by JES routines that are tailored to provide efficient spooling operations in a paging environment. Buffer storage is allocated for JES operations such that the number of page faults incurred is minimized, and direct access storage is allocated such that data transfer time for JES operations is minimized.

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

Any number of readers and writers are supported by JES, subject only to the availability of system resources. MFT supports 3 readers and 36 writers maximum. The maximum number of readers and the maximum number of writers that can be started for a given VS1 control program (both JES and RES requirements) can be specified at system generation. The size and number of spool buffers and the number of spool volumes can also be indicated at system generation. Defaults are assumed for parameters not specified.

All the JES parameters that are specified during system generation except those associated with the resident job list facility (JOBQINT, JOBQEXT, and JOBQNXT) are placed in the JESPARMS member in SYS1.PARMLIB. The JES parameters specified at system generation can be changed during system initialization by specifying a JESxxxxx member with different JES parameters or the JESPARMS member with modified JES parameters. The three job list parameters are system parameters that can be overridden by the operator during system initialization. Thus, for example, the number of readers and writers supported can be increased or decreased without performing a system generation.

During system initialization, enough virtual storage is allocated to the JES areas to support the maximum JES configuration as defined for this system initialization. Virtual storage from the JES area is allocated as readers and writers are started.

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

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

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

<u>JEPS</u>	<u>JECS</u>
• Monitor task	• Spool management
• JES readers	• Buffer management
• JES writers	• DASD work area management
	• Job list management
	• SWADS management

JEPS tasks execute under the control of TCBs that have higher dispatching priority than any of the 52 partitions that can be defined. However, the dispatching priority of readers and writers can be lowered by changing reader/writer procedures. JECS routines operate under the control of the TCB of the task that requested their service. The components, functions, and data flow of JES are shown in Figure 90.20.1.

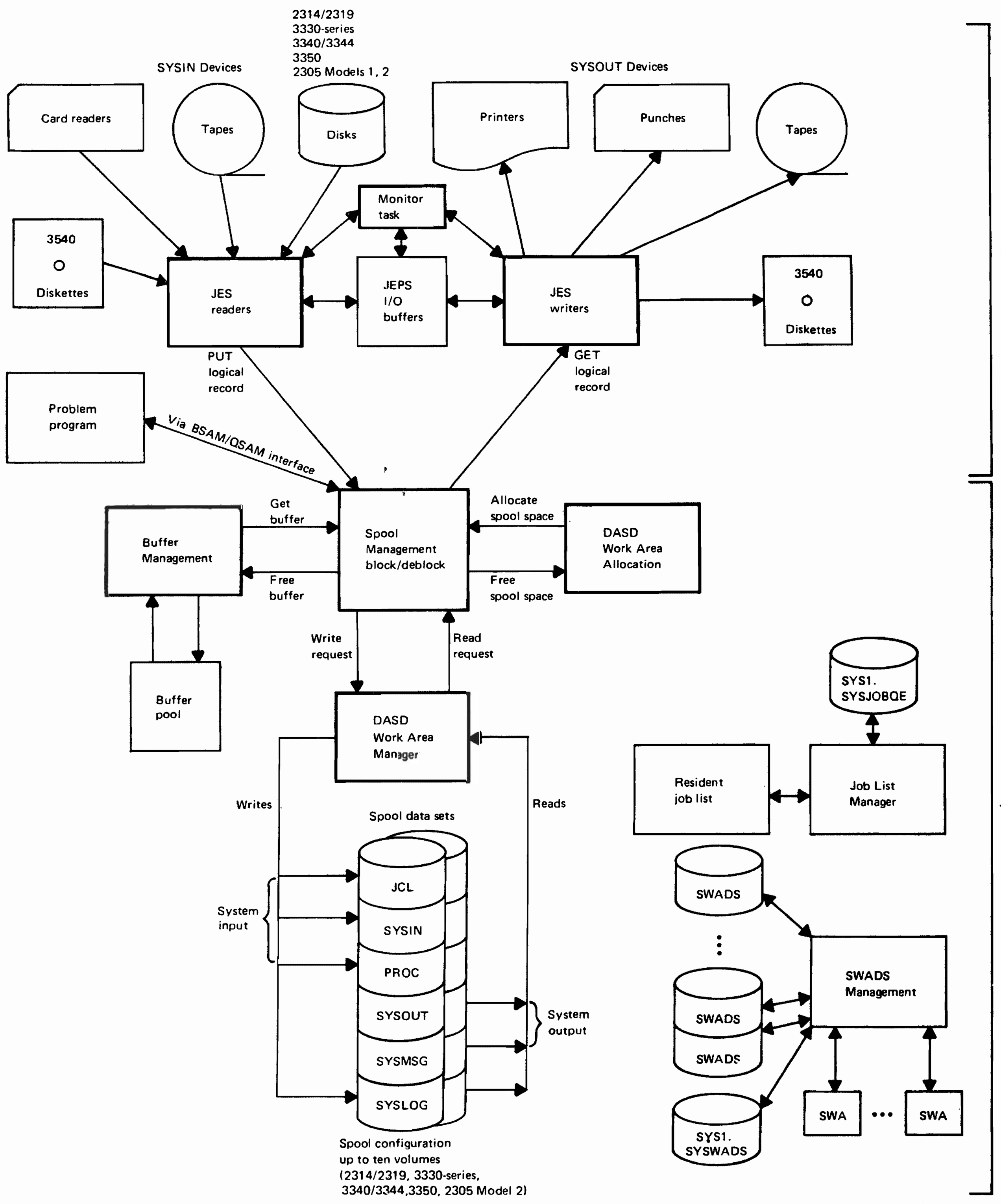


Figure 90.20.1. Components, functions, and data flow of JES

JEPS Monitor Task

The monitor task is responsible for initializing JES readers and writers, which operate as subtasks of the monitor task. When a START or STOP command for a reader or writer procedure is entered, an initiator or terminator is brought into the first available partition in order to invoke the monitor. The monitor task is also automatically invoked to terminate a reader task when end of file occurs on a SYSIN device other than a card reader.

The monitor task obtains or releases buffers for a reader/writer from the reader/writer virtual storage area allocated in the JES area during system initialization, and attaches or detaches a reader/writer task. A reader/writer is started as long as this request does not cause the maximum number of readers/writers specified during system initialization to be exceeded.

OS MFT reader and writer procedures are not compatible with those of JES. Only two reader procedures are IBM-supplied for VS1. One is for unit record SYSIN devices and the other is for tape SYSIN devices. In MFT, there are three IBM-supplied reader procedures. The MFT reader procedures RDR400 and RDR3200 are not provided in VS1.

Optionally, a VS1 reader procedure can contain a SYSABEND or SYSUDUMP DD statement. (IBM-supplied reader procedures for VS1 do not contain a storage dump DD statement.) The dump DD statement causes the system to take a virtual storage dump if the reader task abnormally terminates. If the SYS1.DUMP data set is available, an SVC dump is written to this data set. Otherwise, a SYSABEND or SYSUDUMP dump is provided as specified in the dump DD statement. This dump facility for reader tasks is not provided in MFT.

Two writer procedures are provided in VS1 instead of one as in MFT. One is for a printer output device (with procedure name WTR) and the other is for a tape output device (with procedure name WTRT). Like reader procedures, optionally, writer procedures in VS1 can contain a SYSABEND or SYSUDUMP DD statement to cause a virtual storage dump to be taken if the writer terminates abnormally. The writer procedures in VS1 contain additional parameters to support the 3800 Printing Subsystem.

The dispatching priority of a reader or writer can be lowered by specifying the dispatching priority parameter in the PARM field of the EXEC statement for the reader or writer procedure. Making the dispatching priority of readers and/or writers lower than that of one or more partitions (either system task or problem program) may improve system throughput.

A partition number of from 00 to 14 can be specified in the dispatching priority parameter of the PARM field. The dispatching priority of the reader or writer is determined as follows:

- If the partition number specified is not greater than that of the highest numbered partition currently defined, the reader or writer is assigned a dispatching priority one less than the dispatching priority of the partition specified. When a partition that is part of the time-sliced group of partitions or the dynamic dispatching group of partitions is specified, the reader or writer is assigned a dispatching priority one less than that of the lowest priority partition in the time-sliced group or the dynamic dispatching group.

- If the partition number specified is greater than that of the highest numbered partition currently defined, the reader or writer is assigned a dispatching priority one less than the dispatching priority of the lowest priority (highest numbered) partition currently defined.

When the normal dispatching priority for a reader is desired, the dispatching priority parameter should be blanks, XX, or omitted. Normal dispatching priority for a writer is assigned by omitting the dispatching priority parameter from the PARM field.

JES Readers

A local input stream can be read from any card reader, tape unit, or direct access device that is supported by VS1 and from the 3540 Diskette Input/Output Unit. An input stream on disk can be read from sequential data sets and individual members of partitioned data sets.

The 3540 can be used as a SYSIN device in the following ways:

- A job stream (job control and data) can be contained in one diskette data set.
- A job stream (job control and data) can be contained on card, tape, or disk while additional input data for the job stream is contained in one or more diskette data sets. These diskette data sets are called associated data sets. An associated data set can be contained on multiple diskettes. The maximum number of readers that can be started to read associated data sets is specified as a JES parameter.
- A job stream (job control and data) can be contained in a diskette data set while additional input data for the job stream is contained in one or more diskette (associated) data sets.

Card input streams are read by a special JES access method (JAM). Column binary reading is not supported. Tape and disk input streams, which can be blocked, are read using a special interface to QSAM. Command-chained reads are initiated for a card SYSIN device. The number of cards read per command-chained channel program can be specified at system generation. If this number is not specified, a channel program designed to read five cards is initiated for each I/O operation to a card SYSIN device.

A 3540 sequential access method is used to read 3540 SYSIN data sets. A command-chained channel program that contains 26 read commands is initiated for each read of a SYSIN diskette device. Therefore, each I/O operation reads in a full track of records from the diskette.

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

VS1 input stream data can consist of job control statements (including in-stream procedures and requests to execute procedures), input data sets (multiple per job step), and operator commands. If records other than operator commands are found between jobs in VS1 (such as misplaced data records and/or job control statements), they are flushed, as in MFT. However, in VS1, the operator is informed of the flushing operation via a message.

A JES reader inspects the JOB statement, and defaults for job class and priority are supplied, if necessary. A unique job number is appended to the job name specified to eliminate the possibility of duplicate job names. This job number is used internally only. Job control statements, commands contained within jobs, input data, and any requested procedures that are contained in user procedure libraries or concatenated procedure libraries are passed to the spool management routine of JECS to be written in spool data sets. Procedures that are contained in SYS1.PROCLIB are not written in spool data sets. Commands contained within a job are processed during the initiation of the job. Commands not contained within a job are processed when encountered.

When a job has been completely read, the JES reader builds certain control blocks that describe the job. The job is placed in an input work queue and marked in held status if so indicated in the JOB statement. As in MFT, jobs are queued by job priority within job class and first-in, first-out within equal priorities. The total number of statements read and the real time taken to read the job is made available to SMF, if SMF is included in the VS1 control program.

JES Writers

A JES writer transcribes SYSOUT spool data sets created by job steps and system message spool data sets that contain job scheduler messages and job control statements. The local SYSOUT devices supported are the printers, punches, and tape units supported by VS1 and the 3540 Diskette Input/Output Unit. Up to 19 SYSOUT data sets from the same job that have the same SYSOUT class can be written on one diskette. A special JES access method is used to support printers and punches. Column binary punching is not supported. An interface to QSAM is used to handle a tape SYSOUT device and 3540 SYSOUT data sets are written by a 3540 sequential access method.

As for card reading, command-chained channel programs are used for card punching and printing operations to enable SYSOUT devices to operate near rated speeds. Six records are printed or punched per channel program unless a different value is specified by the user at system generation. A channel program that contains 26 write commands is initiated for each I/O operation to a diskette device to write a full track of records.

The Two-Line Card Print and Multiline Card Print features of the 3525 Card Punch are supported by JES writers for printing operations on cards punched from SYSOUT data sets. The data punched on each card can consist of one or two lines of the data punched in the card (interpret function on lines 1 and 3). The FUNC=I keyword must be specified in the DCB parameter of the SYSOUT DD statement when interpretation of a punched SYSOUT data set is desired.

Alternatively, up to 25 lines of data supplied by the user in the SYSOUT data set can be printed on each card punched from a SYSOUT data set. When this program-controlled printed output facility is used, the SYSOUT data set must include after each punch record a print record containing the appropriate line control character for each line that is to be printed on the card.

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

A JES writer handles all the SYSOUT spool data sets of the same class that are present for a given job before attempting to process SYSOUT spool data sets of the same class that belong to another job. All the job control statements and system messages for a job, which are placed in a system message spool data set instead of the SYS1.SYSJOBQE data set during processing of the job, are printed before all the SYSOUT spool data sets for the job. (MFT writers intersperse the printing of job control statements and system messages with the printing of SYSOUT data sets.)

Optionally, the job log facility can be included in a VS1 control program. When this facility is present, the text contained in all WTO and WTOR macros (including write-to-programmer messages), replies to WTOR messages, and WTL macros (if system log support is not present) for each job processed are written in the system message data set for the job. Job-related messages that are written to the operator by system routines using WTO and WTOR macros are included in the system message data set as well. The first line of text in MLWTO macros is also saved. The saved messages are printed at the beginning of the output for a job since the system message data set is processed before any SYSOUT data sets for a job.

Multiple copies of a given SYSOUT spool data set can be requested in VS1 via the COPIES parameter (not provided in MFT) for the SYSOUT DD statement, in addition to via the WRITER command. Multiple copies of a SYSOUT data set cannot be written to a 3540 diskette device. When printing is to be done on a 3800 Printing Subsystem, group values can be specified in the COPIES parameter. A group value indicates the total number of copies to print and total number of times each page is to be printed.

For example, if group values of 1, 3, and 2 are specified in a COPIES statement, six copies of the data set will be printed in three groups. The first group contains one copy of each page. The second contains three copies of each page, and the third group contains two copies of each page.

Three other new facilities for output writers in VS1 are a checkpointing capability for SYSOUT data sets, an end-of-job separator option, and IBM- and user-defined translate tables for unprintable characters. Optionally, the PARM field of the EXEC statement in a writer procedure can contain a checkpoint interval value, which is a multiple of ten. When such an interval is specified, the writer checkpoints the SYSOUT data set it is processing each time the specified number of pages have been printed or punch/tape logical records have been written. When such checkpoints are taken, if the system is warm-started before the SYSOUT data set is completely written, the operator can specify that writing of the data is to resume at the last checkpoint or at the beginning of the SYSOUT data set.

The end-of-job separator option, supported only for SYSOUT data that is written to a local or non-RTAM-supported remote printer, can also be specified in the PARM field of the EXEC statement in a writer procedure. When this option is chosen, one, two, or three output separator pages (as specified in the PARM field) are printed after the last SYSOUT data set that is printed for each job. The end-of-job separator pages are identical in content to the output separator pages that are printed at the beginning of the printer output of the job (when output separation is specified) except that no asterisks are printed after the last page.

The end-of-job separator facility is designed to assure the user that he has received all the printed output from his job. The end-of-job separator facility can be used only with the system output writer (not with user-written output writers) and requires the specification of output separation as well.

VS1 provides the option of having output writers use a translate table to translate unprintable characters to blanks. The IBM-supplied translate table, which is print-train-independent, or a user-defined translate table can be used. Multiple translate tables, each of which matches a specific print chain, for example, can be user-defined, named, and added to the JES load module (IEFTRT control section). The name of the translation table to be used by a writer is specified in the PARM field of the EXEC statement in the writer procedure or the PARM field of the START writer command.

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

A selective posting facility is implemented in VS1 for JES writers and initiators. Selective posting of a waiting JES writer when a SYSOUT data set is queued is like selective posting of a waiting initiator when an input job is queued. Selective posting is discussed later in this subsection under "Job Scheduler".

Problem Program Access to SYSIN and SYSOUT Data

In VS1, problem programs access SYSIN and SYSOUT spool data sets via QSAM and BSAM, as in MFT. However, in VS1 these sequential access methods interface with a device-independent JES translator, which interfaces with JECS to access SYSIN and SYSOUT spool data sets on spool volumes.

The JES translator module is automatically invoked when a SYSIN/SYSOUT data set is specified by a job step. The translator module is entered each time the problem program requests the reading of a SYSIN record or the writing of a SYSOUT record. The JES translator reformats the request as necessary and passes it to JECS. When JECS has processed the request, this fact is indicated to the JES translator, which posts the appropriate control blocks and, in the case of a SYSIN request, makes the record available to the problem program.

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

JECS Spool Management

Spool management is the central facility that controls all access to spool data sets, as shown in Figure 90.20.1. It receives and processes service requests from JES readers, JES writers, job scheduler components, and executing problem programs. Spool management processing consists of blocking and deblocking system input and output records and requesting services from other JECS components.

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

The OUTLIM facility, available to MFT users only via SMF, is a standard feature of JES. This facility allows the user to indicate the maximum number of logical records that can be placed in a SYSOUT spool data set (up to a maximum of 16,777,215). Spool management also ensures that the OUTLIM quantity for SYSOUT spool data sets is not exceeded and maintains control blocks that indicate all the system input and output data sets associated with each job.

JECS Buffer Management

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

The number of buffers required for optimum spool performance is a function of the number of JES readers, JES writers, partitions, and opened spool data sets that can be active concurrently. If too few buffers are provided, loss of spool performance can occur. Hence, it is better to overestimate than to underestimate buffer requirements. The allocation of more spool buffers than are actually required does not affect system performance since real storage is allocated to spool buffers only if they are used. A maximum of 999 buffers can be specified. (See OS/VS1 System Generation Reference, GC26-3791, for estimating spool buffer needs.)

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

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

DASD Work Area Management

The allocation and deallocation of direct access space to spool data sets and the reading and writing of spool data sets are handled by the DASD work area allocation routine and the DASD work area manager, respectively. Spool volumes contain JCL, SYSIN, PROC, SYSOUT, and system message spool data sets, as well as write-to-programmer messages and the two system log data sets.

The spool volume configuration (DASD work area) can consist of up to ten permanently mounted direct access volumes with a maximum of 64K tracks allocated to the SYS1.SYSPOOL data set. Any mixture of the following direct access device types can be included in the spool configuration (SYS1.SYSPOOL data set): 2314/2319, 3330-series (all models), 3340/3344 (all models), 3350 (in native or 3330 compatibility mode) and 2305 Model 2. The volume serial numbers of the volumes in the spool configuration can be specified at system generation. The volumes specified can be changed during system initialization via a different specification in the JESPARMS member, via specification of a JESxxxxx member, or by the operator via the SET parameter SPOOL. Spool volumes can be added or deleted. Spool volume formatting can also be requested via the SPOOL parameter.

The spool volumes to be used must be mounted prior to an IPL. Any volumes in the specified spool configuration that have not been mounted before the IPL are deleted from the spool configuration for this IPL and the operator is notified. A re-IPL is required to alter the spool volume configuration after system initialization.

Spool volumes need not be dedicated to spooling; however, arm movement can be minimized and increased performance obtained when spool volumes are dedicated to spooling or contain only low-usage data sets in addition to spool data sets. SYS1.SYSPOOL and SYS1.PAGE data sets should not be placed on the same volumes, if possible.

Spool data sets are sequentially organized and contain variable-length blocked spanned records that are written without a key. For card input, blanks after the last character punched are deleted from the resulting logical record. For printer and punch output, blanks after the last character to be printed in a line or punched in a card are deleted from the logical record. This truncation eliminates using spool space to store insignificant blank characters.

Spool volumes must be preformatted with physical records the size of the spool buffers to be used. Preformatting is done during an initial IPL when the operator requests spool volume formatting, or because it is determined that spool volumes require formatting. Reformatting is not required thereafter unless spool buffer size is changed.

The spool buffer size chosen must be a minimum of 436 and a maximum of 99,999 bytes. The default buffer size is 880 bytes. Spool buffer size cannot be larger than the full-track capacity of the smallest capacity track in the spool device configuration. Therefore, if a 2314/2319 is part of the spool configuration, buffer size cannot exceed 7294 bytes regardless of the other direct access device types in the spool configuration. The track overflow feature is not supported for spool data sets, but rotational position sensing is used when it is present for a spool device.

Direct access space on spool volumes is allocated to spool data sets in terms of a logical cylinder instead of a physical cylinder. A logical cylinder consists of a number of tracks, from two to a maximum of 255, all of which need not be contained in the same physical cylinder. The number of physical tracks in a logical cylinder depends on the size of the spool buffer (BUFSIZE parameter), the number of spool buffers that can fit on one track, and a spool allocation unit specification in bytes (ALCUNIT parameter).

The BUFSIZE and ALCUNIT parameters can be specified at system generation and overridden during system initialization. (A cold start rather than a warm start must be performed after either parameter is changed.) The system defaults of 880 bytes for BUFSIZE and 28,672 bytes for ALCUNIT are used if these parameters are not user-specified.

The number of tracks in a logical cylinder is calculated for each spool device type using the formula ALCUNIT / (BUFSIZE x number of buffer records per track). When the default values for both BUFSIZE and

ALCUNIT are used, there are five tracks in a logical cylinder for 2314, 2319, and 3340/3344 spool devices; three tracks in a logical cylinder for 3330-series (all models), 3350 in 3330-compatibility mode, and 2305 Model 2 spool devices; and two tracks in a logical cylinder for 3350 native mode spool devices.

The default specifications for BUFSIZE and ALCUNIT provide approximately 28K bytes of spool space per logical cylinder for each spool device type. These values are designed to optimize spool space usage and allocation for installations with spool data sets (JCL, SYSIN, SYSOUT, etc.) that vary in size. The minimum amount of space that can be allocated to the SYS1.SYSPOOL data set is 20 logical cylinders.

If an installation consistently has jobs that require small spool data sets, specification of a smaller ALCUNIT value, which results in a smaller logical cylinder, can avoid wasting spool space. If an installation consistently has jobs that require large spool data sets, specification of a larger ALCUNIT value, which results in a larger logical cylinder, will cause fewer calls for spool space allocation. However, if neither of these two conditions exists, the default value for the ALCUNIT parameter should be used.

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

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

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

At system generation, a threshold value percentage (from 40 to 90) for spool capacity can be specified or the default percentage of 80 can be used. The operator is notified when the threshold percentage of spool capacity becomes allocated during system operation. At this time, the system automatically holds the input queue to prevent further job selection and stops all started readers. The operator should ensure that writers are started for those SYSOUT classes that have SYSOUT data sets queued and start another writer, if possible. The remaining spool space is then allocated only for starting another writer, processing jobs currently initiated, and terminating problem programs and system readers.

If the percentage of spool allocation continues to rise above the threshold value, the operator is informed of every 5 percent increase. When the spool volumes become so full that only a special reserve of logical cylinders is available, the operator is asked whether the job currently requesting spool space should be canceled. Depending on the reply, the job is canceled or placed in a wait state until a logical cylinder becomes available. The reserve cylinders are allocated only for the purpose of starting another writer or canceling a job as a result of an affirmative operator reply to the cancel request.

The operator continues to be informed of the percentage of allocated logical cylinders until the percentage is reduced to the threshold value. When the allocation percentage decreases to a value of 10 percent less than the threshold value, the operator is informed that spool space is no longer critical. JES readers that were stopped can be restarted and the input queue can be released.

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

- Spool space is allocated and deallocated more quickly via use of an in-storage map rather than by DADSM routines, which must process VTOCs to locate and return direct access space. OPEN and CLOSE processing is also eliminated.

- Spool space is allocated to minimize direct access device arm movement.

- Spool space for a given spool data set is allocated across I/O devices, if possible, to enable spool I/O operations to be overlapped and to help balance the I/O load.

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

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

Job List Management and SWADS Management

The job queue organization and management implemented in VS1 is considerably different from that implemented in MFT. They are designed to reduce contention for the job queue, such as can occur in MFT, and to eliminate duplicate job queue processing code through centralization of job queue processing, which can reduce paging activity.

The basic difference between job queue organization in MFT and VS1 is that all of the work queues and much of the job scheduling data that are contained in the SYS1.SYSJOBQE data set in MFT have been removed from this data set in VS1 and placed in other locations. In VS1, a resident job list in virtual storage, a SYS1.SYSJOBQE data set, scheduler work area data sets (SWADS), a system scheduler work area data set (SYS1.SYSWADS), and scheduler work areas (SWAs) in virtual storage are supported to contain job queues (input and output work queues) and job scheduling data. Job queue processing is handled by the job list manager and job scheduling data is accessed by the SWADS manager.

The resident job list is maintained in virtual storage in pageable SQA. Each entry in the job list represents a unit of work that is to be performed by a job scheduler routine and points to the location of control blocks in SYS1.SYSJOBQE that describe the unit of work. Job list entries are connected to form work queues, which in MFT are in SYS1.SYSJOBQE. The resident job list contains entries for problem program jobs only. It does not contain any entries for system tasks that are started.

Included in the resident job list are from 1 to 36 (A to Z, 0 to 9) input work queues for all locally and remotely submitted (RES) jobs, from 1 to 36 (A to Z, 0 to 9) output work queues for the SYSOUT data sets from locally submitted jobs, and from 1 to 36 (A to Z, 0 to 9) output work queues for each remote (RES) user. Separate hold queues are not maintained for jobs. When a HOLD command is issued for a job, the job remains in the input or output queue of which it is a part and is marked as being in held status.

The resident job list is maintained by the job list manager. Job scheduler routines access this list via the job list manager. JES readers place an entry in the appropriate input queue whenever the reading of a locally or remotely submitted job completes. Terminators place entries in appropriate output queues whenever a locally or remotely submitted job completes. Initiators and JES writers obtain the work they are to handle from this list.

During system initialization, the maximum size of the resident job list of this IPL is calculated using JOBQINT, JOBQEXT, and JOBQNXT parameters. These parameters are specified during system generation and can be overridden during system initialization via a NIPxxxxx member in SYS1.PARMLIB or by the operator in response to message IEA101A.

The amount of space within pageable SQA that is actually allocated to the resident job list during system initialization is the amount specified in the JOBQINT parameter. The JOBQEXT and JOBQNXT parameters are used to dynamically extend the size of the resident job list during system operation, if necessary, as discussed later.

The SYS1.SYSJOBQE data set in VS1 contains job requirements data and accounting data for the problem program jobs that are queued in the resident job list. It does not contain any of the work and hold queues, job scheduler tables, and system messages that are contained in SYS1.SYSJOBQE in MFT. The control blocks in the SYS1.SYSJOBQE data set in VS1 are primarily job-related instead of job-step-related.

The following control blocks are placed in SYS1.SYSJOBQE in VS1: disk entry record (DER), job management record (JMR), SYSOUT class directory (SCD), data set blocks for SYSOUT data sets (DSBs), job accounting table (JACT), and routing table (RTBL). These records are created by reader and interpreter routines. Logical records in the SYS1.SYSJOBQE data set in VS1 are 176 bytes as in MFT; however, job records are blocked in SYS1.SYSJOBQE in VS1 (five logical records per block) to improve performance. SYS1.SYSJOBQE in VS1 is managed by the job list manager.

SYS1.SYSJOBQE is a sequential data set that can reside on one or more volumes, which need not be all of the same direct access device type. The volume(s) that contain the SYS1.SYSJOBQE data set(s) can be specified at system generation via the JOBQVOL parameter. The system residence volume is assumed if the JOBQVOL parameter is not specified. The system generation specification can be overridden during system initialization via the JESPARMS or a JESxxxxx member in SYS1.PARMLIB or the operator can enter the SET parameter Q=cuu/CHNG,F to specify the SYS1.SYSJOBQE configuration if it has not been previously specified, change the previous specification, and/or cause formatting of SYS1.SYSJOBQE to be done during system initialization.

In VS1, job scheduler tables and control blocks are placed in the SYS1.SYSWADS data set, SWADS data sets, and optionally, SWAs. Scheduler data includes the job control table (JCT), step control table (SCT), step control table extension (SCTX), step input/output table (SIOT), account control table (ACT), volume table (VOLT), job file control block (JFCB), data set enqueue table (DSENQ), and data set name table (DSNT). Scheduler data is always written in 176-byte records.

The job scheduler tables and control blocks for all the system tasks and generalized start problem program jobs that are initiated are placed in one SYS1.SYSWADS data set. The 176-byte records in SYS1.SYSWADS are sequentially organized and unblocked. The location of this data set can be indicated by cataloging it. Alternatively, it can be specified during system initialization via the SET parameter SYSW. This parameter can be placed in a SETxxxxx member in SYS1.PARMLIB or entered by the operator. The SYS1.SYSWADS data set is assumed to be on the system residence volume if its location is not specified.

The job scheduler tables and control blocks for a problem program job that is not initiated via a START command are placed in the SWADS data set or SWA that is allocated to the initiator that schedules the job. In VS1, either an SWADS or an SWA must be assigned to each active initiator. Regardless of whether an SWADS or SWA is assigned to an initiator, the scheduler data for any system tasks and generalized start jobs it schedules is written in the SYS1.SYSWADS data set. The SYS1.SYSWADS data set, SWADS data sets, and SWAs are maintained by SWADS management.

An SWADS is allocated to an initiator by including an IEFRDR DD statement in the initiator procedure. The SWADS for an IBM-supplied initiator is a temporary data set that is allocated and formatted when the initiator is started. An SWADS (which cannot be allocated on a virtual volume) is sequentially organized and contains a single extent of a maximum of 32,767 unblocked 176-byte records. Allocation parameters in the SWADS DD statement are used unless they are overridden by the operator via the PARM parameter in the START initiator command.

A permanent instead of temporary data set can be allocated to an SWADS. When a permanent data set is allocated, the SWADS is not formatted when its associated initiator is started unless the SWADS has not been formatted previously or the FMT=Y parameter is specified as a PARM value on the EXEC statement in the initiator procedure or in the START initiator command.

Whenever a job is selected by an initiator and interpreted, the interpreter places all the scheduler tables and control blocks for the entire job in the SWADS for the initiator. Thereafter, initiator, allocation, and terminator routines access the SWADS (via the SWADS manager) to schedule steps of the job. The scheduler data for each successive job that is processed by the same initiator overlays the scheduler data in the SWADS for the previous job. The SWADS data set that is allocated to an IBM-supplied initiator is released when the initiator is stopped.

An SWA is allocated to an initiator by including the SWA keyword in the PARM field of the EXEC statement in the initiator procedure or in the START initiator command that is issued by the operator. If the SWA parameter is supplied, either in the initiator procedure or via the START initiator command, an SWA is used even though an IEFRDR DD statement for an SWADS is present in the initiator procedure. The presence of an SWADS DD statement causes an SWADS data set to be allocated, however, even though an SWA is actually used. The SWA parameter indicates the number of 176-byte records the SWA is to contain.

When an initiator is started and the SWA parameter is specified, an SWA is allocated to the initiator and formatted. The SWA is allocated from pageable PQA virtual storage in the partition in which the initiator is started. There must be at least 64K of virtual storage remaining in a partition after the SWA has been allocated. Thereafter, when a job is selected for processing, the interpreter places scheduler data in the SWA in virtual storage for access by job scheduler components. The scheduler data for each successive job that is processed by the same initiator overlays the scheduler data in the SWA for the previous job. When an initiator is stopped, the SWA that is allocated to it is released.

Use of an SWA in virtual storage instead of an SWADS data set for an initiator can provide improved performance, particularly for jobs whose total job time includes a large amount of scheduler processing time, assuming the system can absorb the increase in paging activity that results from using an SWA.

Note, however, that jobs scheduled by an initiator with an SWA assigned cannot use automatic step restart, automatic checkpoint restart, and deferred checkpoint restart facilities, since the restart reader and checkpoint restart routines require an SWADS data set. When a system failure occurs, jobs in progress whose scheduler data is in an SWA are not reenqueued in the resident job list and their queued SYSIN data sets, if any, are deleted. However, the SYSOUT data sets for these jobs, if any, are queued for processing by a JES writer.

SYS1.SYSJOBQE, SYS1.SYSWADS, and SWADS data sets can be placed on 2314/2319, 3330-series (all models), 3340/3344 (all models), 3350 (in native or 3330-compatibility mode), and 2305 Model 2 direct access devices. RPS is supported when it is present for the device. The same device type need not be used for each of these three types of system data sets and all SWADS need not be on the same type of direct access device.

In VS1, as in MFT, space can be reserved for job queue and job scheduler data that is used in the event that the nonreserved space becomes exhausted. Such space is reserved for initiators to enable them to complete processing of the job they are scheduling and to start writers so that job queue space is freed. In VS1, however, a SYS1.SYSJOBQE extension program is also provided that enables the operator to add space to SYS1.SYSJOBQE during system operation. The amount of virtual storage allocated to the resident job list can also be dynamically expanded during system operation.

The following parameters that allocate reserved space can be specified during system generation and overridden during system initialization (via a NIPxxxxx member in SYS1.PARMLIB or by the operator in response to message IEA101A):

- JOBQLMT to specify the number of records to reserve in SYS1.SYSJOBQE for each initiator started. These reserve records enable an initiator to complete processing of the current job.

- JOBQLST to specify the number of entries in the resident job list that are to be reserved for each initiator started. These entries enable an active job to be terminated.

- JOBQTMT to specify the number of records that are to be reserved in SYS1.SYSJOBQE for starting one initiator, one writer, and the SYS1.SYSJOBQE extension program in the event that queue space becomes critical

- SYSWTMT to specify the number of records that are to be reserved in SYS1.SYSWADS for starting one initiator, one writer, and the SYS1.SYSJOBQE extension program in the event that queue space becomes critical

The RESV parameter can be specified in an initiator procedure or START initiator command to reserve records in an SWADS data set or an SWA. The reserved space is used to terminate abnormally a job whose space requirements exceed the nonreserved SWADS/SWA space.

The JOBQEXT and JOBQNXT parameters are specified during system generation and can be overridden during system initialization to control dynamic expansion of resident job list space. The JOBQEXT parameter indicates the number of job list entries that are to be allocated when existing space in the job list is exhausted. The amount of virtual storage required for the number of entries specified is dynamically obtained from pageable SQA each time the job list area becomes totally allocated and additional space is required. Such an extension of the job list area is automatically released when it is no longer required. The JOBQNXT parameter indicates the maximum number of times (from 0 to 255) job list space can be extended.

The SYS1.SYSJOBQE extension program must be executed to add space to SYS1.SYSJOBQE during system operation. The operator initiates this program in a partition via a START JQEXD command, normally in response to the message JOBQ SPACE CRITICAL - START A JOBQ EXTENSION PROCEDURE (IEF052E). This program is authorized to execute in a system task partition. The START command specifies the name of the extension data set that is to be added to the SYS1.SYSJOBQE data set originally defined. Only one extension data set is added per start of the extension program.

The maximum number of data sets that can be defined in a SYS1.SYSJOBQE configuration, including both initially allocated and extension data sets, is ten. A minimum of two tracks must be allocated to each SYS1.SYSJOBQE data set. More than one SYS1.SYSJOBQE data set can be placed on the same volume. One is named SYS1.SYSJOBQE, while the others are named SYS1.SYSJOBQx, where x is chosen by the operator to form a unique data set name for the volume. The same direct access device type need not be used for all SYS1.SYSJOBQE data sets.

When an extension SYS1.SYSJOBQE data set is no longer required, the job list manager releases it, that is, deletes it from the set of SYS1.SYSJOBQE data sets being used. In order to unallocate the space assigned to a released extension data set, the operator can execute the SYS1.SYSJOBQE extension program to perform the scratch function.

Space in SYS1.SYSWADS, an SWADS data set, or an SWA cannot be extended dynamically during system operation. A job is abnormally terminated if the space being used for its scheduler tables and control blocks runs out during processing of the job.

Job List Verification Program and JESDUMP Service Aid

The job list verification program and JESDUMP service aid are standard functions of JES. The job list verification program is designed to determine whether an error that could cause abnormal system termination exists in the resident job list after it has been manipulated by the job list manager. If an error is detected, a virtual storage dump can be taken using the JESDUMP program (QMGRDUMP entry). The QMGRDUMP entry to the JESDUMP program can also be used to take a dump after the SWADS manager encounters an I/O error. JESDUMP can also be used to take a virtual storage dump after certain spooling conditions are detected (JECSDUMP entry).

If the job list verification and JESDUMP programs are to be used, they must be enabled by modification of the appropriate bytes within JES. This can be done on a selective basis (that is, only at certain times) using the alter/display function of the primary system console device or on a permanent basis using the HMASPZAP service aid.

The job list verification program and the QMGRDUMP entry into the JESDUMP program are activated by altering JESCT+X'0C' to X'C0'. This setting causes the job list verification program to be executed following each manipulation of the resident job list by the job list manager to check the contents of the resident job list for accuracy. If no error is found, processing continues without a dump. If an error is found, the JESDUMP program is entered at the QMGRDUMP entry and a formatted SVC dump of virtual storage is written in the SYS1.DUMP data set. The X'C0' setting for JESCT+X'0C' also causes entry into the JESDUMP program at the QMGRDUMP entry whenever an I/O error occurs during SWADS manager processing of SYS1.SYSJOBQE, SYS1.SWADS, or an SWADS.

The storage dump taken at the time the system abnormally terminates because of a resident job list error does not necessarily contain the information required to locate a job list error that occurred earlier. The JESDUMP program provides a virtual storage dump at the time the resident job list error occurs.

Entry into the JESDUMP program at JECSDUMP is made after spool manager processing occurs when location JESCT+X'0' is altered. When JESDUMP is entered at this point, it checks the passback code from the spooling manager and the value in JESCT+X'0' to determine whether a dump should be taken. The value placed in JESCT+X'0' indicates which one or more of the following conditions should cause a dump to be taken: JECS errors other than those that follow, two OPEN macros issued without an intervening CLOSE, data set not found, spool space exceeded, JECS work area (LRCB) not open, OUTLIM value exceeded, end of file, OUTLIM value equaled, space critical, and data set empty.

If the passback code does not indicate any of the conditions specified in JESCT+X'0', processing continues without a dump. If a specified condition does exist, a formatted SVC dump of virtual storage is written to the SYS1.DUMP data set.

JOB SCHEDULER

The basic design changes embodied in the VS1 job scheduler are inclusion of the interpret function as part of job scheduling and access to the resident job list, the SYS1.SYSWADS data set, SWADS, and SWAs in addition to SYS1.SYSJOBQE to obtain the jobs to be processed and job scheduling data.

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

Initiator

A VS1 initiator is pageable and a large portion of it is reentrant. The sequence of loading initiator modules in VS1 is changed to eliminate the loading of job selection routines at times when they are not required. This is done to improve the performance of job scheduling in VS1.

As in MFT, the initiator operates in a partition to perform its scheduling function. Initiators schedule problem program job steps (both pageable and nonpageable), system tasks, and JES readers and writers. Initiators interface with the job list manager to access the resident job list and the SYS1.SYSJOBQE data set. Initiators interface with SWADS management to access SYS1.SYSWADS, SWADS, and SWAs.

The EXCPVR=NO parameter is supported for the EXEC statement of initiator procedures in VS1. If specified, this new parameter indicates that the EXCP macro instead of the EXCPVR macro is to be used to initiate I/O operations to the SWADS for this initiator, if any. If this parameter is not specifed, the EXCPVR macro is used.

Since the I/O tables associated with an EXCPVR request are maintained in fixed SQA, less paging activity will usually be required to process an SWADS using EXCPVR instead of EXCP, and system performance may be improved. Use of EXCPVR for SWADS processing increases the fixed SQA space requirements. Therefore EXCPVR=NO can be specified when the real storage available for fixed SQA space is limited. The IBM-supplied initiator procedures do not contain the EXCPVR=NO parameter. The EXCPVR=NO parameter can be specified in the START command for an initiator.

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

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

A selective initiator-posting facility is implemented in VS1. In MFT, when a job is placed in the input queue, each initiator that handles the job class to which the job is assigned is posted whether or not the partition the initiator schedules is busy. In VS1, when a job is queued, the selective posting routine determines whether all initiators that handle the job's class are busy. If so, all these initiators are posted. In this case, the first initiator that becomes available processes the job, as would occur in an MFT environment. However, if all the initiators that handle the job's class are not busy, the selective posting routine posts only one of these waiting initiators according to a priority scheme, as follows.

The partitions to which a specific job class is assigned are prioritized for processing jobs with that class according to the priority at which the class was specified within the partition during system generation or system operation (first job class specified has the highest priority, last job class specified has the lowest priority). If a job class is specified at the same priority within two or more partitions, the partition with the highest dispatching priority has the highest priority among the two or more partitions for processing jobs with that class assigned. When more than one partition with a given job class assigned is not busy when a job with that class is queued, the initiator for the available partition with the highest priority for processing jobs with that job class is posted.

For example, assume the following job class assignments have been made to four partitions:

Partition	Class assignment in high-to-low priority sequence within the partition
P0	BAC
P1	AB
P2	AC
P3	CBA

Given the specified assignments, partitions are prioritized for processing each job class as follows.

Job class	High-to-low priority for processing the class
A	P1, P2, P0, P3
B	P0, P1, P3
C	P3, P2, P0

Assuming the priorities above, if a job with class C is queued and only partition P0 is busy, only the initiator in partition P3 is posted and the job is processed in partition P3. In an MFT environment, the initiators for partitions P0, P2, and P3 would be posted and the job would be processed in partition P2.

The selective posting discussion above also applies to the posting of JES writers. That is, when a SYSOUT data set is queued, only one of the writers that handles its SYSOUT class is posted when one or more of these writers are not busy. When two or more writers have the same priority for processing a SYSOUT class, the first writer in the chain of these writers that is maintained by the job list manager is posted.

Interpreter

The interpreter is pageable and a large portion of it is reentrant. It operates as a subroutine of the initiator. The interpreter is invoked at the initiation of each job. The interpreter reads procedures directly from SYS1.PROCLIB (via data management) and interfaces with JECS spool management to read all the JCL and any PROC spool data sets that are associated with the job to be scheduled. It interprets all the job control for the job, constructs the required scheduler control blocks, and writes them in the appropriate SWADS or SWA (using SWADS management) for use by the other job scheduler components.

Interpreter messages are placed in system message spool data sets in VS1 instead of in the SYS1.SYSJOBQE data set as in MFT. Commands are sent to the master scheduler for processing when they are encountered. Jobs that were being interpreted when abnormal system termination occurred do not have to be resubmitted during the warm start procedure, as they do in MFT.

The interpreter accepts all the job control statements supported in MFT. UNIT and SPACE parameters on a SYSOUT DD statement are ignored as they are no longer required. New DD statement parameters for VSAM (discussed in Section 90:30) are added to the job control language as well as the following new parameters:

- ADDRSPC=VIRT or REAL and the REGION parameter on JOB and EXEC statements (discussed in Section 90:10)

- TYPRUN=SCAN on a JOB statement to indicate that the job control for the job is to be analyzed for errors but that the job is not to be executed

- COPIES=nnn on SYSOUT DD statements to request multiple copies of system output data sets

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

- HOLD parameter on SYSOUT DD statements. When HOLD=YES is specified on the DD statement for a SYSOUT data set, the data set is placed in held status in a SYSOUT queue in the resident job list and will not be written by an output writer until the operator releases it by issuing a RELEASE or ROUTE command. Since the operator is not notified by job management when a SYSOUT data set is placed in held status, the operator must be informed by the user. The notification can be accomplished by placing a SEND command in the input stream that sends a message to the operator. ROUTE and SEND are new commands that are provided by RES support (see "Remote Entry Services" later in this subsection).

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

- DSID and MSVGP parameters on DD statements. The DSID parameter is used to assign a data set identification to the SYSIN and SYSOUT data sets that are contained on 3540 diskette devices. The MSVGP parameter is used to assign an identification to a group of mass storage volumes that are contained in a 3850 Mass Storage System.

- CHKPT=EOV on DD statements to cause a checkpoint to be taken by VS1 at end of volume for multivolume QSAM or BSAM data sets

- PROFILE and MPROFILE parameters on JOB statements and the PROFILE subparameter on SYSOUT DD statements to supply installation-specified selection parameters for work classes and priority, as discussed below

- CHARS, FLASH, and BURST parameters on DD statements with or without SYSOUT specified also. These parameters apply only to the 3800 Printing Subsystem. CHARS can specify the names of up to four character arrangement tables that reside in SYS1.IMAGELIB and that are to be used in printing the data set. FLASH specifies the forms overlay frame to be inserted in the 3800 and the number of copies of the data set on which the overlay is to be flashed. BURST indicates whether the printed output is to be burst or left in continuous form.

- CHARS=DUMP and FCB=STD3 parameters on a SYSABEND or SYSUDUMP DD statement. These two parameters are supported only for the 3800 Printing Subsystem and provide the ability to take a high-density ABEND dump for a problem program. These two options are not available for SNAP dumps and system tasks.

  The CHARS=DUMP (condensed line) option causes each line of the dump to contain 64 bytes of storage data plus the EBCDIC translation instead of 32 bytes plus the translation. The FCB=STD3 (condensed page) option causes each page to have 80 lines (8 lines per inch) instead of 55 lines (6 lines per inch). The options can be specified together or singly.

- MODIFY parameter on DD statements with or without SYSOUT also specified. This parameter is supported only for the 3800 Printing Subsystem. It specifies the name of a copy modification module and a character arrangement table for the 3800 that reside in SYS1.IMAGELIB.

- COMPACT parameter on the SYSOUT DD statement for a 3790 Communication System supported by RES. This parameter specifies the compaction table to be used.

In VS1 and MFT, a specific job input class and priority, message class, and SYSOUT data set class can be assigned by the user via the CLASS, PRIORITY, MSGCLASS, and SYSOUT job control parameters, respectively, in JOB and SYSOUT DD statements. Class, in effect, represents a set of processing characteristics and relates work to be processed to a specific initiator or output writer. In VS1, the standard installation-specified selection parameters (ISSP) function can also be used to assign classes and priority to jobs.

The ISSP function enables the user to specify processing characteristics in JOB and SYSOUT DD statements instead of specific classes and priorities. The parameters used to describe processing characteristics are installation-defined. When ISSP is used by a job, the scheduler assigns specific classes and priorities to the job, using a set of installation-defined tables that match installation-defined processing characteristic parameters to specific classes and priorities.

Both standard job control parameters for classes and priority (CLASS, PRIORITY, MSGCLASS, and SYSOUT) and parameters defined by the ISSP function can be used in the job control statements for a job (a specific SYSOUT class specified in some SYSOUT DD statements in the job and ISSP parameters for class specified in other SYSOUT DD statements, for example). Input jobs, message data sets, and SYSOUT data sets are queued in the same way whether standard job control or ISSP parameters are used to assign class and priority.

In order for the ISSP function to be used, a set of exit definition macros must be prepared by the installation, assembled, and link-edited into the JES portion of the VS1 control program (IEFJES load module). The exit definition macros can be assembled during or after system generation. In order to assemble them other than at system generation time, the SYS1.ISPMAC data set, which contains the macros required for the assembly, must be provided. This data set can be created during system generation.

Each exit definition macro defines either input job classes and priorities (JOBCLASS definition) or system messages and job output SYSOUT data set classes (OUTCLASS definition). An exit definition macro defines the following:

- A set of keywords, their acceptable values, and optionally, a default value. These keywords represent processing characteristics and can be used either in JOB statements (if this is a JOBCLASS definition) or JOB and DD statements (if this is an OUTCLASS definition) in place of the standard class and priority parameters.

- A scheduling profile. The statements in a scheduling profile assign job classes (A to Z, 0 to 9) if this is a JOBCLASS definition, or message and SYSOUT classes (A to Z, 0 to 9) if this is an OUTCLASS definition, to the processing characteristics that are specified, using only the keywords defined in this exit definition macro.

- Job priority assignments, if this is a JOBCLASS definition. This portion of the exit definition relates job priorities (0 to 13) to the processing characteristics that are specified, using only the keywords defined in this exit definition macro.

The PROFILE and MPROFILE parameters are added to the JOB and DD job control statements in support of the ISSP function as shown below:

```
// JOB PROFILE="job profile string",MPROFILE="message profile string"

// DD SYSOUT=(...,PROFILE="SYSOUT profile string")
```

The keywords and values that can be specified in the profile string of a PROFILE parameter on a JOB statement are only those that have been defined in a JOBCLASS exit definition. Similarly, the keywords and values that can be specified in the profile string of an MPROFILE parameter or a PROFILE subparameter on a SYSOUT DD statement are only those that have been defined in an OUTCLASS exit definition. If the PROFILE parameter is specified on a JOB statement, it overrides the

CLASS and PRIORITY parameters if they are specified also. Similarly, an MPROFILE parameter on a JOB statement overrides an MSGCLASS parameter if one is present.

When the interpreter encounters an ISSP keyword on a JOB or SYSOUT DD statement, the tables generated by the assembly of exit definition macros are inspected to select the appropriate class and/or priority for assignment. The class or priority assigned to a given set of processing characteristics can be changed via modification and reassembly of the appropriate exit definition macro. As long as the same keywords are used for processing characteristics, the new class or priority is then automatically assigned to jobs with these characteristics without the necessity of changing the job control statements for these jobs.

The DISPLAY command is expanded to enable the operator to display the following when the P parameter is specified:

- The processing characteristics defined for a specific input or output class (IN=class and OUT=class parameters). The processing characteristics contained in the scheduling profile of the JOBCLASS or OUTCLASS exit definition that contains the specified class are displayed
- The class or classes assigned to the set of input or output processing characteristics that are specified in the DISPLAY P command
- All the classes that contain the processing characteristic(s) specified in the DISPLAY P command

Jobs whose job control statements contain ISSP keywords can be executed under control of a VS1 system that does not contain the specific assembled exit definition macros required by the jobs. The first time a job with ISSP keywords is encountered in the job stream, the operator is notified of the absence of the required JOBCLASS or OUTCLASS exit definition table and must specify CANCEL or IGNORE as a response to the message. If CANCEL is specified, this job and all successive jobs that need the JOBCLASS or OUTCLASS table specified in the message are rejected without being executed. This process continues until the next IPL.

If IGNORE is specified, jobs with ISSP keywords will be executed as follows. When a CLASS, MSGCLASS, or PRTY parameter is present on a JOB statement in addition to ISSP keywords, the value specified in this parameter is used in place of the PROFILE or MPROFILE specification. If a CLASS, MSGCLASS, or PRTY parameter is not present, the standard system defaults for these parameters are used in place of the PROFILE and MPROFILE specifications. When a SYSOUT class and a PROFILE keyword are specified in the SYSOUT parameter on a DD statement, the specified SYSOUT class is used instead of the PROFILE specification. If a SYSOUT class is not present, the default for the system message class is used instead of the PROFILE specification.

Allocation

The allocation routine in VS1 operates as a subroutine of the initiator to allocate and deallocate I/O devices to job steps, issue mount messages to the operator, etc., as in MFT. The VS1 allocation routine differs from the MFT allocation routine in that it supports dedicated work data sets (supported in MVT but not in MFT) and a new I/O device allocation algorithm.

The dedicated work data sets facility enables a job step to use disk data sets that are assigned to the initiator that schedules their execution. Job scheduling time is reduced by the elimination of temporary disk data set allocation and deallocation processing. The new I/O device allocation algorithm is designed to minimize contention among I/O devices by better balancing channel loads.

The channel load-balancing algorithm for nonspecific disk device requests that is used by the MFT allocation routine is replaced in VS1 by a new I/O load-balancing algorithm unless the latter is specifically excluded by the user during system generation. In MFT, 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 VS1, a different algorithm for determining the activity on a direct access device is used. This I/O load-balancing algorithm is called by the allocation routine to allocate devices for new 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 direct access device requests when this load-balancing algorithm is used, since the 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 MFT.)

The utilization of a tape or a direct access device is determined in VS1 by counting the number of I/O requests (EXCP macros and PCI interruptions) for the device in a given interval. Data from the previous interval is also maintained. 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 nonspecific direct access 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.

In order to make the most effective use of this 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 (most active before less active).
- Nonspecific volume requests should be made whenever possible.

Terminator

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

Direct SYSOUT (DSO) Writers

The same functions are supported by DSO writers in VS1 as in MFT, except that a VS1 DSO writer can handle up to 15 job classes instead of only 8, as in MFT.

System Management Facilities (SMF)

SMF for VS1 provides all the same functions it does in MFT and is expanded to include additional accounting data and exits. SMF records (SYS1.MANX and SYS1.MANY data sets) can be written only on disk in VS1. They cannot be written on tape as in MFT. The SYS1.MANX and SYS1.MANY data sets must be cataloged in VS1. The SMF keywords PRM, ALT, and MDL are not accepted in VS1. The SID parameter is expanded in VS1 to include the MDL parameter value.

The SMF option desired is chosen at system generation from the following:

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

- BASIC - User-written accounting routines are to be provided. These routines can be newly written or those currently being used with MFT. The latter need not be modified for operation in VS1. The additional JES accounting information is made available as are the user exits IEFUSO, IEFUJP (not provided in MFT), and IEFACTRT.

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

The following are the major differences between SMF in VS1 and MFT:

- Three exits are added to SMF in VS1. The IEFUJP exit is taken at job purge time and can be used to include additional statistics in SMF output records. The IEFUIV exit is taken each time a JOB statement is read from the input stream. The return from this exit can indicate whether the job is to be processed. The IEFU83 exit is taken each time an SMF record is to be written to the SMF data set. The return from this exit can indicate whether the record is to be written.

- The user routine executed at exits IEFUJI, IEFUSI, and IEFACTRT in VS1 can write to installation-defined data sets. This is not permitted in MFT.

- Additional record types (62, 63, 64, 67, 68, and 69) are added for VSAM accounting data.

- Record type 22 is added to provide configuration data about the 3850 Mass Storage System.

- Record type 50 is added to contain statistics about VTAM that can be used for tuning purposes.

- Record type 20 is changed from a data set activity record to a job accounting record.

- Record type 6 (output writer accounting record) is expanded to include accounting information for the 3800 Printing Subsystem.

- Additional record types (43, 44, 45, 47, 48, and 49) are added to contain RTAM accounting data for both binary synchronous and synchronous data link control communications.

The SMF record types and formats produced by SMF routines in VS1 are compatible with those produced in MFT, for the most part. Additional accounting information is supplied, minor changes to existing fields have been made, and certain fields have a different meaning in VS1. For example, in the job step termination record the storage-requested and storage-used fields reflect the virtual storage used. If the job step ran in nonpaged mode, these fields also reflect the real storage used. SMF records that are modified in VS1 are the IPL (type 0), system statistics (type 1), step termination (type 4), job termination (type 5), output writer (type 6), end of day (type 12), and dynamic storage configuration (type 13).

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

- Time required to read the job (elapsed time calculated from JES reader start and stop times)

- Number of cards read

- Job priority and job class

- Elapsed time for SYSOUT print processing and number of lines printed

- Elapsed time for SYSOUT punch processing and number of cards punched

- Elapsed time for SYSOUT tape processing and number of SYSOUT records written to tape

The page supervisor provides the following data to SMF:

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

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

- Number of reclaimed pages for the entire system

The initialization of SMF during system initialization involves the processing of an SMFxxxxx member or the IBM-supplied SMFPRM00 member in SYS1.PARMLIB, as covered in the automated system initialization discussion (see Section 90:10). These members replace the SMFDEFLT member used in MFT.

Note that in MFT the job step wait time limit (number of consecutive minutes all the tasks of a job step can remain in an SVC wait state) can be specified by the user via the JWT parameter of SMF (from 1 to 999 minutes) only if SMF is included in the generated system. In MFT systems without SMF, the wait time limit is always 30 minutes.

In VS1, the wait time limit (from 1 to 932 minutes) can be specified at system generation via the WAIT parameter of the CTRLPROG macro for systems without SMF. If the WAIT parameter is not specified for a VS1 system without SMF, the default wait time is 30 minutes. The wait time limit for VS1 systems with SMF is specified via the JWT parameter. In VS1, as in MFT, the wait time limit can be made ineffective for a job step by specifying TIME=1440 in the EXEC statement for the step.

General Flow of Job Scheduling

Figure 90.20.2 illustrates JES and job scheduling flow in VS1. Active JES readers read input streams. Data read (job control statements, procedures, input data, commands) is passed to spool management without interpretation of job control statements. Spool management requests the allocation of spool space (one logical cylinder is allocated per spool data set) and blocks input stream data, which is written in spool data sets by the DASD work area manager.

Whenever a complete job has been read, the JES reader creates a disk entry record (DER), job management record (JMR), and route table for the job and passes them to the job list manager. Space for these control blocks, the job accounting table, and the SYSOUT class directory is obtained from SYS1.SYSJOBQE by the job list manager, which then writes the control blocks created by the JES reader in SYS1.SYSJOBQE. The job list manager queues the job in an input queue in the resident job list by priority within job class and marks it held if appropriate. The JES reader continues reading its input stream.

Started initiators are selectively posted by the job list manager when a job is queued that has a job class they are assigned to handle. Once a job for an initiator is dequeued from the resident job list and the DER for the job has been read, the initiator passes control to the interpreter. The interpreter obtains the job control statements for the job from the appropriate JCL spool data set(s) via spool management and, if necessary, from SYS1.PROCLIB. Job control statements are interpreted and scheduler control blocks are built for the job. The control blocks are placed (by SWADS management) in the SWADS or SWA associated with the initiator.

Spool space for SYSOUT spool data sets is allocated. One logical cylinder is allocated to each SYSOUT spool data set at this time. Commands within the job are routed to the master scheduler and interpreter messages are placed in a system message spool data set. Control is given to the allocation routine, which attempts to allocate I/O devices to the first step. Allocation messages are written in a system message spool data set and the job step is begun.

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

Initiation of successive steps of the job continues until end of job occurs. The job terminator requests that the job list manager enqueue the SYSOUT spool data sets for the job in output queues in the resident job list by priority within SYSOUT class. The spool space allocated for SYSIN spool data sets for the job is released. The initiator begins processing the next job for which it has been posted, if any.

Started JES writers interface with job list management to select a SYSOUT spool data set with a class they are assigned to handle from the resident job list. A JES writer obtains SYSOUT logical records via spool management. When all the SYSOUT spool data sets for a job are processed, the job list manager purges the job from the resident job list and SYS1.SYSJOBQE, and spool management frees the SYSOUT spool space allocated to the job.

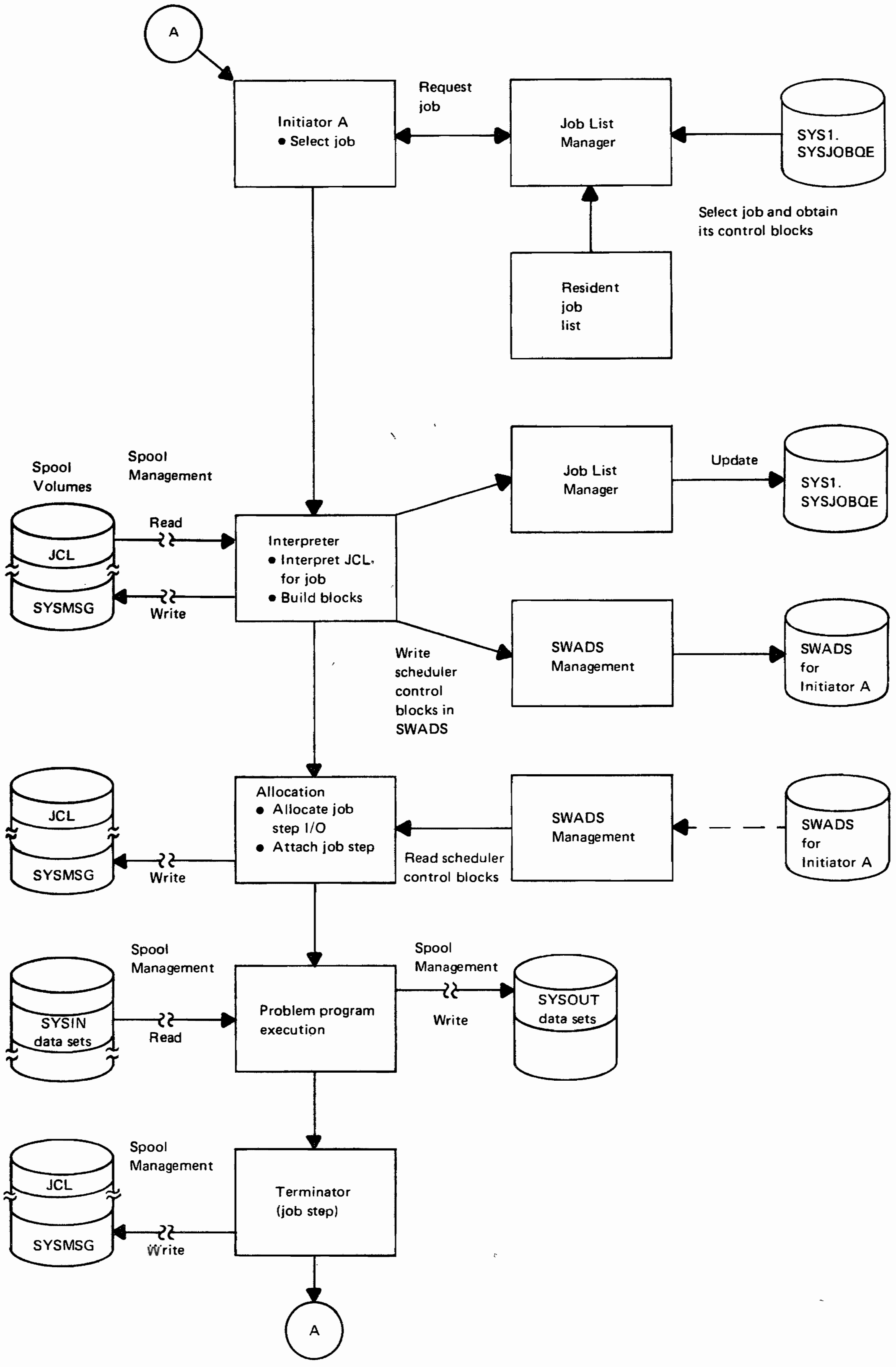


Figure 90.20.2. General flow of JES and job scheduling in OS/VS1

REMOTE ENTRY SERVICES

Remote entry services (RES) is a fully integrated functional extension of JES. Using binary synchronous communications (BSC) and synchronous data link control (SDLC) communications, RES enables remote users to submit jobs to a central computing system via a workstation (terminal) and to receive the output from these jobs.

RES presents remotely submitted jobs to JES readers, which in turn place the jobs in the input work queue in the resident job list. JES writers present output from completed remotely submitted jobs to RES, which transmits the output to remote workstations. Hence, the unit record devices at remote workstations (readers, printers, punches) are logically operated by JES as if they were part of the central system.

The same reader and writer facilities are provided for remote unit record devices as for local unit record devices except that user-written writer routines are not supported for remote devices.

The 3770 Data Communication System (nonprogrammable models) and 3790 Data Communication System are supported as systems network architecture (SNA) terminals using synchronous data link control communications. SNA terminals controlled by RES can be attached to the central system remotely via 3704/3705 Communications Controllers operating in NCP mode or locally via a channel.

The following are supported by RES as non-SNA remote workstations using binary synchronous communications:

- 1130 System
- System/3
- System/360 Models 20 to 195
- System/370 Models 115 to 168 (operating in BC mode only)
- System/370 Model 195
- 2770 Data Communications System
- 2780 Data Transmission Terminal
- 2922 Programmable Terminal
- 3741 Data Station Model 2 and 3741 Programmable Work Station Model 4
- 3780 Data Communications Terminal

The maximum number of remote workstations supported is 200. However, a practical maximum number of remote workstations is from 55 to 65. This maximum is limited by the CPU size and workstation configurations.

USASCII code is supported only for the 2770, 2780, and 3780. Terminals can be attached to point-to-point leased and point-to-point dial-up lines. Multidropped leased lines and dial-up lines are not supported. Lines can be two-wire half-duplex or four-wire half-duplex. Full-duplex lines are not supported. A 2701, 2703, or 3704/3705 is required in the central system configuration. The integrated communications adapter (ICA) instead of a 2701 can be used on a Model 135 or 138.

Remote workstations using binary synchronous communication that include a CPU and the 2922 Programmable Terminal are called BSC CPU workstations. A remote workstation using BSC that does not contain a CPU (a 2770, 2780, 3741, or 3780) is called a BSC non-CPU workstation. BSC CPU workstations operate under the control of standalone workstation programs. Work station programs for System/360 and System/370 models are distributed with the VS1 control program. Work station programs for System/3 and the 1130 system must be ordered separately.

Generation procedures must be performed to tailor a workstation program to the configuration of each CPU workstation that is to be supported by RES. The standalone programs provided for RES remote workstations are the same ones that are provided for HASP II RJE remote workstations.

RES enables a user at a remote workstation to:

- Transmit jobs to a central system for processing using standard VS1 job control statements and operator commands. Two additional job control parameters and five additional operator commands are provided also.
- Route the output from each completed job to the central system or a specific remote workstation, which need not be the one from which it was submitted. Output destination can be indicated in job control statements or via an RES command.
- Display the status of his workstation and the status of his submitted jobs
- Send messages to, and receive messages from, other remote workstation users and the central system operator

RES functions are provided by the following components of VS1:

- The remote terminal access method (RTAM), which handles all data transmission to and from remote workstations
- The SYS1.UADS and SYS1.BRODCAST data sets, which define the attributes of RES workstation users and contain workstation messages, respectively, and the SYS1.RMTMAC and SYS1.RMTOBJ data sets, which are required for RTAM generation
- A standard ACCOUNT Facility program (IKJRBBMP), which is used to create and maintain the SYS1.UADS and SYS1.BRODCAST data sets
- The RES commands LOGON, LOGOFF, ROUTE, SEND, LISTBC, which enable remote users and the central operator to control remote job processing and to communicate with each other. WTO and WTOR macros are extended to support remote users via the addition of the QID operand. (In order to issue a WTO/WTOR macro with the QID operand, a task must have storage protect key zero assigned, operate in supervisor state, or be authorized via the authorized program facility.)
- JES readers and JES writers that interface with RTAM and support remote input and output streams. (Column binary is not supported.)
- An interface with VTAM to enable systems network architecture workstations to communicate with the central system via synchronous data link control communications lines.

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

RTAM

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

RTAM is the only access method used by RES for non-SNA workstations. RTAM executes as a system task in a pseudo partition in the pageable supervisor area in VS1. It supports only the RES terminal network and except for VTAM does not interface with any other terminal networks (TCAM or BTAM, for example) that may be included in the system configuration. RTAM communicates with SNA workstations using the normal application interface to VTAM. SNA workstations must be defined to both RTAM and VTAM.

In order to handle binary synchronous terminals, RTAM is started on a communication line basis and handles all teleprocessing functions for the workstations attached to binary synchronous lines. To handle SNA terminals, RTAM functions on a workstation basis while VTAM handles all teleprocessing functions (communication line control, SNA terminal logons, and resolution of transmission code dependencies) for the SNA terminals attached to lines operating in synchronous data link control mode.

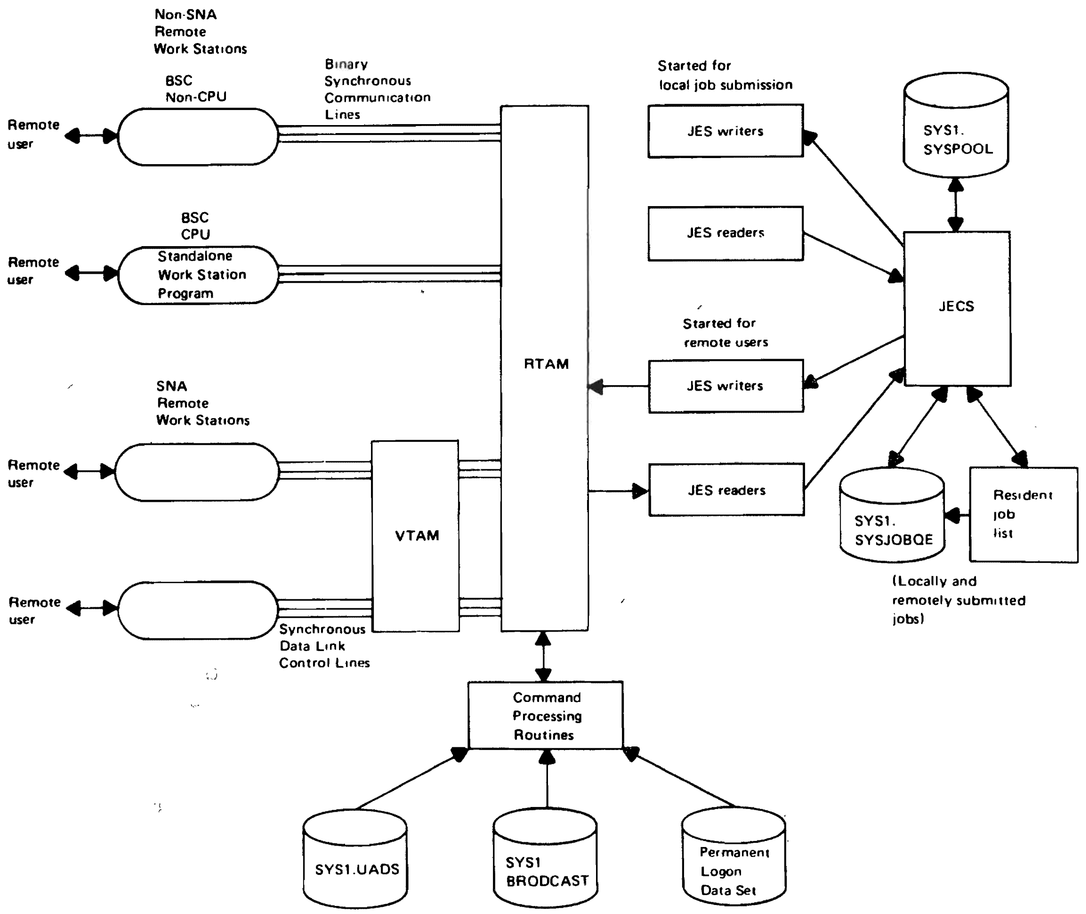


Figure 90.20.3. RES interface with JES

RTAM directly controls BSC non-CPU workstations and interfaces with BSC CPU workstations using a MULTI-LEAVING technique that is not supported by OS RJE. MULTI-LEAVING is a more efficient way to transmit data between two computers using binary synchronous communication facilities because it reduces the number of line turnarounds required and enables more data to be transmitted before line turnaround occurs. MULTI-LEAVING permits data from more than one unit record device in a workstation configuration to be sent during a single transmission, and allows transmission acknowledgment to be sent together with data in the same record.

For example, a workstation can transmit a record containing text from two or more input stream unit record devices to the central system. The central system can respond by sending a record that acknowledges receipt of the text and that includes text for one or more output stream unit record devices in the workstation configuration. This capability eliminates an individual transmission for each text record and each acknowledgment.

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

RTAM interfaces with GTF to provide a trace facility that can be used as a debugging aid. When this trace facility is active, RTAM calls GTF at channel-end time for each line to produce a trace record of the line DCT (device control table) and another trace record consisting of the first 256 bytes of data in the buffer associated with the I/O operation. RTAM tracing is activated by starting GTF, if it is not already operational, and specifying the TRACE=USR and MODE=EXT options.

<u>RES Data Sets and Logon Procedures</u>

The user attributes data set, SYS1.UADS, is a partitioned data set that is required during system operation when RES is used. It is used to control remote user access to the central system via RES. SYS1.UADS contains one or more members for each user that is authorized to access the central system. Members contain control information and identifying attributes for each RES user, such as user identification, one or more passwords (optionally), the name of one or more logon procedures to be used during a logon (optionally), and a job priority limit for the jobs submitted by this remote user.

The logon procedures for RES users can contain one or more commands that are automatically issued during the logon procedure. RES commands and VS1 commands that a remote user can issue from a remote workstation (see Table 90.20.1) can be placed in a logon procedure. The logon procedures that are to be used by remote users and the central operator must be user-defined and placed in SYS1.PROCLIB. An installation can establish logon procedures that are to be used only by authorized remote users.

The scheduling priority of jobs submitted by remote users can be limited by using the PRIORITY parameter in the members of SYS1.UADS. When a job from a remote user is received, the job priority limit in the SYS1.UADS member for the user is compared to the job priority specified in the JOB statement for the job. If the job priority in SYS1.UADS is less than that specified in the JOB statement, it is assigned to the job instead of the priority in the JOB statement. However, the priority assigned to the SYSOUT data sets from a remote job is always that specified on the JOB statement.

During system initialization, a queue identification (QID) table is built in pageable SQA in virtual storage. When a cold start is performed, all the user attribute data from the SYS1.UADS data set is placed in the QID table and a checkpoint of the table is written in SYS1.SYSPOOL. When a warm start is performed, the QID table is read in from SYS1.SYSPOOL.

Whenever a remote user attempts to log on, the QID table is inspected. If there is no entry for the user or an invalid request is made (user specifies a logon procedure he is not authorized to use, for example), the user is not permitted to log on. The SYS1.UADS data set must be created and maintained using the IBM-supplied ACCOUNT Facility program, which provides the capability of adding or deleting users, changing fields in existing user members, listing user members, and listing all the user identifications in the SYS1.UADS data set.

The SYS1.BRODCAST data set is also required when RES is used during system operation. It is created and maintained using the ACCOUNT Facility program. It is used to hold messages that have been issued by remote users and the central operator using the SEND command but that have not yet been sent.

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

RES Commands

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

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

CANCEL	LOG	RELEASE	START
DISPLAY	MODIFY	REPLY	STARTF
HOLD	MONITOR	RESET	STOP,STOPMN
		SETPRT	WRITER

A user at a remote workstation issues a LOGON command to establish connection with the central system, that is, begin a workstation session. The parameters of the LOGON command for SNA and non-SNA terminals are slightly different. During the logon procedure, user identification, terminal identification, and password (if any) are verified using information contained in the QID table. Any JES readers and writers identified in the logon procedure specified in the LOGON command or SYS1.UADS data set are started. If they were requested in the LOGON command, mail and notices contained in SYS1.BRODCAST are retrieved and sent to the user during logon processing.

If the LOGON command is valid, the user can begin transmitting jobs and messages. If the LOGON command is invalid, a message is sent to the central operator, since two-way communication between the workstation and RES is not possible until a logon is successfully performed. The central operator can be contacted by a remote user if the reason for the invalid LOGON is not apparent.

JES readers and writers for any remote user can also be started (using START or STARTF) and stopped by the central operator and a remote user can start and stop readers and writers for his own workstation only. Note that reader and writer procedures for remote workstations must be defined by the user. None are IBM-supplied.

The LOGOFF command is issued by a remote user to indicate communication with the central system is finished, that is, to indicate the workstation session is finished. This command can be entered via an input stream between two jobs. The central operator can also issue the LOGOFF command to terminate operations at a remote workstation. Logoff processing includes the stopping of all readers and writers that were started for the remote user. For an SNA workstation, the LOGOFF command can be submitted to VTAM instead of to RTAM, if desired. RTAM is then notified and takes the required action.

The central operator can issue the LOGON CENTRAL command with or without a password specified. The advantage of this command is that it can also specify a procedure that can contain all the commands required to start the readers and writers required. This command need not be specified in order for the central operator to access RTAM.

The central operator, if logged on, can issue a LOGOFF CENTRAL command to stop all the tasks he started. Once the central operator logs off, no other users can log on or start readers or writers.

If the LOGOFF command does not specify the SLOW parameter, readers and writers for the remote workstation are stopped when they complete processing of the current data set. When SLOW is specified, each reader and writer is stopped when no more work is available for it to process.

When an intervention-required (line down) condition occurs on an active binary synchronous line, the action taken depends on the AUTOLOG option specified for the line at RTAM generation. If AUTOLOG=YES was specified, the line is restarted and the remote user of the line is automatically logged off, which ends the workstation session. This option can be specified to force another logon for security reasons. If AUTOLOG=NO was specified (the default), the remote user is not automatically logged off. The workstation session for an SNA terminal ends automatically if the SNA terminal becomes disconnected from VTAM.

The ROUTE command is provided to allow a remote user to control the destination of the SYSOUT data sets associated with his job. The ROUTE command is effective for a job only if it is issued after the job has been submitted and before the SYSOUT data sets to which it refers have been selected for processing. All keywords for the ROUTE command have a single-character abbreviation.

A user can issue the ROUTE command to cause all his SYSOUT data sets, all the SYSOUT data sets from only one of his jobs, or the SYSOUT data sets with the specified SYSOUT classes from one or all of his jobs to be routed to another remote user or the central operator. The routed SYSOUT data sets are placed in the SYSOUT queue(s) of the specified user based on the SYSOUT class they are currently assigned (as specified in SYSOUT DD statements) unless the CLASS parameter is included in the ROUTE command to change the SYSOUT class. By specifying HOLD=YES in the ROUTE command, a user can cause his SYSOUT data sets to be placed in held status in the output queue to which they are routed.

The ROUTE command can also be issued to override the destination indicated in the DEST parameter on SYSOUT DD statements for submitted jobs or to release the SYSOUT data sets for a job for which the HOLD command was specified.

A remote user can route his output only to another remote RES user whose routing mask is equal to or greater than his own. Routing mask is a value in the range of 0 to 255 that is specified in each user identification member in the SYS1.UADS data set. When a remote user has a routing mask between 0 and 254, his output can be routed to another user by himself and the central operator. When a remote user has a routing mask of 255, only the remote user himself can route his own output.

Messages can be transmitted among remote users and between the operator and remote users via the SEND command. A given message can be sent to one or more (up to 20 maximum) specific remote users or to all of them. The central operator can request that a message be saved in the SYS1.BRODCAST data set instead of being transmitted immediately. The SEND command can also be used to delete previously saved messages from SYS1.BRODCAST. The LISTBC command enables a remote user to list all notices and the mail that belongs only to him. The central operator can list mail belonging to any user as well as all notices.

The DEST and HOLD parameters can be placed on the DD statements for SYSOUT data sets that are created by remotely submitted jobs. A SYSOUT data set for a remote job is returned to the remote user that submitted the job unless the DEST parameter is placed on the SYSOUT DD statement for the output data set. The DEST parameter indicates the location to which the SYSOUT data set is to be sent.

The HOLD=YES parameter can be specified on a SYSOUT DD statement to indicate the SYSOUT data set is to be placed in held status in the SYSOUT queue for the remote user. A held SYSOUT data set is not processed until a ROUTE or RELEASE command is issued. The central operator issues the ROUTE or RELEASE command unless the DEST parameter was also specified on the SYSOUT DD statement. In this case, the ROUTE or RELEASE command must be issued from the location indicated in the DEST parameter.

The central operator or the user at the location specified in a DEST parameter should be notified when HOLD=YES is specified for a SYSOUT data set since this is not automatically done by the system. This notification can be accomplished via a SEND command.

The LISTBC, LOGON CENTRAL, LOGOFF CENTRAL, ROUTE, and SEND commands can be issued by the central operator of a VS1 system with RTAM support included even when RTAM has not been started.

RES Initialization and Termination

When RTAM is included in a VS1 system, during system initialization an RTAM system task partition is allocated adjacent to the JES routines area in the pageable supervisor area. The RTAM module is loaded into the RTAM partition and initialized. The RTAM parameters specified during generation of the RTAM module are used unless they are overridden by parameters in a specified RESxxxxx or the RESPARMS member in SYS1.PARMLIB.

In order to initiate RES processing, the central operator must enter a START command for the RTAM procedure, which is a user-defined procedure. This causes RTAM to be activated and the lines indicated in the RTAM procedure are enabled. If required to handle SNA workstations, the interface with VTAM is also established. The operator is notified if VTAM is not yet active. The central operator can issue the MODIFY command any time thereafter to enable additional lines, disable enabled lines, reenable lines, or activate/deactivate the VTAM interface.

During the RTAM initialization procedure, workstations that are identified in the permanent logon data set are automatically logged on. The permanent logon data set is a user-created sequential or partitioned data set that contains a LOGON command for each workstation that is to be permanently logged on. This optional permanent logon facility enables BSC non-CPU terminals (2770, 2780, and 3780) and SNA terminals that are connected to dial or leased lines to be automatically logged on whenever RTAM is started. A permanently logged-on workstation remains logged on until the user issues a LOGOFF command or the workstation is logged off as a result of a disastrous error, STOP command, or RESTART command.

Once RTAM is started, any terminals attached to the enabled lines that are not part of the permanent logon group must be connected to the system via execution of the logon procedure. While remote RES users cannot log on until RTAM is started, the central operator can log on from his console before RTAM is started by using the CENTRAL user identification, which must be user-defined and placed in the SYS1.UADS data set. Once operations at a workstation that is connected to a switched line are finished, the user can log off and, thereby, release the line and make it available for use by another remote workstation connected to it.

The MODIFY command can also be used to alter certain characteristics of a remote terminal that were established during RTAM generation. MODIFY can be issued to determine whether or not the following are used: compression, transparency, compaction, and horizontal format features. The changes specified take affect at the beginning of processing of the next data set.

The compression feature is supported for 2770, 3780, BSC CPU, and SNA workstations (not 2780 or 3741 workstations). This feature cannot be turned off for BSC CPU workstations. It also cannot be turned on for any SNA workstations if support of the feature is excluded during RTAM generation (SNACOMP=NO).

For 2770 and 3780 workstations, the compression/expansion and transparency features cannot be used simultaneously for the same job. Compression of data sent from a 2770 or 3780 is controlled by the remote operator using the compression/expansion switch on the workstation. Compression of data sent to a 2770 or 3780 is controlled by RTAM parameters. For BSC CPU workstations, compression and transparency can be active at the same time. Compression always occurs on data sent to, and received from, a BSC CPU workstation.

Compression is an optional feature for SNA workstations. If selected, compression of both successive blanks and successive duplicate characters occurs for data sent to, and received from, the SNA workstation. Compression and transparency can be used at the same time for SNA workstations.

Compaction is also optional for SNA workstations. It requires use of the compression feature as well. Compaction is a technique of compressing nonduplicate characters that uses a compaction table. IBM-supplied or user-defined compaction tables can be used.

Horizontal format control for printer data is supported for 2770, 2780, and 3780 workstations only. This feature provides limited blanks compression (a tab character replaces a trailing set of blanks in each group of ten data characters in each print line). Compression is more efficient than horizontal format control but the latter can be used at the same time as the transparency feature.

Table 90.20.2 summarizes the major characteristics of RES workstations.

Table 90.20.2. Characteristics of RES workstations

Characteristic	BSC CPU Work Stations	BSC Non-CPU Work Stations	SNA Work Stations
Line protocol	Binary Synchronous (MULTI-LEAVING technique is used for data transmission)	Binary Synchronous	System network architecture/ synchronous data link control
Runs under the immediate control of own work station programs	Yes. A separate generation of workstation program is required.	No. Controlled by hardware design, RTAM generation parameters, and remote operator.	May or may not
May have console for entering of operator commands and receipt of messages	Yes	No. Commands entered via a card reader and messages printed in between printing of data sets.	Yes
May be permanently logged on	No	Yes	Yes
Multiple readers, printers, and punches attached	Yes	Maximum one reader and one punch. Maximum one printer except for 2770 (up to three) and 3780 (up to two)	Multiple readers and/or printers may be attached.
Interrupt printing/punching of data to enter reader data	No	Printing can be interrupted	Printing and punching can be interrupted. Reading can be interrupted to send data from the console or another reader.
Data transmission code	EBCDIC only	EBDIC or ASCII	EBDIC or ASCII
Transparency feature	Yes, for transmitting and receiving	Yes, for transmitting and receiving	Yes, for transmitting and (required for) receiving
Compression	Yes (cannot be turned off)	Optional for 2770 and 3780 only	Optional
Compaction	No	No	Optional
Horizontal format control	No	Optional for 2770, 2780, and 3780 workstations only	No

Using the MODIFY command, a remote work station can be placed in attended or unattended mode. When a workstation is operating in unattended mode, WTOR messages for the work station are sent to the central operator, instead of the workstation, so that the central operator can issue a reply. WTO messages for the remote station continue to be sent to the station. In addition, for unattended mode RTAM does not attempt to recover from any error that requires operator intervention. Thus, if a binary synchronous line drops, the workstation is logged off and the line is restarted (without regard for the AUTOLOG specification for the line). When operator intervention is required for an SNA workstation, the affected output writer is abnormally terminated.

The central operator issues a STOP command to initiate the termination of RES operations. This command prevents the logon of any additional users, prevents the starting of any additional lines, stops each started line after the logged-on remote user logs off and processing of any input or output currently being handled by the user's readers and writers is completed, stops lines that are to be restarted because of an I/O error, and stops lines that were stopped via a MODIFY command. The actual termination of all RES processing occurs as soon as all lines have been stopped. To stop all lines and users, a specific line or user, all binary synchronous lines, or all SNA users immediately, the MODIFY command with the STOP operand specified should be issued.

Virtual Storage Requirements

The size of the RTAM partition in the pageable supervisor area varies depending on such factors as whether both BSC CPU and BSC non-CPU workstations are supported, whether SNA workstations are supported, the number of lines and remote unit record devices in the RES configuration, the number and size of the buffers in the RES buffer pool for all RES workstations, the number of WTO/WTOR buffers specified for each work station defined, and the amount of contingency space reserved for a WTOL/WTOR buffer pool for all workstations.

The RTAM task (executable code) requires approximately 40K bytes if only BSC CPU (MULTI-LEAVING) or only BSC non-CPU (non-MULTI-LEAVING) workstation support is present. The maximum RTAM module size is approximately 64K for support of BSC CPU, BSC non-CPU, and SNA workstations. When RTAM is started, 6K to 12K bytes of the RTAM task are long-term fixed until RTAM is stopped, as are the 120 bytes of virtual storage that are required for each line defined in the RES configuration. Certain other tables and control blocks are also long-term fixed for the entire time RTAM is in operation. The minimum RTAM partition size is approximately 60K bytes.

RES Advantages over RJE

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

- RES is fully integrated within VS1 and uses the normal system job scheduling facilities (such as JES readers and writers, scheduling routines, and the job list manager). Only the RTAM module is optional.

- RES is designed to operate in a paging environment and can provide better performance than RJE partly because of MULTI-LEAVING support.

- RES requires less real storage for its operation.
- RES SYSOUT classes are independent of the central SYSOUT classes and more flexible SYSOUT routing is provided by RES (a unique set of output queues is associated with each remote RES user that represents the SYSOUT data sets only for that remote user).
- The RES remote user command language is a compatible subset of the OS/VS1 central operator command language (the job entry control language of RJE is not). Jobs can be submitted locally or remotely using the same job control statements and commands.
- System/3, the 2922 Programmable Terminal, the 3770 Data Communication System, 3780 Data Communications Terminal, and 3790 Data Communication System are supported as remote work stations.

CONVERSATIONAL REMOTE JOB ENTRY

The facilities offered by CRJE are the same in MFT and VS1. CRJE can operate in paged mode in a minimum partition of 128K. However, a minimum of 60K of real storage is fixed by CRJE. Therefore, a system with more than 160K of real storage is required to operate CRJE. The 3330-series Model 11 and 3340 disk storage drives are not supported by CRJE.

90:25 TASK MANAGEMENT

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

VS1 provides the same checkpoint restart facilities as MFT and is extended to support facilities of VS1 that are not present in MFT, such as support of VSAM and the 3850 Mass Storage System. Note that checkpoint records are always 2K in size in VS1, and that MSGLEVEL=1 is no longer a required parameter on the JOB statement.

In addition, the VS1 checkpoint facility includes a system routine to take a checkpoint automatically when end of volume occurs for a multivolume QSAM or BSAM data set or concatenated QSAM or BSAM data sets. This capability is requested for a data set by specifying the CHKPT=EOV parameter on the DD statement for the data set. A SYSCKEOV DD statement must also be included in the job step to define the checkpoint data set in which end-of-volume checkpoint records are to be written.

This checkpoint capability eliminates the necessity of having user-written routines to take end-of-volume checkpoints. The volume serial numbers of the tape data sets that were mounted when each checkpoint was taken can be listed using the CHKLIST function of the VSAM Access Method Services Program (see "Virtual Storage Access Method" in Section 90:30).

An abnormal termination dump of virtual storage can be obtained for system tasks that can be started in a partition (GTF, for example, and JES readers and writers in the JES pseudo partition as already discussed) by including a SYSUDUMP or SYSABEND DD statement in the procedure for the system task. An SVC dump of all virtual storage is automatically written to the SYS1.DUMP device when a system task that does not execute in a partition abnormally terminates. The indicative dump in VS1 is expanded to include system and user completion codes, the interruption address, the contents of the floating-point registers, and all request blocks (RBs) from the active RB chain.

A user-written dump formatting routine that receives control during SNAP and ABDUMP processing can be included in a VS1 (but not MFT) system. This routine, which executes in supervisor state with protect key 0 in effect, must be located in SYS1.LINKLIB and its name (IEAAAD03) added to the resident access method list. The formatting routine can determine whether the dump is one that the routine is designed to handle and, if so, format certain data (such as control block information or user work areas). The output produced by the user-written formatting routine is in addition to the dump output from the system.

Other supervisor routines have been altered to provide major new functions, such as dynamic task dispatching, fetch protection, and improvements in system integrity and timing facility support. The following identifies the major functional differences between VS1 and MFT task management routines.

INTERRUPTION SUPERVISOR

Interruption handling is essentially the same in VS1 and MFT; however, additional interruptions are recognized (specifically, segment and page translation exception, translation specification exception, monitor call, program event recording, and SET SYSTEM MASK instruction interruptions), the SVC transient area is 2K instead of 1K, and the SVC table and its mapping table have been restructured to simplify SVC interruption handling processing that is required to locate an SVC routine.

The SPIE facility has been expanded to allow the user program to gain control after a segment translation exception (invalid bit is on in the addressed segment table entry), which is treated as an addressing exception. In addition, authorized routines can gain control after a page fault (page translation exception caused by invalid bit on in the page table entry for the referenced virtual storage page), which normally is handled by page management. A routine is authorized to receive control after a page fault if it operates in supervisor mode, has protect key 0 assigned, or is authorized via the authorized program facility.

An authorized routine gains control after both disabled and enabled page faults and the system lock (described under "Task Supervisor") is set off. Therefore, this capability of the SPIE facility should be used carefully.

The data presented to a user-written SPIE routine has the same format in VS1 as in MFT so that SPIE routines that operate in BC mode will operate in EC mode without modification.

MONITOR CALL instructions are contained in various portions of the control program in order to alert the control program to the occurrence of certain events. For example, IOS uses the hardware monitoring facility to collect statistics about paging operations that are presented to SMF and to monitor the I/O events requested via the generalized trace facility (GTF). When appropriate, GTF is given control after a monitor call interruption occurs. When program event recording is operative, the dynamic support system (DSS) is entered after a PER interruption. (GTF and DSS are discussed in Section 90:40.)

The interruption supervisor also recognizes an SSM (SET SYSTEM MASK instruction) special operation exception that occurs when an SSM instruction is executed. Control is given to a routine that analyzes the masking requests indicated, which are assumed to be in BC mode format. This routine then puts the system in the requested state. (The new supervisor lock, described below, is tested prior to alteration of the system mask, if necessary.)

A new supervisor macro, MODESET, is implemented that is designed to be used in VS1 in place of the SSM instruction. MODESET, which is restricted via the authorized program facility, can be used to request a system mask setting, storage protect key alteration, and the setting of problem program or supervisor state in the PSW. The MODESET macro can be issued only by a task that has protect key 0, operates in supervisor state, or is authorized via the authorized program facility.

TASK SUPERVISOR

VS1 supports the same task management macros as MFT. In VS1, the ATTACH macro also has a TASKLIB operand that can be used to specify a task library. A task library is available only to the task for which it was specified in the ATTACH macro and replaces the job or step library that would otherwise be available to the task. A program that is to use a task library must specify a DCB and issue an OPEN macro for the task library.

Dynamic Dispatching

A significant new optional feature of the VS1 task supervisor is dynamic dispatching, which is also sometimes referred to as heuristic dispatching. The dynamic dispatching and time-slicing options are mutually exclusive within the same VS1 control program.

The dynamic dispatcher dispatches a user-specified 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 larger multiprogramming environments (Model 155 II and up) by more readily adapting task dispatching to the changing CPU and I/O usage requirements of a group of programs.

When dynamic dispatching is included in a VS1 system, at system generation one or more partitions are designated as belonging to the dynamic dispatching group. The partitions specified can be changed during system initialization or the dynamic dispatching capability can be canceled for the IPL. When two or more partitions are specified, they must be contiguous.

All tasks that execute in the dynamic dispatching partitions (including those created via a user-issued ATTACH macro) are dispatched according to dynamic dispatching rules. Tasks that execute in all other partitions are dispatched according to normal partition priority rules. A problem program task cannot issue a CHAP macro to cause itself to be moved into or out of a dynamic dispatching partition. Such CHAP macros are treated as NOPs. This restriction does not apply to system tasks.

For dispatching purposes, the tasks in the dynamic dispatching partitions are treated as a logical subset of all the existing (system and user) tasks in the system. As shown in Figure 90.25.1, tasks are logically connected in high-to-low dispatching priority sequence, with dynamic dispatching tasks logically divided into an I/O-oriented subgroup and a CPU-oriented subgroup. The I/O subgroup is positioned within the dynamic dispatching group to have higher priority than the CPU subgroup. When the dispatcher is ready to give CPU control to a task, the task queue shown is searched from left to right.

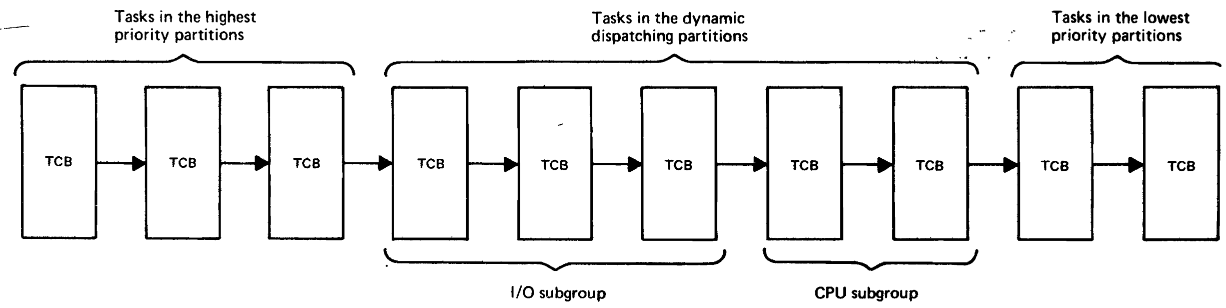

Tasks in the highest priority partitions
Tasks in the dynamic dispatching partitions
Tasks in the lowest priority partitions
TCB
TCB
TCB
TCB
TCB
TCB
TCB
TCB
TCB
TCB
I/O subgroup
CPU subgroup

Figure 90.25.1. Task queue containing a dynamic dispatching group of task:

The operating characteristic of each task in the dynamic dispatching group is determined by constantly monitoring its use of CPU time. Each time a task in the dynamic dispatching group 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 99,999 ms). The initial statistics interval value is user-specified at system generation and can be overridden during system initialization via a SETxxxxx member or by the operator. 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 in such a way that they are dispatched in a cyclic manner 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. Note that I/O-oriented tasks are not positioned within their subgroup according to the amount of the time interval they used, as is done in the automatic priority group facility of VS2. Initially, all tasks in the dynamic dispatching partitions are placed in the I/O-oriented subgroup. When a subtask enters the dynamic dispatching group of tasks as a result of the issuing of an ATTACH macro, it is placed at the beginning of the I/O subgroup. The task-switching rules for dynamic dispatching tasks are summarized in Table 90.25.1.

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 (from 1 to 99 ms). A lower limit (from 1 to 999 ms) for the adjusted time interval is 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 system initialization.

The dynamic task dispatching facility is most useful in larger VS1 environments in which the operating characteristics of many jobs are variable and/or unknown and these jobs do not have a high completion priority. The dynamic dispatcher attempts to balance CPU usage among such jobs if they are executed in dynamic dispatching partitions.

Jobs that require a certain response time (such as teleprocessing jobs) or that require fast turnaround should be executed in a partition with a higher priority than that of any partition in the dynamic dispatching group. Jobs that do not have a high completion priority or that are known to be heavily CPU-oriented can be executed in a partition with a priority lower than that of any partition in the dynamic dispatching group. (See Appendix 3 in _OS/VS1 Planning and Use Guide_, GC24-5090, for additional guidelines regarding the use of dynamic dispatching.)

When dynamic dispatching support is included in a VS1 control program for a Model 135 or 145, extended timer support should also be included if the system to be used has the CPU timer and clock comparator feature installed. Less control program time is needed to support dynamic dispatching when the extended timer facility is present.

Table 90.25.1. Task-switching rules for dynamically dispatched tasks

Reason for Loss of CPU Control	Current Status of Task	New Status of Task	Actions Taken
Preemption by a task with a higher priority than that of any dynamic dispatching task	I/O-oriented	I/O-oriented	• Preempting task is dispatched. Remaining portion of the interval is saved for use next time this dynamic dispatching task is dispatched.
	CPU-oriented	CPU-oriented	• Dynamic dispatching task is moved to the end of the CPU-oriented subgroup. • Preempting task is dispatched.
Time interval ends	I/O-oriented	CPU-oriented	• Dynamic dispatching task is moved to the beginning of the CPU-oriented subgroup. • I/O-oriented subgroup is searched for a ready task beginning with the task queued after the one just moved to the CPU-oriented subgroup.
	CPU-oriented	CPU-oriented	• Dynamic dispatching task is moved to the end of the CPU-oriented subgroup. • CPU-oriented subgroup is searched for a ready task beginning with the task queued after the one just moved.
Voluntary surrender	I/O-oriented	I/O-oriented	• I/O-oriented subgroup is searched for a ready task beginning with the task queued after the one that just gave up CPU control.
Voluntary surrender without expiration of time interval	CPU-oriented	I/O-oriented	• Dynamic dispatching task is moved to the end of the I/O-oriented subgroup.

Table 90.25.1 (continued)

Reason for Loss of CPU Control	Current Status of Task	New Status of Task	Actions Taken
			• CPU-oriented subgroup is searched for a ready task beginning with the task queued after the one just moved.
Voluntary surrender with expiration of time interval	CPU-oriented	CPU-oriented	• Dynamic dispatching task is moved to the end of the CPU-oriented subgroup. • CPU-oriented subgroup is searched for a ready task beginning with the task queued after the one just moved.

Authorized Program Facility

The authorized program facility (APF), not available in MFT, is a system integrity feature that is standard in VS1. It is designed to prevent unauthorized programs from performing functions that are designated as restricted.

Programs (load modules) that are to be authorized via APF must reside in SYS1.SVCLIB, SYS1.IMAGELIB, SYS1.LINKLIB, or an authorized library. A library in an installation is authorized by placing its name in the IEAAPF00 member in SYS1.PARMLIB. In addition, the name of each authorized program must be placed in the list of authorized programs that is maintained in a module (IEFSDPPT) of the job scheduler. (The linkage editor control statement parameter that is supported in VS2 to identify an authorized program at link-edit time is ignored in VS1.)

During system initialization, an APF table is built in the pageable supervisor area (pageable SQA) that contains the names of APF authorized programs. When the first module of a job step is loaded, the scheduler inspects the APF table. If the load module name is in the table, an authorization bit in the JSCB for the job step is turned on. Whenever a task attempts to use a restricted system function, the TESTAUTH routine is invoked by the restricted function (via a branch or the new TESTAUTH macro) to check the authorization code of the requesting task. If the task is not authorized, it is abnormally terminated.

When an authorized module issues an ATTACH, LINK, LOAD, or XCTL macro, the called module is also authorized if it resides in SVS1.SVCLIB, SYS1.LINKLIB, SYS1.IMAGELIB, or an authorized library. If the called module does not reside in one of these libraries, authorization for the entire job step is revoked (authorization bit is turned off).

The system functions that are restricted (either partially or fully) are SPIE, MGCR, WTO/WTOR, CHAP, DEQ, ENQ, DOM, MODESET, PGFIX, PGFREE, PGLOAD, CVOL, EXCPVR, OLTEP, EVENTS, the service interface routine (SIR), and certain VTAM routines. The system utilities and OLTEP are authorized via APF to use the restricted facilities.

Supervisor Locks

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

The processing of a page fault, which requires I/O interruptions to be enabled to allow the I/O interruption for a completed page-in operation to be presented, can allow code that operates with the CPU disabled to be reentered, with improper processing the result.

To prevent this situation, two lock fields are implemented in VS1, a system lock and a supervisor lock, which can be set on (locked) or off (unlocked). When a disabled page fault occurs in an executing task, the appropriate lock, system or supervisor, as indicated by the task, is turned on.

When the system lock is on, no task can be dispatched except one that is related to paging operations. The system is placed in a wait state if the page supervisor is not ready to execute when the system lock is on. When the supervisor lock is on, ready tasks that are to operate with the CPU enabled for interruptions can be dispatched but no code that operates with the CPU disabled can be executed except code that is related to paging and task dispatching operations. If no such task is ready to execute, the system is placed in the wait state. The lock used remains on until the disabled page fault is processed.

Code is included within the control program to recognize an attempt made by code to enter the disabled state by executing an SSM instruction or a MODESET macro, and to place such a task in the wait state, when necessary (the appropriate lock is on).

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

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

DEB Validity Checking

A more comprehensive method of ensuring that a task cannot access a data set associated with another task is provided in VS1 via implementation of an expanded DEB (data extent block) validity-checking scheme. A new DEBCHK macro and SVC routine are implemented. 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 a DEB. The DEBCHK macro can also be used by system programmers (most options of the macro require the issuing task to be operating in supervisor state).

Routines that perform DEB validity checking in MFT, such as OPEN and CLOSE, are modified in VS1 to use the DEBCHK macro for DEB processing.

In VS1, 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-checking routine ensures that the specified DEB is in the DEB table for the task.) A task is abnormally terminated if the DEB validity-checking routine finds the DEB to be invalid in any way.

In VS1, DEB validity checking by OPEN, CLOSE, and IOS is a standard facility. However, DEB validity checking by IOS (that is, extended DEB validity checking) can be excluded during system generation. DEB validity checking is performed only by OPEN and CLOSE routines when extended DEB validity checking is excluded.

Fast Multiple Wait

The fast multiple wait facility, not available in MFT, is standard in VS1. It is designed to improve the performance of the multiple wait facility that is supported in both MFT and VS1. When a program waits for the completion of one or more multiple events using the fast multiple wait facility, it need not include processing to test the ECBs (event control blocks) for all the events being waited upon to determine which one(s) completed. The posted ECBs are made available in a predefined table by the multiple wait facility. The COMPARE AND SWAP and COMPARE DOUBLE AND SWAP instructions are used by the fast multiple wait facility.

The EVENTS macro is provided to support fast multiple wait. It must be issued once to perform each of the following functions: define an EVENTS table for the events that are to be waited on concurrently, initialize all the ECBs that are associated with the events represented by the EVENTS table, and wait on the multiple events. An EVENTS table can be associated with up to 32,767 events. Once the EVENTS macro is issued to establish the multiple wait, as soon as any event represented by the EVENTS table completes, its ECB is posted complete and the address of the ECB is placed in the EVENTS table. The EVENTS macro that establishes the multiple wait can indicate that the program is to be notified whenever one event completes or after multiple (up to a maximum of 504) have completed.

A task that operates with protect key 0 or in supervisor state, or that is authorized via APF can specify the SHARED=YES parameter on an EVENTS macro that defines an EVENTS task. This enables the EVENTS table to be shared with all the other tasks in the partition. The other tasks can initialize ECBs in the shared EVENTS table and/or wait on ECBs in the shared table. Only one task can wait on a shared EVENTS table at a time.

STAE Macro

In MFT and VS1, an STAE (specify task asynchronous exit) macro can be issued by a task to specify the user-written exit routine that is to be entered when the task abnormally terminates. In MFT, a user-written exit routine does not receive control when certain conditions cause the abnormal termination. In VS1, an option is provided that allows the exit routine to be entered when one of these conditions occurs.

In VS1, a program that is authorized via APF can specify the TERM=YES parameter in an STAE macro to indicate that the exit routine is to receive control after an abnormal termination of the task is caused by one of the following: CANCEL command issued by the operator (completion codes 122 and 222), the DETACH macro is issued for a subtask that has not completed (completion code 13E), the execution of a job, job step, or cataloged procedure took longer than the time limit specified in the time parameter of the JOB or EXEC statement or the time specified in the reader procedure (completion code 322), or the output limit specified by the OUTLIM keyword on a SYSOUT DD statement for a job step was exceeded (completion code 722).

Programs not authorized via APF cannot specify the TERM=YES parameter. The TERM=NO parameter can be specified in an STAE macro that is issued by any program to indicate that the exit routine is not to receive control after an abnormal termination caused by one of the preceding. This is the default.

VIRTUAL STORAGE SUPERVISOR

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

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

A validity check routine (IEAXYZ5A) is provided that can be used to validate the following storage management areas and queues, which can be destroyed by routines that operate with protect key zero: fixed SQA, gotten queue elements (GQEs), problem program protected free queue elements (PFQEs), pageable PQA PFQEs, and fixed PQA free queue elements (FQAs). All or only selected queues can be validated when IEAXYZ5A is active and the queues in all or only selected partitions can be checked.

The validity check routine is executed each time the SVC first level interrupt handler is entered. If it finds an error in a queue, it sets the current task nondispatchable, terminates its own operations, and loads a disabled wait state PSW that contains a code to identify the queue in which the error was found. A standalone dump should be taken to preserve the error status.

The validity check routine can be activated by executing the IEAHONST (alias HONESTY) service routine or by link-editing the routine into the fixed nucleus. IEAHONST executes in a problem program partition. After activating the validity check routine to perform checking according to the user-specified options, the IEAHONST routine terminates and the partition becomes available for another job.

IEAHONST can be initiated using the IBM-supplied cataloged procedure HONESTY. The operator can issue a START command or enter job control statements to invoke the HONESTY procedure. The options desired can be specified in the START command or EXEC statement used to invoke the procedure, or supplied via the operator console.

The IEAHONST program is also used to restart the validity check routine with a different set of options specified and to stop its operation. Since system performance is degraded when the validity check routine is operating, it should be used selectively and when queue errors are suspected.

PROGRAM FETCH

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

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

The optional PCI fetch routine is modified for operation in a paging environment. PCI fetch does not use the channel program translation facility of IOS. It uses the EXCPVR macro (discussed in Section 90:30) instead of EXCP.

In VS1, PCI fetch requests the allocation and fixing of up to six page frames (12K) for the execution of each read operation (START I/O instruction). Text records are read into these page frames. During execution of the CCW chain, PCI chaining is suppressed if it is determined that execution of the next CCW list with a text CCW will cause the fixed real storage area associated with the I/O operation to be exceeded. The channel program then terminates and the page frames used during the operation are unfixed. PCI fetch performs address constant relocation during read operations (adds the relocation factor to virtual storage address constants contained in text records), just as in MFT.

When a program is loaded by program 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 pages of the program is required for allocation to other pages, and the page supervisor considers these pages to be eligible for replacement as per its page replacement algorithm. The change and reference bits for each page frame that contains program text are on as a result of the I/O operation that reads in the text. Hence, before the page frames allocated to a program that is being loaded (or to a recently loaded program) can be reassigned, a page-out will be performed. The fact that the change bit is turned on by the fetch operation is what causes the first and only page-out of pages that are never modified (refreshable pages).

The PCI fetch routine should not be included in VS1 control programs that are generated for Model 135 or 145 systems unless these systems have 2314/2319 disk storage. The internal speed of Models 135 and 145 is not fast enough to take advantage of PCI fetch for 3330-series, 3340, 3350, or 2305 Model 2 disk storage.

Note that in OS and OS/VS 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 that are pageable will be paged out only once, since their change bits will never be turned on during execution of the module. Pages of reentrant pageable 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 standard timing facilities of VS1 include support of time of day and date, job step timing and limiting, and interval timing (optional in MFT). These facilities are functionally the same in VS1 and MFT. However, in VS1 the time of day that is maintained in the time-of-day clock is Greenwich Mean Time (GMT) with a base of January 1, 1900, instead of local time with a base of January 1, 1960, as in MFT. The time-of-day clock can be set only during system initialization. The CLOCK parameter must contain the time of day as a GMT value and the new GMT parameter must be specified in order for the time-of-day clock to be set.

Local time of day is also maintained. The difference between GMT and local time can be specified at system generation via the TZ parameter. The time differential can also be specified during system initialization or via the SET command after system initialization, in which case the system generation time differential specification, if any, is overridden. This is accomplished by specifying the CLOCK parameter with a local time value and omitting the GMT parameter. If the TZ parameter is not specified during system generation and a CLOCK parameter without a GMT parameter is not specified during or after system initialization, GMT and local time will be the same. (Note that in VS1, the SET command is used only to change the time differential between local time and GMT and, hence, must not contain a GMT parameter.)

The operator can take one of the following actions regarding time of day during system initialization:

- Set or change the GMT value in the time-of-day clock. The operator is also prompted to specify local time (and date) after the time-of-day clock is set.

- Supply the local time of day (and date) and accept the time-of-day clock GMT value (this causes the time differential between GMT and local time to be changed).

- Accept the current time-of-day clock value and local time differential specification without change.

The local time of day and date are supplied by the timer supervisor whenever a TIME macro is issued. If GMT time-of-day and date values are required, they must be obtained using the STORE CLOCK instruction.

Extended timing facilities are provided in VS1 for systems that have the clock comparator and CPU timer installed. Extended timer support is automatically included in a VS1 control program that is generated to support a Model 135-3, 138, 145-3, 148, 155 II, 158, 165 II, or 168, since the clock comparator and CPU timer are standard in these models. Optionally, extended timer support can be included in a VS1 control program generated for a Model 135 Model 0 or 145 Model 0 or 2, for which the clock comparator and CPU timer are optional.

When extended timer support is not present in a VS1 control program, the interval timer at location 80 is used by job step timing, interval timing (STIMER and TTIMER macros), and if necessary, time-slicing or dynamic dispatching support. When extended timer support is present, the interval timer is used only by time-slicing or dynamic dispatching support. The CPU timer and clock comparator are used by job step timing and interval timing support, which results in reduced execution time for these timing routines because certain program logic that is required to handle the interval timer is eliminated (conversion of doubleword time-of-day clock units to single-word interval timer units and use of the ENQ/DEQ routines). In addition, use of the clock comparator and CPU timer improves the repeatability of timing data gathered by SMF.

The STIMERE macro is also provided by extended timing support. This macro offers timing functions that are not provided by the STIMER and TTIMER macros. The STIMERE macro can be used to set, test, cancel, test and cancel, or replace a specified interval of <u>real</u> time. It cannot be used to set an interval of <u>task</u> time or to establish an interval that is to elapse at a specific time of day. Such intervals must be established using the STIMER macro. (The CPU timer is used to indicate the end of an interval of task time and the clock comparator to indicate the end of a time-of-day interval.)

The STIMERE macro provides the following capabilities that are not provided by the STIMER and TTIMER macros:

- An interval of real time longer than 24 hours (up to approximately 100 hours) can be set.
- A time interval can be specified in microsecond units as well as in binary and decimal units.
- A task can have multiple intervals set at the same time instead of a maximum of one. A user-specified identification is assigned to each interval that is set using the STIMERE macro.
- The time remaining in an interval can be tested and canceled at the same time (that is, by issuing one STIMERE macro).
- A task can replace an established interval that was set by any of its subtasks, as well as one that it set.
- An ECB can be specified when an interval is established upon which the issuing task will subsequently wait. This allows the task to set an interval, perform some processing, and then wait for the interval to expire. The task can also choose (1) to wait after setting an interval or (2) indicate a routine that is to receive control after the interval expires and continue processing, as is possible when the STIMER macro is issued.

90:30 DATA MANAGEMENT

Data management components are altered where necessary to operate in a paging environment and to interface with JES and the modified VS1 input/output supervisor (IOS). The significant functional differences between data management in VS1 and MFT are changes in IOS to handle channel program translation and page fixing, the availability of new access methods called Virtual Storage Access Method (VSAM) and Virtual Telecommunications Access Method (VTAM), and support of a channel-to-channel adapter. VS1 also supports many new System/370 I/O devices that are not supported by MFT for System/370.

CLOSE, EOV, and DADSM routines for VS1 and MFT are functionally equivalent. The OPEN routines in VS1 support all the functions provided by MFT and two additional options: EXTEND and OUTINX. When one of these options is specified for an existing SAM or ISAM data set, records can be added to the end of the data set regardless of the disposition specified in the DD statement for the data set, that is, DISP=MOD does not have to be specified. In all other respects, the EXTEND option is equivalent to OUTPUT and the OUTINX option is equivalent to OUTIN.

VS1 supports all the access methods provided in MFT except QTAM. The same functions the access methods provide in MFT are also supported in VS1. Programs that use these access methods can be executed without modification in VS1 either in paged or nonpaged mode.

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.

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

All the VS1 access methods except VTAM, TCAM, and VSAM interface with IOS via the EXCP macro and, therefore, use the channel program translation and page-fixing facilities of IOS. TCAM can operate in a pageable partition but requires certain of its message control program elements (such as control blocks and the buffer pool) to be long-term fixed in real storage during the entire time TCAM is in operation. TCAM interfaces with IOS via the new EXCPVR macro and performs its own channel program translation. TCAM does not require long-term fixing of any portion of the message processing programs that it services. In VS1, TCAM message queues can be written on 2314/2319, 3330-series (all models), or 3340 (all models) direct access storage devices.

VTAM is the teleprocessing access method that is designed to operate efficiently in a virtual storage environment and that supports advanced communications facilities, such as 3704/3705 Communications Controllers operating in network control program (NCP) mode, systems network architecture (SNA), and synchronous data link control (SDLC) line discipline. VTAM uses the COMPARE AND SWAP and COMPARE DOUBLE AND SWAP instructions, which are optional on Models 135 Model 0 and 145 Models 0 and 2.

VTAM operates in a problem program partition. VTAM application programs operate in one or more other problem program partitions. VTAM must be started in a partition by the operator before it can be used. The minimum size partition required by VTAM is approximately 900K. During the entire time VTAM is operative in a partition, a certain number of page frames it is allocated are fixed. VTAM is terminated using the HALT command. The CANCEL command cannot be used for VTAM.

VTAM can be used with BTAM and TCAM in VS1. VTAM, BTAM, and TCAM, each with a separately defined network and each in a separate partition, can operate concurrently in the same VS1 system. TCAM message control programs and application programs and VTAM application programs can execute in the same VS1 partition and share the resources of a VTAM network. BTAM and VTAM can be used concurrently in the same partition to support separate networks if the requirements of both access methods are met.

CHANNEL-TO-CHANNEL COMMUNICATION SUPPORT

Support of a channel-to-channel adapter that connects two channels in the same or different systems operating under VS1 control is optional. This support enables two job steps in the same or different systems to communicate with each other via the interconnected channels, using read and write CCWs. The EXCP macro is used for this communication.

When channel-to-channel communication support is to be used, a channel-to-channel adapter must be defined during system generation via the IODEVICE macro to cause a UCB (unit control block) to be generated for the adapter. Job steps that are to use the adapter must specify UNIT=CTC in a DD statement to cause allocation of the adapter. For details regarding the use of this facility, see _OS/VS1 Planning and Use Guide_ (GC24-5090).

INPUT/OUTPUT SUPERVISOR

In VS1, 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 fixed SQA. This new list is used for the actual I/O operation. A CCW list with up to 239 CCWs can be translated.

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

- Short-term fixing of the pages associated with an I/O operation to prevent the occurrence of page faults during 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. This includes pages that contain control blocks (IOB, DCB, DEB, ECB/DECB, and AVT), required IOS appendage routines, and buffers.

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

The same five I/O appendage interfaces that are provided in MFT are supported in VS1, and one new appendage interface is defined. There also are new returns from the SIO and the PCI appendages. The new page fix appendage is actually part of the SIO appendage and it is entered using a new entry point into the SIO appendage. The page fix appendage is provided to enable an EXCP user to request short-term fixing of up to seven different virtual storage areas that will be referenced during an EXCP request but that are not automatically fixed by IOS.

A user-written EXCP program with user-written I/O appendages that can incur page faults can use the new appendage to short-term fix the areas referenced by the I/O appendages. The new PGFX parameter for the EXCP data control block (DCB) is provided to indicate that the page fix appendage is to be used.

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

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

VIRTUAL STORAGE ACCESS METHOD

General Description

Virtual Storage Access Method (VSAM) is a new component of OS/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 VS1 environment, VSAM supports the 3850 Mass Storage System and 2314/2319, 3330-series (all models), 3340/3344 (all models), 3350 (native or 3330-compatibility mode), 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/VS1 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.

In addition, the three data organizations supported by VSAM enable it to be used in place of QSAM or BSAM for sequential data sets and in place of BDAM for certain directly organized data sets. The new structure and features of VSAM make it more suited to data base and online environments than other OS/VS1 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 2 of VSAM. The conditional swapping instructions must be present in a system in which OS/VS1 with Release 2 of VSAM is executed.

General Description of VSAM Data Set Organizations

VSAM supports three different data set organizations, key-sequenced, entry-sequenced, and relative record, all of which allow both sequential and direct processing, record addition without data set rewrite, and record deletion. The primary difference among these three 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. Optionally, one or more alternate indexes can be created for a key-sequenced data set to enable logical records to be accessed sequentially and directly by one or more fields in addition to the primary key field. Alternate indexes are not supported by ISAM.

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. Optionally, however, one or more alternate indexes can be constructed to enable logical records to be accessed sequentially and directly by different fields.

A relative record VSAM data set is similar in organization to a fixed-length BDAM data set that is processed by relative record number. Records in a relative record data set are in sequence by ascending relative record number (from 1 to N). A relative record data set consists of a number of fixed-length slots, each of which has a unique relative record number and can contain one logical record. A record is placed in the slot specified by a user-supplied or VSAM-supplied relative record number. Relative record data sets cannot have a primary or alternate index.

Physical Structure of VSAM Data Sets

The way in which the extents of the logical records of a VSAM key-sequenced, entry-sequenced, or relative record 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. Better performance is obtained when a cylinder contains an integral number of control areas rather than a fractional number.

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. Listed below are the default control area sizes.

- For 3330-series and 2305 Model 2 devices, a control area is two tracks and contains 40 control intervals (20 control intervals per track).
- For a 3350 in native mode, a control area is two tracks and contains 54 control intervals (27 control intervals per track).
- For a 2305 Model 1, a control area is three tracks and contains 45 control intervals (15 control intervals per track).
- For a 3340/3344, a control area is four tracks and contains 48 control intervals (12 control intervals per track).
- For a 2314/2319, a control area is four tracks and contains 44 control intervals (11 control intervals per track).

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 in a key-sequenced or entry-sequenced (but not a relative record) data set can also span two or more control intervals within the same control area, in which case it is called a spanned record. If a key-sequenced or entry-sequenced data set is to have spanned logical records, this fact must be specified when the data set is defined, as the default for these organizations is not to permit spanned records.

Spanned records enable logical record size to be greater than maximum control interval size. The maximum size of a nonspanned record is 32,761 bytes. The maximum size of a spanned logical record is control area size minus ten bytes (for VSAM control information) per control interval in the control area.

A spanned record always starts at the beginning of a control interval. If the last segment of a spanned record does not completely fill a control interval, the unused space is allocated as free space that can be used only to extend the size of the spanned record. No other logical record can be placed in this free space. When spanned records are used, the maximum size of a logical record is the size of the control area.

Figure 90.30.1 shows an example of a control area that consists of three control intervals. There are three physical records in each control interval. The number of control intervals in a control area is determined by 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, and a control area contains an integral number of control intervals. The size of a control interval can be specified by the user and is used as long as it is within the limits defined by VSAM; otherwise, a user-specified control interval size is ignored.

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 or relative record data set, control information and an end-of-file indication are placed in each control interval of the control area during preformatting.

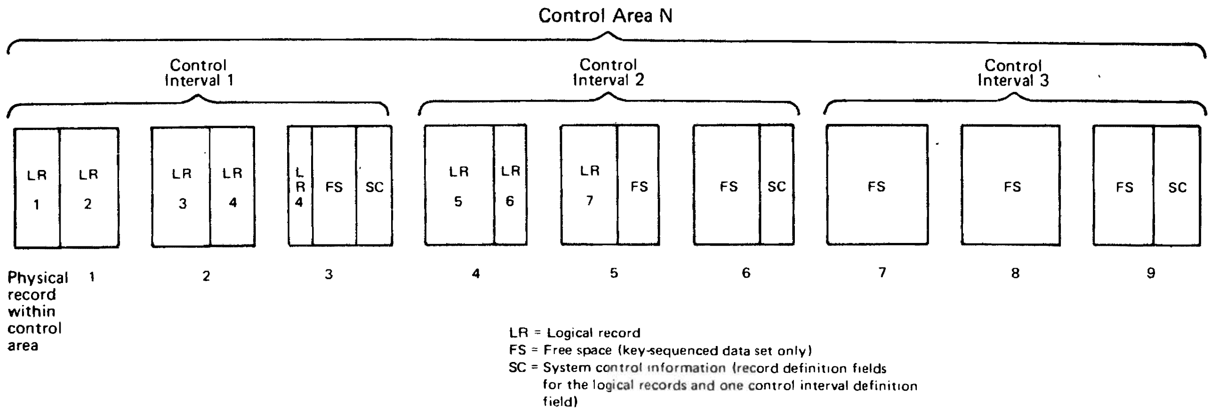

Control Area N
Control Interval 1
Control Interval 2
Control Interval 3
LR 1
LR 2
LR 3
LR 4
LR 4
FS
SC
LR 5
LR 6
LR 7
FS
FS
SC
FS
FS
FS
SC
Physical record within control area
1
2
3
4
5
6
7
8
9
LR = Logical record
FS = Free space (key-sequenced data set only)
SC = System control information (record definition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

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

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, entry-sequenced, or relative record 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 that need not be contiguous. A volume can contain multiple data spaces. The maximum size of a data space is one volume. A data space on a volume can contain one or more data sets and a data set can occupy one or more data spaces on one or more volumes.

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.

When a VSAM data set is defined, it can be allocated space within a previously defined data space or the data set and the data space it is to occupy can be defined at the same time. In the latter case, the data space can contain only the data set defined with it and the data set is called unique.

The REUSABLE attribute can be specified for a VSAM data set when it is defined to allow it to be reused multiple times (as a work file, for example) without having to delete it and then redefine it using the access method services program.

The REUSABLE attribute can be specified for a key-sequenced, entry-sequenced, or relative record data set that resides on one or more volumes. A key-sequenced or entry-sequenced data set with an alternate index or a key-sequenced data set with key ranges specified per volume cannot be reused. However, an alternate index can have the REUSABLE attribute if it does not contain unique keys (that is, does not have the UNIQUE attribute specified).

When a data set with the REUSABLE parameter specified is opened, it is set to the status of a data set opened for the first time for creation (reset to empty status). Any allocated secondary extents are deallocated. Preformatting (putting control information in all the control areas of the data set) is performed but the logical records are not erased.

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 first record in a data set has an RBA of 0. 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 VSAM data set can be moved from one device type to another and its index data set(s) need not be re-created.

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.

When control interval access is used, I/O buffers can be managed by VSAM or the user. When buffers are managed by the user, control intervals cannot be processed in the I/O buffer (a work area must be used) but the improved control interval processing option can be specified. This option provides faster processing by control interval access than does normal control interval processing but requires that certain restrictions be met (see _OS/VS Virtual Storage Access Method Options for Advanced Applications_, GC26-3819, for these restrictions).

Use of the improved control interval processing option also permits the user to have the control blocks and I/O buffers for the VSAM data set long-term fixed in real storage for the duration of processing of the data set instead of short-term fixed only when the control blocks and buffers are in use during an I/O operation. Use of the fixing option can further improve the performance achieved during control interval access. In order to use the fixing option, a program must execute in supervisor state, have protect key 0 assigned, or be authorized via APF.

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 access control block 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, ENDREQ, GETIX, and PUTIX) 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, forward or backward, 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 RPLs 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.

- SHOWCAT (display fields in a VSAM catalog). The SHOWCAT macro can be used to cause selected fields from the catalog entry for a specified data set to be moved to a user-provided work area. The data set whose catalog entry is being inspected need not be open in order for the SHOWCAT macro to be issued.

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 or relative record 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 (including a relative record number) 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).

- GETIX and PUTIX - These macros are used to process an index component of a key-sequenced data set (see discussion later under "Processing a Key-Sequenced Data Set Index").

- 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.

Figure 90.30.2 illustrates the relationships among the most frequently used control macros and the request macros.

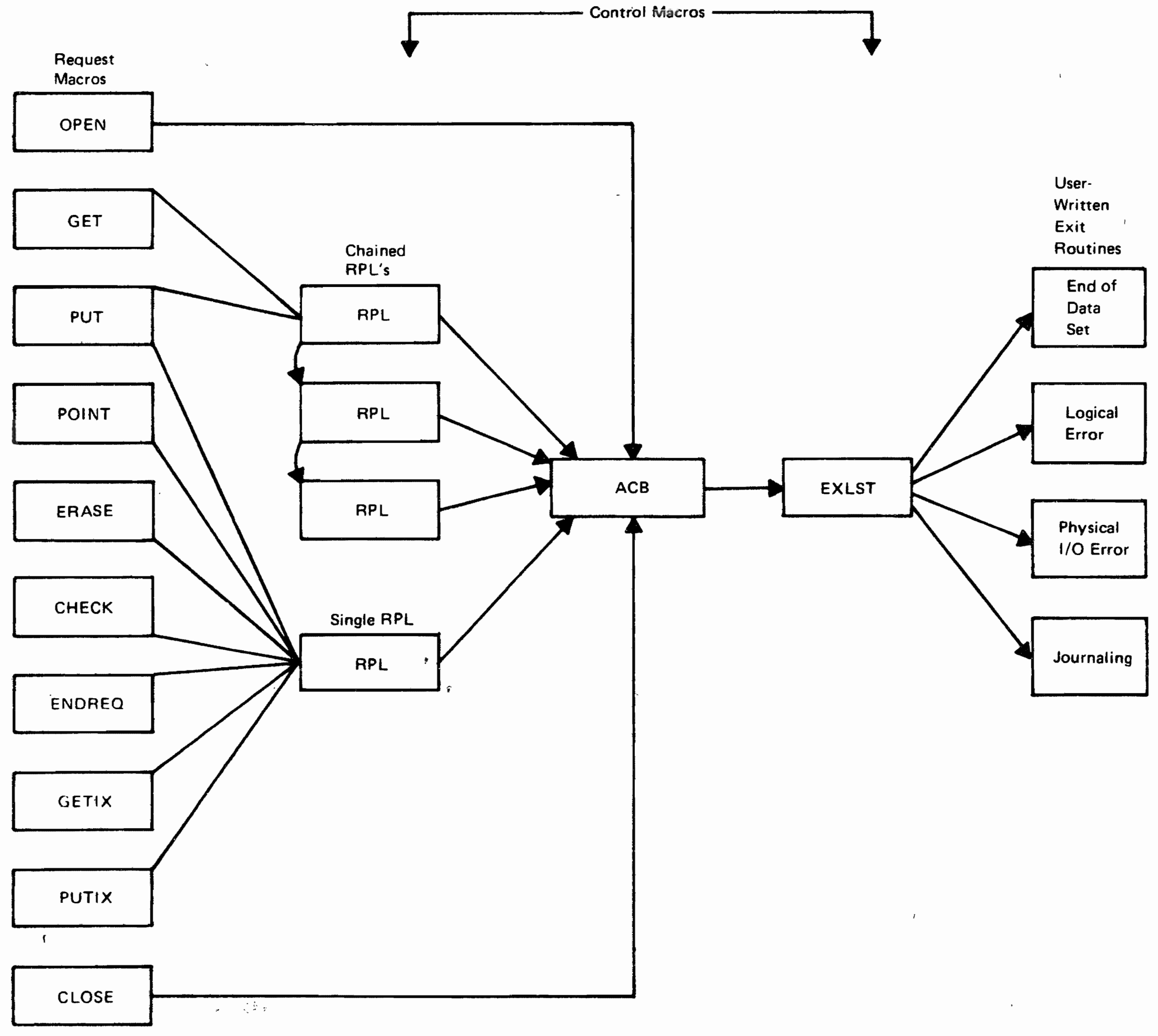


Figure 90.30.2. Relationships among VSAM control and request macros

An interface to VSAM that is designed to be used in a data base/data communications environment is also provided. Five macros are available that enable I/O buffers, I/O-related control blocks, and channel programs to be shared among several VSAM data sets and permit the user to manage I/O buffers. The sharing of I/O resources and the buffer management available can speed up the direct processing of VSAM data sets whose activity is unpredictable and the processing of one transaction that requires access to several data sets.

The BLDVRP and DLVRP macros are provided to build and delete, respectively, a shared resource pool. A resource pool in VS1 can be shared by the VSAM data sets being processed in the same partition. The WRTBFR macro causes the writing of a buffer and can be used when deferred processing is specified in the ACB. When deferred processing is specified, VSAM does not write a buffer after a PUT for direct processing is issued.

The SCHBFR macro is provided to search the shared buffer pool for a particular range of RBAs (locate a buffer) and the MRKBFR macro causes a buffer to be marked for output without issuing a PUT for update. Details regarding the use of these five macros are contained in _OS/VS Virtual Storage Access Method: Options for Advanced Applications_.

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 RPLs (chained RPLs) and synchronous or asynchronous processing. When a request consists of a list of RPLs, 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.

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. 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.

The primary index data set and the logical record data set of a key-sequenced data set form a _cluster_. Each alternate index built for a key-sequenced data set also consists of an index component and a data component. These two components form an alternate index cluster for the key-sequenced data set, which is then referred to as the base cluster.

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 and any alternate index data set cluster, however, can be placed on a device type that is different from that of the logical record data set. The primary index data set and alternate index data sets need not reside on the same type of direct access device either. In addition, the index component of an alternate index can be on a different device type than the data component of the alternate index.

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. If spanned records are used, the primary key must be contained within the first control interval.

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 some 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.

In general, when inserting records into a key-sequenced data set directly or skip-sequentially, control intervals and control areas are split in their middle, and VSAM attempts to use all the free space available in each control interval and control area. When records are inserted sequentially, the affected control interval and control area are split at the point of record insertion and VSAM attempts to reserve the free space quantity defined for the data set.

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 sequence set. All levels above the lowest are collectively referred to as the index set. The sequence set index level points to all the control intervals in the key-sequenced data set and contains the high compressed 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 90.30.3. Where a key is specified, it refers to a primary key.

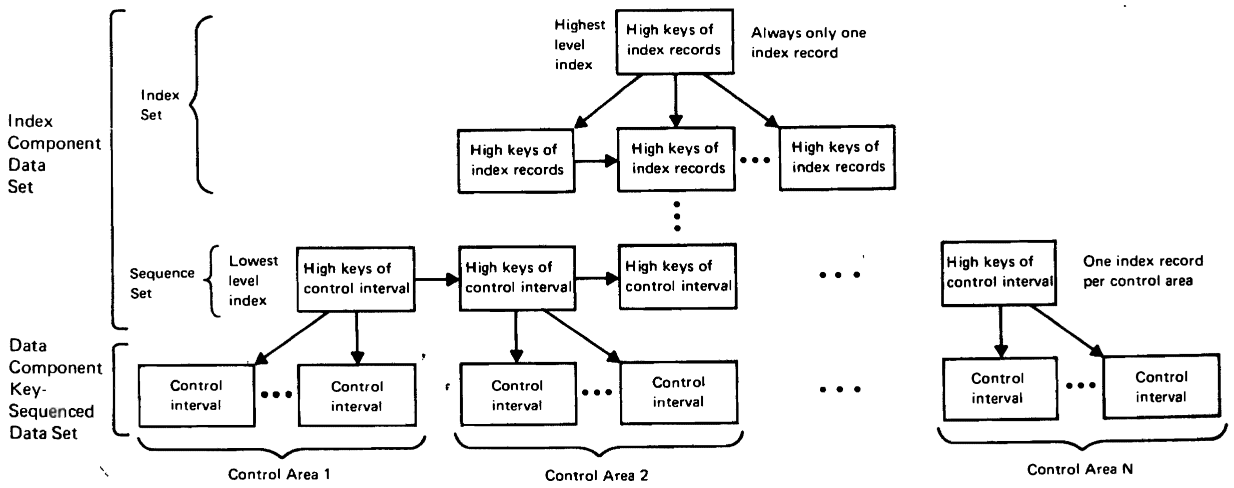

Index Component Data Set
Index Set
Highest level index
High keys of index records
Always only one index record
High keys of index records
High keys of index records
High keys of index records
Sequence Set
Lowest level index
High keys of control interval
High keys of control interval
High keys of control 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 90.30.3. 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. Physical index record size is the same as physical record size in the data component.

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.

Alternate indexes for key-sequenced data sets. Optionally, one or more alternate indexes can be built for an existing key-sequenced data set. An alternate index cannot be built for another alternate index data set or for a key-sequenced data set with the REUSABLE option assigned.

Alternate indexes enable the logical records of a key-sequenced data set to be accessed sequentially and directly by more than one field. This eliminates the necessity of having the same data stored in multiple key-sequenced data sets that are sequenced by different fields for different applications. The support of multiple indexes for a given data set makes VSAM key-sequenced organization particularly suitable for data base applications.

The alternate key for an alternate index can be any fixed-length field in a fixed position in the logical record. An alternate key, like a primary key, can be a minimum of 1 byte and a maximum of 255 bytes in length. If logical records in the data component are spanned, the alternate key field must be present in the first control interval of the spanned record. The alternate key field can overlap the primary key field and any other alternate key fields when multiple alternate indexes are defined.

When an alternate key appears in only one logical record in the base key-sequenced data set, it is a unique alternate key. If it appears in multiple logical records, it is a duplicate or nonunique alternate key. Nonunique alternate keys can appear in a maximum of 32,767 logical records in the base key-sequenced data set as long as the maximum possible length for a spanned record in the data component of the alternate index is not exceeded by the record for that alternate key. The data component of an alternate index is always permitted to have spanned records.

An alternate index can have nonunique alternate keys only if the NONUNIQUE attribute is specified when the alternate index is defined. When the NONUNIQUE parameter is not specified, any attempt to add a nonunique key to an alternate index is rejected as a logical error when VSAM is handling alternate index updating.

The index component of an alternate index cluster contains compressed alternate keys in ascending collating sequence and is physically and logically structured like the primary index for a key-sequenced data set, as shown in Figure 90.30.3. That is, the index component consists of an index set that points to successively lower levels of the alternate index and a sequence set that points to the highest alternate key in each of the control intervals in the data component of the alternate index. Physical index records are fixed in length and contain blocked index entries. Index entries in a primary and alternate index contain the same type of information.

The data component of an alternate index is identical in physical format to the data component of a key-sequenced data set. It contains one variable-length logical record for each unique alternate key value. For a key-sequenced data set, this alternate key record contains system header information, the alternate key value (not compressed), and the primary key field (not compressed) of the logical record in the base key-sequenced data set that contains the alternate key. If the alternate key appears in more than one logical record in the base data set, the primary key of each of these logical records is contained in the alternate key record for that alternate key.

Primary keys are ordered in time-of-arrival sequence within the logical record for a nonunique alternate key. That is, when another primary key is added to an alternate key record in the data component of an alternate index, it is placed at the end of the existing list of primary keys.

A <u>path</u> is the means by which a base key-sequenced data set is related to one of its alternate indexes. A path is defined and named using the access method services program. Optionally, a password can be assigned to the path. One path must be defined for each of the alternate indexes of a key-sequenced data set. When a given alternate index is to be used to process a key-sequenced data set, the path associated with that alternate index must be specified in an OPEN macro. This causes both the key-sequenced data set and the alternate index to be opened.

When a path is opened, only keyed processing requests can be used. Addressed and control interval access are not permitted. The keyed processing that can be performed on a key-sequenced data set using an alternate key is the same as can be performed using a primary key. That is, existing records in the base cluster can be retrieved, updated, or deleted, and new records can be added using alternate key values. These operations can be performed using keyed sequential, keyed skip-sequential, or keyed direct processing.

An alternate index cluster (index and data component data sets) has its own name and can be processed as a key-sequenced data set independently from its associated base key-sequenced data set. To process an alternate index cluster independently, the OPEN macro must state the alternate index name or the path name associated with the alternate index. In the latter case, the AIX option must be specified in the ACB for the alternate index to cause independent processing of the alternate index.

When a key-sequenced data set is being accessed by an alternate key, the index component of the associated alternate index is searched, using the same technique as is used for a primary index, to find a pointer to the appropriate control interval in the data component of the alternate index. When the desired alternate key record is located in the data component, the primary key is obtained and is used in the search of the primary index, which points to the control interval in the base data set that contains the desired logical record.

An alternate index must be defined, using the access method services program, before it can be created. An alternate index can be defined only after its associated base data set has been defined, and it can be loaded only after the base data set has been loaded. An alternate index can be created using the access method services program (BLDINDEX command). Alternatively, a user-written program that performs the same functions as the BLDINDEX command can be used.

The BLDINDEX command causes a sequential scan of the specified key-sequenced base data set, during which alternate key values and the primary keys of the logical records in which they reside are extracted. The extracted alternate keys are sorted into ascending sequence. Alternate index records are constructed from the sorted alternate keys and their associated primary keys. These index records are then placed in the alternate index (a key-sequenced data set).

Alternate index maintenance can be handled by the user or automatically by VSAM. Alternate index updating by VSAM is requested by specifying the UPGRADE attribute for an alternate index when it is defined. Specifying the UPGRADE attribute for an alternate index makes the index a part of the <u>upgrade set</u> of alternate indexes for a given key-sequenced data set. An alternate index can be part of the upgrade set for a key-sequenced data set even though it is not a member of a path for the key-sequenced data set. The maximum number of alternate indexes that can be part of the upgrade set for a given key-sequenced data set is 125.

Whenever a key-sequenced data set is opened for keyed or addressed output operations, VSAM automatically opens for output all the alternate indexes in the upgrade set for the base data set, unless the NOUPDATE job control parameter is specified for the key-sequenced data set. If NOUPDATE is not specified, then, whenever an existing logical record is erased or updated or a new logical record is added during processing of the base data set (by a primary or an alternate key), the upgrade set of alternate indexes is updated as appropriate. This updating is done as part of the processing of the logical record. The journaling exit is not taken during updating of the alternate indexes.

The upgrade set of alternate indexes is not updated by VSAM if control interval access is used to process a base data set. In addition, VSAM does not update any alternate indexes for a base key-sequenced data set when the NOUPGRADE attribute is specified for the alternate index. Updating must be performed by the user. VSAM assumes all required updating of the alternate indexes for a key-sequenced data set has been done and does not make any synchronization checks between a key-sequenced data set and its alternate indexes during OPEN processing.

During processing of a base key-sequenced data set via an alternate index, an error can occur while processing the key-sequenced data set, the alternate index being used to access the base, or an alternate index in the upgrade set. When an error occurs, VSAM returns a function code to the RPL used in the request that indicates which data set was involved in the error.

Key-sequenced data set processing. The records in a key-sequenced data set can be processed sequentially, skip-sequentially, or directly using the primary or an alternate key. Such processing is called keyed sequential, keyed skip-sequential, or keyed direct processing, respectively. Keyed access can be used for a key-sequenced data set that contains nonspanned or spanned records. All volumes of a key-sequenced data set must be mounted at OPEN time 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.

Addressed processing cannot be used with key-sequenced data sets that contain spanned records, since the spanned record may not be contained in physically contiguous control intervals. Spanned records cannot be processed in locate mode (in the I/O buffer). A work area is required.

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. 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.

The following types of operations, which are not supported by ISAM, can be performed on key-sequenced data sets using keyed sequential processing:

- Records can be processed sequentially by an alternate key. Existing logical records can be retrieved, updated, and erased and new records can be added to a key-sequenced data set using keyed sequential processing by an alternate key. Logical records containing the same nonunique alternate key are presented in the sequence in which they were entered in the data component of the alternate index (for both forward and backward processing, which is described below). The sequence set of the alternate index and the primary index are used during this type of processing.

- Records can be processed in descending primary or alternate key sequence. This is called previous or backward processing. GET, PUT (for update only), ERASE, and POINT macros can be used with backward processing as with forward processing. Switching between forward and backward processing can be accomplished using the POINT macro or a GET macro for direct processing.

- A mass sequential insertion technique 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. Previous processing is not supported for skip-sequential operations. Primary or alternate keys can be used. Only the sequence set of the primary index is used for skip-sequential processing using the primary key. When an alternate key is used, the sequence set of the alternate index and the entire primary index are used.

If a nonunique alternate key is encountered during skip-sequential retrieval operations, the first logical record indicated in the alternate index record in the data component of the alternate index is presented in response to the GET macro, and an indication is given that additional records exist. These must be retrieved by keyed sequential processing.

When a relatively small number of transactions that are in primary (or alternate) 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. Skip-sequential processing by alternate key offers the advantage of eliminating a search of the index set of the alternate index for each record to be processed.

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 key value (primary or alternate) 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 (or an entire alternate index and the entire primary index) is searched to locate the requested logical record during keyed direct processing. As for keyed skip-sequential processing, if a nonunique alternate key is specified, only the first logical record with that alternate key is presented and the user must obtain the others via keyed sequential processing.

Addressed sequential can be used to process the logical records of a key-sequenced data set in ascending (forward) or descending (backward) 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 cannot 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. Backward processing is not permitted. 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.

<u>Processing of the primary or alternate index data set for a key-sequenced data set</u>. The primary index component of a key-sequenced data set can be processed independently from the data component. If a key-sequenced data set <u>cluster</u> is opened, the GETIX and PUTIX macros can be used to process the primary index. Index records can be retrieved and updated (GETIX for update followed by a PUTIX for update). Only direct processing by control interval can be used with the GETIX and PUTIX macros. The RBA of the desired control interval must be specified with each request.

The primary index of a key-sequenced data set can also be processed using GET and PUT macros. In this case 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. The alternate indexes for a key-sequenced data set can be processed in the same ways as can a key-sequenced data set.

Processing summary. Table 90.30.1 summarizes the primary types of processing supported for key-sequenced VSAM data sets (control interval processing is not included in the table).

Table 90.30.1. 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. Where keyed processing is specified, the key can be the primary or an alternate key.)

Type of Access	Keyed Sequential (forward and backward processing)	Keyed Skip-Sequential (forward processing only)	Keyed Direct	Addressed Sequential (forward and backward processing)	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.
Create and add (PUT without update)	Creation using primary key only. Forward processing only. 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. Although an entry-sequenced data set consists only of a data component and cannot have a primary index, it is still referred to as a cluster.

All logical record extents of an entry-sequenced data set must be placed on volumes of the same direct access type. However, an entry-sequenced data set and its alternate index data set(s), if any, can be placed on different direct access device types. The index component and the data component for an alternate index can also be on different device types.

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 RBAs of the logical records do not change.

<u>Alternate indexes for entry-sequenced data sets</u>. Optionally, one or more alternate indexes can be built for an existing entry-sequenced data set. An alternate index cannot be built for another alternate index data set or for an entry-sequenced data set with the REUSABLE option assigned. An alternate index for an entry-sequenced data set has the same physical structure, logical organization, and attributes as an alternate index for a key-sequenced data set, and both types of alternate index are created and maintained using the same techniques (see alternate index discussion under "Key-Sequenced Data Set Organization and Processing").

The only way an alternate index for an entry-sequenced data set differs from one for a key-sequenced data set is that it contains RBA values instead of primary keys in its data component. That is, each alternate key record in the data component contains one or more RBA's of the logical record(s) in the entry-sequenced data set that contain that alternate key. The RBAs obtained from the alternate index are used to directly retrieve the required logical records from the entry-sequenced data set.

If an alternate index for an entry-sequenced data set is to be created and/or maintained by the user instead of by using BLDINDEX and VSAM upgrade set support, RBA values must be obtained. Whenever a new logical record is placed in an entry-sequenced data set (either during data set creation or when making additions), VSAM returns the RBA of the record. These RBAs can be gathered and used to create and maintain the alternate index. An alternate index must be updated only when a new record is added to, or deleted from, the base entry-sequenced data set or when the size of an existing record is increased or decreased (a record size change causes a change in the RBA of the record).

<u>Entry-sequenced data set processing</u>. Addressed sequential, addressed direct, and control interval processing are supported for entry-sequenced data sets that do or do not contain spanned records. When addressed sequential is used, records can be processed in ascending or descending 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. When addressed sequential is used to process records in descending RBA sequence, records can be retrieved, updated, and marked deleted. New records cannot be added and the size of existing records cannot be changed.

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. For addressed sequential, only forward processing is permitted. 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.

When an alternate index is created for an entry-sequenced data set, the records it contains can be processed using sequential, skip-sequential, or direct processing by the alternate key value. Records can be retrieved, updated (but not changed in size), and marked deleted when the alternate index is used to access the entry-sequenced data set.

An entry-sequenced data set can also be used like a BDAM data set. Instead of using an alternate index of RBA and control field values to process the records directly, 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 90.30.2 summarizes the primary types of processing supported for VSAM entry-sequenced data sets. Access by control interval is not included in the table.

Table 90.30.2. Types of processing supported for VSAM entry-sequenced data sets. (An entry indicates whether the access type is supported, an alternate key or RBA is required, and alternate keys or RBAs must be presented in sequence.)

Type of Access	Processing Not Using an Alternate Index		Processing Using an Alternate Index		
	Addressed Sequential (forward and backward processing)	Addressed Direct	Keyed Sequential (forward and backward processing)	Keyed Skip-Sequential (forward processing only)	Keyed Direct
Retrieval only (GET without update)	No RBA required	RBAs not in sequence	No keys required	Keys in ascending sequence	Keys not in sequence
Retrieval and update without record size changes (GET and PUT for update)	No RBA required	RBAs not in sequence	No keys required	Keys in ascending sequence	Keys not in sequence
Create and add (PUT without update) after end of file	Forward processing only. No RBA required.				
Delete	Records marked deleted by user identification. No RBAs required.	Records marked deleted by user identification. RBAs not in sequence.	Records marked deleted by user identification. No keys required.	Records marked deleted by user identification. Keys in ascending sequence.	Records marked deleted by user identification. Keys not in sequence.

Relative Record Data Set Organization and Processing

A relative record data set consists of a number of fixed-length slots, 1 to N, where N is the maximum number of records that the data set can contain. A slot has a unique relative record number and can contain one logical record. Logical record size and the RBA of a logical record cannot change.

Each control interval in a relative record data set contains the same number of slots. The record length specified for a relative record data set when it is created determines the size of a slot. Free space cannot be left within control intervals and control areas when a relative record data set is defined. All extents of the data set must reside on the same device type. An index cannot be created for a relative record data set; however, a relative record data set is still considered to be a cluster.

Keyed sequential, keyed skip-sequential, keyed direct, and control interval processing are supported for relative record data sets. The relative record number is used as a key for keyed processing. A relative record data set can be created using keyed sequential, skip-sequential, or direct processing.

Keyed sequential processing of a relative record data set is like keyed sequential processing of a key-sequenced data set. Records can be processed in ascending or descending sequence by relative record number. A key value (relative record number) is not supplied by the user. VSAM retrieves the records in slot number sequence and returns the relative record number of each logical record retrieved. GET, PUT, and ERASE macros can be used to retrieve, update, add, and delete records. The POINT macro can be used to position VSAM forward or backward to a given relative record number.

When the ERASE macro is issued to delete a record during keyed sequential processing, VSAM writes binary zeros in the indicated slot. The slot can then be reused. That is, another record of the same length can be placed in the slot. During sequential retrieval operations, deleted records are not presented to the user. When a new record is added during keyed sequential processing, it is placed in the next highest available slot relative to the current slot position and the relative record number of the selected slot is returned to the user.

Keyed skip-sequential and keyed direct processing can be used to retrieve, update, add, and delete records in a relative record data set. For keyed skip-sequential processing, the relative record number of the desired records must be supplied by the user in ascending sequence. Backward processing is not supported. For keyed direct processing, the relative record numbers supplied need not be in any sequence.

VSAM converts the supplied relative record number to an RBA value to determine the control interval that contains the requested record for keyed skip-sequential and keyed direct processing. If a deleted record is requested, a no-record-found indication is returned. When a record is added to the data set, the relative record number of a slot that does not contain a record must be specified. If a slot past the current end of file indicator is specified, VSAM preformats the file from the current end of file up to and including the control interval that is to contain the new record.

A relative record 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 relative record data set can be created using sequential control interval processing. Only forward processing is permitted for sequential operations.

GET, PUT, POINT, CHECK, and ENDREQ macros can be used with control interval processing. If updating is to be performed, a work area must be used. A relative record data set cannot be extended using control interval processing.

Table 90.30.3 summarizes the primary types of processing supported for a relative record data set. Control interval access is not included in the table.

Table 90.30.3. Types of processing supported for VSAM relative record data sets. (An entry indicates whether the access type is supported, a key is required, and keys must be presented in sequence.)

Type of Access	Keyed Sequential (forward and backward processing)	Keyed Skip-Sequential (forward processing only)	Keyed Direct
Retrieval only (GET without update)	No keys required	Keys in ascending sequence	Keys not in sequence
Retrieval and update without record size change (GET and PUT for update)	No keys required	Keys in ascending sequence	Keys not in sequence
Create and 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

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 VS1 operating system that contains VSAM. (Note that a VSAM master catalog cannot be stored on a volume in a 3850 Mass Storage System.) 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 VS1 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 and secondary allocation, if any

- 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 data sets and their index(es): the index and data components of a key-sequenced data set; the index and data components of an alternate index cluster; the alternate index and the base cluster of a path; and an alternate index and upgrade set and its base cluster.

- 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, alternate index component, path, upgrade set, 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 SHOWCAT, 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.

A recovery facility is available for VSAM catalogs that enables VSAM data sets and catalog entries for both VSAM and non-VSAM data sets to be recovered in the event that a VSAM catalog cannot be read for any reason. Use of the recovery facility for a VSAM catalog is specified via the RECOVERABLE attribute. Use of this facility is optional.

When a catalog is to be recoverable, catalog information for each data set described by the catalog is recorded in both the catalog and a recovery area on the first volume of the data set on which a data space is defined. Thus, each volume identified by a recoverable catalog contains its own catalog information.

A catalog recovery area is automatically reserved on a volume by VSAM when the first data space allocation occurs for the volume. Initially, one cylinder is allocated. If this space becomes filled, one additional cylinder is allocated each time a cylinder is filled. A catalog recovery area can contain a maximum of 16 cylinders.

The location of the catalog recovery area is specified in the format 4 label for the volume and is not indicated in the associated catalog. Whenever an entry in a recoverable catalog is updated, the corresponding catalog information in the catalog recovery area of the affected volume is also automatically updated. This means the affected volume must be mounted.

The EXPORTRA, IMPORTRA, and LISTCRA commands of the access method services program are provided to recover catalog entries and VSAM data sets. The EXPORTRA command accesses the catalog recovery area for the specified VSAM data sets in order to open them and then produce a copy of the specified VSAM data sets. Data set and catalog reorganization occurs during the rewriting process. The IMPORTRA command is then used to reestablish the copied VSAM data set and its catalog entry in a VSAM catalog. For non-VSAM data sets, the EXPORTRA command extracts data from a catalog recovery area that can be used by the IMPORTRA command to redefine the non-VSAM data set in a VSAM catalog.

The LISTCRA command can be used to list the entire contents of one or more catalog recovery areas or to list only those entries that do not have a corresponding entry in the specified VSAM catalog.

The RESETCAT command can be used instead of EXPORTRA and IMPORTRA to recover a catalog that is inconsistent with the catalog recovery areas of the volumes it defines. The RESETCAT command processes only the recoverable catalog and its associated catalog recovery areas; that is, no movement of the data sets defined in the catalog occurs.

RESETCAT can be used to synchronize the entries in a recoverable catalog to the existing level of all the volumes it describes or to a previous level using a restored backup copy of the unusable catalog or restored backup copies of the associated catalog recovery area volumes, as appropriate. When RESETCAT is used, all catalog entries are reset. Selective resetting of specific entries is not permitted. The EXPORTRA/IMPORTRA method can be used to selectively repair specific catalog entries .

A master or user catalog in one VS1 or VS2 operating system can be shared with another VS1 or VS2 operating system as a master or user catalog with one exception. A master catalog in one VS2 operating system can be shared with another VS2 system only as a user catalog (not as a master catalog). Catalog management routines control this sharing.

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 VS1, the calling sequence and commands can be placed in the input stream or issued within a processing program. The ATTACH, LINK, LOAD, or CALL macro can be issued in a program to invoke the access method services 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 paths and 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, entry-sequenced, or relative record) data set. The output can be a VSAM (key-sequenced, entry-sequenced, or relative record) or SAM data set. A range of records that are to be processed can be specified for the input file (by key, RBA, or relative record number). 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.

- Create a backup copy of a VSAM catalog and reload it if necessary. The REPRO function can be used to unload a VSAM catalog to a SAM data set or a VSAM key-sequenced or entry-sequenced data set. The copy cannot be used as a catalog but can be unloaded (using REPRO) into a VSAM catalog if the original catalog becomes unusable. The copy can be reloaded to an earlier or later version of the original catalog that was unloaded or to a newly defined catalog.

- Create an alternate index for a key-sequenced or entry-sequenced data set. Multiple alternate indexes for the same data set can be built at the same time.

- Print all or the specified range of 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. The DELETE function is also used to delete paths and uncatalog non-VSAM data sets.

- 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.

  Exportation can be temporary or permanent. If it is temporary, a copy of the data set is retained in the exporting system. Thus, a copy of the data set exists in both the exporting and the importing systems. If exportation is permanent, the catalog entry and space for the data set are deleted from the exporting system so that the data set is contained only in the importing system.

  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.

- Perform catalog recovery functions using the EXPORTRA, IMPORTRA, and LISTCRA commands or the RESETCAT command, as previously described

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

- Read checkpoint data sets to identify the tape data sets that were in use at the time each checkpoint was taken. The CHKLIST command causes the following to be listed for each tape data set open at the time of the identified checkpoint: data set name, DD name, type of unit on which the volume was mounted, sequence number of the mounted volume, and volume serial numbers of the mounted volumes. This information identifies the tape volumes needed for a restart.

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), alternate indexes, paths, 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(es), and its catalog entry. Any operation (read, add, update, delete) can be performed on the data set and its catalog entry. The master password of the base key-sequenced data set must be specified when an alternate index is to be created for the base.

- 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.

Data Set Sharing

A VSAM data set can be accessed concurrently by two or more subtasks within the same partition and two or more job steps (partitions) 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 partitions is controlled by using the SHAREOPTIONS parameter of the DEFINE command when the VSAM data set is defined. The following types of cross-partition-sharing options are supported:

- The data set can be opened by one user for output processing (to update or add records) or 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) and 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). Control interval splitting should be avoided when this option is used.

Data set sharing by subtasks within the same partition 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 partition 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-partition sharing (as described above) can be used concurrently.

When multiple DD statements are used, multiple subtasks within a partition can share a VSAM data set using the same options as are supported for cross-partition 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. Control interval splitting should be avoided.

- 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-partition 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 <u>OS/VS Virtual Storage Access Method Programmer's Guide</u> (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.

Virtual Storage Requirements

| VSAM routines, which require 382,800 bytes of virtual storage, are automatically made resident in the pageable supervisor area during system initialization when VSAM support is present in a VS1 system. Virtual storage within a problem program partition is required for VSAM control blocks, VSAM buffers, the access method services routines (when used), and ISAM interface routines and control blocks (when used).

Summary

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

VSAM provides new features:

- Three data organizations are supported.
- 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.
- Multiple indexes are supported for key-sequenced and entry-sequenced data sets.
- 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.
- Records can be retrieved sequentially in descending as well as ascending key or RBA sequence.
- 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 90.30.4 compares the features of VSAM and ISAM as supported in OS/VS1.

Table 90.30.4. Comparison table of VSAM and ISAM facilities for OS/VS1

Characteristic	VSAM - OS/VS1	ISAM - OS/VS1
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), 3340/3344 (all models), 3350 (in native and 3330-compatibility mode), 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. One or more alternate indexes are optional.	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. Alternate indexes are not supported.
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. One or more alternate indexes are optional.	Not supported
c. Relative record	Yes Fixed-length records are sequenced by ascending relative record (slot) number sequence. Indexes are not supported.	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 and any alternate index data sets can be on a device type that is different from that which contains the key-sequenced or entry-sequenced logical records.	Yes for all the volumes containing prime and separate overflow area extents. Index levels can be on a device type that is different from that which contains prime and overflow areas.
c. All volumes must be online at OPEN regardless of the type of processing	No 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	Yes, optionally, at the end of each prime cylinder. Free space on tracks within the prime cylinders can be created only by

Table 90.30.4. Comparison table of VSAM and ISAM facilities for OS/VS1 (continued)

Characteristic	VSAM - OS/VS1	ISAM - OS/VS1
	the tracks of a cylinder.	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
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. A logical record can span control intervals. Original records and additions are blocked.	Yes Fixed or variable, blocked or unblocked record formats are used for prime records. Records in an overflow area are always unblocked.
b. Key field is written on disk	No 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 for primary key. Nonunique alternate keys are supported.	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.

Table 90.30.4. Comparison table of VSAM and ISAM facilities for OS/VS1 (continued)

Characteristic	VSAM - OS/VS1	ISAM - OS/VS1
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 or descending primary or alternate key sequence. An alternate index (and the primary index) or 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 An alternate index (and the primary index) or only the sequence set index of the primary index is used.	No
c. Sequential retrieval and update by record address	Yes, via RBA	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.

Table 90.30.4. Comparison table of VSAM and ISAM facilities for OS/VS1 (continued)

Characteristic	VSAM - OS/VS1	ISAM - OS/VS1
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 sets, 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 indexes, 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 partition	Yes	Yes
b. Across partitions (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 90.30.4. Comparison table of VSAM and ISAM facilities for OS/VS1 (continued)

Characteristic	VSAM - OS/VS1	ISAM - OS/VS1
14. Utility program functions	Access method services program can perform the following: • Define and delete direct access space for a VSAM data set, catalog, index, etc. • 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 • Build alternate indexes • Process checkpoint data sets to determine the tape volumes required for a restart	IEBISAM utility can perform the following: • Copy an ISAM data set from one disk volume to another, dropping deletions and merging additions into the prime area • Unload an ISAM data set onto a tape or a disk volume, dropping deletions and creating a backup sequential data set suitable for input to the load operation to re-create the ISAM data set • Load a previously unloaded ISAM data set from tape or disk onto a disk volume, merging additions into the prime area • Retrieve and print the records of an ISAM data set, except deletions, or create a sequentially organized data set from active records

90:35 PAGE MANAGEMENT

GENERAL FUNCTIONS

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

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

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

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

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

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

- Make one or more virtual storage pages addressable (PGLOAD macro). The service performed is like that for PGFIX except that the page frames allocated are not fixed. The PGLOAD macro provides a page-ahead facility. Teleprocessing application programs can be authorized to use PGLOAD to identify the pages required to process the current transaction, for example.

- Page out one or more pages (PGOUT macro). Unchanged page frames specified in a PGOUT macro are placed in the available page queue without a page-out while changed page frames are placed in the page-out queue for a paging device. The PGOUT macro performs the opposite function from the PGLOAD macro and can be used to cause unwanted pages to be paged out immediately so that their assigned page frames become available for reassignment more quickly.

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

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

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

REAL STORAGE MANAGEMENT

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

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

An RSPT entry contains the TCB identification of the partition that caused the page frame to be assigned, the number of the virtual storage page to which the page frame is assigned or zeros if the page frame is unassigned, a fix counter, flags to indicate its status (such as long-term fixed, being paged in or out, conditionally allocated to a nonpageable job step), and queue pointers that indicate the page status queue of which the RSPT entry is a part (if any).

Logically, the following page status queues are maintained:

- Available page 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. Allocated page frames that are considered to be inactive, as per the page replacement algorithm, are placed in this queue. An available page count (APC) is maintained that always reflects the number of page frames in this queue.

  A low threshold value and a high threshold value are also maintained for the APC. These two APC threshold values are system-determined and cannot be changed by the user. The APC low threshold value determines when page frames should be added to the available page queue and does not vary during system operation. The APC high threshold value determines the number of page frames that are added to the available queue and varies during system operation, depending on the number of active initiators. The high threshold value is changed each time an initiator is activated or deactivated.

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

- Logical page I/O-in-progress queue, which indicates the page frames involved in a page-in or page-out operation. RSPT entries for these page frames are not actually connected to form a queue.

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

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

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

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

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

Real Storage Allocation Procedure

The following is done to service a real storage allocation request (refer to Figure 90.35.1). The real storage reclamation routine, a subroutine of the dynamic storage allocation routine, determines whether a page-in can be avoided because the contents of the referenced virtual storage page are still in real storage. This condition exists when the page frame last assigned to the virtual storage page has not yet been reassigned and the required page is not currently in the process of being paged in. If the RSPT entry for a desired page frame is still in the available page queue, an in-use queue, the page-out queue, or the logical fix queue, page reclamation is possible and the page frame is reassigned without a page-in.

Note that when a page frame is deallocated from a virtual storage page, the invalid bit in the associated page table entry is turned on. However, the user bit is left on and the address of the page frame being deallocated is not zeroed. Therefore, a page table entry indicates the page frame last assigned to its associated virtual storage page until the page frame is reassigned.

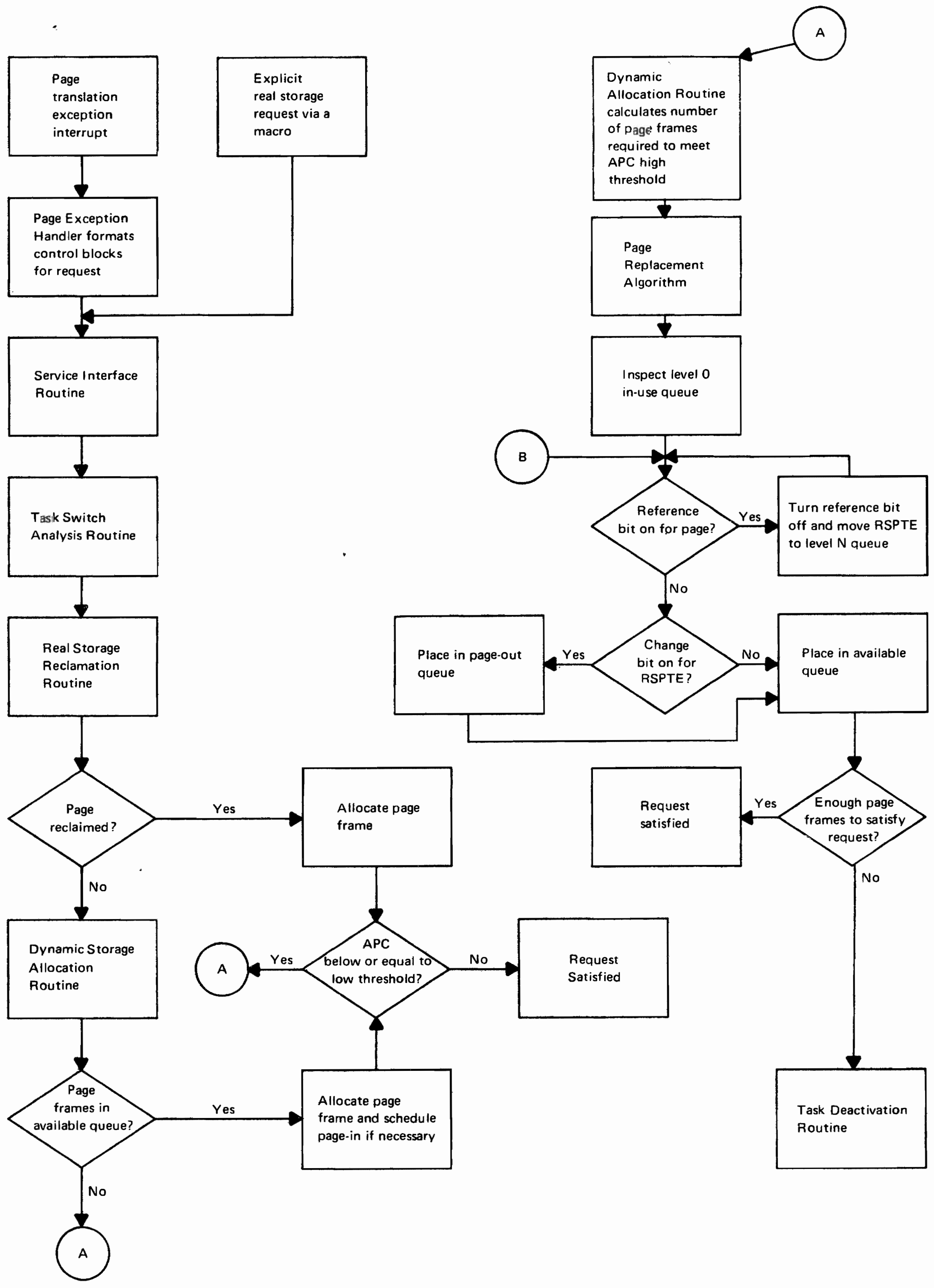

A
Page translation exception interrupt
Explicit real storage request via a macro
Dynamic Allocation Routine calculates number of page frames required to meet APC high threshold
Page Exception Handler formats control blocks for request
Page Replacement Algorithm
Service Interface Routine
Inspect level 0 in-use queue
B
Task Switch Analysis Routine
Reference bit on for page?
Yes
Turn reference bit off and move RSPTE to level N queue
No
Real Storage Reclamation Routine
Place in page-out queue
Yes
Change bit on for RSPTE?
No
Place in available queue
Page reclaimed?
Yes
Allocate page frame
Request satisfied
Yes
Enough page frames to satisfy request?
No
No
Dynamic Storage Allocation Routine
A
Yes
APC below or equal to low threshold?
No
Request Satisfied
Page frames in available queue?
Yes
Allocate page frame and schedule page-in if necessary
Task Deactivation Routine
No
A

Figure 90.35.1. Flow of the real storage allocation procedure

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

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

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

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

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

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

The relative activity of pages is determined by manipulating the in-use queues, which contain the RSPT entries for page frames that are allocated to nonfixed pages. The number of in-use queues that exist at any given time is determined by the number of active initiators. The maximum number of in-use queues supported is twelve.

Each in-use queue is assigned a reference level sequence number, which can be from 0 to 11. The first in-use queue is assigned reference level sequence number 0. The last existing in-use queue at any time is referred to as the level N queue instead of by its actual number.

The level 0 in-use queue contains RSPT entries for the page frames referenced longest ago and, hence, tends to identify the least active pages. The level N queue contains RSPT entries for the page frames most recently allocated and referenced and tends to identify the most active pages. The level N-1 queue indicates the next most active pages, etc.

When nine or more initiators are active, there are twelve active in-use queues. During system initialization, three in-use queues (level 0, level 1, and level 2) are activated to account for the paging activity required by system routines in the pageable supervisor area (JES tasks, pageable type 3 and 4 SVC routines, pageable reentrant routines, pageable access methods, etc.). Thereafter, additional in-use queues are activated as initiators are activated and deactivated as initiators are deactivated.

An additonal in-use queue is activated, which becomes the new level N queue, whenever one of the following events occurs to cause initiator activation:

- A START initiator command is processed to activate an initiator and partition.
- An activated initiator that is waiting for work receives a job to process.
- A partition in deactivated status is reactivated by the operator via a DEFINE command or by the paging algorithm.
- RTAM or VTAM is started by the operator.

The highest level in-use queue (level N) is deactivated (and the level N-1 queue is made the new level N queue) whenever one of the following events occurs to cause initiator deactivation:

- A STOP initiator command is processed to deactivate an initiator and partition.
- An activated initiator enters the wait state because there are no queued jobs with the class(es) it is assigned to handle.
- The partition deactivation routine deactivates an operating partition.
- RTAM or VTAM is stopped by the operator.

A start/stop routine is entered whenever an initiator is activated or deactivated. This routine adds or deletes an in-use queue, changes the APC high threshold value, and changes certain other values that are used by the page replacement algorithm and the page measurement routine, as discussed later.

An RSPT entry is placed at the end of the level N in-use queue when its associated page frame is assigned to a virtual storage page. If the RSPT is taken from the available page queue, its reference bit is turned on. If a page-in operation was required prior to placing the RSPT in the level N queue, the reference bit is already on as a result of the I/O operation.

Page management ensures that the reference bit is turned on when a page frame is allocated so the RSPT entry will cycle through the in-use queues at least twice before it becomes eligible for placement on the available page queue. This technique allows a task to use a page frame it has been assigned before the page frame becomes eligible for assignment to another page.

The activity (frequency of reference) of a page frame is determined by inspecting its reference bit setting at certain intervals. The page measurement routine is called to measure the frequency of reference of page frames in the in-use queues at task dispatch time and system wait time.

Activity measurement is not performed every time a task switch takes place. The frequency of measurement is based on the number of active initiators. As the number of active initiators increases, the frequency of measurement decreases. As the number of active initiators decreases, the frequency of measurement increases. In addition, the number of problem program task switches that occur between measurements can vary.

When the page measurement routine receives control from the task dispatcher immediately before a ready task is to be dispatched, it returns control to the task dispatcher without performing activity measurement if the next task to be dispatched is of higher priority than the communications task or a subtask of a non-problem program partition. If a main task is to be dispatched, the execution frequency value is decremented by one and tested. The execution frequency value determines the number of main task dispatches that must occur before activity measurement is performed.

The value to which the execution frequency is initialized is based on the number of initiators currently active. Each time an initiator is activated or deactivated, the execution frequency value in effect is changed by the start/stop routine to reflect a change in the number of active initiators. The page measurement routine initializes the execution frequency to the appropriate value, using a table of nine frequency values for from one to nine or more active initiators.

Except when the number of active initiators is one, the default execution frequency value is equal to the number of active initiators. For one active initiator, the default execution frequency value is 3. The execution frequency values to be used for various numbers of active initiators can also be established by the operator, using the PAGETUNE command.

When the execution frequency value is decremented, control is returned to the task dispatcher without performing activity measurement if the decremented value is greater than zero. If the decremented value is zero, activity measurement is performed if (1) the page supervisor is neither the task to be dispatched at this time nor the task last dispatched and (2) the number of page frames in the available page queue is greater than the low threshold value for the APC. After activity measurement has been performed, the execution frequency value is reinitialized, based on the number of active initiators, and control is returned to the task dispatcher.

The page measurement routine is called by the page supervisor wait routine when both of the following conditions exist: (1) the system is to enter the enabled wait state because no task is ready to execute and (2) no paging activity is occurring. Activity measurement is always performed at this time, regardless of the current execution frequency value. Once measurement is completed, the execution frequency value is reinitialized, based on the number of active initiators, and control is returned to the page supervisor wait routine.

The page measurement routine performs activity measurement as follows. All the RSPT entries from the reference level 1 queue, if any, are moved to the end of the level 0 queue. Then, the RSPT entries in all reference level queues except the level 0 queue are shifted to the next lowest reference level queue (all level 2 entries are placed in the level 1 queue, all level 3 entries are placed in the level 2 queue, all level N entries are placed in the level N-1 queue). Reference bits are not reset.

Once the level shifting is complete, the reference bit of each RSPT entry in the reference level 0 queue, up to a maximum of 30 entries, unless there is a partition in deactivated status, is inspected beginning with the first entry in the level 0 queue. Only 30 entries are inspected in order to limit the time required to process the level 0 queue. If the reference bit in the page frame associated with an RSPT entry is on, indicating the page was referenced during processing that occurred since frequency of reference was last measured, the RSPT entry is placed at the end of the level N queue and its associated reference bit is turned off. If the reference bit is off, the entry is left in the level 0 queue. Level 0 then contains entries only for pages that have not been referenced since the last measurement (if it contains fewer than 31 entries and no partition is in deactivated status).

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

When the page replacement algorithm receives a replenish request, it attempts to satisfy the request by placing the indicated number of page frames in the available page queue. Initially, only page frames contained in the reference level 0 queue are eligible to satisfy a replenish request.

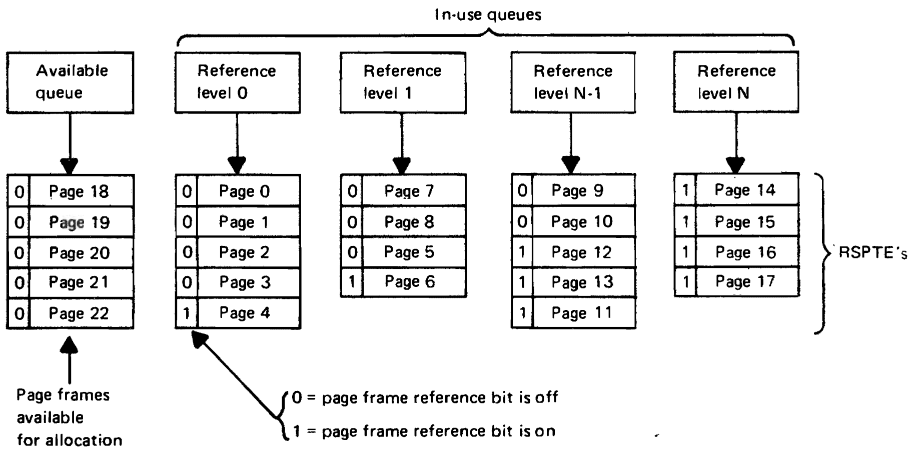
To determine the activity of the pages represented in the level 0 queue, the page replacement algorithm inspects the reference bits associated with the RSPT entries in this queue, starting with the first RSPT entry in the queue. If the reference bit is on for a page frame, its RSPT entry is placed at the end of the reference level N queue and the reference bit is turned off. If the reference bit is off, the change bit determines where the entry is placed. If the change bit is off, the entry is placed in the available page queue. If the change bit is on, the entry is placed in the appropriate page I/O device queue so that the contents of its associated page frame can be written out before the entry is placed in the available page queue.

Inspection of the level 0 queue RSPT entries continues until enough unreferenced page frames to satisfy the request are selected or until the entire level 0 queue has been searched. If the request is not satisfied by the search of the level 0 queue, the number of in-use queues searched thereafter, if any, is determined by the current location of the STOP line. Only queues after the level 0 queue up to the location of the STOP line are searched to satisfy a request.

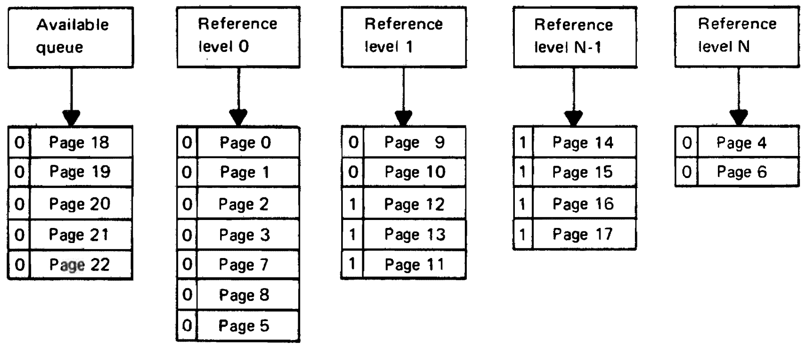
The STOP line is set by the start/stop routine in front of the level 1 queue as long as two or more initiators are active. If only one initiator is active, the STOP line is set after the level N (last) in-use queue. Thus, when more than one initiator are active, only the level 0 queue is searched to satisfy a real storage request. When only one initiator is active, all the in-use queues (level 0 to N) are searched.

Whenever an initiator is activated or deactivated, the new number of active initiators is inspected by the start/stop routine to determine whether the location of the STOP line should be moved. The location of the STOP line can be controlled by the operator via the PAGETUNE command instead of by the system, as discussed later.

Status of page queues and page frames at activity measurement time



Status of page queues and page frames after activity measurement time



Measurement Steps

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

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

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

Figure 90.35.2. Example of page activity measurement

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

Partition Deactivation and Reactivation

The primary function of the deactivation/reactivation module is to adjust the paging activity of the system to the availability of real storage so that thrashing is avoided. When real storage is totally allocated (no unreferenced page frames are in the level 0 queue in a multiprogramming environment) and a real storage request must be satisfied, the deactivation routine attempts to select active page frames to satisfy the request such that paging activity is reduced.

Similarly, the reactivation routine does not attempt to reactivate a partition until it determines that real storage is not being fully utilized so that reactivating a partition will not cause excessive paging to recur. The operator also has control over certain of the parameters that are used by the deactivation routine and can choose to reactivate a deactivated partition if necessary.

Note that the deactivation/reactivation module does not issue a message to the operator when a partition is deactivated or reactivated and no statistics regarding these two actions are maintained by the control program.

The partition deactivation routine attempts to suspend the processing of a pageable problem program task (mark it nondispatchable) so that the nonfixed real storage currently allocated to the partition in which it is executing can be released and made available for allocation. System task partitions and problem program partitions with a nonpageable job step in execution are not eligible for deactivation.

The eligibility for deactivation of each defined pageable problem program partition can be specified by the operator during system initialization and thereafter whenever a DEFINE command is issued. The operator can designate each problem program partition as eligible or not eligible for deactivation. If no specification is made by the operator, all partitions except P0 are eligible for deactivation. P0 is not eligible.

Only eligible pageable problem program partitions that (1) contain at least one dispatchable problem program task, (2) are not in a disabled state, (3) do not have system-must-complete-ENQ's outstanding, and (4) have not caused the system lock to be set are considered for deactivation. In addition, an eligible partition is not deactivated if it currently is the only dispatchable partition. When multiple problem program partitions satisfy the criteria for deactivation, the one with the lowest dispatching priority is chosen for deactivation.

When a partition is selected for deactivation, all the tasks in the partition are marked nondispatchable. The deactivation routine then searches all the in-use queues for page frames that are currently allocated to the deactivation partition. Since the in-use queues contain only the page frames that are allocated to nonfixed pages, page frames that contain fixed pages are not released when a partition is deactivated. The page frames found are placed in the level 0 in-use queue and their reference bit is turned off. A count of the number of page frames that were released by the deactivation routine is maintained for use by the reactivation routine.

The deactivation routine then determines whether a real time interval penalty factor should be assigned to the deactivated partition by calculating the amount of time that has elapsed since the last partition was deactivated. No penalty factor is assigned if more than 30 seconds have elapsed. Otherwise, a penalty factor of from 1 to 16 seconds is assigned to the deactivated partition. This penalty factor is used to prevent the occurrence of rapid deactivation/reactivation cycles.

After this assignment, control is returned to the page replacement algorithm, which again attempts to satisfy the real storage request. If deactivation of the partition did not release enough page frames to satisfy the request, the next lowest priority partition is deactivated if it is eligible.

If no partitions are eligible for deactivation or if deactivation fails to make the required number of page frames available, a determination of whether any deferred V=R allocation requests are pending is made. If such a request is pending, it is overridden, and any page frames conditionally assigned to the V=R request are made available. The operator is notified and can request cancellation of the nonpageable job.

If this procedure does not make enough page frames available, the task deactivation routine moves enough page frames from the in-use queues, starting at the beginning of the level 1 queue, to the level 0 queue to satisfy the real storage request without regard for their reference bit setting. The reference bit in these page frames is set to zero. The page replacement algorithm is then recalled to search the level 0 queue and place unreferenced page frames in the available queue.

The reactivation of deactivated partitions is normally performed by the partition reactivation routine when certain conditions of system activity or inactivity exist. However, the operator can also perform this function when necessary. The partition reactivation routine, which consists of a conditional reactivation portion and an unconditional reactivation portion, is entered when both of the following conditions exist: (1) the system is to enter the enabled wait state because no tasks are ready to execute and (2) no paging activity is in progress. The partition reactivation routine is entered after the page measurement routine performs its activity measurement function.

If the partition deactivation routine determines that no partitions are currently in deactivated status, it returns control to the task dispatcher and the system enters the enabled wait state. Otherwise, conditional partition reactivation is attempted. The highest priority deactivated partition is reactivated by the conditional reactivation routine if certain conditions are met. Only one deactivated partition can be reactivated per entry into the partition reactivation routine.

The highest priority deactivated partition is reactivated (all its tasks are marked dispatchable) when the following three conditions for conditional reactivation are satisfied in the sequence in which they are listed:

1. The real time interval penalty assigned to the deactivated partition by the partition deactivation routine, if any, has elapsed.

2. Another real time interval has elapsed during which paging activity met a satisfactory level. When the conditional deactivation routine determines that the real time interval penalty, if any, has elapsed, it establishes a real time interval

for reactivation based on the number of initiators currently active. Associated with each of these nine real time intervals is a page transmission value that indicates the maximum number of paging operations (total number of page-in and page-out operations) that can occur during the associated real time interval.

A table of system-defined real time intervals and associated page transmission values for from one to nine or more active initiators is used to determine the values assigned. The default real time intervals vary from a minimum of 0 to a maximum of 6 ms. The page transmission value associated with each real time interval varies from a minimum of 0 to a maximum of 16 depending on the amount of available real storage found during system initialization and the slowest direct access device that contains a page data set. The operator can change the values used for real time intervals and page transmissions by issuing the REACT system parameter during system initialization or the PAGETUNE command.

Once a real time interval and page transmission value are established, each time the conditional deactivation routine is entered thereafter, it determines whether the real time interval has elapsed and the number of paging operations that have occurred. If the number of paging operations exceeds the page transmission value in effect before the assigned real time interval expires, the real time interval is reestablished and the accumulated number of paging operations is reset to zero. However, if the real time interval expires and the number of paging operations that occurred in the interval is less than or equal to the page transmission value, the second reactivation condition is deemed to be satisfied.

3. Enough page frames must be potentially available to satisfy the real storage requirements of the partition at the time it was deactivated as determined by the reactivation count. Once the above two conditions have been met, the conditional reactivation routine determines whether the number of page frames in the available page queue plus the number of unreferenced page frames in the level 0 queue is equal to or greater than the reactivation count number of page frames for the deactivated partition plus the APC high threshold number of page frames. If so, the highest priority deactivated partition is reactivated.

Each time the conditional reactivation routine is entered and it cannot reactivate a partition, the unconditional reactivation routine is entered. This routine is designed to prevent the occurrence of a potentially long system wait condition because no deactivated partition can be reactivated under the rules of conditional reactivation. First the unconditional reactivation routine determines whether there are any pages in short-term fixed status. If there are, it determines whether all defined partitions are currently deactivated. If so, the highest priority deactivated partition is reactivated.

When the unconditional reactivation routine finds no pages in short-term fixed status, the indication is that no I/O is in progress. The unconditional reactivation routine enables the CPU for all I/O and all external interruptions. If no such interruption is received within one second after the enabling, the system is assumed to be in an inactive state and on the verge of entering a long or indefinite wait state. Therefore, the highest priority deactivated partition is reactivated without concern for the conditions that must be met for conditional reactivation. No reactivation is performed if an I/O or external interruption is received within one second.

The operator can cause a deactivated partition to be reactivated. This should be done only in cases in which the partition has been in deactivated status for a relatively long period of time and the output of the job in the deactivated partition is needed. The operator can determine whether a partition is deactivated by using the list function of the DEFINE command or issuing a DISPLAY ACTIVE command. Reactivation can be accomplished in several ways:

- The operator can enter the SR parameter for the deactivated partition in response to a DEFINE command to cause immediate reactivation of the partition. Note that if the SR parameter is specified for a partition that is not in deactivated status, the partition will become ineligible for deactivation until processing of the current job is completed. The partition then returns to its previously specified deactivation eligibility status.

- The operator can place the job queue (all input queues) in hold status. As executing jobs terminate, paging activity should be reduced, and thereby should cause the reactivation of deactivated partitions.

- The operator can stop an active partition, and this action should cause reactivation to occur as a result of reduced paging activity.

- The operator can cancel the job in the deactivated partition, stop the deactivated partition, and return the job to the job queue for initiation in another partition.

Careful consideration should be given to determining which partitions are to be ineligible for deactivation. Only those partitions in which critical jobs execute should be declared ineligible and at no time should all active partitions be marked ineligible for deactivation. In the latter situation, the page supervisor is effectively unable to prevent a thrashing condition. In addition, whenever the operator determines that a partition should be reactivated, he should make sure that there is at least one active partition that is marked eligible for deactivation.

The PAGETUNE command is provided to enable the operator to display and alter certain of the parameters that are used to control partition deactivation, page activity measurement, and partition reactivation. The PAGETUNE command is designed to be used only in multiprogramming environments, that is, when two or more partitions are active concurrently. The PAGETUNE command can be issued automatically during system initialization by including it in a CMDxxxxx parameter specification member in SYS1.PARMLIB. It can also be issued anytime during system operation. The new values are effective only for the duration of the current IPL or until another PAGETUNE command is issued.

The following operands can be specified in the PAGETUNE command to perform the functions indicated:

- DISPLAY - Displays the current values of the page management parameters that can be changed by the PAGETUNE command. It should be issued before any of these values are altered. The STOP suboperand causes the current location of the STOP line to be displayed. The PAGEMEAS suboperand causes the nine execution frequency values in effect (for one to nine or more active initiators) to be displayed. The REACT suboperand causes a display of the nine pairs of real time interval/page transmission values in effect (for one to nine or more active initiators). The STATUS suboperand displays the current number of page frames that are available for paging operations. This count excludes the page frames currently allocated to long-term fixed pages.

- STOP - This operand is used to set the location of the STOP line before a level 1 to 9 queue or after the level N queue. Although there can be up to twelve in-use queues, setting the stop line before the level 10 or 11 queue essentially has the same effect as setting it after the level N queue. Partition deactivation is effectively disabled when the STOP line is set after the level N queue. Note that any partitions that are in deactivated status when deactivation is disabled are not automatically reactivated at that time. The system STOP line settings can be reinstated as well through use of this operand.

- PAGEMEAS - This operand is used to alter one of more of the execution frequency values (for one to nine or more active initiators) that determine how frequently the page measurement routine is executed at task dispatch time. The execution frequency value specified for a given number of active initiators can vary from 0 to 99. If a zero frequency value is specified for all nine of these values, execution of the page measurement routine at task dispatch time is suspended. The default execution frequencies can also be reinstated using this operand.

- REACT - This operand is used to alter one or more of the nine pairs of real time interval/page transmission values that are used by the conditional reactivation routine. The real time interval can be a value from 1 to 9 seconds. The page transmission value can vary from 0 to 255. The system default values can also be reinstated using this operand. Note that the REACT system parameter can also be issued during system initialization to establish these values. If the REACT system parameter is specified, it causes all nine pairs of values to be changed. All the real time intervals are set to the value specified in the REACT parameter. All the page transmission values are set to zero. The REACT system parameter can be used, for example, to cause the partition reactivation routine to operate as it did in Release 1 of VS1.

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

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

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

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

If the required contiguous space is conditionally available (the area contains only page frames that are available, allocated but not fixed, or short-term fixed), the RSPT entries for the area are considered conditionally available and are marked for interception and allocation to the current V=R request. Available page frames in the area selected are allocated to the V=R request immediately and the residual allocation count, which indicates the number of additional page frames required to satisfy the request, is updated. As RSPT entries that are flagged as conditionally assigned are intercepted (when a PGRLSE macro is processed or when the available page queue is replenished, for example) and allocated to the V=R request, the residual count is decremented. When the count reaches zero, the V=R allocation request is satisfied.

Real Storage Release

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

EXTERNAL PAGE STORAGE MANAGEMENT

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

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

All page-in queues have priority over all page-out queues. The page-in queues for paging devices are arranged in priority sequence according to the virtual storage addresses mapped on the paging devices, and are followed by page-out queues arranged in the same priority sequence. That is, the page-in queue for the paging device whose page data set is assigned to the highest addressed paged virtual storage has the highest initiation priority. The page-in queue for the paging device assigned to the lowest addressed paged virtual storage (immediately above the V=R line) has the lowest initiation priority within the page-in queues. The page-out queue for the paging device that is to contain the highest addressed paged virtual storage is next in the total paging queue, etc.

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

The page I/O processor routine initiates paging I/O operations in the sequence indicated by the page I/O device queues. Page-in requests for a given page device with a movable arm have priority over page-out requests unless the requests can be merged into a single channel program because they refer to the same cylinder.

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

Rotational position sensing is supported when present for the paging device.

The number of requests in a paging channel program varies by paging device type. Up to three page requests are combined in 2314/2319 and 3340 channel programs to ensure that the channel is not busy for too long when servicing a paging I/O request to this device type. Up to five page requests are chained together in 3330-series channel programs. Up to six page requests are placed in a 2305 Model 2 channel program. Three nondedicated exposures per 2305 volume are used for paging so that three paging channel programs can be active concurrently. For performance reasons, page-out write operations are not verified by reexecution of the CCW list.

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

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

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

90:40 RECOVERY MANAGEMENT

RECOVERY MANAGEMENT SUPPORT

The routines included in recovery management support are machine check handler (MCH), channel check handler (CCH), alternate path retry (APR), and dynamic device reconfiguration (DDR). MCH and CCH are standard. APR is automatically included in the generated system when alternate channels are specified. DDR is automatically included unless specifically excluded by the user during system generation.

The facilities provided by the MCH and CCH routines are functionally equivalent to those supported by OS MFT RMS routines for System/370 models except for a few new features. MCH routines are structured so that a VS1 control program generated for one System/370 model can be executed on other System/370 models. When MCH recognizes that it is operating on a model other than the one for which it was specifically generated, error conditions that require processing by model-dependent routines are handled by model-independent routines.

The SECMODS system generation parameter can be used to include the model-dependent environment recording, editing, and printing (EREP) routines for each model on which a VS1 control program is to operate.

The MODE command is expanded in VS1 to enable the operator to establish full recording mode for single-bit control storage errors for Models 145 and 148.

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 or fixed SQA. 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 system task. If the affected system 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 RSPT entry for the affected page frame is placed in the malfunctioning page queue so that it will not be reassigned. This action is not taken if the error was intermittent.

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

Power Warning feature support, which is not provided for MFT, 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 VS1 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 all the paths to these two 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 VS1 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). These data sets cannot be placed on the system residence volume in VS1. 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, if present, ensures that a volume with a warn data set is swapped only to another 3330-series device that has an uninterruptible power supply.

For data security reasons, both warn data sets are always formatted during system initialization. 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 VS1 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 normal 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 VS1 supervisor so that system operations can continue. When a power disturbance is found to be nontransient, 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 VS1 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 system initialization, the restore routine is loaded into 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 VS1 and it supports the same functions as MFT OLTEP. In VS1, OLTEP can execute in a pageable partition that is a minimum of 128K to control the execution of OLT's. From 4K to 32K of real storage is fixed by OLTEP during the execution of certain OLTs that cannot operate in paged mode. A pageable partition of 192K minimum is required to execute the logout analysis program for a Model 155 II, 158, 165 II, or 168 under OLTEP.

PROBLEM DETERMINATION FACILITIES

Service Aids

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

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

- HMAPTFLE is used to apply PTFs to a system. This aid also produces the job control required to apply the fix. Independent component releases of VS1 are supported (not supported in MFT). HMAPTFLE is functionally superseded by the system modification program (SMP), which is called HMASMP.

- HMASMP (the system modification program) supersedes HMAPTFLE and is also provided for MFT (Releases 21.0, 21.6, 21.7, and 21.8). HMASMP provides an improved and more comprehensive method of applying system modifications (PTF's, component releases, and user modifications) to VS1 distribution and system libraries than does HMAPTFLE.

- HMBLIST replaces the IMAPTFLS and IMDMDMAP service aids of MFT and produces formatted listings that can be used for system serviceability and diagnostic purposes. It can print the following:

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

- HMASPZAP provides the capability of inspecting and modifying any load module in a partitioned data set (PDS) or any specific data record on a direct access device. It also can be used to dump an entire data set, a specific member in a PDS, or any portion of a data set on a direct access device. In VS1, HMASPZAP can be dynamically invoked from an executing problem program via a CALL, LINK, XCTL, or ATTACH macro instruction (not supported in MFT).

- IMCJOBQD can be used to print the entire or selected contents of the resident job list and SYS1.SYSJOBQE, entire contents of the SYS1.SYSWADS data set, and entire contents of all or selected SWADS after a system failure occurs. SWA's cannot be printed using IMCJOBQD. An SVC dump of real storage includes the contents of any active SWA's. An advantage of this service routine over IMCOSJQD, which is discussed below, is that it preserves the status of the system queues and all of the resident job list at the time of a system failure.

  In VS1, the IMCJOBQD service routine is enhanced to allow the operator to selectively print the contents of the local and remote (RES) output queues on a user basis. The new QID parameter can be specified to indicate the local or remote user for which IMCOSJQD is to list the SYSOUT data sets that are queued. For example, the queue records for all SYSOUT data sets, only the SYSOUT data sets of the specified SYSOUT class, or all the SYSOUT data sets in held status in the output queue for one or more remote users can be printed. The queue records for the input jobs queued for specific remote users or the jobs queued in held status in the input queue for all remote users can be printed as well.

- IMCOSJQD provides the same printing functions for job queues and scheduler control blocks as the IMCJOBQD standalone program. However, IMCOSJQD operates as a problem program under VS1 control (concurrently with other problem programs, if desired), using standard access methods, instead of as a standalone program. Therefore, IMCOSJQD can be executed without the necessity of a stop and re-IPL of the system, as is required when IMCJOBQD is used. The other advantage of IMCOSJQD over IMCJOBQD is that it enables the operator to re-IPL using different volumes for SYS1.SYSJOBQE and SYS1.SYSWADS and execute IMCOSJQD to dump the previous SYS1.SYSJOBQE and SYS1.SYSWADS data set.

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

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

  When the dump is written to a 3800 Printing Subsystem, the CHARS=DUMP option can be specified to obtain high-density lines of 64 bytes instead of 32 bytes and the FCB=STD3 option can be specified to obtain high-density pages of 80 lines instead of 55 lines. The CHARS=DUMP option applies only to the printing of storage and not to the control block portion of the dump.

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

- IFCEREP0 formats and prints records contained in SYS1.LOGREC and creates a history tape, if desired, as in MFT. In VS1, a new SPOT CHECK report can be printed from the records in the SYS1.LOGREC data set or an accumulation (history) data set. The SPOT CHECK report lists all the I/O devices that had a permanent error during the time period covered by the error records in the input data set. The number of errors that occurred for each day in the time period (but not the type) is listed in chronological sequence for each I/O device. The listing represents a summarization of the existing OBR and SDR records for each device. The report also lists the total number of each type of error record (MCH, CCH, OBR, EOD, etc.) that exists in the input data set for the time period.

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

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

The trace EDIT function of GTF is a part of the HMDPRDMP service aid and is invoked as a problem program via job control. A minimum 64K pageable partition is required for its execution. The trace EDIT function of VS1 will format only the trace data produced in a VS1 environment. It will not format data traced using GTF in VS2, MFT, or MVT environments. However, MFT programs that use the GTRACE macro can be executed under VS1 control without modification. If user-written EDIT exit routines are being used in MFT, they may require modification for operation in a VS1 environment because of differences in the format of trace data for system events.

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

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

Dynamic Support System (DSS)

The dynamic support system is a general purpose debugging tool that is designed to help locate and temporarily repair a failure in most components of the VS1 control program. DSS uses program event recording hardware in its interface with the operational VS1 operating system. DSS is designed to be used by authorized personnel, such as an IBM FE Programming Systems Representative. Note that DSS does not support Models 138 and 148.

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

The DSS user can:

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

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

90:45 LANGUAGE TRANSLATORS, SERVICE PROGRAMS, AND EMULATORS

SYSTEM ASSEMBLER

The System Assembler is a standard component of VS1 and is the same assembler provided in VS2. It is the only language translator that is a standard component of VS1. Program product and Type I language translators that are to be used with VS1 must be obtained and added to the VS1 system. The System Assembler offers the same functions as OS Assembler F and many enhancements, including improved diagnostics and extended language capabilities. The System Assembler is compatible with OS Assemblers E and F, with a few minor exceptions (see OS/VS System Assembler Language, GC33-4010). The System Assembler is a compatible subset of Assembler H.

The System Assembler supports all the new standard and optional System/370 instructions. The System Assembler and the OS Assembler H (Release 5) program product are the only OS Assembler programs that support all the following System/370 instructions:

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

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

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

LINKAGE EDITOR

The VS1 Linkage Editor program is a standard component of VS1 (and VS2). It can also operate under OS MFT and MVT. A minimum pageable partition of 64K is required for its operation under VS1; however, a partition of 192K bytes or more is recommended. The performance of the linkage editor increases with the availability of additional virtual storage. A partition as large in size as is practical within a given VS1 system (up to a maximum of 999K bytes) should be used to achieve the best performance from the linkage editor.

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

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

The other new feature of the linkage editor is the ability to specify which control sections and common areas of a load module are to be aligned on a 2K or 4K page boundary in virtual storage (via an ALIGN statement). This new facility, like CSECT reordering, can be used to minimize page faults.

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

UTILITIES

The same utilities that are provided in MFT are available in VS1 and are updated to support I/O devices supported by VS1 but not MFT. In addition, the Analysis Program-1 (AP-1) is supported in VS1.

The IEBCOPY system utility is enhanced to allow a partitioned data set to be unloaded to a removable volume (tape or disk) and later reloaded to the same or a different type disk volume. This utility is to be used during a system generation to place distribution libraries supplied with the VS1 starter system on direct access volumes. The starter system, therefore, is independent of the direct access devices that will be used during a system generation.

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

<u>Analysis Program-1 (AP-1)</u>

AP-1 is a problem determination utility program for 3344 and 3350 Direct Access Storage, which do not have removable data volumes. When errors occur on a 3344 or 3350, AP-1 can be used to determine whether the drive is failing or a problem exists on a recording disk. AP-1 operates as a job step under VS1 control.

AP-1 performs a drive test and then, optionally, a data verification test to determine the source of the error. The drive test exercises the drive by issuing SEEK, READ, and WRITE commands. Diagnostic messages are issued if a failure occurs and the customer engineer should be notified.

The data verification test issues a read command with the skip bit on to each data track in the volume, which prevents the data from being read into processor storage. When an uncorrectable error occurs, the appropriate ERP is invoked to attempt to correct the error. If the error is permanent, diagnostic messages are issued and installation recovery procedures should be initiated. AP-1 does not attempt any recovery.

The AP-1 program resides in SYS1.LINKLIB. It can be invoked using standard job control (a TESTDD DD statement identifies the volume to be tested) or an AP-1 procedure can be written that enables the operator to invoke the AP-1 program using a START command.

SORT/MERGE PROGRAMS

The OS Sort/Merge (5734-SM1) and OS/VS Sort/Merge (5740-SM1) program products can be used in a VS1 environment for sorting and merging operations. Both programs can operate in the minimum size virtual partition (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 for this function by the OS Sort/Merge. The OS/VS Sort/Merge supports up to 100 direct access devices for intermediate storage, the specification of secondary space allocation for an intermediate file, and noncontiguous intermediate work files, when the new disk sorting technique is used. Both sort/merge 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 in the partition.

- 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, entry-sequenced, and relative record 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 MFT control will also operate under VS1 control:

- 1401/1440/1460 emulator Version 2 - operates on a Model 135, 138, 145, 148, 155 II, or 158

- 1410/7010 emulator - operates on a Model 145, 148, 155 II, or 158

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

- 7080 emulator - operates on a Model 165 II or 168

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

- DOS emulator Version 2 - operates on a Model 135, 138, 145, 148, 155 II, or 158 to emulate a DOS Version 3 or 4 system (Releases 25, 26, and 27)

The functions supported by the integrated emulator programs listed above when they operate under VS1 are identical to the functions supported by these emulators when they operate under MFT except that VS1 does not support the 2311 as an emulation device. These functions are discussed in appropriate system library publications and in Section 40 of the following System/370 guides:

- A Guide to the IBM System/370 Model 135 (GC20-1738)
- 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 VS1 (and VS2) but not under MFT (or MVT). 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 now 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 VS1 on Models 135, 138, 145, 148, 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.

Version 3 of the DOS emulator program operates under OS/VS1 and OS/VS2 (SVS and MVS) control but not under OS MFT (or MVT) control. Version 3 supports several new features as well as all the functions of Version 2. In addition to DOS Versions 3 and 4 (Releases 25, 26, and 27), Version 3 of the DOS emulator can emulate DOS/VS systems (Releases 28 to 32).

The new features of these two releases of DOS/VS are emulated, such as support of a virtual storage environment, five partitions, the relocating loader, shared virtual area, generic I/O device assignment, 3340/3344 and 3350 disk storage (as a restricted, shared, or substituted device), block multiplexing, rotational position sensing, and VSAM. The 3350 is supported in 3330-compatibility mode only.

Version 3 of the DOS emulator also provides the following new facilities:

- Device substitution. This facility enables a DOS or DOS/VS program that accesses one type of direct access device to access another type of (substituted) direct access device when operating under the DOS emulator. Program modification is not required. However, the DOS system being emulated must support the substituted device (3340 cannot be used as a substituted device for programs that operate under an emulated DOS Version 3 or 4 system, for example).

  Programs that access a 2311 can execute using a 2314, 3330-series, virtual 3330 volume of a 3850 Mass Storage System, or 3340 device. Programs that access a 2314 can execute using a 3330-series device, virtual 3330 volume, or 3340 device. Programs that access a 3330-series device can execute using a 2314 device, virtual 3330 volume, or 3340 device. Programs that access a 3340 can execute using a 2314 device, 3330-series device, or virtual 3330 volume.

- Improvements in operator communication. The following enhancements are provided: easier end-of-block specification in response to a message from an emulated DOS or DOS/VS system, formatting of DOS or DOS/VS messages to distinguish them from OS/VS messages and messages from all other concurrently operating DOS emulator partitions, specification of the alternate DOS/VS supervisor to be used when more than one are present on a DOS/VS SYSRES volume via an emulator control statement instead of by operator intervention during the DOS/VS IPL, and support of user-specified message routing codes instead of only one.

- Support of local 2260 Display System stations and local and remote 3270 Information Display System stations that are accessed using DOS or DOS/VS BTAM.

- Support of the 3540 Diskette Input/Output Unit as an OS/VS device. DOS/VS programs that use the 3540 will not run under the DOS emulator if they attempt to use a file without issuing an OPEN or OPENR macro, issue EXCP macros, use access methods not supported by DOS/VS or modified DOS/VS access methods, or attempt to use a 3540 file created in the same emulator job.

The following restrictions apply to Version 3 of the DOS emulator in addition to the Version 2 restrictions:

- DOS/VS SDAIDS and OLTEP are not supported.
- VSAM is supported only on restricted DOS/VS volumes.
- DOS/VS programs that use the page fault appendage facility are not supported.
- Device substitution is supported only for nonrestricted direct access volumes. In addition, when device substitution is used, the emulated DOS or DOS/VS program must not contain any device-dependent coding other than the DTFs for the devices using the device substitution facility, and the initial block size in the DTF must be less than the track capacity of the substituted device.

In general, the I/O devices that are supported by Version 3 of the DOS emulator operating under VS1 are those that are supported by both the DOS or DOS/VS and the OS/VS1 releases being used. The following I/O devices, which are supported by DOS/VS Releases 29 and up, are not supported by the DOS emulator operating in a VS1 environment:

- 1017 Paper Tape Reader
- 1018 Paper Tape Punch
- 1255, 1259, and 1419 Magnetic Character Readers
- 1287 and 1288 Optical Character Readers in document mode if response times are required for pocket selection
- 2311 Disk Storage Drive
- 2321 Data Cell Drive
- 2560 Multifunction Card Machine
- 3203 Printer
- 3504 Card Reader
- 3881 Optical Mark Reader
- 5203 Printer
- 5213 Console Printer
- 5425 Multifunction Card Unit
- 7770 Audio Response Unit

The minimum virtual storage size for a Version 3 emulator program is 48K bytes. This minimum requirement includes 42K bytes of basic emulator program and 6K bytes for emulator control blocks for up to ten I/O devices per job step. For each additional device above ten, another 300 bytes is required. The size of the emulator program is increased when it supports the following facilities:

- Staged devices (2.5K)
- Shared and/or substituted direct access devices (8K)
- OS ISAM files (8K)
- Emulator service aids (20K)

All the integrated emulator programs for VS1 are pageable. An emulator program generated to operate on a Model 135, 145, 155, or 165 under OS MFT control can operate on a Model 135 or 138, 145 or 148, 158/155 II, or 168/165 II, respectively, under VS1 control. Emulator regeneration is not required. The integrated emulator programs, although SCPs, are not shipped with VS1. They must be ordered separately.

90:50 VM/370 HANDSHAKING OPTION

FUNCTIONS

VM/370 Handshaking support is an optional feature for OS/VS1 (Releases 4 and up). This support is designed to improve the performance of a VS1 system when it operates in a virtual machine under the control of a VM/370 control program that also includes handshaking support (Release 2 PLC 13 and later releases). A VS1 system with handshaking support can also execute in a real machine without any special operator action. In this situation, handshaking support in the VS1 control program is not activated during initialization processing.

Handshaking support consists of a communication path between the VS1 control program and the control program (CP) component of VM/370. This communication path enables a VS1 control program to determine during system initialization whether it is operating in a virtual machine under the control of a CP with handshaking support. If so, the VS1 control program eliminates certain functions it normally performs that are redundant or inefficient in a virtual machine environment and performs other functions that are designed to make it operate more efficiently in a virtual machine. Similarly, handshaking support enables CP to know that it is VS1 that is operating in a given virtual machine and optionally to provide an improved page fault handling capability.

Any VS1 system determines whether it is executing in a virtual machine during system initialization by issuing the STORE CPU ID instruction. A version code of X'FF' returned to the VS1 control program indicates operation in a virtual machine. A VS1 control program with handshaking support then issues a DIAGNOSE instruction (code X'00') to obtain the VM/370 extended identification code. If CP returns an extended identification code, handshaking support is indicated. Failure to return such a code indicates to the VS1 control program that this CP does not support handshaking. When the VS1 control program determines it is operating under the control of a CP with handshaking support, it activates its own handshaking support during system initialization.

The handshaking support in VS1 and VM/370 provides the following capabilities:

- The closing of CP spool files by VS1 at the end of each job step
- Improved handling of page faults for the VS1 virtual machine by CP
- A nonpaged mode of operation for a VS1 system executing in a virtual machine
- Nonexecution by the VS1 control program of certain routines and privileged instructions that are inefficient or unnecessary in a virtual machine environment
- A BTAM AUTOPOLL enhancement

CLOSING CP SPOOL FILES

At the end of each job step and at end of job, VS1 uses the DIAGNOSE instruction to issue a CP CLOSE command for each output data set for the partition that is being written to a CP spool file. This action eliminates the need for operator intervention, or other programmed means, to close CP spool files at the appropriate times.

PAGE FAULT HANDLING

The improved page fault handling capability for a VS1 system operating in a virtual machine is enabled when the virtual machine operator issues a SET PAGEX ON CP command. (The QUERY SET CP command can be issued to determine the status of the page fault handling capability.) If this command is not issued, page faults are handled as usual for the VS1 virtual machine. That is, when a page fault occurs in the VS1 virtual machine, CP places the entire virtual machine in the wait state until CP processing of the page fault is completed (the required page-in is completed). This means the VS1 dispatcher cannot dispatch another ready task and results in the highest priority partition receiving most of the execution time made available to the VS1 virtual machine.

When the SET PAGEX ON command is in effect for a VS1 virtual machine and a page fault occurs, CP does not place the VS1 virtual machine in the page and execution wait state. Instead, CP stores the virtual storage address that caused the page fault in location X'90' of virtual storage in the VS1 virtual machine and presents a program interruption with interruption code X'14' to the VS1 virtual machine.

When the VS1 virtual machine recognizes the X'14' interruption code, it places the task that caused the page fault in the wait state and attempts to dispatch another ready VS1 task. When CP has processed the VS1 page fault, it again presents a code X'14' program interruption to the VS1 virtual machine after turning on a completion bit in the virtual storage address field to indicate page fault processing has been completed. The VS1 control program then posts the affected task ready.

The improved page fault handling capability can be used to increase the amount of multiprogramming that can actually occur in a VS1 virtual machine.

NONPAGED MODE IN A VIRTUAL MACHINE

Nonpaged mode for a VS1 system executing in a virtual machine is an option for VS1 virtual machines that have a minimum of 1024K and a maximum of 16,320K of virtual storage. Virtual storage size must also be a multiple of 64K. Nonpaged mode is invoked if the amount of virtual storage the VS1 system is to support is equal to the amount of virtual storage in the virtual machine, which to the VS1 system is real storage (that is, as far as VS1 is concerned, virtual storage size equals real storage size).

When nonpaged mode of operation is in effect in a VS1 virtual machine, the following occurs:

- VS1 virtual storage is mapped to virtual machine virtual storage (real storage to the VS1 supervisor) on a one-for-one basis and paging by the VS1 supervisor is not performed. VS1 marks all VS1 virtual storage pages fixed during system initialization and the page file is not opened. Use of the privileged instructions LOAD REAL ADDRESS and RESET REFERENCE BIT (very frequently used instructions in a paged mode of operation) is eliminated.

- Page tables are not built in fixed PQA for the defined partitions. All page tables are built in SQA.

- The VS1 I/O supervisor treats all I/O requests as if they came from a nonpaged partition and thus eliminates channel program translation and indirect data address list construction.

- The START I/O appendage used by JES issues LOAD ADDRESS instructions (which are not privileged) instead of LOAD REAL ADDRESS instructions (which are privileged) and does not have to construct indirect data address lists since its I/O operations are performed in nonpaged mode as far as VS1 is concerned.

The nonpaged mode of operation eliminates duplicate paging support by VS1 and enables VS1 to issue fewer privileged instructions.

MISCELLANEOUS ENHANCEMENTS

When a VS1 system with handshaking support executes in a virtual machine, the VS1 control program eliminates certain frequently used procedures and instructions that are redundant or less efficient in a virtual machine environment, as follows:

- The INSERT STORAGE KEY privileged instruction is not issued. Instead VS1 maintains a table of the protect key value assigned to each virtual storage page.
- Seek separation support (which is automatically provided by CP) for 2314/2319 direct access storage facilities is not performed.
- Execution of the ENABLE/DISABLE sequence in the I/O supervisor is eliminated.
- The issuing of a TEST CHANNEL instruction before the issuing of each START I/O instruction is eliminated.

The enhancements above result in a reduction in the CPU time used by the VS1 control program and the CPU time used by CP to handle privileged instructions issued by the VS1 control program. Reducing the total number of privileged instructions issued in a virtual machine is one of the primary ways to increase performance in a virtual machine environment.

BTAM AUTOPOLL

The BTAM AUTOPOLL facility is designed to improve the performance of BTAM in a virtual machine. VS1 informs VM/370 via a DIAGNOSE instruction when a BTAM AUTOPOLL virtual CCW string is modified. This eliminates the need for this checking in CP. Operation of BTAM AUTOPOLL in VM/370 is controlled by an operator command (SET AUTOPOLL ON/OFF). For additional information see _Virtual Machine Facility/370 Features Supplement_ (GC20-1757).

90:55 OS/VS1 ASSIST SUPPORT

Support of the OS/VS1 Extended Control Program Support (ECPS) feature for the Model 158 and the VS1 hardware assist function of the Extended Control Program Support feature on Models 135 (Model 3), 145 (Model 3), 138, and 148 is optional in VS1. During system generation the ECPS parameter of the CTRLPROG macro can be specified to indicate the CPU model to be used and the type of support desired.

The following options are available:

- ECPS parameter is not specified. A nucleus that does not support the VS1 assist function is supported (the default option).

- ECPS=NO is specified. A nucleus that supports the VS1 assist function for the specified model is generated. It includes the VS1 assist privileged instructions only and not the supervisor code these instructions replace. This option provides the best performance improvement.

  When this option is selected, the VS1 control program can execute only on a system that has the VS1 assist function or OS/VS1 ECPS feature installed. Specifically, when ECPS=138R,NO or ECPS=148R,NO is specified, the VS1 control program can execute only on a Model 135-3, 138, 145-3, or 148. When ECPS=158R,NO is specificed, the VS1 operating system can execute on a Model 158 within the OS/VS1 Extended Control Program Support feature installed or a Model 135-3, 138, 145-3, or 148. In the latter case, the VS1 operating system uses only the subset of the VS1 hardware assist function that is included in the OS/VS1 ECPS feature for the Model 158.

  If a VS1 supervisor with only the VS1 assist instructions is IPLed on a system that does not contain a VS1 assist feature, the system enters the wait state with a wait state code that identifies the problem. A VS1 supervisor with VS1 assist support issues a VS1 assist privileged instruction to determine whether the function is present.

- ECPS=YES is specified. A nucleus that supports the VS1 assist function for the specified model is generated. It includes both the VS1 assist instructions and the supervisor code they replace. Selection of this option makes the VS1 control program capable of executing on systems with and without the VS1 assist function installed.

  Note that when ECPS=158R,YES is specified and the generated OS/VS1 control program operates on a Model 135-3, 138, 145-3, or 148, only the subset of the VS1 hardware assist function that is included in the OS/VS1 ECPS feature for the Model 158 is utilized. This subset capability is also utilized when when ECPS=138R,YES or ECPS=148R,YES was specified for a VS1 operating system and it is executed on a Model 158 with the OS/VS1 ECPS feature installed.

  This option provides a lesser performance improvement because more fixed processor storage is required for the resident nucleus and additional instructions must be executed during system operation to determine whether the VS1 assist instruction or corresponding supervisor routine is to be used. (A bit is set in the CVT that must be tested.)

When a VS1 supervisor has both the VS1 assist instructions and the corresponding supervisor routines, the operator has the option of determining whether the VS1 assist instructions or the supervisor routines are to be used for this IPL. A message is issued during system initialization to enable the operator to make the choice.

The VS1 assist support can be utilized when VS1 is operating in virtual machines under the control of a VM/370 system, whether or not the VM/370 hardware assist function for Models 135-3, 138, 145-3, or 148 or the Virtual Machine Assist feature for the Model 158 is also being utilized.

90:60 OS MFT TO OS/VS1 TRANSITION

VS1 is designed to be upward compatible with MFT as of Release 21.8 and, therefore, migration from MFT to VS1 should involve minimal conversion effort. Some additional education of installation personnel is required. For the most part, this involves their becoming knowledgeable about the additional facilities and new environment offered by VS1.

System programmers must become acquainted with new interfaces to VS1 (SMF exits and JES reader and writer procedures, for example). Operators must learn how to use the new operater commands (WRITER, PAGETUNE, and if RES is used, the new RES commands), and how to respond to new system messages, such as those related to paging and spool devices. Application programmers should learn how to use program structuring techniques that are designed to improve system performance in a paging environment. System designers must become familiar with the new factors that affect system performance in a VS1 environment so that the system can be designed and operated in a manner that will achieve the results desired.

Once the VS1 environment to be supported has been determined, a system generation must be performed. A VS1 system control program is generated via a two-stage procedure, which is, in function, much like that required to generate an MFT control program. The system generation macros used to describe the desired control program are identical for MFT and VS1 for like functions. Some of the macros and parameters used in MFT are not required in VS1, while new macros are provided to describe additional or different functions of VS1 (JES and page devices, for example).

As for MFT, a complete, nucleus-only, or I/O-device-only VS1 generation can be performed. Processor-only generations, supported in MFT, are not supported in VS1. All OS program products and Type I and Type II components that are to be used with the generated VS1 SCP must be added to the VS1 operating system after its generation. Since processor generations for language translators cannot be performed using a VS1 system, they must be done using OS MFT or MVT.

The VS1 starter system (available for 2314/2319, any 3330-series model, 3340, or 3350 residence) operates on any System/370 model with real storage of 160K or more that has dynamic address translation, one nine-track tape unit, one SYSIN device, one SYSOUT punch device, one SYSOUT print device, one console device, and four 2314/2319, three 3330-series, two 3350, three 3340 (Model 70/70F), or four 3340 (Model 35) direct access devices.

The VS1 starter system can be used only to generate a VS1 control program and is required only for the first generation. Thereafter, a generated VS1 system (Release 3.1 or higher if CS=YES is specified in the CENPROCS macro, otherwise Release 3 or higher) can be used. If the existing VS1 system is not to be modified, the system generation can be performed concurrently with other executing jobs.

The generated VS1 system can operate on any System/370 Model 135, 138, 145, 148, 158, 155 II, 168, or 165 II that has the hardware features and I/O devices required by the control program. The SECMODS parameter should be specified on the CENPROCS macro at system generation to cause inclusion in the generated VS1 operating system of the model-dependent EREP for the secondary models on which the VS1 control program is to be run, if any.

A new feature of the VS1 generation process is the installation verification procedure (IVP), which is designed to be performed after the VS1 control program is generated. The IVP involves executing an

IBM-supplied job stream (maintained in the SYS1.SAMPLIB data set) under control of the generated VS1 operating system. The function of the IVP is to exercise the generated SCP system components to the degree that general operation of the VS1 operating system and support of the system hardware configuration specified is assured.

The IVP program also displays the amount of virtual storage used by the following: supervisor nucleus, fixed SQA, virtual=real area, partitions, RTAM, VTAM, job entry subsystem, and the pageable supervisor.

If VTAM was included in the generated system for the first time, the control programs that are to execute in the 3704/3705 Communications Controllers controlled by VTAM must be generated using a two-stage generation procedure. The network VTAM is to control must be defined and placed in the VTAM definition library (SYS1.VTAMLST) using the IEBUPDTE utility. Optionally, VTAM START parameters can be placed in SYS1.VTAMLST as well.

When support of an industry subsystem (3600, 3650, 3660, or 3790) is to be used, the required independent release must be obtained and a generation process performed. During this generation, the active load modules for the subsystem support services program are also generated. While industry subsystem support is standard in VS1, IND=YES must be specified during system generation in order to use the support as this parameter causes VSAM, VTAM, and at least three megabytes of virtual storage to be included in the VS1 control program and enables the subsystem support services program to be included.

Existing user-written programs that operate under MFT on a System/370 model must be modified for correct operation under VS1 if they do any of the following (otherwise, existing user-written executable programs, 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.

- Access SYSIN or SYSOUT data sets using the EXCP macro. BSAM or QSAM must be used. In addition, DSCBs and user labels are not supported for SYSIN and SYSOUT spool data sets. Problem programs that handle certain conditions for spooled output data sets (such as a printer overflow condition) by checking bits in the unit record UCB will not operate properly since JES does not create these UCB's for the spooled output data sets it writes.

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

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

- Use QTAM to support teleprocessing operations. These programs must be altered to use TCAM (or VTAM) since QTAM is not supported in VS1. Minimal effort is required for this modification. (See <u>OS/VS TCAM</u>

Programmer's Guide, 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 CCWs) or by executing instructions, if the program is to be run in a pageable partition under VS1. Program modification is not required if such programs operate in nonpaged mode under VS1 control. This situation can apply to 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 OS/VS1 Data Management for System Programmers, GC26-3837, for a discussion of how to include a START I/O appendage in a program with channel programs that are dynamically modified via a PCI appendage.)

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

- Modify the task I/O table (TIOT). In VS1, the TIOT is in pageable PQA (subpool 255) within a partition so that it cannot be modified by the problem program.

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

- Programs that issue the SET STORAGE KEY (SSK) or the INSERT STORAGE KEY (ISK) instruction should be inspected to determine whether implementation of a seven-bit protect key instead of a five-bit protect key affects the processing being performed. If the 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 use the teleprocessing facilities of this language translator must be recompiled and relink-edited since PL/I F uses QTAM, which is not supported in VS1. The PL/I Checkout or PL/I Optimizing Compiler, which use TCAM, should be used for these programs.

- 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. TCAM message processing programs that use the ICOPY, TCOPY, QCOPY, or TCHNG macro must be reassembled and link-edited. TCAM message processing programs that do not use any of these macros must be relink-edited.

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

- User-written reader procedures (PARM field) must be modified to conform to VS1 format.

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

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

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

If desired, the structure of existing user-written MFT programs can be modified to minimize the occurrence of page faults and the use of real storage (as discussed in Section 15:15 or 30:15 of the base publication of which this supplement is a part). Such modification may improve system performance but is not required to enable existing programs (load modules) to operate correctly in a VS1 environment.

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

- A Guide to the IBM System/370 Model 135 (GC20-1738)
- A Guide to the IBM System/370 Model 138 (GC20-1785)
- A Guide to the IBM System/370 Model 145 (GC20-1734)
- A Guide to the IBM System/370 Model 148 (GC20-1784)
- A Guide to the IBM System/370 Model 155 (GC20-1729)
- A Guide to the IBM System/370 Model 158 for System/360 Users (GC20-1781)
- A Guide to the IBM System/370 Model 165 (GC20-1730)
- A Guide to the IBM System/370 Model 168 for System/360 Users (GC20-1787)

90:65 SUMMARY OF ADVANTAGES

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

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

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

Improved Job Scheduling

- Small partition scheduling and transient readers and writers are eliminated.

- Job queue contention is reduced by the implementation of SYS1.SYSWADS, SWADS, SWA's, and a resident job list. The time required to access scheduler data can be further reduced by the use of SWA's.

- Dedicated work files for initiators are supported and can be used to eliminate allocation and deallocation time for temporary disk data sets.

- A job can be placed in held status in the input queue during initiation when data sets it requires are not available.

- An initiator can handle up to 15 job classes instead of a maximum of 3, and 36 job classes are supported instead of only 15. The installation-specified selection parameters facility enables the user to specify the processing characteristics of a job and have the job scheduler assign the appropriate class and/or priority.

- More readers and writers can be active concurrently (the limitation of 3 readers and 36 writers is removed).

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

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

Operational Enhancements

- The time required to initialize a VS1 system without or with changes to previously specified parameters and automatic commands can be significantly reduced by use of the FASTNIP and automated system initialization functions, respectively.
- Significant new operator control over system output (SYSOUT data sets) processing is provided by the WRITER command. The PAGETUNE command enables the operator to modify the paging algorithm if this is necessary for better performance. The functions provided by several commands that are also available in MFT have been expanded in VS1 to give the operator more control or to reduce the amount of keyboarding required to control system operation.
- The operator is relieved of most real storage management functions (such as changing partition sizes and altering partition types for the purpose of managing real storage).
- The JES configuration can be modified during system initialization and does not require another system generation.
- The operator need not keep track of reader and writer partitions.
- A reader handling a card SYSIN device remains active when end of file occurs on the card reader (so the operator need not restart the reader from the console each time the card reader runs out of cards).
- High-priority jobs can be handled more easily. A high-priority partition can be established in virtual storage that is used only for these jobs. While this partition requires dedicated virtual storage (and, therefore, external page storage), real storage (except that required for fixed PQA) is required for the high-priority partition only when a job step is active in the partition.
- Remote users entering jobs via RES use standard OS commands instead of a special job entry control language (as is used in RJE) and a job can be submitted remotely or locally without changing job control or operator commands.
- The VS1 starter system is independent of the direct access device types to be used during a system generation.

Improved System Integrity and Availability

- More control blocks (specifically, those in a problem program partition) are protected from accidental or intentional modification by a problem program.
- Loss of an ABEND dump because of the lack of available storage in a partition can be eliminated. (Partitions can be made large enough to ensure the availability of enough virtual storage to perform

ABEND dump processing and a 12K dump area is provided to be used when a partition is too small.)

- Total system terminations that result from a lack of available SQA space are reduced because the fixed SQA requirement is reduced by the implementation of pageable SQA, fixed SQA is now dynamically expandable, and two page frames are held in reserve for allocation to fixed SQA and fixed PQA.
- Fetch protection as well as store protection is available for all problem program partitions.
- The new authorized program facility is supported to prevent unauthorized use of routines that are identified as having restricted access.
- Extended DEB validity checking is available to ensure that data sets are accessed only by programs that are entitled to access the data.
- A module that checks for missing channel-end and I/O-device-end interruptions during system operation is provided to prevent system waits, indefinite job step waits, and job step cancellations because of an uncompleted I/O operation.

Improved Utilization of Real Storage

- Inefficient use of real storage caused by unused storage within defined partitions and/or residence of inactive portions of a program is minimized. Unused virtual storage in a pageable partition does not have real storage assigned, and real storage allocated to inactive pages of a program is released and allocated to active pages when necessary.
- JES and RES are pageable so that during any time interval they use only the amount of real storage required to handle the current activity. The operator need not perform any function to make real storage assigned to inactive readers or writers available for allocation to other programs.
- The amount of real storage used by reentrant routines (such as SVC's and access methods) made resident in the pageable supervisor area is automatically increased and decreased based on the activity of these routines. The most active modules at any given time will tend to remain resident in real storage without the necessity of preplanning on the part of system designers.
- The amount of storage allocated to fixed SQA dynamically expands and contracts as required. SQA size cannot be varied during processing in MFT.
- Dynamic real storage management is provided for all programs that operate in paged mode in a VS1 environment, regardless of the language in which they are written. Dynamic serial program structure implemented via the use of LINK, LOAD, and XCTL macros and dynamic storage allocation supported via GETMAIN and FREEMAIN macros, all of which are supported by the Assembler Language in MFT, are not supported by all high-level languages.
- The practice of leaving unused real storage between the end of the resident control program and the lowest priority partition in order to leave room for control program expansion can be avoided.

Performance Enhancements

- Job scheduling improvements (as listed previously) are provided and initiator modules are reorganized to perform job scheduling more efficiently.

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

- A new I/O load-balancing algorithm is available to allocate tape and disk I/O devices such 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 dynamic dispatching partitions) on the basis of their changing operational characteristics (more CPU-oriented or I/O-oriented) rather than according to partition priorities.

- Extended timer support of the CPU timer and clock comparator reduces timer supervisor processing time and improves the repeatability of accounting data.

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

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

- SVC area size is increased to 2K (page size) and type 4 SVC routines are loaded in 2K multiples, instead of 1K, to reduce the time required to load type 4 SVC routines. SVC interruption handling processing time is reduced by the restructuring of the SVC table and its mapping table.

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

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

- The fast multiple wait facility is available to speed up the processing involved in determining which of the multiple events being waited on are completed.

New Features

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

- VTAM, a new telecommunications access method, provides far more comprehensive telecommunications support than BTAM, QTAM, or TCAM.

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

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

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

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

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

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

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

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

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

GC20-1752-3

OS/Virtual Storage 1 Features Supplement Printed in U.S.A. GC20-1752-3

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

OS/Virtual Storage 1 GC20-1752-3

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.

GC20-1752-3

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

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

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)