GC20-1756-2
File No. S370-20

Systems

DOS/Virtual Storage Features Supplement

Release 34

This supplement discusses DOS/Virtual Storage (DOS/VS) features and organization as of Release 34. Only concepts and functions of DOS/VS that are new to and significantly different from those of DOS Version 4 are presented in detail. Transition from DOS Version 4 to DOS/VS is discussed also.

This supplement is an optional section that is designed to be inserted in its entirety in any one of the base guides for Models 135, 138, 145, 148, and 158 and the 3031 Processor of System/370. Each of the guides for these processors contains the conceptual and System/370 hardware information required to understand the DOS/VS discussion presented.

Readers who possess more than one of the base processor publications need add this supplement to only one of the documents as the DOS/VS information presented applies to all supported processors unless otherwise indicated in the text.

The contents of this supplement are designed to acquaint the DOS Version 4 knowledgeable reader with the new facilities and the advantages of DOS/VS.

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 processor and DOS Version 4 and who require information about DOS/Virtual Storage (DOS/VS) should obtain this supplement and insert it as Section 80 in one of the appropriate base publications listed below. The features provided by the Advanced Functions-DOS/VS program product are also highlighted.

Base publications for the DOS/VS supplement are:

- 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 3031 Processor Complex of System/370 (GC20-1854)

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 DOS/Virtual Storage 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.

Third Edition (September 1978)

This is a major revision obsoleting GC20-1756-1. The text and illustrations have been updated to reflect new features of Releases 33 and 34. Changes are indicated by a vertical line in the left margin.

This publication applies to DOS/VS Release 34 and is intended for planning purposes only. It will be updated from time to time; however, the reader should remember that the authoritative sources of system information are the system library publications for DOS/VS. These publications will first reflect any changes.

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

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

CONTENTS (Section 80)

Section 80: DOS/Virtual Storage Features 1
80:05 Functions and Hardware Supported 1
80:10 Organization and Initialization of Storage 10
Virtual Storage Organization 10
Real Storage Organization 22
External Page Storage Organization 23
System Initialization 25
80:15 Major Components 31
80:20 The Job Control Program and Operator Commands 40
The Job Control Program 40
Operator Commands 48
80:25 The Supervisor 50
Modifications 50
New Features 53
Page Management 59
80:30 Data Management 80
Access Methods 80
Support of Additional I/O Devices 81
The Channel Scheduler 82
Virtual Storage Access Method 87
80:35 Recovery Management and Debugging Aids 127
MCAR, CCH, RMSR, and OLTEP 127
Debugging Aids 127
80:40 Language Translators, Service Programs, and Emulators . . . 132
DOS/VS Assembler 132
POWER/VS 132
Linkage Editor and Librarian 204
Utilities 206
Sort/Merge Programs 210
Integrated Emulators 212
80:45 Advanced Functions-DOS/VS Program Product 214
Support of Up to Seven Partitions 214
Dynamic Partition Balancing 214
Asynchronous Operator Communications 215
Faster Linkage Editing 215
Device Independence for Private Source Statement and Relocatable Libraries 216
DOS/VS-VM/370 Linkage Facility 216
80:50 DOS Version 4 to DOS/Virtual Storage Transition 218
80:55 Summary of Advantages 224
Expanded, More Flexible Multiprogramming 224
Operational Enhancements 225
Improved Utilization of Real Storage 226
Performance Enhancements 226

Index (Section 80) 228

FIGURES (Section 80)

80.10.1	Organization of virtual storage in DOS/VS	10
80.10.2	Determination of real address area size	17
80.10.3	Use of the SIZE parameter for a real mode job step.	19
80.10.4	Sample allocation of virtual storage to the page pool in the real address area.	20
80.10.5	Organization of real storage in DOS/VS.	22
80.10.6	Example of real storage allocation for a sample five-partition system.	23
80.10.7	Relationship of virtual storage, real storage, and external page storage in DOS/VS	25
80.10.8	Format of a page table entry for a page without a page frame assigned.	27
80.10.9	Contents of page table entries after system initialization.	28
80.15.1	Control and processing components of DOS/VS	31
80.15.2	Organization of a core image library.	33
80.15.3	Layout of the label cylinder for the DOS/VS SYSRES file	37
80.20.1	Example of modification of a cataloged procedure.	45
80.25.1	Organization of a virtual partition when the SIZE parameter is specified.	58
80.25.2	Format of a page frame table entry.	61
80.25.3	Operation of the page selection routine	64
80.25.4	Logical flow of page fault processing	66
80.25.5	Logical flow of PFIX macro processing	69
80.25.6	Logical flow of TFIX routine processing	71
80.25.7	Logical flow of GETREAL routine processing.	74
80.30.1	Organization of a control area for a VSAM file.	90
80.30.2	Relationships among VSAM control and request macros	97
80.30.3	Structure of the primary index for a VSAM key-sequenced file.	100
80.40.1	General operation of the POWER/VS system.	139
80.40.2	Relationship between a queue set, queue records, and a queue entry	146
80.40.3	Example of spool device assignments in POWER/VS without the use of dummy devices.	151
80.40.4	Example of the use of a dummy device when a card reader is used directly by a POWER/VS writer-only partition.	153
80.40.5	Example of the use of dummy devices when a POWER/VS-controlled partition uses more than one spooled printer/punch	154
80.40.6	Input stream device combinations and contents supported by POWER/VS	161
80.40.7	Relationship of POWER/VS functional tasks	172
80.40.8	Layout of the POWER/VS virtual partition.	189

TABLES (Section 80)

80.05.1	Standard features of DOS/VS	4
80.05.2	Optional features of DOS/VS	5
80.05.3	I/O devices, consoles, and terminals supported by DOS/VS.	7
80.05.4	Permissible I/O device type assignments for system logical units	9
80.30.1	The types of processing supported for VSAM key-sequenced files	108
80.30.2	Types of processing supported for VSAM entry-sequenced files	111
80.30.3	Types of processing supported for VSAM relative record files	113
80.30.4	Comparison table of VSAM and ISAM facilities for DOS/VS	122
80.40.1	I/O devices supported by POWER/VS	156
80.40.2	Comparison of POWER/VS and DOS/VS POWER facilities.	194
80.40.3	List of system utility programs	207

SECTION 80: DOS/VIRTUAL STORAGE FEATURES

80:05 FUNCTIONS AND HARDWARE SUPPORTED

DOS/VS, which is Version 5 of DOS, includes features provided by DOS Version 4 and offers major new functions and feature enhancements. The most significant new items of DOS/VS are support of:

- One virtual storage of up to 16,777,216 bytes with 64K segments and 2K pages (using dynamic address translation hardware)
- Up to five, instead of a maximum of three, problem program partitions. During system operation the operator has the capability of varying the partition dispatching priority that was established at system generation.
- A relocating loader to provide program phase relocation at phase load time
- POWER/VS, which replaces the POWER component available for DOS Version 4 and that provided for releases of DOS/VS prior to 31. POWER/VS is distributed as a component of the DOS/VS system and need not be ordered separately. POWER/VS operates in paged mode and provides functions in addition to those provided by previous POWER components.
- Use of the interval timing facility in each partition and by each task in a multitasking environment, instead of in only one partition
- A cataloged procedures facility, which allows job control statements and certain types of input data to be stored in a procedure library so that they need not be placed in an input stream
- An additional access method for direct access storage devices called virtual storage access method (VSAM) that is designed to offer better performance for files with additions and provide more function than ISAM
- New I/O devices, such as 3330 Model 11, 3340/3344, and 3350 direct access storage, and the 3800 Printing Subsystem.
- An additional access method called virtual telecommunications access method (VTAM), which supports network control program mode for the 3704 and 3705 Communications Controllers and provides facilities not available in BTAM or QTAM
- Block multiplexer channels and rotational position sensing
- A new version of the Assembler Language that provides enhanced performance and improved diagnostics
- A shared virtual area in virtual storage in which reentrant, relocatable code can be made resident to be shared by concurrently executing problem programs
- Display mode of operation for the 3277 display as an operator console and the display consoles provided for the processors supported by DOS/VS
- A new core image library organization, and residence in virtual storage of core image library directory entry lists and of second

level directories for core image libraries to improve phase fetching performance

- A more flexible I/O device assignment capability which provides dynamic allocation by the job control program of available I/O devices to symbolic units at job step initiation time based on a generic I/O device type assignment, instead of a specific unit assignment

- Additional tracing and storage dumping facilities that can be used for debugging and statistics gathering (SDAIDS routines)

- The DOS/VS System Installation Productivity Option and improvements in the system generation procedure, both of which are designed to reduce the time required to produce the DOS/VS system required in an installation

- The Advanced Function-DOS/VS program product, which provides several additional capabilities (see Section 80:45)

DOS/VS is upward compatible with DOS Versions 3 and 4 for System/370. System/360 users with BPS, BOS, or TOS installed can move to a System/370 and a DOS/VS virtual storage environment without a large conversion effort as well because of the large degree of source program and data file compatibility that exists between these programming systems and DOS. Transition from DOS Version 4 to DOS/VS is discussed in Section 80:50.

DOS/VS, classified as system control programming (SCP), supports System/370 Models 115 (Models 0 and 2), 125 (Models 0 and 2), 135 (Models 0 and 3), 138, 145 (Models 0, 2, and 3), 148, 155 II, and 158 (Models 1 and 3) operating in EC and translation modes. Support of the 3031 Processor Complex will also be provided. DOS/VS does not support System/370 processors operating in EC mode without dynamic address translation specified, System/370 processors operating in BC mode, any System/360 processors, or Model 158 systems operating in attached processor or multiprocessor mode.

The following minimum system configuration and hardware features are used by DOS/VS:

- Minimum real storage size available for the System/370 processors supported by DOS/VS

- Dynamic address translation and channel indirect data addressing

- Storage protection

- One reader, one punch, and one printer attached via a byte multiplexer channel or, on a Model 115 or 125, natively attached. A cardless system configuration is supported for Models 115 and 125. If a Model 115/125 configuration includes a 3540 Diskette Input/Output Unit, a card reader and card punch are not required. At least one 3741 or 3742 Data Entry Unit in the installation must have the Record Insert feature installed to support program service requirements. The DOS/VS distribution program, standalone Test Copy Utility, and standalone Dump Utility are extended to accept control card-image input from diskettes. IPL control statements cannot be supplied via a 3540 but must be entered via the display operator console.

- One console

- Two direct access devices or logical volumes (2314/2319, 3330-series, 3340, 3344, or 3350) or one direct access device and two

tape units (the data conversion feature is required for 7-track tape units

Tables 80.05.1 and 80.05.2 list the standard and optional features of DOS/VS. DOS/VS supports all the features that are provided in DOS Version 4 except single program initiator (SPI) mode of foreground partition scheduling and 2311 Disk Storage as the system residence device. If one of the IBM-supplied generated DOS/VS supervisors cannot be used, the desired installation-tailored DOS/VS supervisor must be generated, at which time user-selected optional features are included, as is required for a DOS Version 4 supervisor.

Table 80.05.3 lists the I/O devices, consoles, and terminals supported by DOS/VS. All the System/370 I/O devices supported by DOS Version 4 are also supported by DOS/VS. The I/O device types that can be assigned to system logical units are listed in Table 80.05.4. The SYSUSE logical unit is provided only for use by system routines as required. For example, job control uses SYSUSE as the logical unit when it must perform certain I/O operations, such as a magnetic tape rewind in response to an operator command. The tape unit is assigned to SYSUSE in order to perform the operation.

A minimum of ten programmer logical units must be assigned to the background partition and each foreground partition defined must have a minimum of five programmer logical units assigned. The maximum number of programmer logical units in the system depends on the number of partitions generated. The SYSmax value for the system is 241 for one partition, 226 for two partitions, 212 for three partitions, 198 for four partitions, and 184 for five partitions.

The SYSmax value for the foreground 1 partition is alway 241, regardless of the number of partitions defined. The SYSmax value for any partition other than foreground 1 is the SYSmax value for the system less the sum of all programmer logical units assigned to other defined partitions except foreground 1.

Note that the PUB table must contain one entry for each logical volume in a physical device with more than one logical volume. Thus, a DVCGEN macro is required for each of the four logical volumes in a 3344 direct access device and for each of the two 3330 Model 1 volumes in a 3350 direct access device.

Table 80.05.1. Standard features of DOS/VS. These features are automatically included during system generation.

- Support of one virtual storage of user-specified size (up to 16,384K bytes) with 64K segments and 2K pages*
- Batched job mode of job initiation and a one-partition (nonmultiprogramming) environment
- Execution of programs in virtual (paged) mode* and real (nonpaged) mode
- Shared virtual area for the system directory list, reentrant program phases, and a system GETVIS area*
- Symbolic I/O device assignment
- Generic I/O device assignment and volume premounting*
- Cataloged procedures*
- Job control exit and IPL exit facilities*
- Second level directory for the system core image library*
- Storage protection
- SAM, ISAM, and DAM
- Command chaining for I/O retry operations
- Selector channel support
- Tape error statistics
- Display operator console support (standard only for Models 115, 125, 138, and 148)*
- Machine Check Analysis and Recording (MCAR), Channel Check Handler (CCH), and Recovery Management Support Recorder (RMSR) routines (optional for Models 115 and 125 but RMS=YES must be specified for Model 115 and 125 supervisors that are to execute on a Model 135, 138, 145, 148, 155 II, or 158)
- OLTEP (can be omitted for all supported processors)
- Job Control
- Linkage Editor
- Relocating Loader* (can be excluded unless required by a selected optional feature)
- Librarian
- Assembler
- System Utilities
- SDAIDS*

*Facility not supported in DOS Version 4

Table 80.05.2. Optional features of DOS/VS. These features must be requested during system generation or added after the generation is performed.

- Multiprogramming (from two to five* partitions with BJF scheduling)
- Specification of partition dispatching priority*
- Multitasking (up to 15* tasks maximum)
- Supervisor selection at IPL*
- POWER/VS*
- Teleprocessing support (VTAM* and TOLTEP*, BTAM, QTAM)
- Teleprocessing balancing (automatically included when BTAM support in a multiprogramming system, QTAM support, or VTAM support is specified)*
- VSAM*
- Multiple wait
- Magnetic ink character reader (MICR) and optical reader support
- ASCII-code support for tapes
- Access to the time-of-day value in the time-of-day clock via the GETIME macro
- Multiple timer support*
- Task timer support*
- Job accounting interface
- Private core image libraries*
- System input and system output files on the 3540 Diskette Input/Output Unit* and disk
- Independent directory read-in area
- External interruptions
- Abnormal termination exit
- Console buffering (not supported for display-type consoles)
- Display operator console support for a 3277 as the operator console for a supported processor
- Track hold (not supported for the 3540)
- DASD file protection
- Seek separation

*Facility not supported in DOS Version 4

Table 80.05.2 (continued)

- Support of 3330-series (Models 1, 2, and 11* disk storage
- Support of the 3340 direct access storage facility (includes support of the 3344)*
- Support of the 3350 (native and 3330 compatibility modes)*
- Support of the 3800 Printing Subsystem*
- Channel switching for tape
- Burst mode devices on the byte multiplexer channel
- MICR or optical reader/sorter device(s) on the byte multiplexer or a selector channel (burst mode and MICR devices cannot operate concurrently on a byte multiplexer channel)
- Block multiplexing*
- Rotational position sensing*
- Magnetic tape error volume analysis
- Private second level directories for private core image libraries*
- Support of mixed-parity tapes by the following integrated emulators

  Model 20 emulator (Models 115, 125, 135, and 138)
  1401/1440/1460 emulator (Models 115, 125, 135, 138, 145, 148, 155 II, and 158)
  1410/7010 emulator (Models 145, 148, 155 II, and 158)

- Reliability Data Extractor
- Problem Determination Aids (PDAIDS)
- Page fault handling overlap*
- Support of PFIX/PFREE macros*
- Support of PAGEIN, RELPAG, and FCEPGOUT macros*
- Support of GETVIS/FREEVIS macros*
- Support of the VIRTAD and REALAD macros and the REAL parameter on the EXCP macro*
- Support of the synchronous exit facility (SYNCH macro)*
- Fast CCW translation*
- Cross partition event control*
- Supervisor identification in IPL COMPLETE message*

*Facility not supported in DOS Version 4

Table 80.05.3. I/O devices, consoles, and terminals supported by DOS/VS

Card Readers and Punches

1442 Reader/Punch, Models N1 and N2
2501 Card Reader, Models B1 and B2
2520 Card Read Punch, Models B1, B2, B3
2540 Card Read Punch
2560 Multifunction Card Machine, Model A1 (Models 115 and 125 only)
2596 Card Read Punch
3504 Card Reader, Models A1 and A2 (Model 125 only)
3505 Card Reader, Models B1 and B2
3525 Card Punch, Models P1, P2, P3
5425 Multifunction Card Unit, Models A1 and A2 (Models 115 and 125 only)

Printers

1403 Printer, Models N1, 2, 3, 7
1443 Printer, Model N1
3203 Printer, Models 1 and 2 (Models 115 and 125 only)
3203 Printer, Model 4 (Models 138 and 148 only)
3211 Printer
3213 Printer (Model 158 only)
3800 Printing Subsystem
5203 Printer, Model 3 (Model 115 only)
5213 Printer (Model 125 only)

Diskettes

3540 Diskette Input/Output Unit

Direct Access Storage

2311 Disk Storage Drive, Model 1
2314 Direct Access Storage, Models 1, A, and B
2319 Disk Storage, A and B models
2321 Data Cell Drive
3330-Series Disk Storage, all models (RPS, Sixteen-Drive Addressing, and 32-Drive Addressing features are supported)
3340 Direct Access Storage Facility (RPS, Sixteen-Drive Addressing, 32-Drive Addressing, and Fixed Head features are supported)
3344 Direct Access Storage
3350 Direct Access Storage (in native or a 3330 compatibility mode)

Note: The Record Overflow, Two-Channel Switch, Two-Channel Switch Additional, and string switching features available for the direct access devices above are not supported.

Magnetic and Paper Tape

1017 Paper Tape Reader
1018 Paper Tape Punch
2400- and 3400-series Magnetic Tape Units, all models and densities (channel switching for a maximum of two control units is supported)
2495 Tape Cartridge Reader
2671 Paper Tape Reader

Display Units (locally attached)

2260 Display Station
3277 Display Station

Table 80.05.3 (continued)

Optical and Magnetic Character Readers

1287, 1288 Optical Character Readers
1255, 1259, 1419 Magnetic Character Readers
3881 Optical Mark Reader
3886 Optical Character Reader

Consoles

3210, 3215 Console Printer-Keyboards
Model 158 display console (in printer-keyboard mode only) and required 3213 Printer
3277 Display Station attached to a 3272 Control Unit
Display Operator Consoles for Models 115, 125, 138, and 148

Transmission Control Units and Integrated Communications Adapters (ICA's)

2701, 2702, 2703, 2715 Transmission Control Units
3704 Communications Controller
3705 Communications Controller
7770 Audio Response Unit
ICAs for Models 115, 125, 135, and 138

Terminals (Start/Stop)

1030 Data Collection System
1050 Data Communication System
1060 Data Communication System
2260 Display Station
2265 Display Station
2721 Portable Audio Terminal
2740 Communication Terminal
2741 Communication Terminal
2760 Optical Image Unit
83B3 AT&T Terminal
WU115A Teletype
TWX-33/35 AT&T Teletype Terminal
System/7 Sensor Based Information System (as a 2740 Terminal)
Communicating Magnetic Card Selectric Typewriter

Terminals (Binary Synchronous)

2770 Data Communication System
2780 Data Transmission Terminal
2790 Data Communication System
2792 Models 8 and 11 General Banking Stations
2980 General Banking Terminal System
3270 Information Display System
3600 Finance Communication System
3650 Retail Store System
3660 Supermarket System
3735 Programmable Buffered Terminal
3740 Data Entry System
3741 Data Station Model 2
3741 Programmable Terminal Model 4
3767 Communication Terminal
3770 Data Communication System
3780 Data Communication Terminal
3790 Communication System
1130 System (as a processor station)
1800 System (as a processor station)
System/3 (as a processor station)

Table 80.05.3 (continued)

System/7 (as a processor station) System/32 (as a 3770) System/360 Models 20 and up (as a processor station) System/370 models (as a processor station)
Note: 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.

Table 80.05.4. Permissible I/O device type assignments for system logical units

	Logical Units												
Device Type	SYS RDR	SYS IPT	SYS LST	SYS PCH	SYS LOG	SYS RES	SYS CLB	SYS RLB	SYS SLB	SYS REC	SYS VIS	SYS LNK	SYS CAT
1403			X		X								
1442	X	X		X									
1443			X		X								
2501	X	X											
2520	X	X		X									
2540	X	X		X									
2560	X	X		X									
3203			X		X								
3210					X								
3211			X		X								
3800			X		X								
3213					X								
3215					X								
3504	X	X											
3505	X	X											
3525	X	X		X									
3540	X	X	X	X									
5203			X		X								
5425	X	X		X									
Model 115/125, 138/148, or 158 display console					X								
3277					X								
2400-series	X	X	X	X									
3400-series	X	X	X	X									
2311	X	X	X	X						X		X	
2314	X	X	X	X		X	X	X	X	X	X	X	X
2319	X	X	X	X		X	X	X	X	X	X	X	X
3330-series	X	X	X	X		X	X	X	X	X	X	X	X
3340/3344	X	X	X	X		X	X	X	X	X	X	X	X
3350	X	X	X	X		X	X	X	X	X	X	X	X

Note: The DOS/VS supervisor does not assume 80-byte records for SYSRDR and SYSIPT files contained on direct access devices, as does the DOS Version 4 supervisor. Both 80- and 81-byte record sizes are supported. DOS/VS determines whether 80- or 81-byte records are present on a disk SYSRDR or SYSIPT device and establishes the control required to handle the record size found. This change enables a SYSPCH file on disk with 81-byte records to be used as a SYSRDR or SYSIPT file. The "no record found" condition that occurs in a DOS Version 4 environment in such a situation is eliminated in DOS/VS.

80:10 ORGANIZATION AND INITIALIZATION OF STORAGE

VIRTUAL STORAGE ORGANIZATION

DOS/VS supports one partitioned virtual storage of up to 16,777,216 bytes with segments of 64K and pages of 2K. The size of the virtual storage to be supported is user-specified at system generation and cannot be changed during IPL. In DOS/VS, virtual storage is divided into a virtual address area in highest addressed virtual storage and a real address area in lowest addressed virtual storage, as shown in Figure 80.10.1.

The actual size of the real address area is equal to or less than the size of the real storage present in the system. Thus, virtual storage in the real address area can have real storage allocated to it on a virtual storage address equals real storage address basis. The virtual storage addresses in the virtual address area have no equivalent real storage addresses.

Parameters (VSIZE and RSIZE)) can be specified at system generation to indicate the size, in 2K multiples, of the virtual address area, which contains a shared virtual area (SVA), and the real address area. The real address area must be equal to or greater than 64K for the systems supported. The virtual address area must be a minimum of 64K times the number of partitions defined plus the minimum SVA size (which is 64K). The sum of the values supplied by the RSIZE and VSIZE parameters is the virtual storage size supported by the generated supervisor and cannot exceed 16,384K (16,777,216) bytes.

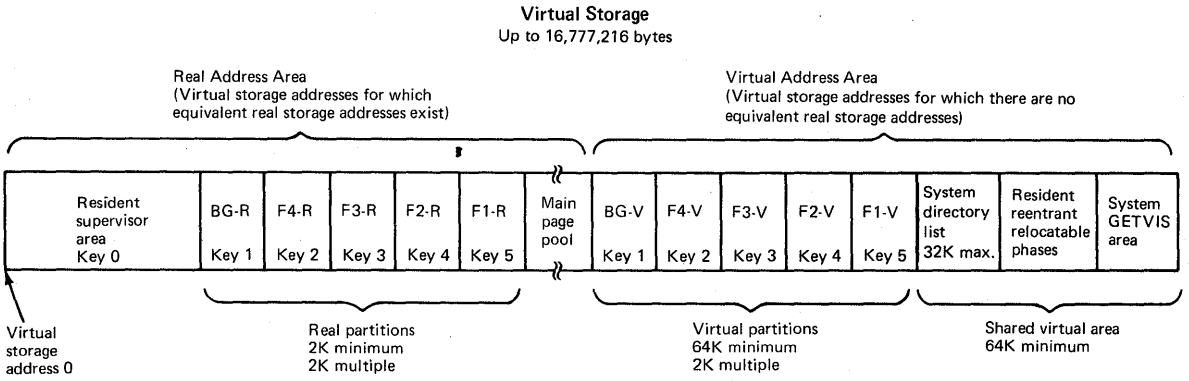


Figure 80.10.1. Organization of virtual storage in DOS/VS

The virtual storage defined can be partitioned to permit concurrent execution of a maximum of five jobs instead of up to three, as in DOS Version 4. One background (BG) partition and up to four foreground (F1, F2, F3, and F4) partitions can be defined in DOS/VS. F3 and F4 partitions are positioned between the F2 and the BG partitions. All facilities available to BJF foreground partitions F1 and F2 are also available to the two additional foreground partitions, F3 and F4, that are supported in DOS/VS. Restrictions on the functions that can be performed in BJF foreground partitions are the same in DOS Version 4 and DOS/VS. The maximum number of partitions that a DOS/VS supervisor is to support is specified at system generation via the NPARTS parameter.

In a DOS/VS environment, each partition specified has a unique portion of the virtual address area assigned to it that is called a virtual partition. Optionally, each partition can also have a unique

portion of the real address area assigned to it that is called a real partition.

Virtual partitions are used for the execution of programs in virtual mode while real partitions are used for programs that must operate in real mode. Real partitions are also required by virtual mode programs that use the optional page fixing facility (PFIX/PFREE macros), which is discussed in Section 80:25. The address space within the virtual address area that is allocated to a given partition and the address space allocated to the same partition within the real address area cannot be used to execute two different job steps at the same time. That is, when a program is executing in a virtual partition, the associated real partition cannot also be used for program execution and vice versa.

All foreground partitions (real and virtual) in DOS/VS are initiated using BJF mode of initiation. SPI mode is not required or supported by DOS/VS. The implementation of virtual storage enables real partitions that are a minimum of 2K in size to be initiated without the necessity of SPI mode since the job control program always executes in a virtual partition. Elimination of SPI mode removes SPI mode restrictions from small real partitions and simplifies job initiation operations. In addition, the operator need learn and use only BJF commands in DOS/VS.

As in a DOS Version 4 environment in which all foreground partitions are scheduled using BJF mode, when POWER/VS is not used in a DOS/VS system each partition (virtual/real pair) must have its own job stream that indicates the jobs that are to be executed in the partition. Virtual mode and real mode job steps can be mixed within the same job. Any mixture of virtual mode and real mode job steps can operate concurrently up to a maximum of the number of active partitions defined.

A variable partition priority facility that is not supported in DOS Version 4 is implemented in DOS/VS. A PRTY parameter can be specified at system generation to assign a dispatching priority to the partitions defined. If this parameter is not included, the default high-to-low dispatching priority for partitions is F1, F2, F3, F4, BG.

The dispatching priority established at system generation can be changed by the operator at any time during system operation using the PRTY attention command. The PRTY command can also be used to cause the currently assigned partition dispatching priorities to be printed on the SYSLOG device. In DOS Version 4, partition dispatching priority is established by the system and cannot be changed by the user.

The variable partition priority facility will be useful, for example, in the handling of high priority jobs when job scheduling is designed to allow a high priority job to be scheduled to execute in any partition that is available at the time the high priority job must be initiated. The priority of the partitions can be altered to assign the desired dispatching priority to the high priority job partition based on the execution characteristics of the job.

Virtual Address Area

The virtual address area contains from one to five virtual partitions and the required shared virtual area. All the virtual storage in the virtual address area that is below the SVA is available for allocation to virtual partitions. One background virtual partition (BG-V) in lowest addressed virtual storage in the virtual address area and from one to four batched foreground virtual partitions (F4-V, F3-V, F2-V, and F1-V) above the background virtual partition can be defined.

The size of each foreground virtual partition can be specified at system generation using the ALLOC macro or during system initialization using the ALLOC (job control or attention) command. The entire virtual address area less the minimum requirement for the SVA is allocated to the background virtual partition if ALLOC is not specified for a multiprogramming system. The ALLOC macro cannot be specified for a one-partition system.

Each virtual partition defined must be a minimum of 64K and a multiple of 2K bytes in size. The maximum size a virtual partition can be is the size of the defined virtual address area less the user-defined or default SVA size. The restriction of 510K bytes maximum for a BJF foreground partition that exists in DOS Version 4 is removed in DOS/VS. Each virtual partition has its own partition save area and label save area in lowest addressed virtual storage in the partition.

The virtual storage sizes specified in an ALLOC macro or command are allocated contiguously to foreground virtual partitions beginning with highest addressed virtual storage in the virtual address area below the SVA. Any remaining virtual storage below the last foreground virtual partition allocated is assigned to the background virtual partition, just as in DOS Version 4.

The number of active virtual partitions established during system initialization and their sizes can be changed by the operator during system operation, if necessary, using the ALLOC command, as in DOS Version 4. However, there should be less need for partition redefinition in a DOS/VS environment because the implementation of virtual storage enables larger partitions to be defined and more partitions can be defined without inefficient use of real storage.

A job step executing in virtual mode in a virtual partition operates with dynamic address translation (DAT) mode specified so that translation of storage addresses in instructions is performed by DAT hardware. Translation of storage addresses in the channel programs for a virtual mode job step is performed by an extension to the channel scheduler routine. The default execution mode for a job step is virtual. If a program is to operate in real mode, the REAL parameter must be specified on the EXEC job control statement for the job step.

Page frames are allocated to a virtual mode program dynamically as the program is loaded and as it executes. When a page fault occurs in a virtual mode program, a page frame is allocated, as per the rules of the page replacement algorithm, and a page-in is performed when necessary. The real storage allocation routine recognizes the first time a page frame must be allocated to a virtual storage page in a virtual partition. When this situation occurs, no page-in is performed and the page frame allocated is zeroed for data security protection.

The EXEC statement in DOS/VS also has a SIZE parameter that is not supported in DOS Version 4. If this parameter is not specified for a job step that is to operate in virtual mode, the job step has access to all the virtual storage currently allocated to the virtual partition in which the job step executes. If SIZE is specified, the virtual partition is divided into two portions. The lowest addressed portion of the virtual partition is designated as the user area and is the size specified by the size value in the SIZE parameter. The job step executes in the user area. The upper portion of the virtual partition is a virtual storage pool that can be used by the virtual mode program that is executing in the user area.

Virtual storage in the pool created by the SIZE parameter can be allocated and deallocated using the GETVIS and FREEVIS supervisor macros, which are discussed in Section 80:25 under "Virtual Storage Management Facility". The virtual storage pool is also referred to as a

partition GETVIS area. The virtual storage access method (VSAM), for example, uses the GETVIS and FREEVIS macros. Therefore, the SIZE parameter must be specified for job steps that use VSAM. The SIZE parameter must also be specified for job steps that are to use rotational position sensing support.

The SIZE parameter has two AUTO options. If AUTO is specified instead of a size value, the job control program will use the size of the program that is to execute in the virtual partition as the size value for the user area. Program size is taken from the core image library directory entry for the phase that is specified in the EXEC statement. AUTO can also be specified together with a size value. In this case, the size value used for the user area is the sum of the phase size and the size value specified.

The SIZE parameter can also be used to limit the amount of virtual storage in a virtual partition to which a virtual mode program has access. For example, SIZE can be specified for programs, such as sorts, that are designed to make use of all the storage available in the partition in which they execute and that have a random, short duration reference pattern during execution. Best performance is achieved for these types of programs when real storage nearly equal to the amount of virtual storage they use can be made dynamically available to them without causing excessive paging. Specification of the SIZE parameter for a sort program that operates in a relatively large virtual partition can improve sort performance by limiting the amount of virtual storage available to the sort when an equal amount of real storage is not dynamically available.

Programs that must execute in virtual mode in a DOS/VS environment are the job control program, POWER/VS, and any problem programs that are to use rotational position sensing support, GETVIS/FREEVIS macros, VSAM, the ISAM interface routine for VSAM, the VSAM access method services program, or VTAM.

The shared virtual area is required in all systems but cannot be used in nonmultiprogramming systems. The SVA is located in highest addressed virtual storage in the virtual address area. The size of the SVA can be specified at system generation using the SVA parameter on the ALLOC macro. The SVA must be a minimum of 64K and a multiple of 2K. The SVA size stated at system generation in the SVA parameter can be overridden by the operator immediately following IPL.

The SVA can contain resident program phases that can be shared by concurrently executing partitions, a system directory list, and a system GETVIS area. Phases that are to be placed in the SVA must be contained in the system core image library. They also must be relocatable, reentrant (never modified during their execution), and pageable. Once phases are fetched from the system core image library and made resident in the SVA, they are paged in from external page storage as required, as are phases that are executing in virtual partitions. However, phases in the SVA should never be paged out during system operation since they should never be modified.

Program phases that are to be placed in the SVA are user selected. Frequently used phases that are required by concurrently executing problem programs should be chosen for SVA residence. These phases can be shared by both virtual and real mode programs. Placing such routines in the SVA can be of benefit to system performance. First, only one copy of an SVA-resident phase will ever be present in real storage at a time, regardless of how many programs or tasks are using the phase concurrently. This makes more real storage available for paging. Second, a phase in the SVA will tend to remain in real storage during periods when it is heavily used, since the page replacement algorithm is

designed to keep the most active pages present in real storage. Thus, use of the SVA can reduce the total number of page faults that occur.

Note that the $JOBACCT dummy phase for job accounting can be loaded in the shared virtual area to improve the performance of systems that use POWER/VS job accounting.

The system directory list is resident in lowest addressed virtual storage in the SVA and can be a maximum of 32K bytes in size. It can contain core image library directory entries for user-selected phases that are present in the system core image library. Normally, the system directory list would consist of directory entries for the most frequently used system and user-written phases in the system core image library.

The system directory list also contains directory entries for all phases that are resident in the SVA, if any. A directory entry in the system directory list points to the location of the phase the directory entry describes. It contains the track and record address of the phase in the system core image library or the virtual storage address of the phase in the SVA.

The system directory list is shared by all partitions and the supervisor. Entries in this list are in ascending alphabetic sequence by phase name. When a FETCH or LOAD macro is issued within any partition or by the supervisor, the system directory list in the SVA is searched before the system core image library directory on disk is searched. Use of the system directory list can reduce the amount of time required to locate frequently used supervisor transient routines, system programs, and selected user-written programs by eliminating for these phases all searching of the system core image library directory on disk.

Implementation of a system directory list, in conjunction with other new DOS/VS features, such as second level directories and a new core image library organization, also eliminates the need for subdirectories on disk for the system core image library (transient directory, OPEN directory, etc.).

If defined, the system GETVIS area is located in highest addressed virtual storage in the SVA. It must be a multiple of 4K in size and smaller than the SVA size. The size of the system GETVIS area, up to a maximum size of 12,168K bytes, can be specified at system generation and overridden during system initialization. The system GETVIS area is utilized by rotational position sensing support (see Section 80:30).

Use of the SVA is optional in a multiprogramming environment. That is, there is no defined list of directory entries that must be made resident in the system directory list, there is no requirement to have any program phases resident in the SVA, and a system GETVIS area need not be defined unless rotational position sensing support is to be used. Lists of suggested system phases, VSAM phases, and RPS phases whose directory entries should be placed in the system directory list are IBM-supplied, however. For performance reasons, it is highly desirable to make these directory entries resident in the system directory list. If no phases or directory entries are made resident in the SVA, it contains only required control information in its lowest addressed virtual storage.

Optionally, a problem program can build one or more local directory lists within a partition that contain the directory entries of frequently used phases. The local directory list(s) in one partition cannot be used by problem programs that are executing in other partitions. A local directory list is constructed within a program using the GENL macro to specify the names of phases whose directory entries are to be included in the local directory list. Phase names should be listed in ascending alphabetic sequence.

Directory entries for phases contained in the system core image library (which may or may not be resident in the SVA) and/or a private core image library that a problem program is to use can be included in a local directory list. When the GENL macro is assembled, each directory entry constructed contains only the name of the phase. Other directory data is placed in the local directory list entry for a phase the first time a FETCH or LOAD macro is issued for the phase and the phase is located in the SVA or a core image library.

A local directory list is searched when a problem program issues a FETCH or LOAD macro that specifies the name of the local directory list. If the directory entry of the specified phase is found in a local directory list or the system directory list, the phase (unless it is resident in the SVA) is fetched from the appropriate core image library without any directory searching on disk. The local directory list facility can be used in a nonmultiprogramming as well as a multiprogramming system. (The system directory list cannot be used in a nonmultiprogramming environment.)

The SVA is initially created immediately following an IPL when the SET SDL=CREATE command is issued. SVA-defining statements can be supplied in one of two ways. The operator can enter the SET SDL command using the SYSLOG device after the size of the SVA and the system GETVIS area have been established. Control statements that name the phases whose directory entries are to be placed in the system directory list must be contained in the SYSRDR device. Inclusion of the SVA parameter on a control statement indicates the phase is to be made resident in the SVA as well.

Alternatively, a SET SDL command followed by SVA-defining control statements can be placed in the procedure library and called by the operator via an EXEC job control command. A user-defined procedure or one of the IBM-supplied SVA-defining procedures can be used. (See procedure library discussion in Section 80:20 under "The Job Control Program").

Use of the procedure library during creation of the SVA is suggested as an operational aid. The SVA-defining statements in the procedure can be modified immediately following IPL if the operator enters the SET SDL command and supplies modifier statements via the SYSLOG device. Job control reads the SVA-defining control statements from the procedure library or SYSRDR device as indicated and calls the $MAINDIR service program, which then constructs the SVA.

The $MAINDIR program, which is not provided in DOS Version 4, searches the system core image library directory on disk for the phases specified by the SVA-defining control statements. If the entry for a specified phase is found, the entry is read in and placed in the system directory list. If the maximum system directory list size is reached during loading, the operator is notified and no more SVA-defining statements are processed. If a specified phase is not currently cataloged in the core image library, a dummy entry for the phase is placed in the system directory list. This entry is identified as being inactive and the operator is notified.

If an SVA-defining control statement contains the SVA parameter but the specified phase is not marked eligible for SVA residence, the phase is not loaded into the SVA. A phase is identified as being eligible for residence in the SVA by inclusion of the SVA parameter on its PHASE statement when the phase is link-edited. Phases that are to be made resident in the SVA are allocated virtual storage above the system directory list, fetched from the system core image library, and placed in the SVA. In order to reduce page faults, each phase is aligned on a page boundary in the SVA, unless it fits in the space left in the last used virtual storage page.

Once the $MAINDIR program finishes processing all the SVA-defining control statements, it writes the contents of the SVA (system directory list and resident phases) in external page storage, which is called the page data set in DOS/VS. After an SVA has been written in the page data set, it can be reused and SVA creation after each IPL can be eliminated. During IPL, job control determines whether the page data set contains a usable SVA. If one is present, the operator is notified that a warm start copy of the SVA exists. The operator can indicate that this copy is to be used as the SVA for this IPL or that a new SVA is to be created.

During system operation, whenever a new phase is cataloged in or an existing phase is deleted from the system core image library, a check is made to determine whether there is a directory entry in the system directory list for this phase. If so and if a cataloging operation was just performed, the entry in the system directory list is updated and made active. If the entry indicates the phase is to be made resident in the SVA, the phase is fetched and placed in the SVA if it is marked SVA eligible. If a delete operation was just performed, the affected directory entry in the system directory list is flagged as unusable. In addition, whenever an existing phase is recataloged, if its directory entry is in the SVA, the existing entry is updated.

A new copy of the SVA must be built if existing directory entries in the system directory list are to be physically deleted, or if directory entries are to be added. These types of system directory list modifications are not supported during normal processing. These functions must be accomplished by changing the control statements that define the contents of the SVA and recreating the SVA following an IPL. (Note that the ALTER command cannot be used to change the contents of the SVA.)

Real Address Area

Normally, the range of virtual storage addresses in the real address area is the same as the range of real storage addresses provided by the real storage present in the computing system. The storage size specified in the RSIZE system generation parameter should be the size of the real storage present in the system in which the generated supervisor is to operate. The minimum size of the real address area is 64K.

The range of virtual storage addresses in the real address area is not the same as the range of real storage addresses for the real storage present in the system used when a DOS/VS supervisor executes in a system that has more real storage than was defined by the RSIZE parameter at system generation.

If the real storage present in the system is larger in size than the system generation specification, the real storage above the amount specified by the RSIZE parameter will not be used and the operator is notified during IPL. (The first address in the virtual address area is determined at system generation time, based on the specified RSIZE, and cannot be changed except by another supervisor generation.) Thus, when real storage is added to a system in which DOS/VS is being used, the supervisor must be regenerated using an RSIZE parameter that specifies the new real storage size in order for the additional real storage to be utilized.

If the real storage present in the system used is smaller in size than the RSIZE specification (such as when a system backs up another system that has more real storage), the virtual storage in the real address area between the address of the end of real storage and the address of the beginning of the virtual address area is not used.

Figure 80.10.2 illustrates how the size of the real address area is determined at IPL.

The real address area contains the resident supervisor area, from one to five contiguous real partitions (optionally), and the main page pool area. The resident supervisor operates in real mode in the supervisor area with DAT mode specified most of the time, since the supervisor references virtual storage addresses outside its area and address errors would occur if DAT was not operative.

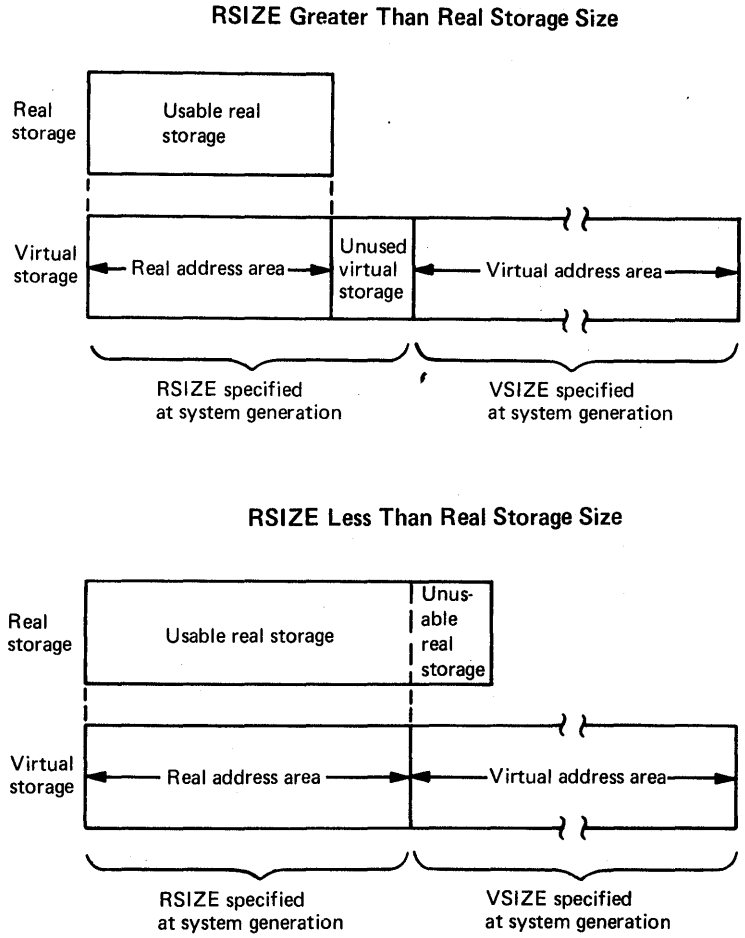


Figure 80.10.2. Determination of real address area size

Each virtual partition defined in the virtual address area can have a corresponding real partition defined in the real address area. Real partitions are designated as BG-R, F4-R, F3-R, F2-R, and F1-R. Each real partition has its own partition save area and label save area reserved in lowest addressed virtual storage in the real partition.

A given partition can have a virtual partition defined without also having a corresponding real partition defined unless a real mode program is to operate in the partition or unless a program operating in the virtual partition is to use page fixing macros. However, a real partition cannot be defined unless its corresponding virtual partition is defined as well. This requirement exists because the job control program always operates in virtual mode. The job control program must

execute in a virtual partition in order to schedule the execution of real mode job steps in the corresponding real partition.

The size of each real partition, including the BG-R partition, can be specified using the ALLOCR macro at system generation or the ALLOCR (job control or attention) command after IPL. The partitions defined by ALLOCR are called permanent real partitions. A real partition need not be the same size as its corresponding virtual partition. If not given a zero size allocation, a real partition must be a multiple of 2K bytes in size and can be a minimum of 2K. The maximum size of a real partition is the size of the real address area less the size of the supervisor area, less the size of the minimum main page pool area requirement.

Virtual storage is allocated to real partitions beginning with the virtual storage adjacent to the resident supervisor area. The address space in the real address area that is located above the last defined real partition is allocated to the main page pool. If ALLOCR is not specified, all virtual storage in the real address area that is located above the resident supervisor area is allocated to the main page pool. The ALLOCR macro cannot be specified for a one-partition system.

The minimum size of the main page pool in the real address area of virtual storage is:

- 18K if phases are to be made resident in the SVA for use by executing programs plus 2K if multitasking support is present
- 18K minus the size of the smallest real partition, if this partition is 0 to 18K in a multiprogramming system without the page fixing option plus an additional 2K if multitasking support is present. If the smallest real partition is larger than 18K, 0K is the minimum main page pool size. In both situations, the main page pool must be 4K if the system directory list is to be used.
- 18K in a multiprogramming system with the page fixing option plus 2K if multitasking support is present

A job step that is to execute in real mode is directed to operate in a real partition by using the REAL parameter of the EXEC statement. Programs that operate in real mode execute with DAT mode operative even though they are not paged. However, channel program translation is not performed for real mode programs.

Real storage is allocated to a real mode program when the program is scheduled for execution. Page frames are allocated such that the real storage assigned to the real partition has addresses equal to those of the virtual storage allocated to the real partition. Page frames allocated to a real partition are, in effect, permanently fixed and remain unavailable for allocation to virtual mode programs for the duration of the real mode job step.

At the completion of a real mode job step, the page frames allocated to the real partition are released and added to the list of assignable page frames. These page frames can then be allocated as required to any executing virtual mode programs or to another real mode program that is initiated in the now available real partition.

The SIZE parameter can also be specified for real mode programs. The storage size indicated in the SIZE parameter specifies the low-order portion of the real partition that is to be allocated to the real mode job step, which becomes a temporary real partition that exists for the duration of the execution of the job step. The temporary real partition must be a multiple of 2K in size.

When the SIZE parameter is specified for a real mode program, page frames are allocated only to the temporary partition. The virtual storage within the permanent partition that is located above the temporary partition defined by the SIZE parameter is not available to the real mode program during its execution. Page frames with addresses equal to those of this unallocated virtual storage in the permanent partition are made available for allocation to executing virtual mode programs. The two AUTO options of the SIZE parameter described previously can be specified for real as well as virtual partitions.

If the SIZE parameter is not specified, all the virtual storage defined in the permanent real partition has real storage assigned and is available to the real mode program. Therefore, the SIZE parameter can be used to make real storage that will not be used by a real mode program but that otherwise would be allocated to the real partition available for allocation to executing virtual mode programs. (See Figure 80.10.3.)

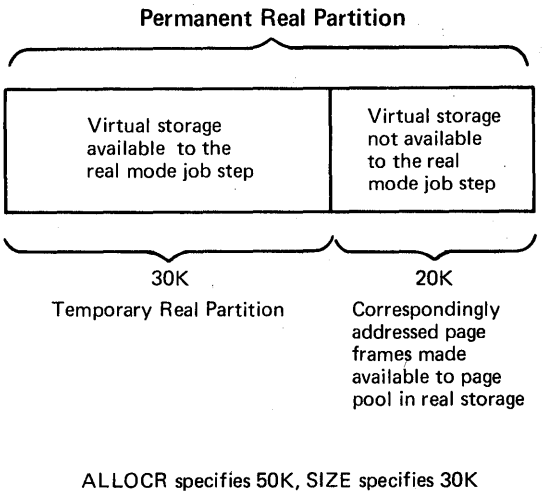


Figure 80.10.3. Use of the SIZE parameter for a real mode job step

When a job step is initiated in a real partition, all the page frames with real storage addresses equal to the virtual storage addresses in the temporary or permanent real partition must be available before the real mode program is loaded. If any of the required page frames are found to contain nonfixed pages of an executing virtual mode program, page-outs are scheduled for those nonfixed pages that have been changed and the real mode program is fetched at the completion of all the necessary page-out operations. If any required page frames contain a temporarily fixed page for an executing virtual mode program, loading of the real mode program is delayed until all the I/O operations that caused the temporary fixing are completed, the pages are unfixed, and any required page-outs have been performed.

The components of DOS/VS that must operate in real mode are CLTEP, QTAM message control programs, and QTAM message processing programs. A user-written program must execute in real mode in a DOS/VS environment if it:

- Contains a channel program that is modified while the channel program is active
- Is highly time dependent (involves a time-dependent I/O operation, such as stacker selection in a magnetic ink character reader)

- Uses an I/O appendage routine, unless the PFIX macro is used to fix the appendage routine and all pages the appendage references
- Contains a channel program for an I/O device type that is not supported by DOS/VS, unless channel program translation and page fixing are performed by the user
- Disables I/O interruptions prior to referring to user data and causes a page fault
- Changes the instruction address in a new PSW, for example in the new PSW for program check, and then forces a program check to put the system in supervisor state
- Executes privileged instructions, such as LOAD PSW or SET STORAGE KEY, that affect the physical state of the system
- Issues START I/O instructions without ensuring that the addresses used by the channel are real addresses.

Existing user-written programs that are operating under DOS Version 4 and that must operate in real mode under DOS/VS need not be modified to enable them to run in real mode. These programs may require modification for other reasons, as discussed in Section 80:50.

The page pool in virtual storage is all the address space in the real address area that is not allocated to the supervisor area or real partitions. The page pool consists of the main page pool area (the available virtual storage in the real address area located above the last real partition currently defined), the virtual storage allocated to each real partition whose corresponding virtual partition is currently active, and any virtual storage from permanent real partitions that is made available by use of the SIZE parameter. Thus, the number of virtual storage pages in the page pool in the real address area can vary as real mode programs execute and if real partition sizes are changed.

When no real partitions are defined, the page pool area and the main page pool area are the same. Figure 80.10.4 shows an example of the virtual storage allocated to the page pool when two real partitions are defined and a real mode program is executing for which the SIZE parameter was specified.

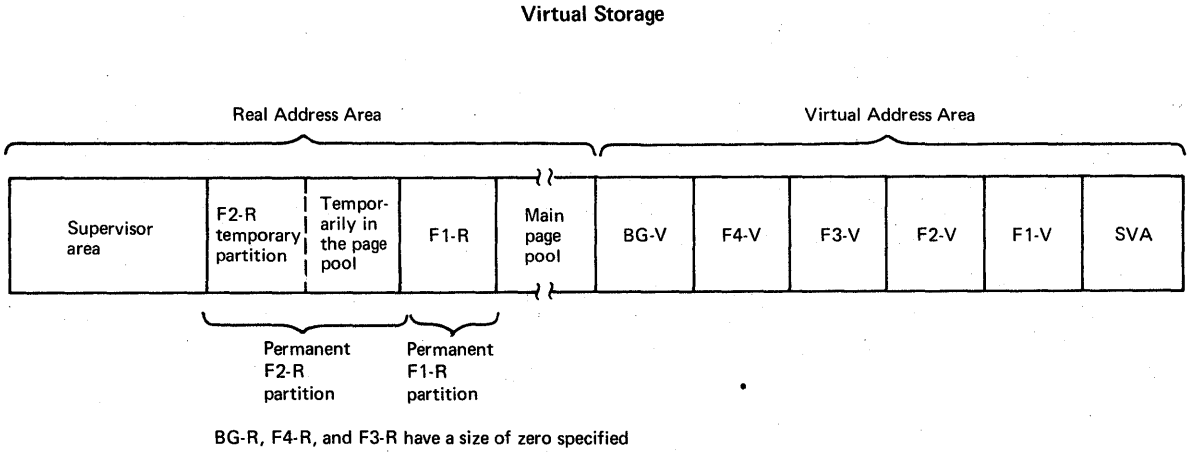


Figure 80.10.4. Sample allocation of virtual storage to the page pool in the real address area

The virtual storage in the page pool in the real address area is not available for allocation to executing programs. This condition exists because the page frames in real storage that have addresses equal to those of the virtual storage pages in the page pool in the real address area are available for allocation to programs that are operating in virtual mode. This is more fully discussed under "Real Storage Organization" below.

Storage Protection

Storage protection functionally equivalent to that provided in DOS Version 4 is supported as a standard feature in DOS/VS. Fetch protection is not supported in either DOS Version 4 or DOS/VS. Protect keys are assigned to defined areas in virtual storage in DOS/VS. The resident supervisor area is assigned protect key 0. The protect key assigned to a partition (virtual/real pair) depends on the number of partitions defined by the NPARTS parameter. If all five partitions are defined, keys 1, 2, 3, 4, and 5 are assigned to partitions BG, F4, F3, F2, and F1, respectively. If three partitions are defined, for example, keys 1, 2, and 3 are assigned to BG, F2, and F1, respectively.

An allocated page frame is set with the protect key assigned to the area or partition to which the page frame is allocated. All unallocated page frames are set with protect key 0. When a page frame is allocated to a virtual storage page within a virtual partition (as a result of a page fault, for example), the protect key assigned to that virtual partition is set for the page frame. The page frame retains this key until the page frame is allocated to another virtual partition, at which time it is set with the key of that partition, or until the page frame is unallocated and made available for reassignment.

Page frames allocated to a real partition are set with the protect key of the real partition when a job step is initiated in the real partition. The page frames retain this key until the job step terminates, at which time they are set with protect key 0. When the SIZE parameter is specified for a real partition, only the page frames assigned to the temporary partition are set with the key of the real partition.

The SVA is assigned storage protect key 0. A phase that is resident in the SVA usually executes using the PSW of the partition that invokes its execution and, thus, with the protect key of that partition in effect. This approach prevents user-written routines that execute in the SVA from modifying any real storage that is not associated with the partition that invokes their execution.

Checkpoint/Restart

The checkpoint/restart facilities provided in DOS/VS are functionally equivalent to those provided in DOS Version 4. However, the formula for calculating the number of tracks required to contain a given number of sets of checkpoint records is changed in DOS/VS. When the SIZE parameter has been specified for a virtual partition, the entire partition should be checkpointed so that the partition GETVIS area is included in the checkpoint.

In a DOS/VS environment, the operator must ensure that a checkpointed job step that ran in virtual mode is restarted in the same virtual partition and that a job step that ran in real mode is restarted in the same real partition that was used for the checkpoints. If the program was using any phases that were resident in the SVA, these phases must be present in the SVA in the same locations as they were when checkpoints were taken.

In addition, when a virtual mode job step that uses the page fixing option is restarted, the associated real partition must be located in the same place in the real address area and be of the same size as when the checkpoints were taken. Pages that were fixed in the real partition when the checkpoint was taken are automatically refixed when the virtual mode job step is restarted.

REAL STORAGE ORGANIZATION

A maximum of 6144K bytes of real storage is supported in DOS/VS. The organization of real storage in DOS/VS is shown in Figure 80.10.5. Real storage is divided into the resident supervisor area and the page pool area. The resident supervisor area is allocated lowest addressed real storage. The addresses in the resident supervisor area in real storage are equal to the addresses in the resident supervisor area in virtual storage. In effect, the page frames in the resident supervisor area in real storage are permanently fixed.

When a SEND address is not specified at system generation, any real storage between the address of the end of the generated supervisor and the address of the end of the supervisor area (which is rounded to a multiple of 2K when necessary) is allocated to the buffer area used for channel program translation (see discussion under "The Channel Scheduler" in Section 80:30). If a SEND address is specified, space between this address and the address of the end of the generated supervisor is not allocated to the channel program translation buffer area. A SEND address must be a multiple of 2K.

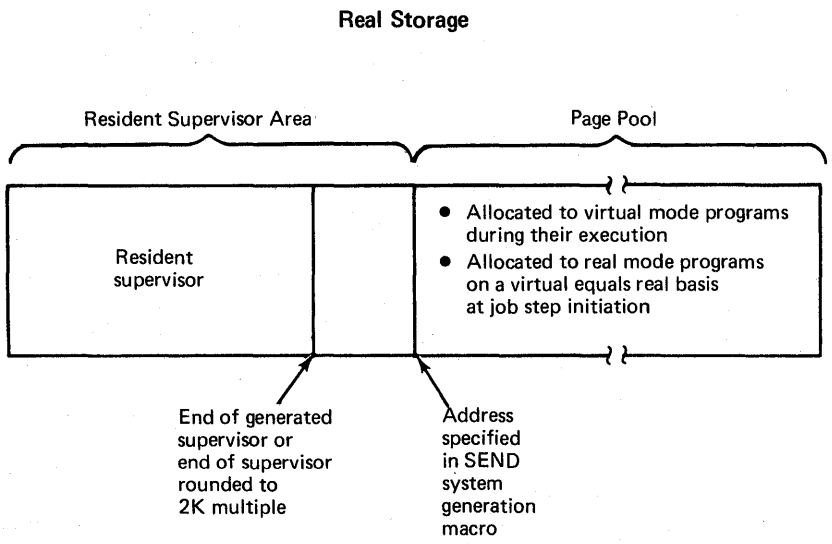


Figure 80.10.5. Organization of real storage in DOS/VS

All the real storage above the resident supervisor area is called the page pool and is available for allocation to virtual mode and real mode programs. Real storage for virtual mode programs that are executing in a virtual partition or the SVA is allocated from the page pool. When no real partitions are defined or if real partitions are defined but no real mode programs are executing, all real storage above the resident supervisor is in the page pool and available for allocation to virtual mode programs.

When one or more real mode programs are executing, all real storage above the resident supervisor that is not currently allocated to real mode partitions is in the page pool and available for allocation to virtual mode programs. The page pool in real storage is the same size as the page pool in the real address area in virtual storage.

Figure 80.10.6 shows how real storage is allocated for a sample five-partition environment in which five virtual partitions and three real partitions are defined. At the time shown, programs are operating in F1-V, F3-V, BG-V, and F2-R. The figure is meant to illustrate the way in which real storage is allocated and not necessarily a desirable partition configuration.

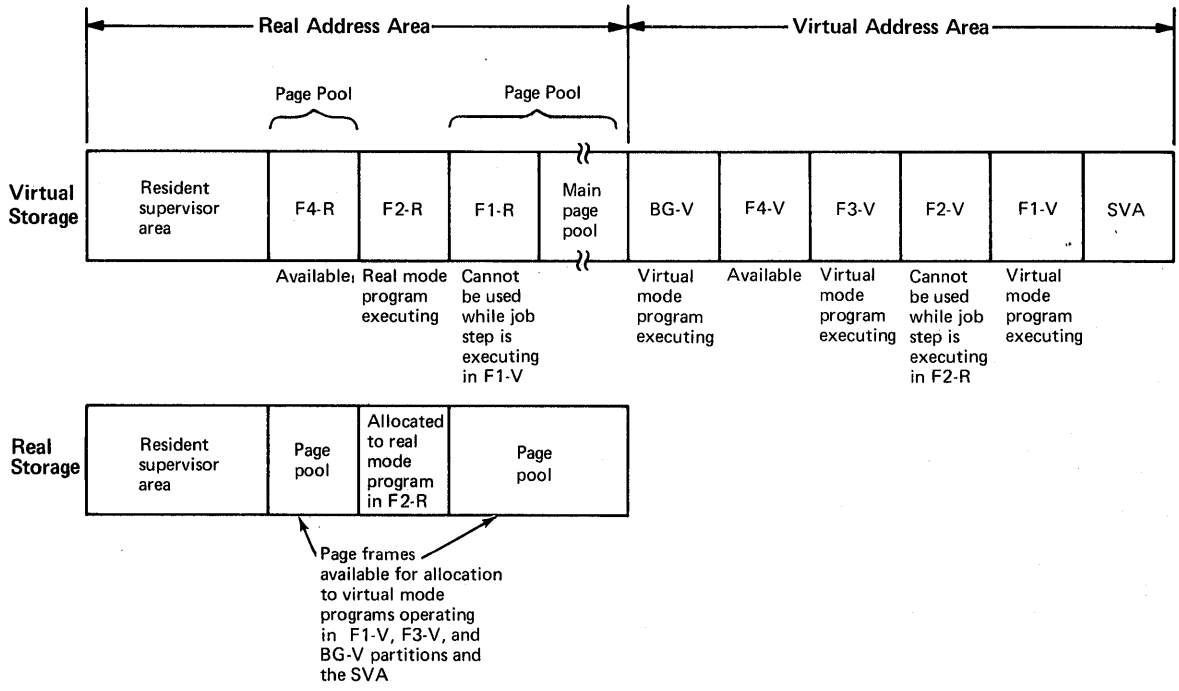


Figure 80.10.6. Example of real storage allocation for a sample five-partition system

EXTERNAL PAGE STORAGE ORGANIZATION

External page storage is used to contain the contents of pageable virtual storage, which in DOS/VS is that virtual storage contained in the virtual address area. The direct access storage allocated as external page storage is called the page data set. The page data set can be placed on one 2314, 2319, 3330-series, 3340, 3344, or 3350 direct access device and is assigned the symbolic name SYSVIS. The page data set need not reside on the same direct access device type as SYSRES.

The page data set is sequentially organized and must begin on a cylinder boundary. It must consist of only one extent and is formatted with records of 2K bytes called slots. Unblocked records without a key are written, and the track overflow feature is not used. The number of 2K slots per track and cylinder of each supported paging device type is shown below.

Device Type	Slots per Track	Slots per Cylinder
2314/2319	3	60
3330-series (Model 1, 2, or 11)	6	114
3340 or 3344 logical volume	3	36
3350 (native mode	8	240

There must be one slot in the page data set for each virtual storage page contained in the virtual address area. The number of slots required, therefore, is the storage size specified in the VSIZE parameter (rounded to a 2K multiple if necessary) divided by 2. The amount of disk space required for the page data set is calculated during system generation using the VSIZE specification. This amount of space is allocated to the page data set by the system, starting at the user-specified cylinder address.

The virtual address area is statically mapped on a one-to-one basis with the page data set extent. That is, the contents of any given virtual storage page are always placed in the same slot, and the first virtual storage page in the virtual address area is associated with the first slot in the page data set extent, etc., as shown in Figure 80.10.7.

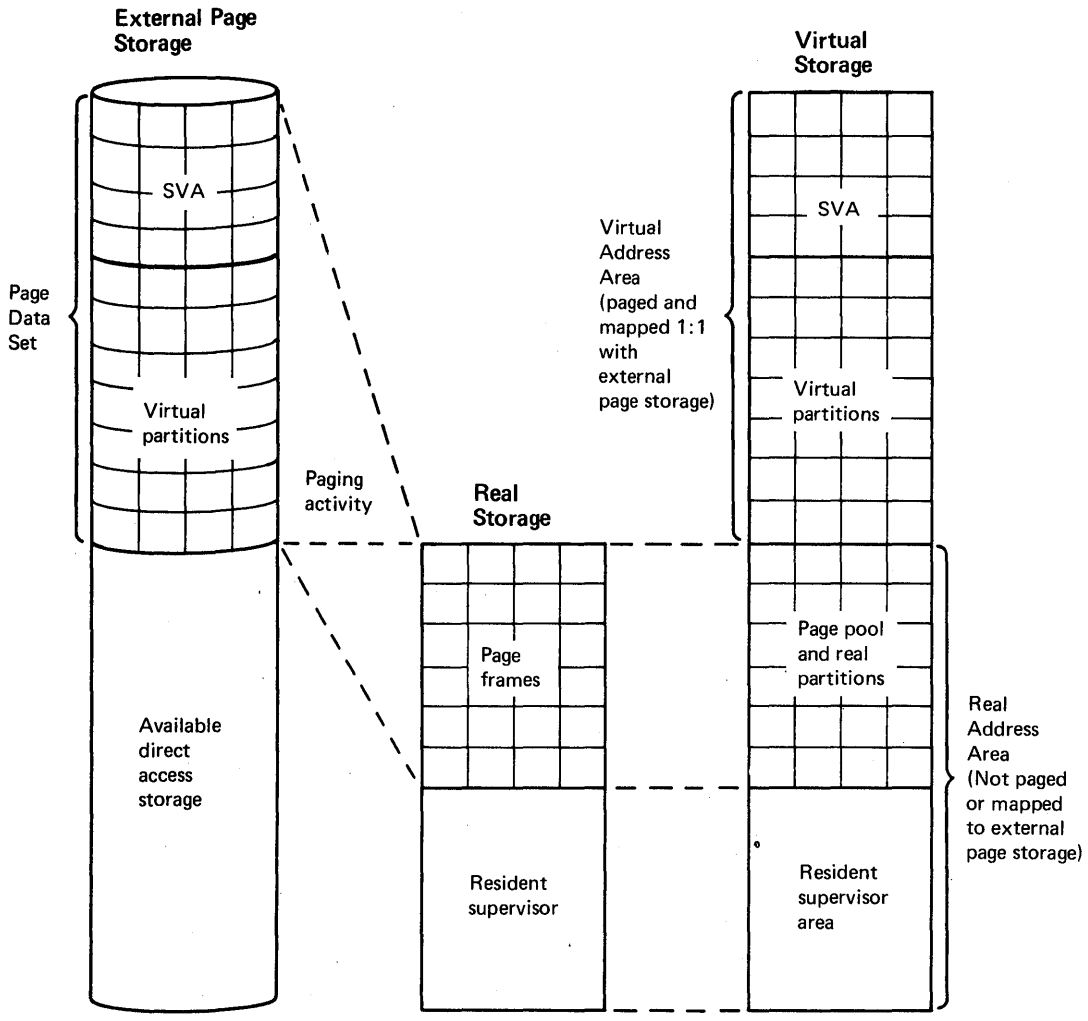


Figure 80.10.7. Relationship of virtual storage, real storage, and external page storage in DOS/VS

SYSTEM INITIALIZATION

The functions the operator performs during the initialization of a DOS/VS supervisor are like those required to initialize a DOS Version 4 supervisor except that for DOS/VS more parameters can be supplied by the operator, more functions are performed, and the supervisor to be loaded can be selected from among several that are contained on the system residence volume.

Once the IPL routines are loaded, the system enters a wait state. The operator must indicate the device that is to be used as the operator console by pressing the request key or END/ENTER key, as appropriate, on the selected device. This generates an interruption that causes the IPL program to assign the device that caused the interruption as the SYSLOG device.

If an IPL communication device list exists, the IPL program will check the list and will not accept the device that caused the interruption unless its address is in the list. This SYSLOG assignment remains valid until the next IPL and overrides any SYSLOG assignment

made during system generation. (Note that for a one-partition DOS/VS system, a printer cannot be assigned as the SYSLOG device during IPL.)

The IPL communication device list facility can be used to restrict the devices that will be accepted as a SYSLOG device. This facility is particularly useful for DOS/VS installations with locally attached terminals, such a 3277 displays. It can be used to prevent a device that is not under the control of the operator from being assigned as the SYSLOG device.

To use the optional IPL communication device list facility, a phase consisting of the addresses of the acceptable SYSLOG devices (and any device that can be used to submit IPL commands) must be assembled and link edited to the system core image library with the phase name $$A$CDL0 assigned. Up to eight addresses can be specified. The IPL program checks for the presence of this phase, and automatically loads it into real storage if it is present in the system core image library.

The capability of manually overriding an entry in the IPL communication device list or manually creating such a list (if one does not exist) when the first IPL wait occurs is provided. These assignments are temporary and apply only to the current IPL.

DOS/VS supports an optional supervisor select facility that is not provided in DOS Version 4. The DOS/VS system residence volume can contain multiple supervisors. The one to be loaded during an IPL is selected by the operator if the default supervisor (with phase name $$A$SUP1) is not to be used. This facility enables an installation to have multiple supervisors with different options specified without having to maintain multiple system residence volumes and having to change system residence volumes in order to change supervisors.

During the IPL procedure, the resident supervisor selected (via the SYSLOG device) is loaded into real storage after which the system is again placed in an enabled wait state. The operator must indicate the I/O device that is to be used to communicate IPL commands to the operating system by generating an interruption from that device. The device established as the SYSLOG device, a card reader that was or was not assigned as the SYSRDR device, or a 3540 Diskette Input/Output Unit (containing unblocked IPL commands) can be used as the IPL communication device.

The following commands, not defined in DOS Version 4, must or can be specified either during or following a DOS/VS IPL:

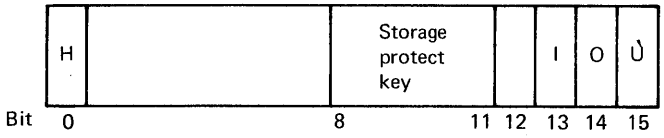
- DPD - Defines the page data set (SYSVIS), overrides parameters supplied at system generation, or indicates whether formatting is required (must be specified at the end of the IPL procedure). A re-IPL is required to change the I/O device assigned to SYSVIS.
- ALLOCR - Defines or overrides the system generation definition of real partition sizes (optional)
- PRTY - Overrides the partition dispatching priorities established during system generation (optional)
- CAT - Assigns an I/O device to the SYSCAT file (VSAM master catalog) or overrides the system generation assignment (optional). A re-IPL is required to change the I/O device assigned to SYSCAT.
- SET SVA - Overrides the SVA and system GETVIS area sizes specified at system generation (optional). A re-IPL is required to change these sizes.

- SET SDL - Indicates the SVA is to be created and written in the page data set (optional). A re-IPL is required to recreate the SVA if system directory list entries are to be added or physically deleted.

The segment table and page tables required to describe the virtual storage size indicated at system generation are built in the supervisor and initialized during system generation. When required, the page tables are modified during system initialization to reflect unusable virtual storage between the address of the end of real storage and the beginning address of the virtual address area that exists because real storage is smaller than the RSIZE value specified at system generation.

All segment table entries have their invalid bit turned off to indicate that the required page tables exist. The segment table is never modified during system operation. The virtual storage defined at system generation need not be a multiple of 64K. However, the segment table will define a virtual storage that is a 64K multiple since each entry represents a 64K segment.

There is one full length (32 entry) page table for each segment defined by the segment table. The format of a page table entry for a virtual storage page that does not have a page frame assigned is shown in Figure 80.10.8. Bit 0 in a page table entry is used to indicate invalid (unusable) address space within the virtual storage size defined by the segment table. For example, when the user-specified virtual storage size is not a multiple of 64K, the page table entries for pages between the end of specified virtual storage and virtual storage defined by the segment table will have bit 0 on to indicate unusable virtual storage.



Bit	
0	A one indicates invalid (unusable) address space within the defined virtual storage.
8-11	When bit 13 is a one, these bits contain the storage protect key of the partition (virtual or real). If bit 13 is a zero, bits 0 to 12 contain the high-order 13 bits of the address of the assigned page frame.
13	Invalid bit off indicates real storage address in bits 0 to 12 can be used for translation. Invalid bit on indicates entry cannot be used for translation.
14	Always zero.
15	User bit indicates whether a page-in is required. A one indicates a page-in is not required.

Figure 80.10.8. Format of a page table entry for a page without a page frame assigned

Bits 8-11 contain the storage protect key assigned to the virtual storage page that the page table entry describes. This storage protect key is set for a page frame when it is allocated to a virtual storage page and the address of the page frame is inserted into bits 0 to 12 of the page table entry. The invalid bit (13) indicates whether the page table entry can be used for address translation and the user bit (15) indicates whether a page-in is required when a page frame is assigned.

When a job step is initiated, all user bits in the page table entries for the partition are turned on to indicate the fact that a page-in is not required the first time a page frame is assigned to any virtual storage page in the partition. The first time a page is paged out, the user bit is turned off in the appropriate page table entry. The contents of page table entries after system initialization are shown in Figure 80.10.9.

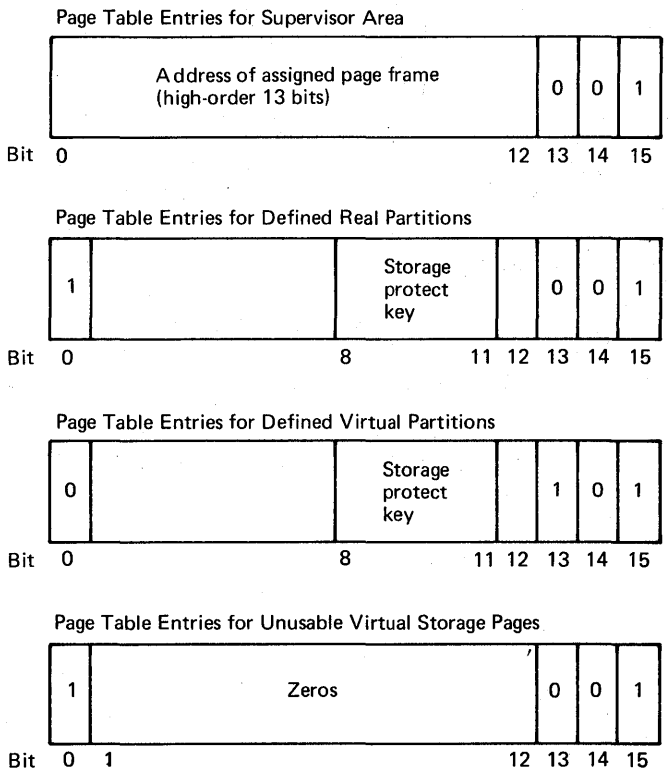


Figure 80.10.9. Contents of page table entries after system initialization

The page data set is opened, initialized, and, if necessary, formatted with slots during IPL. If the page data set was not defined at system generation (in terms of device address, beginning cylinder address, and, optionally, disk pack volume serial number) via the DPD macro, the operator must define it during IPL using the DPD operator command. The system generation specification can also be overridden during IPL using the DPD command. The direct access device and disk pack assigned to SYSVIS during IPL cannot be changed without a re-IPL.

The page data set extent must be formatted with 2K slots during IPL the first time the page data set is used. Thereafter, the formatted page data set can be reused without reformatting. Reformatting is required if the size of the page data set is extended or if a new extent (on the same or a different volume) is specified. The DPD command always must be specified at IPL. It indicates whether formatting of the page data set is required and the end of the IPL procedure.

During the IPL procedure, each direct access device listed in the PUB table is tested for operational status. Any disk device that is not operational is marked as not being available (device down indication) in its PUB table entry.

The PUB table is also scanned during IPL for printers that have a forms control buffer (FCB) and Universal Character Set buffer (UCB), that is, for 3203, 3211, and 5203 device types. The standard FCB image is automatically loaded into any of these types of printers in the configuration. The A11 print chain image is automatically loaded in the UCB of a 3211 and the AN print chain image is loaded into a 3203 or 5203 UCB.

Note that no attempt is made to load the FCB of dummy printer devices defined for POWER/VS use. A check is made to determine whether the printer exists before buffer loading is initiated.

The SYSBUFLD program is provided in DOS/VS, as in DOS Version 4, to load an FCB or UCB buffer image contained in a core image library or the SYSIPT device between two job steps or jobs during system operation. The UCB job control command is provided to load the UCB buffer of a 1403. In DOS/VS, however, FCB and UCB buffers can be loaded in other ways during system operation.

The LFCB and LUCB attention commands are provided in DOS/VS to enable the operator to load an image into an FCB or UCB, respectively, at any time during system operation. In addition, the LFCB macro can be issued in an executing program to load an FCB. The operator is notified when FCB loading via the LFCB macro is completed. FCB and UCB images to be loaded via the LFCB and LUCB commands and LFCB macro must be contained in a core image library.

Care should be exercised in the use of the LFCB command and macro and the LUCB command while a printer is actually printing as there is no way of knowing exactly when the printer will complete printing using the current FCB or UCB image. In addition, when an LFCB or LUCB command is issued, initiation of any other I/O operation is suspended until processing of the command is completed. Therefore, these commands should be used with caution when teleprocessing devices or devices such as a 1275 or 1419 are also in operation.

The LFCB and LUCB commands can be used, for example, to change the FCB or UCB image when printing has begun with the wrong image loaded. The LFCB macro could be issued in a user-written abnormal termination routine that causes a dump to be written to a 3211 printer for which the indexing facility was being used. A certain number of characters can be lost on each line of the printed dump unless an FCB image that does not specify indexing is used for the dump.

The SETPRT macro provides a way of specifying printer setup characteristics for 3800, 3211, and 3203 Model 4 printers during program execution. It can be issued by user-written programs and control program routines. The QSETPRT macro can be used to determine the printer setup concurrently in effect for a 3800, 3211, or 3203 Model 4 and/or to build a parameter list that can be passed to a SETPRT macro.

After the IPL procedure has been completed and the job control program has been loaded into the virtual background partition, the job control program fetches an exit routine named $SYSOPEN to be executed as an overlay phase. A user-written routine can replace the dummy $SYSOPEN phase that is provided in the system core image library of the DOS/VS distribution volume.

The $SYSOPEN exit can be used in an installation to perform security checks after IPL, such as determining who performed the IPL, checking that the correct DOS/VS system residence volume is mounted, and whether the correct date was entered. This capability is not provided in DOS Version 4.

The user-written $SYSOPEN overlay phase, which executes with storage protect key zero in effect, must be self-relocating, not larger than 4K bytes, and perform any required I/O operation using the EXCP macro. I/O operations can be performed and SVC instructions can be issued in user-written $$B-transient routines. Any routines called by the $SYSOPEN exit routine must be present in a core image library for which a device assignment was made prior to IPL.

The job control program cannot read any job control statements while the $SYSOPEN exit routine is being executed. Thus, if a labelled file is opened by the exit routine, the labels must be present in the standard label area, partition label area, or user label area and the device on which the file resides must have had an I/O assignment before the IPL. Data can be also read from and written to the system console.

If the USERID=id parameter of the FOPT system generation parameter is specified, the message issued at the completion of the IPL procedure contains a supervisor identification (IPL COMPLETE FOR DOS/VS REL XX.X ECLEVEL=nn).

80:15 MAJOR COMPONENTS

The major control and processing program components of DOS/VS are shown in Figure 80.15.1. Components that are identified as SCP are distributed as part of DOS/VS. Type I programs, program products, and emulator programs are not distributed with DOS/VS and must be obtained separately. The DOS/VS COBOL Compiler and the DOS/VS Sort/Merge program products execute only in a DOS/VS environment.

CONTROL PROGRAMS (SCP)

- Job Control**
- Supervisor
- Data Manangment
  - Physical IOCS
  - Logical IOCS
    - SAM
    - DAM
    - ISAM
    - VSAM**
    - BTAM
    - QTAM*
    - VTAM**
- Recovery Management
  - MCAR/CCH/RMSR/EREP
  - OLTEP*
  - Problem Determination Aids (PDAIDS)
  - System Debugging Aids (SDAIDS)

PROBLEM PROGRAMS (SCP and PP)

- Service
  - POWER/VS** (SCP)
  - Linkage Editor (SCP)
  - Librarian (SCP)
  - System Utilities (SCP)
  - Access Method Services for VSAM** (SCP)
  - ASCII Magnetic Tape Utilities (PP)
  - 1288 Basic Unformatted Read Support (PP)
  - DOS Sort/Merge 5743-SM1 (PP)
  - DOS/VS Sort/Merge 5746-SM2 (PP)
  - Integrated Emulators
    - Model 20 (SCP)
    - 1401/1440/1460 (SCP)
    - 1410/7010 (SCP)
  - 3704/3705 System Support Programs (SCP)
  - Subsystem Support Services (SCP)

PROBLEM PROGRAMS (SCP and PP)

- Language Translators
  - Assembler (SCP)
  - RPG II (PP)
  - Full ANS COBOL Version 3 and Library (PP)
  - Subset ANS COBOL (PP)
  - DOS/VS COBOL Compiler and Library (PP)
  - FORTRAN IV Library - Option 1 (PP)
  - PL/I Optimizing Compiler (PP)
  - PL/I Resident Library (PP)
  - PL/I Transient Library (PP)
  - ITF PL/I (PP)*
  - ITF BASIC (PP)*

PROBLEM PROGRAMS (Type I and User-written)

- Language Translators and Service Programs
  - COBOL D (360N-CB-452)
  - COBOL LCP (360N-CV-489)
  - Full ANS COBOL Version 2 (360S-CB-482)
  - FORTRAN F (360N-FO-479)
  - FORTRAN F Library (360N-LM-480)
  - PL/I D (360N-PL-464)
  - Sort/Merge (360N-SM-483)
  - Group 1 Utilities (360N-UT-461)*
  - Group 2 Utilities (360N-UT-462)*
  - Group 3 Utilities (360N-UT-463)*
  - Multiprogramming Support Utility Macros (360N-UT-471)*
- General
  - Application-oriented program products (some run in virtual or real mode and some only in real mode)
  - User-written application programs

*Must execute in real mode
**Must execute in virtual mode

Figure 80.15.1. Control and processing program components of DOS/VS

The components of DOS/VS and DOS Version 4 are similar. Existing components have been extended to support a virtual storage environment and other new features. DOS/VS supports all the system logical units, system libraries, and private libraries that are supported in DOS Version 4. In addition, DOS/VS supports a page data set (SYSVIS), a procedure library which is part of SYSRES, and a master catalog for VSAM files (SYSCAT).

The new features of DOS/VS and the most significant functional differences between DOS/VS and DOS Version 4 are presented in the discussions that follow.

A new core image library organization and a second level directory facility are implemented in DOS/VS in order to reduce program phase locate time and eliminate certain restrictions that exist for previous DOS releases. The organization of a core image library and its directory in DOS/VS is similar to the partitioned data set (PDS) organization implemented in OS for libraries. The five subdirectories of the system core image library (transient directory, OPEN directory, etc.) and the system work area used by librarian routines are deleted from the SYSRES volume as a result of the new core image library organization.

In DOS/VS, the directory of a core image library consists of directory records written in count, key, and data disk record format, as shown in Figure 80.15.2. The key area is 8 bytes in size and the data area is 256 bytes. A directory data record contains a number of variable-length directory entries, each of which provides required information about a program phase as it exists in the associated core image library. Included in a directory entry are the name of the phase and its location in a core image library in relative track address and record number (TTR) format.

The directory entries within a core image library directory are maintained in ascending alphabetic sequence by phase name. The highest phase name in each directory record (256-byte data area) is written in the 8-byte key that precedes that data area. A minimum of two tracks is required for a core image library directory.

Text data and control information for a phase are written in 1K (1024) byte records (instead of 1688- or 1504-byte records, as in previous DOS releases), regardless of the type of direct access device on which the core image library resides. All text records for a given phase are written first. All relocation list dictionary data for the phase is written after the last text record.

Programs are placed in a core image library in the sequence in which they are loaded and are not maintained in any sequence. When a program is cataloged in a core image library, it is placed in available space after the last existing program in the library. However, the directory entry for the program is inserted in alphameric sequence in the directory for that core image library. (If a directory entry with the same phase name as the new cataloged phase is present in the directory, it is deleted.)

This organization enables the directory entry for a program to be located by a channel search of a directory track. A SEARCH KEY HIGH/EQUAL command with the name of the desired program (phase) is used to locate the directory record that contains the required directory entry and only one directory record need be read into storage. In DOS Version 4, each directory record, beginning with the first record, must be read into storage in order to be searched for the required directory entry.

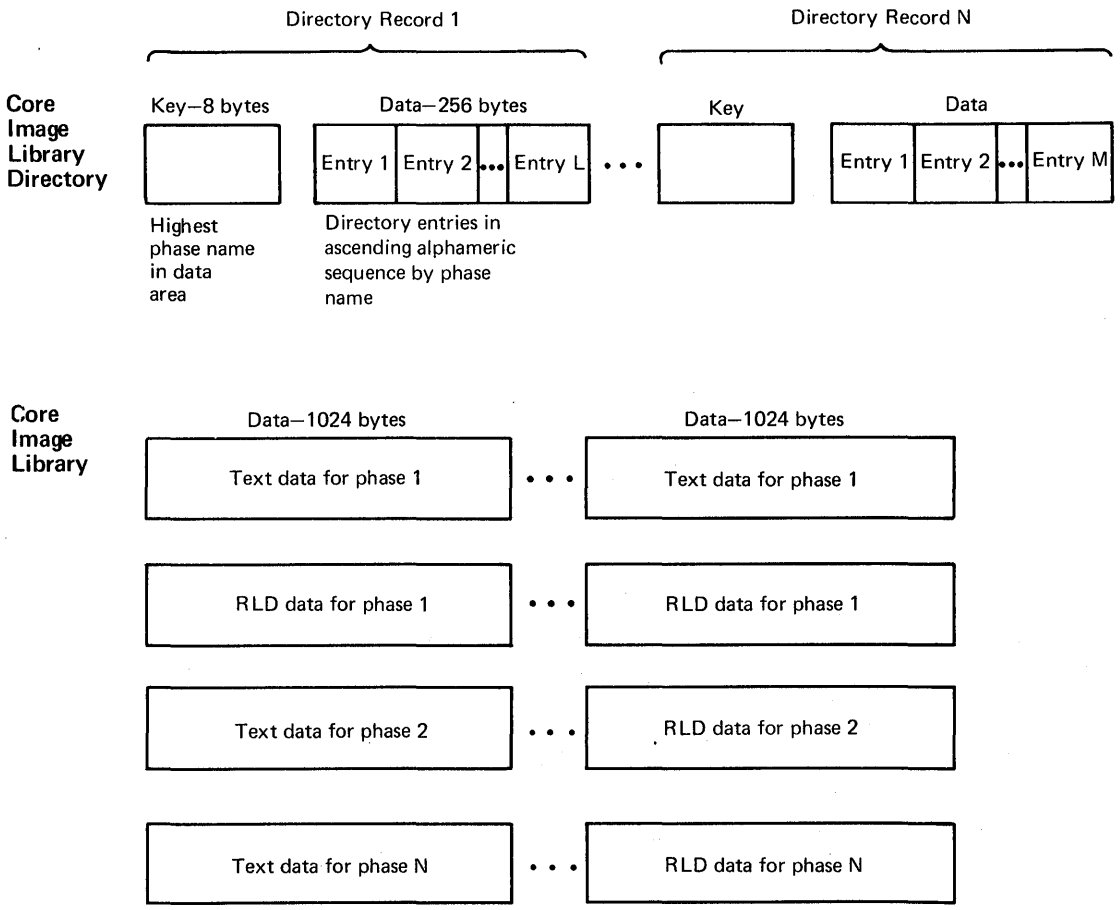


Figure 80.15.2. Organization of a core image library

The new core image library organization is used for the system core image library and all private core image libraries. The first ten tracks of a private core image library volume are no longer required for subdirectories, a librarian work area, etc. Only one track is required for directory records.

In addition, in DOS/VS a private core image library can be placed on a direct access device type that is different from that used for the system core image library and need no longer begin on a cylinder boundary. This latter capability and the availability of tracks in the first cylinder of the volume (as a result of the elimination of subdirectories) permit phases to be stored in the first cylinder of a private core image library. The time required to locate a phase contained in the first cylinder is reduced by the elimination of a seek after the directory is searched. Therefore, as many of the most frequently used routines in a private core image library as will fit should be stored in the first cylinder.

The second level directory facility is implemented to reduce the search for a required directory entry to one directory track. This facility is provided for both nonmultiprogramming and multiprogramming systems. A system second level directory for the system core image library is always created. One private second level directory is created for each private core image library only if the private second level directory option is specified at supervisor generation. Second

level directories are contained in the supervisor. The number of entries in the system second level directory and the private second level directories can be specified at system generation. The minimum number of entries for each is five.

The second level directory for a core image library contains the highest phase name on each track in the directory for that core image library (one entry for each track in the directory if there are enough entries in the second level directory). In order to determine the directory track at which the search for a directory entry is to begin, the second level directory for a core image library, if any, is inspected before a directory search is performed on disk. For best performance, therefore, the number of entries specified for a second level directory should be equal to or greater that the number of tracks in the core image library directory. If a private second level directory is not present for a private core image library, the search for the specified phase name begins at the first track in the directory.

Entries are placed in the second level directory for a private core image library at the time the job control program processes the ASSGN statement for the private core image library. The second level directory for the system core image library is completed during IPL after assignment of the SYSRES device has been made. The $MAINDIR service routine is called to place entries in second level directories.

As a result of the implementation of directory lists and second level directories, the sequence of searching to determine the location of a phase that is requested by a partition is different than in DOS Version 4. When the requested phase is not a $-phase and the new SYS=YES parameter is not specified in the FETCH or LOAD macro, the search sequence for the directory entry of the phase name specified is as follows:

- Local directory list in the partition, if one is specified in the FETCH or LOAD macro. If an active entry for the phase name is found in the local directory list, searching terminates and the phase is loaded from the private core image library assigned to the partition or from the system core image library.
- Link directory on disk of the private core image library for the partition, if a private core image library is present for the partition
- Link directory on disk of the system core image library if it is present and a private core image library is not present
- Private second level directory in the supervisor (if a private core image library and such a directory are present for the partition)
- Private core image library main directory (of cataloged phases) on disk (if a private core image library is present). The search begins at the directory track found during the private second level directory search (if such a directory is present) or at the first track of the main directory. If the phase name is found in the main directory, the search terminates and the phase is fetched from the private core image library.
- System directory list in the SVA. If the phase name is found in this list, searching terminates. The phase is then fetched from the system core image library if it is not resident in the SVA.
- System second level directory in the supervisor

- System core image library main directory on disk. The search begins at the directory track indicated by the search of the system second level directory. The phase is fetched from the system core image library if the phase name is found.

If the FETCH or LOAD macro requests a $-phase or if the SYS=YES parameter is specified for a phase that is not a $-phase, the search sequence is as follows:

- Local directory list if one is specified in the FETCH or LOAD macro
- System directory list in the SVA
- System second level directory in the supervisor
- System core image library main directory on disk beginning with the directory track specified in the system second level directory
- Private second level directory in the supervisor if a private core image library and such a directory are present for the partition
- Private core image library main directory on disk if a private core image library is present. The search begins at the directory track found in the private second level directory, if any, or at the first track of the main directory.
- Private core image library link directory on disk if a private core image library with a link directory is present
- System core image library link directory if it is present and a private core image library is not present

The DE=YES parameter, not supported in DOS Version 4, can be specified on a FETCH or LOAD macro to indicate that a pointer to a directory entry is provided by the macro in place of the phase name. If the specified directory entry is active, it is used and all searching for the directory entry is bypassed. If the directory entry is not active, it is located using one of the search sequences listed above and the local directory list entry is completed by the supervisor.

A TXT parameter for the LOAD macro, not supported in DOS Version 4, determines whether the phase is actually loaded into virtual storage after it is located. If TXT=NO is specified, the local directory list entry is completed without loading of the phase. TXT=NO can be specified to cause a directory entry to be filled in for later use in FETCH/LOAD macros without taking the time required to load the text. It is also a means by which a test can be made for the presence of the phase in any library and to determine the specific library in which the phase resides.

The new core image library organization, second level directories, and directory lists that are supported in DOS/VS can result in a significant reduction in the time required to locate a phase, particularly when the system core image library and/or private core image libraries are relatively large. In addition, supervisor transient routines and other frequently used system programs, such as job control, the linkage editor, and librarian routines, can be fetched more quickly. This occurs because when the DOS/VS system core image library is built, all $$- and $-phases are placed at the beginning of the library immediately after the directory. This can minimize the disk arm movement required to access these phases.

The contents of the DOS/VS SYSRES file, which always starts at track 1 in cylinder 0, is the following:

- System Directory
- IPL retrieval program
- Core Image Library Directory
  Directory of cataloged phases (main directory)
  Directory of linked phases (link directory)
- Core Image Library
- Relocatable Library Directory (optional area)
- Relocatable Library (optional area)
- Source Statement Library Directory (optional area)
- Source Statement Library (optional area)
- Procedure Library Directory (optional area)
- Procedure Library (optional area)
- Label cylinder(s)

One label cylinder is allocated for each direct access device type supported as a SYSRES device except the 3340. For the 3340, two adjacent cylinders are allocated as label cylinders. The second cylinder can be used to store permanent standard labels for all partitions defined for the system. Two cylinders are allocated for the 3340 since for a five partition system, only two tracks would be available for permanent standard labels for all partitions if only one cylinder were allocated.

As a result of the support of two more partitions in DOS/VS than in DOS Version 4, the organization of the label cylinder on the SYSRES file varies depending on the number of partitions the supervisor is designed to support, as shown in Figure 80.15.3. Only the first of the two label cylinders for a 3340 is shown. The second label cylinder for a 3340 contains 12 tracks for standard labels for all partitions.

The IPL program is modified to determine whether the number of partitions supported by the supervisor being loaded is equal to the number of partitions for which the existing label cylinder on SYSRES is organized. If not, the required format is established for the label cylinder and the existing contents are moved within the label cylinder as necessary.

This IPL facility enables the supervisor on an existing SYSRES volume to be replaced by a supervisor that supports a different number of partitions than the replaced supervisor without having to recreate the SYSRES file on a new volume in order to build a new label cylinder. For example, if a supervisor that supports five partitions replaces a supervisor that supports three, any standard labels on label cylinder tracks 6-9 (as shown in Figure 80.15.3) would be destroyed by the storing of labels for the additional two partitions supported, if the label cylinder were not reorganized first.

DOS/VS supports all the primary operator console devices that are provided for Models 115 to 158. However, the display console for the Model 158 is supported in printer-keyboard mode only and the 3213 Printer is required as a hardcopy output console device. Display mode operations for the Model 158 display console (but not the 3213 Printer or light pen) are supported by the Advanced Functions-DOS/VS program product.

Display operator console (DOC) support is provided for the display console for Models 115 and 125, for the display console of Models 138 and 148 operating in 115/125 Console Display Emulation or normal (3277) display mode, and a 3277 display station attached to a byte multiplexer channel via a 3272 Control Unit. Model 115, 125, 138, and 148 display consoles are also supported operating in printer-keyboard mode.

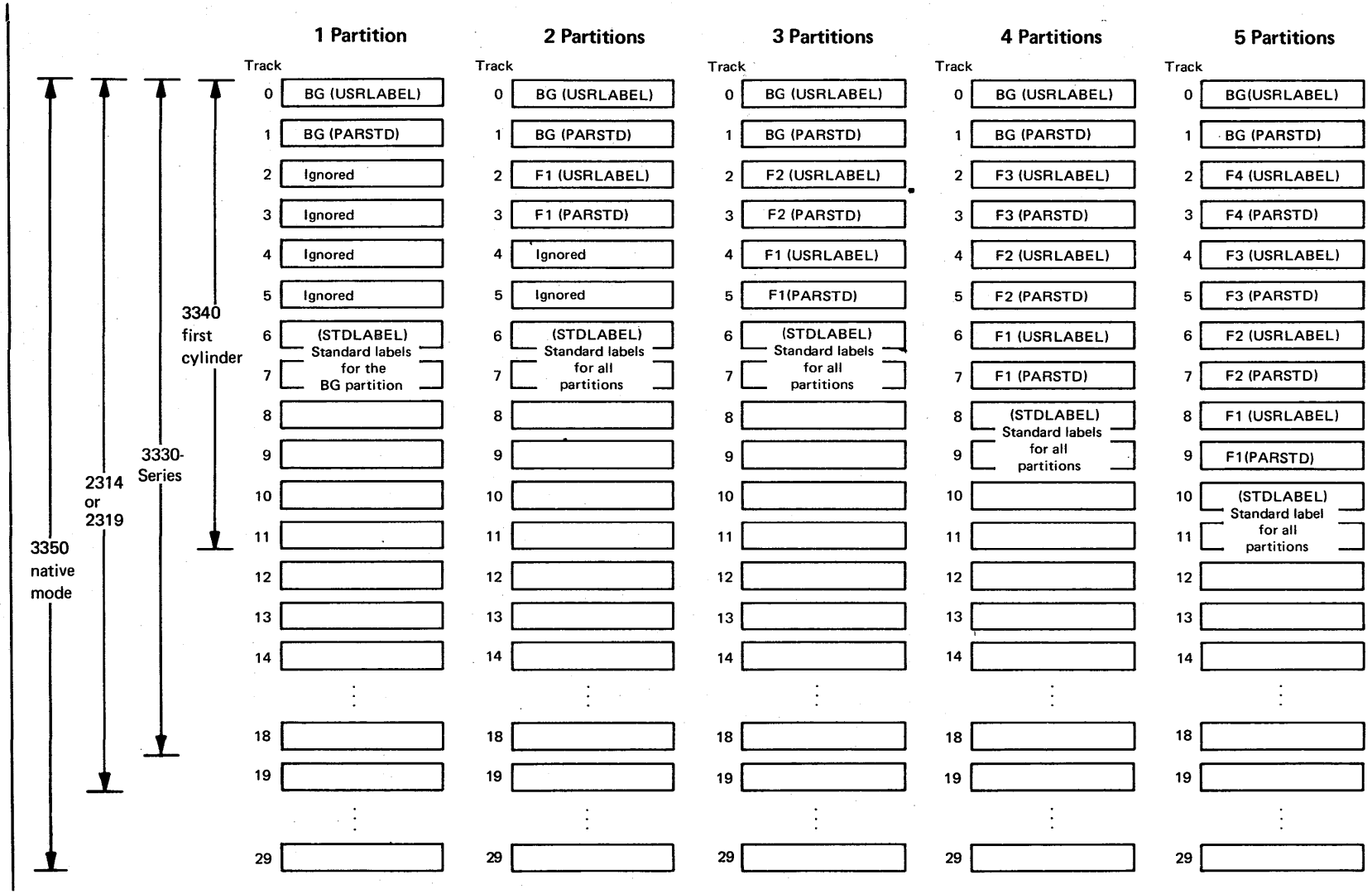


Figure 80.15.3. Layout of the first (3340) or only label cylinder for the DOS/VS SYSRES file

DOC support is automatically included in supervisors that are generated for Models 115, 125, 138, and 148. Normal display mode support is included for Models 138 and 148. Support of 115/125 Console Display emulation mode must be specifically requested for a Model 138/148 console. The twelve program function keys that are available on the display console for Models 138 and 148 are not supported by DOS/VS.

DOC support of the display consoles for Models 115 and 125 and 115/125 Console Display Emulation mode for Model 138 and 148 consoles provides a screen display of 16 lines of 56 characters each, only the first twelve of which are available to the operator. The last four lines are a system hardware status display area that is used by customer engineers. On the Model 138/148 display console, which has 24 lines of 80 characters each, the 56 character lines are centered within the 80-character line and every other line position is used to display the twelve operator lines.

The twelve operator lines are the following (from top to bottom of the screen):

- Message area of eight lines that is used to display messages from the DOS/VS system and user-written programs. A message longer than 56 characters is continued on the next line
- Instruction line for the display of messages to inform the operator of incorrect usage of the K control command, which is used by the operator for screen contents control and to indicate operating conditions of which the operator should be aware
- Entry area of two lines that is used to display entered commands
- Warning line for the display of messages regarding problems that must be resolved by the operator

Display operator console support is also provided for Model 138 and 148 display consoles operating in normal display (3277) mode and the 3277 Display Station. DOC support of the 3277 as an operator console is optional for Models 115, 125, 135, 145, 155 II, and 158. DOC support of display mode for Models 138 and 148 is included by default. When DOC support of the 3277 is included for a Model 115, 125, 138, or 148, DOC support of 115/125 display mode cannot be included in the same supervisor (that is, 115/125 Console Display emulation mode cannot be used for a Model 138/148 console or the display console of a Model 115/125 cannot be used as the operator console).

The screen of a 3277 display or a Model 138/148 display console operating in display mode is divided into the following areas (from top to bottom):

- Message area of 20 lines for display messages from the DOS/VS system user-written programs. A message longer than 77 characters is continued on the next line.
- Instruction line for the display of messages to inform the operator of incorrect usage of the K control command
- Entry area of two lines that is used to display entered commands
- Warning line for the display of messages regarding problems that must be resolved by the operator

The K and D commands are provided to enable the operator to control screen operations when display mode support is used. When the message area on the screen becomes full, the K command must be used to delete

some or all of the messages. The D command is used to redisplay messages contained in the hard-copy file that were deleted previously.

The audible alarm on the Model 115, 125, 138, and 148 display console is sounded when the operator must respond to a message, when the operator makes an error entering the K command, or when the message "MESSAGE WAITING" is displayed.

When DOC support of 115/125 display mode is utilized, the 5213 Console Printer is optional for Models 115 and 125, while the 3286 Model 2 or 3287 Model 1 or 2 Printer is optional for Models 138 and 148. When present, the 5213 or 3286 is used as a hard-copy console printer and is not uniquely addressable. All lines displayed on the display console are automatically written to the hard-copy printer.

If a console printer is present in a Model 115, 125, 138, or 148 configuration, use of a hard-copy file on disk is optional when DOC support of 115/125 display mode is utilized. If a console printer is not present, use of the hard-copy file is required when using 115/125 display mode support.

When the hard-copy file is present, each message displayed on the screen and all information keyed in by the operator are automatically written in the hard-copy file by display console support. This file must be allocated an extent on the SYSREC disk device.

The 3286 or 3287 Printer is not supported for hard copy when normal display (3277) mode is utilized for a Model 138/148 display console. When a 3277 or a Model 138/148 console is used in display mode, a hard-copy file is required.

The hard-copy file must be created after the first IPL procedure and if the SYSREC file is damaged. This is accomplished by issuing the SET HC=CREATE command after the ready message appears. Whenever a new hard-copy file is created, the existing one is deleted.

The PRINTLOG utility program is used to write the hard copy file to the SYSLST device. All existing messages in the file or only selected messages can be written. Selected message types (all action, decision, information, etc.), messages associated with a specific job, those issued on a specific date, or those entered since the last running of the PRINTLOG utility can be printed.

The display consoles for Models 115, 125, 138, and 148 are also supported in printer-keyboard mode. When this mode is used, the display console (screen and keyboard) is treated like a 3210/3215 Console Printer. Each message is printed using two lines beginning with the first two lines. When the screen is full, the top six lines are automatically deleted and the remaining lines are moved up to leave space at the bottom of the screen for new messages.

A console printer (5213, 3286, or 3287) is optional for printer-keyboard mode operations. However, its use is recommended for hard-copy backup (if present, all data displayed on the screen is also written to the console printer), since a hard-copy file is not supported for printer-keyboard mode of operation. Messages that are automatically deleted cannot be redisplayed (since there is no hard-copy file) but are available on the hard-copy console printer. A console printer (5213, 3286 Model 2, or 3287 Model 1 or 2) cannot be uniquely addressed when printer-keyboard mode is used.

80:20 THE JOB CONTROL PROGRAM AND OPERATOR COMMANDS

THE JOB CONTROL PROGRAM

The DOS/VS job control program provides the same functions as the DOS Version 4 job control program and is extended to support a virtual storage environment, a job control exit facility, a cataloged procedures facility, and generic I/O device assignment. The job control program is also modified as required to support other new features of DOS/VS, such as the shared virtual area and the new core image library organization.

In a DOS/VS environment, the job control program executes in virtual mode in a virtual partition that must be a minimum of 64K. The job control program is modified to issue the seize system SVC, which causes all task dispatching to be suspended only when it is absolutely necessary. This change is designed to be of benefit to system performance by reducing the amount of serialized processing that is caused by the job control program.

The same job control statements and parameters are supported in DOS/VS and DOS Version 4 except for the following new and modified support in DOS/VS:

- The REAL and SIZE parameters (previously discussed) can be specified on the EXEC statement as can the PROC parameter, which is associated with the cataloged procedures facility discussed below.

- A VSAM code can be specified on a DLBL statement to indicate VSAM file organization and the BUFSD parameter can be included to override the buffer assignment made in the ACB in the program or when the VSAM file was defined using Access Method Services.

- The BLKSIZE parameter is added to the DLBL statement. This parameter can be specified only for sequentially organized disk files allocated to a 3350 or 3330 Model 11 that are defined using the DTFSD macro. RPS support must also be utilized. The BLKSIZE parameter can be used to override the block size specified in the DTF (BLKSIZE must specify a larger block size value than the DTF). This parameter enables a larger block size to be used for a file that was contained on a device type other than a 3350 or 3330 Model 11 without program modification and recompilation.

- The OVEND statement is provided for the cataloged procedures facility.

- F3 and F4 parameters are added to job control statements as required to provide support of the two additional foreground partitions that can be defined in DOS/VS.

- The PARTDUMP option can be specified in the OPTION statement. This option causes a dump that contains less system information than that provided by the DUMP option to be written to the SYSLST device when abnormal termination of a partition occurs. In addition to the contents of virtual storage in the partition, the PARTDUMP option causes the contents of the registers as well as the address and contents of the partition communication region, PUB table, PUB owner table, partition LUB table, JIB table, and partition DIB table to be written to SYSLST. The PARTDUMP option can also be specified as a standard job control option at system generation by including the DUMP=PART parameter in the STDJC macro.

- The SETPRT statement is provided to be used to specify printer setup characteristics for 3800, 3211, and 3203 Model 4 printers.

Part of the additional processing the job control program must perform in support of a virtual storage environment is initialization of the appropriate page table entries when a job step is initiated in a partition. As part of EXEC statement processing, the page table entries for the virtual partition or the real partition and its associated virtual partition are initialized. In addition, any copy blocks for the partition that are being used by the fast CCW translation routines (if this option is present) are released and temporarily fixed page frames associated with these copy blocks are freed. If the System/370 model being used has a translation lookaside buffer, its contents are invalidated.

When a virtual mode program is to execute, the invalid address bit (bit 0) in the page table entries for the virtual partition is set to zero. The invalid and user bits in the page table entries are set to one (both on). The page table for the associated real partition is not modified at this time since it is placed in its system initialization status (invalid address bits set to one and invalid bits set to zero) each time a real mode job step completes execution.

Since the invalid and user bits are on for all pages of the virtual partition except the first (which contains the save area), the first reference to any page in the virtual partition except the first will cause a page fault and the allocation of a page frame without a page-in.

Since the invalid bit is off for all virtual storage pages in the real partition, any reference to a virtual storage address that is contained in the real partition during execution of a virtual mode program will cause address translation to proceed using bits 0 to 12 of the page table entry. However, the presence of a one in bit 0 (high-order bit of the 24-bit real storage address) will generate a translated real address that is not present in any System/370 model supported by DOS/VS. An addressing exception interruption will occur.

In this manner, executing programs are prevented from accessing temporarily unusable virtual storage in a real partition when a program is executing in its associated virtual partition.

When a real mode program is to execute, page table entries for the real partition and its associated virtual partition are initialized as follows:

- Real partition - A valid real address is placed in bits 0 to 12 and the invalid bit is set to zero (off). The real address in an entry is equal to the address of the virtual storage page with which the entry is associated.

- Virtual partition - The invalid address bit is set to one (on) and the invalid bit is set to zero (off)

These settings enable address translation to be performed for virtual storage pages in the real partition such that a virtual storage address translates to an equal real storage address, and prevent access to the virtual storage in the associated virtual partition during real mode program execution.

Job Control Exit Facility

The job control exit facility is standard in DOS/VS multiprogramming systems. It enables a user-written routine to inspect each job control statement after it has been read and before it is processed by the job control program. The operand and comments fields can be modified (positions 11 to 71 of the job control statement) but the operation field cannot be changed in any way.

A user-written job control exit routine must be named $JOBEXIT. It must be reentrant and made resident in the SVA. The exit routine cannot issue any SVC instructions, perform any I/O operations, or request cancellation of the job step. The exit routine executes with storage protect key 0 in effect. A user-written exit routine replaces the IBM-supplied dummy $JOBEXIT routine that returns control to the calling routine without performing any function. The job control exit is always taken after each job control statement is read.

When the job control exit routine is entered, general registers indicate the location of the statement in a buffer, address of the partition communication region, address of the system communication region, address of the job control vector table, and return address to the job control program. Once the exit routine completes processing, it must place a return code of 0 in general register 15 if the job control statement is to be processed. Any other return code causes the statement to be treated as a comments statement.

Cataloged Procedures

The cataloged procedures facility is a standard feature of DOS/VS. It provides the capability of storing in a procedure library frequently used job control statements for job steps. Optionally, control statements and input data for system service and utility programs can also be placed in a cataloged procedure. Once a procedure is cataloged, it can be invoked via the EXEC job control statement/command and its statements and data are included in the input stream just as if they were physically present in the SYSRDR or, optionally, SYSIN or SYSIPT device.

When a cataloged procedure is used, job control statements in the procedure can be modified, if required, by job control statements in the input stream. Modifications are effective only for the duration of the job step to which they apply and do not affect the procedure as it is cataloged in the procedure library.

The cataloged procedures facility is an operational aid. It can be used to reduce the total number of input stream cards the operator must handle. This can speed up operations and reduce the possibility of errors caused by card mishandling. While an input stream also can be placed on tape or disk to minimize handling, this approach does not provide the flexibility of temporary modification by cards, as is provided for cataloged procedures.

Use of the cataloged procedures facility also reduces the total number of job control statements that must be created and maintained in the installation since the job control statements for frequently used job steps that do not require partition-dependent devices (assemblies, utilities, for example) need be stored only once and can be used by any number of different jobs, regardless of the partition in which the jobs execute.

The procedure library is a system library that is part of the SYSRES file. It consists of one or more complete cylinders, as determined by the user. Private procedure libraries are not supported. Use of the cataloged procedures facility is optional.

The IBM-supplied procedure library contains procedures for linking and deleting DOS/VS components during a system generation and four procedures (SDL, RPS, VSAMSVA, and VSAMRPS) that are provided for use during system operation. The procedures can be used during the creation of the SVA.

The SDL procedure contains the control statements that are required to place the directory entries of selected system phases in the system

directory list in the SVA. It is designed for use with a DOS/VS system without VSAM or RPS support. The RPS procedure contains control statements for loading selected system phases and RPS phases but no VSAM phases. The VSAMSVA procedure contains control statements for loading both the suggested system phases and VSAM phases but no RPS phases. If the VSAMSVA procedure is used, the SVA must be a minimum of 302K. The VSAMRPS procedure contains control statements for loading selected system phases, RPS phases, and VSAM phases. The contents of the supplied procedures can be modified using the PSERV librarian program.

The procedure library contains a directory which points to the location of each cataloged procedure. Each procedure has a unique name (up to eight characters in length) and consists of DOS/VS job control statements. Any POWER/VS JECL statements contained within a cataloged procedure are treated as comment statements. Optionally, a procedure can also contain SYSIPT data stored in 80-byte card image format.

As an installation aid, the procedure library also contains coded samples for procedures to delete and link system components, create standard labels, create private libraries, and define VSAM files.

The librarian functions provided for other DOS/VS system libraries are provided for the procedure library also via extensions to existing librarian programs. Cataloging, deleting, renaming, condensing, allocating and reallocating, setting the condense limit, checking the condense limit and automatic condensing, library copying, and library directory displaying are supported for the procedure library. In addition, procedure punching or displaying is provided via the PSERV program, which is provided in support of the cataloged procedures facility.

A cataloged procedure can contain statements for one or more job steps that are to be executed as part of the same job. Statements for job steps that are part of different jobs cannot be placed in the same cataloged procedure. The following types of statements can be included in a cataloged procedure:

- Job control statements for one or more job steps. All types of job control statements except a /& statement can be cataloged. The following statements cannot be included in a cataloged procedure since they are not accepted when a procedure is processed: ASSGN SYSRDR, RESET SYS, RESET ALL, RESET SYSRDR, and CLOSE SYSRDR. ASSGN SYSIPT, RESET SYSIPT, and CLOSE SYSIPT statements can be placed in a cataloged procedure only if SYSIPT data is not contained in the procedure. Nested cataloged procedures are not supported. That is, an EXEC statement in a cataloged procedure cannot invoke another cataloged procedure.

- Linkage editor control statements

- SYSIPT data for IBM-supplied language translators, utilities, and service programs. For example, a source program could be cataloged for input to an assemble job step. Cataloged SYSIPT data is read using a DTFCP or DTFDI logic module. Therefore, the SYSFIL option must be specified during system generation to cause inclusion of the system-to-programmer interface required by the DTFCP and DTFDI logic modules, if SYSIPT data is to be cataloged. A problem program can also read SYSIPT data from a cataloged procedure using the DTFDI logic module.

  The first job step in a job that accesses SYSIPT data determines the required location of SYSIPT data for all steps in the job. That is, if the first job step accesses SYSIPT data that is not cataloged, no steps in the job can access cataloged SYSIPT data.

- SET SDL command and SVA-defining control statements and other IPL commands

A cataloged procedure is invoked using the PROC=procedure name parameter on an EXEC statement/command contained in the input stream. If the cataloged procedure facility is to be used for a job that can execute in different partitions at different times and that has job control statements with partition-dependent data, a cataloged procedure must be defined for each partition in which the job step can execute. The partition-related cataloged procedure facility can then be utilized to enable one EXEC statement to be used to invoke the correct procedure at execution time regardless of the partition in which the job is to execute.

In order to use one EXEC statement, the following conventions must be used to define the partition-related procedure names that are assigned when the set of procedures for the job step is cataloged:

- The first character of the name must be a $
- The second character must identify the partition for which the procedure is defined (B, 1, 2, 3, or 4 to indicate the BG, F1, F2, F3, or F4 partition, respectively)
- The last six characters can be any valid procedure name characters. The same six characters must be used in each procedure name.

The procedure names assigned, therefore, will differ only by the partition identification character in the second position. The procedure name specified in the EXEC statement must consist of a $ character in the first two positions and the six common procedure name characters being used. When the job control program finds a $ character in the first position of a procedure name, it replaces the $ character in the second position with the partition identification character that indicates the partition in which the job control program is currently executing. In this manner, the procedure name that is required to select the correct partition-related procedure for this execution is automatically created.

If the partition-related procedures facility is not used, one EXEC statement with the appropriate procedure name specified is required for each partition in which the job step can execute. This approach can be utilized when the partition to be used for each execution of the job step is preplanned. However, it cannot be used when a job step is to be executed in any partition that happens to be available at the time the job step is initiated if partition-dependent job control data is required.

A modification facility is also provided that enables job control statements in a cataloged procedure (except for JOB statements) to be modified for the duration of the job step(s) involved by job control statements (called modifier statements) contained in the input stream. Job control statements can be added to the cataloged procedure and existing job control statements in the procedure can be deleted entirely or altered.

Job control statements that are to be referenced by modifier statements must be named using columns 73-79. The modification facility cannot be used to alter cataloged SYSIPT data. Such data must be modified using the appropriate librarian program.

If modifier statements are present in the input stream, the OV (overwrite) parameter must be included on the EXEC statement that invokes the procedure and an OVEND statement must follow the last modifier statement for the procedure. Modifier statements must be

placed in the input stream in the same sequence in which the job control statements they reference appear in the cataloged procedure.

The job control program is expanded to recognize a cataloged procedure request, locate the requested procedure in the procedure library, include the procedure in the input stream, and make the modifications indicated by any modifier statements in the input stream.

A cataloged procedure can also be invoked and modified by the operator. Appropriate EXEC and modifier statements can be supplied using SYSLOG. Figure 80.20.1 illustrates modification of a cataloged procedure.

Label statements are written on partition temporary label tracks in the label cylinder on SYSRES as usual whether or not the cataloged procedures facility is utilized.

Note that if an unending job, such as POWER/VS, is invoked via the procedure library, the procedure library cannot be updated while the unending job is executing. The recommended approach, therefore, is not to place the EXEC statement for POWER/VS in a cataloged procedure.

Job control statements in the cataloged procedure named EVA are shown below. Statements with an identification in columns 73-79 can be referenced by modifier statements in the input stream.

```
                                                        Columns
                                                        73-79
//  ASSGN SYS010,DISK,VOL=111111,SHR
//  ASSGN SYS011,TAPE
//  TLBL SYS011,'FILE-IN'
//  DLBL MASTER,'FILE-OUT'                              DLOUT
//  EXTENT SYS010,111111,1,0,200,200                    EXOUT
//  EXEC PROGRAM
/+      END CF PROCEDURE
```

Assuming the file name in the DLBL statement must be changed and an additional EXTENT statement must be supplied, the following job control is placed in the input stream (the A in column 80 indicates the modifier statement is an addition):

```
                                                        Columns
                                                        73-80
//  JOB USER
//  EXEC PROC=EVA,OV
//  DLBL MASTER,'FILE-USER'                             DLOUT  M
//  EXTENT SYS010,111111,1,1,1000,100                   EXOUT  A
//  OVEND
/&
```

The following modified job control is used:

```
//  ASSGN SYS010,DISK,VOL=111111,SHR
//  ASSGN SYS011,TAPE
//  TLBL SYS011,'FILE-IN'
//  DLBL MASTER,'FILE-USER'                             DLOUT
//  EXTENT SYS010,111111,1,0,200,200                    EXOUT
//  EXTENT SYS010,111111,1,1,1000,100                   EXOUT  A
//  EXEC PROGRAM
EOP EVA
```

Figure 80.20.1. Example of modification of a cataloged procedure

Generic I/O Device Assignment

A generic I/O device assignment facility is supported by the job control program in DOS/VS. The ASSGN job control statement and command are modified to permit general as well as specific I/O device assignments to be made. Job control is able to perform I/O device selection in response to general requests.

In DOS Version 4, the assignment of I/O devices to symbolic units is a user function and assignments are made strictly in terms of specific hexadecimal I/O device addresses. In DOS/VS, I/O device assignment can also be made in nonspecific terms, such as by generic device type (TAPE, DISK, 3410, 3330, for example), and at job step initiation time an available specific device of the generic type indicated will be selected dynamically by the job control program.

The use of generic names for device assignments makes jobs partition independent, which eliminates much of the I/O device assignment preplanning that is required when only specific hexadecimal device addresses are used. Use of the generic device assignment facility can also reduce the job control changes that are normally required when I/O devices are added to the configuration or existing device types are changed. In addition, jobs can be executed on systems with differing real device addresses without changing ASSGN statements.

Instead of a hexadecimal I/O device address (X'cuu'), one of the new unit parameters discussed below can be specified on an ASSGN statement or command to indicate the I/O device that is to be assigned to the specified symbolic unit (SYSxxx). In addition, two other new parameters are supported. The SHR parameter can be specified only on ASSGN statements that are for direct access devices. The VOL parameter can be included to specify a volume serial number when the request is for a tape or disk unit. The mode and form parameters on an ASSGN statement are unchanged. However, the TEMP or PERM parameter can now be specified on an ASSGN job control statement as well as on a command.

The new format of the ASSGN command/statement is

```
[//]ASSGN SYSxxx,  address          [,mode][,form][,VOL=volserno][,SHR]
                   generic type
                   address-list
                   SYSyyy
```

The address parameters supported are the same as in DOS Version 4 (X'cuu', UA, or IGN). New unit assignment parameters for the ASSGN job control command and statement shown above are as follows.

Generic Type. A device-class name or a device-type name can be specified. The device-class names supported are READER, PRINTER, PUNCH, TAPE, DISK, and DISKETTE. User-defined device-class names are not supported, nor is a mixed device type device class (TAPEDISK, for example). The device-type names supported are most of the card reader, card punch, printer, tape, and direct access device types supported by DOS/VS (3505R, 3505P, 3410T9, 3420T7, 3330, for example).

Specification of a generic type of device without the VOL or SHR parameters causes the job control program to attempt to select an available I/O device (one without partition ownership flags on) of the type indicated. The PUB table is inspected beginning with the first entry for channel 0. If a device-type name is specified on the ASSGN statement, the PUB table is searched for an unassigned device with that specific device-type code (all 3330-series device entries inspected, for example). The first unassigned device of the required type is selected for assignment.

If a device-class name is specified, the PUB table is searched for entries with the specific device-type codes that are included in that device class. The search is made in ascending order by device type within the class. For example, if DISK is specified, PUB table entries are inspected for a 2311, 2314/2319, 3330-series, 3340/3344, or 3350 device-type code in the sequence listed. The first unassigned direct access device encountered is selected for the assignment. (If DISK is specified in an installation with a mixture of direct access device types, the program must be capable of handling device-type independence among direct access devices since any type of direct access device can be selected.)

When an available device of the type required is found, it is assigned to the specified symbolic unit and a message is issued to the operator that includes the hexadecimal I/O address of the assigned device. If the specified device type is not present in the I/O configuration or if all devices of the required type are already assigned, the operator is notified. The action taken then depends on the option in effect for this situation.

At system generation, the ACANCEL parameter can be specified on the STDJC macro to indicate whether a job is to be canceled when job control cannot perform the requested device assignment. The standard option specified at system generation can be overridden for the duration of a job by including an OPTION statement in the job stream with the ACANCEL or NOACANCEL parameter specified. The standard option becomes effective again at end of job. If the job control option ACANCEL is in effect, a job is canceled when an I/O device assignment cannot be made. If the NOACANCEL option is in effect, the operator can enter whatever commands are required either to make the assignment or cancel the job.

The SHR parameter can be specified for a generic type request that is for a direct access device type only. Inclusion of SHR permits the selection of a direct access device that is already assigned to another symbolic unit. Both assigned and unassigned direct access device PUB table entries of the type indicated are inspected, as previously described, when SHR is specified.

The VOL parameter can be specified on a generic type request for a magnetic tape or direct access device whether or not the SHR parameter is also present. When a volume serial number is specified via the VOL parameter, the job control program searches the device entries in the PUB table as previously described. At the time the entry for an eligible device is inspected, the job control program also determines whether a volume is mounted on the device the entry describes. If so, the volume label is read and the volume serial number of the mounted volume is compared with the volume serial number given in the VOL parameter.

If a match is found, that unit is assigned. If a match is not found on any of the inspected units, the unit assigned is the last assignable device encountered during the search and the operator is instructed to mount the volume specified by the VOL parameter on the selected unit. Once the volume has been mounted, the operator must enter the NEWVOL attention command to indicate this fact. If no assignable device was found, the operator is notified, as in the case when VOL is not specified, and has the same options.

The VOL parameter allows the operator to mount a tape or disk volume that does not have a specific I/O device assigned on any available tape or disk unit prior to job step initiation. In effect, by premounting these volumes, the operator rather than the job control program is making the device selection. Premounted volumes must have standard volume labels. If the premounting facility is to be used, the volume serial numbers assigned to disk and tape volumes in the installation

must be unique so that volume serial number is a positive identification.

Address List. From one to seven specific hexadecimal I/O device addresses for devices of the same type can be specified. The PUB table entry for each device included in the list is inspected. Entries are inspected in the sequence in which they are listed in the ASSGN statement. The SHR parameter can be specified for a list of direct access devices. When SHR is not indicated, the first unassigned device encountered in the list is selected for the assignment. If SHR is specified, the first assigned or unassigned direct access device encountered in the list is selected. The operator is notified of a successful assignment. If none of the specified I/O devices can be assigned, the operator is notified and has the same options as when a generic type request cannot be satisfied.

If the VOL parameter is given, the job control program determines whether the volume specified is mounted on one of the units listed. A selection is made under the same conditions as described for a generic type assignment with the VOL parameter specified.

The address list facility could be used, for example, when a nonspecific tape unit request but a specific tape speed are desired. This selectivity is accomplished by including only those tape units with the required speed in the address list.

SYSyyy. SYSyyy can be any system or programmer logical unit. This type of request indicates the symbolic unit SYSxxx is to be given the same assignment as is currently in effect for the symbolic unit SYSyyy. When this parameter is specified for direct access devices, for example, it enables the user to indicate that a file is to be placed on the same direct access device as another file without knowing the address of that direct access device. The SHR parameter is implied and the VOL parameter does not apply.

The VOL parameter can also be included on an ASSGN statement or command that specifies a hexadecimal I/O device address. The SHR parameter is assumed when a specific direct access device is given since direct access devices can be shared in DOS Version 4.

The generic I/O assignment capability also complements the RELEASE macro. When an assigned I/O device is no longer required by a problem program, it can be released, which makes it immediately available for allocation to another symbolic unit. I/O units released in this manner can be selected by the job control program to satisfy a nonspecific request; the operator need not become involved, as is required in DOS Version 4.

The flexibility inherent in dynamic I/O device assignment by the operating system rather than preplanned assignment by the user enables DOS/VS to be more responsive to a changing daily workload and simplifies the planning required to increase the level of multiprogramming in an installation (number of partitions active concurrently).

OPERATOR COMMANDS

The DOS Version 4 IPL commands and the DPD and CAT commands are accepted by the DOS/VS IPL program. The DOS/VS job control program accepts the same job control commands as the DOS Version 4 job control program. In addition, the ALLOCR command and the additional operands for the SET command already described are accepted by the DOS/VS job control program. The SETPRT job control can be used to specify printer setup characteristics in a DOS/VS environment. The job control commands

common to DOS/VS and DOS Version 4 are functionally equivalent and have the same formats except where additonal parameters are required in DOS/VS commands to support foreground partitions F3 and F4.

The output of the LISTIO and MAP commands is extended in DOS/VS. The LISTIO command includes SYSVIS in the system units listed and the output of the MAP command includes the following new items:

- The size and highest virtual storage address of each virtual partition currently defined (in addition to the size and highest virtual storage address of each real partition defined)
- The size of real partitions with a real mode program in execution. The value listed is the size given in the SIZE parameter for the job step (if this parameter is specified) or the size of the defined permanent real partition.
- Partition priority
- The size and highest real storage address of the main page pool. The size given is the amount of real storage above the address of the last real partition currently defined in the real address area. It does not include the real storage currently available to the page pool as a result of inoperative real partitions or specification of the SIZE parameter for a real partition.
- The size and highest virtual storage address of the SVA
- The size and highest virtual storage address of the system GETVIS area in the SVA

In addition to all the attention commands that are accepted in DOS Version 4, DOS/VS accepts the ALLOCR, LFCB, LUCB, PRTY, and NEWVOL attention commands previously discussed. DOS/VS also accepts the SETDF attention command, which enables the operator to specify default printer setup characteristics for 3800 printers. The TPBAL attention command, not available in DOS Version 4, is discussed under "Page Management" in Section 80:25. MAP and LISTIO attention commands provide the same extended output as MAP and LISTIO job control commands.

Note that in a DOS/VS multiprogramming system, the CANCEL attention command must specify the partition containing the job to be canceled (there is no default to the BG partition). This reduces the chance of an operator inadvertantly canceling a background partition job when trying to cancel a foreground partition.

MODIFICATIONS

The minimum DOS/VS supervisor size is increased from the 14K required to support a single-partition DOS Version 4 environment. The increase results primarily from the addition of virtual storage support and more standard features. While a DOS Version 4 supervisor is limited to a maximum size of 32K, support of multiprogramming environments requires DOS/VS supervisors larger than 32K.

The minimum supervisor size for a single-partition DOS/VS environment (no options included and 2311 or 2314/2319 disk storage only) varies depending on the System/370 processor supported. Shown below are the minimum supervisor sizes for Release 34.

Model 115	30K
Model 125	30K
Model 135	30K
Model 138	34K
Model 145	30K
Model 148	34K
Model 155 II	30K
Model 158	30K

A change to the supervisor patch area makes supervisor patches easier to make in DOS/VS than in DOS Version 4. A low core patch area of 64 bytes (labelled IJBPATCH) is provided that enables patches to be made using absolute addresses (no base register required). The high core patch area is 300 bytes.

DOS/VS supervisor code is modified as required to support EC instead of BC mode of system operation (different PSW format and interruption codes in permanently assigned locations above address 127, for example). The DOS/VS supervisor recognizes the same interruptions as a DOS Version 4 supervisor as well as program event recording (if SDAIDS are used) and translation interruptions: translation specification, segment translation, and page translation exceptions.

A translation specification error causes the system to be placed in a wait state. Since all invalid bits in the segment table are always off, a segment translation exception can occur only if a storage location outside of the segment table is addressed, which indicates the virtual storage address to be translated is outside the virtual storage size supported. When a segment translation exception occurs, therefore, the interruption code is changed to that for an addressing exception and the interruption is handled just as if an actual addressing exception program check had occurred.

A page translation exception occurs in a DOS/VS environment only if the invalid bit is on in the addressed page table entry, since all page tables contain the maximum number of entries possible for a 64K segment size and a 2K page size. All page translation exceptions are handled, therefore, as page faults.

Code is included in the DOS/VS supervisor to ensure proper system operation when a disabled page fault occurs. In DOS/VS, a disabled page fault is a page fault that occurs during the execution of a routine that has disabled the processor for external and I/O interruptions. A DOS/VS supervisor routine operates with external and/or I/O interruptions disabled because (1) it is not reentrant and, therefore, should not be

reentered before it completes execution, or (2) it is reentrant but processes a serially reusable resource.

The processing of a page fault, which requires the processor to be enabled for I/O interruptions so that the I/O interruption for a completed page-in can be presented, could allow a routine that operates with the processor disabled to be reentered, with improper processing the result. Provisions must be made to handle this situation.

The method used to handle a page fault in a DOS/VS supervisor routine that operates with the processor disabled for interruptions varies depending on the type of routine. For example, a gating technique (NOP/BRANCH instruction switch) is used for certain SVC routines that are not reentrant. When a disabled page fault occurs during the execution of these SVC routines, the instruction switch is set to a BRANCH. The task that issued the SVC routine is marked waiting for I/O, the page request is enqueued, and the dispatcher receives control. The highest-priority ready task is given processor control.

During the time the task that issued the SVC routine is waiting for the disabled page fault to be processed, the SVC routine may be entered by another task. When the gating instruction is executed, a branch is taken to a routine that marks the requesting task resource-bound and establishes the SVC instruction as the first instruction the task will execute when it again receives CPU control. The dispatcher is entered to give control to the highest-priority ready task.

As soon as the disabled page fault for the SVC routine is serviced, the gating switch is set to a NOP instruction, tasks that were marked resource-bound waiting for this SVC routine are taken out of I/O bound status, and the SVC routine is dispatched. This technique enables processing to continue during the time required to handle a disabled page fault and prevents a nonreentrant SVC routine from being reentered until the page fault has been serviced and the SVC routine completes its execution.

Disabled page faults are not permitted in user tasks, B-transient routines, and I/O appendage routines. If a disabled page fault occurs in one of these types of routines, the associated user task is abnormally terminated. Note also that an enabled page fault that occurs during the operation of a MICR stacker selection routine causes the associated task to be canceled.

The DOS/VS supervisor disables the processor for interruptions caused by the execution of SET SYSTEM MASK (SSM) instructions. The SSM instruction is not used in DOS/VS to enable or disable the processor for I/O and/or external interruptions. The STORE THEN AND SYSTEM MASK and STORE THEN OR SYSTEM MASK instructions are used instead.

The DOS/VS supervisor is also modified to minimize the impact of the new EC mode PSW format on existing user-written asynchronous routines that are entered via a STXIT macro and that inspect the contents of the PSW. Before giving processor control to a STXIT routine, the DOS/VS supervisor moves the interruption code, instruction length code, condition code, program mask, and instruction address fields from the appropriate EC mode old PSW to the PSW field within the required 72-byte save area for the asynchronous routine. These fields are placed in their BC mode instead of their EC mode locations in the user PSW save area. When the STXIT routine returns control to the supervisor, these five fields are moved from the BC mode PSW save area to the appropriate EC mode PSW.

This implementation enables existing user-written STXIT routines that operate in BC mode to execute in a DOS/VS (EC mode) environment and

access or modify these five PSW fields without having to be modified, since the PSW save area remains in BC mode.

The STXIT macro in DOS/VS can specify the TT parameter in addition to the AB, IT, PC, and OC operands that can be specified in DOS Version 4. The TT parameter specifies a user-written task timer exit routine (see discussion of the task timer facility later in this subsection). The DOS/VS EXIT macro also has TT and AB parameters, which are not supported in DOS/VS.

The TT parameter is used to exit from a task timer exit routine. The AB parameter can be specified on an EXIT macro only to exit from an abnormal termination routine for a main task. The EXIT AB macro causes the abnormal termination condition and ABEND indication for the main task to be reset and control is returned to the instruction after the EXIT AB macro. Thus, the abnormal termination routine must clear the abnormal condition.

The EXIT AB macro cannot be issued by the abnormal termination exit routine for a subtask. The exit routine must end with a CANCEL, DETACH, DUMP, JDUMP, or EOJ macro.

The system mask, mode bits (in BC mode PSW bit positions 12 to 15), and protection key are not placed in the user PSW save area since these fields are meaningful primarily to the supervisor and should not be modified by a problem program. It is assumed that existing user-written STXIT routines that are operating in a DOS Version 3 or 4 environment do not access these PSW fields. It is also assumed that a subtask in existing DOS Version 3 and 4 systems does not inspect or modify the PSW in the save area of the main task or any other subtask in the partition in which the subtask is executing. Therefore, the PSW is placed in main task and subtask save areas in EC mode format.

The contents of each partition communication region in the supervisor are modified as follows:

- Bytes 32-35 contain the virtual storage address of the upper limit of the address space available to the currently executing program in the partition. This is the limit specified at system generation or via the ALLOC (virtual partition) or ALLOCR (real partition) command when SIZE was not specified in the EXEC command. When the SIZE parameter was specified, the limit is determined by the SIZE parameter value.

- Bytes 48-51 contain the virtual storage address of the end of the virtual storage defined at system generation.

Note also that bytes 8-9 and 10-11 in a communication region contain 16-bit rather than 15-bit addresses when the supervisor is larger than 32K.

A partition-independent system communication region, which is not present in DOS Version 4, is defined in the DOS/VS supervisor. This area contains data that is duplicated in the communication region extension area for each batched partition in a DOS Version 4 environment. The communication region extension area address in each partition communication region in a DOS/VS environment points to the location of the option table in the partition-independent system communication region. For the sake of compatibility, the layout of the option table in the system communication region is the same in DOS/VS and DOS Version 4.

Several system tasks are defined within the DOS/VS supervisor that have higher dispatching priority than any partition task. System tasks in high-to-low dispatching priority sequence are recovery management

support (RMS), the page manager, the PAGEIN task, program fetch, display operator console support, and the error recovery routines (ERPs). Any ready system task is given processor control before a ready partition task.

In DOS/VS, the value in the time of day clock is Greenwich Mean Time (GMT) with a base value of January 1, 1900 instead of local time. In order to have the supervisor display and accept local time values instead of GMT values, the ZONE parameter must be specified at system generation or during system initialization to indicate the difference between GMT and local time. A ZONE parameter is added to the SET command so that the time differential can be specified during system initialization. The zone values at system initialization override the zone values specified at system generation.

A GMT and a LOCAL parameter are supported for the GETIME macro to enable a programmer to obtain GMT or local time, respectively. As in DOS Version 4, in DOS/VS the job accounting facility uses the interval timer rather than the time of day clock for timing information.

The DOS/VS supervisor also contains a fetch table in support of the new core image library organization. This table is used by job control, the linkage editor, librarian routines, and program fetch. The fetch table contains one entry for the system core image library and one entry for each partition defined at system generation if support of private core image libraries is included in the supervisor. The address of the fetch table is contained in the system communication region.

The fetch table entry for the system core image library contains the address of the second level directory for this library, a condense counter, the disk address of the directory for this library, the disk address of the link area for this library, the number of tracks per cylinder in this library, and the number of library blocks per track in the library. The fetch table entry for a partition provides the same information about the private core image library that is assigned to the partition, if any. The $MAINDIR service program is called as required by other system routines to place entries in the fetch table.

NEW FEATURES

The DOS/VS supervisor supports other functions that are not provided in DOS Version 4 in addition to those already discussed. The following new features are also available in DOS/VS.

Relocating Loader. The relocating loader is a standard feature of DOS/VS but it can be deleted from the generated system during system generation by specifying RELLDR=NO in the FOPT macro unless the reloading loader is required by another selected feature (RPS support, VSAM, VTAM, GETVIS/FREEVIS macros, OLTEP, or RETAIN).

The relocating loader provides program relocation at execution time that enables a single phase or multiphase program that is in relocatable format to be loaded into any virtual storage location for execution. A relocatable phase can also be relocated when an executing program issues a LOAD or FETCH macro to bring the phase into virtual storage.

Use of the relocating loader eliminates the need to relink-edit (or reassemble and relink-edit) relocatable program phases when partition starting addresses change because of an increase in supervisor size or because partition sizes are increased or decreased. In addition, this facility enables a relocatable program to execute in any partition without the necessity of having multiple copies of the program cataloged in a core image library. The need for writing self-relocating programs is also eliminated.

When the relocating loader is present in a DOS/VS supervisor, program phases are flagged in their core image library directory entry as relocatable or nonrelocatable. Absolute and self-relocating program phases are flagged as being nonrelocatable. Relocatable phases can be produced by the DOS/VS linkage editor program, which is modified to produce relocatable as well as nonrelocatable phases.

A relocatable phase contains relocation information (a relocation list dictionary--RLD) that is not present in nonrelocatable phases. This information identifies the location of address constants in the phase that must be modified by the appropriate relocation factor when the phase is relocated. The relocation factor is calculated by the relocating loader when a phase is to be fetched or loaded and is the difference between the beginning virtual storage address specified at link-edit time and the beginning virtual storage address of the partition (or SVA area location) in which the phase is now to be loaded.

The ACTION linkage editor control statement can specify the REL parameter to indicate that a relocatable phase is to be produced. Relocatable phases are produced by the linkage editor as the default when the relocating loader is present in the supervisor and the ACTION statement does not contain a relocation parameter. Nonrelocatable phases are produced (1) when the NOREL parameter is specified on the ACTION statement, (2) if the relocating loader is not present in the system and the ACTION statement does not contain a relocation parameter, or (3) when the phase is identified as self-relocating.

Self-relocating phases are considered to be nonrelocatable by the relocating loader so that address constant relocation will not be performed by the program fetch routine since this function is handled in the self-relocating phase. (Note that a self-relocating program can be initiated in a nonmultiprogramming DOS/VS environment using the EXEC statement, which is not permitted in DOS Version 4.)

The DOS/VS linkage editor produces a relocatable phase when possible if the REL parameter is specified for the phase, whether or not the supervisor with which the linkage editor is operating has the relocating loader included. However, a supervisor without the relocating loader will not relocate a relocatable phase. The relocatable phase is loaded at the virtual storage address indicated at link-edit time, just as if it were a nonrelocatable phase.

The object modules of existing programs that are to be made relocatable must be processed once by the DOS/VS linkage editor. After relocatable program phases have been created and cataloged in a core image library, relink-editing of these programs for the sole purpose of relocating them to a different starting virtual storage address is not required.

Page-formatted core image libraries are not supported in DOS/VS and an entire program must be fetched before it can begin executing. The program fetch routine performs the channel program translation and temporary page fixing required for I/O operations that load virtual mode programs. Program fetch obtains and fixes one or more page frames for each read operation that is initiated to bring in text records and performs the required channel program translation. (Program fetch obtains and fixes as many page frames as it can without causing a page-out.)

A PBDY parameter can be specified at link-edit time to cause the load point of a phase to be aligned on a 2K page boundary. This facility can be used to avoid the situation in which the instructions and/or data in a text record cross a 2K boundary.

Program fetch processing of relocatable phases in DOS/VS is designed to minimize the possibility of incurring page faults during relocation processing. The RLD records for each text record are read after the text record is read and address constant relocation processing is performed on each text record immediately after the text is read in. Text record reading is overlapped with relocation processing.

A real mode program phase is loaded by initiating one I/O operation for each cylinder in which the phase is contained. Each I/O operation starts a chained CCW list that consists of one read CCW for each 1024-byte library block record in the cylinder for the phase. Text records for a real mode phase are read directly into the page frames that have been assigned to the real partition that is to be used.

When a virtual mode program is fetched from a core image library in DOS/VS, the program fetch routine does not force all or any part of the program to be written in the page data set as part of the program loading procedure. Page frames containing pages of a program that is being loaded are subject to reassignment as per the page replacement algorithm, as are the page frames containing nonfixed pages of executing virtual mode programs. During the loading of a virtual mode program, pages that have already been loaded may be paged out while the balance of the program is being fetched, if the real storage they occupy is required for allocation to pages of other virtual partitions.

The read operation that loads program text into a page frame causes the change bit for the page frame to be turned on. If the page frame allocated to a reentrant page is taken for reassignment, the fact that the change bit is on as a result of program loading causes the first and only page-out of the reentrant page.

Multitasking. Inclusion of the asynchronous processing option in a DOS/VS supervisor provides support of a maximum of 15 concurrently operating tasks, three more than the maximum supported in DOS Version 4. One main task can operate in each partition while the number of subtasks supported can be allocated among active partitions as desired. The total number of subtasks permitted depends on the number of partitions defined. The number of partitions plus the number of subtasks cannot exceed 15, as shown below.

Number of Partitions	Maximum Number of Subtasks
2	13
3	12
4	11
5	10

The CHAP macro, not provided in DOS Version 4, can be issued by a subtask in a multitasking partition. CHAP causes the subtask that issues it to be assigned the lowest dispatching priority of all subtasks in the partition. The CHAP macro is ignored if issued by a main task. If multitasking support is not present in a DOS/VS system, any task issuing the CHAP macro is canceled.

Synchronous Exit Facility. This option is provided for use in multitasking partitions. It provides a SYNCH macro that is used to give processor control to a synchronous exit routine. The exit routine executes until any SVC is issued. The service requested by the SVC is not performed but control is returned to the routine that issued the SYNCH macro (instruction after the SYNCH macro). A program must be operating in supervisor state in order to issue the SYNCH macro. An EC-mode PSW and general register values specified by the SYNCH macro are supplied to the exit routine in a save area.

Page Fault Handling Overlap. This optional facility is provided for virtual mode programs that perform private subtasking, that is, that handle their own subtasking instead of using the DOS/VS multitasking facility. When private subtasking is used in a partition, the main task does not use the ATTACH macro to create subtasks, and the main task and all its subtasks operate under one program information block (PIB).

Use of the page fault handling overlap facility enables page fault handling for private subtasks within a virtual partition to be overlapped with private subtask execution in the partition. When an executing private task in the partition encounters a page fault, the affected task can be placed in the wait state and processor control can be given to another ready private task in the same partition (by user programming). Without use of the page fault overlap facility, the entire virtual partition is placed in the wait state after a page fault occurs for any task in the partition.

In order to use the page fault handling overlap facility, the main task in a virtual partition in which private subtasking is being performed must issue the new SETPFA macro to indicate the address of a user-written page fault appendage routine that is to be given control whenever a page fault occurs for a private task in the partition. The page fault appendage routine is located in the partition. This routine and all the virtual storage pages it references must be fixed by the user (via the PFIX macro) before the SETPFA macro is issued.

A user-written page fault appendage routine executes in supervisor state with the processor disabled for I/O interruptions and protect key 0 assigned. Page faults caused by a page fault appendage routine are not valid and cause task termination.

When the page fault appendage routine receives control after a page fault occurs for any private task in the partition, it can place the affected private task in the wait state and dispatch another ready private task. The appendage routine then returns control to the supervisor indicating whether the page fault is to be processed. After a page fault has been serviced, control is returned to the page fault appendage routine, which can post the affected task ready.

Multiple Timer Support. The interval timing option supported in DOS Version 4, which can be used by only one partition at a time, is replaced by the multiple timer facility in DOS/VS. When the interval timing option is included in a DOS/VS supervisor, the interval timing facility is available for concurrent use by all defined partitions and each of their tasks, if multitasking support is included in the supervisor. Each task, however, can have only one interval pending at a time. The interval timer at location 80 is used for interval timing. In DOS/VS, the real time interval set via a SETIME macro can be specified in 1/300ths of a second units in addition to units of one second, as in DOS Version 4.

The interval timing facility available to each partition or task in DOS/VS provides the same capabilities as those provided in DOS Version 4. However, one additional timer interface macro (TTIMER) is supported in DOS/VS. A task can use the TTIMER macro to cancel a pending interval of time it has established or to ascertain the amount of time remaining in an established interval. The functions provided by the other timer interface macros are the same in DOS/VS and DOS Version 4, as are the formats of these macros.

Elimination of the DOS Version 4 restriction that allows the interval timing facility to be used by only one partition at a time enables concurrent execution in a DOS/VS environment of programs that require the interval timing facility.

Task Timer. The task timer option requires the presence of the clock comparator and CPU timer, which are standard on all System/370 processors supported by DOS/VS except Models 135 (Model 0) and 145 (Models 0 and 2). The TIME parameter of the FOPT macro specifies the one partition that can use the task timer facility and only the main task in the owning partition can use task timing.

The SETT macro is used to set a task time interval (up to 21,474,836 milliseconds) that will be decremented only when the task that can use task timing is executing. If SETT is issued by a program in a partition that does not own the task timer, the program is canceled. When the task interval elapses, the user-written exit routine specified via the STXIT TT macro is entered. If a user-written exit routine was not specified via a STXIT TT, the interruption that occurs when the task interval elapses is ignored. The EXIT TT macro is used to return control from the user-written task timer routine.

The TEST TT macro can be issued by the task that owns the task timer to cancel the remaining portion of an interval previously set (without entering the user-written exit routine) or to obtain the amount of time remaining in the interval (in hundreds of milliseconds). If a program other than that executing in the owning partition issues a TEST TT, the program is canceled.

When a program is restarted from a checkpoint, the interval that was outstanding before the restart, if any, is not reestablished.

RUNMODE Macro. The RUNMODE macro can be included in Assembler Language programs to determine the mode in which they are currently operating. For example, RUNMODE can be used in a program that can operate either in virtual or real mode when this information is needed for proper processing.

Virtual Storage Management Facility. A virtual storage pool (GETVIS area) in highest addressed virtual storage in a virtual partition is set aside when the SIZE parameter is specified for a virtual mode program, as shown in Figure 80.25.1. The starting address of this pool is contained in bytes 32-35 of the partition communication region. The user area in lowest addressed virtual storage in the virtual partition must be a minimum of 2K bytes. The minimum size of the virtual storage pool is 4K and the maximum size is 12,168K bytes. The virtual storage pool is assigned the same storage protect key as the user area. The first 2K bytes of the pool contain control information and are not available for allocation.

The virtual mode program that is executing in the user area of the virtual partition can obtain virtual storage from this pool, in multiples of 128 bytes, using the GETVIS macro. A request can indicate that the virtual storage allocated is to be aligned on a page boundary. The POOL parameter can be used to cause allocation requests to be packed within virtual storage pages, since the search for the specified space begins at the user-specified virtual storage address. The FREEVIS macro is provided to free virtual storage obtained using the GETVIS macro. To ensure proper allocation and deallocation of virtual storage in the pool, only GETVIS and FREEVIS macros should be used.

Support of the GETVIS/FREEVIS macros must be requested at system generation. These macros are required if VSAM, 3800 Printing Subsystem support, rotational position sensing support, VTAM, or the AP-1 Program for 3344 and 3350 disk storage is to be utilized.

The GETVIS and FREEVIS macros are also used to obtain and free virtual storage in the system GETVIS area in the SVA, if such an area has been defined. Virtual storage is obtained from this area in 512-byte multiples.

The CDLOAD macro is provided to enable a program to load a program phase into the partition GETVIS area. When a CDLOAD macro is issued, the supervisor first determines whether the specified phase is already present in the SVA or partition GETVIS area. If so, the phase is not loaded. If the phase is not currently resident, the required amount of space is obtained from the partition GETVIS area and the specified phase is loaded. Control is returned to the instruction after the CDLOAD macro. The PAGE parameter can be specified to cause the phase to be loaded on a page boundary.

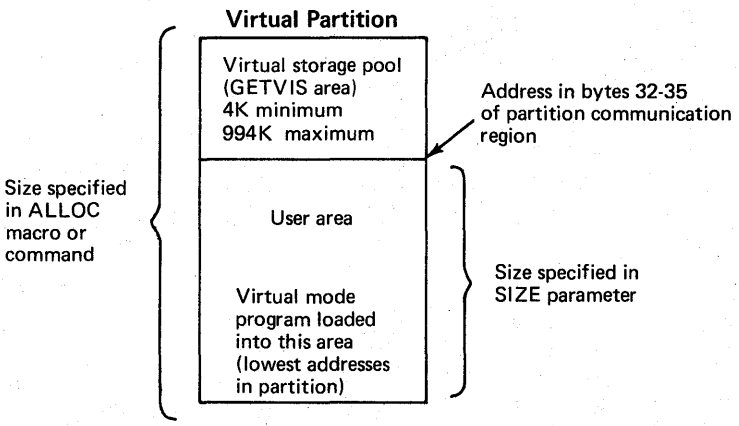


Figure 80.25.1. Organization of a virtual partition when the SIZE parameter is specified

Cross Partition Event Control. Cross partition event control is an optional supervisor feature. It provides a cross partition communication facility that can be used to synchronize the execution of programs in two or more partitions. It enables a task in one partition to wait for the completion of a user-defined event associated with another partition and to be notified when the other partition signals completion of the event.

When cross partition event control is requested at system generation, an XECB table is generated in the supervisor. It contains room for a minimum of four XECB entries or the user-specified number of XECB entries (up to a maximum of eight for each partition defined). The XECBTAB macro is provided to enable a task to (1) name and define an XECB table entry, (2) delete an existing XECB table entry that it defined previously, (3) determine whether a specific XECB table entry has already been defined, (4) reset an entry, or (5) delete all entries in the XECB table and cause tasks to be posted ready that are waiting for an XECB to be posted by the task deleting the entries.

When the XECBTAB macro is used to name and define a new entry, it also specifies whether the task that is issuing the XECBTAB macro is to post the XECB entry being defined (ACCESS=XPOST) or wait for posting by another task (ACCESS=XWAIT). The XPOST macro is provided to enable a task to post a specific XECB entry complete. If a task is currently waiting on the posted XECB entry, it is made ready. The waiting task can be in the same partition as the task that issued the XPOST macro or another partition. The XWAIT macro is provided to enable a task to wait for a specific XECB entry to be posted complete by another task executing in the same or a different partition. The task that defines an ECB with ACCESS=WAIT can also use the WAIT and WAITM macros to wait on the ECB.

The XECBTAB macro with a reset specification enables a task that defined an ECB to clear the information that associates the ECB with the task. This enables another task to issue an XWAIT or XPOST macro to the reset ECB.

PAGE MANAGEMENT

General Functions and Operation

Page management is the portion of a DOS/VS supervisor that implements demand paging and provides the programming required by dynamic address translation hardware for support of a virtual storage environment. Page management consists of a set of routines that manage real storage and external page storage.

Page management performs the following functions:

- Allocation of page frames when page faults occur and in response to specific allocation requests
- Permanent and temporary fixing and unfixing of pages
- Modification of the page tables and other required tables to reflect the allocation and deallocation of real storage
- Scheduling of page-in and page-out requests as required
- Partition deactivation when paging activity is determined to be too high, and partition reactivation when possible

Page management is entered when a page fault occurs, PFIX/PFREE macro is issued, temporary fix or free (TFIX/TFREE) request is made, GETREAL/FREEREAL request is issued, or a RELPAG, FCEPGOUT, or PAGEIN macro is issued. In certain cases, the request can be serviced immediately by a page handling routine, such as when a temporary fix request is received for a page that is present in real storage and already temporarily fixed. If the request cannot be handled immediately, it is placed at the end of the page queue, the requesting task is placed in the wait state, and the page manager system task is entered to service the request, if possible. Otherwise, control returns to the dispatcher so that CPU control can be given to a ready task.

The page manager system task always must be activated to process a page fault. However, it may or may not be activated to process the following types of requests: GETREAL, TFIX, PFIX, or PAGEIN. It is never activated to process the following requests: FREEREAL, TFREE, PFREE, RELPAG, or FCEPGOUT.

When the page manager system task receives control, the first request in the page queue is selected and the required functions are performed. This may involve execution of the page selection routine to choose a page frame for allocation to a page and execution of the page I/O routine to perform the necessary paging operations. When the request has been processed, it is removed from the page queue and the affected task is marked ready. The page manager system task then processes the next queued request, if any.

Page management routines maintain a page frame table (PFT) which contains one eight-byte entry for each page frame in the real storage defined by the RSIZE parameter at system generation. This table, which is located near the end of the supervisor area with the segment table and the page tables, indicates the status of real storage at all times. Its entries are used by the page selection routine to allocate a page

frame when required. The system communication region contains a pointer to the beginning of the segment table and a pointer to the beginning of the page frame table.

A page frame table entry (PFTE), shown in Figure 80.25.2, contains the following:

- A counter (of 11 bits) indicating the number of temporary fixes currently in effect for the associated page frame.
- A nonfixable (NFF) bit which indicates that the page to which this page frame is allocated cannot be fixed. This bit is turned on when a TFIX request is received for a page for which there is a PFIX request still pending, and when a PFIX request is received for a page that is temporarily fixed in a page frame that is not eligible for permanent fixing.
- A bit which indicates whether the associated page frame is failing. This malfunctioning bit is turned on by recovery management routines when an uncorrectable real storage error occurs for a page frame.
- A selection pool (SP) bit which indicates whether the associated page frame is part of the selection pool. Both available and assigned page frames are contained in the selection pool. All the page frames in real storage that are not (1) allocated to an active real partition, (2) allocated to the PDAID alternate buffer area or SDAIDS buffer area, (3) permanently or temporarily fixed, or (4) contained in the supervisor area are part of the selection pool. Page frames in the selection pool are inspected by the page selection routine when a page frame must be chosen for allocation.
- A second nonfixable (NF) bit which indicates whether a page frame is temporarily fixable. This bit is used primarily to prevent temporary fixing of specific page frames that must be assigned to a real partition or that must be permanently fixed. (See discussions under "GETREAL/FREEREAL Requests" and "PFIX and PFREE Macros", respectively.)
- A page number field which contains all ones if the associated page frame is available (not allocated to a page) or contains the number of the virtual storage page to which the associated page frame is allocated (address of the virtual storage page divided by 2048).
- A forward pointer and a backward pointer which are used to connect PFTE's in the selection pool together to form queues that are used by the page selection routine. (See discussion under "Page Replacement Algorithm" below.)

When PFIX/PFREE macro support is included in the supervisor, a page frame table appendage (PFTA) with the same number of entries as the page frame table is also maintained. For each entry in the PFT, there is a corresponding one-byte entry in the PFTA in the same relative table position. This entry is used as the permanent fix counter for the associated page frame. Each counter indicates the number of PFIX macros currently in effect for the page in the associated page frame. A zero value in the counter indicates the page is not permanently fixed. The address of the beginning of the page frame table appendage is contained in the system communication region.

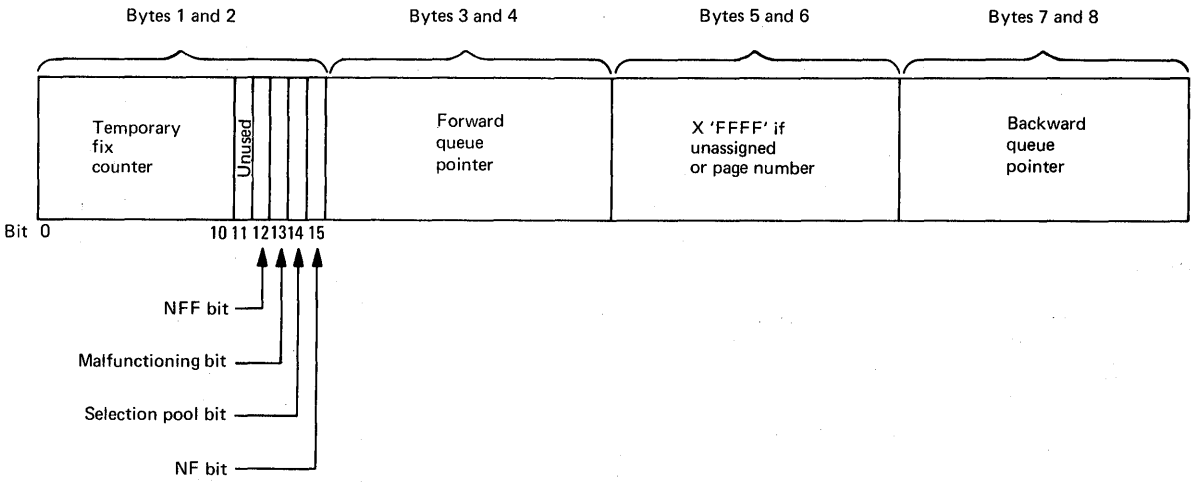


Figure 80.25.2. Format of a page frame table entry

The page frame table and, if present, the page frame table appendage are initialized during system generation. If the real storage present in the system is smaller in size than the RSIZE value specified at system generation, the PFTE's for unusable page frames are marked unusable during system initialization. All entries in the PFTA are initialized to zero. PFTE's for usable page frames are initialized as follows:

- PFTE's for page frames that are allocated to the supervisor area are marked not in the selection pool. The malfunctioning bit, nonfixable bits, and temporary fix counter are set to zero. The page number field contains the number of the associated page frame (real address of the page frame divided by 2048).

- PFTE's for all page frames located above the supervisor area (those in the page pool) are marked as being in the selection pool. These PFTE's are marked available and chained together. The temporary fix counter, malfunctioning bit, and nonfixable bits are set to zero for PFTE's in the selection pool.

The page tables, page frame table, and page frame table appendage are updated by page management routines, as required, as page requests are serviced.

Page Replacement Algorithm

The page selection routine is entered to select a page frame for allocation to a virtual storage page. Thus, the occurrence of a page fault or the issuing of a specific request usually causes entry into the page selection routine.

The PTFE's for the page frames that are eligible for allocation belong to the selection pool. The PFTE's in the selection pool are connected via their forward and backward pointers to form five queues. These queues are structured to indicate the relative activity of the page frames in the selection pool and which page frames have been assigned most recently. The page replacement algorithm used by the page selection routine is designed to keep the most frequently referenced

(active) pages and most recently assigned pages present in real storage. Page frames that have been most recently referenced are considered to be the most active. Page frames that have not been referenced for the longest period of time are considered to be the least active.

An unreferenced page frame that has not been changed is selected for allocation before an unreferenced page frame that has been changed, since such a page frame can be made available without performing a page-out operation. Similarly, when there are no unreferenced page frames, referenced unchanged page frames are selected for allocation before those that were referenced and changed. Page reclamation is not attempted in DOS/VS. This would involve inspecting the selection pool to determine whether the page that requires a page frame is still present in real storage (waiting to be paged out, for example), in which case the page frame containing the page would be reassigned and a page-in would be avoided.

The selection pool contains one hold queue and one queue for each possible combination of reference and change bit settings. Possible reference and change (R,C) bit settings for a page frame are 0,0; 0,1; 1,0; and 1,1. The five PFTE queues in the selection pool and their contents are the following:

- Q00 - identifies page frames that were unreferenced and unchanged since the last time they were inspected by the page selection routine. Only PFTE's for page frames with a 0,0; 1,0; or 1,1 reference and change bit setting can be in this queue. At system initialization, the PFTE's for all page frames above the supervisor area are placed in Q00 and the other four queues are empty.

- Q01 - identifies page frames that were unreferenced since the last time they were inspected by the page selection routine but that were changed at some previous time. Only PFTE's for page frames with a 0,1 or 1,1 reference and change setting can be in this queue.

- Q10 - identifies page frames that were referenced since the last time they were inspected by the page selection routine but that were not changed. PFTE's associated with page frames with a 0,0; 1,0; or 1,1 reference and change setting can be in this queue.

- Q11 - identifies page frames that were referenced and changed since the last time they were inspected by the page selection routine. Only PFTE's for page frames with a 0,1 or 1,1 reference and change setting can be in this queue.

- HQ (hold queue) - contains the page frames that were most recently allocated. This queue is implemented to try to ensure that just allocated page frames are not reassigned immediately before they can be used.

Each of the five queues is maintained in first-in, first-out sequence. This is done to preserve a record of the comparative length of time in the queue among PFTE's in the same queue.

The following steps are taken by the page selection routine to choose a page frame for allocation:

- Q00 is inspected from top to bottom. The first PFTE for a page frame with a 0,0 reference and change setting is selected for allocation whether or not it is marked available. Each PFTE for a page frame with a setting other than 0,0 that is inspected while searching for a page frame with a 0,0 setting is moved from Q00 to the end of the queue for that setting and the reference bit for the page frame is turned off. Q00 is searched until a PFTE for a page

frame with a 0,0 setting is found or until Q00 is depleted of PFTE's.

- Q01 is inspected from top to bottom if Q00 is depleted without finding a PFTE with a 0,0 setting. The first PFTE for a page frame with a 0,1 setting is selected for allocation. Inspected PFTE's for page frames with a 1,1 setting are moved from Q01 to the end of Q11 and the reference bit for the page frame is turned off. Q01 is searched until a PFTE for a page frame with a 0,1 setting is found or until it is depleted of PFTE's.

- If Q01 becomes depleted without the selection of a page frame, queue exchanging is performed if the selection pool is not empty. All the PFTE's in Q10 are placed in Q00, all the PFTE's in Q11 are placed in Q01, and all the PFTE's in HQ are placed in Q10. Reference bit settings are not changed. This procedure replenishes Q00 and Q01, if possible, and leaves Q11 and HQ depleted. Q00 is then searched for a PFTE for a page frame with a 0,0 setting and the process continues as described.

- If the entire selection pool is empty, an exit is taken to a routine that attempts to make page frames available for the selection pool. This routine attempts to make temporarily fixed page frames available by resetting certain system routines (program fetch, the CCW translation routine, and the SVC 44 routine that writes error records in the SYSREC file) that have partially completed temporary fix (TFIX) requests outstanding. If one or more page frames are made available by resetting one or more of the routines listed, the page manager is reentered to perform page frame selection. If page frames cannot be made available, page frame selection cannot be completed at this time and control is given to the highest-priority ready task.

Page frame selection is illustrated in Figure 80.25.3. Once a page frame is chosen, its PFTE is inspected to determine whether the page frame is available or currently assigned. If the page frame is assigned, a page-out is required if the change bit is on and the appropriate page table entry must be invalidated. The page I/O routine is entered to perform any required paging operations. The user bit for the page frame selected indicates whether a page-in to the selected page frame is required.

Once the required paging operations have been performed and/or the selected page frame has been cleared to zero, the appropriate page table entry and PFTE are updated, and reference and change bits for the page frame are set to zero. If the allocation was made to service a page fault, the PFTE for the page frame is placed at the end of the hold queue. If the allocation was made as a result of a PFIX or TFIX request, the appropriate fix counter is increased by one and the PFTE is removed from the selection pool if this was the first fix request issued for the page.

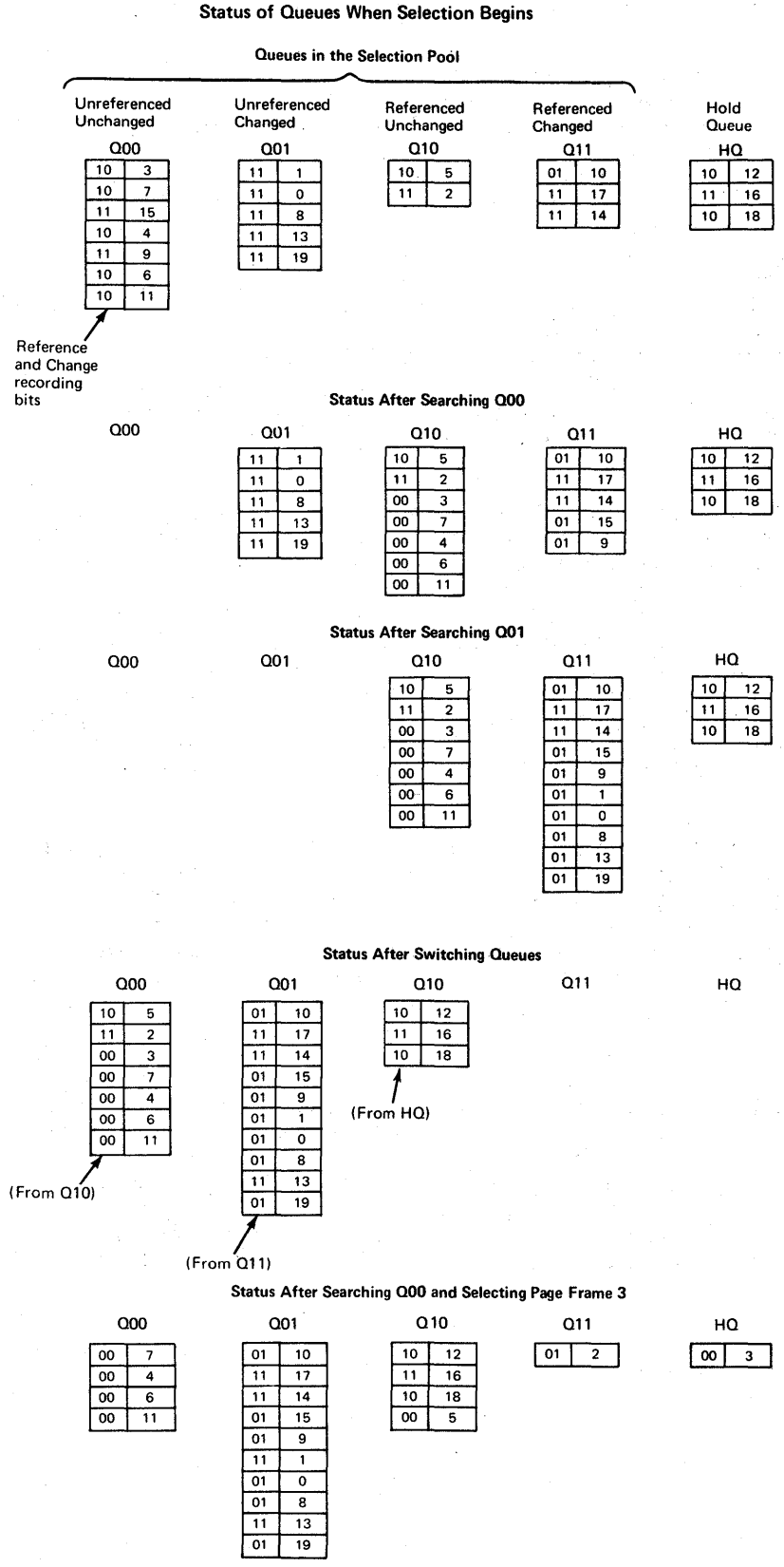


Figure 80.25.3. Operation of the page selection routine

Page I/O Routine

The page I/O routine is entered after a page frame has been selected for allocation in order to initiate one of three paging operations: a page-in only, a page-out only, or a page-out immediately followed by a page-in to the same page frame. The page I/O routine initiates one paging operation at a time consisting of one page read or write request.

Before issuing a page I/O request, the page I/O routine inserts the real address of the page frame involved in the read or write CCW of the paging channel program. Using the address of the virtual storage page associated with the paging operation, the page I/O routine determines the disk address of the slot to be accessed by the paging I/O operation. When a page-out followed by a page-in is required, the page I/O routine builds the two required channel programs and issues two paging I/O requests.

A request for a paging I/O operation is placed at the end of the I/O request queue for the direct access device that contains the page data set. A paging I/O request is not given priority over nonpaging I/O requests that may also be queued for the paging device. The rotating scheduling technique used for I/O initiation on a given channel in a DOS Version 4 environment is also used in DOS/VS. Hence, the paging I/O device is not given initiation priority over other I/O devices connected to the same channel. Since the page data set is not given any special initiation priority for I/O operations, it should not be placed on a direct access volume that contains high-activity files.

Verification of page-out I/O operations is not performed for performance reasons. If an uncorrectable I/O error occurs for a page-in or page-out operation, the system is abnormally terminated.

Servicing Page Requests

The way in which page requests are serviced by page management routines is as follows.

Page Faults. When a page fault occurs for a user-written task that operates with the CPU enabled for interruptions or for a system task that is permitted to cause a disabled page fault, a page fault request is placed in the page queue to be processed by the page manager system task. The processing of a page fault involves execution of the page selection routine to choose a page frame for allocation to the page that caused the page fault.

The page frame selected may be available or currently assigned to some other page. If the selected page frame is currently unassigned, the required page is read into the page frame, unless the associated user bit in the page table entry for the page indicates a page-in is not necessary. If a page-in is not required, the allocated page frame is cleared to zero for data security protection. If the page frame selected is currently assigned, a page-out is performed before the page-in is initiated if the change bit for the selected page frame is on. Otherwise, a page-out is not required.

The selected page frame is moved to the end of the hold queue and its reference and change bits are set to zero. Appropriate page table entries are altered to indicate page frame assignment and, when a page has been replaced, page frame unassignment. Figure 80.25.4 illustrates the logical flow of page fault processing.

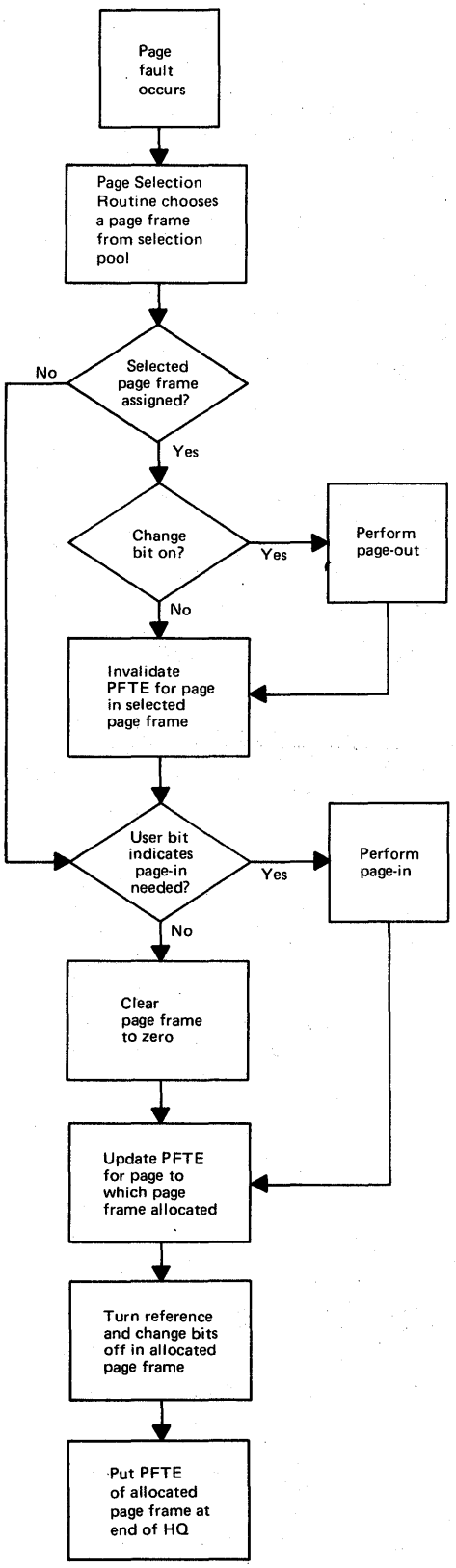


Figure 80.25.4. Logical flow of page fault processing

PFIX and PFREE Macros. Support of these macros can be included in the supervisor during system generation. When this option is selected, virtual mode problem programs can issue PFIX macros to cause pages to be permanently fixed and PFREE macros to unfix these pages. If a PFIX or PFREE macro is issued by a program that is running in real mode, the request is ignored. If a PFIX is issued for a page whose permanent fix counter contains a value of 255, the requesting task is canceled. PFIX requests are also issued to refix the required pages when a checkpointed program is being restarted.

When a program that uses PFIX and PFREE executes in a virtual partition, the corresponding real partition must have virtual storage allocated. Page frames with real storage addresses equal to the virtual storage addresses assigned to the real partition are the only page frames that can be allocated when a PFIX macro is issued by a program executing in the corresponding virtual partition. Hence, the maximum number of pages that can be permanently fixed by a virtual mode program that is executing in any virtual partition is equal to the number of pages contained in the associated real partition.

The PFIX macro supplies the virtual storage address(es) of the page(s) within a virtual partition that are to be permanently fixed. The PFIX routine handles one page at a time when multiple pages are to be fixed. The PFIX routine determines whether the page to be fixed is currently in real storage.

If the page is present in real storage and already permanently fixed (for example, when multitasking support is present in the supervisor), the permanent fix counter for the page is incremented by one and control is returned to the requester. If a page is present in real storage but not permanently fixed, the page frame the page is allocated is inspected to determine whether the page frame is eligible for permanent fixing. A page frame is eligible only if its address is equal to the address of one of the virtual storage pages in the real partition that is associated with the virtual partition from which the PFIX macro was issued.

If the page frame is eligible, it is marked permanently fixed and its PFTE is removed from the selection pool. If the page frame is not eligible and if the page to be fixed is not marked temporarily fixed, the PFIX routine attempts to select an eligible page frame. The one chosen is the first eligible page frame that currently is not permanently or temporarily fixed. The page to be fixed is moved to the selected eligible page frame and marked permanently fixed. The PFTE for the selected page frame is removed from the selection pool. If the selected eligible page frame was assigned to another page, its contents are moved to the vacated ineligible page frame (contents of selected and ineligible page frames are exchanged).

If the page to be permanently fixed is present in an ineligible page frame and marked temporarily fixed, the request cannot be processed until the page is unfixed. The PFTE for the ineligible assigned page frame is marked as containing a page that is not fixable (NFF bit is turned on) and the requesting task is placed in the wait state. Once the page has been unfixed (temporary fix counter goes to zero), the page-not-fixable bit is turned off for the page and the PFIX request can be processed, as previously described for the situation in which the page is present in an ineligible page frame and not temporarily fixed.

When the page to be permanently fixed is not present in real storage, the PFIX routine selects an eligible nonfixed page frame, if possible, marks the page frame not fixable (NF bit is turned on), saves the address of the selected page frame, and queues the PFIX request in the page queue. When the page manager system task processes the PFIX

request, a page frame is obtained using the normal selection procedure and a page-in is performed.

A page-in is always performed when a PFIX is issued for a page that is not present in real storage, unless the user bit indicates a page-in is not required. (There is no way to indicate in the PFIX macro that logically a page-in is not required, such as when the page to be fixed contains an I/O area. However, the RELPAG macro can be issued before the PFIX macro to release the page frame without a page-out.)

At the completion of the page-in, the loaded page is moved to the previously selected eligible page frame and marked permanently fixed. The contents of the selected eligible page frame, if any, are moved to the vacated page frame. The PFTE for the eligible page frame is removed from the selection pool.

If a task issues a PFIX macro to fix a page in a virtual partition for which another PFIX request is still being processed, the task is placed in a PFIX-bound wait state. A return code indicates whether a PFIX request was satisfied. Either all the pages indicated in a PFIX request are fixed or none are fixed.

A PFIX request cannot be satisfied if it requires more page frames that can be allocated to the defined associated real partition or if all the page frames allocated to the real partition are currently marked permanently fixed. However, if no eligible page frames are available because some are temporarily fixed, the PFIX request can be handled once unfixing occurs.

If the fast CCW translation option is present in the supervisor, the page frames associated with the channel programs in the copy blocks saved by the fast CCW translation routine are released. The real partition is again checked for available page frames. If there still are no available page frames, a bit is turned on for the partition that will cause the fast CCW translation routine to release the page frames associated with currently active channel programs when the channel programs complete.

Whether or not fast CCW translation is present in the supervisor, all the page frames in the real partition are then marked not fixable (NF bit is turned on) and the requesting task is placed in the wait state until a TFREE request causes an eligible page frame to be unfixed. Figure 80.25.5 illustrates the logical flow of PFIX processing.

The PFREE macro supplies the addresses of the pages that are to be unfixed. The PFREE macro causes the PFIX counter for each page specified to be decremented by one. If a PFREE request causes the PFIX counter to go to zero and the page is not temporarily fixed, the PFTE for the page frame that contains the unfixed page is placed in the selection pool (at the end of the hold queue). The page is then available for a page-out as per the page replacement algorithm.

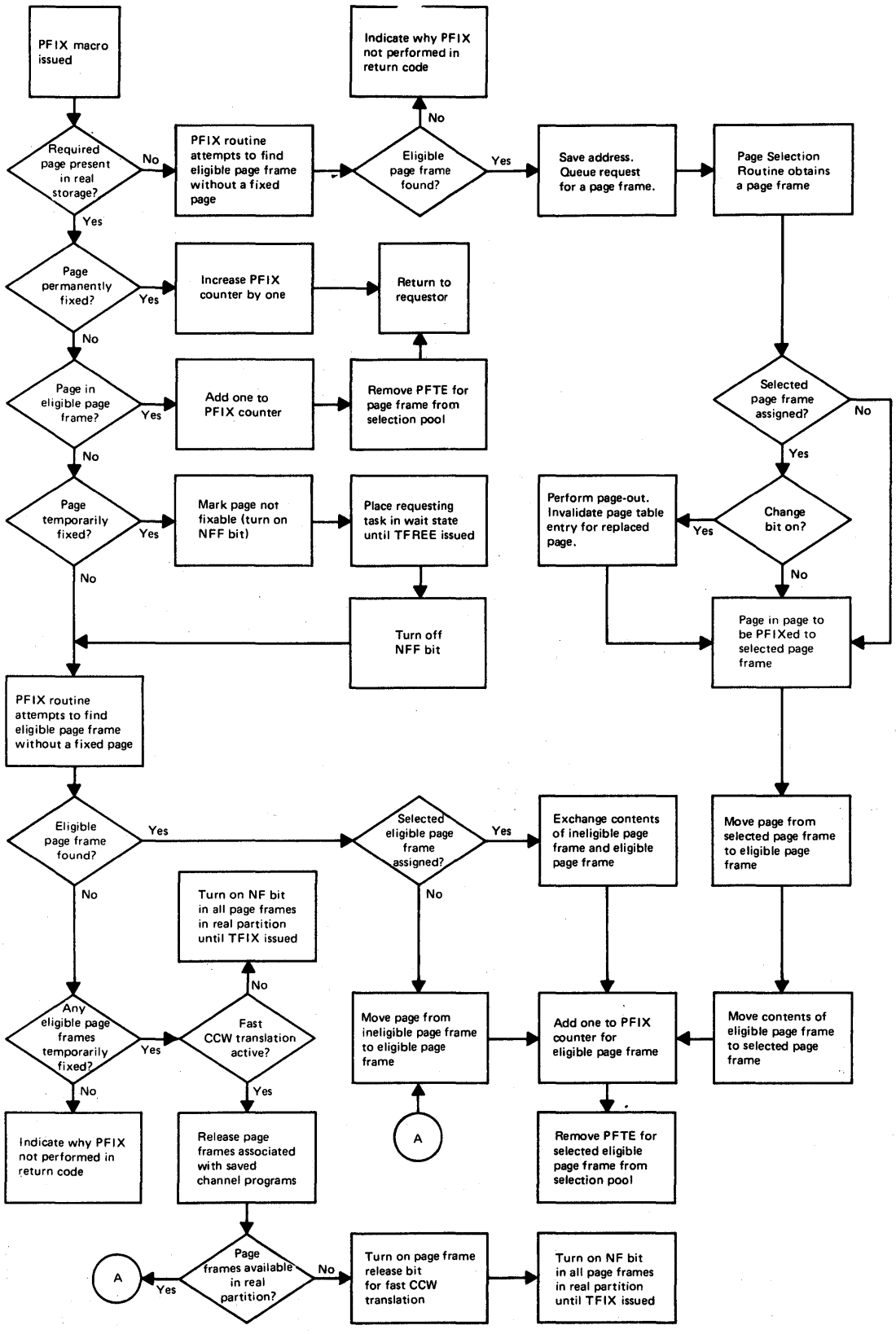


Figure 80.25.5. Logical flow of PFIX macro processing

TFIX/TFREE Requests. The channel program translation routine and the program fetch routine (when loading a virtual mode program) issue requests to temporarily fix and free pages used in I/O operations. The TFIX routine receives control when a temporary fix request is made. If the page indicated is already present in real storage and already temporarily or permanently fixed, the TFIX routine increases the temporary fix counter for the page by one, and control is returned to the requester.

If the page to be temporarily fixed is already present in real storage in a page frame that is temporarily fixable and the page itself is temporarily fixable but not yet marked temporarily fixed (both nonfixable bits for the page frame are off), the TFIX routine processes the request by adding one to the temporary fix counter for the page frame. The PFTE for the page frame is removed from the selection pool. If the page to be temporarily fixed is in real storage and temporarily fixable but is present in a page frame that is not temporarily fixable (NF bit is on as a result of a pending PFIX or GETREAL request), the TFIX routine inspects the selection pool to locate a page frame that is temporarily fixable.

First, Q00 is inspected for an available page frame. If one is not found, the entire page frame table is inspected. The first usable page frame that is part of the selection pool and not marked nonfixable (NF bit is off) is selected for allocation. The page in the nonfixable page frame is moved to the selected temporarily fixable page frame and marked temporarily fixed. If the selected temporarily fixable page frame already contained a page, this page is moved to the nonfixable page frame (contents of nonfixable and fixable page frames are exchanged). The page frame containing the temporarily fixed page is removed from the selection pool.

The TFIX request cannot be serviced by the TFIX routine and is placed at the end of the page queue if (1) the page to be fixed is present in a page frame that is not temporarily fixable and the TFIX routine cannot find a page frame that is temporarily fixable, (2) the page to be temporarily fixed is not in real storage, or (3) the page is present but marked not temporarily fixable (NFF bit is on) as a result of a pending PFIX request.

When the page manager system task processes a TFIX request, the page selection routine is entered to select a page frame that is temporarily fixable if (1) a page frame was not previously available or (2) the page was not present in real storage. If a page frame cannot be allocated because the selection pool is currently empty, the routine previously described is entered to reset certain system routines, if possible, to make page frames available.

When a page frame can be allocated for the first situation (a page frame was not previously available), the page is moved from the nonfixable page frame to the selected fixable page frame and marked temporarily fixed. If the selected fixable page frame was already assigned, its contents are moved to the vacated nonfixable page frame. The PFTE for the selected temporarily fixed page frame is removed from the selection pool.

When a page frame can be allocated for the second situation (page was not present in real storage), a page-out is performed if the change bit for the selected fixable page frame is on. The page to be fixed is paged into the selected fixable page frame and marked temporarily fixed. The PFTE for the selected temporarily fixed page frame is removed from the selection pool.

If the TFIX request was queued because the page was present in real storage and marked not temporarily fixable, the request can be serviced

when the page manager encounters it because this situation no longer exists. (The PFIX request that caused the TFIX request to be queued will already have been processed.) The TFIX request is handled as already described depending on whether the page to be fixed is present in a temporarily fixable page frame. Figure 80.25.6 shows the logical flow of TFIX routine processing.

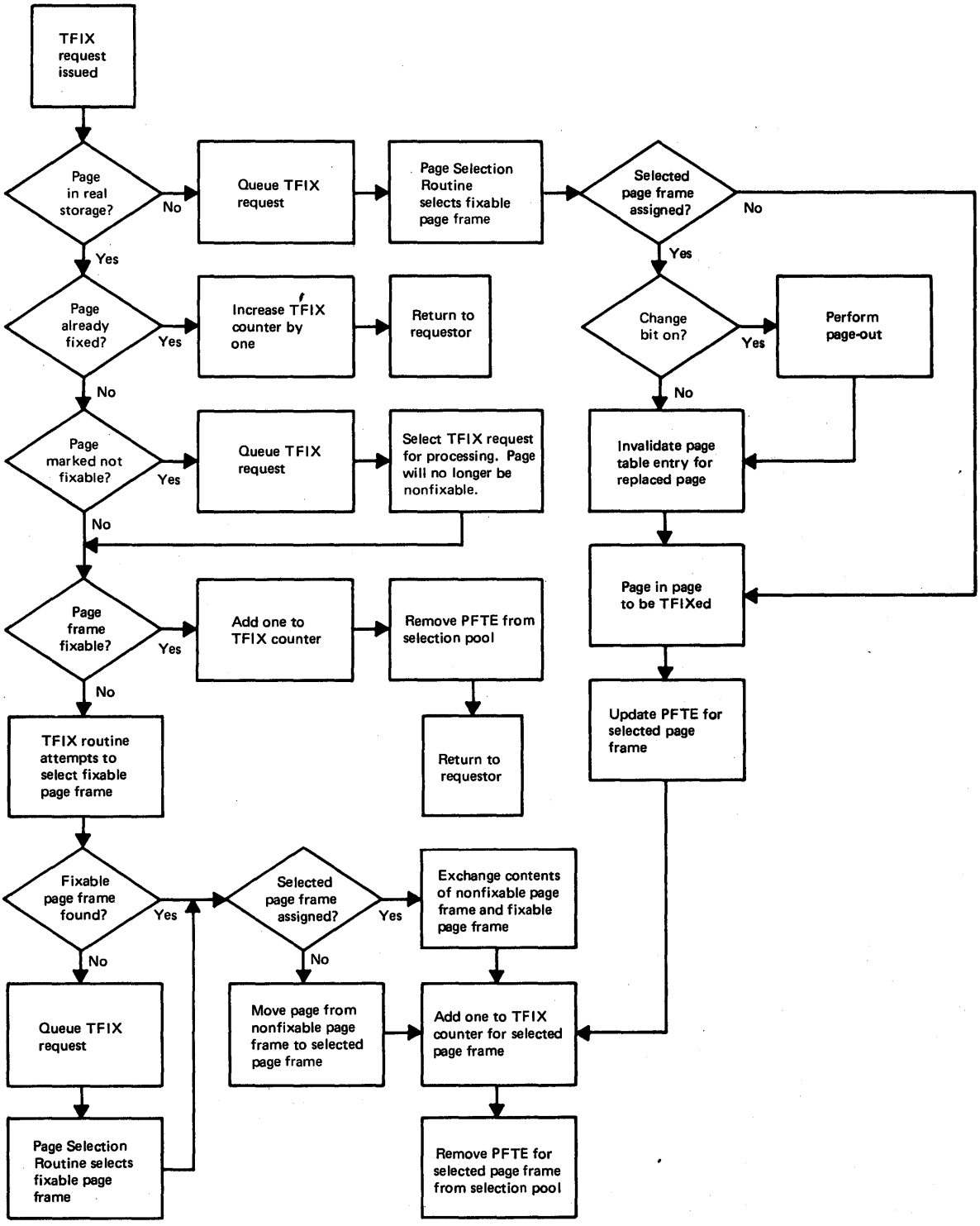


Figure 80.25.6. Logical flow of TFIX routine processing

The TFREE routine receives control when a temporary free request for page(s) is issued. Routines such as the channel program translation extension and program fetch issue TFREE requests to release temporarily fixed pages.

The temporary fix counter is decremented by one. If this TFREE request causes the TFIX counter for the page to go to zero and the PFIX counter for the page is also zero, the page is no longer fixed and the page frame in which it is contained is returned to the selection pool. Any task that is waiting for this page to be unfixed (such as one that issued a PFIX request) or for the page frame the page is assigned (such as one that issued a GETREAL or PFIX request that involves the page frame) is posted ready. If the program fetch routine issued the TFREE request, the PFTE for the affected page frame is placed at the beginning of Q00 or Q01 depending on the setting of the change bit in the page frame. Otherwise, the PFTE for the page frame is placed at the end of the hold queue.

GETREAL/FREEREAL Requests. The GETREAL routine is entered during the initiation of a real mode program in order to allocate page frames to the virtual storage pages in the real partition to be used. The GETREAL routine is also called when required to allocate real storage for the PDAID alternate buffer area or the SDAIDS buffer area. The FREEREAL routine is entered during real mode job step termination processing to make available the page frames that have been allocated by the GETREAL routine.

When a GETREAL request is issued, the low and high virtual storage addresses of the real partition, SDAIDS buffer, or PDAID buffer to which real storage is to be assigned are passed to the GETREAL routine. In order for the request to be satisfied, all page frames with addresses equal to those in the indicated virtual storage area must be available. That is, the required page frames must be unassigned or assigned only to pages that are not temporarily fixed. (None of the required page frames can contain permanently fixed pages since this is possible only if a virtual mode program were executing in the virtual partition corresponding to the real partition that is currently being initiated. This situation is not permitted.)

If there is a temporarily fixed page in one or more of the required page frames, processing of the GETREAL request is delayed until all these temporarily fixed pages are unfixed. If the GETREAL request is for real storage for a real partition, initiation of the real mode job step is also delayed.

When the GETREAL routine receives control, first it marks the PFTE's for all the required page frames not temporarily fixable (NF bit is turned on). This is done to prevent any temporary fixing of these required page frames as processing of the GETREAL request proceeds.

A check is then made to determine whether any of the required page frames contain a temporarily fixed page. If so and the fast CCW translation option is not present in the supervisor, the requesting task (job control program) is placed in the wait state with an indication that it is waiting for required temporarily fixed pages to be unfixed. As these pages become unfixed, the job control task is posted. Once all the required pages are unfixed, the GETREAL request can be processed.

When the required area contains temporarily fixed pages and the fast CCW translation option is present in the supervisor, the page frames currently allocated to the channel programs in the copy blocks saved by the fast CCW translation routines are released. The required area is again checked for temporarily fixed pages. If it still contains such pages, a bit is set on for the partition to cause the fast CCW

translation routine to release the page frames associated with the currently active channel programs when the channel programs complete.

If none of the required page frames are temporarily fixed when the GETREAL routine is entered or after the required temporarily fixed page frames have been unfixed, a request for the first required page frame is placed in the page queue by the GETREAL routine to be processed by the page manager routine. Once this request is satisfied, a request for the next required page frame is queued. Requests are issued in ascending page frame address beginning with the lowest addressed page frame. This process continues until all required page frames have been assigned. The PFTE's for the assigned page frames are removed from the selection pool.

If the page manager finds the first of the required page frames marked unusable (malfunctioning bit is on), the entire GETREAL request is not processed (no real storage area is allocated). If a required page frame after the first is found to be marked unusable, processing of the GETREAL request terminates. The allocated area consists of the page frames allocated up to this point in processing of the request. The page frames above the unusable page frame that are required to satisfy this GETREAL request are marked temporarily fixable (NF bit is turned off). If the GETREAL request is for a real partition, the partition communication region is updated to indicate the upper byte of the real partition actually allocated.

The page frames allocated to a real partition may be available or assigned to a nonfixed page of some other task at the time they are allocated to satisfy a GETREAL request. When a required page frame is already assigned to another page, it is taken away from that page without the allocation of another page frame to that page.

A check is made to determine whether the page being replaced has been changed. If it has been, the page is paged out before the page frame to which it was assigned is cleared to zeros and allocated to the real partition. The page table entry for each page that is so replaced is changed to indicate the fact that real storage is no longer allocated to the page. Similarly, the page table entry for each page in the real partition is updated to reflect the allocation of real storage.

The FREEREAL routine is entered to process requests to free the real storage allocated via a GETREAL request. The PFTE's for the page frames to be freed are marked available and temporarily fixable and placed at the beginning of the selection pool (top of Q00). The reference and change bits for the freed page frames are turned off. Page table entries are invalidated as required. Figure 80.25.7 shows the logical flow of GETREAL routine processing.

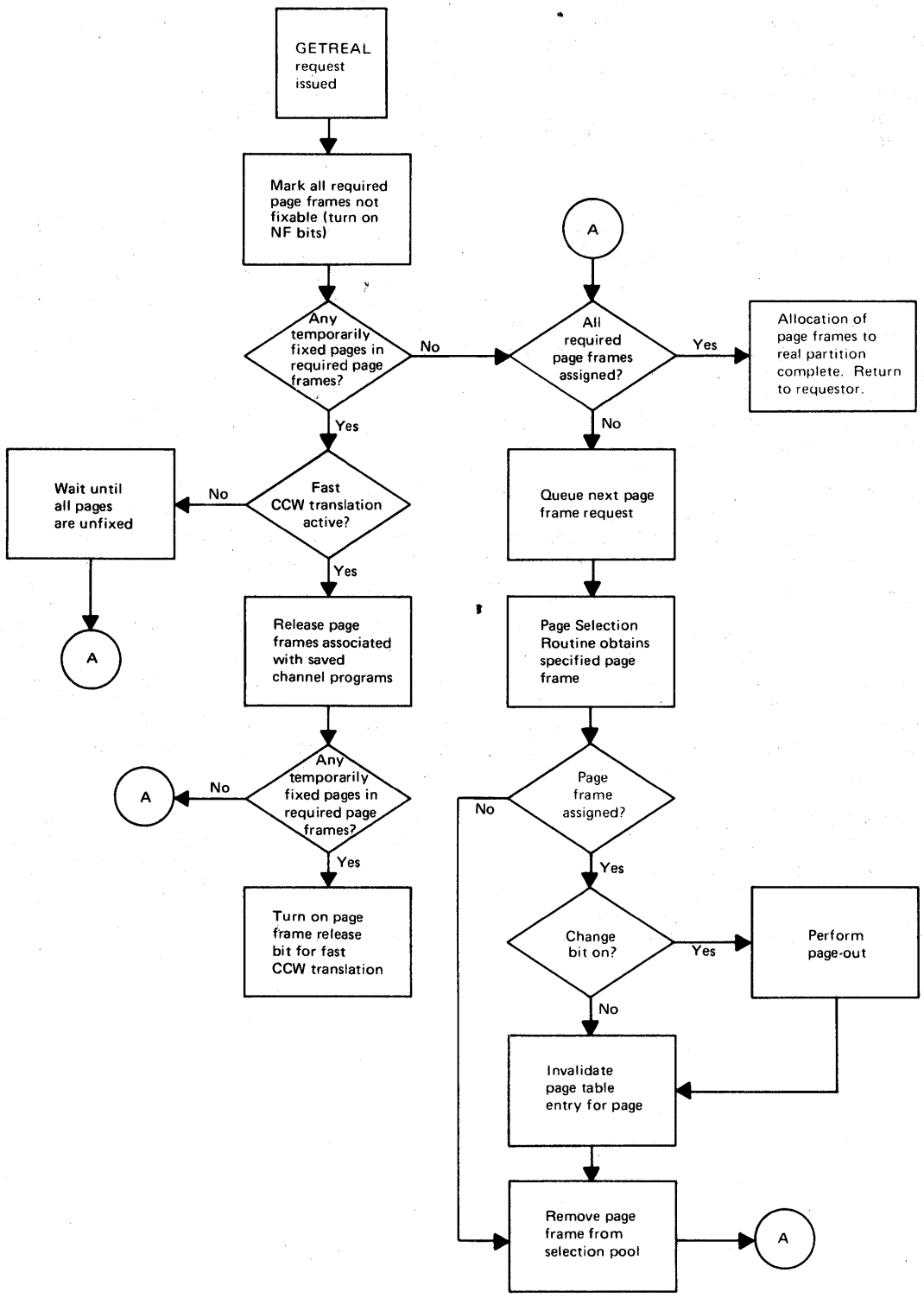


Figure 80.25.7. Logical flow of GETREAL routine processing

RELPAG, FCEPGOUT, and PAGEIN Macros. Support of these macros is a system generation option (PAGEIN parameter of the SUPVR macro). They enable a virtual mode problem program to control page-in and page-out operations within the virtual partition in which it is executing. The RELPAG macro can be issued to cause the page manager to make any page frames that are allocated to the specified virtual storage area available for reallocation.

Only virtual storage pages that are completely contained both within the specified virtual storage area and within the virtual partition from which the RELPAG macro is issued can have their page frames released. The RELPAG macro can be issued for pages that are no longer required and whose contents are not to be saved even if they are changed.

The following can occur when a RELPAG macro is issued:

- All the virtual storage pages that are completely within the specified area are within the virtual partition and any page frames assigned to these virtual storage pages are released. (Nothing is done for a virtual storage page that does not currently have a page frame assigned.)

- The release function is performed only for some of the virtual storage pages in the specified area. A release is not performed for those virtual storage pages that (1) are partially or completely outside the virtual partition, (2) are temporarily or permanently fixed, or (3) currently have a page request queued (PFIX, TFIX, or page fault pending).

- The request is ignored because the requesting program is operating in real mode.

- The requesting program is abnormally terminated because support of the RELPAG macro is not present in the DOS/VS supervisor.

Return codes indicate the actions taken by the page manager. The PFTE's for the page frames that are released during RELPAG macro processing are placed at the beginning of Q00 and their reference and change bits are set to zero. The appropriate page table entries are also updated (invalid and user bits are set to one). No page-out is performed for the released page frames. The next time a reference is made to a virtual storage page whose page frame has been released, a zeroed page frame will be assigned without a page-in operation.

The FCEPGOUT macro is used to make the page frames allocated to the specified virtual storage area available for reallocation and to cause a page-out of changed pages when reallocation occurs. The FCEPGOUT macro can be issued for pages that are no longer required whose contents are to be saved if they were changed.

Forced page-out processing is not performed on virtual storage pages that (1) are temporarily or permanently fixed, (2) have a page request queued, or (3) are not completely contained both within the specified virtual storage area and the requesting virtual partition. A request from a real mode program is ignored and the requesting program is abnormally terminated if support of the FCEPGOUT macro is not present in the DOS/VS supervisor. Nothing is done for virtual storage pages that do not currently have a page frame assigned. Condition codes are returned to indicate the processing performed.

When a virtual storage page is eligible for a forced page-out and has a page frame assigned, the reference bit in the allocated page frame is set to zero. The PFTE for the allocated page frame is moved to the beginning of Q00 if the change bit for the page frame is zero or to the beginning of Q01 if the change bit for the page frame is one. Hence, a

page-out of a changed page does not occur until the page frame it is allocated is taken for reassignment.

The PAGEIN macro is used to cause one or more pages to be brought into real storage before they are referenced if they are not already in real storage. The PAGEIN macro specifies one or more virtual storage areas within a virtual partition for which page-ins are to be performed. If all the specified virtual storage areas are not completely contained within the virtual partition from which the PAGEIN macro is issued or if the request is issued by a real mode program, the PAGEIN request is ignored.

Processing of a PAGEIN macro is as follows. The PAGEIN macro SVC routine receives control when the PAGEIN macro is issued. The SVC routine checks for a valid request (all areas are within the virtual partition). If the request if valid, the SVC routine constructs an entry in a PAGEIN macro request table. The entry describes the request and identifies the requesting task. The size of this table is determined at system generation based on the value specified in the PAGEIN parameter in the SUPVR macro. The table is maintained in first-in, first-out sequence. A PAGEIN macro is ignored if there are no available entries in the table when it is issued. Whenever a task is terminated, this table is scanned. Any requests that belong to the task are deleted.

After a valid PAGEIN macro request has been placed in the PAGEIN macro request table, the SVC routine passes control directly to the PAGEIN task if possible (no other higher priority system tasks are ready to execute). When the PAGEIN task receives control either directly or via the normal task selection procedure, it processes the next request in the PAGIN macro request table. The PAGEIN task executes asynchronously with the tasks that issue PAGEIN macros. For each request, the PAGEIN task processes each page that is contained within the area(s) specified in the request (PAGEIN macro).

This processing consists first of determining whether the page is presently in real storage. If it is and is also permanently or temporarily fixed, no further processing of the page is performed. If the page is present and not fixed, the PFTE for the page frame that contains the page is placed at the end of the hold queue and the reference bit in the page frame is set to zero. This gives the page lowest priority for page replacement.

If the page is not in real storage, the PAGEIN task places a page-in request for this page in the page queue. This request is then handled just as if it occurred as a result of a page fault except that there is no exit to the user-written page fault handling routine for the partition if such a routine is active.

Optionally, the PAGEIN macro can also specify an event control block (ECB). This ECB must be contained within the virtual partition from which the PAGEIN macro is issued or the PAGEIN macro is ignored. The ECB can be specified in a WAIT macro or tested via an instruction to determine whether processing of the PAGEIN macro has been completed and how. The ECB contains return information bits that indicate error conditions such as a full request table or specified areas outside of the virtual partition.

Partition Deactivation/Reactivation

The partition deactivation routine, which is always present in a multiprogramming configuration, monitors the paging activity of the system. When too much paging activity is occurring, as determined by established threshold values, partition deactivation is performed.

The deactivation routine is entered every time a certain number of page-ins occur. The time interval between two successive entries into the deactivation routine is called a measurement period. The deactivation routine calculates the exponential average of page-ins per second that occurred since the deactivation routine was last entered. This exponential average is one that takes into account the average number of page-ins per second that occurred during this and all previous measurement periods and gives proportionally less weight to the older measurements.

The exponential average of page-ins for a measurement period is also used to restrict the operation of fast CCW translation when this option is present in the supervisor. When the exponential average of page-ins is calculated, it is compared with a constant value that controls the restriction of fast CCW translation. As long as the exponential average of page-ins is less than this constant, fast CCW translation is not restricted.

When the calculated exponential average of page-ins equals or exceeds the constant value for fast CCW translation restriction, a bit is set on to cause the fast CCW translation routine to release all page frames associated with each active channel program as soon as the channel program completes. This bit is turned off when the exponential average of page-ins drops below the constant value provided the threshold minimum time period that is used to control conditional partition reactivation (discussed below) has elapsed since the last time the bit was turned off.

A reentry rate for the measurement period is also calculated. This rate is the total number of page-ins that occurred during the measurement period for all pages in the virtual address area that were paged out previously during this measurement period. The deactivation routine compares the reentry rate for the measurement period against a reentry threshold value and the calculated exponential average of page-ins for the measurement period against a page-in rate threshold value. If both thresholds were equalled or exceeded during the measurement period, the deactivation routine is entered to select a virtual partition for deactivation. The deactivation routine is not entered if any task seized the system.

The lowest-priority virtual partition in operation that (1) is not in the process of being canceled, (2) does not have a request in the page queue (if it uses the page fault handling overlap facility), and (3) is not using the logical transient area is selected when deactivation is required. Real partitions are never deactivated.

After the selected virtual partition is suspended (all tasks in the partition marked deactivated), the nonfixed page frames currently assigned to the virtual partition are made available for reallocation. Fixed pages are not released. When a page frame is released as a result of deactivation, its PFTE is placed at the beginning of Q00 or Q01 depending on whether its change bit is off or on, respectively. All copy blocks saved by the fast CCW translation routine are released as are the temporarily fixed page frames associated with the channel programs in the saved copy blocks.

A count of the number of nonfixed pages that were present in real storage for the partition at the time of deactivation is retained. This is called the reactivation count. Any entries in the page queue for the deactivated partition are deleted. If the two threshold values are exceeded again after the first partition is deactivated, the next lowest-priority virtual partition is placed in deactivated status, if possible.

The reactivation routine is entered immediately before the system is to enter the enabled wait state because no task is ready to execute. If one or more virtual partitions are currently deactivated and either of the following conditions exist, the highest-priority deactivated virtual partition is unconditionally reactivated:

- No other virtual partitions are currently active
- No I/O requests are queued for any device in the system configuration except requests for console or teleprocessing devices, or requests with an error condition

When unconditional reactivation is not required, conditional reactivation processing is performed as follows. If the threshold minimum time period has elapsed since the last time the reactivation routine was called and no paging I/O operation is currently in progress, the exponential average of page-ins that occurred since the last time the reactivation routine was entered is calculated. The highest-priority deactivated partition is reactivated if (1) the calculated exponential average of page-ins value is less than the established reactivation threshold page-in value and (2) the number of page frames in the selection pool is equal to or greater than the reactivation count for the highest-priority deactivated partition. Otherwise, no partition is reactivated.

The five threshold values used by the deactivation and reactivation routines are determined by the system and cannot be specified by the user at system generation. Threshold values vary depending on the System/370 model and the direct access device type that is used for the page data set.

Teleprocessing Balancing

The teleprocessing balancing option is automatically included in a multiprogramming DOS/VS supervisor that contains any teleprocessing access method (BTAM, QTAM, or VTAM). In a mixed teleprocessing and batch environment, it enables teleprocessing performance to be improved at the expense of degraded performance for one or more batch partitions. This facility is designed to be used only by teleprocessing access methods and data base/data communications interface programs, such as CICS/VS.

The teleprocessing balancing facility is activated when the operator issues the TPBAL attention command to specify the number of batch partitions that can be deactivated. The system selects the lowest priority partitions as being eligible for deactivation by the teleprocessing balancing facility and notifies the operator of the affected partitions. The procedure for deactivating partitions is modified so that the number specified in the TPBAL command is the maximum number of partitions that can be deactivated by the teleprocessing balancing facility.

When the teleprocessing balancing facility is active, the partitions eligible for deactivation are displayed whenever a PRTY command is issued to alter or display partition priority. A TPBAL command without any operands can be issued at any time to determine whether the teleprocessing balancing facility is active and to determine the partitions that can be deactivated by the facility.

The TPIN and TPOUT macros are provided to cause actual partition deactivation and reactivation, respectively, when the teleprocessing balancing facility is active. These macros have no effect if they are issued when teleprocessing balancing is inactive.

The TPIN macro should be issued by a teleprocessing access method when it requires priority for use of the resources of the system, such as immediately before each time a transaction is to be processed. The TPIN macro causes the three deactivation threshold values to be set to zero and partition reactivation to be disabled. An indication is also set to prevent deactivation of the partition from which the TPIN macro was issued.

After a TPIN macro has been issued, the next time the page manager is entered to handle a page-in request, the number of lowest priority active virtual partitions specified in the TPBAL command are deactivated if they do not own the B-transient area and are not scheduled for cancellation. No deactivation occurs if the only active virtual partition is the one from which the TPIN macro was issued.

When the teleprocessing access method no longer requires priority for use of the resources of the system, such as at the completion of processing each transaction, the TPOUT macro should be issued. This macro enables partition reactivation and sets an indicator that will cause the page manager to restore the three deactivation threshold values to what they were before the last TPIN macro was issued. When the reactivation routine is next entered, unconditional reactivation of the deactivated partitions is attempted.

Note that when partitions are deactivated as a result of a TPIN macro, they are not reactivated until a TPOUT macro is issued.

80:30 DATA MANAGEMENT

Data management routines in DOS/VS are modified as required to support and operate in a virtual storage environment. The channel scheduler is extended to support block multiplexing, channel program translation, and page fixing. An interface to the channel scheduler extension is also provided to enable channel program translation to be handled by problem programs. In addition, rotational position sensing (RPS) and two new access methods (virtual storage access method and virtual telecommunications access method) are supported. DOS/VS also supports I/O devices that are not supported by DOS Version 3 or 4 and other enhancements are provided as well.

The capability of handling up to 24 sense bytes of data for an I/O device, instead of a maximum of six, is provided in DOS Version 4. However, only six sense bytes are displayed in an I/O error message. In DOS/VS, up to 24 sense bytes are handled and also displayed with I/O error messages. Another DOS/VS enhancement gives the operator the capability of adding a variable number of tracks to a sequential disk file when the allocated extents become filled during problem program execution.

Several incompatibilities between DOS files and OS data sets are removed in DOS/VS to facilitate data interchange between DOS/VS and OS/VS as follows:

- The default values for the generation number and version number fields in a HDR1 tape label are changed in DOS/VS from 0001 and 01, respectively, to EBCDIC blanks since OS/VS does not use these fields.

- The volume serial number specified in EXTENT statements is left-justified and padded on the right with blanks, as is done in OS/VS, instead of right-justified and padded on the left with zeros, as is done in DOS Version 4. The volume serial number in TLBL statements in DOS/VS can be enclosed in quotes and is handled as in OS/VS (left-justified and padded on the right with blanks).

- DOS/VS ensures that the block count field in a HDR1 tape label contains zeros so that OS/VS can process the tape correctly. The TPLAB (but not the TLBL) statement can be used to place a value in this field that causes erroneous processing by OS/VS.

- DOS/VS will flag disk files that are DOS libraries (system or private) in the file type code field in the Format 1 disk file label. This flag identifies these DOS files as not interchangeable with OS/VS since DOS library organization and OS library organization are not compatible.

- Support of a spanned record that spans two volumes (last physical block of one volume and first physical block of the next volume), which is provided in OS/VS, is also provided in DOS/VS for both input and output files.

ACCESS METHODS

SAM, ISAM, and DAM provide the same facilities in DOS/VS and DOS Version 4, and the same restrictions (such as CCW's with a count no larger than 32K) apply to both DOS versions. In DOS/VS, however, these access methods also support rotational position sensing. SAM, ISAM, and DAM use the channel program translation and page fixing capabilities of the channel scheduler extension.

The functions supported by BTAM and QTAM are the same in DOS/VS and DOS Version 4. BTAM can operate in virtual or real mode. QTAM message control programs and QTAM message processing programs must operate in real mode. When a program that uses BTAM operates in virtual mode, a corresponding real partition must be defined for the virtual partition. This is required because before issuing an EXCP macro, BTAM issues the PFIX macro to fix all pages that will be referenced during the I/O operation so that a page fault cannot occur and delay an immediate response from the communications device. At the completion of the I/O operation, a PFREE is issued to unfix all the pages that were fixed with the PFIX macro. BTAM performs its own channel program translation.

VTAM is an optional teleprocessing access method that can be included in a DOS/VS system that also contains BTAM and/or QTAM. VTAM must execute in a virtual partition and the corresponding real partition must be defined as well, since VTAM uses PFIX and PFREE macros. The minimum VTAM virtual partition size is in excess of 700K bytes.

The application programs that use VTAM must execute in a partition that has a lower priority than the VTAM partition. VTAM application programs can execute in virtual or real mode. When VTAM support is included in a DOS/VS supervisor, support of three partitions is automatically included during system generation if three or more partitions are not user-specified. VTAM uses the conditional swapping instructions (COMPARE AND SWAP and COMPARE DOUBLE AND SWAP).

VTAM requires the presence of the following DOS/VS features: STXIT macro support, GETVIS and FREEVIS macros, PFIX and PFREE macros, EXCP macro with the REAL parameter, relocating loader, time-of-day clock, and multiple wait.

The DOS/VS high-level languages do not support a parameter that can be used to cause I/O areas to be aligned on page boundaries. However, if the load point of a phase is page-aligned using the PDBY parameter, an Assembler Language programmer can define I/O areas that are page-boundary-aligned and/or ensure that buffers are packed such that they do not cross page boundaries when they are smaller than 2K in size.

SUPPORT OF ADDITIONAL I/O DEVICES

The following I/O devices are supported by DOS/VS but not by DOS Version 4:

- 3203 Printer and 5203 Printer as a programmer logical unit, SYSLST, or SYSLOG device (like the support provided for the 3211 Printer). The 3203 and 5203 are also supported by the Assembler language translator, POWER/VS, System Utilities, OLTEP, the Model 20 emulator, and the 1401/1440/1460 emulator.

- The 2560 Multifunction Card Machine and 5425 Multifunction Card Unit as a programmer logical unit, SYSRDR, SYSIPT, or SYSPCH device. The 2560 and 5425 are also supported by the Assembler Language translator, POWER/VS, System Utilities, OLTEP, and the DOS/VS distribution program.

- 3340 Direct Access Storage Facility and 3330-series Model 11 disk storage for the same functions as 3330-series Model 1 and 2 disk storage (except that ISAM does not support the 3330 Model 11). Dynamic data module size recognition is provided (module size need not be specified with the 3340 device parameter) and multivolume sequential files can reside on different 3348 data module types. Only DLBL and EXTENT job control statements are accepted for 3340 files. VOL, DLAB, and XTENT statements cannot be used.

Note that the PUB table entry for a 3340 device does specify whether the drive has the optional RPS feature installed. However, it does not indicate whether the optional Fixed Head feature is present.

- 3350 Direct Access Storage (all modes) and 3344 Direct Access Storage for the same functions as 3330-series and 3340 disk storage, except that ISAM does not support the 3350. ISAM programs can utilize 3350 (and 3330 Model 11) devices by utilizing VSAM and the ISAM Interface Program.
- Models 4, 6, and 8 of the 3420 Magnetic Tape Subsystem at 6250-BPI density
- 3540 Diskette Input/Output Unit as a programmer logical unit, SYSRDR, SYSIPT, SYSPCH, or SYSLST device. The 3540 is also supported by POWER/VS, System Utilities, and OLTEP. Only DLBL and EXTENT job control statements are accepted for the 3540. VOL, DLAB, and XTENT statements cannot be used.
- 3600 Finance Communication System, 3650 Retail Store System, and 3660 Supermarket System
- 3740 Data Entry System
- 3767 Data Communication Terminal
- 3770 Data Communication System
- 3780 Data Communication Terminal
- 3790 Communication System
- 3800 Printing Subsystem
- 3881 Optical Mark Reader
- 3886 Optical Character Reader

THE CHANNEL SCHEDULER

Block Multiplexer Channel Support

Support of block multiplexing is optional. It is automatically included in a DOS/VS supervisor when rotational position sensing support is requested. Block multiplexing support can improve channel utilization when I/O devices that are capable of disconnecting from a block multiplexer channel during operation of a channel program are present in the system configuration, such as direct access devices with rotational position sensing (3330-series, 3340, 3344, and 3350) 3270 display system units, and buffered unit record devices (2540, 3505, 3525, 1403, 3211, 3800, for example).

When block multiplexing support is present in a supervisor, block multiplexer mode of operation is always established for all installed block multiplexer channels during IPL (the channel mode bit in control register 0 is set to block multiplexer mode) and channel-available interruptions are processed. Block multiplexer mode remains effective during system operation and cannot be changed to selector mode by the operator.

If the seek separation option is not present in the supervisor, chained seeks instead of standalone seeks are issued for all movable arm direct access devices. This provides seek overlap for direct access devices that have the RPS feature but not for direct access devices without the feature. If the seek separation facility is present in the supervisor, standalone seeks are issued for all movable arm direct access devices that do not have the rotational position sensing facility. Chained seeks are still issued for direct access devices that have the RPS feature.

Hence, if direct access devices with and without RPS are included in a configuration in which block multiplexing support is present, seek separation should be included in the supervisor to provide seek overlap for non-RPS direct access devices.

Note that block multiplexing support cannot be included in a DOS/VS supervisor for a Model 115 or 125 that is to use the 2311/3330, 2311/3340, or 2314/3340 Compatibility feature. Block multiplexing support can be included in a DOS/VS supervisor for a Model 135 or 138 that is to use the 2314/3340 Compatibility feature. In this case, block multiplexing is supported for 3340 devices that are being used in native mode but not for those that are emulating 2314/2319 devices.

Rotational Position Sensing Support

Rotational position sensing support is optional. It requires the inclusion of block multiplexing support and GETVIS/FREEVIS macros in the DOS/VS supervisor. The relocating loader must also be present in the DOS/VS system. RPS support is provided for 3340 direct access devices that have the RPS feature and for 3330-series, 3344, and 3350 direct access devices, for which RPS is a standard hardware feature. RPS support is utilized by system routines for which an advantage can be gained and by the access methods that support 3330-series, 3340, 3344, and 3350 direct access devices. Specifically, RPS is supported by the following:

- All the access methods that support direct access devices (SAM, DAM, ISAM, and VSAM). RPS is used in the I/O operations performed by the access method logic modules but not for the I/O operations performed by the OPEN and CLOSE routines. The access methods support RPS only for problem programs that execute in a virtual partition. RPS support is not provided for real mode programs. Any problem programs (user-written, language translators, program products, etc.) that use these access methods have RPS support provided.
- IPL and Job Control
- Program fetch and page manager system tasks
- POWER/VS
- Linkage Editor
- Librarian
- Checkpoint/Restart
- System Utilities

DOS/VS access method support of direct access devices consists of a set of logic modules without RPS support and another set with RPS support. The RPS version of a logic module is generated when the RPS parameter is included in the SDMOD, CPMOD, DIMOD, DAMOD, or ISMOD macro. The RPS version of a logic module is larger than the non-RPS version,

assuming the same functions are supported. The RPS versions of the disk logic modules are reentrant and relocatable. They are contained in the system core image library and execute only in the SVA. This enables them to be shared by concurrently executing problem programs.

When a problem program is assembled or link-edited, the non-RPS versions of the required disk logic modules are included in the program. If RPS support is provided for any of the direct access devices assigned to the problem program when it executes, the non-RPS versions of the logic modules that are contained in the problem program are not used for these devices.

The DTF's that are defined in the problem program for disk devices are the same whether or not RPS support is to be used. (No RPS-related parameters are added to the existing disk DTF's.) A DTF extension that is required for RPS support is dynamically created in the GETVIS area within the virtual partition. Therefore, in order to use RPS support, a problem program must specify the SIZE parameter on the EXEC statement to make available in the virtual partition a GETVIS area of sufficient size. RPS support requires approximately 6K bytes in the partition GETVIS area.

RPS access method support is automatically provided for a given direct access device when the DTF for the device is opened and all the following conditions exist: RPS support is included in the generated DOS/VS system, the disk device is being accessed by a program that is executing in a virtual partition, the disk device has the RPS feature, and sufficient space in both the system GETVIS area and the user GETVIS area in the partition is available for the RPS logic module.

When the OPEN routine determines that RPS support is to be provided for a disk device, space in the GETVIS area of the partition is obtained for the DTF extension that is required for RPS support. This DTF extension area is used for the construction of RPS channel programs and contains other required work and save areas. The DTF in the problem program is modified to point to the location of the DTF extension and an indicator is set on in the DTF to identify it as one with an extension. The modified DTF is returned to its original state when it is closed.

OPEN also determines whether the RPS version of the required logic module is currently in the SVA. If not, space is obtained from the system GETVIS area in the SVA and the required RPS logic module is loaded in this space. The DTF in the problem program is then linked to this logic module in the SVA. If the required logic module is already present in the SVA as a result of having been loaded previously for another DTF, the partition usage indicator for the logic module is turned on.

A set of usage indicators is maintained in a directory list within the SVA that indicates the dynamically loaded RPS logic modules that are currently being used by the executing job in each active partition. These indicators are turned off at end of job. When all the partition indicators for a given dynamically loaded RPS logic module are off (indicating no partition is currently using the logic module), a FREEVIS macro is issued to free the space in the SVA that is occupied by that logic module. Any RPS logic module can be made permanently resident in the SVA by including it in the list of phases that are specified when the SVA is created during an IPL.

The RPS logic module used for a given DTF may be larger than the minimum logic module the DTF actually requires since superset logic modules are used for RPS support. This is done to reduce the total amount of space that is required in the SVA to contain the RPS logic modules that are required by concurrently executing programs.

The way in which RPS support is implemented eliminates the need to modify existing problem programs that use LIOCS access methods in order for them to use RPS support. The only user-written programs with logical IOCS that must be modified to operate correctly when RPS support is used are those that access fields in the disk DTF that are contained in the DTF extension in the partition GETVIS area when RPS is used.

User-written programs that utilize PIOCS should be modified to include sector commands in their disk channel programs so that they too use rotational position sensing. Otherwise, these non-RPS channel programs will monopolize channel time and eliminate the effectiveness of RPS support. (Using RPS and block multiplexing effectively is discussed in each of the base publications for this supplement.) The SECTVAL macro is provided to calculate sector value using the record number and length specifications of the desired record.

Rotational position sensing support requires 100K bytes in the SVA. The entire 100K requirement can be allocated in the system GETVIS area or 12K can be allocated in the system GETVIS area and the remaining 88K in the resident reenterable program phases area.

Dynamic Link Support for 3350 and 3330 Model 11 Disk Storage

Dynamic link support is provided to enable existing DOS/VS programs that access direct access device types other than the 3350 in native or 3330 Model 11 compatibility mode and the 3330 Model 11 to access the same files on these devices without modifying, reassembling, and relink editing these programs. The inclusion of RPS support in the generated supervisor is required in order to utilized dynamic link support.

When a direct access file is opened, dynamic link support automatically invokes the required 3350 or 3330 Model 11 RPS logic module for the file (as required) if all the requirements for utilizing RPS support (outlined previously) are met. If the dynamic link support requirements cannot be met by a given program, 3350 native and 3330 Model 11 compatibility mode or 3330 Model 11 support can be obtained by link editing the relocatable version of the program (if available) or reassembling (with new device specifications) and link editing the program with the required 3350 or 3330 Model 11 logic module(s).

Channel Program Translation and Page Fixing

In DOS/VS, the channel scheduler supports channel program translation and page fixing for programs operating in virtual mode. These functions are provided only for the I/O device types that are supported by DOS/VS. Channel program translation and page fixing must be performed by the user for unsupported I/O devices. Several macros are provided to enable a problem program to handle these functions.

The channel program translation routine is called by the channel scheduler when required. This routine is reentrant and provides the following support:

- Translation of the virtual storage addresses contained in CCW lists. When the fast CCW translation option is not present in the supervisor, channel program translation is performed for each I/O operation requested by a virtual mode problem program via the EXCP macro without the REAL parameter specified. The translation is performed as part of the EXCP processing that occurs before the I/O request is queued. The CCW list with virtual storage addresses that is indicated in an I/O request is placed in one or more available copy blocks located in an area above the resident supervisor that is reserved for use by the channel program translation routine.

Address translation is performed on the copied CCW list in the copy block(s).

During system generation, the number of copy blocks to be reserved above the supervisor can be specified. The default and minimum specifications depend on the number of partitions defined and whether the fast CCW translation option is included in the supervisor. A maximum of 450 copy blocks can be defined. A copy block is 72 bytes in length. The CCB for the I/O operation is also placed in a copy block and translated. The new translated CCW list is used for the actual I/O operation.

If there are not enough copy blocks defined to contain the CCB, the translated CCW list, and any required IDAL's for a given I/O operation, the task that initiated the I/O request is canceled. If the required number of copy blocks is not available when the EXCP macro is issued, the task that issued the I/O request must wait until enough copy blocks become available. Channel program translation is not performed for I/O requests issued by (1) system tasks, (2) real mode programs, (3) virtual mode programs that use EXCP with the REAL parameter to request I/O operations, or (4) any task when the request is for a console I/O operation and the console buffering option is included in the supervisor.

When the fast CCW translation option is included in the supervisor, an attempt is made to save translated channel programs for reuse so that retranslation is avoided. Fast CCW translation is attempted for all virtual channel programs except those containing non-contiguous CCW lists and those associated with BTAM.

When fast CCW translation is present and an EXCP macro is issued, the channel program translation routine first determines whether a translated version of the channel program has been saved. This is accomplished by inspecting the queue of saved copy blocks with translated channel programs that is maintained for the partition that issued the request. If a saved translated version is found, no translation is performed. Any page frames associated with the saved translated channel program are temporarily fixed if necessary and the saved channel program is used.

When a saved version of the channel program is not found, channel program translation is performed as usual and the translated version is saved. The saved translated channel programs for a partition are released (copy blocks in which they are contained are returned to available status) whenever job step termination occurs.

- Construction of indirect data address lists (IDAL's), when necessary. If the I/O area specified in one or more CCW's in a channel program crosses a virtual storage page boundary or is larger than 2K bytes, the appropriate IDAL's consisting of indirect data address words (IDAW's) are constructed in available copy blocks above the supervisor. (Checking is not performed to determine whether an I/O area that crosses a virtual storage page boundary is actually assigned contiguous page frames.)

- Temporary fixing of the pages that will be referenced during an I/O operation, to prevent the occurrence of page faults during the I/O operation. The channel scheduler issues a TFIX request for all the pages containing I/O areas referenced by the channel program. If the I/O request requires more page frames than can ever be allocated (number of page frames in the page pool less the number of permanently fixed pages is smaller than the required number), the task that issued the request is terminated.

Since the CCB and translated channel program are contained in copy blocks in the fixed supervisor area, fixing of these pages is not required. Any I/O appendages that are to be used must be fixed by the program that issues the EXCP macro before the macro is issued, since the channel scheduler does not perform this function.

- Translation of the real storage address in the channel status word to a virtual storage address at the completion of an I/O operation. In addition, a TFREE request is issued for pages that were temporarily fixed prior to the I/O operation. All the copy blocks associated with the I/O operation except the one containing the CCB are released.

  An indicator is set to cause the task dispatcher to move appropriate parts of the copied CCB to the original CCB and free the copy block. Updating of the original CCB could cause a page fault. Hence, this function is performed by the task dispatcher if the original CCB is not present in real storage at the completion of the I/O operation, since the task dispatcher is permitted to cause a page fault and the I/O interruption handler is not.

In order to enable a virtual mode problem program to handle channel program translation and page fixing for I/O operations, a REAL parameter for the EXCP macro and the REALAD and VIRTAD macros are supported as a system generation option. A problem program operating in virtual mode specifies REAL on an EXCP macro to indicate that the CCW list specified has already been translated, any required IDAL's have been constructed, and all required page fixing has been done. The channel scheduler queues the I/O request and starts the I/O operation without performing these functions. The REAL parameter is ignored if the EXCP macro is issued by a real mode program.

The REALAD macro causes the real storage address associated with the virtual storage address specified to be returned while the VIRTAD causes the virtual storage address associated with the specified real storage address to be returned. In both cases, the page referenced must be permanently fixed. The pages containing areas to be referenced during I/O operations can be fixed and unfixed using the PFIX and PFREE macros.

VIRTUAL STORAGE ACCESS METHOD

General Description

Virtual Storage Access Method (VSAM) is a new component of DOS/VS data management. VSAM provides a file organization and access method for direct access devices that is different from existing DOS file organizations and access methods for direct access devices (SAM, ISAM, and DAM). In a DOS/VS environment, VSAM supports 2314/2319, 3330-series, 3340, 3344, and 3350 direct access devices. Rotational position sensing is supported when the feature is present on the direct access device and RPS support is included in the DOS/VS supervisor.

VSAM uses System/370 instructions and is designed to operate efficiently in a paging environment. Hence, like DOS/VS, VSAM can operate only on System/370 models with dynamic address translation hardware and cannot run on System/360 models. VSAM also requires the inclusion of the GETVIS/FREEVIS macros and the relocating loader in the DOS/VS supervisor.

The VSAM support provided in DOS/VS is compatible with that provided in OS/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/VS 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 DOS/VS 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 file is high.

In addition, the three data organizations supported by VSAM enable it to be used in place of SAM for sequential files and in place of DAM for certain directly organized files. The new structure and features of VSAM make it more suited to data base and online environments than other DOS/VS access methods.

VSAM support consists of the following:

- Access method routines (logic modules) with which the user interfaces to process logical records in VSAM files. Most of these routines are relocatable and reentrant.
- VSAM catalog/space allocation routines that manage direct access volumes and space used by VSAM files and catalogs. VSAM files are cataloged in the new required VSAM master catalog (SYSCAT) or, optionally, a VSAM user catalog.
- The access method services multifunction service program, which provides required VSAM services, such as file 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 relocatable and reentrant.

This discussion describes Release 2 of VSAM. The conditional swapping instructions must be present in a system in which DOS/VS with Release 2 of VSAM is executed.

General Description of VSAM File 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 file 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 file in ascending collating sequence by a key field, which is called the primary key. Records added after the key-sequenced file is created are inserted in primary key sequence and existing logical records are moved when necessary. In VSAM key-sequenced file 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 file 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 file 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 file, 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 file (like a SAM file). Records added to an existing entry-sequenced file are placed at the end of the file 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 file. 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 file is similar in organization to a fixed-length DAM file that is processed by relative record number. Records in a relative record file are in sequence by ascending relative record number (from 1 to N). A relative record file 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 files cannot have a primary or alternate index.

Physical Structure of VSAM Files

The way in which the extents of the logical records of a VSAM key-sequenced, entry-sequenced, or relative record file 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 file that contains logical records is divided into a number of control areas. Each control area contains a number of control intervals that are on contiguous tracks on the direct access device. A control interval is composed of one or more fixed-length physical disk records.

Unlike physical records in an ISAM file, the physical records in a VSAM file 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 minimum of 512 and a maximum of 32,768 (32K) bytes in size and contains an integral number of physical records. If a control interval is less than or equal to 8192 bytes in size, it must be a multiple of 512 bytes. If a control interval is greater than 8192 bytes, it must be a multiple of 2048 bytes in size.

A control interval contains logical records, free space (for key-sequenced files 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 file 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 file can span physical records within a control interval. A logical record in a key-sequenced or entry-sequenced (but not a relative record) file 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 file is to have spanned logical records, this fact must be specified when the file 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 logical record is 32,761 bytes. The maximum size of a spanned logical record is the 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.

Figure 80.30.1 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 file is chosen taking into account the amount of space allocated to the file, index and data control interval size, and buffer space available to the file. The maximum size of a control area on disk is one cylinder, and control area contains an integral number of control intervals. The size of a control interval can be specified by the user and is used as long as it is within the limits defined by VSAM; otherwise, a user-specified control interval size is ignored.

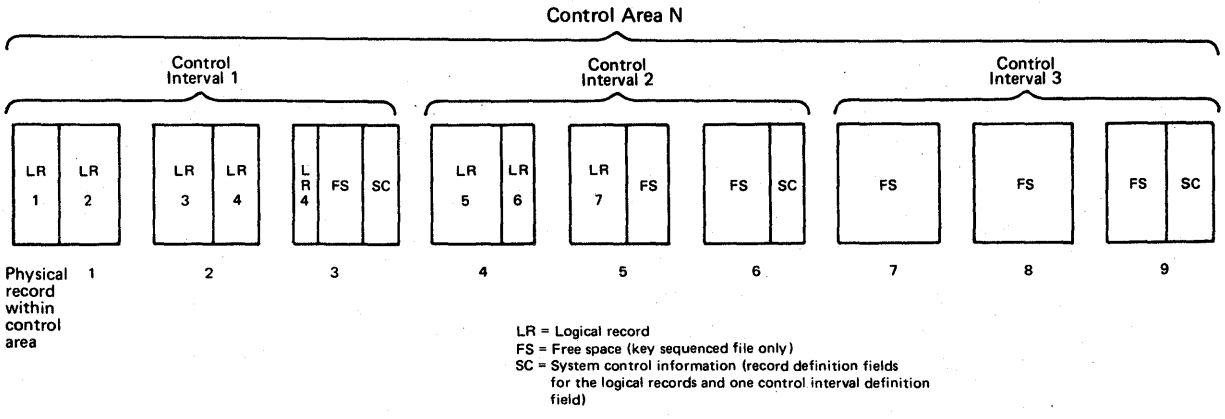


Figure 80.30.1. Organization of a control area for a VSAM file

When a VSAM file is loaded, VSAM does or does not preformat control areas, depending on the attribute specified when the file 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 file 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 file. For an entry-sequenced or relative record file, control information and an

end-of-file indication is placed in each control interval of the control area during preformatting.

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

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 file 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 file.

Like an ISAM file, a VSAM file can be multi-extent and multivolume. Secondary space allocation can be specified when a key-sequenced, entry-sequenced, or relative record file is defined so that the file can be extended when logical records are added, if necessary (this facility is not supported in ISAM). A VSAM file can have a maximum of 123 extents of logical records.

VSAM files can be placed on disk volumes that contain files with other organizations. Space on a volume that is defined for exclusive use by VSAM is called a data space. A VSAM data space can consist of a maximum of 16 extents on a volume that need not be contiguous. The maximum size of a data space is one volume. A volume can contain multiple data spaces. A data space on a volume can contain one or more files and a file can occupy one or more data spaces on one or more volumes.

A direct access space management facility, not supported for other DOS/VS file organizations, is supported for VSAM files. This facility relieves the user of having to keep track of the specific extents allocated to VSAM files. Direct access space on volumes is made available for allocation to VSAM files using the access method services program (DEFINE function). At this time, EXTENT statements are used to indicate the specific tracks on each volume that are to be reserved for VSAM files.

When a VSAM file is actually created, EXTENT statements need specify only the amount of space that is to be allocated to the file. Specific tracks need not be given. The space management program allocates the requested space from extents previously reserved for VSAM files. Thereafter, whenever the VSAM file is processed, EXTENT statements, one per volume in the file, need specify only the symbolic units that are to contain the VSAM file and the volume serial numbers of the volumes containing the file. Specific extents need not be given.

Before a VSAM file 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 file 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 a DOS/VS file utility.

When a VSAM file is defined, it can be allocated space within a previously defined data space or the file 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 file defined with it and the file is called unique. Additional extents cannot be added to a unique file after it is defined.

The REUSABLE attribute can be specified for a VSAM file 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 file that resides on one or more volumes. A key-sequenced or entry-sequenced file with an alternate index or a key-sequenced file 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 file with the REUSABLE parameter specified is opened, it is set to the status of a file 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 file) is performed but the logical records are not erased.

The data in a VSAM file 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 file 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 file. The first record in a file 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 file, 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 file. As a result, VSAM files are device type independent. A VSAM file can be moved from one device type to another and its index file(s) need not be recreated.

The logical records of a VSAM file 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 file is processed by the user on the basis of logical records and VSAM always manages the I/O buffers.

Access to a VSAM file 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 file on a control interval basis. That is, each read or write accesses an entire control interval of data. A file 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).

VSAM Macros

The macros provided to define and process VSAM files 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 DTF macro that is used for other DOS/VS file 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 file that is to be processed by the program. The access control block for a VSAM file must be opened before any processing of the file can occur.

  The ACB specifies the following for a VSAM file: name of the DLBL statement for the file, 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 file (keyed, addressed, and/or control interval, sequential, skip-sequential, and/or direct, etc.), the number of requests that can be outstanding concurrently for the file using this ACB, and the address and length of an error message area to be used by the OPEN, CLOSE, and TCLOSE macros.

- 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 file. 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 file is reached (EODAD exit), a logical error occurs (LERAD exit), an uncorrectable physical I/O error occurs (SYNAD exit), to perform a journaling operation (JRNAD exit), or to overlap VSAM I/O operations with user processing (EXCPAD). Each exit routine can be marked active or inactive. An exit routine that is inactive is not entered when its associated condition occurs.

  The journaling exit must be marked active before the file with which it is to be used is opened; this exit cannot be marked inactive during processing of the file. The other exits can be activated and deactivated (via the MODCB macro) during processing of the files with which they are used. The exits to be used during the processing of a given VSAM file 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 file; each time data is shifted within a control interval or moved to another control interval (key-sequenced files only); and each time a segment of a spanned record is read or written.

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

When the EXCPAD exit is not specified and a GET or PUT macro is issued, control is not returned to the instuction after the GET/PUT macro until all processing of the request, including any required I/O operations, is completed.

When the EXCPAD exit is specified, the user-written EXCPAD routine is given control whenever VSAM starts an I/O operation as a result of a request macro. The exit routine executes concurrently with the VSAM I/O operation. When the exit routine completes its execution, it must return control to VSAM. VSAM then returns control to the instruction after the request macro as soon as the I/O operation completes successfully.

The EXCPAD exit enables the user to overlap problem program processing with VSAM I/O operations. It is analogous in intent to the asynchronous processing facility that is supported in OS/VS VSAM.

- 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, and ENDREQ) must specify the address of a request parameter list to indicate the processing to be performed. The same RPL can be specified in more than one type of request macro.

  An RPL macro specifies the following: the ACB of the file 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, key greater than, RBA, or relative record number); the type of processing for this request, such as keyed or addressed, sequential or direct, forward or backward, etc.

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

- GENCB (generate a control block or list). The GENCB macro can be used to generate an ACB, EXLST, or RPL during program execution instead of program assembly. The GENCB macro can be used to eliminate changing these control macros and reassembling VSAM problem 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 (entries cannot be added or deleted), 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 (show or display the catalog). The SHOWCAT macro is used to retrieve information from a VSAM catalog about any unopened VSAM file and place it in a user-specified area.

The following request macros are used to process VSAM files:

- OPEN - A VSAM file must be opened before it can be processed by other request macros. The OPEN macro provides the same types of processing functions for VSAM files as for other types of files. OPEN verifies that the required volumes of the VSAM file are mounted, 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, causes loading into the virtual partition of the required VSAM routines for the processing specified (unless the VSAM routines are resident in the SVA), and verifies that the password given is correct.

  Both sequential and direct processing can be performed on a VSAM file using one OPEN macro and one ACB. Closing and reopening of the file to switch modes, as is required for an ISAM file, 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 file during its creation or insert a new record in an existing file. 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 file. The record is physically removed from the file. 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 file 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.

- ENDREQ - This macro is used to free VSAM from keeping track of a position in a file. VSAM can maintain knowledge of the same number of positions as the number of requests that can be outstanding concurrently (specified in the ACB or GENCB macro).

- CLOSE - The CLOSE macro provides the same types of processing functions for VSAM files as for other types of files. 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 file, if necessary (if the location of the end-of-file indicator has changed, for example), and update the statistics for the file in its catalog entry. The access method control block(s) for the file (such as the ACB's) are restored to what they were before the file was opened and virtual storage that was obtained during OPEN processing for additional VSAM control blocks and VSAM routines, if any, is released.

  Once a VSAM file has been closed, it must be reopened before any additional processing can be performed on it. A TCLOSE (temporary CLOSE) macro can be issued to cause VSAM to complete any outstanding I/O operations, update the catalog if necessary, and write any required statistics. TCLOSE also formats the last used control interval in the file if the file is being loaded or extended and the SPEED option was specified. Processing can continue after a TCLOSE macro without the issuing of an OPEN macro.

Figure 80.30.2 illustrates the relationships among the most frequently used control macros and the request macros.

Note that several parts of a VSAM file 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 file. 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 file at one time. The number of concurrent requests that can be outstanding for a file is specified in the ACB but is extended by VSAM during processing if necessary.

A shared resource facility is provided in VSAM that enables buffers, I/O control blocks, and channel programs to be shared among several VSAM files within the same partition. Without the use of this facility, buffers and control blocks are allocated to each VSAM file when it is opened and freed when the file is closed. Thus, while a VSAM file is opened, the buffers and control blocks cannot be shared.

The BLDVRP, DLVRP, and WRTBFR macros are provided to support the shared resource facility. BLDVRP is used to define a VSAM resource pool (of buffers, control blocks, and channel programs) that is to be shared by VSAM files that specify the shared resource facility in their ACB (via the LSR option). The DLVRP macro deletes a resource pool.

The WRTBFR macro is provided for I/O buffer management. It is used to cause the writing of buffers whose writing has been deferred. When the shared resource facility is used, the writing of a buffer specified in direct PUT requests can be delayed (that is, a direct PUT request does not cause the affected buffer to be written immediately). Deferral of writing can save I/O operations if subsequent requests are for records in buffers whose writing has been delayed. Buffer writing can be delayed until a WRTBFR macro is issued to cause writing, a buffer from the shared pool is required for a GET request, or a file that is sharing the pool is closed.

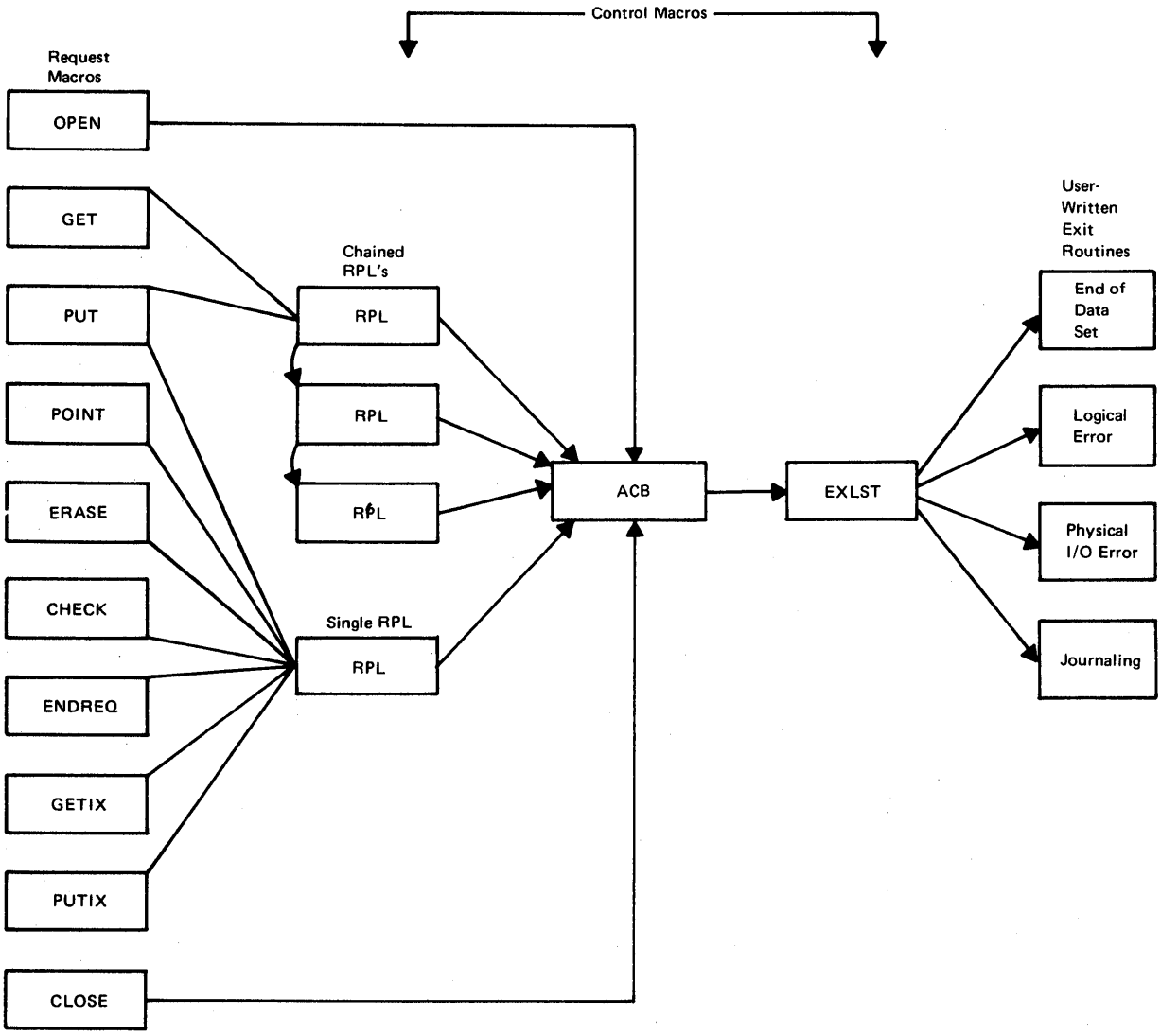


Figure 80.30.2. Relationships among VSAM control and request macros

The shared buffer facility is designed to optimize the use of storage resources for and speed up the direct processing of VSAM files (1) whose activity is unpredictable or (2) that are used in an environment in which a transaction may require the processing of several files. It is, therefore, useful for data base/data communication environments.

Concurrently outstanding requests for a file can be any combination of sequential and direct processing requests. Each outstanding request can specify one RPL or a list of RPL's (chained RPL's). When chained RPL's are specified in a request, control is not returned to the user until all requests in the list have been processed.

Key-Sequenced File Organization and Processing

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

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

All extents of logical records (the data component extents) in a key-sequenced file must reside on direct access volumes of the same type. Extents of logical records for a given key-sequenced file do not have to be contiguous across volumes as is required for DOS ISAM files. The primary index file and any alternate index file cluster, however, can be placed on a device type that is different from that of the logical record file. The primary index file and alternate index files 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 file is created, the range of primary key values that are to be allocated to each volume of the data component file can be user-specified. This cannot be done for an ISAM file. Unlike ISAM file volumes, all volumes of the data component of a key-sequenced file 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 file of a key-sequenced file 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 file is divided into control areas and control intervals in order to distribute free space throughout the file for the addition of logical records. When a key-sequenced file 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 file 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 file 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 file. Hence, less primary index maintenance is required for a key-sequenced VSAM file set than for an ISAM file.

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 file was defined. A new control area is established and the contents of approximately half of the control intervals in the full control area are moved to the new control area. The new logical record is then inserted in the appropriate control area in primary key sequence.

The time required to sequentially retrieve records is only slightly affected by control area splitting. Since the amount of space allocated to the file is affected by control area splitting, the number of split control areas in a key-sequenced file should be a factor that is considered when determining whether or not to reorganize the file.

Logical records can be physically deleted from a key-sequenced file, using the ERASE macro, and the length of a logical record in a variable-length key-sequenced file 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 file, 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 file. This design makes VSAM key-sequenced organization more suited than ISAM to an online environment.

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

The primary index for a key-sequenced VSAM file also has a totally different logical structure from that used for an ISAM index. A key-sequenced primary index file 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 file 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 file, it is a nondense index level.

The structure of the primary index for a VSAM key-sequenced file is shown in Figure 80.30.3. Where a key is specified, it refers to a primary key.

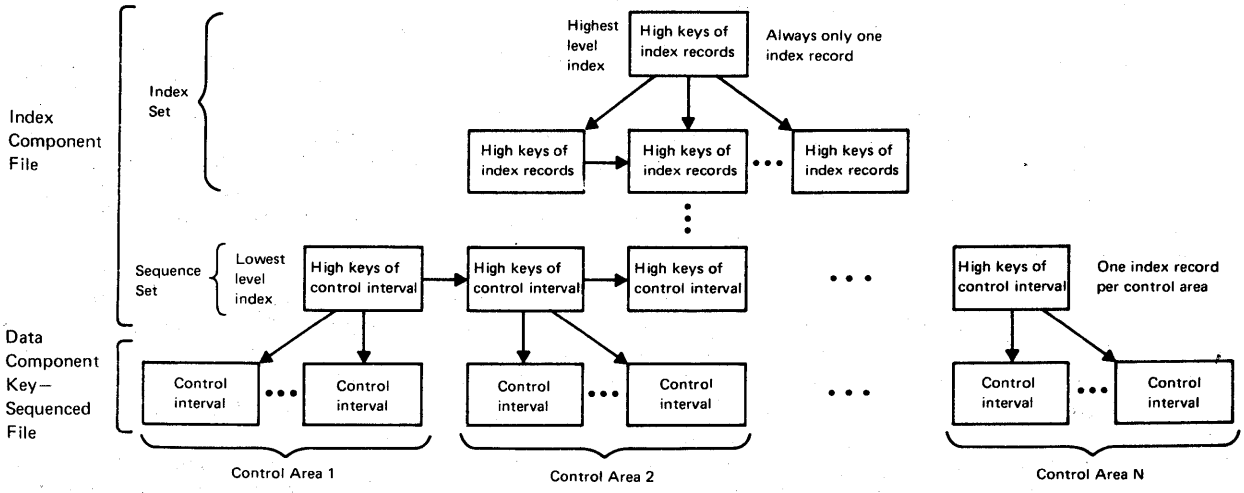


Figure 80.30.3. Structure of the primary index for a VSAM key-sequenced file

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 file. 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 file 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 file 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 file 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 file (index set levels) and stored with the logical records in the data component file. 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 file).

When the sequence set index level is stored within the data component file, 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 file 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 the VSAM index structure over the 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 file 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 technique; 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 file. Primary index records are stored in control intervals as are the logical records in the data component of a key-sequenced file. However, there is only one physical index record written in a control interval, control intervals are not grouped into control areas, and no free space is left within a control interval between a logical record (index entry) and its control information. The maximum size of a control interval in an index component is 2048 bytes for a 2314/2319 and 4096 bytes for a 3330-series, 3340, 3344, and 3350 in 3330 Model 1 compatibility mode. Control interval size can be 512, 1024, 2048, or 4069 bytes only for an index 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 files. Optionally, one or more alternate indexes can be built for an existing key-sequenced file. An alternate index cannot be built for another alternate index file or for a key-sequenced file with the REUSABLE option assigned.

Alternate indexes enable the logical records of a key-sequenced file 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 files that are sequenced by different fields for different applications. The support of multiple indexes for a given file 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 file, 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 file 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 file, as shown in Figure 80.30.3. 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 file. It contains one variable-length logical record for each unique alternate key value. For a key-sequenced file, 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 file that contains the alternate key. If the alternate key appears in more than one logical record in the base file, 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 path is the means by which a base key-sequenced file 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 file. When a given alternate index is to be used to process a key-sequenced file, the path associated with that alternate index must be specified in an OPEN macro. This causes both the key-sequenced file 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 file 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 files) has its own name and can be processed as a key-sequenced file independently from its associated base key-sequenced file. 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 file 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 file 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 file has been defined, and it can be loaded only after the base file 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 file, 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 file).

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 upgrade set of alternate indexes for a given key-sequenced file. An alternate index can be part of the upgrade set for a key-sequenced file even though it is not a member of a path for

the key-sequenced file. The maximum number of alternate indexes that can be part of the upgrade set for a given key-sequenced file is 125.

Whenever a key-sequenced file is opened for keyed or addressed output operations, VSAM automatically opens for output all the alternate indexes in the upgrade set for the base file, unless the NOUPDATE job control parameter is specified for the key-sequenced file. 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 file (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 file. In addition, VSAM does not update any alternate indexes for a base key-sequenced file 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 file has been done and does not make any synchronization checks between a key-sequenced file and its alternate indexes during OPEN processing.

During processing of a base key-sequenced file via an alternate index, an error can occur while processing the key-sequenced file, 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 file was involved in the error.

Key-sequenced file processing. The records in a key-sequenced file 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 key-sequenced files that contain nonspanned or spanned records. All volumes of a key-sequenced file 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 file 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 files.

Addressed processing cannot be used with key-sequenced files 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 file 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 file by RBA need not be modified if direct access device type is changed.

Keyed sequential processing of a key-sequenced file is like sequential processing of an ISAM file. It is used to load a key-sequenced file and to retrieve, update, delete, and add logical records to an existing key-sequenced file. 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 files 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 file 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 file).

  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 file 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 file is like direct processing of an ISAM file. 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.

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 file 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 file 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 file 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 file 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. 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, 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 file when control interval processing is utilized.

Processing of the primary or alternate index file for a key-sequenced file. The primary index of a key-sequenced file can be processed using GET and PUT macros. The index component (file) must be opened alone. The primary index can then be processed like an entry-sequenced file. It can be accessed using addressed or control interval processing. The alternate indexes for a key-sequenced file can be processed in the same ways as can a key-sequenced file.

Processing summary. Table 80.30.1 summarizes the primary types of processing supported for key-sequenced VSAM files (control interval processing is not included in the table).

Table 80.30.1. Types of processing supported for VSAM key-sequenced files. (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. Addressed processing via an alternate index path is not permitted.)

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 record size changes. No RBA's required.	Retrieval and update only. No record size changes. 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 File Organization and Processing

The logical records in an entry-sequenced file are ordered by the sequence in which they are presented for entry into the file. Free space cannot be left within the control intervals and control areas of an entry-sequenced file when it is defined. Additions to an existing entry-sequenced file are placed in any available space left at the end of the file. Extents can be added to an existing entry-sequenced file if secondary allocation was specified when the file was defined. Although an entry-sequenced file 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 file must be placed on volumes of the same direct access type. However, an entry-sequenced file 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 files. 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 file is also used when the size of an existing record in a variable-length entry-sequenced file 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 file.

The only time a change is made in the RBA of a logical record in an entry-sequenced file 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 file. An entry-sequenced file can also be moved from one direct access device type to another, and programs need not be modified because the RBA's of the logical records do not change.

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

The only way an alternate index for an entry-sequenced file differs from one for a key-sequenced file 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 file that contain that alternate key. The RBA's obtained from the alternate index are used to directly retrieve the required logical records from the entry-sequenced file.

If an alternate index for an entry-sequenced file 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 file (either during file creation or when making additions), VSAM returns the RBA of the record. These RBA's 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 file 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).

Entry-sequenced file processing. Addressed sequential, addressed direct, and control interval processing are supported for entry-sequenced files 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 file 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 file can be retrieved and updated (but new control intervals cannot be added) using sequential or direct access and a new entry-sequenced file can be created using sequential control interval processing. GET, PUT, POINT, 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 file, 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 file.

An entry-sequenced file can also be used like a DAM file. 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 file). The entry-sequenced file must be preformatted with dummy records before the logical records are placed in the file.

Table 80.30.2 summarizes the primary types of processing supported for VSAM entry-sequenced files. Access by control interval is not included in the table.

Table 80.30.2. Types of processing supported for VSAM entry-sequenced files. (An entry indicates whether the access type is supported, an alternate key or RBA is required, and alternate keys or RBA's 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	RBA's 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	RBA's not in sequence	No Keys required	Keys in ascending sequence	Keys not in sequence
Create and add after end of file (PUT without update)	Forward processing only. No RBA required.				
Delete	Records marked deleted by user identification. No RBA's required.	Records marked deleted by user identification. RBA's 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 File Organization and Processing

A relative record file consists of a number of fixed-length slots, 1 to N, where N is the maximum number of logical records that the file 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 file contains the same number of slots. The record length specified for a relative record file 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 file is defined. All extents of the file must reside on the same device type. An index cannot be created for a relative record file; however, a relative record file is still considered to be a cluster.

Keyed sequential, keyed skip-sequential, keyed direct, and control interval processing are supported for relative record files. The

relative record number is used as a key for keyed processing. A relative record file can be created using keyed sequential, skip-sequential, or direct processing.

Keyed sequential processing of a relative record file is like keyed sequential processing of a key-sequenced file. 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 file. 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 passed 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 file can be processed by control interval using addressed sequential or addressed direct (by RBA) access. The control intervals in an existing entry-sequenced file can be retrieved and updated (but new control intervals cannot be added) using sequential or direct access, and a new relative record file 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 file cannot be extended using control interval processing.

Table 80.30.3 summarizes the primary types of processing supported for a relative record file. Control interval access is not included in the table.

Table 80.30.3. Types of processing supported for VSAM relative record files. (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

All VSAM files (index as well as those with logical records) must be cataloged in the VSAM master catalog, which is always on logical unit SYSCAT, or in a VSAM user catalog. Information required to process a VSAM file, such as its location and physical characteristics, is contained in a VSAM catalog. This cataloging facility is similar to the data set cataloging facility provided for all OS and OS/VS data sets and is a new data management facility for DOS/VS users. Non-VSAM files can also be defined in a VSAM catalog.

There must be one VSAM master catalog for a DOS/VS operating system that contains VSAM. Optionally, one or more VSAM user catalogs can be defined. Each catalog is an individual file and can be assigned any programmer logical unit. Each VSAM user catalog has an entry in the VSAM master catalog. Each VSAM file is cataloged in the VSAM master catalog or a user catalog, but not both. All VSAM files on the same volume must be cataloged in the same VSAM catalog. Duplicate file names in the same VSAM catalog are not permitted and a given file cannot 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 file to be portable from one system to another without having to use the access method services program to process VSAM master catalogs.

A VSAM user catalog can be made available to a job by specifying IJSYSCUC as the file name in the DLBL statement for the catalog. The job catalog is then used for processing all VSAM files specified by the job unless it is overridden by the CATALOG parameter of the access method services program or the CAT parameter of the DLBL statement.

The following information is recorded in the catalog record for a VSAM file:

- Device type and volume serial numbers of volumes containing the file

- Location of the extents of the file and secondary space allocation, if any
- Attributes of the file, 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 file, 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 files and their index(es): the index and data components of a key-sequenced file; 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 file 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 files. Therefore, a volume containing a VSAM file need not be mounted in order to determine whether it contains available space. VSAM catalog/DADSM routines are used to process the catalog and to allocate space on VSAM catalog and file volumes.

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

A VSAM catalog is logically structured as a key-sequenced file 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 file. Keyed, addressed, and control interval processing are permitted.

A recovery facility is available for VSAM catalogs that enables VSAM files and their catalog entries to be recovered in the event that a VSAM catalog cannot be read for any reason. This recovery facility cannot be used for VSAM catalogs that contain non-VSAM files. 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 file described by the catalog is recorded both in the catalog and a recovery area on the first volume of the file 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. 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 files. The EXPORTRA command accesses the catalog recovery area for the specified VSAM files in order to open them and then produce a copy of the specified VSAM files. The IMPORTRA command is then used to reestablish the copied VSAM file and its catalog entry 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 these 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.

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 files. Facilities to convert ISAM and SAM files 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 DOS/VS, the access method services program is called from a problem program or invoked by executing the IDCAMS program, which can be executed as a job step with commands supplied via the input stream.

The access method services program is used to:

- Define and allocate direct access space for all VSAM files, alternate indexes, and VSAM catalogs. The DEFINE function must be used to describe a VSAM file or catalog before any data is placed in the file or the catalog. The DEFINE function is also used to define paths and data spaces and to catalog non-VSAM files in a VSAM catalog.

- Create, reorganize, and back up VSAM files. Input to the REPRO function can be an ISAM, SAM, or VSAM (key-sequenced, entry-sequenced, or relative record) file. The output can be a VSAM (key-sequenced, entry-sequenced, or relative record) or SAM file. 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 file to VSAM key-sequenced format, initially create a VSAM file from sequenced records, merge new logical records into an existing VSAM file, and reorganize a VSAM file.

- 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 file or a VSAM key-sequenced or entry-sequenced file. 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 file. Multiple alternate indexes for the same file can be built at the same time.

- Print all or the specified range of logical records of a SAM, ISAM, or VSAM file 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 file can be modified by altering the catalog entry for the file.

- Delete files, data spaces, alternate 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 files.

- Perform processing required to make a VSAM file portable from one System/370 to another if a VSAM user catalog is not available. The EXPORT command is used to copy a VSAM file (any organization) to a tape or disk volume as a sequentially organized file. Required information is extracted from the catalog entry for the file and written on the transporting volume as well. The IMPORT command is used to create a VSAM file and its catalog entry from the file created by an EXPORT command.

  Exportation can be temporary or permanent. If it is temporary, a copy of the file is retained in the exporting system. Thus, a copy of the file exists in both the exporting and importing systems. If exportation is permanent, the catalog entry and space for the file are deleted from the exporting system so that the file 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 files. 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 file (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 file. If the catalog information does not agree with the actual end-of-file or end-of-key range in the file, the catalog information is updated.

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

Since VSAM files must be cataloged, and the access method services program must be used to define and allocate space for VSAM files, a

minimum number of job control parameters for DD statements are used by VSAM. The only changes to job control required for VSAM support are in the DLBL statement. Note that VOL, XTENT, and DLAB label statements are not supported for a VSAM file. EXTENT and DLBL statements must be used.

The DLBL statement has a new code parameter to identify VSAM files and CAT and BUFSP parameters. Optionally, the CAT parameter can be specified to indicate the VSAM user catalog containing the VSAM file to be accessed. The CAT parameter specification overrides the job catalog defined for the job, if any. Optionally, the BUFSP parameter can be specified to indicate the amount of buffer space to be used in processing the VSAM file. This value overrides the buffer space specified in the catalog entry and/or ACB for the file.

Password Protection

A full 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 file and the ACB does not specify a password, the operator must supply the correct password in order for the file to be opened. Up to seven retries can be made as specified when the file is defined.

If the ACB does not specify the correct password or the operator does not supply the correct password in the number of attempts permitted, the VSAM file is not opened. The user can also specify that the operator is not to supply the password. In this case, it must be supplied correctly via the ACB in order for the file to be opened.

Multiple levels of protection are provided:

- Full access, which allows access to a file, its index(es), and its catalog entry. Any operation (read, add, update, delete) can be performed on the file and its catalog entry. The master password of the base key-sequenced file 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 file is permitted (definition of new objects and alteration of existing objects), but an entry cannot be deleted.

- Read access, which allows access to a file for read operations only. Read access to the catalog entries of the file is permitted also. No writing is allowed.

A password can be defined for a given VSAM file 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 file, the password given when the file is opened establishes the level of protection to be in effect for this OPEN.

Authorization to process a VSAM file can be supplemented by a user-written security authorization routine. 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 file. This record can supply data to the user-written security authorization routine during its processing.

File Sharing

A VSAM file can be accessed concurrently by two or more subtasks within the same partition and two or more job steps (partitions). Both types of sharing can be used for a VSAM file at the same time. The type of file sharing permitted for two or more partitions is controlled by using the SHAREOPTIONS parameter of the DEFINE command when the VSAM file is defined. Cross-system sharing of VSAM files is not supported.

The following types of cross-partition-sharing options are supported:

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

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

- The file can be opened by any number of users for both read and write operations. VSAM provides write integrity by using the DOS/VS track hold facility and read integrity only for records read for update (SHAREOPTIONS 4). When this option is chosen, each task can open only one ACB for a file at a time and a given ACB can be opened by only one task at a time.

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 files to be used to process key-sequenced VSAM files without modification of ISAM macros. The VSAM files can be newly created or those that have been converted from ISAM format to VSAM key-sequenced format. The ISAM interface routine requires the presence of the track hold feature in the DOS/VS supervisor.

The ISAM interface routine permits VSAM key-sequenced files to be processed by both ISAM programs and VSAM programs. This allows existing ISAM application programs to be used as well as additional applications that take advantage of new VSAM facilities to process the same VSAM files. The ISAM interface routine for DOS/VS VSAM can be used in Assembler, ANS COBOL, PL/I, and RPG II programs. The high-level language translators that support VSAM directly, without use of the ISAM interface, are the DOS PL/I Optimizing Compiler and Libraries, DOS/VS COBOL, and Full ANS COBOL program products.

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. The only ISAM facility that the ISAM interface routine does not support is record retrieval by record ID (CCHHR). In addition, no device-dependent data is returned when the ERREXT parameter is specified in the DTF. Similarly, with a few exceptions, if VSAM facilities that are not supported by ISAM are to be used, an existing ISAM program must be modified to define a VSAM file 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 ISAM job control statements for the ISAM file must be changed to VSAM job control statements as appropriate. The ISAM interface routine and the access method services program simplify the amount of effort required to replace ISAM file organization with VSAM organization within an installation.

Virtual Storage Requirements

DOS/VS VSAM uses GETVIS and FREEVIS macros. Hence, a problem program that uses VSAM must always execute in virtual mode. VSAM programs must also specify the SIZE parameter on the EXEC statement to make a virtual storage pool above the problem program area available for VSAM use. This GETVIS area is used to contain the VSAM logic module, VSAM I/O areas, the ISAM interface routine, if required, and VSAM control blocks other than access control blocks (ACB's), request parameter lists (RPL's), and the exit lists (EXLST's).

The problem program area contains only VSAM ACB's, RPL's, EXLST's, and the expansions of VSAM macros. If the problem program uses the GENCB macro to generate VSAM control blocks, the ACB's, RPL's, and EXLST's will reside in the partition GETVIS area instead of the problem program area. VSAM uses the channel program translation and page fixing facilities of the channel scheduler extension.

A minimum of 302K plus buffer and control block requirements are needed within a virtual partition for VSAM routines when the access method services program is not used. If this program is used, a virtual partition must contain up to 450K bytes (depending on the functions to be used) for exclusive VSAM use. VSAM routines are dynamically loaded into the GETVIS area of a virtual partition when a VSAM file is opened (unless VSAM routines are made resident in the SVA).

The VSAM routines that are relocatable and reentrant can be made resident in the SVA so they can be shared by concurrently executing problem programs. Making VSAM routines resident in the SVA also saves the time that is required to dynamically load them into a partition each time they are required. Most of the VSAM logic modules, some VSAM catalog/space management routines, and the ISAM interface routine can be made resident in the SVA. No access method services routines can be made resident in the SVA. The control statements required to make these VSAM routines SVA-resident are contained in the IBM-supplied cataloged procedures VSAMSVA and VSAMRPS. The SVA-eligible VSAM phases specified in VSAMSVA require 302K of virtual storage in the SVA. The system GETVIS area in the SVA is not used by VSAM.

When VSAM routines are made resident in the SVA, additional virtual storage for control blocks and buffers must be made available in each partition in which VSAM is to execute. This amount is 30K bytes plus the following minimum for each VSAM file: 3K bytes plus 2 times data control interval size plus the index control interval size. If VSAM routines are not made resident in the SVA, 302K bytes plus the buffer and control block requirements must be available in each virtual partition that will use VSAM.

Note that VSAM has no logic module generation macros. When VSAM is included in a DOS/VS system, standard VSAM modules are placed in the system core image library during system generation. The capability of assembling or link-editing VSAM modules with a problem program does not exist.

Summary

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

VSAM provides new features:

- Three data organizations are supported.
- Variable- as well as fixed-length logical records are supported.
- Files 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 files.
- Additions and index entries are blocked, and disk space requirements are therefore reduced.
- Direct access space management for VSAM files relieves the user of this funtion. Key-sequenced VSAM files can be allocated to specific volumes by key range.
- Secondary space allocation is supported so that an existing file can be extended whenever additions are made (not just when a load function is performed).
- Free space for additions can be allocated at more frequent intervals throughout the allocated extents when a key-sequenced file is created.
- A record deletion facility is supported by the access method.
- Free space reclamation capabilities are provided. This factor can eliminate or significantly reduce the frequency of key-sequenced file 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.
- Multiple levels of password protection are provided and user-written security protection routines are supported.
- Disk volumes containing VSAM data sets/files 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 file.
- 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 file 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 80.30.4 compares the features of VSAM and ISAM as supported in DOS/VS.

Table 80.30.4. Comparison table of VSAM and ISAM facilities for DOS/VS

Characteristic	VSAM - DOS/VS	ISAM - DOS/VS
1. Supporting DOS environments	DOS/VS	DOS Version 3, DOS Version 4, and DOS/VS
2. Direct access devices supported	2314/2319, 3330-series (all models), 3340, 3344, and 3350 (in native and 3330 compatibility modes)	2311, 2321, 2314/2319, 3330-series (Models 1 and 2), 3340, 3344, and 3350 (in 3330 Model 1 compatibility mode)
a. RPS supported	Yes	Only in a DOS/VS environment
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 files. The key-sequenced file contains logical records, and, optionally, distributed free space for additions and the sequence set index level. 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 file. The prime area contains logical records, the track index, and, optionally, overflow tracks in each cylinder for additions. A separate additions area can exist also. The cylinder and master index levels are a separate extent.
b. Entry-sequenced	Yes Records are sequenced by the order in which they are placed in the file. Records are added to the end of an existing file. 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 file	Yes	Yes
a. Secondary space allocation indicated at creation	Yes	No The space originally specified can be extended only when the LOAD function is used to add records after the last existing record.
b. Specification of key range by volume during file creation	Yes	No
c. Volumes of the same device type required	Yes for logical record extents. The primary index file and any alternate index files 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.
d. 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.	Yes

Table 80.30.4. Comparison table of VSAM and ISAM facilities for DOS/VS (continued)

Characteristic	VSAM - DOS/VS	ISAM - DOS/VS
e. Free space available within the logical record area	Yes (for key-sequenced files only) within control intervals and control areas. Free space is distributed within the tracks of a cylinder.	Yes, optionally, at the end of each prime cylinder. Free space cannot be left on tracks within the prime cylinders.
f. File is direct access device-type independent	Yes RBA pointers are used in the control interval and in the index.	No Record address ID (CCHHR) is used in index pointers.
5. Key-sequenced organization file 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. Orginal records and additions are blocked.	Variable length is not supported. Fixed blocked or unblocked record formats are used for prime records. Records in an overflow area are always unblocked.
b. Key field is written on disk	No Records are written in count and data format.	Yes Records are written in count, key, and data format.
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	One 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. One master index level is optional.
b. Nondense index	Yes	Yes
c. Key field written	No Index records are written in count and data disk record format.	Yes Index records are written in count, key, and data disk record format.
d. Index records are blocked	Yes	No
e. Index record size	Fixed length and determined by system.	Data field is always 10 bytes. Key field is key size.
f. Keys are compressed	Yes Both front and rear compression are performed to eliminate redundant characters in adjacent keys in the index.	No Full key is always written.
g. Index record replicated on track to reduce rotational delay	Yes, as an option	No

Table 80.30.4. Comparison table of VSAM and ISAM facilities for DOS/VS (continued)

Characteristic	VSAM - DOS/VS	ISAM - DOS/VS
h. Sequence set index level placed adjacent to logical records	Optional If chosen, sequence set index records are replicated at the beginning of each control interval area.	Standard Track index is always on the first track(s) of prime cylinders.
i. Index resident in virtual storage	Standard As many index records as will fit in the user-specified buffer can be resident, up to a maximum of all index set records.	Optional A portion or all of the cylinder index can be made resident. Residence of the master index is not supported.
j. Multiple indexes for the same key-sequenced or entry-sequenced file	Yes	No
7. Types of processing supported for key-sequenced files		
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 level of the primary index is used.	Yes Each logical record is presented in key sequence. The track index is used.
b. Skip sequential retrieval and update (by keys specified in 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 presenting RBA's in sequence	Positioning via a SETL macro using record ID (CCHHR) is supported. Record must be retrieved sequentially after positioning.
d. Sequential updating by sequenced keys without retrieving records	No	Yes
e. Direct retrieval and update by generic key, equal key, or key-greater-than the specified key	Yes	Yes for equal key. Generic key and key greater than specified key can be used in a SETL macro for positioning. The record must be retrieved separately using sequential mode.
f. Direct retrieval and update by record address	Yes, via RBA	Yes, via record ID (CCHHR)
g. Additions by direct processing	Yes	Yes
h. Additions by mass insertion using sequential processing and key sequenced additions	Yes	No
i. Additions by skip-sequential processing with keys specified in sequence	Yes	No
j. Multiple-request processing supported within a single program or a program and its subtasks	Yes with one ACB	Yes with multiple DTF's

Table 80.30.4. Comparison table of VSAM and ISAM facilities for DOS/VS (continued)

Characteristic	VSAM - DOS/VS	ISAM - DOS/VS
k. Concurrent sequential and direct processing of the same file with a single OPEN	Yes	No The file must be closed and reopened to change modes. Alternatively, two DTF's, one for sequential and one for direct processing, can be used.
l. Deletions physically removed	Yes Records are shifted and free space is reclaimed.	ISAM does not recognize deleted records. The user can mark records deleted. Space can be reclaimed if an update operation retrieves the deleted record and writes a new one of the exact same size in its place.
m. Logical records can be lengthened or shortened	Yes, and space is reclaimed for a shortened record.	Variable-length records are not supported.
n. Write check after a write	Optional	Optional
o. Locate and move mode processing	Locate mode for read-only operations and move mode are supported.	Yes
p. 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	No	Yes
9. Password protection	Yes Levels supported for the user are: • Master access - allows access to the file, its index files, 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 file and its indexes for retrieval, updating, deletions, and additions. Limited modification of catalog entries for the file is permitted but an entry cannot be deleted. • Read access - allows retrieval of data records (no writing of any kind).	Yes Two levels of protection are provided. If the correct password is presented, the file can be opened for read only or for read and write processing.
a. User-written authorization routines supported	Yes	No
10. File sharing		
a. Within a partition	Yes, concurrent updating at the track level prevented with the track hold option	Same as VSAM
b. Across partitions	Yes, as above	Same as VSAM
11. File cataloging	Required The VSAM master catalog or a VSAM user catalog must be used.	Cataloging is not supported in DOS/VS except for VSAM files

Table 80.30.4. Comparison table of VSAM and ISAM facilities for DOS/VS (continued)

Characteristic	VSAM - DOS/VS	ISAM - DOS/VS
12. Languages supporting VSAM	Assembler DOS/VS COBOL Release 1 program product Full ANS COBOL Subset ANS COBOL (via ISAM interface routine) PL/I Optimizing Compiler and Libraries Release 4 program product RPG II (via ISAM interface routine) program product	Assembler COBOL program product PL/I program product RPG program product
13. VSAM file direct input to sort/merge	Yes (to DOS/VS Sort/Merge 5746-SM1 program product)	No
14. Utility program functions	Access method services program can perform the following: • Define and delete direct access space for a VSAM file, catalog, alternate index, etc. • List, alter, or delete an existing VSAM catalog entry • Create new and reorganize existing VSAM files • Copy a VSAM, ISAM, or SAM disk file to a new SAM file or into an existing VSAM file • List some or all of the records in a VSAM, ISAM, or SAM file • Perform functions required to make a VSAM file portable from one system to another • Verify and reestablish, if necessary, the end-of-file marker in one VSAM file. • Build alternate indexes	No specific utility program is provided for ISAM files.
15. Resource sharing by files in the same partition	Yes	No

80:35 RECOVERY MANAGEMENT AND DEBUGGING AIDS

MCAR, CCH, RMSR, AND OLTEP

MCAR, CCH, and RMSR are standard features that are automatically included in any DOS/VS supervisor that is generated to support a Model 135, 138, 145, 148, 155 II, or 158. OLTEP is automatically included in supervisors for these models unless it is specifically omitted. MCAR, CCH, and RMSR are optional for any DOS/VS supervisor that supports a Model 115 or 125. OLTEP is included by default for these two models unless it is specifically deleted at system generation.

MCAR, CCH, and RMSR provide the same support in DOS/VS as in DOS Version 4. In addition, MCAR is extended to mark a page frame unavailable for allocation when an uncorrectable storage error occurs that cannot be validated by the DRAP (dynamic reallocation of a partition) routine. The malfunctioning bit in the PFTE for the page frame is turned on and the PFTE is taken out of the selection pool. The operator is notified when a page frame is so deleted and the task involved is abnormally terminated.

When the unusable page frame is located within a real storage area that can be allocated to a currently defined real partition, the DRAP routine reduces the size of the real partition, just as is done in a DOS Version 4 environment (ending real partition address is lowered to eliminate the unusable page frame from the real partition).

CCH is modified in DOS/VS to terminate system operations when a retry fails to correct a channel error involving an I/O operation on the paging device.

OLTEP is modified as required to operate in a DOS/VS environment. In DOS/VS, OLTEP must execute in the background real partition, which must be a minimum of 16K (20K when RETAIN is active). The operator should ensure that the lowest dispatching priority is assigned to the background partition when OLTEP is executing in it. This minimizes the impact of OLTEP execution on system performance.

OLTEP can be scheduled by the POWER/VS program in a DOS/VS environment. Modifications are made to OLTEP to prevent it from performing test operations on unit record I/O devices being used by POWER/VS for card reading, card punching, and printing. This prevents the OLT from receiving erroneous results.

DEBUGGING AIDS

The debugging aids provided in DOS Version 4 are modified as required to operate correctly in DOS/VS. The standalone and abnormal termination dumps, the PDUMP and DUMP macros, and the DUMP, ALTER, and DISPLAY commands are supported in DOS/VS. They support the same functions in both DOS versions, as do the PDAIDS traces (FETCH/LOAD Trace, I/O Trace, SVC Trace, QTAM Trace, and Transient Dump). PDAIDS in DOS/VS also include a VTAM trace and a VTAM buffer pool trace.

Note that the DUMP macro causes only supervisor control blocks (instead of the entire supervisor area) to be dumped if the DUMP=PART option was specified in the STDJC generation macro or an OPTION PARTDUMP job control statement was included for the job.

The JDUMP macro is also provided in DOS/VS and, like the DUMP macro, can be used to cause dumping of the supervisor area, general registers, and the real or virtual partition that issued the JDUMP macro. But while DUMP causes canceling only of the job step for which the dump was

taken when multitasking is not in effect in the partition, JDUMP causes the entire job to be canceled. When the DUMP macro is issued by a subtask, only that subtask is canceled. When the DUMP macro is issued by the main task, the entire job step is canceled.

The standalone dump can be used to display the contents of real storage only. Therefore, a SYSVIS Dump system utility program (also called the Page Data Set Dump) is provided in DOS/VS that displays the contents of the virtual address area (which is contained in the page data set). All of the virtual address area, a specific virtual partition, or selected virtual storage pages can be dumped. The SYSVIS Dump can be executed after an abnormal system termination occurs and the standalone dump has been executed. If this utility is executed immediately after the re-IPL, as per operating instructions, no paging activity occurs between the abnormal termination and dumping of the page data set.

The output can be written to the SYSLST device, which can be a printer, tape, or disk device. Alternatively, when the abnormal termination occurs, SYSVIS Dump output can be directed to a SYS001 device, which can be a tape or disk unit. When this latter option is chosen, a high-speed dump of the user-indicated virtual storage area occurs and normal system operation can continue sooner. The dump output on SYS001 can then be printed later during normal processing using the SYSVIS Dump program.

DOS/VS supports a high-speed standalone dump program, not supported by DOS Version 4, that dumps the contents of virtual and real storage to tape or disk. The DOSVSDMP program is provided to generate the standalone dump program and print the dump output. DOSVSDMP is supplied as a self-relocating program in the system relocatable library. It must be link edited to a core image library before it is used the first time.

When a high-speed dump is to be taken, the DOSVSDMP program must be executed under DOS/VS control. This program creates a standalone dump program on the user-specified tape or disk volume. Execution of the standalone dump is initiated by IPLing the device that contains the resulting dump program.

The standalone dump program writes a dump to the tape or disk volume on which it resides. The dump contains the contents of all virtual storage in page number sequence and the contents of all real storage in page frame sequence. Pages not residing in real storage are obtained from the page data set. The system is placed in the wait state at the completion of the dump.

The DOS/VS system can be IPLed after the high-speed dump is taken. When the dump is to be printed, the DOSVSDMP program must be executed under DOS/VS control. The dump can be printed with or without formatting.

DOS/VS also provides significant new tracing and dumping facilities via the system debugging aids (SDAIDS) routines. This facility is standard in DOS/VS. It can be used to debug both control and problem programs and to collect statistics about page faults and page request handling operations.

Using program event recording, program interruption hardware, and an interface to the page manager, SDAIDS routines can monitor the occurrence of up to eight different system-defined events, as specified by the operator. When an event that is being monitored occurs, SDAIDS receives control to collect and print status information on a printer, after which normal processing continues.

The printer used by SDAIDS need not be dedicated to printing SDAIDS output, but the printer cannot be attached to a channel that has burst mode I/O devices operating on it while SDAIDS is operative. SDAIDS and any one of the PDAIDS traces can operate concurrently.

SDAIDS routines can monitor the occurrence of the following events, which are designated as elementary or dedicated:

- Successful execution of any type of branch instruction that is located within a specified area of virtual storage (elementary event). Program event recording (PER) hardware is enabled to detect this event.
- Fetching of an instruction from a specified area of virtual storage (elementary event). PER hardware is enabled to detect this event.
- Alteration of storage within a specified virtual storage area (elementary event). PER hardware is enabled to detect this event.
- Alteration of the contents of the general registers specified (elementary event). PER hardware is enabled to detect this event.
- Page translation exception (page fault) caused by any instruction within a specified area of virtual storage (elementary event). This event is detected by programmed checking that is done by SDAIDS after a page translation exception interruption occurs.
- Program check interruption that occurs as a result of any program interruption except a segment translation exception, page translation exception, special operation, monitor event, or program event (program interruption codes X'01' to X'0F', X'10', and X'12' are handled). This is a dedicated event and is monitored for all executing tasks. It cannot be limited to one or more specific partitions.
- Enqueuing of a request for page frame assignment because a page fault has occurred or a PFIX, TFIX, or GETREAL request was issued (dedicated event). This event is monitored for all executing tasks and its occurrence is signalled by page management. This event can be used to obtain information about the sequence in which a given task is requesting pages.
- Assignment of a page frame to a virtual storage request (dedicated event). This event is monitored for all executing tasks and its occurrence is signaled by page management after a page request has been serviced. Essentially, this event provides a page fault handling trace by providing information about the sequence in which page-ins are serviced.

The one or more events to be monitored are indicated by the operator via the system console device when SDAIDS routines are initialized or at a later time. The amount of status data collected for all elementary events varies depending on the output class specified by the operator (1 to 8 and Q). The contents of the following can be recorded when an elementary event occurs; program old PSW, time of day clock, general registers, fixed locations in lowest real storage, entire supervisor area, communication regions, control registers, segment table, page tables, the page frame table, the real storage allocated to the real address area, and a nondestroying dump of a user-specified portion of virtual storage.

Only pages within the virtual storage addresses specified, which can encompass all of the defined virtual storage, that are present in real storage are dumped (the page data set is not read) and system operation continues normally after a nondestroying dump is taken. Various

combinations of the items listed can be dumped by specifying the appropriate output class (1 to 8). The instruction or the virtual storage page that caused the elementary event is also identified in the output.

The output that is collected when a class from 1 to 8 is specified is dumped to the printer immediately after an event that is being monitored occurs. When output class Q is specified, only certain data is saved when an event occurs. This data is stored in an SDAIDS buffer area that is provided in real storage when class Q is specified. The saved data is printed as a block of output each time a program check event occurs or as soon as the SDAIDS buffer is filled. The status data saved for the Q output class is the program old PSW, the time of day, the contents of general registers 0, 1, 2, 13, 14, and 15, and program event recording mask settings (contents of control registers 9, 10, and 11).

The data recorded for a dedicated event is fixed by event type. When a program check occurs, the contents of the program old PSW, time of day clock, entire supervisor area, general registers, control registers, and the instruction that caused the interruption are saved. When a request for a page frame is enqueued, the type of request (PFIX, TFIX, page fault, or GETREAL) is indicated as are the associated task identification and the virtual storage page and its storage protect key. When a page frame is assigned, the address of the page frame is indicated in addition to all the data supplied when a request for page frame assignment is enqueued.

The operator can also specify the stop-on-event or the stop-on-address option. If either of these options is in effect, processing stops after output is printed or saved when an event occurs (a wait state PSW is loaded). The operator can then do any combination of the following: use hands-on debugging aids, cause a nondestroying dump of all of real storage to be printed, or initiate the monitoring of one or more additional SDAIDS events. Processing continues after the operator presses the external interrupt key. The status of real storage and registers are not disturbed by the dumping procedure and system operation continues normally after the printout.

The difference between the stop-on-event and the stop-on-address options is that specification of the latter option causes the system to stop after any of the events being traced occurs. When the stop-on-event option is used, the system stops only when a specified elementary or dedicated event occurs. System operation continues after other events occur and the required data is printed or saved in real storage.

Initiation of SDAIDS is requested by entering // EXEC SDAID from the SYSLOG or SYSRDR device any time after IPL is completed. Once initiated, SDAIDS remains operative until the attention routine command ENDSD or the job control statement // EXEC ENDSD is entered by the operator. Initialization of SDAIDS can be performed in a virtual or real partition. If a real partition is used, it must be a minimum of 12K.

SDAIDS routines operate in an area of real storage called the SD area, which must be a minimum of 6K. Since the page frames in the SD area are removed from the selection pool, these page frames cannot be reassigned and, thus, effectively are permanently fixed. The operator can specify the size of the SD area (up to a maximum of 999K) when SDAIDS is initialized in order to allocate additional buffer space for storing the data collected for events.

When SDAIDS is initiated, available page frames from the page pool with addresses above the the last real partition defined in the real address area are allocated to the SD area. The available page frames with the highest addresses are chosen. The operator is notified if

enough page frames are not available and the SDAIDS initiation process is terminated.

Once SDAIDS has an SD area allocated, the initialization process begins. Using the console, the operator enters parameters (events to be monitored and options to be effective) in response to the listing of keywords. Once all parameters have been specified, information is typed on the console to indicate the parameters in effect and to enable the operator to change the specifications at a later time during system operation.

Once SDAIDS initialization is completed, the partition used for initialization is released and SDAIDS routines make no further use of DOS/VS services. SDAIDS receives CPU control as soon as any of the events being monitored is recognized. It operates without DAT mode specified and with the CPU disabled for external and I/O interruptions. (SDAIDS cannot be used to debug time-dependent programs because I/O interruptions are disabled.)

After SDAIDS has collected the required status information and printed the output requested, control is returned to the point at which the event occurred so that operation of SDAIDS is transparent to executing programs.

80:40 LANGUAGE TRANSLATORS, SERVICE PROGRAMS, AND EMULATORS

DOS/VS ASSEMBLER

The DOS/VS Assembler is the only language translator that is a standard component of DOS/VS. The DOS/VS Assembler provides the same functions as DOS Assemblers D and F. It also offers a few extensions and improved diagnostics. The DOS/VS Assembler accepts all the source statements accepted by Assemblers D and F, and is a subset version of the System Assembler provided for OS/VS1 and OS/VS2. The DOS/VS Assembler is upward-compatible with the OS/VS Assembler.

The DOS/VS Assembler supports all System/370 instructions except INSERT PSW KEY, SET PSW KEY FROM ADDRESS, CLEAR I/O, and multiprocessing instructions (SET PREFIX, STORE PREFIX, SIGNAL PROCESSOR, and STORE CPU ADDRESS). It is the only DOS Assembler that supports the following System/370 instructions:

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

The DOS/VS Assembler can operate in virtual or real mode. If a real partition is used, it must be a minimum of 20K bytes. When a larger real partition is made available, the additional storage is used to improve performance. Only direct access devices are supported for intermediate work storage during assemblies.

The performance of the DOS/VS Assembler can be up to thirty percent better than that of Assembler D. The increase in performance is primarily the result of a change made to the source statement library. The copy sublibrary of the source statement library contains source code that can be included in source programs using COPY statements, just as in DOS Version 4. The macro sublibrary in DOS/VS, however, contains IBM-supplied Assembler Language macro definitions in preedited (partially processed) format. Use of a preedited format for macros speeds up the assembly process for programs that use these macros. User-written macros can be converted to preedited format using the EDECK option of the Assembler.

The RLD and NORLD options can be specified to cause printing or suppression of printing of the Relocation Dictionary at the end of an assembly.

POWER/VS

Functions and General Operation

POWER/VS (Priority Output Writers, Execution Processors, and Input Readers/Virtual Storage) is an optional program that is distributed as part of DOS/VS. The Type III, Class A Maintenance POWER II program that can be used with DOS Versions 3 and 4 is still available but cannot be used with DOS/VS. The version of POWER provided for use with DOS/VS Releases 28, 29, and 30 (called DOS/VS POWER) cannot be used with DOS/VS Releases 31 and up. POWER/VS executes with DOS/VS Releases 30 and up.

The POWER/VS program operates in a partition under control of the DOS/VS supervisor to provide services for up to four other partitions, each of which is identified as a POWER/VS-controlled partition when it is started. When the Advanced Function-DOS/VS program product is

installed, POWER/VS will support up to six partitions. POWER/VS is designed to improve system throughput in a multiprogramming environment by providing job scheduling by priority within class and automatic data transcription to and from unit record devices (card readers, punches, and printers) overlapped with job step execution.

Reading input streams from unit record devices and transcribing the input data they contain to intermediate disk storage and transcribing the printer and punch output from completed job steps from intermediate disk storage to printers and punches, both concurrent with job execution, are called spooling.

When POWER/VS is used, card reading for the jobs to be executed in POWER/VS controlled partitions is overlapped with card punching and printing operations for completed jobs/job steps and with job step execution. Job steps can execute faster as they are no longer limited by peripheral I/O operations on relatively slower speed unit record devices. Another advantage of POWER/VS is that it enables one card reader, one punch, and one printer to be shared by multiple partitions. Without POWER/VS, each partition must have its own system input device and system output devices (or disk extents).

Problem programs that currently handle unit record I/O operations directly need not be modified in order to interface with the POWER/VS program. POWER/VS execution processor tasks receive CPU control for all I/O requests from the partitions POWER/VS schedules that are for unit record devices that are to be spooled and performs the required functions in a manner that is transparent to the programs that issue the I/O requests. Whenever an I/O request (SVC 0) is issued, the DOS/VS supervisor inspects the specified device address to determine whether the request is for one of the spooled devices. If so, CPU control is given to POWER/VS, which in turn passes control to the appropriate execution processor task.

Two basic versions of POWER/VS can be generated: one that handles job scheduling and input and output spooling operations for local unit record devices only (a reader/writer system without remote job entry support) and one that supports job scheduling and input and output spooling operations for both local and remote unit record devices (a reader/writer system with remote job entry support).

The remote job entry facility included in POWER/VS can support only binary synchronous terminals (RJE,BSC support), only system network architecture (SNA) terminals using synchronous data link control (RJE, SNA), or both BSC and SNA terminals (RJE,SNA with BSC support included). POWER/VS interfaces with VTAM to provide RJE support of SNA terminals.

A writer-only version, as supported in DOS/VS POWER, is not supported in POWER/VS. However, a reader/writer POWER/VS system can control one or more writer-only partitions. A writer-only partition exists when no spooled card reader is specified for a partition when it is started to operate under POWER/VS control. A writer-only partition is useful for DOS/VS jobs that perform stacker selection of input cards, since this is not possible under POWER/VS, or that use a card reader feature not supported by POWER/VS, such as column binary. Stacker selection requests for card input from spooled card readers issued by programs executing in POWER/VS controlled partitions are ignored by POWER/VS.

Whether POWER/VS is to be used must be specified at DOS/VS system generation. When POWER/VS is to be used, support of three partitions, page handling overlap, PFIX/PFREE macros, VIRTAD/REALAD macros, and the EXCP macro with the REAL parameter are automatically included in the generated DOS/VS supervisor if they have not been otherwise requested. An NPARTS specification of at least 2 is required. If the job accounting facilitiy of POWER/VS is to be used, job accounting support

must also be included in the DOS/VS supervisor. A system with a minimum of 96K of real storage is required for the operation of DOS/VS with POWER/VS.

POWER/VS is distributed as a set of generation macros and self-relocating executable program phases. The macros are used to define a POWER/VS system that is tailored to the functions desired by an installation. When these macros are assembled and link edited, they create a phase that causes those POWER/VS program phases required to support the features and options desired to be loaded into the POWER/VS partition during POWER/VS initialization.

If the POWER/VS system defined by the IBM-supplied generated POWER/VS phase for a reader/writer system is not suitable for an installation, the desired POWER/VS system must be defined using the POWER/VS generation macros, assembled, and link edited to the system core image library. Since only a system description phase must be assembled and link edited, significantly less time (a few minutes) is required for the POWER/VS generation procedure, than for the DOS/VS POWER generation procedure, which involves the link editing of DOS/VS POWER object modules. In addition, the number of generation macros required for POWER/VS is reduced (14 DOS/VS POWER generation macros have no POWER/VS counterpart) and a separate generation for RJE support modules is not required for RJE under POWER/VS.

One (and only one) partition must be dedicated to the POWER/VS program when it is operating. POWER/VS can execute in any partition, including the background, as long as the partition POWER/VS is assigned has a higher dispatching priority than all the partitions it is to service. However, if POWER/VS operates in the background partition, programs that can execute only in the background partition (such as OLTEP) cannot be executed while POWER/VS is active. POWER/VS operates only in virtual mode.

POWER/VS uses the PFIX and PFREE macros and, therefore, the real partition corresponding to the virtual partition in which POWER/VS executes must be allocated. POWER/VS cannot operate as a subtask of a main task.

Any POWER/VS system can handle job scheduling and local and remote spooling operations involving only BSC terminals for four partitions (or 6 when the Advanced Functions-DOS/VS program program product is installed). A POWER/VS system using RJE,SNA support can control up to three (or 5) partitions, since VTAM must operate in a partition. The number of partitions to be handled by POWER/VS is not specified at generation time as it is for other versions of POWER. The main task and any subtasks in a POWER/VS-controlled partition can use the spooling capability POWER/VS provides.

Once POWER/VS is initiated in a partition by the operator, partitions POWER/VS is to control can be started and stopped by the operator at any time until POWER/VS is terminated. POWER/VS can schedule both virtual and real partitions.

If the PRTY command is issued to change the dispatching priority of the POWER/VS partition and it violates the priority rule, the command is ignored and the operator is informed. In addition, a DOS/VS CANCEL command is rejected if it specifies the partition in which POWER/VS is operating. The POWER/VS partition will not be deactivated by the partition deactivation routine of DOS/VS page management.

POWER/VS is started and stopped in a partition and its operation is controlled using a set of POWER/VS operator commands. These commands are accepted by the DOS/VS attention routine. Once started, POWER/VS can read one or more input streams of jobs from local card readers

and/or 3540 diskette devices. When remote job entry support is included in a POWER/VS system, input streams can also be read from one or more remote terminal units. The POWER/VS routines that read input streams are called read tasks.

An input stream contains jobs that are to be executed in the partitions POWER/VS controls. An input stream can also contain card and/or diskette input for the job steps defined. Jobs are defined using DOS/VS job control statements only or a combination of DOS/VS and POWER/VS job control statements.

A POWER/VS job entry control language (JECL) is provided that is used to request certain POWER/VS processing facilities as well as to aid in the definition of jobs. The JECL statements supported by POWER/VS are JOB, CTL, RDR, LST (or PRT), PUN, SLI, DATA, EOJ, /* (end of job step), and /& (end of job). The DATA, SLI, /*, and /& statements are used only with the source library inclusion facility.

POWER/VS writes one input work queue of the jobs contained in all the local and remote job streams it reads to intermediate direct access storage and transcribes job control statements (JECL and DOS/VS) and input data in the input streams to spool input files on intermediate direct access storage. Note that in this discussion of POWER/VS, spool input file is used to logically describe the card or diskette input in an input stream for a job step that POWER/VS writes to intermediate direct access storage. Similarly, spool output file or spooled printer/punch file is used to logically describe the printer/punch output from a job step that POWER/VS writes to intermediate direct access storage for later transcription to a printer or punch.

Job classes, scheduling, and queues. Using the input work queue, POWER/VS schedules the execution of jobs in the partitions it controls by job priority within job class. Each partition POWER/VS controls can be assigned one partition-dependent job class. The partition-dependent classes are 0 to 4, which correspond to partitions BG to F4, respectively. When the Advanced Functions-DOS/VS program product is installed, partition-dependent numbers 0 to 6 are used for partitions BG to F6.

Each POWER/VS-controlled partition can also be assigned from one to four partition-independent job classes. The partition-independent job classes are A to Z. The same partition-independent job class can be assigned to more than one partition.

Job classes, both partition-dependent and partition-independent, are assigned to a partition via the POWER/VS PSTART command. The classes a POWER/VS controlled partition is assigned determine the jobs that can be executed in the partition.

If no job class is assigned to a partition, it is assigned only its partition-dependent class by default. If one or more partition-independent job classes are assigned to a partition, the partition-dependent class for the partition is not also assigned automatically (it must be specified in the PSTART command in order for it to be assigned).

Each job submitted to POWER/VS also has a job class and job priority assigned (by the user or by default) that indicates the partition(s) in which it can be executed. If a partition-independent class (A to Z) is assigned, the job is executed in the first POWER/VS-controlled partition with the same partition-independent class assigned that becomes available. If a job has a partition-dependent job class assigned, the job can be executed only in the associated partition. The job priority, 0 to 9 low to high priority, determines the sequence of scheduling jobs with the same job class assigned. Jobs with the same job class and priority assigned are scheduled on a first-in, first-out basis.

The order in which partition-independent classes are specified when a partition is started under POWER/VS control determines the high to low priority of the classes within the partition. This means, for example, if a partition is assigned classes C, B, and A in that sequence, all queued jobs with class C assigned will be scheduled to execute in the partition before any queued jobs with class B assigned. Queued class A jobs will not be scheduled for the partition until all queued class B jobs are scheduled.

If a partition is not assigned a partition-independent class, only jobs assigned the partition-dependent class of the partition can be executed in it. The classes assigned to a partition and their priority within the partition can be changed by stopping the partition via a POWER/VS PSTOP command and starting it again with a different set and/or sequence of classes specified.

Partition-independent job classes can be assigned to jobs on the basis of job characteristics, such as virtual storage requirements, I/O device requirements, I/O-oriented processing, CPU-oriented processing, high or low priority for execution, etc., to enable jobs with the same characteristics to be scheduled to execute in the same partition. However, since a partition can have more than one partition-independent class assigned and the same class can be assigned to more than one partition, job classes can be assigned to make certain jobs eligible for execution in more than one partition.

Further flexibility is provided by the fact that up until the time a queued job is selected for execution, its input class can be altered by the operator. The scheduling flexibility provided by POWER/VS via input classes can result in more balanced partition usage, particularly in an environment in which workload characteristics can vary.

During the execution of job steps in partitions POWER/VS is scheduling, POWER/VS execution processor (execution read, execution list, and execution punch) tasks handle read and write requests for the unit record devices that are to be spooled for the partition. (The spooled devices are specified when the partition is started under POWER/VS control.) That is, when a card read is issued by an executing job step to a spooled card reader, POWER/VS supplies the card or diskette record from the spool input file on intermediate disk storage it created when reading the input in the job stream. Similarly, when a punch or print request is issued to a spooled device, POWER/VS writes the record to a spool output file on its intermediate disk storage. Spooled output can be written to magnetic tape instead of disk.

The intermediate direct access storage used by POWER/VS is dedicated to the POWER/VS system. This disk storage contains the POWER/VS queue file, data file, and optionally account file. The queue file contains three queues, which are the reader queue (also called the input queue), list queue, and punch queue. The list and punch queues are also called the output queue. The queue file also contains a master record that summarizes the contents of the three queues. The master record is used to warm start POWER/VS.

The reader queue contains one read queue entry for each POWER/VS job read by POWER/VS. A read queue entry is used in the scheduling of the POWER/VS job it represents and usually is maintained in the reader queue until the job has executed successfully. A read queue entry can also be deleted by the operator before its associated job has been scheduled or kept in the queue after its associated job has been executed.

The list queue usually contains a single list queue entry for each spooled printer file written to disk by POWER/VS for the steps of the local and remote, if any, jobs it schedules. In effect, each list queue entry represents one output job to be processed by POWER/VS. A list

queue entry usually is maintained in the list queue until the spool file it describes has been transcribed to a printer by POWER/VS or it is deleted by the operator. However, a list queue entry can be kept in the list queue after transcription of its associated spool file.

The punch queue usually contains a single punch queue entry for each spooled punch file written to disk by POWER/VS for the steps of the local and remote, if any, jobs it schedules. In effect, each punch queue entry represents one output job to be processed by POWER/VS. A punch queue entry usually is maintained in the punch queue until the spool file it describes has been transcribed to a punch by POWER/VS or it is deleted by the operator. However, a punch queue entry can be kept in the punch queue after the spool file it describes has been transcribed.

More than one list/punch queue entry for a spool output file is created when the file is segmented (as discussed later). In this case, a list/punch queue entry is created for each segment.

The job control and card/diskette input contained in the input streams read by POWER/VS are written in spool input files in the data file. These spool files are pointed to by the read queue entries. The data file also contains the spooled punch and spooled printer files that POWER/VS writes for executing job steps. These spool files are pointed to by punch queue and list queue entries, respectively. When the optional accounting facility of POWER/VS is used, the accounting data collected is written in the account file.

The spooled printer and punch files written by POWER/VS can be assigned an output class, A to Z, using POWER/VS JECL (LST and PUN) statements in the input stream. If a LST/PCH statement is not provided for a spool output file or does not specify a class, the output class specified in the PSTART command for the partition from which the spool file was created is assigned. Class A is assigned by default if a class is not specified for a spool output file. The priority assigned to spool output files can be user-specified or by default is the same as was assigned to the POWER/VS job for scheduling purposes when the job was read. The output class and priority of a queued spool file can be changed by the operator any time before the file is selected for transcription.

The reader queue logically contains 31 or 33 subqueues, one for each partition-dependent class (0 to 4 or 6) and one for each partition-independent class (A to Z). Similarly, the list queue and punch queue each logically contain 26 subqueues, one for each possible output class (A to Z). The reader queue, list queue, and punch queue entries each are maintained in priority sequence within class. Entries in the same queue with the same class and priority are queued on a first-in, first-out basis within their class. Note that when the operator issues a PALTER command to alter any characteristic of a queued entry, except priority, the entry is requeued at the end of the class queue in which it belongs.

The POWER/VS routines that transcribe spooled printer and punch files to printers and punches are called list tasks and punch tasks, respectively. Each list or punch task can be assigned from one to four output classes (A to Z) when it is started. Class A is assigned by default if a class is not specified when a list or punch task is started. A list/punch task can transcribe only those printer/punch files that have an output class assigned that the task handles. A queued spooled printer/punch file is transcribed by the first list/punch task that becomes available and has assigned the same output class.

The sequence in which the output classes are specified for a list or punch task determines the priority of handling the classes. For

example, if classes C and B are assigned to a list task in that sequence, the list task will transcribe any queued spooled printer files with class C assigned before transcribing any spooled printer files with class B. All list tasks are scheduled from the single list queue and all punch tasks are scheduled from the single punch queue. Figure 80.40.1 shows the general operation of a POWER/VS system.

Disposition attributes. The entries in the reader, list, and punch queues also have a disposition attribute assigned that determines when POWER/VS schedules the input and output jobs defined by the entries. The disposition attribute is assigned via a JECL statement in the input stream or by default. The JECL JOB statement is used for input jobs. The JECL LST and PCH statements are used for spooled output files. The disposition attribute of an input or output queue entry can be changed by the operator any time before the entry is selected for processing.

Disposition attributes are the following:

- D - dispatch and delete. When the D attribute is specified in a read queue entry, the job it describes is automatically scheduled to execute by priority within job class in the first available eligible partition as previously described. When the job completes its execution, its read queue entry is deleted. For a punch or list queue entry, the D attribute causes the associated spooled punch or printer file to be transcribed automatically to a local punch or printer or sent to a remote terminal according to priority within job class. After transcription is completed, the punch or list queue entry is deleted.

- H - hold job. For a read queue entry, the H attribute causes the associated job to remain in the reader queue until the operator changes the disposition attribute to D via a PALTER command or issues a PRELEASE command for the job. Similarly, the H attribute in a punch or list queue entry, prevents transcription of the associated spool output file until the operator issues a PALTER or PRELEASE command for the file. This attribute enables the processing of input and output jobs to be deferred.

- K - dispatch and keep. When specified in a read queue entry, the associated job is automatically scheduled for execution according to priority and job class. However, when the job completes, the read queue entry is not deleted from the reader queue and its disposition is changed to L. Similarly, a spooled printer or punch file with a disposition of K in its output queue entry will be transcribed automatically according to priority and class but after the file has been processed, its entry remains in the output queue and its disposition is changed to L. This attribute enables a job or spool file to be retained after it is processed so that it can be processed again at a later time without resubmission or reexecution.

- L - leave in queue. The L attribute in a read queue entry prevents the job from being scheduled for execution until the operator changes its disposition to D or K via a PALTER command or issues a PRELEASE command for the job. A spooled printer or punch file with the L attribute in its queue entry is not transcribed until the operator issues a PALTER or PRELEASE command for the file.

The D, H, K, and L disposition attributes apply only to spool output files that are written to disk. The T disposition attribute can be specified for a spooled printer or punch file to cause POWER/VS to write it to tape instead of disk. In this case, the spool output file is transcribed to a printer or punch by the list or punch task started for the tape unit that contains the spool output file. Spool output files from RJE jobs POWER/VS schedules cannot be written to tape.

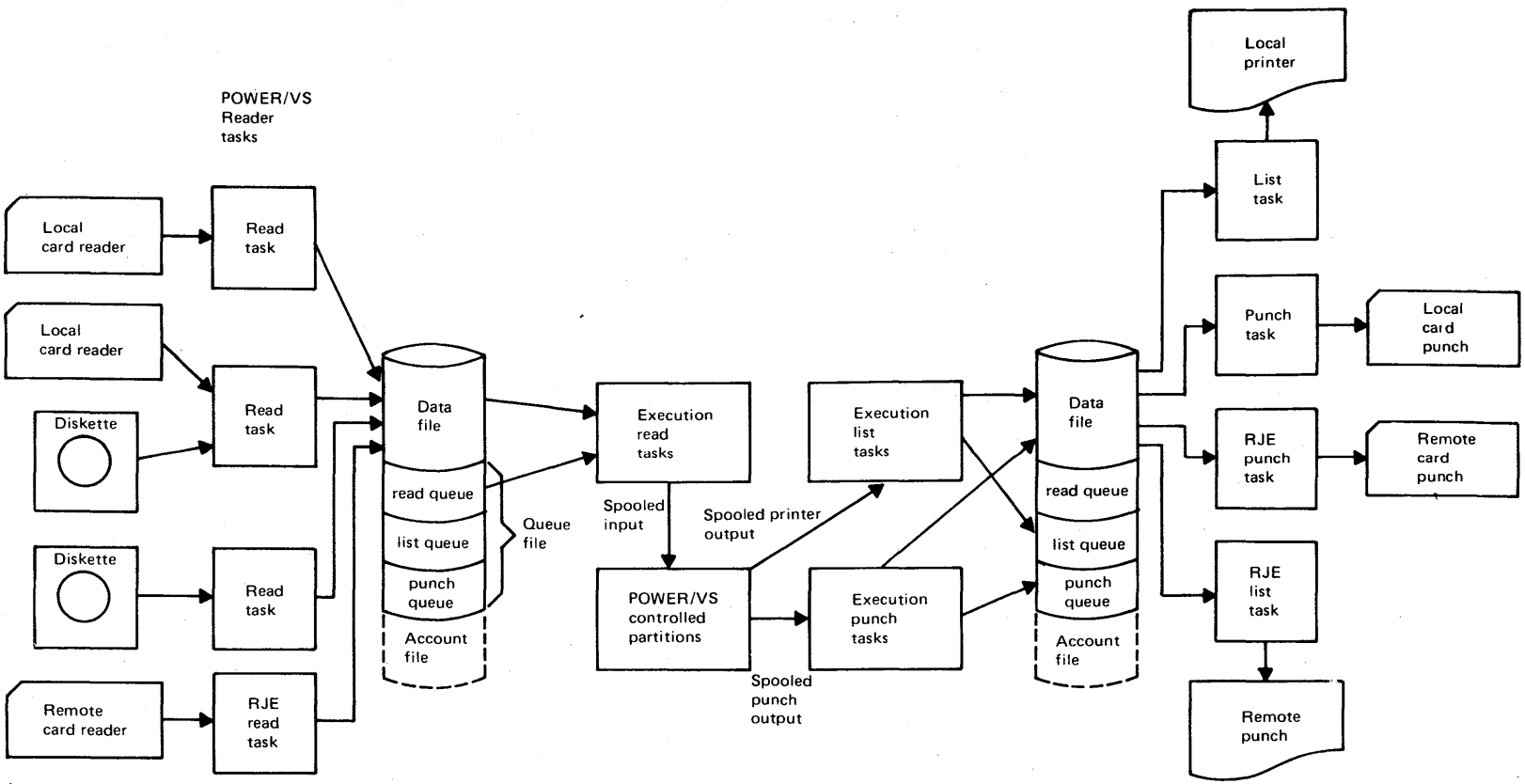


Figure 80.40.1. General operation of the POWER/VS system

The I disposition attribute can be specified only for spooled punch files written to disk. It causes the spool output file to be placed in the reader queue as an input job instead of transcribed to a punch device. This attribute should be used only for spooled punch output that is in executable format and contains all required DOS/VS and POWER/VS job control.

The N disposition attribute can be specified to indicate that printer or punch output that is written to a spooled device is not to be spooled but is to be printed or punched directly from the executing job step. That is, if a spooled output file is written to printer device 00E (for example), printer device 00E is specified as a spooled device for the partition, and the spool output file has the N attribute, POWER/VS will attempt to write the printer output directly to real printer 00E. If this printer is not available (currently assigned to another partition, for example), POWER/VS ignores the N attribute and writes the printer output to a spool file on disk. The N attribute cannot be specified for the spooled output from remotely submitted jobs.

Job Entry Control Language

All POWER/VS JECL statements contain the characters * $$ in the first four positions and can also contain comments. Continuation statements are supported. Parameters are specified in positional or keyword format. Some parameters can be specified in either format. Others (such as those not supported by DOS/VS POWER) can be specified only in keyword format. The positional format of POWER/VS JECL statements is the same as the positional format of the equivalent DOS/VS POWER JECL statements for compatibility purposes (although the POWER/VS parameter may have more possible operands).

Since JECL statements contain an * in the first position, they are treated as comments statements by the DOS/VS job control program when it reads an input stream containing JECL statements. Thus, a job stream with JECL statements can be executed under POWER/VS control or not without modification. This * also enables POWER/VS JECL statements to be included in jobs that are to execute in writer-only partitions, since such jobs are read by the job control program.

The job entry control language of POWER/VS supplements the DOS/VS job control language. While DOS/VS job control statements must be used as usual for the jobs that execute under POWER/VS control, the use of POWER/VS JECL is optional except when certain POWER/VS facilities are to be used.

POWER/VS JECL statements must be used to specify certain attributes for a POWER/VS job if the default or POWER/VS generation values are not to be used, if any spooled output is to be written to tape instead of intermediate disk storage, and to request utilization of the source library inclusion facility.

Specifically, JECL statements are used to indicate the following POWER/VS job attributes: input class for jobs, output class for spooled printer and punch files, priority for scheduling input jobs and transcribing spooled printer and punch files, input disposition for input jobs, output disposition for spooled printer and punch files, output limitation for the spooled printer and punch output of a job, number of copies, forms identification, segmentation of spooled output files, the FCB and/or UCB image or carriage control tape format to be used for printing spooled printer files, the printer setup to be used for a 3800 Printing Subsystem, tape address if intermediate disk storage is not to be used, and the destination of spooled printer and punch files created by a remotely submitted job.

When POWER/VS is used, there is a distinction between POWER/VS jobs and DOS/VS jobs. The JECL JOB and EOJ statements are used to delimit POWER/VS jobs. DOS/VS jobs are delimited by the DOS/VS JOB and /& job control statements as usual. A POWER/VS read task queues POWER/VS jobs in the reader queue rather than DOS/VS jobs and it is POWER/VS jobs that are scheduled on a priority within class basis.

A POWER/VS job can contain multiple DOS/VS jobs (JECL JOB statement followed by two or more sets of DOS/VS JOB and /& statements followed by a JECL EOJ statement). In this case, a single read queue entry for all the DOS/VS jobs contained in the POWER/VS job is constructed. This POWER/VS job definition technique can be used to cause POWER/VS to schedule multiple DOS/VS jobs as if they were one job so that they are executed in a given sequence and one after the other, (not concurrently). In this situation, however, once POWER/VS has scheduled the POWER/VS job for execution, the DOS/VS system still treats each DOS/VS job in the POWER/VS job as a separate job as usual. Thus, if one of the DOS/VS jobs in a POWER/VS job is canceled, the DOS/VS jobs that follow the canceled job in the POWER/VS job are still executed.

A DOS/VS job (but not a DOS/VS job step) can contain multiple POWER/VS jobs. In this case, a POWER/VS job consists of one or more steps of a DOS/VS job., One read queue entry for each POWER/VS job in the DOS/VS job is constructed. This POWER/VS job definition technique was used in DOS/VS POWER to segment spooled printer and punch output and to assign different attributes to the spooled printer and punch output from different steps of a DOS/VS job. POWER/VS supports a segmentation facility and permits multiple LST and PCH statements to appear in one POWER/VS job. Therefore, there is little need to define POWER/VS jobs that consist of one or more steps of a DOS/VS job.

If a JECL JOB and EOJ statement are not present for a DOS/VS job or are provided on a one-for-one basis, the POWER/VS and DOS/VS job definition is the same. The job name assigned to the POWER/VS job is taken from the DOS/VS JOB statement if a POWER/VS JECL JOB statement is not present. Note that when job name is specified in a POWER/VS command, the POWER/VS job name rather than a DOS/VS job name must be used.

The JECL statements and their functions are as follows (exceptions for writer-only partitions are discussed later):

- JOB, which can be placed between DOS/VS jobs or job steps to delimit POWER/VS jobs. If a two-to-eight character job name is specified, it is placed in the read queue entry for the POWER/VS job. Job name can contain alphameric, slash, dash, and period characters. The job attributes that can be assigned using the JOB statement are job class, priority for scheduling the job (which is also the priority for transcribing its spooled printer and punch output if priority is not specified on LST/PCH statements), and input job disposition.

  If a class is not specified in a JECL JOB statement, the class specified in the JECL CTL statement in effect for this input stream, if any, is assigned to the job. If no CTL statement is in effect, the class assigned to the read task that read the job is assigned to the POWER/VS job. The default priority assigned is the priority specified in the PRI parameter of the POWER system generation macro.

  Up to 16 bytes of user information can be included in the JECL JOB statement. This information is placed in the account records for the POWER/VS job, if POWER/VS accounting is used, and in the separator pages for spooled printer output of the POWER/VS job, if output separation is used. If the job logging facility is in effect, user information from the POWER/VS JOB statement is included

in the data displayed on the SYSLOG device at POWER/VS job start time.

- EOJ, which normally is placed after a DOS/VS EXEC or /& statement to indicate the end of a POWER/VS job. The EOJ statement is accepted by POWER/VS wherever it is placed in the input stream.

- CTL, which can be placed before any JECL or DOS/VS JOB statement to establish the input class (partition-dependent or partition-independent) that is to be assigned to POWER/VS jobs for which a class was not specified in the JECL JOB statement. A CTL statement remains in effect until another CTL statement is encountered or until the read task terminates.

- LST (or PRT), which can be placed anywhere in an input stream (including within data input) to specify attributes for the spooled printer output of a POWER/VS job. Any number of LST statements can be included in a POWER/VS job and JECL JOB and EOJ statements do not have to be used in order to include LST statements for a POWER/VS job. A LST statement remains in effect until another LST statement for the same spooled printer (as indicated via the LST parameter) is encountered in the POWER/VS job or until the last statement of the POWER/VS job is read. If no LST statements are included in a POWER/VS job definition, the attributes established during POWER/VS generation are assigned to the spooled printer output from the job.

  LST is used to specify for spooled printer files output class, priority, disposition, spooling of printer output to a tape volume instead of the POWER/VS data file, the identification of the RJE user to which the spooled printer file is to be sent or the central operator identification, the number of copies to be printed, the number of output separator pages and whether they are to appear after each copy for multiple copies, the output limitation values, the identification (alphameric) of the form to be used for printing, the format of the carriage control tape (LTAB parameter) or the FCB image to be used in printing the output, the number of pages in a segment, the phase name of the UCB image to be used for transcription, the address of the spooled printer to which the attributes in this LST statement apply, and specifications for the 3800 printer.

  For a 3800, the following can be specified: number of copies in a group (up to eight group values can be specified), whether the output is to go to the burster-trimmer stacker, up to four character arrangement tables to be used during file printing, whether the 3800 is to be set with defaults specified in a SETDF command, the name of the forms overlay frame to be used, and the copy modification phase and character arrangement table to use when printing copy modification text.

  The LTAB parameter, which specifies the carriage control tape format, is used when a job step whose spooled printer output will be written to a printer depends on sensing channel 12 or 9 (a non-FCB type printer) to indicate page overflow for printer output. The LTAB parameter specifies the distance between the channels in the carriage control tape that is to be used for actual printing. The LTAB specification is used to establish an internal representation of the page format.

  During execution of the job step, the execution list task initiates a line counter at the beginning of each page of spooled output to a given spooled printer device and increments the counter using line print and skip to channel commands from the executing program and the internal representation of the page format. When the counter indicates a full page of lines has been written, the execution list

tasks turns on the overflow bit in the CCB for the I/O operation to the spooled printer to indicate end of page to the executing program. If the LTAB parameter is not specified for a non-FCB type printer, the specifications in the LTAB parameter of the POWER generation macro are used.

The LTAB parameter specification is ignored when the FCB parameter is also present in a LST statement. In this case or when only the FCB parameter is specified, the internal representation of the page format is established using the image specified in the FCB parameter. If an LFCB macro is issued by the executing program, the execution list task updates the internal representation of the page format to reflect the specified FCB image. If no FCB parameter (or LST statement) is specified for a printer with an FCB, the standard FCB load for the printer type (that used during IPL) is loaded.

The LST parameter specifies the spooled printer to which the LST statement applies in terms of the spooled printer device address or the logical unit assigned to the spooled printer. When the LST parameter is not specified in a LST statement, the attributes the LST statement contains apply to the first printer specified as a spooled device in the PRINTERS statement when the partition was started under POWER/VS control. The LST parameter must be used, therefore, when a job step is to write to more than one spooled printer and different attributes are to apply to different spooled printers.

- PUN, which can be placed anywhere in an input stream (including within input data) and provides the capability of specifying the same types of attributes for spooled punch output as the LST statement supplies for spooled printer output except UCB and FCB images, carriage control spacing, and a compaction table (UCB, FCB, LTAB, and CMPACT parameters).

- RDR, which can be placed anywhere in an input stream to indicate that input is to be read from one or more diskette devices (up to 255) and inserted in the input stream being read from the card reader. The diskette file can contain POWER/VS and/or DOS/VS job control statements and input data for job steps. The number of diskettes to be processed, sequence checking for multiple diskettes, file verification, and automatic ejection and feeding of a new diskette when end-of-file is reached on the current diskette can be requested.

- SLI, which can be placed anywhere in the input stream to specify the name of a book that is to be read during job step execution and inserted in the input being supplied to the job step. The book can be contained in the private source statement library assigned to the POWER/VS partition or in the system source statement library. Source library update statements can be included in the input stream to modify the contents of the specified book (add, delete, and replace statements in the book). Use of the SLI statement does not require the inclusion of JECL JOB and EOJ statements.

- DATA, which must be preceded by a JECL SLI statement. The DATA statement is used to insert card or diskette data from the input stream in the data file into a book being read for submission to an executing job step.

Intermediate Storage

POWER/VS requires direct access storage dedicated to its use for the data file, queue file, and if POWER/VS accounting is to be used, account file. These POWER/VS files can be allocated on 2314/2319, 3330-series

(all models), 3340 (all models), 3344, or 3350 (in native or 3330 compatibility mode) disk storage. DLBL and EXTENT statements for these files must be user supplied. The POWER/VS files can be placed on disk volumes that contain other DOS/VS files (that is, the volumes containing POWER/VS files need not be dedicated to POWER/VS). However, for performance reasons, it is recommended that POWER/VS files not be placed on disk volumes that contain heavily used files.

- Tape can be used for intermediate storage of spooled printer and punch files from locally submitted jobs only. Multiple spool files can be written on a tape volume and a spool file can be contained on multiple tape volumes. Tape intermediate storage can be used, for example, when direct access space is limited or when spooled printer or punch output is to be saved for transcription at a later time.

  Note that no label processing is performed on tapes to which spool files are written. That is, the execution list/punch task begins writing a spool file to the specified tape without reading the tape to check for the presence of labels and without writing a label. Tape positioning is a user responsibility.

Data file. The file name of the POWER/VS data file is IJDFILE. The data file can consist of from one to five extents, each of which must begin and end on a cylinder boundary (split cylinders are not supported). Each extent of the data file must be on a separate volume. Programmer logical units SYS002 to SYS006 for the POWER/VS partition are used to access the data file extents. All extents of the data file must reside on disk volumes of the same type. When a 3340 is used for the data file, all extents must reside on the same type 3348 Data Module. However, the data file, queue file, and account file each can be placed on a different device type of those supported for POWER/VS intermediate disk storage.

The space allocated to the data file is divided into track groups. The maximum number of track groups permitted in the entire data file is the number that will fit in the space allocated in the one to five extents of the file. A track group contains an integral number of cylinders as specified by the user via the TRACKGP POWER/VS generation parameter or as assigned by default, depending on the device type used for the data file. The minimum number of tracks in a track group is one and the maximum is one cylinder of tracks. If the number of tracks in a cylinder on the device type used for the data file is not a whole multiple of the number of tracks in a track group, the remaining tracks in the cylinder are allocated to the last track group in the cylinder.

The size of the physical records written in the track groups of the data file is the size specified in the DBLK POWER/VS generation parameter. DBLK can be a minimum of 544 and a maximum of 2008 bytes. If the UCS parameter is to be used on a LST or PRT statement for a 3211 printer, the minimum DBLK size is 608. If a DBLK size is not user specified, a default block size is chosen based on the device type used, as shown below.

Device type	Default DBLK size	Approximate number of 80 column cards per block	Approximate number of 132 character lines per block
2314/2319	920	11	7
3330-series Models 1, 2, and 11 and 3350 in 3330 compatibility mode	952	12	7
3340 and 3344	808	10	6
3350	960	12	7

Note that trailing blanks in card input that is spooled to the data file are eliminated, as are trailing blanks in spooled output cards and spooled output lines written to the data file. Therefore, the figures shown above indicate the minimum number of cards and lines per data block. Note also that specification of a DBLK size smaller than the default for a given device increases disk arm activity and can affect POWER/VS performance. In addition, the default block sizes are not necessarily optimal for track utilization. Therefore, block size should be that which best utilizes the track capacity of the data file device type, subject to the availability of the required real storage for buffers.

Block size can be changed only when a cold start of the POWER/VS system is performed. When POWER/VS is warm started, the block size established during the last cold start is utilized.

The DBLK specification determines the size of the data buffers that are used by the POWER/VS reader, execution processor, and writer tasks. These buffers are allocated in the POWER/VS partition. A data buffer is a multiple of 32 minus 8 bytes in size, that is, (N X 32)-8 bytes where N can vary from 16 to 63. Data block size is equal to the smallest integral multiple of 32 minus 8 bytes that is equal to or greater than the DBLK specification. The same data block size is used for both the input and the output buffers assigned to a reader or writer task.

If a reader task, RJE reader task, or execution list/punch task requires a track group and none is available in the data file, the requesting task is placed in the wait state, the operator is notified, and system operation continues. The operator can do nothing or make space available by deleting an existing queue entry. He can also start additional writer tasks.

Queue file. The file name of the queue file is IJQFILE. The queue file consists of only one extent and is accessed using programmer logical unit SYS001 for the POWER/VS partition. Split cylinders are not supported for the queue file. The queue file contains one master record, which is written as the first record of file, and as many queue records as will fit in the space allocated to the queue file extent. The master record and each queue record is 152 bytes in length. The number of queue records allocated should be one for each track group contained in the data file plus a few additional records for POWER/VS use.

A copy of the master record is kept in the POWER/VS partition. Whenever this record is updated, it is rewritten to the queue file so that it will be up to date if needed for a warm start.

A queue record in the queue file is required to point to the location of each assigned track group in the data file. If the total number of queue records is less than the total number of track groups, the extra track groups cannot be utilized. The operator is notified when the queue file is totally allocated and system operations continue.

If a queue record is required for assignment to a track group and none is available, the requesting POWER/VS task (reader task, RJE reader task, or execution list/punch task) is placed in the wait state until a queue record becomes available. The operator is notified of the out-of-space condition.

Each POWER/VS job read by a POWER/VS read task is assigned one or more track groups to contain the DOS/VS job control statements, any POWER/VS job control statements, and any card or diskette input for the job steps in the DOS/VS job(s). The track groups allocated to a given POWER/VS job are not necessarily contiguous in the data file. Space in the last or only track group assigned to POWER/VS job that is not used

by the job remains unused. It is not allocated to any other POWER/VS job.

The queue records that are allocated to the track groups for a given POWER/VS job form a queue set and are chained together via pointers. The queue set itself is contained in one of the input class chains within the total input (reader) queue. The read queue entry for a job consists of the queue set records in the queue file plus the track groups in the data file pointed to by the queue set records. Figure 80.40.2 illustrates the relationship between a queue set, queue records, and a queue entry.

When output segmentation is not used, each spooled printer file and each spooled punch file written by the steps of a given DOS/VS job is allocated one or more track groups as required. Therefore, each spooled printer file and each spooled punch file created by a DOS/VS job has a list or punch queue entry, as appropriate, that consists of queue set records plus the associated track groups in the data file. The queue set for each spooled printer file is contained in one of the output class chains within the list queue and the queue set for each spooled punch file is contained in one of the output class chains within the punch queue.

When output segmentation is used, a spool file consists of multiple segments. Each segment of each spooled printer and each spooled punch file is allocated one or more track groups as required. Therefore, a segmented spooled printer or punch file has one list or punch queue entry for each of its segments instead of only one queue entry.

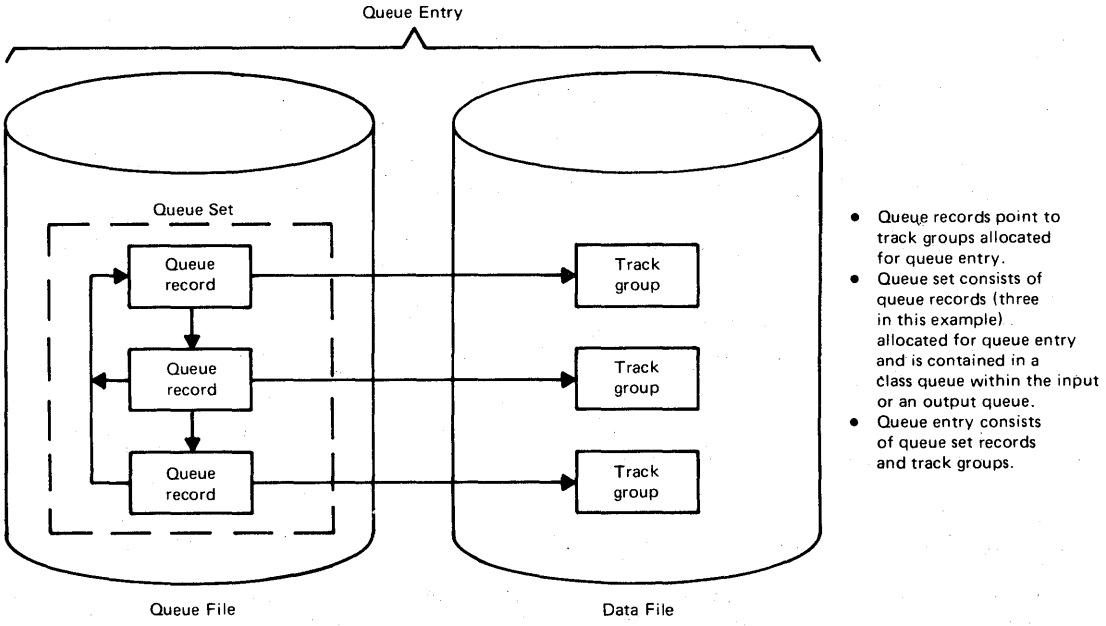


Figure 80.40.2. Relationship between a queue set, queue records, and a queue entry

During POWER/VS operation, status information about the queue file is maintained in the POWER/VS partition. There is a pointer field for each class in the reader queue, list queue, and the punch queue. The pointer for a class in a queue indicates the record address of the first and last queue record in the class chain. The pointer field also indicates whether there is a queue entry in the class chain that can be scheduled

for execution (reader queue pointers) or transcription (list and punch queue pointers).

Account file. The optional account file has the filename IJAFILE and is accessed using the programmer logical unit SYS000 for the POWER/VS partition. It consists of only one extent. The account file contains six different types of records and is written as a DOS/VS sequential, variable-length unblocked file in chronological order (see discussion under "Job Accounting").

Starting and Stopping POWER/VS

POWER/VS can be started in a virtual partition any time after DOS/VS initilization. Before starting POWER/VS, the operator should ensure that the following conditions exist:

- The partition in which POWER/VS is to execute is stopped, if it was previously active, and has assigned the amount of virtual and real storage required by the POWER/VS system to be used.

  Note that in addition to the programmer logical units SYS000 to SYS006 that are used for the POWER/VS intermediate disk storage files, enough programmer logical units must be defined for the POWER/VS partition to accommodate all the reader and writer tasks (including RJE reader and writer tasks) that can be active concurrently. Programmer logical units used for reader tasks and writer tasks are assigned and unassigned automatically by POWER/VS as required.

- The unit record devices to be used by POWER/VS during its initialization are unassigned.

- The partitions POWER/VS is to schedule have lower priority than that of the partition in which POWER/VS is to be initiated and have been stopped or unbatched.

- The POWER/VS files (queue file, data file, and account file if POWER/VS accounting is to be used) are mounted and ready for processing.

After the above conditions are met, the operator can issue the DOS/VS START command for the partition to be used by POWER/VS. Initiation of POWER/VS can then be accomplished manually or using the AUTOSTART facility. When the AUTOSTART facility is used, POWER/VS start-up commands can be entered via the SYSIPT device instead of via the SYSLOG device.

When the AUTOSTART facility is not used, the operator must do the following to initiate POWER/VS:

- If not already assigned, assign the SYSRDR device for the POWER/VS partition if job control statements for the POWER/VS files (called POWER/VS initiation statements) are to be read from a card reader instead of entered via the SYSLOG device.

- If not already assigned, assign a SYSLST device if a status report is to be printed after a warm start of POWER/VS or if a dump of the POWER/VS partition is desired if abnormal termination of POWER/VS occurs during its initialization. SYSLST need not be assigned if neither output is desired.

- If not already assigned, assign the queue file (SYS001), data file (SYS002 to SYS006), and the account file (SYS000); supply DLBL and

EXTENT statements for these POWER/VS files; if the account file is to be dumped to a standard labeled tape or a disk, supply DLBL and EXTENT statements for the file; and supply the EXEC statement for the POWER/VS system to be started. These POWER/VS initiation statements can be entered via the SYSRDR device for the POWER/VS partition, if the device was assigned, or entered via the SYSLOG device. DLBL and EXTENT statements need not be supplied if they are stored on the label cylinder of the system residence volume.

POWER/VS initialization begins after the above steps have been performed. During the initialization, POWER/VS code is loaded into the POWER/VS partition and POWER/VS files are opened and, optionally, formatted. The FORMAT QUEUES message is issued during POWER/VS initialization to enable the operator to specify whether formatting should occur. If formatting of the queue or data file is requested, a cold start is performed. The queue file only, account file only, queue file and account file, queue file and data file, or queue file, data file, and account file can be formatted.

When formatting of the queue or data file is not requested, a warm start is performed. A warm start causes POWER/VS to begin processing the jobs and spool output files that currently exist in the POWER/VS queue and data files. If the SYSLST device for the POWER/VS partition is assigned, a warm start causes a status report to be printed that lists the jobs in the reader queue and the spool files in the list and punch queues. The following is listed for each queued entry as appropriate: job name, job number, priority, disposition, class, number of copies, number of pages or cards, and forms identification.

The status report also lists information about the queue, data, and account files. For the queue file, the number of queue records currently assigned and available is given. The total number of tracks in the file, track group size, and data block size for the data file is given. For the account file, the total number of tracks allocated and the percentage of the file that is filled is given.

When POWER/VS initialization is completed, a message is issued to the operator and the SYSRDR, SYSIPT, and SYSLST devices for the POWER/VS partition are unassigned. The operator must issue POWER/VS PSTART commands to start the partitions POWER/VS is to control, POWER/VS reader and writer tasks, communications lines POWER/VS RJE,BSC is to handle (if any), and activate the VTAM interface if RJE,SNA support is to be used.

When a partition is started with a PSTART command, the operator is prompted to specify the device addresses of the unit record devices for which POWER/VS is to provide spooling. One reader, one diskette, up to eight printers, and up to eight punches can be specified for each POWER/VS controlled partition. POWER/VS will intercept all I/O requests from the partition for the logical units assigned to the unit record devices designated as spooled devices.

If no jobs with the class(es) assigned to a partition are queued at the time the partition is started with a PSTART command or any time thereafter until the partition is stopped with a POWER/VS PSTOP command, the partition enters the wait state and the operator is notified. Processing in the partition starts or resumes automatically as soon as a POWER/VS job with a class the partition is assigned is available in the reader queue.

When the AUTOSTART facility is to be used, after the POWER/VS partition is started the operator must assign the POWER/VS partition SYSIN device to a card reader, tape, or disk. The POWER/VS initiation cards followed by POWER/VS AUTOSTART statements can then be placed in the assigned card reader.

AUTOSTART statements can include the FORMAT statement to specify whether POWER/VS file formatting is desired; PSTART commands to start the partitions POWER/VS to control; READER, PRINTERS, and PUNCHES statements to indicate the unit record devices to be spooled in the started partitions (READER=NO defines a writer-only partition while specification of PRINTERS=NO and PUNCHES=NO defines a reader-only partition); PSTART commands to start POWER/VS reader, list, and punch tasks; and PSTART commands to start RJE,BSC lines and activate the VTAM interface.

AUTOSTART statements must be followed by a DOS/VS /* job control statement, which can be followed by the input stream POWER/VS is to read from the card reader. POWER/VS initiation statements and AUTOSTART statements each can be supplied via a cataloged procedure instead of via a card reader. Note that the EXEC POWER statement should not be supplied via a cataloged procedure as this approach prevents any updating of the procedure library during the entire time POWER/VS is in operation.

Initialization of POWER/VS via the AUTOSTART facility is performed as for a manual initialization as the input deck is read. The operator is prompted for a formatting specification for POWER/VS files if a FORMAT statement is not supplied in the AUTOSTART input and notified if any PSTART or spool device specification statements are incorrect.

Normal termination of POWER/VS is accomplished by issuing the POWER/VS PEND command. The PEND command causes all POWER/VS tasks to be terminated after processing associated with the current queue entry is completed. That is, reader tasks continue until their current POWER/VS job is read and queued in the reader queue, writer tasks continue until the current spool file or spool file segment is transcribed, and POWER/VS jobs executing in POWER/VS controlled partitions are allowed to complete.

The PEND command also causes the POWER/VS partition and all POWER/VS controlled partitions to be returned to normal DOS/VS operation after they complete their current operations. If a printer address is specified in the PEND command, a status report similar to the one printed after a warm start is printed before POWER/VS termination.

When immediate termination of POWER/VS is required because of an emergency, the PEND command with the KILL operand is used. This command causes read tasks to terminate immediately without completing the reading of any POWER/VS job they are currently handling. Such jobs are not queued and must be resubmitted when POWER/VS is restarted. Writer tasks are stopped immediately also. The spool output files being transcribed by the terminated writers remain queued in the list and punch queues and will be transcribed from the beginning if a warm start of POWER/VS is initiated later. POWER/VS jobs executing in POWER/VS controlled partitions are canceled and remained queued in the reader queue.

If a printer address is specified in the PEND KILL command, a DUMP macro is issued to print all virtual storage in the POWER/VS partition before POWER/VS termination. The supervisor area included in this dump depends on the dump specification in the STDJC system generation macro or the OPTION statement in effect for the POWER/VS partition.

The operator is notified when POWER/VS is to be abnormally terminated because of an error and can request that a dump of the POWER/VS partition be taken. All POWER/VS controlled partitions are also abnormally terminated when the POWER/VS partition terminates abnormally.

Dummy Assignments and Dummy Devices

ASSGN statements have to be provided for the devices that are to be spooled for POWER/VS controlled partitions, just as for any other I/O devices. In addition, any device that is specified as a spooled device in a READER, PRINTERS, or PUNCHES statement must be defined in the PUB table in the DOS/VS supervisor as usual (via the DVCGEN DOS/VS system generation parameter or the ADD IPL statement).

However, since the I/O requests to a logical unit assigned to a spooled device are intercepted by POWER/VS, a physical unit record device is not actually used for these I/O operations and the assignment for a spooled device is in effect a dummy assignment. Further, if the physical device that is specified as a spooled device does not actually exist in the system configuration, the dummy assignment is actually being made to a dummy device.

In certain situations in POWER/VS, assignments for spooled devices must be made to dummy devices. The need for dummy devices occurs much less often for POWER/VS than for DOS/VS POWER because of a change in the way assignments for spooled devices in POWER/VS-controlled partitions are handled by the DOS/VS job control program.

In POWER/VS, logical units in different POWER/VS-controlled partitions can be assigned to the same physical unit record device if that physical unit is identified as a spooled device for each of the partitions. In addition, the same physical unit record device can also be assigned to the POWER/VS partition for use by a reader/writer task. As a result, spooled devices for POWER/VS-controlled partitions do not have to be assigned to dummy devices except in certain situations (discussed later).

For example, assume POWER/VS is to execute in partition F1 to control the operation of partitions BG, F2, and F3, each of which is to use one spooled card reader, one spooled punch, and one spooled printer. The same card reader, punch, and printer can be assigned to each of the POWER/VS-controlled partitions. To avoid the use of dummy devices, these three devices can be those physically present in the I/O configuration and used by the POWER/VS partition (that is, by reader and writer tasks). This is illustrated in Figure 80.40.3. Note that the assignments for the spooled devices for POWER/VS-controlled partitions can be made during DOS/VS system generation as well as during system operation.

In a DOS/VS POWER environment, the situation illustrated in Figure 80.40.3 requires the definition and assignment of one set of dummy devices (card reader, punch, and printer) for the spooled devices for one POWER-controlled partition and another set for the spooled devices for a second POWER-controlled partition. The third POWER-controlled partition can have real unit record devices assigned to it (the same that are used by reader and writer tasks in the POWER partition). The dummy device requirement exists because for DOS/VS POWER, logical units in different POWER-controlled partitions cannot be assigned to the same unit record device.

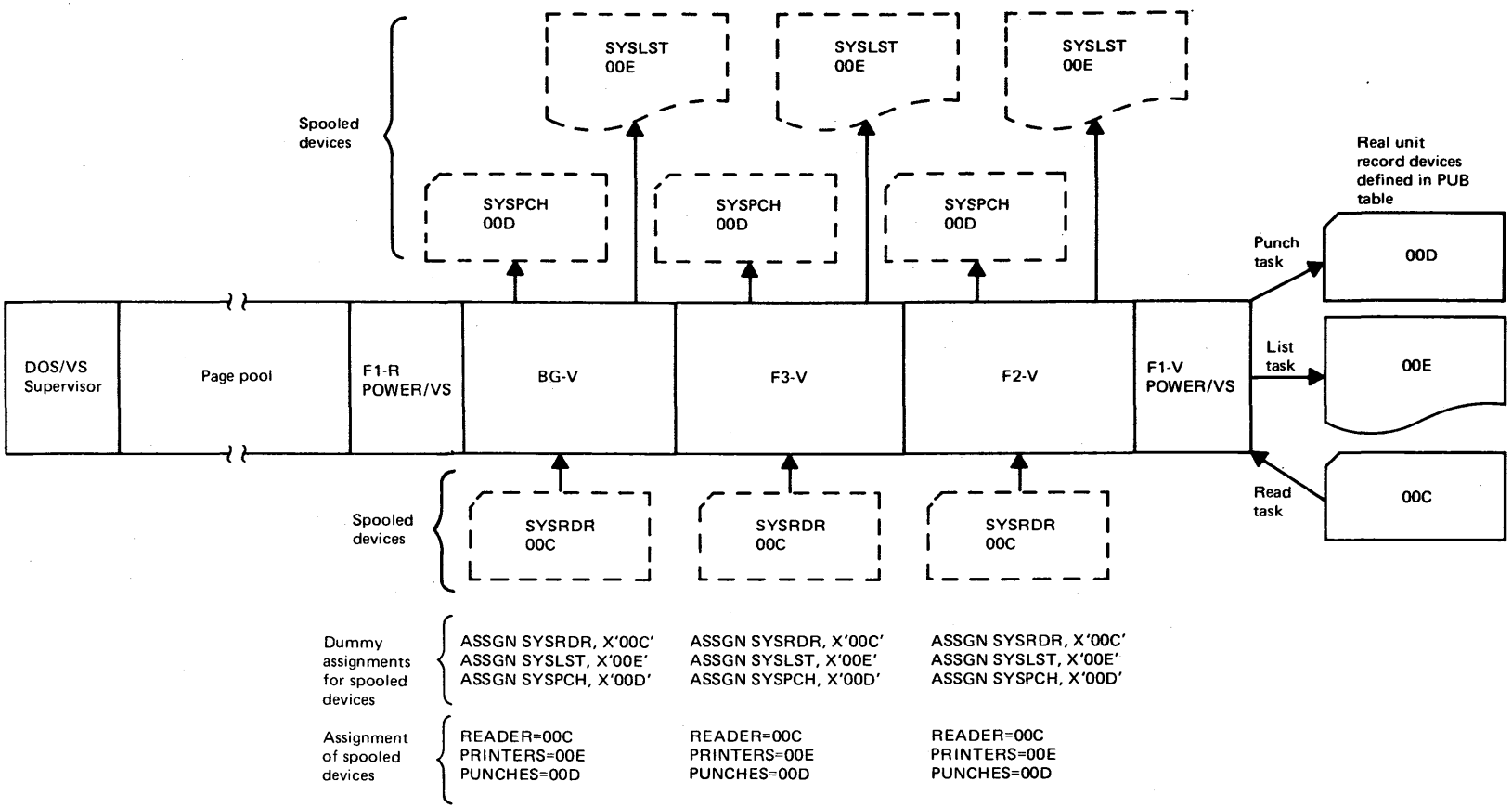


Figure 80.40.3. Example of spool device assignments in POWER/VS without the use of dummy devices

For POWER/VS, the following situations require the use of dummy devices:

- A writer-only POWER/VS-controlled partition is to read directly from a card reader, say 00C. The other POWER/VS-controlled partitions that are to read from a spooled card reader cannot assign the spooled reader to 00C because 00C is owned by the writer-only partition. Therefore, a dummy device must be defined in the PUB table and assigned to the spooled card readers.

  Figure 80.40.4 illustrates this situation. POWER/VS operates in partition F1 to control partitions BG, F2, and F3. In the example, BG is the writer-only partition. Dummy devices are not required for the spooled printers and punches used by the POWER/VS-controlled partitions. Note that reader 00C cannot be used by a reader task in the POWER/VS partition while it is being used directly by the writer-only BG partition.

- A reader-only POWER/VS-controlled partition is to write directly to a printer/punch. This situation is like that for a writer-only partition. The POWER/VS-controlled partitions that are to use a spooled printer/punch must have a dummy device assigned to the spooled device.
- A POWER/VS-controlled partition is to use more than one spooled printer or punch. Different logical units in the partition cannot be assigned to the same printer/punch device. Therefore, one of the spooled printer/punches can be assigned to a real printer/punch while the other spooled printers/punches are assigned to dummy devices. Figure 80.40.5 illustrates the situation in which POWER/VS executes in the F1 partition to control the operation of the BG and F2 partitions. F2 is using three spooled printers.

- A 1442, 2560, or 5425 is to be used by a POWER/VS-controlled partition for both card reading and punching. These devices have only one device address. Since the two logical units to be used for reading and punching cannot be assigned to the same device, one logical unit must be assigned to a dummy device. The other can be assigned to a real device.

- A cardless system (Model 115 or 125) is to perform spooling to diskette devices. A dummy device must be assigned to each spooled device.

The addresses selected for dummy devices should be selected such that they do not cause incorrect assignment of device addresses to UCW's for the byte multiplexer channel. Incorrect UCW assignment can cause a UCW folding condition that can place the system in a hard wait state. Note that address 01D should not be used as a dummy punch device for a Model 115 or 125 system in which POWER/VS is being used.

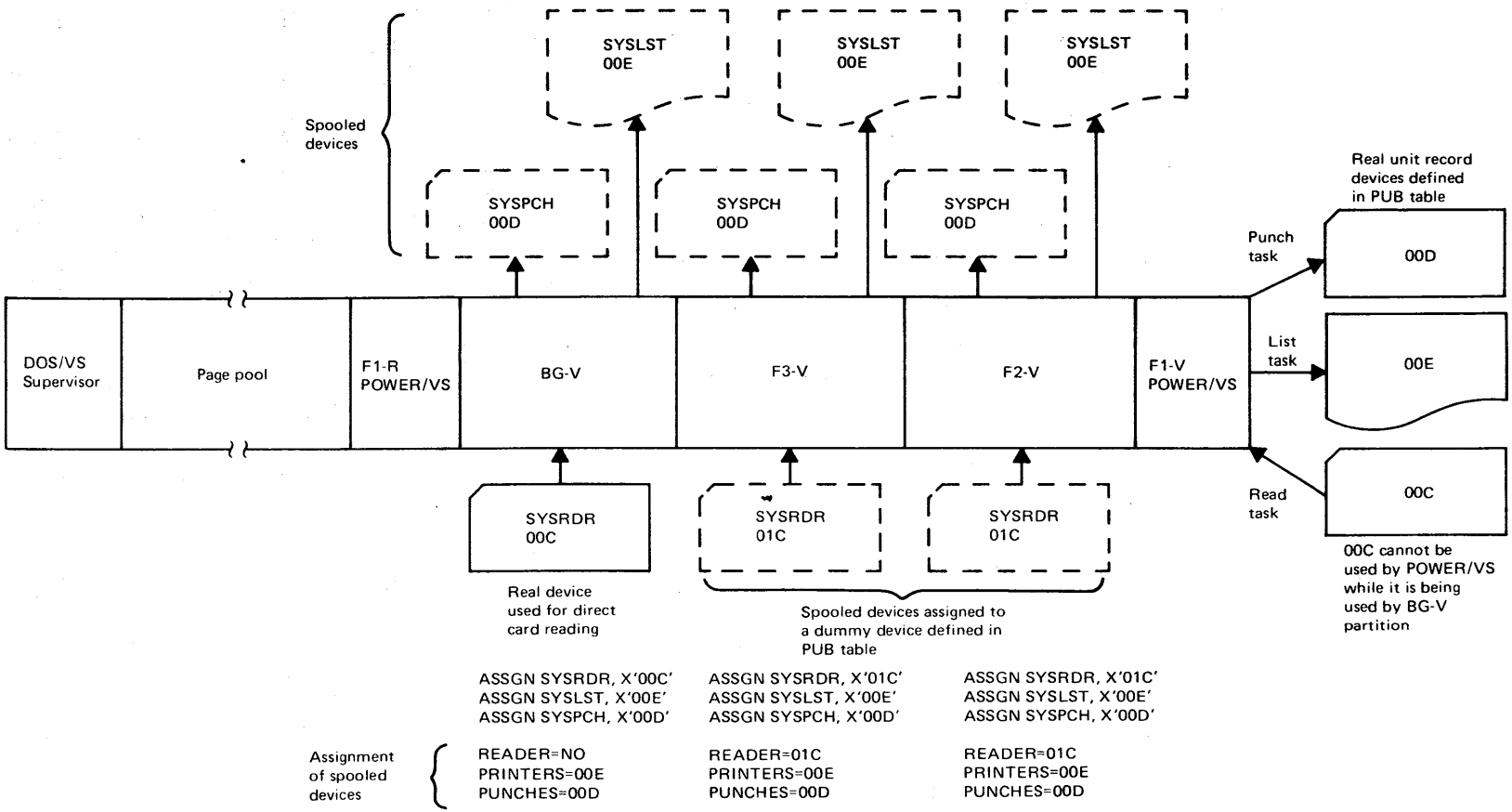


Figure 80.40.4. Example of the use of a dummy device when a card reader is used directly by a POWER/VS writer-only partition

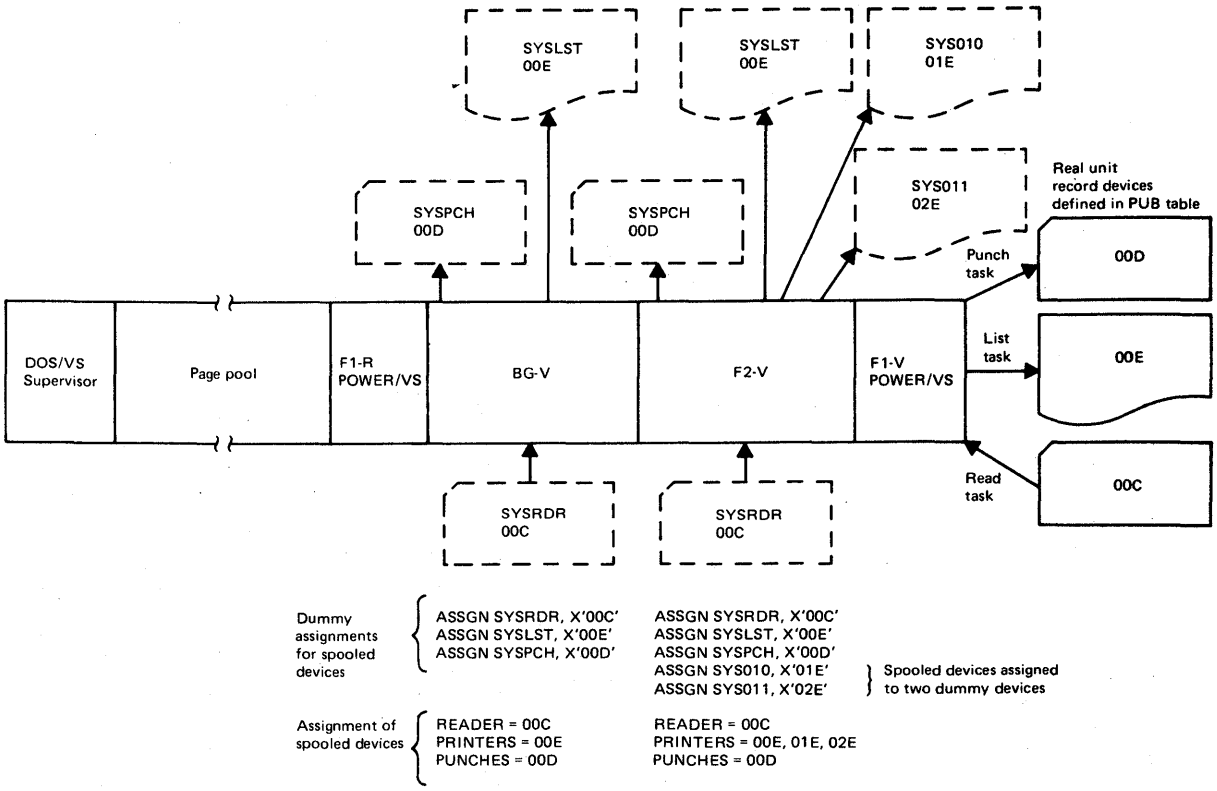


Figure 80.40.5. Example of the use of a dummy device when a POWER/VS-controlled partition uses more than one spooled printer

POWER/VS Tasks

The POWER/VS program consists of a group of control and functional tasks that perform job scheduling and spooling operations. The POWER/VS program provides its own multitasking support (private subtasking) and does not require the presence of multitasking support in the DOS/VS supervisor under which it operates. The POWER/VS program contains a page fault overlap appendage routine to enable it to retain CPU control when a page fault occurs in the POWER/VS partition so that another ready POWER/VS task, if any, can be dispatched.

Multitasking in the POWER/VS partition is controlled by the main task, which is never referenced externally. Other control tasks are the following:

- Command task, which initiates and terminates other POWER/VS tasks and handles POWER/VS commands entered by the operator.
- Wait task, which places the POWER/VS partition in the wait state when necessary.
- RJE,BSC line manager task, which controls the communications lines supported by POWER/VS when RJE,BSC support is utilized. This task is attached when the first BSC line is started and stopped when the last BSC line is stopped.

- RJE,SNA manager task, which controls the activation of transmission processing to and from an SNA remote terminal. The task is attached when the central operator issues a PSTART RJE,SNA command to begin remote job entry operations for SNA terminals.
- RJE,SNA logon tasks, which initialize and establish a session between POWER/VS and a remote SNA terminal.
- RJE,SNA logoff task, which terminates a session between POWER/VS and a remote SNA terminal.
- RJE,SNA message task, which handles the sending of messages to remote SNA terminals.
- Spool manager task, which controls the activation and deactivation of the internal reader task and the spool/command manager list task. This task is attached during POWER/VS initialization when POWER/VS cross partition communication support is included in the system and detached when POWER/VS is terminated.
- Internal reader task, which performs the read operation when a PUTSPOOL cross partition communication macro is issued.
- Spool/command manager list task, which performs the retrieval operation when a GETSPOOL cross partition communication macro is issued and the command invocation for a CTLSPOOL cross partition communication macro.
- Account task, which handles the disposition of the account file that is written by POWER/VS job accounting support.
- Status task, which writes the status of the queue file to the SYSLOG device, a local printer, or a remote printer.

The POWER/VS functional tasks are the following:

- Reader/writer tasks that consist of read tasks, list tasks, and punch tasks.
- Execution processor tasks that consist of execution read tasks, execution list tasks, and execution punch tasks.
- RJE tasks that consist of RJE,BSC and RJE,SNA read tasks, RJE,BSC and RJE,SNA list tasks, and RJE,BSC and RJE,SNA punch tasks. These RJE tasks are discussed later under "Remote Job Entry Support".

The POWER/VS main task maintains a task selection list for dispatching purposes in the following high to low priority sequence:

- POWER/VS initiator or terminator task
- RJE,BSC line manager task
- RJE,SNA manager task
- Command processor tasks
- RJE reader/writer tasks in the sequence they were started with PSTART commands
- Writer tasks in the sequence they were started
- Execution processor tasks for POWER/VS-controlled partitions in the sequence in which the partitions were started

- Reader tasks in the sequence they were started
- Command task

If none of the above tasks are ready to execute when POWER/VS searches the task selection list for a dispatchable task, POWER/VS issues an SVC 7 instruction to return control to the DOS/VS supervisor.

Read tasks. A read task reads an input stream from a card reader and/or 3540 diskette device. For each POWER/VS job read, a read queue entry is placed in the reader queue. The read task places the information contained in the JECL JOB (or defaults for these parameters) in the read queue entry for each POWER/VS job it reads. In addition, all POWER/VS JECL, DOS/VS job control, and card and/or diskette input in the input stream for the POWER/VS job are transcribed to spool input files in the POWER/VS data file on disk. The read task processes only JECL JOB, RDR, CTL, and EOJ statements.

A read task is started in the POWER/VS partition when the POWER/VS PSTART RDR command is issued by the operator or supplied in the input to the AUTOSTART facility. The PSTART command indicates the address of the card reader, diskette device, or card reader and diskette device to be assigned to the read task (and owned by the POWER/VS partition). The device specified must be available, that is, not currently assigned to another partition.

When a read task is started, it is assigned an available programmer logical unit in the POWER/VS partition for its read operations to the specified device. The read task is not started if no programmer logical unit is available in the POWER/VS partition. As many read tasks as there are available card readers and/or diskette devices can operate concurrently. See Table 80.40.1 for the specific card readers and features supported by POWER/VS read tasks.

When a read task is started for a card reader, one input buffer is allocated to it from the POWER/VS partition unless the PSTART command specifies that two input buffers are to be allocated. A read task for an input stream on a diskette device is allocated only one input buffer. POWER/VS attempts to obtain one buffer large enough to read in one entire diskette track at a time. One output buffer in the POWER/VS partition is assigned to a read task for disk (data file and queue file) write operations.

A read task has the standard name RDR assigned to it. Read tasks are distinguished from each other by a suffix to the standard name that consists of the device address of the card reader or diskette device the task is assigned. Read tasks are reentrant.

Optionally, an input class can also be specified on a PSTART RDR command to assign an input class to the read task. The input class assigned to a read task by the operator or by default is assigned to each of the POWER/VS jobs read by the read task unless a JECL JOB statement that specifies an input class was included for the job or a JECL CTL statement is currently in effect for this input stream. The input class specified in a JECL JOB statement overrides any CTL statement class in effect. If an input class is not specified for a read task, it is assigned class A by default. The same input class can be assigned to more than one read task.

Table 80.40.1. I/O devices supported by POWER/VS

Card Reading Devices

1442, 2520, and 2540 Card Read Punches
2501 Card Reader
2560 Multifunction Card Machine
3504 and 3505 Card Readers (Optical Mark Read and Read Column Eliminate are not supported)
3521, 3525 Card Punch with Card Read feature
5425 Multifunction Card Unit (96-column card reading is supported)

Note: Column binary reading, stacker selection, and interpreting are not supported for the card devices listed above. Reading and punching into the same card is not supported.

Card Punching Devices

1442, 2520, and 2540 Card Read Punches
2560 Multifunction Card Machine
3521, 3525 Card Punch
5425 Multifunction Card Unit (96-column card punching is supported)

Note: Program-controlled stacker selection is supported for the 2560 and 5425.

Printing Devices

1403, 1443, 3203, 3211, and 5203 Printers (UCB and FCB loading are supported)
3800 Printing Subsystem
2560 Multifunction Card Machine with card printing
3521, 3525 Card Punch with card printing
3784 Line Printer
5425 Multifunction Card Unit with card printing

Note: Punch and interpret as well as punch and multiline print operations on the same card are supported. For the 3525, multiline printing, automatic line positioning, user-controlled line positioning, and print overflow are supported.

Intermediate Storage (for the queue file, data file, and account file)

2314 Direct Access Storage Facility (A and B models)
2319 Disk Storage
3330-Series Disk Storage,all models
3340 and 3344 Direct Access Storage (all models)
3350 Direct Access Storage (all modes)
2400-Series Magnetic Tape units (for spooled punch and print data only)
3400-Series Magnetic Tape Units (for spooled punch and print data only)

Diskette Devices

3540 Diskette Input/Output Unit (as an input stream device only)

Table 80.40.1 (continued)

Remote Job Entry Terminals
2770 Data Communication System (BSC support)
2780 Data Transmission Terminal (BSC support)
3741 Data Station Model 2 (BSC support as a 2780)
3741 Programmable Work Station Model 4 (BSC support as a 2780)
3771 Communication Terminal (BSC support as a 2770 and SNA support as a nonprogrammable terminal)
3773 Communication Terminal (BSC support as a 2770 and SNA support as a nonprogrammable terminal)
3774 Communication Terminal (BSC support as a 2770 and SNA support as a nonprogrammable terminal)
3775 Communication Terminal (BSC support as a 2770 and SNA support as a nonprogrammable terminal)
3776 Communication Terminal (BSC support as a 3780 and SNA support as a nonprogrammable terminal)
3780 Data Communication Terminal (BSC support)
3790 Communication System (SNA support)
System/32 (as a 3770)
Teleprocessing Control Units for RJE Terminals
2701 Data Adapter Unit
2703 Transmission Control Unit
3704/3705 Communications Controllers operating in emulation mode
Integrated Communication Adapters for Models 115, 125, 135, and 138

The input (job) priority assigned to each POWER/VS job read by a read task is that specified in a JECL JOB statement included for the job, if any. If a JECL JOB statement is not present for a POWER/VS job or did not specify a priority, the default priority specified during POWER/VS generation is assigned to the job. The disposition assigned to each POWER/VS job read by a read task is that specified in the JECL JOB statement included for the job. If a JECL JOB statement was not included or did not specify a disposition, the default disposition of D is assigned to the POWER/VS job, which causes it to be automatically scheduled for execution by priority within input class.

A unique job (sequence) number (1 to 65,535) is assigned to each POWER/VS job read by a read task. The job name and job number together uniquely identify a POWER/VS job in case duplicate job names are used. The job number assigned to a POWER/VS job is also placed in the list queue and punch queue entries for the spooled output of the job. When a cold start of POWER/VS is performed, job number is reset to 1.

The input stream a POWER/VS read task reads from a card reader can contain DOS/VS job control and, optionally, POWER/VS JECL statements. Optionally, card input data for the job steps defined can be included as well. Multiple cards decks can be included for a job step. These decks will be read sequentially from the spooled card reader.

Input for the job steps defined in a card input stream can also be supplied using the source library inclusion (SLI) facility. The JECL SLI statement can be included in the input stream to specify the name of a source statement library book and, optionally, a sublibrary name. If a private source statement library is assigned to the POWER/VS partition, it is searched first for the specified book. The system source statement library is searched next or when no private source statement library is assigned to the POWER/VS partition.

The searching for and reading of a source statement book is not performed by the read task. This function is performed by the execution read task when it encounters an SLI statement during execution of the DOS/VS job step for which the SLI statement was included.

When the SLI facility is used, source library update statements can be placed in the card input stream to insert additional source statements in the book to be read and to replace or delete existing source statements. The changes are made to the source statements presented to the job step (and not to the book in the source statement library).

In addition, the JECL DATA statement can be placed in the input stream after an SLI statement to cause the card input deck following the DATA statement to be included in the data transcribed to the POWER/VS data file. This data is then presented to the DOS/VS job step during its execution as an insert to the source statement book specified in the SLI statement that precedes the DATA statement (as if the deck were actually present in the book being read from the source statement library). The input following a DATA statement in the input stream is presented to the job step at the time the execution read task encounters a DATA statement in the source statement book it is reading that has the name specified in the DATA statement in the input stream. The DATA statement is ignored for a writer-only partition.

A book that will be included in the input stream using the SLI facility can also contain DOS/VS job control statements and JECL LST and PUN statements. The latter cannot have continuation statements. If columns 1-4 of LST and PUN statements in an SLI book contain the four characters * $x, where x is a character other than a blank or $, the SLI book can be cataloged using a POWER/VS-controlled partition (LST and PUT statements are treated as comments). Otherwise, a partition not controlled by POWER/VS must be used to catalog the SLI book.

The partition-independent naming convention supported for partition related cataloged procedures can be used to name SLI books containing job control where necessary. If the first two characters of the SLI book name are $$, when the SLI book is encountered during processing of the job stream POWER/VS changes the second $ to B or 1 through 4 (or 6), depending on the partition for which the SLI book is being processed.

The following restrictions apply to the use of the SLI facility: nested SLI statements are not permitted; an SLI book may invoke a DOS/VS procedure but a DOS/VS procedure may not invoke an SLI book; each SLI book represents a POWER/VS procedure and may not contain embedded JOB statements (but one POWER/VS job may contain multiple SLI statements); LST and PCH statements in an SLI book cannot be continued; the SYSRDR, SYSIPT, and SYSIN devices cannot be reassigned with an SLI book; and the private source statement library containing the SLI books to be used must be assigned to the POWER/VS partition before POWER/VS is started.

The 3540 is also supported as an input stream device for data mode and SYSIN mode operations. When the 3540 is used to supply data only (no job control statements) for an input stream contained on cards, data mode is in effect. Multiple diskettes (up to 255) containing data records from 1 to 128 characters in length can be used in data mode. Multiple data files per 3540 volume can also be read in data mode.

Data mode is invoked by including a card reader and a 3540 physical device address in the PSTART command for a read task and placing one or more JECL RDR statements in the card input stream. The RDR statement specifies the physical device the job step is to use to read the input data. Multiple RDR statements can be included in one POWER/VS job, but they all must specify the same physical device. When a RDR statement is encountered during reading of the card input stream, the read task

begins reading input from the 3540 diskette device specified in the PSTART command for the read task and writes it to the data file. The RDR statement can request volume sequence checking when multiple diskettes are to be read.

Job steps that are to use the data read from the 3540 use DTFDU or DTFDI to read data from the logical unit assigned to the 3540 that was specified in the RDR statement. The logical unit cannot be assigned to the same physical device as the spooled card reader for the partition. (Note that POWER/VS intercepts I/O requests to the SYSIN device for a POWER/VS controlled partition only if it is assigned to a card reader.)

Note that although a diskette file can be inserted in the middle of a card input stream using data mode, card data cannot be inserted in the middle of a diskette file and one diskette file cannot be inserted in the middle of another diskette file.

When SYSIN mode is used for a 3540, the input stream can consist of job control statements (both DOS/VS and POWER/VS JECL) and data contained in both a card reader and on a 3540. The 3540 input can be multivolume and consist of job control and data 80 or 81 characters in length. As for data mode, the card reader and 3540 to be used in SYSIN mode are specified in the PSTART command, and RDR statement(s) in the input stream indicate when the read task is to read from the 3540. However, any input data from the card reader and 3540 for each job step must be read from the card reader that is designated as the spooled card device and DTFDU cannot be used.

A POWER/VS read task can also read an input stream that is totally contained on a 3540 diskette device using SYSIN mode. The input stream on a 3540 diskette can contain DOS/VS job control and optionally POWER/VS JECL statements (80 or 81 characters in length) as well as input data (80 or 81 character records) for the job steps defined. Input streams on diskette devices can be multivolume (up to 255 volumes). The PSTART command specifies the 3540 that contains the input stream. Data supplied via a diskette input stream is read into a POWER/VS-controlled partition from a spooled card reader.

The source library inclusion facility and DATA statement can be utilized with diskette input streams as described for card input streams. However, a RDR statement cannot be included in a diskette input stream. Figure 80.40.6 summarizes the input stream device combinations and contents supported by POWER/VS.

A user-written routine that receives control after each DOS/VS job control (including comment, /*, and /&) or POWER/VS JECL statement is read from an input stream by any read task can be included in POWER/VS. This routine executes in the POWER/VS partition and must be relocatable (or self-relocating if the relocating loader is not present in the DOS/VS supervisor) and reentrant. If the user exit routine terminates abnormally, it also causes POWER/VS to terminate abnormally.

The user-written reader exit routine can inspect the statement just read and return a code to POWER/VS to indicate the action to be taken. This reader exit routine can indicate the statement is to be processed normally, deleted, or replaced with one or more statements supplied by the exit routine. Alternatively, the exit routine can indicate that the POWER/VS or DOS/VS job containing the statement is to be flushed.

Note that only //, *, $$, /*, and /& statements in the input stream are passed to the exit routine, but the contents of a referenced book or procedure is not. In addition, when a DOS/VS $JOBEXIT routine is included to inspect job control statements at execution time, the contents of books included via the JECL SLI statement and procedures invoked via the EXEC PROC statement are passed to the $JOBEXIT routine

but POWER/VS JECL is not (because it is deleted from the input stream by the execution read task).

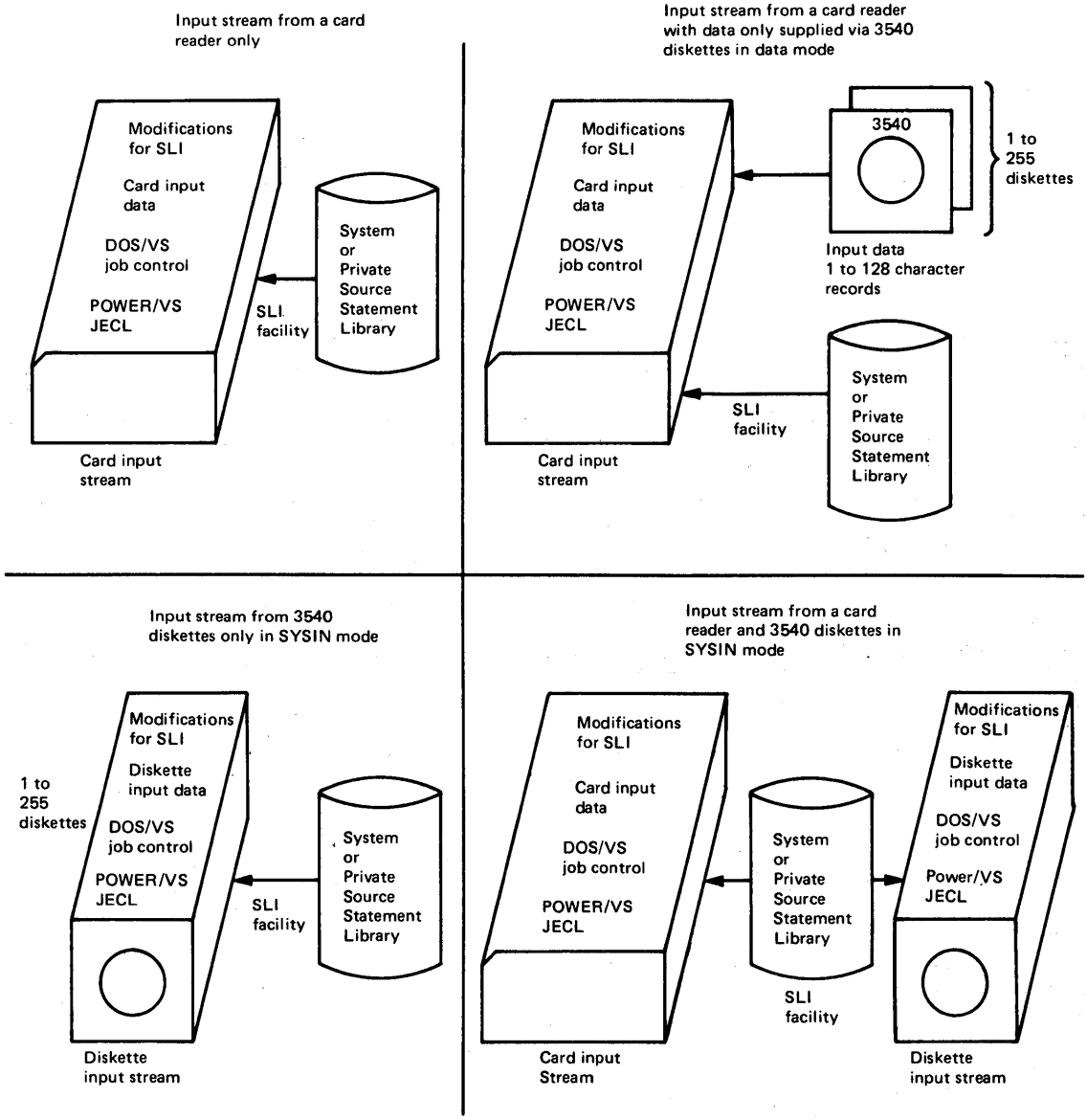


Figure 80.40.6. Input stream device combinations and contents supported by POWER/VS

A read task consists of a physical read routine and a logical read routine. The physical read routine performs device-dependent read operations to the assigned input stream device and passes the data read to the logical read routine. The logical read routine performs the functions required to write the input stream to the POWER/VS data file and place read queue entries for the POWER/VS jobs read in the appropriate position in the reader queue in the POWER/VS queue file.

One copy of a device-dependent physical read routine is used by multiple read tasks that are assigned the same type input stream device. Only one copy of the logical read routine is required. It can be associated with any number of device-dependent physical read tasks.

If the POWER/VS queue file or data file is located on direct access storage that has the rotational position sensing feature and RPS support is included in the DOS/VS supervisor, SET SECTOR commands are used in the channel programs initiated by the logical read routine to write queue records and the input stream to these files.

Read tasks construct real channel programs and issue EXCP macros with the REAL parameter specified to handle card reading, queue file writing, and data file writing so that channel program translation is avoided. A command-chained channel program that reads multiple cards or diskette records is initiated for each read operation. The number of cards or diskette records read per I/O operation to the input stream device depends on the buffer size specified during POWER/VS generation in the DBLK parameter.

When the end of a job stream in a card reader is reached, the associated read task enters the wait state and the buffers in the POWER/VS partition the task is assigned are released. The operator is notified that the read task is waiting for work. When an unsolicited device end from the card reader is received (indicating additional cards have been placed in the reader and it is again ready), one or two buffers are allocated to the read task and it is automatically restarted to resume reading the input stream. When end of file is reached for a job stream contained on one or more 3540 diskette devices, the associated read task is terminated.

A read task is also terminated when the operator issues the POWER/VS PSTOP command. If the read task is in the process of reading an input stream when the PSTOP command is issued, the read queue entry for the POWER/VS job being read is not placed in the reader queue unless the EOJ parameter is specified. EOJ causes the read task to continue reading the input stream until the current read queue entry is completed and queued, after which the read task is terminated. The buffers assigned to a read task are released when the task is terminated and the operator is notified of the termination.

Execution read tasks. An execution read task is automatically started in the POWER/VS partition when a partition is brought under the control of POWER/VS via a POWER/VS PSTART command and is stopped when a PSTOP command is issued to terminate POWER/VS control of a partition. An available buffer in the POWER/VS partition is allocated to the execution read task for disk read operations when it is started and deallocated when the execution read task is stopped. Only one execution read task can be active for a partition that is controlled by POWER/VS.

An execution read task has the standard name "ppR" assigned in which the pp represents the POWER/VS-controlled partition the task is to service. Execution read tasks are reentrant. The execution read task for a partition receives CPU control for each read request from the partition for that card reader or diskette device for which spooling was specified when the partition was started with a PSTART command. More than one logical unit in the partition can be assigned to the card reader that is spooled.

The execution read task performs the job scheduling function for its associated reader/writer or reader-only partition. It also reads the input in the data file for the job it is to initiate, performs required processing of certain JECL statements in the job input, deletes all JECL statements, and supplies DOS/VS job control statements to the DOS/VS job control program. The execution read task processes all JECL statements except CTL and RDR, since all required processing of these two statements has been done by a read task.

When a reader-only or reader/writer partition is started under POWER/VS control, job scheduling occurs as follows. After the job

control program is loaded into the partition, the first read it issues to the spooled card reader device causes the execution read task to receive control. The execution read task searches the input queue for a job with a class this partition is assigned.

If no job can be scheduled in the partition, the execution read task places the partition in the wait state and notifies the operator that the partition is waiting for work. The buffer allocated to the execution read task is released and the execution read task also enters the wait state. Whenever a read task places a job in the input queue, it determines whether there is a waiting partition that can process the job. If so, the appropriate execution read task is given control to schedule the job in its partition. In addition, if the operator changes a job class or disposition that makes a queued job available for a waiting partition, the appropriate execution read task is also reactivated.

When there is a job the execution read task can schedule, the read queue entry for the job is read by the execution read task to obtain the location of track group(s) in the data file for the job. The execution read task then reads in the first physical record in the spool input file for the job, which should contain a JECL or DOS/VS JOB statement as the first logical record.

When the execution read task encounters a JECL JOB statement, job logging is performed. The job logging facility is included in all POWER/VS systems. It is activated by default unless JLOG=NO is specified during POWER/VS generation. When job logging is active, the execution read task displays the following on the SYSLOG device when it encounters a JECL JOB statement:

- POWER/VS job name and job number
- User information from the POWER/VS JECL JOB statement, if any
- User identification of the remote user that submitted the job (for remotely submitted jobs only)

After performing job logging, the execution read task deletes the JECL JOB statement (does not present it to the job control program) and inspects the next logical input record. If this is a LST or PCH statement, the execution read task obtains an available list or punch queue record and inserts the information from the LST/PCH statement in the queue record. A track group is not allocated at this time for the spool file the LST or PCH statement describes.

When a LST statement with the UCB or FCB parameter specified is encountered by the execution read task, it issues a UCB or FCB load request to the spooled printer to which the LST statement applies. This causes the execution list task to receive control. When the execution list task determines the request is for a UCB or FCB load, it writes the load channel program and image to the data file so that the load operation will be the first I/O operation performed when the spool file is transcribed by a list task.

If an erroneous LST (PRT) or PUN statement is encountered the operator is notified and can correct the error or flush the affected job.

After processing a LST/PCH statement, the execution read task deletes the statement and inspects the next logical record. If it is a DOS/VS JOB statement, the execution read task moves the statement from the POWER/VS partition to the partition it is controlling (to the input area specified in the card read request by the job control program). The job control program, which has been waiting since issuing a read request to

the spooled card reader, is then given CPU control to process the DOS/VS JOB statement.

Each time the job control program issues a read request to the spooled card reader, the execution read task receives control and inspects the next logical record in the spool input file for this POWER/VS job (performing disk read operations as they are required). Any JECL statements encountered are processed by the execution read task and deleted while all other statements are presented to the job control program.

When the DOS/VS job control program receives a DOS/VS EXEC statement from the execution read task, it loads the specified phase for execution. Thereafter, any read request issued to the spooled card reader by the executing program are intercepted by the execution read task.

When the execution read task intercepts a request from an executing program, it simulates the card or diskette read operation by passing an input record from the appropriate spool input file in the POWER/VS data file to the requesting job step. The record is from the spool input file that was transcribed to the data file by a read task when the job step was read. When reading from the POWER/VS data file, an execution read task uses real channel programs, the EXCP macro with the REAL parameter, and sector commands if the RPS feature is present for POWER/VS direct access storage and RPS support is present in the DOS/VS supervisor.

When the execution read task reads an SLI statement in the spooled input, it searches for the specified book. If the book is found, the source statements it contains are presented to the executing program each time the program issues a read request to the spooled card reader just as if the source statements had been present in the spool input file.

The execution read task continues to intercept spooled card reader requests during program execution and to process any JECL statements it encounters until end of POWER/VS job occurs, which will also be the end of a DOS/VS job or job step. This condition causes the DOS/VS job control program to be brought into the partition. When job control issues a card read request to the spooled card reader, the execution read task receives control and searches the input queue for another POWER/VS job to schedule in the partition.

For a writer-only partition, jobs to be executed in the partition are submitted via a card reader, tape, or disk SYSRDR/SYSIN device and are read by the job control program as usual. These jobs must contain JECL JOB and EOJ statements and may contain LST and PCH statements to specify spool output file characteristics for jobs. Continuation of a JOB statement is not supported for jobs that are to execute in writer-only partitions. JECL RDR, CTL, SLI, and DATA statements in the input stream are treated as comments.

The JECL JOB and EOJ statements must be included when LST or PCH statements are present for a job that is to execute in a writer-only partition. LST and PCH statements for writer-only partition jobs cannot be placed within input data for the job. They must be presented at job control time. If SLI or DATA statements are encountered in a job, they are ignored and an error message is issued.

When a writer-only partition is started, an execution read task for the partition is started as usual. The execution read task intercepts all I/O requests to the SYSLOG device in order to locate any JECL statements in the jobs being read by the job control program. JECL statements are treated as comments by the job control program and,

therefore, are listed on SYSLOG. The execution read task performs the required processing of JECL JOB, LST, PCH, and EOJ statements.

An execution read task for any POWER/VS-controlled partition also starts and stops the execution list and execution punch tasks that also are to service the partition that the execution read task is assigned. The execution read task, execution list task, and execution punch task for a given POWER/VS-controlled partition are collectively called the execution processor for the partition.

Execution list and execution punch tasks. The execution read task starts an execution list task in the POWER/VS partition for a POWER/VS-controlled partition the first time an executing job step issues a request to any one of the printer devices that are to be spooled for the partition. Similarly, an execution punch task is started in the POWER/VS partition by the execution read task the first time an executing job step issues a request to any one of the punch devices that are to be spooled for the partition. When started, an execution list/punch task obtains a track group from the data file in which to write spooled data.

When end of POWER/VS job occurs in the partition, the execution read task stops the execution list and execution punch tasks it started for the partition and the buffers allocated to the stopped tasks are released.

An execution list task has the standard name "ppL" and an execution punch task has the standard name "ppP" in which the pp represents the POWER/VS-controlled partition the task is to service. Execution list and punch tasks are reentrant.

An available buffer from the POWER/VS partition is allocated to an execution list or execution punch task for disk output operations when it is started and deallocated when the task is terminated. A maximum of one execution and one execution punch task can be active at a time for a POWER/VS-controlled partition. Execution read, list, and punch tasks for a partition operate asynchronously with each other and with other POWER/VS tasks.

The execution list task for a partition receives CPU control for each request from the executing job step that is issued to the one to eight printer devices designated to be spooled for the partition. When an execution list task intercepts a request, it simulates the print operation by moving the data to be printed and the channel program required to perform the printing to its buffer in the POWER/VS partition. The channel programs contained within a buffer are chained together so they can be initiated by a list task using one EXCP macro. When the buffer is filled, the execution list task writes the buffer to a spooled printer file in the POWER/VS data file (or to tape, if this option was specified).

An execution list task checks each I/O request it intercepts to determine whether it contains a command to load the FCB or UCB of the specified spooled printer. When such a request is received, the execution list task writes the specified image to the data file for use when the spool file being written is transcribed to a real printer. If the FCB is being loaded, the execution list task also updates the internal representation of the page format for the specified spooled printer. As discussed later, segmentation also occurs.

The execution punch task for a partition receives CPU control for each request from the executing job step that is issued to the one to eight punch devices designated to be spooled for the partition. When the execution punch task intercepts a request, it simulates the punch operation by moving the data to be punched and the channel program

required to perform the punching to its buffer in the POWER/VS partition. The channel programs contained within a buffer are chained together so they can be initiated (by a punch task) using one EXCP macro. When the buffer is filled, the execution list task writes the buffer to a spooled punch file in the POWER/VS data file (or to tape, if this option was specified.)

The execution list and execution punch tasks use real channel programs, the EXCP macro with the REAL parameter, and sector commands if the RPS feature is present for POWER/VS direct access storage and RPS support is present in the DOS/VS supervisor to write spooled output to the POWER/VS data file and queue entries to the queue file for the spooled files.

When the output segmentation facility of POWER/VS is not used, list and punch queue entries created for spooled printer and punch files from a POWER/VS job are not placed in the appropriate output queues in the POWER/VS queue file until all steps of the job have been processed. Thus, transcription of spool files to printer and punch devices cannot begin until end of POWER/VS job occurs.

The output segmentation facility enables a print or punch queue entry to be queued while a DOS/VS job step is still executing. This facility enables the transcription of spool output files for a job step to occur concurrently with execution of the job step if a writer task is available. Output segmentation can be used only when spooled printer or punch output is written to disk.

Output segmentation can be controlled using count-driven, data-driven, or program-driven segmentation. When count-driven segmentation is to be used, the RBS parameter can be specified at POWER/VS generation to indicate the number of pages (up to 999,999) and/or number of punched cards (up to 999,999) that are to comprise a segment. Each segment created by count-driven segmentation is assigned the same job number as the associated reader queue entry.

The count-driven segmentation values specified at POWER/VS generation can be overridden for a job step by including in the input stream a JECL LST or PUN statement with the RBS parameter specified for the POWER/VS job that contains the job step. These statements can also be used to specify segment values when output segmentation was not specified during POWER/VS generation. If count-driven segmentation is specified for a spool file that has the I attribute, segmentation is suppressed.

Data-driven segmentation occurs when more than one JECL LST/PCH statement that specifies the same spooled printer/punch is included in one POWER/VS job. Data-driven segmentation cannot be used in a writer-only partition. Normally, data-driven segmentation is controlled by placing multiple LST/PCH statements for the same spooled device within input data in the input stream.

When the execution read task encounters a LST/PCH statement when it is reading spooled input for a job step, it initiates closing of the spool file associated with the last LST/PCH statement that specified the same spooled device as this LST/PCH statement. A new segment using the specifications in this LST/PCH statement is then begun (a list/punch queue record and track group are obtained).

The list/punch queue entry for the first segment created by data-driven-segmentation is assigned the same job number as the associated reader queue entry. However, each subsequent segment is assigned a unique job number. This enables the operator to manipulate individual segments. The job number in the account file for the job is that of the reader queue entry so that all output accounting data for a single job can be sorted together.

Program-driven segmentation occurs when a LFCB, SEGMENT, or SETPRT macro is issued in a program. Segmentation occurs for a spooled printer file when the executing job step issues an LFCB macro instruction to change the FCB image to be used. All printer records written in the spool file up to this point are included in the current segment and another segment is begun.

The SEGMENT macro can be used to segment printer or punch spooled output. It can be used only in POWER/VS-controlled partitions that have been started as multitasking partitions. When issued, the SEGMENT macro causes a new segment to be begun for the spooled device specified in its DEVADDR parameter. A return code indicates the success or failure of the execution of the SEGMENT macro. The first segment created is assigned the job number in the associated reader queue entry. Each successive segment is assigned a unique job number.

Optionally, the SEGMENT macro can also specify the four-character identification of the form to be used for the new segment (NAME parameter), and the location of a JECL (JOB, LST, or PUN) statement within the program that is to be used for the new segment (JECL parameter). The NAME operand should not be specified in a SEGMENT macro that specifies a JOB JECL statement. JOB statements sould be specified in a SEGMENT macro only to provide new user information. The NAME parameter can be used to assign a unique name to segments with DISP=I (output that is to be returned to the reader queue).

When a JECL PRT or PUN statement is specified in a SEGMENT macro, it can contain any valid parameters and PRT or PUN statements need not be submitted via the job stream. The parameters specified in a JECL statement submitted via the SEGMENT macro override the corresponding parameters that are in effect for the spooled device.

The SEGMENT macro provides the programmer with the ability to change the attributes of different segments of a spooled file as well as to direct different segments to different spooled devices.

Segmentation also occurs when the SETPRT macro for the 3800 printer is issued to request a printer setup that requires operator intervention, change copy grouping, or specify a copy number greater than 1.

When a printer or punch segment is completed, the execution list or punch task updates the associated list or punch queue entry and places it in the appropriate class chain within the list or punch queue. Segments that belong to the same spooled printer file are chained together in the list queue. Similarly, segments that belong to the same spooled punch file are chained together in the punch queue. Any command issued to a segmented spooled printer or punch file is effective for all segment queue entries in the chain.

Once the queue entry for a segment has been placed in a class queue within the list or punch queue, it can be selected for transcription by a writer task. The queue entry for a segment is deleted as soon as transcription of the segment the required number of times has been completed if the entry has the D disposition attribute. The segments of a given segmented spool output file are not necessarily transcribed consecutively, since queue entries for other spool files with the same class as that of the segmented spool file can be queued between the queue entries for segments of a given spool file. When a segmented printer spool file is being transcribed, backspacing can be done only to the beginning of the segment currently being processed.

The output limitation facility can be used to control the number of lines and/or cards spooled by a POWER/VS job. The print line and card limits to be established for each job can be specified in the STDLINE

and STDCARD POWER/VS generation macros, respectively. The limit for each can be a value up to 999,999. A zero value indicates no output limitation. The JECL LST and PUN statements can be used to override the limits specified at POWER/VS generation or to request output limiting when it was not specified at POWER/VS generation.

When the execution list or punch task determines that the print line or card output limit for the executing POWER/VS job has been reached, the operator is notified and processing continues. The operator can terminate further job processing by issuing a POWER/VS PFLUSH command. The read queue entry for the flushed POWER/VS job is deleted unless the HOLD parameter is specified in the PFLUSH command.

If the operator does not issue a PFLUSH command, job processing continues until an additional number of print lines or punch records have been written by the execution list/punch task. The additional limits are also specified in the STDLINE and STDCARD POWER/VS generation parameters or in the JECL LST and PUN statements. When the additional limit is reached (up to 999,999 lines or cards), the operator is again notified and has the same options of flushing the job or letting it continue. The operator continues to be notified each time the additional limit is reached until the POWER/VS job is flushed or completes.

List and punch (writer) tasks. List and punch tasks transcribe to printers and punches spooled printer and punch files, respectively, that are written to the POWER/VS data file (or tape) by execution list and execution punch tasks. All active list tasks are scheduled to transcribe spooled printer files using the list queue. All active punch tasks are scheduled to transcribe spooled punch files using the punch queue. List and punch tasks operate asynchronously with each other and other POWER/VS tasks.

A writer task is started in the POWER/VS partition when the POWER/VS PSTART LST/PUN command is issued by the operator or supplied in the input to the AUTOSTART facility. As many list tasks can be started as there are available printers and as many punch tasks can be started as there are available punches in the I/O device configuration.

The address of the printer or punch device that is to be assigned to a writer task and owned by the POWER/VS partition is specified in the PSTART command. Table 80.40.1 lists the printer and punch units and features supported by POWER/VS writer tasks. The device specified in the PSTART command must be available, that is, not currently assigned to another partition. When a writer is started, it is assigned an available programmer logical unit assigned to the POWER/VS partition. The writer task is not started if no programmer logical unit is available in the POWER/VS partition.

When a list task is started, buffers are assigned according to the n parameter specification in the PSTART command for the list task. A maximum of two input and two output buffers can be assigned to a list task. Buffers are allocated from the POWERVS partition. When the output buffer is filled, the list task initiates a write I/O operation to the printer followed by a read I/O operation to the data file to overlap printer output and disk input I/O operations.

When a punch task is started, it is assigned one input buffer for disk (data file) read operations and one buffer for punch operations. These buffers are also allocated from the POWER/VS partition. A punch operation is initiated as soon as the output buffer is filled. A disk read operation is initiated as soon as the punch operation completes.

A list task has the standard name LST assigned to it. A list task is distinguished from other list tasks by a suffix to its standard name

that consists of the device address of the printer the task is assigned. Similarly, a punch task has the standard name PUN and is distinguished from other punch tasks by a suffix to its standard name that consists of the device address of the punch the task is assigned. Writer tasks are reentrant.

Optionally, the PSTART command can specify the output classes a writer task started to disk is to handle. From one to four classes, A to Z, can be specified. Class A is assigned to a writer task if none is specified in the PSTART command. More than one list/punch task can be assigned the same class.

The class(es) assigned to a writer task determine the spool output files it can transcribe and the sequence in which the classes are specified determines the priority of processing output classes for that writer, as previously described.

A started writer task transcribes spool files contained in the POWER/VS data file unless the address of a tape unit containing spool files is specified in the PSTART command. When a tape unit is specified, a class cannot also be specified and the list or punch task transcribes only the spool file(s) contained on the specified tape unit.

When a list/punch task is started, it searches the list/punch queue for a job it can transcribe. When there are no queued spool files with the class(es) a started writer task is assigned to handle, the writer task is placed in the wait state, its buffers are unallocated, and the operator is notified that the writer task is waiting for work. Whenever an execution list/punch task queues a spool file, it checks for a waiting writer that can process the spool file. If one is found, buffers are allocated to the writer task and it is automatically restarted to begin transcribing the file. In addition, if the operator changes the class or disposition of an output queue entry and makes a job available a waiting writer can process, the writer is reactivated.

Before transcribing a printer spool file or each segment of a printer spool file, the list task loads the FCB requested for the file/segment or the default for the printer type to be used. An FCB load is not performed if the required FCB load is already present in the FCB of the printer to be used.

If more than one copy of a spool output file on disk is to be printed or punched, the JECL LST or PUN statement, respectively, can be included in the input stream to specify the number of copies desired (up to 99). The number of copies parameter is ignored for spool files on tape. The tape must be reprinted after transcription by a writer task in order to obtain additional copies.

When a spool output file or segment has been transcribed the number of times specified in its copies attribute, the list/punch queue entry for the spool file is deleted (queue record in the queue file and allocated track groups in the data file are made available) unless the spool file has the "keep" disposition.

Cards punched by a punch task are directed to pocket 1 if stacker selection is not specified for the punch spool file. When a punch task has completed the punching of a spooled punch file, transcription of the next queued spooled punch file or segment (if any) begins immediately unless the PAUSE=YES parameter was specified at POWER/VS generation. If it was, the punch task issues a forms change message to the operator before beginning transcription of the next spool file and enters the wait state. This enables the operator to remove the deck of cards just punched. The PGO command must then be issued to reactivate the waiting punch task. As an alternate to this procedure, the job separation facility can be used to separate the card decks punched by punch tasks.

The job separation facility can be used to place job separation pages before and after each transcribed spooled printer file and job separation cards before and after each transcribed spool punch file. For segmented files, separation pages/cards are provided before and after each segment is transcribed. The number of separation pages and/or cards (from 0 to 9) to be provided is specified during POWER/VS generation using the JSEP parameter. The JECL LST and PUN statements can be used to override these specifications and to request job separation if a value of zero was specified for either amount in the JSEP parameter.

When both multiple copies and job separation are requested, beginning separator pages/cards are provided before each copy and ending separator pages are provided only after the last copy unless COPYSEP=YES is specified at POWER/VS generation or via a JECL statement. When COPYSEP=YES is specified, separators are placed before and after each copy of a file/segment. Separation after each copy can also be specified via the JSEP parameter or a LST/PUN statement. The COPYSEP specification is used when JSEP does not specify any separation parameter for multiple copies.

Note that when JSEP=1 is specified, four separation pages are printed before transcription of a spool file/segment and four separation pages are printed after transcription is completed. When JSEP=2 is specified, six separation pages are printed before and after transcription.

Job name, job class, job priority and job number are printed on a separator page in large block letters. The word LAST is also printed on the separator pages for a unsegmented file and the last segment of a segmented file. Forms lengths of 44 through 99 lines are supported with printing across the perforation.

Except for a 5425 MFCM, when punch separation is requested, each spool file punched is preceded by from two to eight cards with 12-11-0-8-9 punches in all columns and one card that contains the job name. Two blank cards follow the last card of output from the spooled punch file. For a 5425, from 1 to 9 cards with the job name punched 12 times per card precede each spool file punched. Stacker selection is ignored for punched output when output separation is requested and cards are placed in the default stacker for the punch device.

The JECL LST and PUN statements can specify via the FNO parameter the identification of the form to be used when transcribing spooled printer and punch output. The operator is notified by a writer task when the forms identification for the spool file to be transcribed is different from that for the spool file it just transcribed. The writer task then enters the wait state so that the operator can change the form. The operator can also use the PSETUP command at this time to perform forms alignment. The writer task is restarted when a PGO command is issued. If the first queue entry processed by a writer task after it is started contains a forms identification of blanks, a mount message is not issued.

The FNO parameter should be used when the LTAB parameter is specified to describe a page format, since the FNO parameter causes a pause after a forms change that gives the operator the opportunity to change the carriage control tape on the printer.

A writer task consists of the logical write routine and a physical write routine. The logical write routine reads records from the spooled printer/punch files in the POWER/VS data file (list and punch queues). A physical list or physical punch routine transcribes spooled printer or punch file records read by the logical write routine to a printer or punch.

Physical list and physical punch routines are device-type dependent. If more than one list/punch task is started to the same device type, only one copy of the required physical list/punch routine is used. Only one logical write (list/punch) routine is required to interface with all the device-dependent physical list and physical punch tasks required by the started writer tasks.

Rotational position sensing is supported by writer tasks if the feature is present on the device type that contains the POWER/VS file and RPS support is included in the DOS/VS supervisor. Writer tasks issue EXCP macros with the REAL parameter specified to initiate the command-chained channel programs in spool output files that transcribe spooled output to printers and punches. When an unrecoverable I/O error occurs on a printer or punch during spool file transcription, the operator can cancel the writer task involved, indicate the error is to be ignored, or restart transcription one page/card back.

A writer task is terminated by issuing the POWER/VS PSTOP command. If a writer task is transcribing a spool output file when the PSTOP command is issued and the EOJ parameter is not specified, the writer task is terminated immediately and the list/punch queue entry for the spool file being transcribed remains in the list/punch queue. If EOJ is specified in the PSTOP command, the writer task is not terminated until it completes transcribing the spool file or spool file segment it is currently processing. When EOJ is specified, the list/punch queue entry for the spool file or file segment is deleted after it has been transcribed and the writer task is then terminated.

The RESTART parameter can be specified on the PSTOP command instead of EOJ. In this case, the writer task is terminated immediately without completing transcription and the spool file or spool file segment being processed remains queued. However, when a writer task for the class of the partially processed spool output file is again started, it begins transcription with the record (in the partially processed spool file or segment) after the last record processed before transcription was terminated. When neither EOJ nor RESTART, which are mutually exclusive, is specified in a PSTOP command and a writer is restarted, it begins transcription with the partially processed spool file or segment but at the first record of the file.

The relationship of POWER/VS functional tasks to each other is shown in Figure 80.40.7.

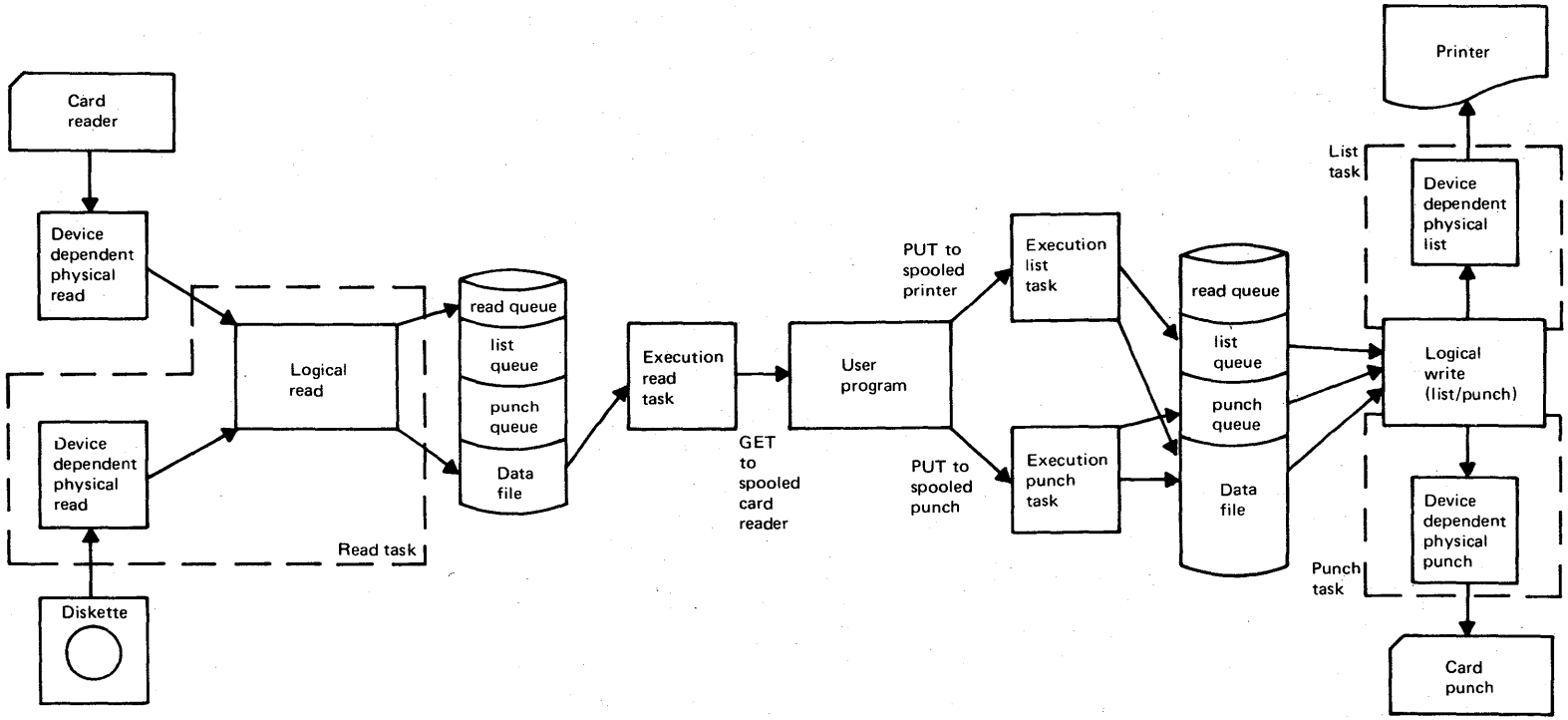


Figure 80.40.7. Relationship of POWER/VS functional tasks

Cross Partition Communication

Cross partition communication, which utilizes the SPL, PUTSPOOL, GETSPOOL, and CTLSPOOL macros, is an optional facility of POWER/VS that is included when SPCOL=YES is specified during POWER/VS generation. The cross partition event control facility is required in the DOS/VS supervisor to support cross partition communication.

The cross partition communication macros provide any partition (whether controlled by POWER/VS or not) the ability to submit jobs to POWER/VS for execution in a POWER/VS-controlled partition, receive the spooled output from a job executed in a POWER/VS partition, and modify the status and attributes of jobs in POWER/VS queues.

Before a cross partition communication macro is issued, the XECBTAB macro must be issued to define a cross partition event control block for the macro to be used. The XECBTAB macro must specify TYPE=DEFINE, XECB=SPMXECB for a GETSPOOL or CTLSPOOL macro or XECB=ICRXECB for a PUTSPOOL macro, and ACCESS=WAIT.

The SPL macro must be used to define the spool parameter lists to be used by PUTSPOOL, GETSPOOL, and CTLSPOOL macros. The SPL macro supplies such data as jobname (default is DUMMY), output disposition (default is K), output class (default is A), the operation requested (for a CTLSPOOL macro), the address of the buffer area that contains the job stream to be submitted or that will receive spooled output, and a buffer area for POWER/VS use and feedback information.

The SPL macro causes a parameter list to be established that will be used by the PUTSPOOL, GETSPOOL, or CTLSPOOL macro that specifies the SPL macro. Parameters within an SPL list can be overridden by specifying the corresponding parameter in the PUTSPOOL, GETSPOOL, or CTLSPOOL macro.

The PUTSPOOL macro is used to submit an input job stream contained in a buffer area in the partition to the POWER/VS input queue for execution in a POWER/VS-controlled partition. The GETSPOOL macro is used to request retrieval of one record (print line) in a list spooled output file. The requested record is moved into a buffer in the partition, as indicated in the RPL list or GETSPOOL macro. The carriage control character associated with the print line can be obtained also. Records can be retrieved sequentially or directly using a line number.

The CTLSPOOL macro can be used to issue five of the POWER/VS commands to perform the following:

- Alter the attributes of a POWER/VS job (class, disposition, priority, and remote terminal designation) via the PALTER command
- Cancel a submitted job (in the reader queue) before it is selected for execution via the PCANCEL command
- Delete from the printer queue via the PDELETE command the list output of a submitted job after its execution
- Display the status of a submitted job via the PDISPLAY command
- Release POWER/VS jobs for further processing via the PRELEASE command

Job Accounting

When support of the job accounting interface is included in a DOS/VS supervisor, job accounting support can also be included in the POWER/VS system. The job accounting support in POWER/VS provides its own

accounting information and collects job accounting data from the DOS/VS job accounting interface. This information is collected for each DOS/VS job step that executes in a partition POWER/VS is controlling and written in the POWER/VS account file.

The following types of account records are written by POWER/VS:

- Reader. One reader account record is constructed for each read queue entry created by a POWER/VS reader task for a POWER/VS job. This record contains such data as the POWER/VS job name, job number, job read start and stop times, input class, input priority, number of records read, and number of tracks of intermediate disk storage required. If a user-written reader exit routine is included in POWER/VS, the number of records read reflects the number of records added or deleted by the user routine.

- List. One list account record is written for each list queue entry created. This record contains such data as POWER/VS job name, job number, print start and stop times, printer device address, output class, output priority, print forms identification, number of lines printed, number of pages printed (including extra pages printed as a result of operator commands, such as PSETUP), and total number of copies printed.

- Punch. One punch account record is written for each punch queue entry created. This record contains such data as POWER/VS job name, job number, punch start and stop times, punch device address, output class, output priority, punch forms identification, number of records punched, and total number of copies punched.

- Execution. One execution account record is created for each DOS/VS job step. This record contains information provided by POWER/VS accounting support and that provided by the DOS/VS job accounting interface. One execution account record is also written for the POWER/VS partition during a normal termination of POWER/VS.

- Line. A line account record is written whenever an RJE user terminates his session. This record contains such data as the user identification, signon and signoff times, line password, line address, total number of transmissions, total number of timeouts, and total number of line errors.

- SNA. An SNA account record is written whenever an RJE,SNA user terminates his session. It contains information as in a line account record.

Records are written to the account file in chronological order. If the account file is not formatted during POWER/VS initialization, new account records are written after the last existing record of the file. When the account file becomes 80 percent full, the operator is notified. When the account file becomes completely filled, the operator is notified and any task requiring the account file is placed in the wait state until the operator makes space available, which can be accomplished using the POWER/VS PACCOUNT command.

The PACCOUNT command can specify that the account file is to be deleted or that it is to be written to tape, disk, or a spooled punch file. When the account file is spooled to disk, it is assigned a disposition of hold, priority 1, and output class P This account spool file can be punched by a POWER/VS punch task with class P assigned after its disposition of hold is changed. The account file can also be processed directly after the termination of POWER/VS operation. The user is responsible for any sorting or summarization of accounting data. The account file can be processed directly by the DOS/VS Sort/Merge program product.

When job accounting support is included in POWER/VS, a user-written job accounting routine ($JOBACCT) is not required unless (1) user accounting information is to be placed in the execution account records generated by POWER/VS, (2) there is a partition that does not operate under POWER/VS control and accounting information for this partition is desired, or (3) accounting information for the POWER/VS partition is desired.

Up to sixteen bytes of user information can be placed in the execution account record. To do this, a user-written $JOBACCT routine that issues the PUTACCT macro must be included in the DOS/VS system. This routine is entered at the end of each DOS/VS job step.

Central Operator Commands

All POWER/VS commands for the central operator begin with the letter P. Except for PEND, each central operator command has a one-character abbreviation. The POWER/VS command language consists of the following types of commands:

- Task management commands that enable the operator to initiate, continue, and terminate POWER/VS tasks.
- Queue management commands that enable the operator to display and modify the contents of queue entries in the POWER/VS queue file.
- Miscellaneous commands that enable the operator to perform forms alignment on printers, save the POWER/VS account file, and communicate with remote users.

The following are the POWER/VS task management commands:

- PSTART, which is used to start a reader task to a card reader and/or diskette device, start a writer task (to a printer, punch, or tape device,) bring a partition under the control of POWER/VS, restart a partition that is under POWER/VS control, start an RJE,BSC task and optionally specify a password the remote operator must supply, or activate the VTAM interface. The PSTART command can also specify one input class for a read task or up to four output classes for a writer task. For a partition, PSTART can specify up to four input classes and one output class that is to be assigned to spool output files for which a class was not specified in a LST/PCH statement. For a card read task, the number of buffers, 1 or 2, to be used to read cards can be specified.

  The PSTART command for a partition can specify the MT parameter to indicate that multitasking is to be used in the POWER/VS-controlled partition and the partition will be long-running or never-ending. When no more spooled input is available for such a partition, only the task that reads spooled input is placed in the wait state. All other tasks continue to operate. Segmentation in such a partition must be controlled using the SEGMENT macro or, for printers with an FCB, program-driven segmentation (LFCB macro).

  The job submitted to a partition with MT specified should contain LST and PUN statements for each spooled output device that can be used by the partition. Each task can spool output to its own printer and punch. The ENQ and DEQ macros should be used to serialize tasks that are to use the same spooled reader, printer, or punch.

  When a writer task is started to a tape unit, a check for the presence of a VOL1 label on the mounted tape is made. If one is found, the operator is notified and can mount a new tape, switch to a disk device, or ignore the condition.

- PSTOP, which is used to stop a reader task, writer task, or RJE (BSC or SNA) task, release a partition from POWER/VS control, or deactivate the VTAM interface. When PSTOP is issued for a partition with a job in execution, the command is not effective until processing of the entire POWER/VS job is completed. After the partition is released from POWER/VS control, all POWER/VS-controlled devices for the partition are unassigned. The partition can then be restarted with a DOS/VS START command or restarted under POWER/VS control by issuing a POWER/VS PSTART command.

  For a read or RJE read task, the point at which the task is terminated depends on whether the EOJ parameter is specified, as previously discussed. For a writer task, the EOJ or RESTART parameter can be specified or not to indicate when task termination should occur. The operator is notified when the task or partition is actually terminated.

- PGO, which is used to restart a POWER/VS writer task or a POWER/VS-controlled partition when the writer or partition is waiting for an operator action (such as forms mounting or changing for a printer, the clearing of a unit check condition on a printer, or mounting of a tape volume). PGO cannot be used to restart a writer or partition that was stopped by a PSTOP command.

- PEND, which is used to terminate the POWER/VS system normally or abnormally with or without a status report or dump, as previously described. The operator is notified when termination of POWER/VS is completed. The printer assigned to the POWER/VS partition remains so assigned.

- PCANCEL, which terminates the printing initiated by a PDISPLAY queue management command

- PFLUSH, which terminates transcription by an active reader task, transcription by an active writer task, or processing in an active POWER/VS-controlled partition. When PFLUSH is issued for a reader task, further processing of the current POWER/VS job is terminated (except for certain LST/PRT statements), a read queue entry is not built for the job, and the reader task continues reading with the next job in the input stream. During reading of the POWER/VS job to be flushed, the read task checks for the presence of LST/PRT statements that specify a UCB or FCB image. If one is found, the read task loads the specified UCB and/or FCB image (for use by a successive job, if necessary).

  For an active writer task, the PFLUSH command causes further processing of the current spool file to be terminated and deletion of the list/punch queue entry for the spool file unless HOLD was specified. When HOLD is specified, processing is terminated; however, the list/punch queue entry is not deleted but is placed in hold status (DISP=K). The list/punch queue entry for a spool file with multiple copies specified is also held whether or not HOLD is specified. The current copy count is saved so that when processing is restarted, it starts at the beginning of the copy that was flushed. The writer task continues processing with the next queued entry with an output class it is assigned to handle.

  When PFLUSH is issued for a partition, further processing of the current POWER/VS job is terminated. If the POWER/VS job consists of multiple DOS/VS jobs, the currently executing DOS/VS job is canceled and the remaining DOS/VS jobs in the POWER/VS job are flushed. The read queue entry for the POWER/VS job is deleted unless the HOLD parameter is specified. The HOLD parameter causes termination of processing but the read queue entry is placed in hold status instead of deleted. The operator is notified when a read queue entry is

deleted. The next POWER/VS job in the read queue with a class the partition is assigned to handle is then started in the flushed partition.

- PRESTART, which causes a POWER/VS writer that is currently transcribing a spool file to forward or backward space a specified number of pages or cards. A signed or unsigned value from 0 to 9999 can be specified to indicate forward spacing (plus sign) or backward spacing (minus sign) from the point of interruption. No sign indicates a specified page/card count from the beginning of the spool file or current segment for a segmented file. Printing or punching resumes at the beginning of the spool file/segment if no value or too large a minus value is specified. When the writer is transcribing to a 3800 printer, the PRESTART can specify a new copy group index.

The following are the POWER/VS queue management commands:

- PDISPLAY, which is used to display the status of POWER/VS jobs and spool files, queued RJE messages that are for all RJE users and the users who submitted them, or the status of POWER/VS resources.

  For a POWER/VS job in the reader queue, status information includes job name, job number, job priority, disposition, class, and (for RJE jobs only) remote identification of the user that submitted the job. For a spool file in the list or punch queue, status information includes that displayed for a POWER/VS job plus number of pages/cards to be written, number of copies, forms identification, and remote identification of the user that is to receive the spool file.

  If a status display is requested for a spool file that is in the process of being transcribed, the number of copies reflects the number of copies remaining to be transcribed, including the current copy, and the number of pages/cards indicates the number still to be printed/punched for the current copy.

  When multiple queues are involved in a display, the name of the queue is printed before the status data for its requested entries. An indication is given when a specific queue is empty. Lines of status data from a given PDISPLAY command can be interspersed with DOS/VS messages.

  A status display can be requested for the following: a specific POWER/VS job in the reader, list, or punch queue; all POWER/VS jobs in the reader, list, or punch queue or in all three queues; all POWER/VS jobs that are in hold or leave status in the reader, list, or punch queue or in all three queues; all POWER/VS jobs that are dispatchable or in keep status in the reader, list, or punch queue or in all three queues; or all POWER/VS jobs whose job name begins with the specified one to seven characters in the reader, list, or punch queue or in all three queues; or all POWER/VS jobs with the specified class in the reader, list, or punch queue.

  A status display can also be requested for the following: all RJE jobs in the reader, list, or punch queue or in all three queues; all RJE jobs in the reader, list, or punch queue that were submitted by or routed to the specified remote user; or all POWER/VS jobs in the reader, list, or punch queue or all three queues that were submitted from or routed to the central location.

  Display can also be used to request a list of all active reader and writer tasks and the POWER/VS jobs and spool files they are currently processing, a list of all the system messages to which the operator has not yet responded, a listing of the current time and

date (includes the current number of fixed pages and POWER/VS tasks) or a list of the free queue file records, track groups, and account file records.

- PALTER, which is used to alter one or more attributes of a specific POWER/VS job, all POWER/VS jobs whose job name begins with the specified one to seven characters, or all POWER/VS jobs with the specified class in either the reader, or list, or punch queue. Changes are made to the read, list, or punch queue entries for the affected jobs. The attributes that can be changed are job priority, disposition, class, number of copies for spool output files, the destination of spool output files created by RJE jobs, and whether or not compaction is to be performed for output sent to a remote SNA workstation.

  Attributes cannot be changed for any POWER/VS job in the reader queue that has been scheduled and is executing. If a spool file is in the process of being transcribed, only its number of copies attribute can be changed (unless the output is being transmitted to a 3790 workstation with PDIR=FMHZ specified). However, if a spool file being transcribed is segmented and the list/punch queue entry for its first segment has already been deleted, no attributes can be changed. Note that the attributes of a spool output file indicated in a LST/PCH statement cannot be changed until the spool file has actually been created (job step that creates the spool file has executed).

- PDELETE, which is used to delete one or more POWER/VS jobs from the reader, list, or punch queue, a specific queued message to all RJE users, or all messages submitted by the central operator for all users. If a job in the reader queue is being processed or a spool file in the list or punch queue is being transcribed, it is not affected by a PDELETE command. A specific POWER/VS job, all POWER/VS jobs, all POWER/VS jobs with the specified class, or all POWER/VS jobs whose job name begins with the specified one to seven characters in a given queue can be deleted with a PDELETE command.

- PRELEASE, which is used to change the disposition of one or more POWER/VS jobs in the hold or leave state in a given queue to dispatchable so that they can be processed. The queue entry of a job originally in the hold state is deleted after it is processed. The queue entry of a job originally in the leave state is returned to the leave state after it is processed. A specific POWER/VS job, all POWER/VS jobs, all POWER/VS jobs with the specified class, or all POWER/VS jobs whose job name begins with the specified one to seven characters in the reader, list, or punch queue can be released. Note that a job with a disposition of leave is not released by a PRELEASE command with the ALL operand or a class operand specified.

The following are POWER/VS miscellaneous commands:

- PBRDCST, which is used by the central operator to send a message to one or all RJE users. Messages generated by the PBRDCST command are stored in a message area in virtual storage in the POWER/VS partition. Messages to all users are given a sequence number that can be determined using the PDISPLAY command and can be a maximum of 46 characters in length. A message from the operator to one terminal can be a maximum of 49 characters in length.

  Broadcast messages are sent to a remote user only when the user requests them via a POWER/VS RJE DISPLAY MSG command. When a message is sent to a remote user, it is deleted from the message area in virtual storage unless it is a message for all users. All-user-type messages can be deleted by the central operator.

A maximum of 255 messages can be stored in the message area in the POWER/VS partition. Of these, a maximum of 16 can be of the all-user type. When the all-user portion of the message area is filled, the next all-user message causes the 16 existing messages to be deleted and a message to be issued to indicate the deletion.

Note that output generated as a result of an RJE DISPLAY command and responses to other commands issued by RJE users are also stored in the message area of the POWER/VS partition. They are sent to the appropriate RJE user as soon as his line becomes available for a write operation.

- PINQUIRE, which is used to obtain status information about one or all POWER/VS RJE lines. The status of an RJE line can be one of the following: PROCESSING, which is applicable to both BSC and SNA users (the remote user has entered a valid SIGNON or LOGON command), INACTIVE, which is applicable to BSC lines only (the central operator has started the line but a user is not currently signed on), NOT INITIATED, which is applicable to BSC lines only (the central operator has not started the line), NOT SUPPORTED, which is applicable to BSC lines only (the line was not specified during POWER/VS generation or is not defined in the PUB table for some reason), NOT LOGGED ON (no SNA user is currently logged on to the specified logical unit), LOGGED ON (an SNA session is logged on but no processing is active), or LOGGING ON (an SNA session is in the process of being logged on).

- PACCOUNT, which is used to delete the account file or write it to a labelled or unlabelled tape, disk of the same type as contains the account file, or cards. When the account file is written to disk or a labelled tape, the label information for the file must be present in the label cylinder.

- PSETUP, which is used to specify one or more setup pages for the purpose of forms alignment. The setup pages have X's in all positions for all characters that are to be printed on the pages. Adjustment can be made during printing of the setup pages. The PGO command is issued to resume normal printing (with the first setup page) when alignment is correct. The PSETUP command can be issued as many times as are required to perform alignment.

Remote Job Entry Support

The optional RJE support provided by POWER/VS enables jobs to be submitted from a remote terminal to a DOS/VS system for execution. Remotely submitted jobs are defined using DOS/VS and, optionally, POWER/VS job control statements and are placed in the single input queue (by priority within job class) that is used by POWER/VS for all job scheduling.

Spooled printer and punch output from remotely submitted jobs is queued by class in the list and punch queues in the same way as is spooled output from locally submitted jobs. The spooled output from a remote job can be sent to the remote user that submitted the job, sent to another remote user, or transcribed to local printers and punches in the DOS/VS system configuration.

Note that when RJE is supported in a POWER/VS system, spooled output from locally submitted jobs can be sent to a remote user. This is accomplished using the remote parameter in the LST or PUN JECL statement.

The terminals supported by POWER/VS RJE are listed in Table 80.40.1 See DOS/VS POWER/VS Installation Guide and Reference, GC33-6048, for the

supported and unsupported devices and features for binary synchronous and SNA terminals.

POWER/VS RJE,BSC. RJE,BSC support can handle up to 25 RJE users/lines. Point-to-point switched and non-switched lines can be used. POWER/VS RJE,BSC uses its own remote terminal access method (RTAM) and does not require the inclusion of a teleprocessing access method in the DOS/VS system with which it is used. The RTAM support in POWER/VS is similar to that used by OS/VS1 RES and HASP RJE.

During POWER/VS generation, one PLINE macro must be specified to define the hardware characteristics (control unit or integrated communications adapter) of each communications line to be supported by POWER/VS RJE,BSC. The PLINE macro specifies the line address, availability of the transparency feature, transmission code to be used, password the user must supply in order to sign on using this line, and a timeout value. The timeout value is the number of minutes the terminal signed on via this line can remain idle before a signoff is forced.

For each PLINE macro specified during POWER/VS generation, a PRMT macro must also be included to define the hardware characteristics of the BSC terminal attached to the line, identify the remote user with a remote identification (1 to 200), and indicate where spooled output from this user's jobs is to be sent. The operator at the DOS/VS installation (central operator) has remote identification 0.

Before any remote users can sign on to POWER/VS RJE,BSC, one or more of the defined communications lines must be started using POWER/VS PSTART commands. RJE lines can be started automatically during POWER/VS initialization or the central operator can start lines any time after POWER/VS has been started.

A PSTART command specifies a line address and causes the specified line to be logically attached to POWER/VS RJE,BSC. The PSTART command can also specify a password that must be presented by the user when signing on via the line. If a password is not specified, the default password defined during POWER/VS generation is used. The PSTART command causes an RJE,BSC read task to be started for the line in the POWER/VS partition and the allocation of one input and one output buffer. Only one RJE,BSC read task is initiated per started RJE,BSC line. An RJE,BSC read task has the standard name RDR suffixed with the line address and a physical identifier for the card reader it is assigned.

The RJE,BSC line manager task is started automatically during POWER/VS initiation and activated when the first RJE,BSC line is started with a PSTART command. This task remains attached until the last RJE,BSC line is stopped, at which time it is detached. The line manager task controls all line activity for the started RJE,BSC lines using RTAM.

Once RJE,BSC lines are started, remote users sign on, sign off, and communicate with the central system and other remote users using POWER/VS RJE,BSC terminal commands. These commands are submitted via the remote terminal keyboards, or card readers, or diskettes in 3741 stations. The POWER/VS RJE SIGNON command is used to logically connect a terminal to POWER/VS RJE,BSC. This command must be the first command submitted after an RJE,BSC line is started.

The SIGNON command identifies the terminal to be used with this line, specifies the remote operator identification assigned to the line during POWER/VS generation, and a password. The password must be that specified in the PSTART command when the line was started, if any, or the password specified in the PLINE macro. If a password is not specified for a line in the PSTART command or PLINE macro, a password need not be specified in any SIGNON command issued for the line.

After an RJE user has signed on, he can submit POWER/VS RJE terminal commands and input jobs (an input stream) via a card reader, keyboard,or the diskette file(s) in a 3741 data station. The 3741 is supported as a 2780 without the multiple record transmission feature.

The input stream submitted via a 3741 can contain fixed-length, unblocked 80-character records on multiple diskettes. Spooled output files transmitted to the 3741 by an RJE,BSC writer task can contain fixed-length, unblocked, 126-character maximum records. The size of a spool output file for a 3741 is limited to the number of diskettes that can be loaded at one time, since diskette unloading and loading are not supported. Therefore, a spool output file for a 3741 is limited to one diskette or, when the second disk feature is installed on the 3741, two diskettes. EBCDIC and ASCII codes are supported for input and output.

RJE terminal commands must be placed between POWER/VS job definitions or they are treated as comments. As soon as the terminal card reader or 3741 is made ready, the input it contains is read by the RJE,BSC read task started for the line. Jobs and data read are placed in the POWER/VS queue and data files and are scheduled using the same rules as are used for locally submitted jobs. Terminal commands in the input stream are removed and sent to the RJE command processing routine for processing.

Output messages generated as a result of commands in the input stream, system messages regarding the input stream, and any messages from the central operator that are generated during the reading of an input stream are stored in the message area in the POWER/VS partition until the complete input stream has been read.

When end of file is reached on the remote input device, reading stops and messages that were generated during input stream reading are sent to the terminal printer or, for a 3741, written to a mounted diskette. The RJE,BSC read task then enters the wait state until (1) the remote input device is again made ready, at which time reading is automatically resumed if the line is available, (2) a signoff is forced by the expiration of the timeout specified for the line during POWER/VS generation, if any, or (3) the central operator issues a POWER/VS PSTOP command to terminate the RJE,BSC read task.

The spooled printer and punch output from the jobs submitted by a remote user is routed to the central operator or the remote user specified for the user/line when the POWER/VS system was generated. This destination specification can be overridden using JECL LST and PCH statements in the remote jobs submitted.

If an RJE user is to receive any output at his remote terminal (other than the messages sent automatically after a job stream has been read), POWER/VS RJE START commands must be issued to start an RJE,BSC list task and/or RJE,BSC punch task for his line. When the writer tasks are started, a remote user can receive broadcast messages (from other remote users), diagnostic messages, and spooled output from the jobs he submitted as well as spooled output routed to him from other remote users that has the class(es) specified in the START writer commands for his line.

System, diagnostic, and broadcast messages for a remote user are displayed on the terminal printer or written to diskette for a 3741 in between the printing of spooled printer files.

One RJE,BSC list and one RJE,BSC punch task can be started for each RJE,BSC line defined. Up to four output classes can be assigned to each RJE,BSC writer task. An RJE,BSC writer task has the standard name LST or PUN suffixed with a line address and a physical identifier for the print/punch/3741 device it is assigned.

When an RJE,BSC writer task is started, it transcribes all the spooled output with the classes it is assigned to handle that is queued for its associated RJE user. At the completion of the transmission of each spool output file to a terminal printer or punch, an RJE,BSC writer task pauses for 6 seconds to allow the remote user to hit the start button on the terminal card reader. This enables the remote user at a 2770, 2780, or 3780 to submit jobs between the transcription of spool output files to his terminal.

When a spool output file has been transmitted to a 3741, the RJE,BSC writer task pauses. At the end of the timeout interval, the 3741 is signed off. The user must sign on again to cause transcription of spool files to resume.

When an RJE,BSC list and an RJE,BSC punch task are active for the same line, any time both tasks have queued spool files to transcribe, the list spool files will be transmitted before the punch spool files.

When no more spool files are queued for an RJE,BSC writer task, it enters the wait state and the line is automatically placed in read mode so that another input stream can be read. For a 2770, 2780, or 3780, card reading begins automatically if the reader is ready. For a 3741 attached via a leased line, a message is displayed on the 3741 screen and the operator can place the 3741 in transmit mode to resume read operations.

After the input stream has been read and any resulting messages have been sent, the line is automatically placed in write mode and a writer task for the line resumes operation if any spool files of its class have been queued for this RJE user during input stream reading. If no spool files are queued, the writer task enters the wait state until additional spooled output with the appropriate class is queued, a signoff is forced by expiration of the timeout interval specified for the line, if any, or the writer task is stopped via a POWER/VS RJE STOP command.

If a remote user wishes to interrupt the transcription of a spool file to his printer or punch, in order to submit a command or another job for example, interruption is accomplished by placing the printer/punch in not ready and then ready status. This causes the line to be placed in read mode and any cards in the terminal reader are then read. If a FLUSH, RESTART, or STOP command for the interrupted writer task is encountered in the input stream, it becomes effective when end of file occurs for the card reader. Otherwise, at end of file the writer automatically resumes transcription with the record after the last one transcribed before the interruption occurred.

Note that when an RJE,BSC list or punch task is interrupted for another reason, such as when an I/O error occurs, the RJE user can also submit a FLUSH, RESTART, or STOP command via the terminal card reader.

While signed on, an RJE user can issue the POWER/VS RJE BRDCST command to send a message to the central operator, a specific remote user, or all users. A message for a specific remote use is displayed on the user's terminal printer or written to diskette immediately if the user is receiving messages (issued a START MSG command). A message to all users is queued in the message area in the POWER/VS partition. Each user must issue the POWER/VS RJE DISPLAY command in order to have an all-users message sent to his terminal.

Note that POWER/VS RJE logs any line errors and timeouts that occur for the lines it supports in the DOS/VS SYSREC file. Therefore, the SYSREC file should have more space allocated to it when POWER/VS RJE is used than would otherwise be allocated.

POWER/VS RJE,SNA. RJE,SNA supports the 3770 and 3790 attached to a 3704/3705 operating in NCP mode using synchronous data link control. RJE,SNA supports up to 200 SNA logical units active concurrently and uses VTAM to handle physical line management. RJE,SNA interfaces with VTAM via the VTAM Application Program Interface and, thus, appears to VTAM as an application program.

RJE,SNA support requires a system with a minimum of 256K of real storage and a DOS/VS supervisor with VTAM and multitasking support (in addition to the other options required by POWER/VS). RJE,SNA in POWER/VS can also include support of BSC (2770, 2780, 3741, and 3780) terminals as well as support of 3770 terminals operating in 2770 or 3780 mode. When both BSC and SNA terminals are supported, of the 200 logical units supported, 25 can be BSC terminals.

The POWER and PRINT macros are used to generate a POWER/VS RJE,SNA system that supports only SNA terminals. The PLINE macro is not required unless BSC terminals are also to be supported. The SNA parameter is added to the POWER macro to request SNA support. It specifies the maximum number of SNA logical units that can be active concurrently.

One PRMT macro must be specified for each SNA terminal to be supported. It specifies the remote identification of the user of the terminal, indicates where spooled output from this user's jobs is to be routed, a password that must be supplied by the user at logon (optional), and the way messages to the terminal are to be handled. If the remote terminal configuration contains a line printer in addition to the standard console printer, specification of CONSOLE=YES causes POWER/VS to print all messages on the console printer immediately after they are generated (interrupting any printer transmission currently in progress) instead of waiting until the current spooled printer file is totally transcribed.

Before a POWER/VS system with RJE support is started, VTAM must be started in a partition and the 3704/3705 units connected to SNA terminals that are to be controlled by RJE,SNA must contain an NCP. A POWER/VS system with RJE,SNA support can then be started like any other POWER/VS system and after it is activated, the central operator must issue a PSTART RJE,SNA command before any SNA terminal user can log on. This PSTART command activates the interface between VTAM and POWER/VS. If BSC terminals are also supported, a PSTART must be issued to activate each line to be used, as for RJE,BSC support (either by the central operator or via the AUTOSTART procedure).

After the interface to VTAM is activated, users of remote SNA terminals must issue the LOGON (or SIGNON) command to logically connect the SNA terminal to VTAM, which then passes to POWER/VS the required logon information. The LOGON command must specify a password if one was specified in the PRMT macro for this terminal, and optionally can specify up to 16 bytes of user information that will be placed in the SNA account record for the session. If the VTAM/USS tailoring services are used to build a USS table, the SIGNON command used to connect to RJE,BSC can be used instead of a LOGON command for a 3770 but not a 3790. The LOGON and SIGNON commands can be entered only via the terminal keyboard.

A 3790 workstation can have a maximum of five logical units defined, each of which can have a session active. Data transmission can occur concurrently on the five logical units. The following operations can occur concurrently: transmission of a card input stream from the logical card reader contained on the disk in the 3791, transmission of three output printer spool files, and transmission of console data (input from or output to the keyboard).

Once a user has connected to RJE,SNA, commands and input jobs (POWER/VS JECL and user data) can be submitted to POWER/VS. For a 3770, commands can be entered via the terminal keyboard, a card reader (except for LOGON, SIGNON, and LOGOFF), or in card image format from a diskette (except for LOGON, SIGNON, and LOGOFF). For a 3790, commands can be entered via a keyboard or the logical card reader, except for LOGON and LOGOFF which must be entered via a keyboard.

POWER/VS JECL and user data from a 3770 can be submitted via a card reader or diskette only. For a 3790, the logical card reader is used. All input from the terminal keyboard is interpreted as remote terminal operator commands.

A remote SNA terminal user can receive spooled printer and punch output only after entering START LST and/or START PCH commands. No spooled output is sent until such START commands are issued. Output to a remote 3770 user can be printed, punched, or written to a diskette. The transmission of spooled printer and punch files (but not messages) to a 3770 terminal can be interrupted by the remote user, if necessary, for the transmission of commands of input jobs via the terminal console, a card reader, or a diskette.

For a 3790, up to three list tasks can operate concurrently. The START command indicates the device to which the spool printer file will be sent. START LST1 and START LST2 commands cause transcription to the two line printers defined as logical printers 1 and 2, respectively. START LST3 causes transcription to the disk in the 3791 for later transcription to a line printer.

Whenever multiple copies are specified for a printer spool file that is to be sent to a 3790, the file is automatically sent to disk even if it is directed to line printer 1 or 2. Thus, only one transmission is required for multiple copies. Printer spool files can be sent to a 3790 in compacted, as well as compressed, form to reduce the amount of data that is transmitted. As for a 3770, a data transmission to or from a 3790 component can be interrupted. The transmission of a printer file can be interrupted in order to enter a terminal operator command or submit input from the logical card reader. Transmission of input from the logical card reader can be interrupted for the sending of a terminal operator command.

Messages for a remote SNA terminal user are written to the standard console printer for the 3770. Messages are transmitted immediately after they are generated if CONSOLE=YES was specified for the terminal. That is, the transmission of a printer spool file is interrupted to send the message. Thus, if an additional line printer is not available for spooled output, messages are interspersed with printed spool output to the standard console printer when CONSOLE=YES is specified.

When CONSOLE=NO is specified for a 3770 terminal, messages received during transcription of a spooled printer file are not transmitted until printing of the file is completed. Any message received during the transcription of a punch spool file causes punching to be interrupted and the message is sent to the console printer.

Messages for a remote 3790 are sent to the operator console device (3793 keyboard/printer or 3277 display)when the console is in a state in which terminal commands can be entered.

A remote SNA terminal user terminates a session by issuing a LOGOFF or SIGNOFF (3770 only) command. A LOGOFF command can specify a conditional or unconditional termination and can be entered only via the terminal keyboard. A SIGNOFF command always causes a conditional terminal logoff and can be entered via the terminal keyboard or a card reader. A conditional logoff request causes VTAM to disconnect the SNA

terminal from POWER/VS after processing of the current job completes. A message is sent to the terminal when the disconnection is completed. An unconditional logoff request causes VTAM to terminate the session immediately without waiting for the completion of any job transmission that is in progress.

The central operator can issue a PSTOP RJE,SNA command with the remote user identification to terminate the session of a remote user. To terminate all RJE,SNA processing, the PSTOP RJE,SNA command without a remote user specification must be issued. The termination of RJE,SNA is immediate unless EOJ is also specified. Termination occurs only after all transcription of the jobs in progress (both input to POWER/VS and spool files being sent to remote users) is completed when EOJ is specified.

Terminal commands. The commands provided for POWER/VS RJE,BSC and RJE,SNA users are a subset of the commands provided for the central POWER/VS operator. There are four types of POWER/VS RJE terminal commands: terminal control commands that are used to stop and start user sessions, task management commands that are used to control RJE writer tasks (but not RJE read tasks), queue management commands that are used to control the processing of remote jobs, and miscellaneous commands that control terminal printer spacing and message handling. POWER/VS RJE terminal commands contain an asterisk, blank, and two periods in the first four positions. A single letter abbreviation can be used for all RJE commands except SIGNON and SIGNOFF.

The following are the POWER/VS RJE terminal control commands:

- SIGNON, which is used to establish a logical connection between a remote terminal and POWER/VS RJE via a specified line. SIGNON is always used for remote BSC terminals and, optionally, can be used for remote SNA 3770 (but not 3790) terminals if the proper VTAM USS table is constructed. A SIGNON command initiates a terminal session that is terminated when the remote user issues a SIGNOFF command, a signoff is forced by expiration of the timeout interval, or another SIGNON command is issued (without an intervening SIGNOFF) to initiate a new terminal session. The central operator is notified when a remote user successfully signs on.

  The SIGNON command specifies the remote identification assigned to the terminal via the PRMT macro, a password (optionally), and up to 16 bytes of user information (optionally), which are placed in the line account record for this session. Note that an invalid SIGNON command is interpreted as a SIGNOFF command and the terminal from which it was received is disconnected if it is on a switched communications line after input in the card reader is flushed.

- SIGNOFF, which is issued to terminate a session at the next end-of-file condition on the card reader. SIGNOFF is always used for remote BSC terminals and can be used for remote 3770 (but not 3790) SNA terminals. POWER/VS logically disconnects the terminal and stops any active writers for the terminal. The central operator is notified when a remote user signs off. A signoff is forced when a terminal has been idle for a number of minutes, as specified during POWER/VS generation.

- LOGON, which is used to establish a logical connection between a remote SNA terminal and POWER/VS RJE,SNA. LOGON must be used instead of SIGNON for a 3790. This command initiates a terminal session that is terminated when the remote user issues a LOGOFF or SIGNOFF (3770 only) command or the central operator issues a PSTOP RJE,SNA command with the user specified. LOGON specifies the APPLID (POWER) parameter to cause VTAM to connect the terminal to POWER/VS, the name of an entry in the VTAM logon mod table that defines the

parameters to be used for this terminal while it operates under POWER/VS, the remote identification of the user that is using the terminal, and optionally, a password and 16 bytes of user information.

- LOGOFF, which is used to terminate a session for an SNA terminal. LOGOFF must be used instead of SIGNOFF for a 3790. The LOGOFF command specifies APPLID (POWER) to cause VTAM to logically disconnect the terminal from POWER/VS and the UNCOND or COND parameter to request an unconditional or conditional termination, as previously described.

The POWER/VS RJE (BSC and SNA) task management commands are the following:

- START, which is used to start an RJE writer task and specify from one to four output classes (A to Z) the writer is to handle. The START MSG command is used to restart the receipt of messages on the terminal printer after a STOP MSG command was issued to stop the receipt of messages.

- STOP, which is used to stop an RJE writer task or the receipt of broadcast and system messages on the terminal printer. If EOJ is specified, the writer task does not stop until it has finished transcribing the spool file it is currently processing and the queue entry for the spool file has been deleted. The writer stops immediately if EOJ is not specified and the spool file remains queued.

  If RESTART is specified instead of EOJ, the writer stops immediately but processing of the spool file currently being transcribed will be resumed at the record after the last one processed before the writer stopped. If neither EOJ nor RESTART is specified, transcription of the spool file being processed when the STOP was issued begins at the beginning of the spool file.

- FLUSH, which is used to terminate processing by a writer task of the spool file it is transcribing. If HOLD is specified, the queue entry for the spool file is not deleted. The queue entry is deleted if HOLD is not specified. In either case, the writer continues processing with the next entry in the class queue, if any. FLUSH can only be issued for a writer task that has been interrupted by the remote user, central operator, or an action-type POWER/VS message.

- RESTART, which is used to terminate processing of a spool file by an RJE writer task and have it resume at a specified point. Processing can be resumed at the beginning of the spool file or up to 9999 pages/cards forward or backward in the spool file. RESTART can be issued only for an RJE writer task that has been interrupted by the remote user, the central operator, or an action-type POWER/VS message.

- GO, which is used to restart an RJE writer task after it was stopped because of a forms mount message. The GO command does not apply to the 3741.

The following are POWER/VS RJE (BSC and SNA) queue management commands (which a remote user can issue only for input jobs he submitted, the spooled output from his jobs, and spooled output from other remotely submitted jobs that is routed to him):

- DISPLAY, which is used to display the status of POWER/VS RJE jobs. A status display can be requested for the following: a specific POWER/VS RJE job in the reader, list, or punch queue; all POWER/VS

RJE jobs in the reader, list, or punch queue or in all three queues; all POWER/VS RJE jobs that are in hold or leave status in the reader, list, or punch queue or in all three queues; all POWER/VS RJE jobs that are dispatchable or in keep status in the reader, list, or punch queue or in all three queues; all POWER/VS RJE jobs whose job name begins with the specified one to seven characters in the reader, list, or punch queue or in all three queues; and all POWER/VS RJE jobs with a specified class in the reader, list, or punch queue; all the ALLUSER messages and their originators; and current time, date, number of fixed pages, and current number of POWER/VS tasks.

- ALTER, which is used to alter one or more attributes of a specific POWER/VS RJE job, all POWER/VS RJE jobs whose job name begins with the specified one to seven characters, or all POWER/VS RJE jobs with the specified class in either the reader, or list, or punch queue. Changes are made to the read, list, or punch queue entries for the affected jobs. The attributes that can be changed are job priority, disposition, class, number of copies for spool output files, and the destination (remote user identification of 0 to 200) of spool output files for RJE jobs.

  Attributes cannot be changed for any POWER/VS RJE job in the reader queue that has been scheduled and is executing. If a spool file is in the process of being transcribed, only its number of copies attribute can be changed. However, if a spool file being transcribed is segmented and the list/punch queue entry for its first segment has already been deleted, no attributes can be changed.

- DELETE, which is used to delete one or more POWER/VS RJE jobs from the reader, list, or punch queue, a specific queued message to all RJE users submitted by this user, or all messages to all users submitted by this remote user. If a job in the reader queue is being processed or a spool file in the list or punch queue is being transcribed, it is not affected by a DELETE command. A specific POWER/VS RJE job, all POWER/VS RJE jobs, all POWER/VS RJE jobs with the specified class, or all POWER/VS RJE jobs whose job name begins with the specified one to seven characters in a given queue can be deleted with a DELETE command.

- RELEASE, which is used to change the disposition of one or more POWER/VS RJE jobs in the hold or leave state in a given queue to dispatchable so that they can be processed. The queue entry of a job originally in the hold state is deleted after it is processed. The queue entry of a job originally in the leave state is reset to leave after it is processed. A specific POWER/VS RJE job, all POWER/VS RJE jobs, all POWER/VS RJE jobs with the specified class, or all POWER/VS RJE jobs whose job name begins with the specified one to seven characters in the reader, list, or punch queue can be released.

The miscellaneous POWER/VS RJE terminal commands are the following:

- BRDCST, which is used in RJE,BSC and RJE,SNA to send a message of up to 40 characters to one remote user, all remote users, or the central operator. A message to a specific user is sent to that user immediately if his terminal is ready to receive messages. Otherwise, it is stored in the message area in the POWER/VS partition. Messages sent to all users are stored in the message area also but are displayed only when a user requests their display via a DISPLAY command. Up to 256 broadcast messages can be stored at one time. If this limit is exceeded, all messages for the remote user with the greatest number of stored messages are deleted.

- INQUIRE, which is used to request status information about one specific or all RJE,BSC lines. A status of PROCESSING indicates a remote user has entered a valid SIGNON command for the line. INACTIVE status is indicated when the line has been started by the central operator with a PSTART command but no user is currently signed on. A NOT INITIATED status indicates the line has not been started by the central operator. The NOT SUPPORTED status indicates the line was not specified during POWER/VS generation.

- SETUP, which is used in RJE,BSC and RJE,SNA to specify one or more setup pages for the purpose of forms alignment. The setup pages have X's in all positions for all characters that are to be printed on the pages. Adjustment can be made during printing of the setup pages. The GO command is issued to resume normal printing (with the first setup page) when alignment is correct. The SETUP command does not apply to the 3741.

Operator Messages

POWER/VS issues three types of messages to the operator: information, decision, and action. No operator response is required for informational messages and processing continues after such messages are issued. Informational messages for POWER/VS RJE users are stored in the message area in the POWER/VS partition and sent when the terminal printer is available. For a 3741, informational messages are recorded on the diskette unless a STOP MSG command has been issued by the remote user.

Decision messages require an immediate reply from the operator and are never sent to an RJE user. The operator console is unlocked after a decision message is issued and the entire DOS/VS system waits for the operator's reply (unless the Advanced Functions-DOS/VS program product is installed).

An action message is issued when an operator action is required. An action message for an RJE user is displayed on the remote printer for the user. When a POWER/VS task issues an action message, it is placed in the wait state. The central operator can restart the waiting task by issuing a POWER/VS PGO command after taking the appropriate action or by issuing a PFLUSH command to discontinue execution of the affected POWER/VS job. An RJE user can issue a POWER/VS RJE RESTART command to continue processing or a POWER/VS RJE FLUSH command to discontinue processing of the affected POWER/VS job.

Virtual Storage Requirements

The virtual partition in which POWER/VS operates is divided into three areas, as shown in Figure 80.40.8. The permanent area in lowest addressed virtual storage in the partition is always 8K bytes. It contains the POWER/VS nucleus and control tables that are not pageable. This area is permanently fixed when POWER/VS is started and is not unfixed until operation of POWER/VS is terminated.

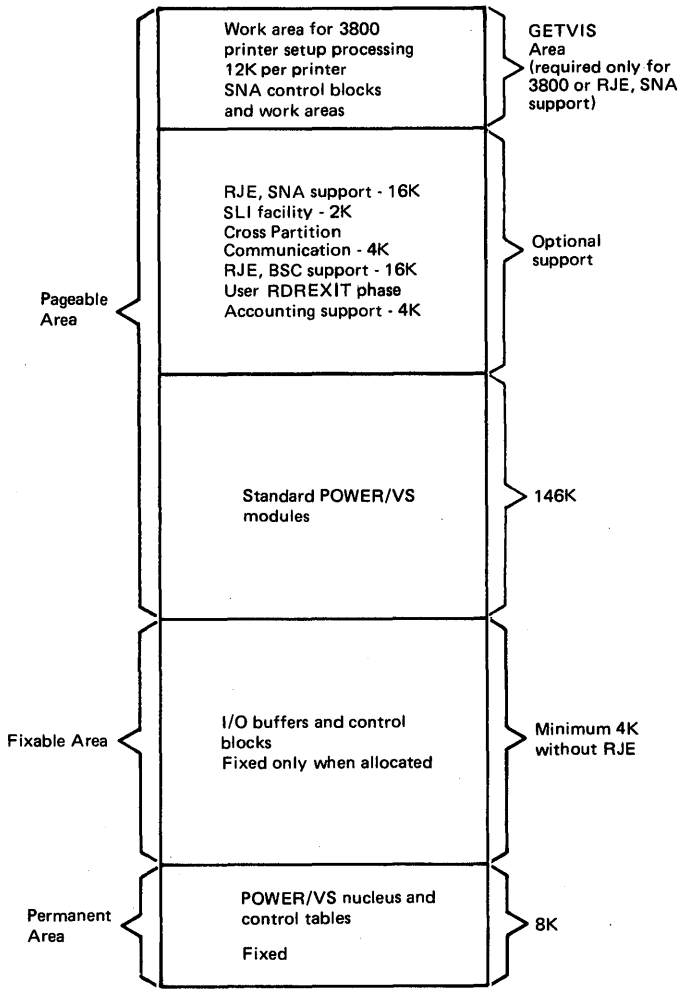


Figure 80.40.8. Layout of the POWER/VS virtual partition

The fixable area is above the permanent area. It contains data buffers and control blocks that are used by POWER/VS tasks. When a POWER/VS task is started, the buffers and control blocks it requires are allocated from the fixable area and permanently fixed. When a POWER/VS task terminates, its buffers and control blocks are unallocated and unfixed.

The size of the fixable area is variable. It depends on the size and number of the buffers to be used, the maximum number of local and remote reader and writer tasks that can be active concurrently, the maximum number of execution processor tasks active concurrently (maximum three per POWER/VS-controlled partition), whether the source library inclusion facility is to be used, whether cross partition communication is supported, the maximum number of RJE,BSC lines to be used, and the buffer size to be used to process the SIGNON command for each RJE terminal. The fixable area is a minimum of 4K bytes when RJE support is not used.

The pageable area in highest addressed virtual storage in the partition contains the POWER/VS code that is pageable. The pageable area requires a minimum of 146K for basic POWER/VS support and well over

200K when all POWER/VS options (that is, source library inclusion, job accounting, cross partition communication, 3800 printer support, RJE,BSC and RJE,SNA) are selected. The inclusion of a user-written reader exit routine increases the requirement for the pageable area. Since the IBM-supplied code in the pageable area is reentrant, a page-out is not required when a page frame allocated to this area is taken for reassignment.

When RJE,SNA or 3800 printer support is included in POWER/VS, a partition GETVIS area is required also. The size of the GETVIS area is 2K plus (1) 12K for each 3800 printer for which setup processing is to be done concurrently plus (2) the requirements for remote control blocks, SNA unit control blocks, and work areas, which vary depending on the number of SNA terminals supported.

The amount of virtual storage that should be allocated to the real partition POWER/VS is to use is the sum of the sizes of the permanent area and the fixable area. This makes the real partition large enough to support the number of permanently fixed pages required when the maximum POWER/VS configuration is active.

The minimum size of the POWER/VS real partition is 10K without RJE support and 14K with RJE support (BSC and/or SNA). The maximum real partition size that POWER/VS will use is 128K. If the virtual storage in the real partition being used by POWER/VS is totally allocated when a buffer must be permanently fixed for allocation to a POWER/VS task, the operator is notified and the task that needs the buffer is placed in the wait state until virtual storage in the real partition becomes available.

POWER/VS Generation

The IBM-supplied system core image library on the DOS/VS distribution volume contains a reader/writer POWER/VS system (with phase name POWER) that was generated using all the default values for the POWER generation macro. This POWER/VS system does not support RJE. If this version is not suitable, the desired POWER/VS system must be defined using the POWER/VS generation macros and a generation must be performed.

One macro (POWER) is provided to generate a non-RJE POWER/VS system. Three macros (POWER, PLINE, and PRINT) must be used to generate a POWER/VS system that supports only RJE,BSC or RJE,SNA with binary synchronous terminals as well. Only the POWER and PLINE macros are required to generate a POWER/VS system with RJE,SNA support without binary synchronous terminals.

The POWER/VS generation macro(s) must be assembled to generate the POWER/VS generation table object module, which also contains the code required to invoke the POWER/VS initialization module. The initialization module loads and initializes POWER/VS program phases. The generation table object module must be link edited to the system core image library. As many generation table phases can be cataloged in the system core image library as the number of different POWER/VS systems required in an installation.

The phase name of the POWER/VS system is specified in the POWER generation macro. If a name is not user specified, POWER is assigned to the POWER/VS system by default.

Advantages Of POWER/VS Over Other POWER Components

The POWER component provided for DOS/VS Releases 28 to 30 supports several functions not supported by the Type III Class A Maintenance

POWER II program for DOS Versions 3 and 4, which cannot be used with DOS/VS. These additional functions are also supported by POWER/VS.

The functions provided by DOS/VS POWER for Releases 28 to 30 but not by DOS Version 3 and 4 POWER are support of up to four partitions, a punch checkpoint/restart facility, command enhancements, the 5425 Multifunction Card Unit, the 2560 Multifunction Card Machine, the 3340 Direct Access Storage Facility, the 3540 Diskette Input/Output Unit, the 3203 and 5203 Printers, print/punch operations on the 3525 Card Punch, the 3780 Data Communication Terminal, and certain optional hardware features for the 2770 Data Transmission Terminal (2203 Printer, Space Compression/Expansion, and Buffer Expansion, Additional). Messages issued to the operator by DOS/VS POWER are modified to follow the standard DOS format for job control messages.

The punch checkpoint/restart facility in DOS/VS POWER provides the capability of stopping a 2560 MFCM or 5425 MFCU during the punching of a file so that the device can be used as a reader. Once read operations have been completed, punch operations on the 2560 or 5425 can be resumed at the point at which they were stopped. Command enhancements provide the following new capabilities:

- A single DISPLAY command instead of multiple commands can be issued to list the entries in all reader, list, or punch queues.
- The FREE operand can be used to list all POWER jobs that are available for execution.
- The LOCAL operand can be used to list all POWER jobs that have been submitted locally.

With the exception of the all-users output class for spooled output from RJE jobs, POWER/VS provides the same functions as DOS/VS POWER as well as many enhancements to both local and RJE support. The enhancements provide new capabilities, improved operational characteristics, and easier installation and can improve system performance.

POWER/VS is designed to take advantage of virtual storage. Hence, more features are included as standard in POWER/VS than in DOS/VS POWER without a corresponding increase in real storage cost. In addition, the fact that the fixed real storage requirements of POWER/VS are less than those of DOS/VS POWER can enable POWER/VS to be used effectively in smaller DOS/VS installations than DOS/VS POWER. In addition, POWER/VS supports SNA devices, which are not supported by DOS/VS POWER.

The most significant new features of POWER/VS local support when compared with DOS/VS POWER local support are the following:

- POWER/VS is pageable and requires less fixed real storage than DOS/VS POWER for like configurations. For a reader/writer system that supports one partition, DOS/VS POWER requires a minimum 20K real partition while POWER/VS requires a minimum real partition of about 10K to 12K.
- Partition-independent input classes are supported by POWER/VS to enable a job to execute in any available partition instead of a predetermined partition only. In addition, there is only one set of input and output queues for the entire system in POWER/VS instead of one set per partition as in DOS/VS POWER. This means the operator can spend less time starting and stopping readers and writers. The job scheduling flexibility provided by POWER/VS combined with the relocating loader, generic device assignment, and partition-related cataloged procedures DOS/VS features simplifies the operation of a multiple batch partition DOS/VS system.

- Output segmentation for spooled printer and punch files is supported directly by POWER/VS and does not require any special techniques or user programming as is required in DOS/VS POWER. Output segmentation can be used to reduce the amount of disk storage required for the data file and the turnaround time of certain jobs.

- The AUTOSTART facility in POWER/VS enables multiple read tasks, list tasks, punch tasks, partitions, and RJE lines to be started automatically instead of only one read, list, and punch task, as in DOS/VS POWER.

- The job accounting support in POWER/VS merges the POWER/VS and DOS/VS accounting data in the POWER/VS account file. In DOS/VS POWER, a user-supplied accounting routine is required to collect DOS/VS accounting data for POWER-controlled partitions.

- Up to eight spooled printers and up to eight spooled punches are supported for each POWER/VS-controlled partition to enable a job step to create multiple print and punch spool files concurrently. DOS/VS POWER supports only one spooled printer and one spooled punch per partition.

- The FCB and UCB image to be used for a printer spool file can be specified in a LST statement to cause loading to be performed automatically by POWER/VS before the transcription of a printer spool file. The SYSBUFLD program must be executed in DOS/VS POWER to load a UCB or FCB as these images cannot be specified in a PRT statement.

- The FCB image being used for a given spooled printer can be changed during execution of a job step in POWER/VS, which enables a job step to serially create multiple reports with different formats. This cannot be done in DOS/VS POWER.

- Additional job disposition attributes are supported in POWER/VS that enable an executed job or a transcribed spool file to remain queued for a rerun or transcription of additional copies at a later time. In DOS/VS POWER, a job or spool file is deleted from the queue file after it has been processed.

- Additional status information is provided by the POWER/VS DISPLAY command that can enable the operator to make more effective decisions during system operation.

- A POWER/VS generation requires only a few minutes and does not involve the link editing of relocatable object modules, as is required in a DOS/VS POWER generation.

The most significant new features of POWER/VS RJE,BSC support are the following:

- Most RJE code in POWER/VS is pageable instead of fixed as in DOS/VS POWER.

- Up to 25 communications lines are supported by POWER/VS RJE,BSC instead of a maximum of five as in DOS/VS POWER RJE.

- Remote job scheduling is improved in POWER/VS. POWER/VS enables an RJE user to easily interrupt the transcription of a spool file to a terminal printer or punch so that a job stream can be sent to the central system. Once the job stream has been read, POWER/VS automatically resumes transcription of the spool file from the point of interruption. From an operational point of view, this type of interruption requires more effort on the part of the DOS/VS POWER

RJE user (stopping of the writer, forward spacing to the point of interruption, and restarting the writer).

- Scheduling of RJE spool file transcription is improved in POWER/VS by the support of hot writers and output classes for spooled RJE output. In POWER/VS, when a spool file is queued for an RJE user, it is transcribed automatically by the started writer for its class and terminal as soon as the terminal is available. This eliminates the need for the RJE user to specifically inquire about the existence of spool files for his terminal as is generally required in a DOS/VS POWER RJE environment.

- The teleprocessing access method used by POWER/VS RJE,BSC is a DOS/VS version of RTAM instead of BTAM, which is used in DOS/VS POWER RJE. RTAM offers several advantages over BTAM. RTAM is mostly pageable instead of mostly fixed like BTAM. RTAM requires less virtual storage (about 8K to 10K less) than BTAM and provides the hot writer support described above.

- The commands available to the POWER/VS RJE user provide the user with more control over his jobs than the commands available to the DOS/VS POWER RJE user. Specifically, ALTER, DISPLAY, and SETUP commands are provided for POWER/VS RJE users.

- The transperency feature is required for a POWER/VS RJE terminal only if object decks are to be punched at the terminal. A DOS/VS POWER RJE terminal that is to punch any cards must have the feature.

- POWER/VS provides accounting data (line account record) for RJE terminals which is not provided by DOS/VS POWER.

- A separate generation of POWER/VS RJE code is not required as for DOS/VS POWER RJE code.

Table 80.40.2 provides a detailed comparison of POWER/VS and DOS/VS POWER features.

Table 80.40.2. Comparison of POWER/VS and DOS/VS POWER features

Feature	POWER/VS	DOS/VS POWER
DOS/VS releases supporting	30 and up	28, 29, and 30
System versions supported		
•Reader/writer - local	Yes	Yes
•Writer-only	No but writer only POWER/VS-controlled partitions can be defined in reader/ writer systems	Yes
•Reader/writer with RJE support	Yes (RJE support for both BSC and SNA devices is provided).	Yes (RJE support for BSC devices only)
DOS/VS supervisor options required	Two partitions, page handling overlap, PFIX/PFREE macros, VIRTAD/REALAD macros, EXCP macro with the REAL parameter, and job accounting if POWER/VS job accounting is to be used. Three partitions and VTAM are required for RJE,SNA support.	Two partitions and BTAM if RJE support is to be used. Note that TP=BTAM must be specified for the DOS/VS supervisor for any DOS/VS POWER system so that channel appendage support is included.
Generating requirements	Short assembly and link edit of macros required to produce a generation table module that initializes POWER/VS as desired.	Assembly of POWER generation macros required. Output used to link edit required POWER modules.
Minimum system real storage requirement for operation	96K (256 for RJE,SNA support)	64K
Mode of operation of POWER program	Virtual only and the corresponding real partition must be defined.	Real only
POWER partition storage requirements	Virtual partition minimum is 158K without RJE. Real partition minimum is 10K, maximum is 128K. These requirements do not vary depending on the number of partitions supported.	Real partition minimums for support of one partition are 18K for a writer-only system, 20K for a reader/ writer system, and 42K for a reader/writer system) with RJE support of one line. An additional 4K is required for each additional partition supported.

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
Reentrant code	Yes, in pageable area	No
Operates as a main task only	Yes	Yes
Number of partitions supported	Always four (or 6) of lower priority	From one to four of lower priority as specified at POWER generation
Mode of partitions supported	Virtual or real	Virtual or real
Maximum number of unit record devices spooled per POWER-controlled partition	One reader, up to eight printers, and up to eight punches. Any logical unit can be assigned to devices that are to be spooled.	One reader, one printer, and one punch. Only the SYSRDR, SYSIN, SYSLST, and SYSPCH logical units (and the other logical units assigned to the same devices as these) are spooled.
Subtasks as well as main task in a POWER-controlled partition can use spooled devices	Yes	No
Input job classes supported for POWER-controlled partitions	Partition-dependent 0 to 4 (or 6) and partition-independent A to Z. Up to four partition independent classes can be assigned to each partition and the same class can be assigned to more than one partition	Partition-dependent 0 to 4 only
Output (spool file) classes	A to Z	A to Z
Input and output queues in the queue file	One reader queue, one list queue, and one punch queue used to schedule all POWER/VS-controlled partitions and writer tasks.	One reader queue, one list queue, and one punch queue are maintained for each POWER-controlled partition.
Maximum number of input jobs and spool files in queue file	Not limited	512
Job priority (input and output)	0 to 9	0 to 9

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
Job dispositions	Input - dispatch and delete, dispatch and keep, hold, and leave in queue. Output - same as input plus write to intermediate tape instead of disk, do not spool, and transfer spool file to reader queue without transcription.	Input - dispatch and hold. Output - dispatch, hold, write to tape instead of disk, and do not spool.
Maximum number of read tasks supported	Limited only by the number of card readers and diskette devices in the system I/O configuration. There is no defined limit on the number of POWER/VS tasks that can operate concurrently.	Limited by the maximum number of spooled devices supported by POWER (26)
Types of readers	Partition independent	Partition dependent and partition independent
Reader exit available	Yes	No
Number of buffers assigned to a read task	For a card input stream, one input unless two is specified in the START command and one output. For a diskette input stream, one input and one output.	Same as POWER/VS
Hot reader capability	Yes	Yes
Source library inclusion facility with modifications via the input stream	Yes	Yes
Device types supported by reader tasks	1442, 2520, 2540, 2501, 2560, 3504, 3505, 3521, 3525, 3540, and 5425	Same as POWER/VS
Maximum number of writer tasks supported	Limited only by the number of printers and punches in the system I/O configuration	Limited by the maximum number of spooled devices supported by POWER (26)

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
Output segmentation for printer and punch spool files	Yes	No
Page separation	Yes - up to nine pages as specified at POWER/VS generation or via a JECL LST statement	Optional - up to two pages as specified at POWER generation only
Card separation	Yes - up to nine cards as specified during POWER/VS generation or via the JECL PCH statement	No
Output limitation for printer and punch spool files per POWER job	Yes - when limit is reached, operator is notified and processing continues. Operator need not respond or can flush/ cancel the job.	Yes - when limit is reached, processing stops and operator must issue command to cancel or continue.
Device types supported by list tasks	1403, 1443, 3211, 3203, 5203, 2560, 3521, 3525, 5425, and 3800	Same as POWER/VS except for 3800
Number of buffers allocated to a list task	One or two input and one or two output	One input and one or two output as indicated in start command.
Device types supported by punch tasks	1442, 2520, 2540, 2560, 3525, and 5425	Same as POWER/VS
Number of buffers allocated to a punch task	One input and one output	One input and one or two output as indicated in start command
FCB and UCB loaded by writer task	Yes if image is specified in the LST statement	FCB and UCB images cannot be specified in the PRT statement.
Forms change during DOS/VS job step execution	Yes, via LFCB macro	No
Forms setup assistance	Yes, via PSETUP command	No PSETUP command. Have to stop and restart printing from the beginning of the file.
Multiple copies of spooled printer and punch files	Yes	Yes

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
Backspacing and forward spacing in a spool file during transcription	Yes for printer and punch files	Yes for printer files only
Submission of jobs to POWER from a partition and receipt of spooled printer output by a partition	Yes	No
Devices supported for intermediate disk storage	2314/2319, 3330-series (all models), 3340/3344 (all models), and 3350 (all modes)	2311, 2314/2319, 3330-series Models 1 and 2, and 3340 (all models)
Queue file record file	Always 152 bytes	From 272 to 520 bytes
Buffer size for I/O operations to data file	544 to 2008	528 to 4000 bytes (except for 2311 which has a maximum of 3625 bytes)
RPS supported for operations to intermediate disk storage	Yes	Yes
Automatic initiation of POWER program	Yes - multiple read tasks, list tasks, partitions, and RJE lines can be started via AUTOSTART facility. This facility is not specified at POWER/VS generation.	Yes - one read, one list, and one punch task can be started automatically. The AUTOSTART facility must be specified at POWER generation.
Warm start	Yes	Yes
Status report printed after POWER termination	Yes, if requested via PEND command	No
Job logging facility when job is scheduled for execution	Yes	No
Job accounting	Requires the inclusion of DOS/VS accounting in the supervisor. POWER/VS and DOS/VS job accounting data are merged into one account file by POWER/VS.	Does not require the inclusion of DOS/VS accounting in the supervisor. POWER accounting data is separate from DOS/VS accounting data. A user accounting routine must be written to obtain DOS/VS accounting data for POWER-controlled partitions.

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
Account file printing.	PACCOUNT command is provided to write account file to tape, disk, or a spooled punch under POWER/VS control.	J command is provided to write account file to tape or a punch.
Job entry control language	Statements supported are CTL, DATA, EOJ, JOB, LST (PRT), PUN RDR, SLI, /*, and /&.	Same as POWER/VS except for LST.
	Statements begin with * $$.	Same as POWER/VS
	Continuation statements are supported. Some parameters are keyword only and some are both positional and keyword.	Continuation statements are not supported. All parameters are positional only.
One character abbreviations for non-RJE command	Yes, except PEND	Yes
HOLD parameter for FLUSH command	Yes	No
CANCEL command to terminate printing of output from a DISPLAY command	Yes	No
Job status attributes displayed by the DISPLAY command on the SYSLOG device	Yes, including forms identification for a spooled printer file, remaining number of pages and copies to be printed, and input class for jobs in the reader queue.	Yes but does not included items listed under POWER/VS.
ALTER command for queued jobs and spool files	Yes including alteration of input class.	Yes but input class cannot be altered.
Dummy devices	Required only when a writer-only partition reads cards directly from a reader, a reader-only partition writes directly to a printer or punch, a partition writes to more than one spooled printer or punch, or a 1442, 2560, or 5425 is used for both reading and punching.	Required whenever POWER controls spooling in more more than one partition.

Table 80.40.2 (continued)

Feature	POWER/VS	DOS/VS POWER
RJE Support of BSC and SNA devices	Yes	No (BSC only)
Number of RJE,BSC lines supported	Up to 25 point to point leased or switched lines	Up to 5 point to point leased or switched lines
Maximum number of RJE,BSC user identifications	200	100
Teleprocessing access method used by RJE,BSC	RTAM	BTAM
RJE,BSC passwords	By line	By terminal and/or user
Hot RJE readers and writers	Yes	No
RJE accounting	Yes in a line account record written at the end of each session	No
USASCII code supported for RJE terminals	Yes	No
Transparency feature for RJE terminals	Required only if punching object decks	Required for any punching
Route RJE spooled output to another user	Yes - to any other RJE user or the central operator. The ALLUSERS class for RJE spooled output is not supported.	Yes to any other RJE user or the central operator.
RJE,BSC line timeout	Limit is established on a line basis.	One limit is established for all lines.
RJE writer classes	Yes	No
Terminals supported by RJE,BSC	2770, 2780, 3780, and 3741 (and 3770)	Same as POWER/VS except for 3741 and 3770
Terminal commands	Subset of central operator commands. SETUP, DISPLAY, and ALTER commands provided to give RJE user more control over own jobs and output.	SETUP, ALTER, and DISPLAY commands not provided.
Generation requirements for RJE support	RJE support generated along with the rest of POWER/VS system.	Separate generation of RJE support is required.

Conversion from DOS/VS POWER to POWER/VS

The following identifies the major areas of incompatibility between DOS/VS POWER and POWER/VS, differences in features provided by both systems, and steps that must be taken to convert to POWER/VS. This information will be of value to DOS/VS users who are currently using DOS/VS POWER and who intend to convert to POWER/VS.

The local support provided by POWER/VS is generally compatible with that provided by DOS/VS POWER. Therefore, little conversion effort should be required to convert from DOS/VS POWER to POWER/VS when the same facilities are used. Additional conversion effort (such as the addition of POWER/VS or alteration of existing DOS/VS POWER JECL statements) is required to utilize some of the new facilities of POWER/VS (such as input class scheduling and output segmentation).

The general items of incompatibility and difference are:

- POWER/VS requires DOS/VS supervisor options that are not required by DOS/VS POWER and does not require BTAM for RJE,BSC support. POWER/VS also has less requirement for dummy devices. These differences may require reassembly of the existing DOS/VS supervisor for use with POWER/VS.

- The ALLOCR statement for the DOS/VS POWER partition should be changed to specify the POWER/VS real storage requirement and an ALLOC statement must be included to define the corresponding virtual partition.

- A new set of messages exists in POWER/VS. Therefore, all run books and user-written documentation for programmers, etc. should be reviewed.

- POWER/VS direct access files are not compatibile in format with DOS/VS POWER direct access files.

- The IBM 2311 Disk Storage Drive is not supported as an intermediate disk storage device.

- Remote (RJE,BSC) users are defined by a remote identification number instead of a name.

- POWER/VS uses its own access methods. Therefore, all users of routines or programs that depend on any access method characteristic of earlier POWER implementations should carefully review their usage under POWER/VS.

- The DOS/VS POWER AUTOSTART feature has been replaced by a new AUTOSTART feature. No decision-type message is issued during POWER/VS initialization, but the system is automatically started according to the control cards in the reader, if SYSIPT is assigned to a unit record device. With these cards any number of reader/writer tasks, RJE,BSC lines, and POWER/VS-controlled partitions as well as RJE,SNA activation can be started automatically. If partition-independent job classes are to be assigned to the partitions that are automatically started, PSTART commands must be modified to specify the classes. Thus, existing startup procedures for DOS/VS POWER may require revision.

- If the POWER/VS reader exit is to be used, the user-written exit routine must be written, assembled, and link edited to the system core image library before POWER/VS is initiated.

- If output segmentation is to be utilized in POWER/VS and the size of the data file is to be reduced, the EXTENT statement(s) for this file must be modified.

- If a user-written accounting routine exists to log DOS/VS accounting data accumulated for DOS/VS POWER-controlled partitions, this routine should be modified to determine whether the partition is operating under POWER/VS control before performing the logging operation. If so, the user accounting routine should not do the logging.

- User-written programs (and sorts) that process the account file written by the DOS/VS POWER accounting routine must be modified to process the new account file record format and contents.

Incompatibilities between DOS/VS and POWER/VS program generation are:

- The number of generation parameters is reduced in POWER/VS because many more features are standard in POWER/VS than in DOS/VS POWER. Thus, the existing POWER generation input must be modified to use POWER/VS generation parameters. The following DOS/VS POWER generation parameters have no equivalent in POWER/VS:

  DIAG
  DISK
  MAXJOBS
  NUMDDKS
  NTRKGP
  ADDITR
  AUTOSTR
  MAXBUFS
  MAXCCB
  MAXRW
  POWPART
  RDRCLOS
  SLI
  TAPE

  The following POWER/VS generation macros have no equivalent in DOS/VS POWER:

  JLOG
  JSEP
  RBS
  RDREXIT
  COPYSEP
  CLRPRT
  MRKFRM

- For POWER/VS RJE, a separate generation is not required for the BTMOD, RJE blocks, RJE block name list, or userid list. The RJE components are generated along with the rest of the POWER/VS system in one assembly.

- The default phase name for the POWER/VS reader/writer system is POWER. The default phase name for the DOS/VS POWER system is FGPSPOOL.

In general, job streams being used with DOS/VS POWER can be used with POWER/VS with little or no modification. Job entry control language differences and incompatibilities are:

- For DOS/VS POWER job streams, the JECL JOB and EOJ statements are required for a job if the JECL PRT or PUN statement is to be used. Only one PRT and one PUN statement per POWER job are permitted. The JECL PRT (or LST) and PUN statements can be included in POWER/VS jobs without the use of JECL JOB and EOJ statements. Multiple LST and PUN statements are permitted in a POWER/VS job.

- JECL statements can be placed anywhere in a DOS/VS POWER job stream. Certain JECL statements cannot be intermixed with data in a POWER/VS job stream. The functions, such as output segmentation, DOS/VS POWER users accomplished by placing JECL statements in data input can be accomplished in other ways in POWER/VS. This can require program modification, such as the addition of the LFCB macro to serially generate multiple reports with different forms control and removal of dummy card reading.

- The new JECL features in POWER/VS are supported only in keyword form. Features supported by DOS/VS POWER and POWER/VS are supported using the positional format in POWER/VS. DOS/VS POWER JECL statements using the positional form and no continuation are accepted by POWER/VS without modification except for userid in the JOB statement and DISP=T in the PRT and PUN statements, which are ignored by POWER/VS. For RJE users, this means all output is routed according to the LSTROUT and PUNROUT specifications in the RMT macro or the REMOTE parameter in the POWER/VS LST and PUN statements.

- If partition-independent job classes are to be used, existing JOB statements must be modified when the default class of A is not to be assigned or CTL statements can be added to the existing job stream.

Operator command language differences are:

- The local POWER/VS command language is a compatible extension of the DOS/VS POWER command language. Corresponding POWER/VS and DOS/VS commands are compatible with a few exceptions (such as the cancel command). The DOS/VS POWER H (hold job entry), L (delete job entry), M (display or alter counter), O (change output destination of a job entry), and Z (diagnostic trace) commands are not supported by POWER/VS. Except for the diagnostic trace, the functions provided by these commands in DOS/VS POWER are provided by other commands in POWER/VS.

- References to partition queues (BGRDR, FQLST, etc.) are invalid.

- The cuu parameter must be specified in all POWER/VS task management commands (the queue parameter is not accepted in POWER/VS). For example, PCANCEL RDR is not valid in POWER/VS. PCANCEL 00C is valid.

- All DOS/VS POWER RJE parameters in the central operator commands are incompatible with POWER/VS RJE parameters

- The priority in the D (display) and R (release) commands must be specified as n in POWER/VS instead of Pn.

- The class in the D (display) command must be specified as an alphabetic character in POWER/VS instead of as CLASSn. A display of time and date is not supported by the PDISPLAY command in POWER/VS.

- The B and T parameters of the D (display) command in DOS/VS POWER are not valid in the POWER/VS PDISPLAY command.

- The priority parameter of the A (alter) command is positional in DOS/VS POWER and keyword in POWER/VS.

- The terminate POWER/VS command PEND has no abbreviation. The abbreviation E is used for this command in DOS/VS POWER.

- A completely new set of RJE terminal commands for remote users is defined for POWER/VS that is incompatible with the RJE terminal commands of DOS/VS POWER. POWER/VS RJE terminal commands are a subset of the local POWER/VS commands. The terminal command

statements in remote job steams being used with DOS/VS POWER must be changed to the corresponding POWER/VS terminal commands. Note also that for POWER/VS RJE support, each remote user must start an RJE list and RJE punch task for his line if he is to receive output. This is not required for DOS/VS POWER RJE.

LINKAGE EDITOR AND LIBRARIAN

The linkage editor program and the librarian programs can operate in a virtual or real partition in DOS/VS. These programs support the same functions in DOS/VS as in DOS Version 4 and are extended to support new features of DOS/VS as required. The linkage editor is modified to produce relocatable program phases that can be loaded by the relocating loader, as discussed in Section 80:25. When the REL parameter is specified on the ACTION statement for a phase, the DOS/VS linkage editor determines whether the phase can be made relocatable by inspecting the way the load address is specified in the PHASE statement.

A phase can be made relocatable if any one of the following formats is used:

- symbol (phase) + or - relocation
- * + or - relocation
- S + relocation
- ROOT

A phase cannot be made relocatable if one of the following formats (which specify an absolute address) is used:

- + displacement or, for a self-relocating phase, +0
- F + address.

When the ACTION statement is used to indicate that the specified address is relative to a specific partition, the beginning virtual storage address of the virtual partition, rather than the real partition, is used to calculate the load address. Phases that are to be executed in a real partition must be self-relocating, or relocatable, or nonrelocatable (absolute phase) and link-edited with the beginning virtual storage address of the real partition as its load address.

The librarian routines are extended in DOS/VS to support the relocating loader and the cataloged procedures facility. The linkage editor is also modified to support the new core image library organization and to no longer use a librarian work area on disk. Two limitations imposed by the size of the librarian work areas in DOS Version 4 are thereby removed in DOS/VS. Phase size is unlimited and an unlimited number of phases can be cataloged in one linkage editor job step.

The linkage editor also supports two additional parameters on a PHASE statement. The SVA parameter identifies reentrant, relocatable phases as being eligible for residence in the SVA, as already discussed. When a PHASE statement contains the SVA parameter, the linkage editor determines whether or not the phase actually is relocatable (by inspecting the load address specified on the PHASE statement). If it is not, the SVA parameter is ignored and the phase is not marked SVA-eligible.

The linkage editor cannot determine whether a relocatable phase is actually reentrant. However, if the phase is not reentrant, a storage

protection program check occurs if there is an attempt to modify the phase while it is executing in the SVA. The only way a phase can be marked SVA-eligible is by link-editing it.

The PBDY parameter can be specified to request alignment of the load point of a phase on a page boundary. If the load point specified is not so aligned, the next higher page boundary is used as the load point.

Whenever the linkage editor places a phase in a core image library, either in catalog or link mode, the $MAINDIR service program is called to perform required core image library directory updating. If a second level directory for the core image library exists in the supervisor, $MAINDIR updates it as required. The linkage editor also calls the new $LIBSTAT service program after a phase is cataloged in order to display the new status of the core image library.

The maintenance and service librarian programs provided in DOS/VS are the new $MAINDIR and $LIBSTAT service programs and all those provided in DOS Version 4 except for $MAINEOJ and $MAINTEJP, which are not part of DOS/VS. Service programs are modified as required to support the new core image library organization and to request execution of $MAINDIR and $LIBSTAT when necessary.

The $MAINDIR program is called by the job control program, the linkage editor, and librarian routines. It performs the following major functions:

- Constructs an SVA during system initialization. This includes placing selected directory entries from the system core image library in the system directory list in the SVA, loading selected phases into the SVA, and writing the contents of the SVA in the page data set.
- Updates the directory of a core image library when required (such as when a phase is added or the library is reorganized)
- Builds and updates second level directories for the system and private core image libraries
- Updates the library descriptor entry in the directory of a core image library or in the link area
- Updates the link area address in an entry in the fetch table

The $LIBSTAT program prints status reports of the SVA and all system and private libraries. It performs the functions of the $MAINEOJ and $MAINEJP service programs that are deleted from DOS/VS. $LIBSTAT is called by the linkage editor after it catalogs a program and by appropriate librarian routines after their execution in order to display the status of the library that was just updated.

The CORGZ librarian program is updated in DOS/VS to provide a library merge function that is useful when a new release of DOS/VS is installed. The CORGZ program can be used to compare the directory entries of a current system library (core image, relocatable, source, or procedure) with directory entries of a new library. When a member of the old library does not have a corresponding member in the new library, the member is automatically placed in the new library.

Note that the merge function of the CORGZ librarian program when performed on the SYSRES volume and most functions of the MAINT librarian program can be performed only in the background partition. Only the delete and renaming functions of the MAINT program can be executed in a foreground partition and only for a private core image library.

The PDZAP program, not provided in DOS Version 4, is provided in DOS/VS to enable the operator to display and make changes to a phase in a system or private core image library. PDZAP can execute in any partition. It can be invoked from the SYSLOG device or via job control in the SYSIPT card reader. The changes to be made must be supplied via the device used to invoke PDZAP. Up to 16 bytes of data in a phase can be displayed and/or changed at a time.

UTILITIES

The System Utility Programs component is a standard facility of DOS/VS. The utility programs supplied in this component and the minimum size real partition required for their execution are listed in Table 80.40.3. System utility programs can operate in virtual or real mode. Various initialization and copying functions for the tape and disk units supported in DOS/VS and the 3540 Diskette Input/Output Unit are provided by these utilities. All the utilities that are part of the System Utility Programs component (370N-UT-491) for DOS Version 4 are also provided for DOS/VS and the same functions are supported.

Unit record, tape, and disk I/O devices that are supported in DOS/VS but not in DOS Version 4 are supported by the DOS/VS system utilities. In addition, the IEBIMAGE, Copy and Restore Diskette, Fast Copy Disk Volume, Deblock, and Page Data Set Dump, Backup and Restore System, Copy File and Maintain Object Module, Maintain System History, and List System History system utilities are provided for DOS/VS. A Print Hardcopy File utility is also provided to print the hard copy disk file that is created by DOC support for a Model 115, 125, 138, or 148 console or 3277 display and the Analysis Program-1 utility is provided to support problem determination functions for 3344 and 3350 volumes.

IEBIMAGE

The IEBIMAGE utility is provided to create and maintain printer control phases in the system core image library. It handles the following types of phases:

- Forms control buffer for printers that have an FCB (3800, 3211, 3203, and 5203)
- Copy modification for 3800 printers
- Character arrangement table for 3800 printers
- Graphic character modification for 3800 printers

Fast Copy Disk

The Fast Copy Disk Volume utility provides a fast copy facility for 3330-series, 3350, and 3340/3344 disk storage volumes. It can be used to copy the entire contents of one 3330, 3340, or 3350 volume to another 3330, 3340/3344, or 3350 volume, respectively, or to dump the entire volume to tape. The tape can then be used in a restore operation. This utility copies an entire volume only, which can contain any combination of DOS data formats and organizations. This utility cannot be used to copy only a portion of a 3330, 3340/3344, or 3350 volume.

The Fast Copy Disk Volume utility is provided in two versions. One version of the program is standalone and card-resident. This version must be loaded from a card reader. The other version runs as a problem program under DOS/VS control in a virtual or real partition. The Fast Copy Disk Volume utility is designed to provide a faster copying

facility for 3330-series, 3340/3344, and 3350 volumes than the Copy and Restore Disk Utility that operates under DOS/VS control.

Table 80.40.3. List of system utility programs and minimum real partition sizes

Assign Alternate Track Data Cell-24K
Assign Alternate Track Disk-10K
Backup System-16K
Clear Data Cell-14K
Clear Disk-24K (less for non-3350 devices)
Copy Disk to Card-10K
Copy Disk to Disk-10K
Copy Disk or Data Cell to Tape-10K
Copy/Restore Diskette-20K
Copy File and Maintain Object Module (OBJMAINT)-48K
Deblock-16K
Fast Copy Disk Volume-30K (less for non-3350 devices)
Initialize Data Cell-14K
Initialize Disk-14K (24K for a 3350)
Initialize Tape-10K
List System History (HISTLIST) - 64K
Maintain System History (PTFHIST)-64K
Print Hardcopy File (PRINTLOG)-6K
Restore Card to Disk-10K
Restore System-36K
Restore Tape to Disk or Data Cell-10K
VTOC Display-14K

Deblock

The Deblock utility, which operates under DOS/VS control, is primarily designed to deblock, block, list, or copy a DOS/VS system distribution tape or disk volume. This utility performs the following:

- Reads a tape or disk volume with 80-byte records contained in 3440-byte blocks and writes unblocked 80-byte records to tape or disk or punches cards (deblock function)
- Reads an unblocked tape that contains 80/81-byte records or card input and writes 80-byte records in 3440-byte blocks on tape or disk (block function)
- Reads a tape or disk volume with 80-byte records contained in 3440-byte blocks and prints the records (list function)
- Reads cards or 80/81-byte unblocked tape records and punches cards or writes 80-byte unblocked tape or disk records (copy function). The copy function can also be used to read 80-byte records in 3440-byte blocks on tape or disk and write 80-byte records in 3440-byte blocks on tape or disk.
- Reads a tape or disk volume with 80-byte records contained in 3440-byte blocks and writes user-selected records to tape or disk in 80-byte unblocked format or punches the selected records.

The card readers, punches, tape units, and disk units that can be used with the Deblock utility are all those supported by DOS/VS data management.

Backup and Restore System

Backup System and Restore System are two separate utility programs that produce a device independent backup tape of the system and/or private DOS/VS libraries and permit restoration of this copy to all direct access devices supported by DOS/VS as SYSRES. The programs have the following possible applications:

- The Backup program alone can be used to create a backup copy on tape of a DOS/VS system for future use.
- The Restore program alone can be used to restore such a backup copy of the DOS/VS system to disk.
- The Restore program alone can also be used to restore the DOS/VS distribution tape to disk prior to system generation.
- Backup and Restore together can be used to transfer DOS/VS libraries from one type of disk device to another type.
- Backup and Restore together can also be used to condense the DOS/VS libraries.

These programs do not support the IBM 5425 Multi-Function Card Unit. The IBM 2311 Disk Storage Drive may be used to store temporarily the stand-alone version of the Restore program.

The Backup System utility operates in a partition under DOS/VS control. It can be used to:

- Create a backup copy on tape of the system files and of private libraries in a format suitable for later use with the Restore utility.
- Create a stand-alone backup tape, that is, a tape complete with the necessary supervisor, job control, and restore programs for restoring the files and libraries without the use of DOS/VS.

The Restore System utility operates in a partition under DOS/VS control. A standalone version of this utility can be created on tape. The Restore System utility can be used to:

- Restore a DOS/VS system from a tape created by the Backup program.
- Restore the DOS/VS distribution tape to disk.

Copy File and Maintain Object Module

The Copy File and Maintain Object Module (OBJMAINT) program is a multi-purpose utility with three major functions: file-to-file copying of card image files (of particular importance to cardless system and diskette users), maintenance of programs in object format, and processing of PTFs.

File-to-file utility functions that can be performed with OBJMAINT include:

- Copying of card image files from card, tape, diskette, and sequential disk.
- Blocking and deblocking of files on tape or sequential disk.
- Selection or exclusion of specific jobs while copying a SYSIN file.

- Listing of data in 80/80 format (one line per logical input record), including normally unprintable characters and JOB statements along with the total count of statements in each job of a SYSIN file.

The object program maintenance function provides for updating object modules and phases. The object modules may be SYSPCH output from language translators or punched from the relocatable library (using RSERV). The phases may be punched from the core image library (using CSERV).

IBM provides program temporary fixes (PTFs) to correct errors in IBM supplied programs. These PTFs normally consist of object modules (including user REP statements containing the code for updating the module) that will replace existing modules on the relocatable or core image library. The desired PTFs, which normally are included in a PTF file in blocked SYSIN format, must be deblocked prior to application. A PTF may be updated in the same manner as other programs in object format.

Functions provided for maintenance of both object modules and PTFs include:

- Selective copying of object modules or PTF jobs
- Deblocking PTFs from a blocked PTF file
- Selective updating of object modules, phases, or PTFs via user REP statements
- Removal of previously added user REP statements
- Expansion or truncation of a control section within an object module
- Combined object module expansion or truncation and user REP addition within the same job step
- Comprehensive listing options, such as listing of normally unprintable characters (for example, EDECKs), suppression of listing of non-object module jobs, formatted listing of TXT or RLD statements, listing of a TXT statement on a single line, listing of JOB statements with a total count of statements within each job, and 80/80 listing of all statements.

Maintain System History

Histories of installed PTFs will be maintained in the DOS/VS system. The system history is kept in the form of two books in the source statement library, namely: the system control program history (Y.PTFSCP) and the program product history (Y.PTFPP). The Maintain System History (PTFHIST) utility is designed to simplify this maintenance and performs the following tasks:

- Selects specified PTFs from a PTF file.
- Generates job control statements to punch a backout PTF. A backout PTF consists of control statements that can be used to remove the PTF at a later time, if this should be necessary, and to restore the libraries to their pre-PTF condition.
- Generates job control statements to update the system history.
- Lists the PTF index of a master file or the job control statements within a PTF file.

The program is also able to process summary PTFs. These are distributed in the form of three private libraries (CL, RL, SL) containing all of the macros, modules, and phases of these PTFs. In addition, the private source statement library contains a book with job control statements. This job stream contains one job for each PTF. Each job contains the comment statements and the COPY statements required to merge the modules for the PTF into the system library.

List System History

The List System History utility performs the following:

- Lists the contents of the history books Y.PTFSCP and Y.PTFPP, which are generated and maintained by the Maintain System History utility, or of any other history book with the same format.
- Provides edited and sorted cross-reference lists of APAR, local fixes, PTFs, and affected library members, with pointers to entries in the book printout.
- Provides an edited list of lost APARs and an error report

Analysis Program-1 (AP-1)

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 DOS/VS 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 a core image library. It can be invoked via the job stream or the operator console. The ASSGN SYS000 statement specifies the logical volume to be tested and the UPSI statement indicates the test to be performed. AP-1 can operate only in virtual mode.

SORT/MERGE PROGRAMS

The DOS Sort/Merge 5743-SM1 and DOS/VS Sort/Merge 5746-SM2 program products and the DOS Sort/Merge (360-SM-483) Type I program can be used for sorting and merging operations in DOS/VS. The DOS/VS Sort/Merge operates only under DOS/VS control and does not operate under DOS Version 4 as do the other two sorts. The DOS/VS Sort/Merge provides facilities that are not supported by the other two sorts and is designed to provide better performance in virtual mode than these two sorts. The performance of the DOS/VS Sort/Merge and the DOS Sort/Merge (5743-SM1) is approximately the same when they operate in real mode. The DOS/VS Sort/Merge requires a minimum of 32K bytes if it operates in a real partition. The DOS Sort/Merge (5743-SM1) can operate in 10K.

The performance increase that can be achieved in virtual mode results from the use of private CCW translation by the DOS/VS Sort/Merge program. In order for the sort/merge program to do its own CCW translation, an associated real partition of at least 10K bytes must be defined for the virtual partition in which the sort/merge is operating. In addition, the DOS/VS supervisor being used must include support of the following macros: PFIX, PFREE, VIRTAD, REALAD, and EXCP with the REAL parameter. If any of these requirements are not met, the DOS/VS Sort/Merge will not perform CCW translation and this function is handled by the channel program translation routine of the channel scheduler as usual.

The DOS/VS Sort/Merge 5746-SM2 program product provides the following facilities that are not available in the DOS Sort/Merge program product:

- Support of 3340/3344 and 3350 (in 3330 Model 1 compatibility mode) Direct Access Storage as an input, output, and intermediate work file. Tape work files are not supported; however two different disk device types can be used for work files and a mixture of tape and disk devices is permitted for input to a sort or merge operation. Both the DOS/VS and the DOS sort/merge program products also 2314/2319 disk storage, and 3330-series Models 1 and 2 disk storage for input, output, and intermediate work files. The DOS/Sort/Merge also support 2400- and 3400- series tape for work files and all work files must be of the same type.

- Support of rotational position sensing for 3330-series Model 1 and 2, 3340, 3344, and 3350 disk devices if RPS support is included in the DOS/VS supervisor being used.

- Split cylinder input and output SAM files are supported.

- Support of VSAM files as input to and output from sorting and merging operations.

- An improved disk sorting technique is implemented that reduces work file space requirements and provides performance improvements for presequenced files.

- Under certain conditions, sorting can be performed in processor storage, regardless of whether work files have been allocated.

- Reentrant code is used in most of the program modules. These can therefore be stored in the SVA increase system performance.

- Subtasking of the program is allowed, that is, more than one sort program can be used concurrently in a partition.

- The support required by the merge facility of the DOS/VS COBOL Compiler and Library Release 1 program product. A user-written routine can be incorporated in the DOS/VS Sort/Merge that reads the input files that are to be merged.

- Four new control statements that can be used to specify (1) selection of the records that are to be included in the sort or merge operation, (2) reformatting of input records such that only selected portions of each sort or merge input record are placed in the output record, (3) summary sorting, and (4) the user-defined collating sequence that is to be used. When summary sorting is requested, user-specified summary fields in records with equal control fields are added and the sums are placed in one of the records, which is called the summary record. The other records are deleted.

Except for support of 2311 Disk Storage for intermediate work files, the DOS/VS Sort/Merge provides the same facilities as the DOS Sort/Merge (5743-SM1) and is compatible with it. (The 2311 is supported only as an input and output device by the DOS/VS Sort/Merge.) Control statements and user-written exit routines that are used with the DOS Sort/Merge (5743-SM1) program can be used with the DOS/VS Sort/Merge program without modification except when they specify 2311 disk storage for intermediate work files.

INTEGRATED EMULATORS

The integrated emulator programs for DOS/VS Release 34 are not distributed with the DOS/VS system and must be ordered separately. DOS/VS supports integrated emulation of the same systems as DOS Version 4: Model 20, 1401/1440/1460, and 1410/7010. All the integrated emulators can execute in virtual or real mode in DOS/VS. The integrated emulator programs that are provided for DOS/VS support the same functions as the emulator programs that are provided for DOS Version 4.

In addition, the emulators for DOS/VS can use System/370 I/O devices that are supported in DOS/VS but not DOS Version 4. Specifically, the 3340 Direct Access Storage Facility and 3344 Disk Storage are supported by the Model 20, 1401/1440/1460, and 1410/7010 emulator programs for emulation of Model 20 or 1400-series disks. The 3203 and 5203 Printers are supported by the Model 20 and 1401/1440/1460 Emulator programs for emulation of Model 20 or 1400-series printers. The 3420 Magnetic Tape Unit Models 4, 6, and 8 at 6250-BPI, as well as 1600-BPI, density are supported by the Model 20, 1401/1440/1460, and 1410/7010 emulators as emulation devices.

The functions supported by the integrated emulators 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 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/360 Users (GC20-1781)

The minimum virtual storage requirement for a 1401/1440/1460 or 1410/7010 emulator program that operates under DOS/VS is slightly higher than the minimum required for these emulators when they operate under DOS Version 4. Listed below are the approximate minimum virtual storage requirements for 1400/7010 emulators operating in a DOS/VS environment. The minimum size of the partition required for emulation operations is the sum of the emulator size listed below, the storage size of the 1400/7010 system being emulated, and the buffers used.

Emulated Operations	Minimum Emulator Program Size
1401/1440/1460 unit record only	20K
1401/1440/1460 unit record and 6 tapes	26K
1401/1440/1460 unit record, 6 tapes, 4 disks	30K
1410/7010 unit record and 6 tapes	26K
1410/7010 unit record, 6 tapes, 4 disks	36K

One new facility is provided in DOS/VS for Model 20 emulator users. The Model 20-DOS/VS Disk Interchange program is offered as a transition aid. This program can be used to convert direct access files that are in Model 20 emulator format to System/370 DOS/VS format. The reverse can also be done as long as the Model 20 emulator format extents are initialized by a Model 20 job step running in emulation mode prior to the conversion run. Sequential, indexed sequential, and direct organization files can be converted.

This program enables a file to be processed by Model 20 programs in emulation mode and by DOS/VS programs. It also provides the capability of converting Model 20 emulator format files to DOS/VS format files when an installation converts from Model 20 emulation operations to native System/370 mode operations.

In addition, performance options are available in the Model 20 emulator for DOS/VS. The read ahead option can be specified for 2501 cards readers and/or tape units. This causes the emulator to use two buffers for I/O operations to tape and 2501 units instead of only one so that I/O operations can be overlapped with processing. To further improve performance, 1442 card punch and printer operations have been speeded up in the DOS/VS version of the Model 20 emulator.

If the read ahead feature is specified for the 2501 card reader, column binary reading cannot be performed on the 2501 unless the read ahead option is cancelled for this execution of the Model 20 emulator. Also, any Model 20 program that permits correction of a card just read cannot be emulated when the read ahead option is active for 2501 readers.

When the read ahead option is specified for tape units, the device independence option cannot be included in the generated Model 20 emulator (or vice versa) and the checkpoint function should not be used.

80:45 ADVANCED FUNCTIONS-DOS/VS PROGRAM PRODUCT

The Advanced Functions-DOS/VS program product provides the following:

- Support of a maximum of seven, instead of five partitions
- Dynamic partition balancing
- Asynchronous operator communications
- Faster linkage editing
- Device independence for private source statement and relocatable libraries
- DOS/VS-VM/370 Linkage facility

The Advanced Functions program product requires a DOS/VS Release 34 supervisor as a base. It is distributed in a private source statement library and a private relocatable library. Thus, the functions provided can be added to an existing DOS/VS system (by link editing the relocatable modules desired to the system core image library) or included when a supervisor generation is performed to tailor a DOS/VS supervisor.

All the facilities of the Advanced Function-DOS/VS program product can be included in a DOS/VS system. However, inclusion of the following facilities is optional: support of up to seven partitions, dynamic partition balancing, asynchronous operator communications, and the DOS/VS-VM/370 Linkage facility. A DOS/VS system with the Advanced Functions-DOS/VS program product installed must execute in a processor with a minimum of 96K.

Details regarding the operation, installation, and storage requirements of the Advanced Functions-DOS/VS program product are contained in Advanced Function-DOS/VS, System Information, SC33-6041.

SUPPORT OF UP TO SEVEN PARTITIONS

For a DOS/VS system that is resident on a 2314/2319, 3330-series, or 3350 volume, a maximum of seven partitions can be supported. When the system residence volume is a 3340 or 3344, a maximum of six partitions are supported. The additional partitions are F5 and F6 and can be utilized in the same manner as the other foreground partitions. When support of these additional partitions is included in a DOS/VS supervisor, POWER/VS support can be used to handle data transcription and job scheduling for the additional partitions.

DYNAMIC PARTITION BALANCING

Dynamic partition balancing provides for the dispatching of a set of user-specified partitions according to their operating characteristics, more CPU-oriented or more I/O-oriented, instead of according to a dispatching priority assigned to the partitions. The two or more partitions that are to be dispatched according to dynamic partition balancing rules can be specified at system generation or at any time during system operation using the PRTY command. The PRTY specification overrides any system generation or any previous PRTY command specification.

The PRTY command specifies those partitions that are to be dynamically dispatched and the dispatching priority of this group relative to the partitions that are not in the dynamically dispatched

group. The CPU utilization of each partition in the dynamically dispatched group is monitored each time it is dispatched. The partitions with higher CPU utilization are dispatched after partitions with lower CPU utilization. Since CPU utilization is constantly monitored, the dispatching priorities of the partitions in the dynamically dispatched group vary as the execution characteristics of the partitions vary.

Dynamic partition balancing can be utilized to attempt to balance CPU utilization among partitions whose operating characteristics are unknown or vary widely. This facility can be used to prevent one or more CPU-oriented programs that are executing in high priority partitions from monopolizing CPU utilization. The clock comparator and CPU timer are required for dynamic partition balancing (an optional feature only for Models 135 and 145).

ASYNCHRONOUS OPERATOR COMMUNICATIONS

When the asynchronous operator communications facility is installed, the operator need not respond to messages that require a response in the sequence in which the messages are written to the console and can enter commands in between responses to messages. In addition, when one partition has outstanding a message that requires a response, DOS/VS does not prevent other partitions from issuing a message that requires a response. Each task in the system can have one message requiring a response outstanding at a time.

To support asynchronous operator communications, a reply identifier is assigned to each message that requires a response and written to the console along with the message. When entering a reply, the operator uses this reply identifier to indicate the message to which he is responding. The REPLID command is provided to enable the operator to request a rewriting of the outstanding messages to the console.

Asynchronous operator communications support is applicable to all the console types supported by DOS/VS. In addition, this support enables a Model 158 display console to be utilized in display mode (that is, as a 3277 display). However, light pen support and support of the 3213 printer as a hard copy printer are not provided for a Model 158 display console operating in display mode.

Asynchronous operator communication support eliminates the need for an operator to respond to a message before another command is issued as well as the delay partitions can experience waiting for the operator to reply to a message.

FASTER LINKAGE EDITING

The linkage editor provided by the Advanced Functions-DOS/VS program product reads multiple directory records per read request (EXCP macro) when searching for a specific entry (instead of reading one directory record at a time) and uses two buffers instead of one to overlap processing and reading. This improved directory lookup implementation can reduce linkage editing time. The amount of reduction experienced depends on the number of relocatable modules that are to be included in the resulting phase, the size of the relocatable library, and how far the required directory entries are from the beginning of the directory.

DEVICE INDEPENDENCE FOR PRIVATE SOURCE STATEMENT AND RELOCATABLE LIBRARIES

Private source statement and relocatable libraries for a DOS/VS system with Advanced Functions-DOS/VS installed can be placed on direct access device types that are different from the types used for the system source statement and relocatable libraries. This flexibility eliminates the need to copy existing source statement and relocatable libraries when the device type of the system source statement and relocatable libraries is changed and enables private source statement and relocatable libraries to be portable from one DOS/VS installation to another (as long as the required direct access device types are present).

DOS/VS-VM/370 LINKAGE FACILITY

This linkage facility is designed to improve the performance of DOS/VS system when it executes in a virtual machine by reducing CPU utilization. It also provides operational enhancements. The following are supported by the DOS/VS-VM/370 Linkage facility:

- Nonpaged mode of operation for the DOS/VS system. By eliminating the paging that normally is performed by DOS/VS, the need for a paging device is eliminated and the execution of code in DOS/VS that is redundant in a VM/370 environment is avoided. The DOS/VS supervisor functions that are completely or partially avoided are paging, load levelling, page fixing and freeing, handling of supervisor calls, CCW translation, and seek separation.

- Pseudo Page Fault Handling. When this capability is enabled via the appropriate VM/370 SET command, CP does not place the entire DOS/VS virtual machine in a wait state when a page fault occurs for an executing task. Instead, CP initiates the required page-in operation and returns control to DOS/VS to enable it to dispatch the next ready task. This capability will be of the most benefit to DOS/VS systems with multiple operating partitions and those that use multitasking.

- BTAM Dynamic CCW Modification. This capability reduces the CP processing required to handle BTAM autopoll channel programs. The PCI interruption and processing that CP otherwise utilizes to test for the modification of virtual BTAM autopoll channel programs is eliminated

- CP CLOSE Support. This support provides for the automatic closing of spooled printer and punch files created by POWER/VS at the end of each POWER/VS job. For Assembler Language programs that do not execute under POWER/VS, the CPCLOSE macro can be used to close spooled files. This support eliminates the need for the operator to close spool files.

- Improved Job Accounting. To improve the accuracy and repeatability of DOS/VS job accounting in a virtual machine, CP updates the appropriate timing facility before DOS/VS is given control to perform job accounting. CP also gains control immediately after a timing facility is changed by DOS/VS so that CP can record the change. This capability of the VM/370 Linkage facility is not required for a Model 135 Model 3, 138, 145 Model 3, or 148, as it is automatically provided for these models.

  Note that a DOS/VS supervisor with the DOS/VS-VM/370 Linkage facility installed cannot execute in a real machine. It must execute in a virtual machine under VM/370 control. In addition, the shared virtual area must be created after each IPL of a DOS/VS

system with the VM/370 linkage facility (an existing copy of the shared virtual area cannot be utilized). VM/370 Release 4 (or later) is required to utilize the DOS/VS-VM/370 Linkage facility.

80:50 DOS VERSION 4 TO DOS/VS TRANSITION

Since DOS/VS is designed to be upward-compatible with DOS Version 4, conversion from DOS Version 4 to DOS/VS should involve minimal conversion effort. The amount of work required to install DOS/VS depends to a degree on the new optional DOS/VS features used (relocating loader, cataloged procedures, for example). Transition from a one-partition to a multiprogramming environment requires additional system planning, as would be the case regardless of the DOS version being used.

Installation personnel should become familiar with the additional facilities and new environment offered by DOS/VS. System programmers must become acquainted with the new interfaces to DOS/VS (PFIX/PFREE and GETVIS/FREEVIS and other new macros, for example) that are to be used. Operators must learn how to use the required new operator commands and how to respond to new system messages, such as those related to paging operations and real storage assignment.

Application programmers should learn how to use program structuring techniques that are designed to enhance program performance in a paged environment. System designers must become familiar with the new factors that affect system performance in a DOS/VS environment so that the system can be designed and operated in a manner that will achieve the results desired.

Once the DOS/VS environment to be supported has been defined, a system generation procedure similar to that required for a DOS Version 4 supervisor can be performed to generate the desired DOS/VS supervisor. The system generation macros for DOS Version 4 and DOS/VS are the same for like functions, with a few exceptions. Additional system generation macros and parameters are provided for DOS/VS to describe its new features.

The DOS/VS distribution system is distributed on one or more disk packs or one magnetic tape reel for restoration to one or more disk volumes. The distribution volume contains the DOS/VS system in a SYSRES file that contains a core image, relocatable, source, and procedure library. On a tape reel, the SYSRES file is preceded by an initialize disk and a restore tape to disk program that are used to restore the SYSRES file to disk.

Significant changes to the system generation procedure have been made for DOS/VS that reduce considerably the time required to perform an initial generation or release-to-release generations. Improvements to the DOS/VS system generation procedure (not available for DOS Version 4) are the following:

- A system generation (including the restore function) can be performed in any batch partition concurrently with normal production processing in other partitions in a multiprogramming environment. In an installation with an operational DOS/VS system, this eliminates the need for dedicating system time to the system generation function. Production work need not be interrupted until the new system is to be IPLed.

- Two device-independent backup/restore utility programs are provided. They enable system and/or private libraries to be dumped to tape (with automatic condensing being performed) and restored to disk. Dumped libraries can be restored to a disk device type different from that of the disk from which they were dumped and library size can be increased or decreased during the restore.

- All IBM-supplied system components are pre-linked in the system core image library to eliminate this step during system generation. New

deletion procedures are provided to delete unwanted system components.

- Seven preassembled supervisors, named $$A$SUP0 through $$A$SUP6, are supplied in the core image library. Each contains relocating loader support. If one of these supervisors is suitable for the installation, the supervisor generation procedure (system generation macro preparation and generation process) is eliminated. The source statement library contains the source code for each of the supplied supervisors in the A.Sublibrary. The MAINT librarian program can be used to tailor one of these supervisors when necessary. A supervisor generation can then be performed to obtain a tailored supervisor.

- The distribution system contains preassembled I/O modules for the RPG II and PL/I Optimizer Compiler language translators and preassembled UCS buffer load modules for 1403-N1 train arrangements in the relocatable library.

- A new function is provided that performs an automatic merge of the existing system and private libraries into the new system and private libraries. The directories of the existing system and private libraries are compared with the directories of the new system and private libraries. Modules in the old directories that have no match in the new directories are automatically merged into the new library.

  This function enables user-written programs and program products to be merged into the new system without the necessity of user preparation of the required control cards and ensures that back-level IBM-supplied system components are not accidentally included in the new system.

- The distribution system is shipped with PTFs preapplied, eliminating the necessity to apply them at the installation.

A new feature of the DOS/VS generation process is the installation verification procedure (IVP), which is designed to be performed after the DOS/VS supervisor is generated. The verification procedure involves an IVP generation step and an IVP execution step. During the IVP generation step, a tailored IVP job stream is produced based on the user-supplied input that describes the system environment the generated DOS/VS supervisor is designed to support. The generated IVP job stream is then executed under control of the generated DOS/VS supervisor.

The function of the IVP is to exercise the generated SCP system components to the degree that general operation of the DOS/VS operating system and support of the system hardware configuration specified is assured. Optional SCP features that are tested by IVP if present in the generated supervisor are procedure cataloging, emulator programs (all), OLTEP, and POWER/VS.

The first time VTAM is included in the generated supervisor, the network control programs to be used in the 3704/3705 Communications Controllers supported by VTAM must be generated and placed in a core image library. If POWER/VS is to be used for the first time and the IBM-supplied generated POWER/VS initialization module is not suitable, a POWER/VS generation must be performed. A generation procedure must also be performed when any of the industry systems (3600, 3650, 3660, or 3790) are to be used. The Subsystem Support Services utility is generated concurrent with the generation of an industry system.

All DOS program products and Type I and Type II components that are to be used with the generated DOS/VS SCP must be added to the DOS/VS operating system after its generation.

The time required to obtain an operational DOS/VS system can be reduced by utilizing the DOS/VS System Installation Productivity Option (IPO) instead of performing the system generation procedure. DOS/VS System IPO can be used to install Release 34, the Advanced Functions-DOS/VS program product, certain Systems Network Architecture (SNA) support (such as NCP/VS, 3790 Host Support Service, and 3600 Host Support Service), certain data base/data communications support (DL/I DOS/VS and CICS/DOS/VS, for example), and certain other language translator, sort, and utility program products.

Existing user-written programs (phases) that operate under DOS Version 4 can be used without modification in a DOS/VS environment unless they do any of the following:

- Reference permanently assigned locations in lower real storage whose contents vary depending on whether BC or EC mode is specified.
- Issue the LPSW or SSM instruction or directly reference fields in old or new PSW locations (such as the system mask field and the interruption code field) whose function or location is affected by which mode, BC or EC, is specified.
- Depend on an interface to the DOS Version 4 supervisor (code, control blocks, and areas) that is release-dependent.
- Build user standard header or trailer labels in the supervisor area prior to issuing the LBRET macro. The DOS/VS supervisor is always store-protected so a problem program running under a partition protect key cannot modify the supervisor area.
- Modify an active channel program with data being read (channel contains self-modifying CCW's) or by executing instructions, if the program is to run in virtual mode. Program modification is not required if these programs operate in real mode in DOS/VS. Such programs do not execute correctly in virtual mode because the modification affects the channel program with virtual storage addresses rather than the translated channel program that is actually controlling the I/O operation.
- Use the EXCP macro and user-written I/O appendages if the program is to operate in virtual mode. The program must ensure that the I/O appendage routine and all areas it references are fixed before the initiation of each I/O operation that uses the appendage, so that a disabled page fault cannot occur in the appendage. Such modification is not required if the program is to execute in real mode.
- Access mode-dependent fields in the PSW in the save area of any other task in the partition in which the program is executing. As discussed in Section 80:25, the PSW is stored in EC instead of BC mode format.
- Access the system mask, mode bits, or protection key field in the PSW save area for a STXIT routine. As discussed in Section 80:25, these fields are not stored in the BC mode PSW save area.
- Directly access the interval timer in location 80 to determine the amount of time remaining in an established interval. Such programs must be modified to use the TTIMER macro.

The following steps must also be taken (if they apply to the existing DOS Version 4 installation):

- Existing BTAM programs must use the new BTMOD logic module provided in DOS/VS, which is modified to operate in a paging environment.

For source programs that contain BTMOD, reassembly and relink-editing of each program is required. Each program that incorporates BTMOD during link-editing must be relink-edited after the required BTMOD is reassembled.

- Existing QTAM message control programs must be reassembled and relink-edited using DOS/VS. QTAM message processing programs need not be reassembled.

- If existing Assembler Language programs that invoke user-written macros are to be reassembled using the new DOS/VS Assembler, the macros must be placed in the COPY sublibrary and COPY statements for the user-written macros that are referenced in a program must be included at the beginning of the existing source programs. Alternatively, existing user-written Assembler macros can be converted to preedited format and placed in the macro sublibrary using the DOS/VS Assembler. No source program modification is required when the second approach is used.

- 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 instead of a five-bit protect key affects the processing being performed. It must be remembered that the SSK instruction 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.

- User-written programs that handle I/O operations for unsupported I/O device types must be modified if they are to operate in virtual mode in DOS/VS. Channel program translation and page fixing must be done by the user and the REAL parameter must be added to the EXCP macros that request I/O operations on the unsupported device.

- If POWER is currently being used, the transition steps outlined in the POWER/VS discussion in Section 80:40 must be taken to convert to POWER/VS.

- Integrated emulator programs that were generated to operate under DOS Version 4 on a Model 135, 145 or 155 cannot operate under a DOS/VS supervisor. The required emulator program(s) must be regenerated under DOS/VS control on a Model 135, 145, or 155 II/158, respectively. Modification of existing emulator generation control cards is not required.

The job control statements for existing problem programs do not require alteration, except for the EXEC statements for programs that must operate in real mode. The REAL parameter must be added to the EXEC statements of real mode programs. Optionally, the SIZE parameter can be added as desired. If I/O device type changes are being made, job control statements must be modified as is required when such changes are made in any DOS environment. Similarly, if more partitions are to be operative in the DOS/VS environment than in the DOS Version 4 environment, new job streams must be created.

If the cataloged procedures facility is to be used in the DOS/VS environment, the procedure library must be built. Job control statements in existing job streams for job steps that are to be cataloged can be altered as required (to use the modify capability, for example) and cataloged in the procedure library. Existing job streams must then be changed as necessary to invoke cataloged procedures.

When the relocating loader is to be used in a DOS/VS environment, all existing programs that are to be relocatable must be relink-edited (or reassembled and relink-edited if object modules are not available) and

placed in the appropriate core image library. Linkage editor control statements do not have to be modified to request relocatability if the default for the linkage editor is to produce relocatable phases when the beginning address specified in the PHASE statement permits. If there are any programs that specify a beginning address that is relocatable but that are not to be made relocatable, the NOREL parameter must be added to the ACTION statement for these programs before they are reassembled.

Existing absolute and self-relocating programs that are to remain nonrelocatable need not be relink-edited. However, if a self-relocating program is to be made relocatable, all self-relocating code must be removed from the program and it must be reassembled. A relocatable beginning address must be specified on the PHASE statement before the new object module is link-edited.

Files used by existing DOS Version 4 programs can be used without alteration in DOS/VS, assuming device type or access method changes are not made. If VSAM is to be used to replace ISAM, the affected files must be converted from ISAM to VSAM format, as discussed in Section 80:30, and appropriate changes to existing ISAM job control statements must be made.

If desired, the structure of existing user-written DOS Version 4 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 (phases) to operate correctly in a DOS/VS environment.

Private core image libraries used with a DOS Version 3 or 4 system cannot be used with DOS/VS and must be converted to the new core image library format implemented in DOS/VS.

When CORGZ is used to create a new SYSRES file in DOS/VS, the ALLOC statement must be used; it is no longer optional. If any user-written programs directly access a core image library directory or library blocks, they must be modified as required by the new core image library format.

If phases in addition to those specified in the IBM-supplied cataloged procedures are to be made resident in the SVA, control statements must be prepared for these phases and they should be added to the appropriate cataloged procedure. All user-written phases that are to be made resident in the SVA must be marked SVA-eligible in their PHASE statements and cataloged in the system core image library. This requires at least a link edit of these phases.

If the generic I/O assignment facility is to be used, existing ASSGN statements must be altered as required. Use of this facility may also necessitate changing of the volume serial numbers of some volumes to ensure unique volume serials within the installation.

Programs that are to use the local directory list facility must be modified to include the GENL macro, and FETCH/LOAD statements must be altered to specify the resident directory list(s). Reassembly and relink-editing of these programs is required.

When RPS support is to be utilized, RPS versions of the required disk logic modules must be generated and placed in the system core image library. Any user-written programs that access fields contained in the DTF extension when RPS is used must be modified. A system GETVIS area must be defined and the SVA must be made large enough to contain the RPS logic modules to be used in addition to any existing contents.

For transition from a System/360 DOS Version 3 environment to a System/370 DOS/VS 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 158 for System/360 Users (GC20-1781)

80:55 SUMMARY OF ADVANTAGES

As a growth system for DOS Version 4 users, DOS/VS offers many new facilities. Some are designed to enhance the usability of multiprogramming in a DOS environment by eliminating some of the job scheduling preplanning required in DOS Version 4 and by providing operational enhancements. Other new facilities can improve real storage utilization or are designed to improve supervisor performance. Others provide new functions not available to DOS Version 4 users.

A DOS/VS operating system can be designed to be more responsive to a changing daily workload than a DOS Version 4 operating system, and DOS/VS 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 DOS/VS supports many new features, including functions exclusive to System/370 (not provided in System/360), such as EC mode and dynamic address translation, DOS/VS remains upward-compatible with DOS Version 4. Supervisor modifications that are required to handle new features are, with a few exceptions, transparent to the user so that operators and programmers interface with DOS/VS using basically the same operator commands, job control statements, files, and programs as they use in a DOS Version 4 environment.

The single most significant new feature of DOS/VS 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 base publication of which this supplement is a part (either in Section 15:05 or 30:05). DOS/VS offers additional specific advantages over DOS Version 4, some of which also result from the implementation of virtual storage. These are summarized below.

EXPANDED, MORE FLEXIBLE MULTIPROGRAMMING

- Two additional batched foreground partitions are supported for all functions provided for the two batched foreground partitions in a DOS Version 4 environment. When the Advanced Functions-DOS/VS program product is installed, four more partitions are supported than in DOS Version 4.

- A maximum of 15 tasks, instead of 12, is supported in a multitasking environment.

- The interval timing facility can be used concurrently by all partitions and tasks instead of by only one at a time, enabling concurrent operation of programs that require the interval timing facility.

- The relocating loader enables a relocatable program to be loaded at any partition starting address, as determined at load time, without the necessity of maintaining more than one version of the program in executable format or relink-editing the program to relocate it.

- All virtual partitions can be defined of equal size and large enough to contain the largest existing application that is to execute in any available partition. The dispatching priority of partitions can be changed during system operations if required.

- I/O devices can be dynamically selected by the job control program for allocation to job steps at initiation time. This facility can eliminate much of the I/O device assignment preplanning required in a DOS Version 4 environment and provides more flexibility in job

scheduling by reducing requirements for partition-dedicated I/O devices.

OPERATIONAL ENHANCEMENTS

- The SPI mode of program initiation is eliminated; however, real partitions still can be as small as 2K bytes.
- The use of cataloged procedures can minimize the amount of card handling the operator is required to do for each job stream and can reduce the maintenance required for frequently used job control.
- High-priority jobs can be handled more easily. One partition can be defined to handle only high-priority jobs. The dispatching priority of this partition can be changed when the high-priority job step is initiated, if necessary, to reflect the operational characteristics of the program. While this partition requires dedicated virtual storage (and, therefore, external page storage), real storage is required for the high-priority partition only when a job step is active in it.
- Virtual storage can be defined and organized to relieve the operator of some real storage management functions (such as changing partition sizes for the purpose of reallocating real storage during system operation).
- The relocating loader significantly reduces the number of operations that must be performed when supervisor size is increased by the inclusion of new function support, a new release of DOS/VS is installed that requires a larger supervisor, or partition starting addresses change.
- The generic I/O assignment capability enables device assignment to be partition-independent and labeled tape and disk volumes to be premounted on available drives prior to job step initiation.
- Multiple supervisors can be placed on one system residence volume, avoiding the need to have multiple system residence volumes and volume changing to change supervisors.
- POWER/VS requires less operator involvement than DOS/VS POWER.
- The system generation procedure has been simplified and can be performed in a partition concurrent with normal system operations instead of only in a standalone environment.
- DOS/VS System IPO can be used to further simplify DOS/VS release installation time.
- Display mode support for the 3277 and the display consoles for supported processors provides faster communication from the system to the operator.
- The asynchronous operator communications facility of the Advanced Functions-DOS/VS program product gives the operator flexibility in responding to messages.
- Private source statement and relocatable libraries can be placed on device types different from system source statement and relocatable libraries when Advanced Functions-DOS/VS is installed to ease the installation of new direct access device types and aid portability.
- Installation of 3330 Model 11 and 3350 devices operating in a native mode is eased by the dynamic link support that eliminates the need

to modify and reassemble existing programs to specify the new device type

IMPROVED UTILIZATION OF REAL STORAGE

- Inefficient use of real storage caused by unused storage within defined partition sizes and/or residence of inactive portions of the program can be significantly reduced. Unused virtual storage in a virtual partition does not have real storage assigned, and real storage allocated to inactive pages of a virtual mode program is released and reallocated to active pages in the system when necessary. The SIZE parameter can be specified for real mode programs to free real storage defined for the real partition that will not be used by the real mode program.

- Dynamic real storage management is provided for all programs that operate in virtual mode in a DOS/VS environment, regardless of the language in which they are written. Limited real storage management in a nonplanned overlay program can be handled during program execution only by an Assembler Language programmer in DOS Version 4 (use of FETCH/LOAD macros, for example).

PERFORMANCE ENHANCEMENTS

- Concurrent operation of additional partitions can increase system throughput for larger system configurations in which required resources (such as CPU time, channel time, real storage, etc.) are available but unutilized in a three-partition nonvirtual storage environment.

- The availability of five (or seven) partitions, instead of a maximum of three, can enable POWER/VS to be used in configurations in which support of only three partitions precludes use of POWER/VS. POWER/VS can increase system throughput by operating unit record I/O devices near rated speeds and overlapping peripheral unit record I/O operations with job step processing.

- Faster program fetching results from the new organization of the core image library that enables the location of a phase in a core image library to be determined more quickly (resident second level directories) or the directory entry for a phase to be located more quickly (resident system and local directory lists).

- Use of the shared virtual area for frequently used routines can reduce the total number of page faults encountered.

- The new preedited format of Assembler Language macros enables Assembler Language programs to be assembled faster.

- Temporary halting of task dispatching during job control processing is minimized to reduce the amount of serialized system processing that occurs in a multiprogramming environment.

- Use of block multiplexing and rotational position sensing can increase system throughput by better utilizing available channel time change as a result of the reorganization that takes place.

- The use of VSAM instead of ISAM organization can improve processing performance for files with a large number of additions.

- The newer faster I/O devices (such as 3344 and 3350 direct access devices and the 3800 Printing Subsystem) are supported.

- **The time to perform a system generation has been significantly reduced by changes to the generation procedure and the support of DOS/VS System IPO.**
- Dynamic partition balancing support in Advanced Functions-DOS/VS can improve throughput for installations with jobs of widely varying or unknown operating characteristics.
- **The DOS/VS-VM/370 Linkage facility in Advanced Functions-DOS/VS is designed to improve the performance of DOS/VS when it executes in a virtual machine in a VM/370 environment.**
- **The linkage editor provided by Advanced Functions-DOS/VS is designed to reduce linkage editing through an improved directory search technique.**

access methods
  BTAM 81
  DAM 80
  ISAM 80
  QTAM 81
  SAM 80
  VSAM 87
  VTAM 81
ACTION linkage editor control statement 54,204
advantages summary 224
Advanced Functions-DOS/VS program product
  additional partition support 214
  asynchronous operator communications 215
  device independence for private libraries 216
  DOS/VS-VM/370 Linkage facility 216
  dynamic partition balancing 214
  faster linkage editing 215
ALLOC command and macro 12
ALLOCR command and macro 18
Assembler Language 132
ASSGN command and statement 46
assignments for system logical units 9
attention commands 49
AUTO option of SIZE parameter 13,19

background partition 10,11,17
BJF mode 11
block multiplexing 82
BTAM 81
buffer loading for printers 29

cardless systems 2
CAT command 26
cataloged procedures 42
  contents 43
  example 45
  IBM-supplied 42
  modification 44
  partition-related 44
  procedure library 42
CDLOAD macro 58
channel check handler 127
channel program translation
  channel scheduler support 85
  macros for user use 87
CHAP macro 55
checkpoint/restart 21
clock comparator 57
commands
  attention 49
  IPL 26
  job control 48
communication region
  changes 52
  partition-related 52
  system-related 52
consoles
  display mode 38
  hardcopy file 39

  hardcopy printer 39
  Model 115/125 console display emulation mode 38
  printer-keyboard mode 39
  types supported 8,36
control program components 31
copy blocks 86
core image library
  organization 32
  private 33
  system 33
CPU timer 57
CPU's supported by DOS/VS 2
cross-partition event control 58

DAM 80
data interchange
   DOS and OS 80
   Model 20 and DOS/VS 213
data management
  access methods 80
  block multiplexer channel support 82
  channel program translation and page fixing 85
  new features supported 80
  new I/O devices supported 81
  rotational position sensing support 83
  VSAM 87
deactivation, partition 76
debugging aids
  DUMP macro 127
  JDUMP macro 127
  PDAIDS 127
  SDAIDS 128
  SYSVIS dump 128
disabled page faults 51
display operator console support 36
DOSVSDMP program 128
DOS/VS
  components 31
  minimum hardware configuration 2
  systems supported 2
  transition from DOS Version 4 218
DPD command 26
DUMP macro 127
dynamic address translation 2,12,17,18
dynamic link support for 3350 and 3330 Model 11 devices 85

emulator programs 212
EXCP macro 85
external page storage
  direct access devices supported 23
  initialization 28
  organization 24
  page capacity by device type 24

fast CCW translation 68,72,86
FCEPGOUT macro 75
features
  optional 5
  standard 4
fetch protection 21
fetch table in supervisor 53
foreground partitions
  in real address area 17
  in virtual address area 11-12
  restrictions 10

  scheduling 11
forms control buffer loading 29
FREEREAL requests 72
FREEVIS macro 57

generic I/O device assignment 46
GENL macro 14
GETIME macro 53
GETREAL requests 72
GETVIS area
  partition 57
  system 14
GETVIS macro 57

high-speed standalone dump 128
hold queue 62

IEBIMAGE utility 206
indirect data address list 86
initialization, system 25
I/O devices supported 7
interval timer 56
IPL communication device list 25
IPL procedure
  buffer loading for printers 29
  operator commands 26
  supervisor initialization 27
  supervisor selection 26
  $SYSOPEN exit routine execution after IPL 29
ISAM 80
installation verification procedure 219

JDUMP macro 127
job control
  commands 48
  exit facility 41
  new statements and parameters 40
  program 40

label cylinder organization 36-37
language translators supported 31
LFCB command and macro 29
locating a directory entry during program loading 34-35
librarian 205
linkage editor 204
local directory lists 14
LUCB command 29

machine check analysis and recording 127
main page pool 18
minimum system configuration 2
models supported 2
modifier statements, cataloged procedures 44
multiple timer support 56
multitasking 55

nonrelocatable phases 54
NOREL parameter on the ACTION statement 54
NPARTS parameter 10

OLTEP 127
operator commands 48
operator consoles 8,36
optional features 5
OVEND statement 44

page data set
  device types supported 23
  initialization 28
  organization 24
page fault
  disabled 51
  processing 65
page fault handling overlap 56
page fixing 67,70
page frame table 59-61
page frame table appendage 60
page I/O routine 65
page management
  FCEPGOUT macro 75
  functions 59
  general operation 59
  GETREAL/FREEREAL requests 72
  hold queue 62
  page fault processing 65
  page frame table 59-61
  page frame table appendage 60
  page I/O routine 65
  page replacement algorithm 61
  PAGEIN macro 76
  PFTE queues 62
  PFIX/PFREE macros 67
  RELPAG macro 75
  selection pool 62
  TFIX/TFREE requests 70
page pool
   in real storage 22
   in virtual storage 18
page replacement algorithm 61
page selection routine 63
page tables
  initialization 28
  location 27
  modification by job control program 41
page translation exception 50
PAGEIN macro 76
PARTDUMP option 127
partition-related cataloged procedures 44
partitions
  deactivation/reactiviation 76
  number supported 10
  priority 11
  real 17
  virtual 11
PBDY parameter on a PHASE statement 54,205
PDAIDS 127
PDZAP program 206
permanent fixing 67
PFIX macro 67
PFREE macro 68
PFTE queues 62
POWER for DOS Versions 3 and 4 132
POWER for DOS/VS 132
POWER/VS
  account file 147
  accounting facilities 173
  advantages 190
  AUTOSTART 147
  buffers 145
  commands

non-RJE 175
RJE 185
comparison with DOS/VS POWER 194
compatibility with DOS/VS POWER 201
cross partition communication 173
CTL statement 142
data file 144
DATA statement 143
DBLK generation macro 144
devices supported 157
diskette support 159
disposition attributes 138
dummy assignments 150
dummy devices 150
execution processors
list 165
punch 165
read 162
EOJ statement 142
functions 132,134
general operation 133
generation 133,190
initiation 147
input classes 135
input streams 158
intermediate storage 143
job classes 135,137
job definition 141
job entry control language 135,140
job logging 163
job number 158
job priority 135,137
job queuing 137
job scheduling 135,137
job separation 170
JOB statement 141
list queue 136
list tasks 168
LST statement 142
operator commands
non-RJE 175
RJE 185
operator messages 188
output classes 137
output limiting 167
output segmentation 166
partitions controlled by 134
priority 135,137,158
PUN statement 143
punch queue 137
punch tasks 168
queue file 136,145
RDR statement 143
read tasks 156
reader-only partition 149,152
reader queue 136,137
real storage requirements 134,190
remote job entry
functions supported 179
tasks 155,181,184
terminal commands 185
terminals supported 158
RJE,BSC support 180
RJE,SNA support 183
RJE tasks 155

SEGMENT macro 167
SLI statement 143
source library inclusion facility 158,164
spooling, definition 133
tasks 154
termination 149
track groups 144
versions 133
virtual storage requirements 188
warm start 148
writer-only partition 133,149,152,164
writer tasks 168
PRINTLOG utility program 39
private core image libraries 33
private second level directories 33
problem determination aids 127
procedure library 42
program event recording 128
program fetch 55
program loading
real mode programs 55
search sequence for a directory entry 34
virtual mode programs 54
programmer logical units 3
PRTY
command 26
parameter 11

QTAM 81
QSETPRT macro 29

reactivation, partition 78
real address area 16
real mode
program operation in 18
programs that must operate in 19
REAL parameter
on EXCP macro 85
on EXEC statement 18
real partitions
allocation 18
definition 11
number 17
permanent 18
size 18
SIZE parameter 18
temporary 18
real storage
allocation 61
minimum system requirements 2
minimum supervisor requirements 50
organization 22
REALAD macro 87
recovery management support recorder 127
REL parameter on ACTION statement 54,204
RELPAG macro 75
relocating loader 53
resident supervisor
in real storage 22
in virtual storage 17
minimum size 50
rotational position sensing 83
RUNMODE macro 57

SAM 80

SDAIDS 128
second level directory
  private 33
  system 33
SECTVAL macro 85
segment table
  initialization 27
  location 27
segment translation exception 50
selection pool 62
SEND system generation macro 22
service programs
  librarian 205
  linkage editor 204
SETPRT job control statement 40
SETPRT macro 29
SET SYSTEM MASK instruction 51
SET SDL command 15,27
SET SVA command 26
shared virtual area
  contents 13
  creation 15
  size 13
SIZE parameter 13,18,57
slots (see external page storage)
sort/merge programs 210
SPI mode 11
standalone dump program 128
standard features 4
storage protection 21
STORE THEN AND SYSTEM MASK instruction 51
STORE THEN OR SYSTEM MASK instruction 51
STXIT routines 51
supervisor
  loading an alternate during IPL 26
  minimum sizes by model 50
  modifications 50-53
  new features 53-59
supervisor area
  in real storage 22
  in virtual storage 17
  minimum size 50
  patch area 50
SVA eligibility 15,204
SVA parameter
  on PHASE statements 15,204
  on SVA-defining control statements 15
  on the ALLOC macro at system generation 13
synchronous exit facility 55
SYSBUFLD program 29
SYSCAT logical unit 26,113
SYSRES file contents 36
system components 31
system configuration, minimum 2
system debugging aids 127
system directory list 14
system generation 218
system GETVIS area in the SVA 14
system initialization 25
system logical units 9
system tasks 52
SYSVIS dump program 128
SYSVIS logical unit 23,28

task timer feature 57

teleprocessing balancing 78
temporary fixing 70
terminals supported 8
TFIX/TFREE requests 70
time-of-day clock support 53
timing facilities 56-57
TPIN macro 78
TPOUT macro 78
transition from DOS Version 4 to DOS/VS 218
translation specification exception 50
TTIMER macro 56
Type I language translator support 31

utilities 206

variable partition priority 11
VIRTAD macro 87
virtual address area 11
virtual mode
  program operation in 12
  programs that must operate in 13
virtual partitions
  allocation 12
  definition 10
  number 11
  size 12
  SIZE parameter 13,58
virtual storage
  organization 10
  size supported 10
virtual storage management facility 57
volume premounting 47
VSAM
  access method services program 115
  addressed processing
    entry-sequenced files 110
    key-sequenced files 107
  advantages 120
  alternate indexes
    for entry-sequenced files 109
    for key-sequenced files 103
  alternate key 103
  asynchronous processing exit 93
  backward processing 106
  catalogs 113
  chained parameter list 94,98
  clusters 98
  comparison with ISAM 122
  compatibility with OS/VS VSAM 88
  concurrent request processing 96
  control area 89
  control area splitting 99
  control interval 89
  control interval processing 92,107,110,112
  control interval splitting 99
  data space 91
  devices supported 87
  entry-sequenced files
    organization 109
    processing 110
  file sharing 118
  free space 98
  general description 87
  high-level languages supported 118
  index processing 108

ISAM interface routine 118
journaling 93
keyed processing
  key-sequenced files 105
  relative record files 112
key-sequenced files
  organization 98
  processing 105
macros 93
mass sequential insertion 106
organizations supported 88
password protection 117
paths 104
physical structure of files 89
preformatting 90
primary index file
  logical structure 100
  physical structure 102
  processing 105
  searching 101-102
primary key 98
relative byte address 92
relative record files
  organization 111
  processing 112
reusable files 92
shared resource facility 96
space allocation 91
spanned record 90
stored record 89
SVA residence 119
types of processing supported
  entry-sequenced files 111
  key-sequenced files 108
  relative record files 113
use of GETVIS/FREEVIS macros 119
virtual storage requirements 119
VTAM 81

XECBTAB macro 58
XPOST macro 58
XWAIT macro 58

$JOBEXIT exit routine 160
$LIBSTAT program 205
$MAINDIR program 205,15,34,53
$SYSOPEN exit routine 29

DOS/Virtual Storage Features Supplement

Printed in U.S.A. GC20-1756-2

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

DOS/Virtual Storage
Features Supplement
GC20-1756-2

READER'S COMMENT FORM

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name and mailing address:

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

What is your occupation? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape | **Please Do Not Staple** | Fold and tape

First Class
Permit 40
Armonk
New York

Business Reply Mail

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

Postage will be paid by:

International Business Machines Corporation
Department 824
1133 Westchester Avenue
White Plains, New York 10604

Fold and tape | **Please Do Not Staple** | Fold and tape

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Cut or Fold Along Line

DOS/Virtual Storage Features Supplement

Printed in U.S.A. GC20-1756-2