GC20-1754-2
File No. S370-01

Systems

A Guide to the IBM System/370 Model 158

IBM

GC20-1754-2
File No. S370-01

Systems

A Guide to the IBM System/370 Model 158

This guide presents hardware, programming systems, and other pertinent information about the IBM System/370 Model 158 that describes its significant new features and advantages. Knowledge of the IBM System/370 Model 155 is assumed. Features common to Models 155 and 158 are indicated but not discussed in detail. The contents of the guide are intended to acquaint the reader with the Model 158 and to be of benefit in planning for its installation.

Associated with this guide are four optional supplements that describe operating systems for the Model 158 that support a virtual storage environment. Each supplement has its own form number and must be ordered individually, if required. Optional supplements are the following:

- *DOS/Virtual Storage Features Supplement* (GC20-1756)
- *OS/Virtual Storage 1 Features Supplement* (GC20-1752)
- *OS/Virtual Storage 2 Features Supplement* (GC201753)
- *Virtual Machine Facility/370 Features Supplement* (GC20-1757)

IBM

Third Edition (August 1975)

This is a major revision obsoleting GC20-1754-1. Text has been added to include information about the Model 3 system (3158-3 Processor Unit), 3056 Remote System Console, Remote Support Facility (RSF), and 3340 direct access storage facility. The 3330 section, virtual machine concepts section and all summary tables have also been updated. Miscellaneous changes have been made throughout the text. Changes to the text and illustrations are indicated by a vertical line in the left margin.

This guide 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 the Model 158, its associated components and its programming support. 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/Systems, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604. Comments become the property of IBM.

PREFACE

It is assumed that the reader of this publication is familiar with System/370 Model 155 hardware features, channels, I/O devices, and programming support as described in A Guide to the IBM System/370 Model 155 (GC20-1729), and/or system library publications concerning Model 155 hardware and programming systems support. This guide discusses in detail only the hardware features of the Model 158 that are different from those of the Model 155 and the programming support provided for new features of the Model 158.

There are two versions of the Model 158; the Model 1 and the Model 3. The hardware differences between Model 1 of the Model 158 and the Model 155 are discussed in Sections 01 to 50. The differences between Models 3 and 1 of the Model 158 are discussed in Section 65.

The Model 158 is not compared with a Model 155 II, which is a purchased Model 155 with the optional Dynamic Address Translation Facility installed. However, functional descriptions of Model 158 features that are also part of the Dynamic Address Translation Facility of the Model 155 II apply to the Model 155 II as well, unless otherwise noted. This publication applies to systems with 60-cycle power.

The total Model 158 guide consists of this base publication (Sections 01 to 70), which covers virtual storage and virtual machine concepts and Model 158 hardware and I/O devices, and from one to four optional supplements (Sections 80 to 110). The optional supplements describe the facilities of the IBM programming systems that support a virtual storage environment using the dynamic address translation hardware of the Model 158. Each optional supplement has its own unique form number and each supplement desired must be ordered separately and inserted in this base publication, which is distributed without the automatic inclusion of any optional supplements.

The following optional supplements can be inserted in this base publication:

- DOS/Virtual Storage Features Supplement (GC20-1756) - assumes knowledge of DOS Version 4
- OS/Virtual Storage 1 Features Supplement (GC20-1752) - assumes knowledge of OS MFT
- OS/Virtual Storage 2 Release 1 Features Supplement (GC20-1753) - assumes knowledge of OS MVT
- Virtual Machine Facility/370 Features Supplement (GC20-1757) - does not assume knowledge of CP-67/CMS

All optional supplements also assume knowledge of virtual storage, dynamic address translation, and other new Model 158 features as described in this base publication or appropriate system library manuals. However, no optional supplement requires knowledge of the contents of any other optional supplement.

This base publication, as well as each optional supplement, begins with page 1 and includes its own table of contents and index. The base publication or supplement title is printed at the bottom of each page as a means of identification.

The optional programming systems supplements contain System/370 model-independent information, unless otherwise noted, and are designed to be included in the guides for System/370 Models 135, 145, 158, and 168, as shown below.

Base Publications	Supplements			
	DOS/VS Features Supplement (GC20-1756)	OS/VS1 Features Supplement (GC20-1752)	OS/VS2 Release 1 Features Supplement (GC20-1753)	VM/370 Features Supplement (GC20-1757)
A Guide to the IBM System/370 Model 135 (GC20-1738-4 or later editions)	X	X		X
A Guide to the IBM System/370 Model 145 (GC20-1734-2 or later editions)	X	X	X	X
A Guide to the IBM System/370 Model 158 (GC20-1754)	X	X	X	X
A Guide to the IBM System/370 Model 168 (GC20-1755)		X	X	X

CONTENTS

Base Publication Sections (Sections 01 to 70)

FIGURES (Sections 01 to 70)

TABLES (Sections 01 to 70)

SECTION 01: SYSTEM HIGHLIGHTS OF MODELS 1 AND 3

The System/370 Model 158 is an advanced function growth system for System/370 Models 145 and 155 and System/360 Models 50, 65, and 67. The Model 158 provides major new functions that are not basic to System/360 architecture. The Model 158 has new features, and new programming systems support that are designed to facilitate application development and maintenance. In addition, a Model 158 and its new programming support can ease entry into, and expansion of, online data processing operations.

The Model 158 makes new functions available to Model 50, 65, 145, and 155 users without requiring a major conversion effort, since the Model 158 and its programming support are upward compatible with these models and their programming support. DOS Version 4, OS MFT, and OS MVT can be used on a Model 158. However, the Model 158 has standard features that are designed to support a virtual storage environment, and new versions of OS and DOS operating systems are provided that use these features.

Compatible growth from a System/360 operating system to a Model 158 virtual storage environment can be achieved using the System/370 operating systems: DOS/Virtual Storage (DOS/VS), OS/Virtual Storage 1 (OS/VS1), and OS/Virtual Storage 2 (OS/VS2), which are based on DOS Version 4, OS MFT, and OS MVT, respectively. These operating systems run only on System/370 models with extended System/370 functions; that is, on those with extended control mode of system operation and dynamic address translation facilities. They cannot operate on System/360 models. In addition to implementing virtual storage, the System/370 operating systems offer many other new capabilities and performance-oriented enhancements that are not provided by DOS Version 4 or OS MFT and MVT.

A virtual machine environment is supported by Virtual Machine Facility/370 (VM/370), the successor to CP-67/CMS for System/370. While CP-67/CMS is available only to Model 67 System/360 users, VM/370 operates on System/370 Models 135, 145, 155 II, 158, 165 II, and 168. Model 67 users who have CP-67/CMS installed can use VM/370 on a Model 158 with some conversion effort. The optional Virtual Machine Assist (VMA) feature can be installed on a Model 158 to improve the performance of certain operating systems that execute in a virtual machine under VM/370 control.

Transition with little or no reprogramming is provided for users who are emulating a DOS Version 3 or 4 environment under OS, a 1401/1440/1460/1410/7010 system under DOS or OS, or a 7070/7074 system under OS, as well as for users of 1400-series and 7070/7074 systems, since the integrated emulators for these systems are also supported by the System/370 operating systems.

Two models of the Model 158 are provided. The Model 3 is an advanced version of the Model 1. The Model 3 has operational enhancements and CPU hardware features that give it improved availability and faster internal performance than the Model 1. The higher performance features of the Model 3 consist of hardware implementation differences in the Model 158 CPU (made possible by engineering design modifications and technology advances) and a larger high-speed buffer.

The performance features of the Model 3 do not require any programming support. Thus, programs that execute correctly on the Model 1 will execute correctly on the Model 3 without any programming changes,

assuming they have no timing dependencies and do not access model-dependent logout areas that differ for the two models.

Highlights of the Model 158, Models 1 and 3, when compared with a Model 155, are as follows (features are the same for Models 1 and 3 except where stated otherwise):

- A basic control (BC) mode and an extended control (EC) mode of system operation are standard. Only BC mode is provided in the Model 155. EC mode of operation provides additional system control and supports new functions that are not provided in System/360 or a Model 155.

- Internal performance of a Model 158 operating in BC mode is faster than that of a Model 155. The instruction execution rate of the Model 158 Model 1 is generally in the range of 20 to 50 percent faster than that of the Model 155 when identical system configurations, identical programs, and the same operating system are used. The increased internal performance of the Model 1 results from several improvements, among which are faster execution of several instructions and faster cycle times of processor storage in the Model 158.

  The internal performance of Model 3 of the Model 158 is generally in the range of 5 to 11 percent faster than that of the Model 1 when identical system configurations, identical programs, and the same operating system are used. The increase in Model 3 internal performance is primarily due to the larger high-speed buffer it contains, implementation of a more effective buffer assignment algorithm, faster execution of certain frequently used instructions, and a faster read cycle time.

- Dynamic address translation (DAT) is a standard facility that can be made operative only when the Model 158 is in EC mode. It provides hardware translation of addresses during program execution. One virtual storage of up to 16 million bytes or multiple virtual storages of up to 16 million bytes each can be supported using DAT hardware. (The amount of virtual storage that can be efficiently supported by a Model 158 depends on the hardware configuration and job stream characteristics.) Channel indirect data addressing is also standard and is provided to handle I/O operations when dynamic address translation is used. Channel indirect data addressing enables the channels to access an I/O buffer that is contained in noncontiguous processor storage areas.

- Program event recording (PER) is standard and can be made operative when the Model 158 is in EC mode. It is designed to be used as a problem determination aid. This feature includes hardware that monitors the following during program execution: successful branches, the alteration of general registers, and instruction fetching from and alterations of specified areas of processor storage.

- A CPU timer and clock comparator are standard. The CPU timer provides an interval timing capability similar to that of the interval timer at location 80 but it is updated every microsecond, as is the time of day clock. The clock comparator can be used to cause an interruption when the time of day clock passes a specified value. These items provide higher resolution timing facilities than the interval timer and enable more efficient timing services routines to be written.

- New instructions that support dynamic address translation, the new timing hardware, and system control facilities are added to the System/370 instructions available for the Model 155.

- Extended precision floating-point, 1401/40/60,1410/7010 Compatibility, 7070/7074 Compatibility, and OS/DOS Compatibility are no-charge optional features. Extended precision floating-point and 7070/7074 Compatibility can be installed in the same Model 158. (They are mutually exclusive in a Model 155.)

- Processor storage is implemented using monolithic technology instead of discrete ferrite cores. Processor storage sizes of 512K, 1024K, 1536K, 2048K, 3072K, and 4096K are available. Monolithic storage for the Model 158 is faster and more compact than core storage for the Model 155. The cycle time of processor storage varies from 690 to 1035 nanoseconds for a Model 1 and from 690 to 920 nanoseconds for a Model 3, depending on the operation performed, which is significantly faster than the 2070-nanosecond cycle time of processor storage for the Model 155. A four-megabyte Model 158 is a little more than half the size of a two-megabyte Model 155.

- The optional Power Warning feature, when installed on a Model 158 with uninterrupted power supplies, provides a warning machine check interruption when the utility-supplied power is approximately 18 percent below the rated voltage. Program support of this interruption, which is provided by OS MVT Releases 21.6, 21.7, and 21.8, OS/VS1 as of Release 3, and OS/VS2 as of Release 1.6, is designed to permit an orderly system shutdown after a power line disturbance occurs, when necessary, so that operations can be restarted once the power supply is stabilized.

- The high-speed buffer organization and assignment algorithms implemented in Models 1 and 3 of the Model 158 are different from those implemented in the Model 155. They are designed to enable the Model 158 CPU to fetch data from the buffer, instead of from processor storage, somewhat more frequently than does the Model 155 CPU. In addition, a 16K high-speed buffer is standard in the Model 3. The Model 155 and Model 1 of the Model 158 each have an 8K high-speed buffer.

- Reloadable monolithic control storage, instead of read-only storage, is used for microprogram residence. Use of reloadable control storage offers the advantage of improved microcode serviceability.

- A display console with keyboard and light pen are standard. The display console functionally replaces a console typewriter-keyboard and almost all the lights and switches on the system control panel on the front of the Model 155 CPU. The display console is to be used by maintenance personnel as well as console operators. The display console offers faster operator-to-system communication via the light pen and faster message display than a typewriter-keyboard console when operating in display mode. When display mode is used, hard copy can be obtained, as an option, via the 85-cps 3213 Printer. The display console can also operate in printer-keyboard mode, which enables it to accept 1052, 3210, and 3215 printer-keyboard commands. The 3213 printer is required for printer-keyboard mode.

- The 3056 Remote System Console can be attached to a Model 158 via a cable up to 200 feet in length to provide a remote free-standing display console in addition to the standard display console on the Model 158 CPU. The optional 3056 Remote System Console provides a remote operator with almost all the functional capabilities as are provided by the standard display console.

- The maximum aggregate data rate of five block multiplexer channels operating concurrently in a Model 158 is 6.75 megabytes per second, compared with 5.4 megabytes per second in a Model 155.

- The byte and block multiplexer channels of a Model 3 can have more shared and nonshared subchannels than can the channels in a Model 1 or the Model 155. In addition, a shared subchannel in the Model 3 can be shared by a maximum of 32 instead of 16 devices.

- 3330-series disk storage (all models) and/or 3340 direct access storage facilities can be attached to a Model 158 block multiplexer channel via the optional Integrated Storage Controls (ISC) feature as well as via 3830 Storage Control (Models 1 and 2). The ISC feature provides dual direct access storage control functions equivalent to two 3830 Storage Control Model 2 units, except for four-channel switching. Up to four strings of from two to eight drives each can be attached to each of the two logical storage controls in the ISC feature, for a total of eight 3330-series and/or 3340 strings (64 drives) attached via the ISC feature. Optionally, the staging adapter feature can be installed on the ISC to permit attachment of the 3850 Mass Storage System via ISC in addition to via 3830 Storage Control Model 3.

- The 3340 direct access storage facility can be attached to the Model 158 via 3830 Storage Control Model 2 and the Integrated Storage Controls feature. The 3340 facility is intermediate capacity direct access storage that, because of its unique design and advanced technology, offers advantages over 2314 disk storage in addition to those provided by 3330-series disk storage. Automatic error correction features and multiple requesting are standard on the 3340. Rotational position sensing is optional.

  The storage medium for 3340 disk storage is the removable interchangeable 3348 Data Module which is a sealed cartridge that is never opened by the operator. In addition to the disks on which data is written, the 3348 Data Module contains a spindle, access arms, and read/write heads. The 3340 Disk Storage Drive contains the mechanical and electrical components required to operate the 3348 Data Module.

  The 3340 facility has an 885 KB/sec data transfer rate, average seek time of 25 ms, and full rotation time of 20.2 ms. A 3348 Data Module has a maximum capacity of approximately 35 million bytes or 70 million bytes, depending on the model. One model of the 3348 offers fixed heads for zero seek time to approximately 502,000 bytes maximum and movable heads for an average seek time of 25 ms to the remaining bytes in the data module. A string of from two to eight 3340 drives can be configured. From one to four 3340 strings can be attached to the 3830 Model 2 and to each of the logical controls in ISC. Any model of the 3348 can be mounted on a 3340 drive. Therefore, 3340 string capacity can vary from 70 million to 560 million bytes in 35 and/or 70 million byte increments.

  The sealed cartridge design of the 3340 facility offers the advantages of multiple capacities per 3340 drive, increased data reliability, and simplified data module loading and unloading procedures.

- 3344 Direct Access Storage can be attached to a Model 158 via 3830 Storage Control Model 2 and Integrated Storage Controls. It offers significantly increased maximum online capacity per drive for 3340 users without the necessity of program conversion. The 3344 is fixed media disk storage. Data is recorded on nonremovable disks. The 3344 is designed to eliminate operator handling, eliminate exposure to external contamination (like the 3348 Data Module), and provide high reliability.

  The 3344 has the same data transfer rate, average seek time, and full rotation time as the 3340. However, the maximum capacity of a

3344 drive is 280 megabytes, or the equivalent of four 70-million-byte 3348 data modules. The 3344 is a two-drive unit that attaches to the 3340 Model A2. A 3340/3344 string can contain any mixture of 3344 and 3340 units (as long as the first is a 3340 Model A2) for a maximum of eight drives with a maximum capacity of over 1.8 billion bytes.

Automatic error correction, rotational position sensing, and multiple requesting are standard in the 3344. Fixed head models that contain fixed heads for zero access time to a portion of the data and movable heads for access to the balance of the data are also available.

- 3350 Direct Access Storage can be attached to a Model 158 via 3830 Storage Control Model 2 and Integrated Storage Controls. The 3350 is very large capacity, high-speed, fixed media direct access storage. Data is stored on nonremovable disks. The 3350 is designed to eliminate operator handling, eliminate exposure to external contamination, and provide high reliability.

  The 3350 has a data transfer rate of 1198 KB/sec, average seek time of 25 ms, and full rotation time of 16.8 ms. A 3350 drive operating in native mode has a maximum capacity of 317.5 megabytes. A 3350 string can contain from two to eight drives in two drive increments for a maximum string capacity of over 2.5 billion bytes of online disk storage.

  The Standard Selective Format feature enables the format of each 3350 to be set by programming during volume initialization. A 3350 drive can operate in 3350 native mode, 3330 Model 1 compatibility mode, or 3330 Model 11 compatibility mode. When operating in 3330 Model 1 compatibility mode, a 3350 drive is the equivalent of two 3330 Model 1 drives in capacity. When operating in 3330 Model 11 compatibility mode, a 3350 drive is the equivalent of one 3330 Model 11 drive in capacity. This feature enables 3330-series user to obtain the price performance and functional advantages of the 3350 without program conversion.

  Automatic error correction, rotational position sensing, and multiple requesting features are standard. The 3350 is also available in fixed head models. These models provide fixed heads for zero access time to a portion of the data and movable heads for access to the balance of the data.

- A remote support facility (RSF) utilizing the standard service processor unit, not provided for the Model 155, is standard for the Model 158. RSF enables a customer engineer specialist at a remote location (System Support Center) to execute diagnostics and assist in locating hardware failures in a Model 158. This may eliminate the necessity of a trip to the installation by the customer engineer specialist. RSF is designed to improve system availability by reducing the amount of time required to locate hardware failures.

The Model 158 is designed primarily to support a virtual storage environment which allows programmers to write and execute programs that are larger than the processor storage available to them. When virtual storage is supported, restraints normally imposed by the amount of processor storage actually available in a system are eased. The removal of certain restraints can enable applications to be installed more easily, and can be valuable in the installation and operation of online applications. While some of the new hardware features of the Model 158 and some of the new facilities supported by System/370 operating systems are designed to improve performance, a virtual storage environment is designed primarily to help improve the productivity of data processing personnel and enhance the operational flexibility of the installation.

SECTION 10: PHYSICAL DESIGN AND SYSTEM TECHNOLOGY FOR MODELS 1 AND 3

The System/370 Model 158 is shown in Figure 10.1. It is air cooled and similar in physical appearance to the Model 155 except for the console panel on the front of the Model 158 CPU frame and the display console. A Model 155 cannot be converted to a Model 158.

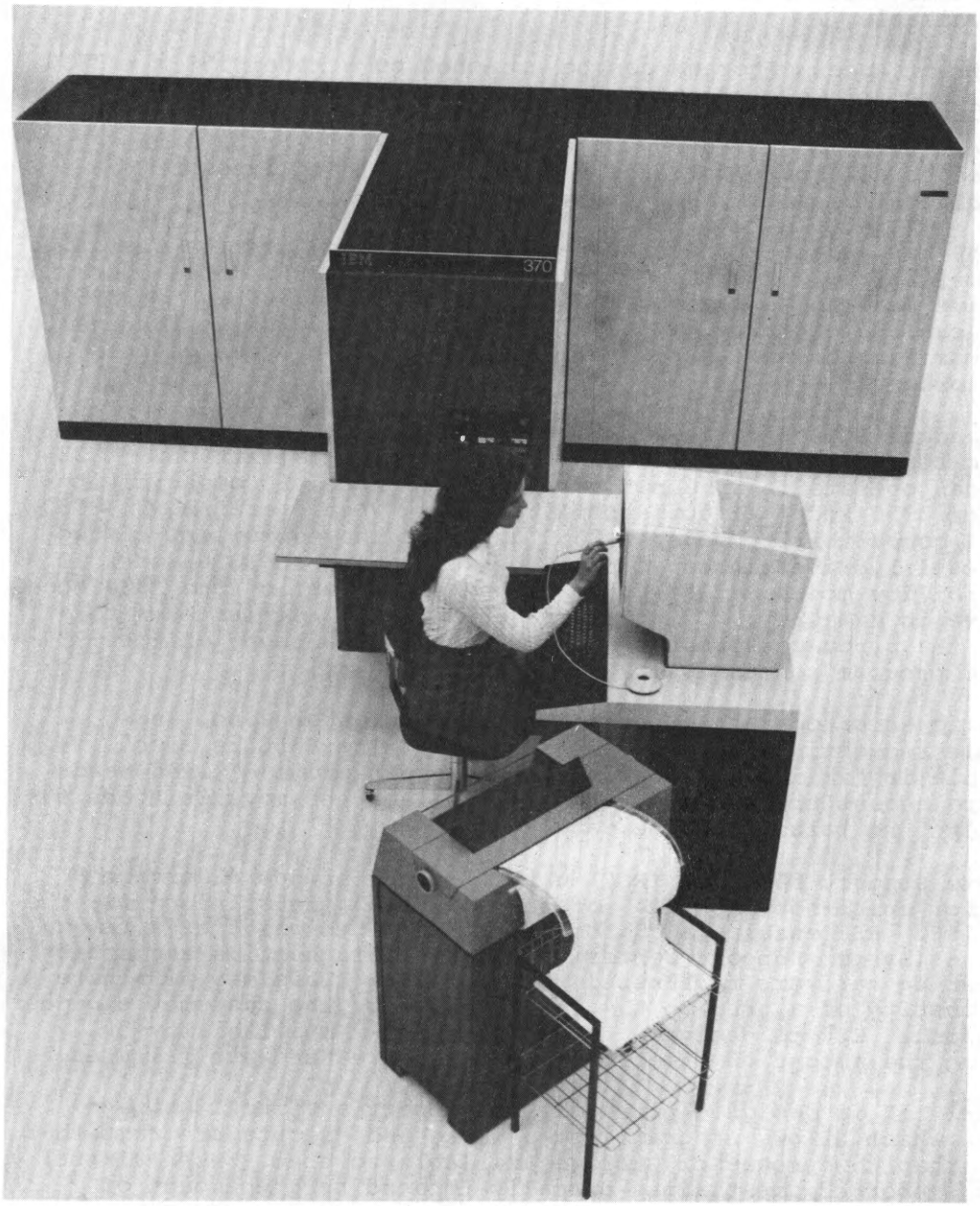


Figure 10.1. System/370 Model 158 (design model)

The physical size of Models 1 and 3 of the Model 158 is the same. the physical size of a Model 158 with more than 512K of processor storage is significantly smaller than the size of a Model 155 with more

than 512K as a result of the implementation of monolithic, instead of magnetic core, processor storage. Processor storage is contained within the CPU frames of the Model 158. A 512K Model 158 is a half frame larger than a Model 155 with 512K because of the additional hardware features of the Model 158. The two models are the same height. The size of a Model 158 is the same regardless of the amount of processor storage installed. A four-megabyte Model 158 is a little more than half the size of a Model 155 with two megabytes of processor storage.

Monolithic technology is used to implement all logic and all storage (processor, local, control, and buffer) in the Model 158. Use of monolithic technology for processor storage, as well as for logic, represents a significant technological advance in storage implementation. The monolithic storage implemented in the Model 158 provides several advantages over the wired, discrete ferrite core storage implemented in the Model 155.

Monolithic storage is similar in design to monolithic logic circuitry, the latter representing a technological advance over the solid logic technology (SLT) introduced with the announcement of System/360. Since the technology associated with monolithic storage is like that used to produce monolithic logic, monolithic storage can be batch-fabricated.

Solid Logic Technology (SLT)

Monolithic technology is a breakaway from the hybrid circuit design concept of SLT and can best be explained by comparison with SLT. As shown in Figure 10.2, SLT circuits were implemented on half-inch ceramic squares called substrates. Metallic lands on the substrate formed interconnections onto which the components were soldered. These components consisted of transistors and diodes, which were integrated on silicon chips about the size of a pinhead, and thin film resistors. An SLT chip usually contained one type of component, and several chips and resistors were needed to form a circuit. In general, an SLT substrate contained a single circuit.

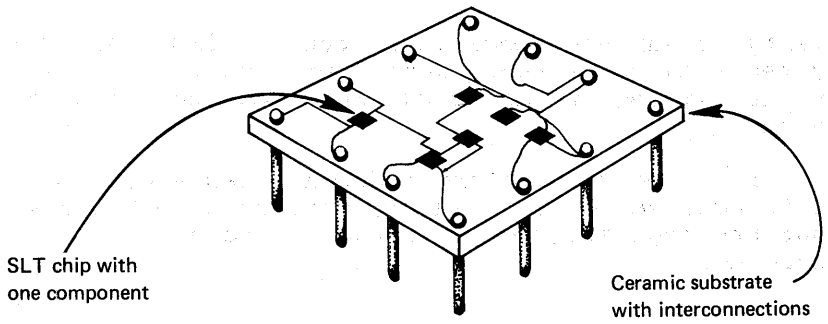


Figure 10.2. SLT substrate

Monolithic System Technology (MST)

Monolithic system technology also makes use of a half-inch-square ceramic substrate with metal interconnections onto which chips are placed. However, in monolithic logic circuitry, large numbers of elementary components, such as transistors and resistors, are integrated on a single chip. Unlike an SLT chip, an MST logic chip usually contains several interconnected logic circuits instead of only one component. MST logic modules, each consisting of one substrate, are

mounted on circuit cards, which are in turn mounted on circuit boards (as in SLT logic).

MST logic offers the following advantages over SLT:

- MST logic circuitry is intrinsically more reliable because many circuit connections are made on the chip, significantly reducing the number of external connections.
- Faster circuit speeds can be obtained because the path between circuits is considerably shorter.
- Space requirements for logic circuitry are reduced by the significantly higher density of components per chip.

Monolithic Storage

Monolithic storage design incorporates the same concepts described for monolithic logic. However, storage cells that are used to contain storage bits instead of logic circuits are implemented on a metal oxide semiconductor chip. In the Model 158, a monolithic storage array chip is approximately 1/8 x 3/16 of an inch in size and contains a large number of interconnected circuits. These circuits form storage bits and support circuitry (decoding, addressing, and sensing) on the chip.

Since power is required to maintain a one or zero state in a monolithic storage bit, data is lost when power is turned off, and monolithic storage is, therefore, said to be volatile. This is not true of core storage, which retains a magnetized state when power is removed.

The following are the advantages of monolithic over core storage:

- Faster storage speeds can be obtained, first, because of the shorter paths between storage circuitry and, second, because of the nondestructive read-out capability of monolithic storage. Since core storage read-out is destructive, a regeneration cycle is required after a read and is also used before a write. This type of regeneration cycle is not required for monolithic storage.
- Storage serviceability is enhanced because storage is implemented in accessible, easily replaceable storage cards. Diagnostic routines need only identify the failing storage card, which can be replaced in a matter of minutes.
- Space requirements for system storage are reduced. Dense bit packaging per chip is achieved by the use of monolithic technology and by the fact that the regularity of a storage pattern lends itself to such packaging.

20:05 ARCHITECTURE DESIGN

Extended System/370 architecture embodies two different modes of system operation, basic control (BC) mode and extended control (EC) mode, as determined by bit 12 of the current PSW. When a Model 158 operates in BC mode, the contents, layout, and function of permanently assigned processor storage locations 0 to 127 are identical to these locations in System/360 Models 22 and up (except 44 and 67) with the exception of the use of PSW bit 12. BC mode is essentially the System/360-compatible mode of System/370 operation.

When EC mode is operative in the Model 158, the format of the PSW is altered and the number of permanently assigned locations extends beyond processor storage address 127. Changes to the PSW consist of the removal of certain fields to create space for additional mode and mask bits that are required for new functions, such as dynamic address translation and program event recording. The removed fields are assigned to locations above 127 and to a control register.

EC mode is effective when PSW bit 12 is a one. BC mode is effective when this bit is a zero. BC mode is established during initial program reset. Therefore, a control program must turn on bit 12 of the PSW in order to cause EC mode to become operative. As a result, control and processing programs written for System/360 (Models 22 and up except 44 and 67) will run without modification in BC mode on a System/370 Model 158 (either a Model 1 or a Model 3) that has a comparable hardware configuration, with the following exceptions:

1. Time-dependent programs. (They may or may not execute correctly.)

2. Programs that use machine-dependent data such as that which is logged in the machine-dependent logout area. (OS SER and DOS MCRR error-logging routines for System/360 models will not execute correctly.)

3. Programs that use the ASCII mode bit in the PSW (bit 12). ASCII mode is not implemented, and this bit is used in System/370 to specify BC or EC mode of operation.

4. Programs that depend on the nonusable lower processor storage area being smaller than 1184 bytes. This area can be reduced to 512 bytes by moving the CPU extended logout area.

5. Programs deliberately written to cause certain program checks.

6. Programs that depend on devices or facilities not implemented in the Model 158.

7. Programs that use model-dependent operations of the System/370 Model 158 that are not necessarily compatible with the same operations on System/360 models.

8. Programs that depend on the validity of storage data after system power has been turned off and then on.

Only BC mode is implemented in the Model 155. Hence, control and processing programs that currently operate on a Model 155 will run without modification in BC mode on a Model 158 (either a Model 1 or a

Model 3) that has a comparable hardware configuration, with the following exceptions:

1. Time-dependent programs. (They may or may not execute correctly.)

2. Programs that use machine-dependent data such as that which is logged in the machine-dependent logout area.

3. Programs deliberately written to cause certain program checks.

4. Programs that depend on the validity of storage data after system power has been turned off and then on.

The following operating systems generated for System/360 models can operate on a Model 158 (Models 1 and 3) in BC mode, if necessary: (1) DOS Version 3 operating systems generated for a Model 50 and (2) OS PCP, MFT, and MVT operating systems generated for Models 50, 65, and 67. They are subject to the eight compatibility restraints in the first list and, therefore, should be run with the Model 158 set to hard stop on machine checks (see Section 60:30 in A Guide to the IBM System/370 Model 155, GC20-1729, for a discussion of why this is necessary).

DOS Version 3 and 4 supervisors generated for a Model 155 can be used on a Model 158 Model 1 in BC mode if the Model 158 is set to hard stop on machine check errors. DOS Versions 3 and 4 support 3215 mode only for the display console. Processing programs used with these DOS versions are subject to the four compatibility restraints in the second list.

OS MFT and MVT support of the Model 158 (Model 1) operating in BC mode is provided as of OS Release 21.6. DIDOCS with 3270 support (provided as of Release 21.7) must be included in the generated control program in order to support display mode of operation for the display console. An OS MFT or MVT control program generated for the Model 155 using OS Release 21.7 or later also can be used on a Model 158 to support BC mode operations (the Model 158 should be specified as an alternate CPU via the SECMODS macro at system generation). DIDOCS with 3270 support should be included in the Model 155 control program when it is generated in order to support display mode for the display console when the control program is used on a Model 158. Only printer-keyboard mode is supported by the OS starter systems provided for the system generation procedure.

Support of Model 158 Model 1 systems operating in EC mode is provided by DOS/VS, OS/VS1, OS/VS2 Releases 1 and up, and VM/370, each of which is designated as system control programming (SCP). All of these programming systems support virtual storage using dynamic address translation, which operates only when the system is in EC mode. OS/VS2 Release 2 supports multiple virtual storages and Model 158 tightly coupled and loosely coupled multiprocessing configurations. VM/370 supports a virtual machine environment.

User-written processing programs that operate on a Model 155 under DOS (Version 3 or 4), OS MFT, or OS MVT control and that are not time-dependent can be used with DOS/VS, OS/VS1, or OS/VS2 Release 1, respectively, with little or no modification, as discussed in the optional programming systems supplements (Sections 80 to 100). Hence, compatible growth from a System/360 or a BC mode nonvirtual storage environment to an EC mode virtual storage environment is provided.

The following are standard features of the Model 158 (Model 1) that are functionally identical to the same features of the Model 155:

- Instruction set that includes System/360 instructions and the following System/370 instructions:

  COMPARE LOGICAL CHARACTERS UNDER MASK
  COMPARE LOGICAL LONG
  INSERT CHARACTERS UNDER MASK
  LOAD CONTROL, STORE CONTROL
  MONITOR CALL
  MOVE LONG
  SHIFT AND ROUND DECIMAL
  START I/O FAST RELEASE (executed as a START I/O)
  STORE CHANNEL ID
  STORE CHARACTERS UNDER MASK
  STORE CLOCK, SET CLOCK
  STORE CPU ID

- Monitoring feature
- Store and fetch protection
- Multiple control registers (more registers are implemented in the Model 158 than in the Model 155)*
- Interval timer (3.3 millisecond resolution)
- Time of day clock
- Byte-oriented operands
- Extended external interruption masking
- Expanded machine check interruption class (more data is logged by the Model 158)
- RAS features (instruction retry, ECC on processor storage, limited channel logout, and command retry)
- 8K of high-speed buffer storage (different assignment algorithm in the Model 158)
- Byte multiplexer channel 0
- Block multiplexer channels 1 and 2 (including selector channel mode)

The following are optional features of the Model 158 (Model 1) that are functionally identical to the same features on the Model 155:

- Extended precision floating point (no-charge)
- 1401/40/60,1410/7010 Compatibility (no-charge)
- 7070/7074 Compatibility (no-charge)
- OS/DOS Compatibility (no-charge)
- Block multiplexer channels 3, 4, and 5
- Second byte multiplexer channel (replaces channel 4)
- Channel-to-Channel Adapter
- Direct Control

The following are standard features of the Model 158 (Model 1) that are not available for the Model 155:

- Implementation of EC mode of system operation*
- Dynamic address translation*
- Reference and change recording*
- Channel indirect data addressing*
- CPU timer and clock comparator*
- Program event recording*
- Program interruption for SET SYSTEM MASK instruction*
- Store status function*
- Faster execution of certain classes of instructions
- Monolithic processor storage (instead of core storage)
- Reloadable monolithic control storage
- Display console with keyboard and light pen
- Service processor unit natively attached to the Model 158 CPU

*Part of the Dynamic Address Translation Facility of a Model 155 II. The functional descriptions of these items in this publication apply to their implementation in both the Model 158 and the Model 155 II, unless otherwise indicated.

- New instructions*

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

The following are optional features of the Model 158 (Model 1) that are not available for the Model 155:

- 3056 Remote System Console (attached via a cable up to 200 feet in length)
- 3213 Printer for hard copy (attached via a cable up to 100 feet in length
- Integrated Storage Controls for attachment of 3330-series and/or 3340 disk storage, or the 3850 Mass Storage System
- Staging adapter for Integrated Storage Controls
- Two-Channel Switch for Integrated Storage Controls
- Power Warning
- Virtual Machine Assist (no-charge)
- Multiprocessing (3058 Multisystem Unit)

*Part of the Dynamic Address Translation Facility of a Model 155 II. The functional descriptions of these items in this publication apply to their implementation in both the Model 158 and the Model 155 II, unless otherwise indicated.

All the new features of the Model 158 Model 1 except Integrated Storage Controls, multiprocessing, and those related to implementing virtual storage (such as dynamic address translation and reference and change recording) or virtual machines (VMA feature) are discussed in the remainder of this section.

20:10 THE CENTRAL PROCESSING UNIT

Like the Model 155, the Model 158 has a CPU cycle time of 115 nanoseconds and an internal data path that is four bytes wide. The implementation of local storage, instruction prefetching, expanded external interruption masking, and parity checking are the same in the two models. Control registers in addition to the five implemented in the Model 155 are implemented in the Model 158 in order to support new EC-mode-only functions. Additional control registers are implemented in the Model 155 II as well.

EXTENDED CONTROL MODE

Extended control mode, unlike basic control mode, is exclusively a System/370 mode and is not implemented in System/360. Note that IBM-supplied operating systems do not support System/370 models operating in EC mode without dynamic address translation operative also. Facilities that depend on which mode is in effect are discussed below. Any item not covered operates identically in BC and EC modes. (The entire discussion of EC/BC mode differences applies to the Model 155 II also, except for Figure 20.10.3.)

Change in PSW Format

When a System/370 operates in EC mode, the format of the PSW differs from the BC mode format. Both PSW formats are shown in Figure 20.10.1. In EC mode, the PSW does not contain individual channel mask bits, an instruction length code, or the interruption code for a supervisor call, external, or program interruption. The channel masks are contained in

control register 2, and the other fields are allocated permanently assigned locations in the fixed lower processor storage area above address 127.

Removal of the fields indicated provides room in the EC mode PSW for control of new features that are unique to EC mode (such as PER and DAT) and for the addition of summary mask bits (such as channel and I/O masks). Use of a single mask bit to control the operation of an entire facility (such as program event recording) or an entire interruption class (such as I/O and external) simplifies the coding required to enable and disable the system for these interruptions.

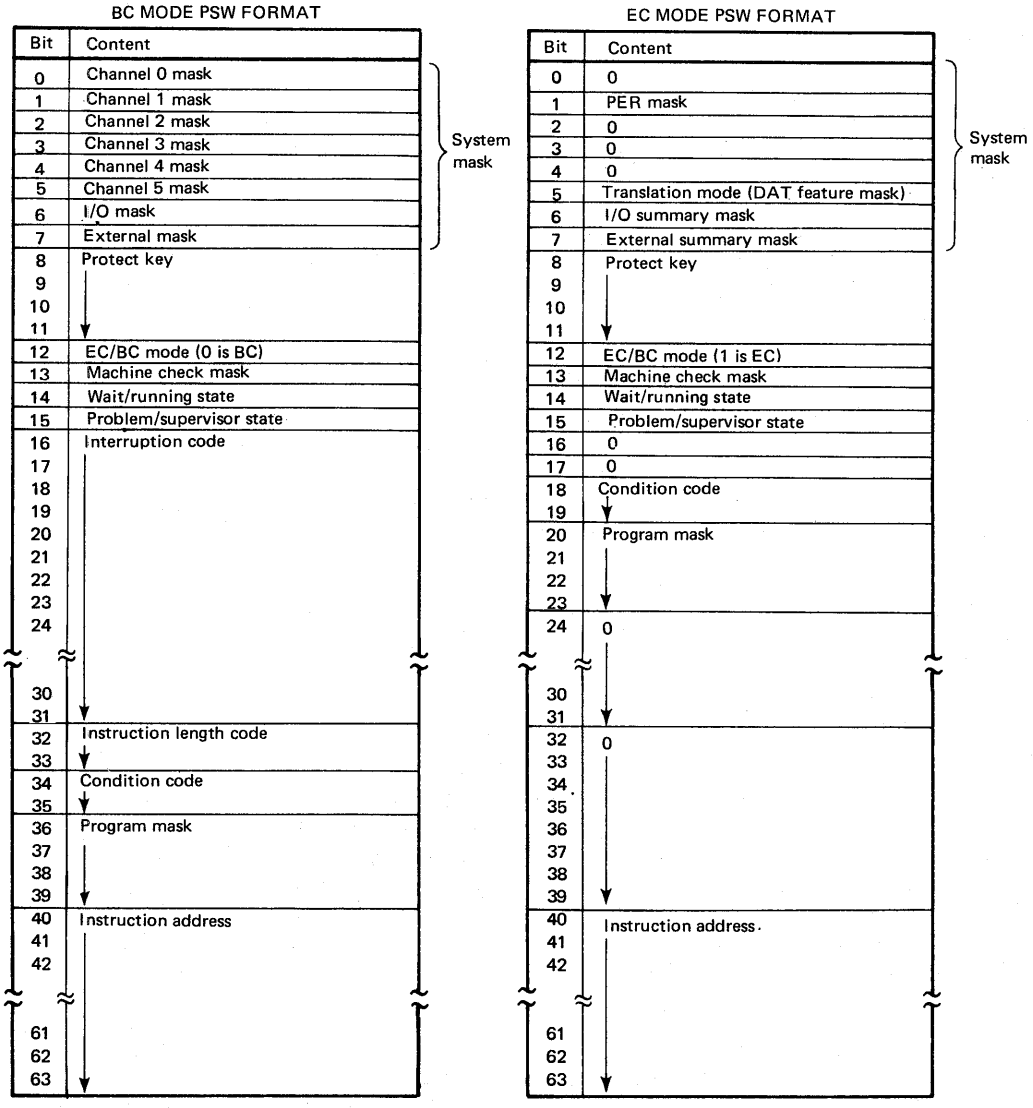


Figure 20.10.1. BC mode and EC mode PSW formats

Change in Permanently Assigned Processor Storage Locations

When a System/370 operates in EC mode, the number of permanently assigned locations in lower processor storage is increased to include fields for storing instruction length codes, interruption codes (for

supervisor call, external, and program interruptions), program event recording data, the I/O device address for an I/O interruption, and an exception address for the DAT feature. The model-independent BC mode and EC mode fixed storage areas for System/370 models are shown in Figure 20.10.2.

The balance of the fixed area for the Model 158 (Model 1), that which has model-dependent fields, is shown in Figure 20.10.3. This model-dependent area is not affected by whether EC or BC mode is specified except for locations 185 to 187, which contain the I/O address after an I/O interruption and an IPL only when EC mode is in effect. The machine check interruption procedure and the format of the data logged on a machine check are the same in EC and BC modes, except for differences in the PSW format and the permanently assigned locations previously discussed.

Channel Masking Changes

When a System/370 operates in EC mode, interruptions from each channel are controlled by the summary I/O mask bit in the current PSW (bit 6) and an individual channel mask bit in control register 2. In the Model 158, bits 0 to 5 in control register 2 are assigned to control channels 0 to 5, respectively. Both the summary mask bit and the appropriate individual channel mask bit must be on for an interruption from a given channel to occur. In BC mode, interruptions from channels 0 to 5 are controlled only by channel mask bits in the current PSW (bits 0 to 5).

Changes to Certain System/370 Instruction Definitions

All Model 158 instructions are valid in BC and EC modes. However, because of differences between the PSW format and the permanently assigned storage locations in EC and BC modes, the definition of certain instructions is affected. Instructions provided for both System/360 and System/370 whose definition is altered for EC mode are:

BRANCH AND LINK (RR, RX)	SET STORAGE KEY
INSERT STORAGE KEY	SET SYSTEM MASK
LOAD PSW	SUPERVISOR CALL
SET PROGRAM MASK	

Revised definitions of these instructions to include BC/EC mode differences are contained in System/370 Principles of Operation (GA22-7000-2 or later editions). Programs that operate in BC mode and that use LOAD PSW and/or SET SYSTEM MASK (SSM) instructions must be modified to operate correctly in EC mode. The eight-byte PSW to be loaded by LPSW instructions and the eight-bit system mask to be set by SSM instructions must be changed to EC mode format. (Programs that use SSM instructions and that are to be executed in an OS/VS1 or OS/VS2 environment need not be modified because the interruption for SSM instructions and an SSM simulation routine, as described next, are supported.)

Programs that use the other instructions listed do not have to be changed to operate correctly in EC mode, unless they use other facilities that are mode dependent. Programs that operate in BC mode and that use the STORE THEN OR SYSTEM MASK and STORE THEN AND SYSTEM MASK instructions (not provided for System/360) must also be modified to operate correctly in EC mode.

BC MODE FIXED AREA 0-159

Decimal locations					
0	IPL PSW				
8	IPL CCW 1				
16	IPL CCW 2				
24	External old PSW				
32	Supervisor call old PSW				
40	Program old PSW				
48	Machine check old PSW				
56	I/O old PSW				
64	Channel status word – CSW				
72	Channel address word – CAW	76	Unused		
80	Interval timer	84	Unused		
88	External new PSW				
96	Supervisor call new PSW				
104	Program new PSW				
112	Machine check new PSW				
120	I/O new PSW				
128	0	132	0		
136	0	140	0		
144	0	148 0	Monitor class	0	
152	0	156 0	Monitor code		

- Model independent among System/360 and System/370 models in BC mode except for PSW bit 12
- Processed by the control program

EC MODE FIXED AREA 0–159

0	IPL PSW						
8	IPL CCW 1						
16	IPL CCW 2						
24	External old PSW						
32	Supervisor call old PSW						
40	Program old PSW						
48	Machine check old PSW						
56	I/O old PSW						
64	Channel status word – CSW						
72	Channel address word – CAW			76	Unused		
80	Interval timer			84	Unused		
88	External new PSW						
96	Supervisor call new PSW						
104	Program new PSW						
112	Machine check new PSW						
120	I/O new PSW						
128	0			132	0	External int. code	
136	0	ILC	SVC int. code	140 0	ILC	Program int. code	
144	0	Translation excp. addr.		148 0	Monitor class	PER code	0
152	0	PER address		156 0	Monitor code		

- Model independent among System/370 models in EC mode
- PSW format is different from that of BC mode PSW
- Processed by the control program

Figure 20.10.2. Model 158 model-independent fixed storage locations for BC and EC modes

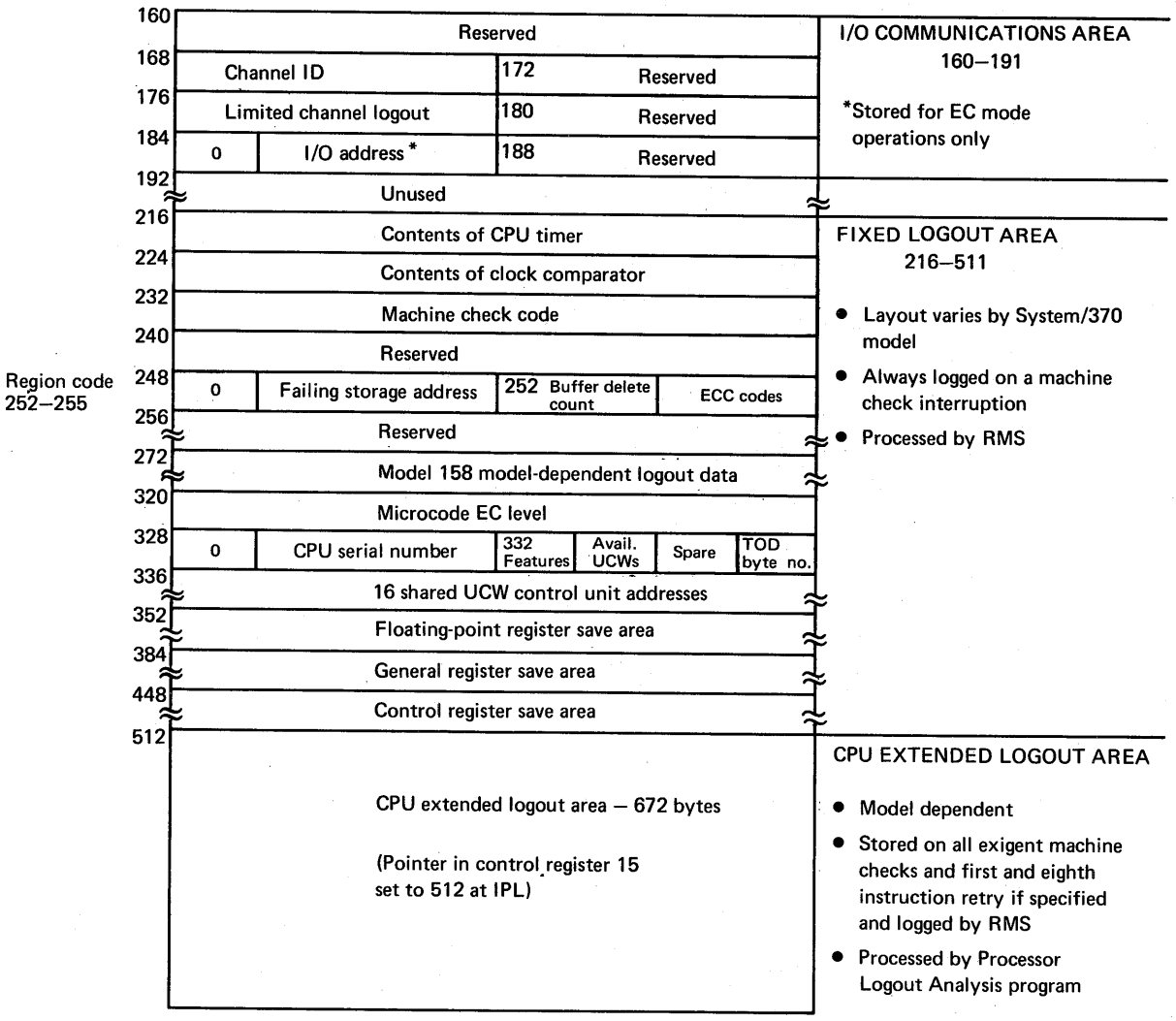


Figure 20.10.3. Model 158 (Model 1) model-dependent fixed storage locations

Program Interruption for Set System Mask Instruction

When a System/370 is operating in EC mode, execution of the SET SYSTEM MASK instruction is under the control of the SSM mask bit in control register 0. When the SSM mask bit is a one, an attempt to execute an SSM instruction causes a program interruption without execution of the SSM instruction. When the SSM mask bit is a zero, SSM instructions are executed as usual.

This interruption is implemented to enable existing programs that were written for System/360 models or for System/370 BC mode of operation to execute correctly in EC mode without modification of the system mask field addressed by existing SSM instructions. When an interruption for an SSM instruction occurs, the contents of the BC mode format system mask indicated by the SSM instruction can be inspected, and the appropriate EC mode mask bits can then be set by an SSM simulation routine.

Expansion of Storage Protect Key Size

The size of the storage protect key associated with each 2K storage block is expanded from five to seven bits in the Model 158. The two additional bits (reference and change) are included for use with dynamic address translation and are discussed in Section 30:10. The SET STORAGE KEY instruction sets a seven-bit key regardless of the mode, BC or EC, in effect. The INSERT STORAGE KEY loads a five-bit or a seven-bit key into the register indicated, depending on whether BC or EC mode, respectively, is in effect.

Program Event Recording

Program event recording (PER), a standard feature of the Model 158, is designed to assist in program debugging by enabling a program to be alerted to any combination of the following events via a program interruption:

- Successful execution of any type of branch instruction
- Alteration of the contents of the general registers designated by the user
- Fetching of an instruction from a processor storage area defined by the user
- Alteration of the contents of a processor storage area defined by the user

The PER feature can operate only when EC mode is in effect and the PER mask, bit 1 of the current PSW, is on. Control register 9 (bits 0 to 3) is used to specify which of the four PER event types are to be monitored. A PER program interruption is taken after the occurrence of an event only if both the PER mask bit and the applicable event mask bit in control register 9 are on. Control register 9 (bits 16 to 31) also specifies which of the 16 general registers are to be monitored if monitoring of this event is specified. Control registers 10 and 11 indicate the beginning address and the ending address, respectively, of the contiguous processor storage area that is to be monitored for instruction fetching and/or alteration.

When an event that is being monitored is detected, PER hardware causes a program interruption, if the PER mask bit is on, and identification of the type of event is stored in the fixed processor storage area (location 150). The address of the instruction associated with the event is also stored (locations 153 to 155). Program event interruptions are lost if they occur when the PER mask bit or the particular event mask bit is off.

If dynamic address translation mode is also specified when PER is active, virtual storage addresses instead of real storage addresses (discussed in Section 30) are placed in the control registers to monitor references to a contiguous virtual storage area.

Note that significant CPU performance degradation occurs when successful branches, storage alterations, or general register alterations are being monitored.

Both the PER facility and the monitoring feature are provided for debugging purposes. The two features differ from one another in (1) the number of events that can be defined, (2) whether the events are defined by the hardware or the programmer, and (3) whether the hardware or the programmer checks for the events and causes the interruptions. When PER is used, once the events to be monitored have been designated by the user, CPU hardware checks for the occurrence of the events and causes

the interruption. When the monitoring feature is used, the user defines the events to be monitored (up to 16 classes with up to 16 million monitor codes each instead of only four events), determines when the events occur, and causes program interruptions by issuing MONITOR CALL instructions.

NEW INSTRUCTIONS

STORE THEN AND SYSTEM MASK and STORE THEN OR SYSTEM MASK are two new privileged instructions that affect the system mask (bits 0 to 7 in the current PSW). The STORE THEN AND SYSTEM MASK instruction provides, via a single instruction, the capability of storing the current system mask for later restoration, while selectively zeroing certain system mask bits. The STORE THEN OR SYSTEM MASK provides system mask storing and selective setting of system mask bits to ones. These two instructions simplify the coding required to alter the system mask, particularly when the existing settings must be saved.

COMPARE AND SWAP and COMPARE DOUBLE AND SWAP instructions provide the capability of controlling access to a shared real storage area in a multiprogramming or multiprocessing environment. Although the TEST AND SET instruction can also be used for this purpose, these compare instructions enable a program to leave a message when the shared area is in use. This message can be inspected, via a compare and swap instruction, by other programs that share the real storage area. The virtual telecommunication access method (VTAM), OS/VS2 Releases 2 and up, and VSAM Release 2 use these two instructions.

The INSERT PSW KEY privileged instruction enables a program to place in general register 2 the four-bit storage protection key from the current PSW. The SET PSW KEY FROM ADDRESS privileged instruction enables a program to place a protect key contained in general register 2 or processor storage in the current PSW. When a control program is requested to access a given processor storage location by a problem program, these two instructions can be used by the control program during its processing of the request to determine whether or not the problem program is authorized to access the specified processor storage location.

The CLEAR I/O privileged instruction can be used together with the HALT DEVICE instruction to terminate all I/O activity on a given channel. CLEAR I/O, INSERT PSW KEY, and SET PSW KEY FROM ADDRESS are used by OS/VS2 Releases 2 and up.

The new instructions discussed above are provided in the Model 155 II also. Other new instructions provided for the Model 158 are related to specific features (such as dynamic address translation, the CPU timer, and the clock comparator) and are discussed with these features.

IMPROVED INSTRUCTION EXECUTION SPEED

Additional hardware is provided in the Model 158 to improve the execution speed of certain classes of operations, when compared with the Model 155. These classes include the following:

- All binary, floating-point, and decimal multiply operations
- All eight general register shift operations
- Certain move character operations
- Convert to binary and convert to decimal operations

The Model 158 also has improved instruction prefetching via the use of a 64-word buffer in addition to three one-word buffers.

CLOCK COMPARATOR AND CPU TIMER

These timing facilities are standard on the Model 158. (They are also provided in a Model 155 II.) The clock comparator provides a means of causing an external interruption when the time of day clock has passed a time specified by a program. This feature can be used to initiate an action, terminate an operation, or inspect an activity, for example, at specific clock times during system operation.

The clock comparator has the same format as the time of day clock and is set to zero during initial program reset. The SET CLOCK COMPARATOR privileged instruction is provided to place a value that represents a time of day in the clock comparator. When clock comparator interruptions are specified via the external interruption summary mask bit in the current PSW and the clock comparator subclass mask bit in control register 0, an external interruption occurs when the time of day clock value is greater than the clock comparator value. Bits 0 to 51 of the time of day clock and the clock comparator are compared. If clock comparator interruptions are masked when this condition occurs, the interruption remains pending only as long as the time of day clock value remains higher than the value in the clock comparator. The STORE CLOCK COMPARATOR privileged instruction can be used to obtain the current value of the clock comparator.

The use of a clock comparator to cause an interruption when a specified time is passed, instead of using the interval timer at location 80, offers two advantages. First, the time of day clock increments when the system is in the stopped state while the interval timer does not. Hence, if a system stop occurs during processing and the system is restarted, the clock comparator can still cause an interruption at the time requested. The interruption caused by the interval timer in such a situation is late. Second, implementing the time of day clock and the clock comparator in the same doubleword format eliminates having to convert doubleword time of day clock units to single-word interval timer units.

The CPU timer provides a means of causing an external interruption when an interval of time specified by a program has elapsed. The CPU timer is implemented as a binary counter with a format identical to that of the time of day clock; however, bit 0 of the CPU timer is considered to be a sign. The CPU timer has a maximum time period half as large as that of the time of day clock and the same resolution of one microsecond. When both the CPU timer and the time of day clock are running, the stepping rates of the two are synchronized so that they are stepped at exactly the same rate.

The CPU timer is set to zero at initial program reset, and the SET CPU TIMER privileged instruction is provided to place an interval of time in the CPU timer. The STORE CPU TIMER privileged instruction can be used to obtain the current CPU timer value. The CPU timer decrements every microsecond. If the external interruption summary mask bit in the current PSW and the CPU timer subclass mask bit in control register 0 are on, an external interruption occurs whenever the CPU timer value is negative (not just when the timer goes from positive to negative), indicating that the time interval has elapsed. The CPU timer decrements when the CPU is executing instructions (including instruction retry operations) and while the CPU is in the wait state. It is not decremented when the system is in the stopped state.

While providing essentially the same function as the interval timer at location 80, the CPU timer provides advantages over the interval timer as follows: Task processing intervals of less than 3.3 milliseconds are accurately measured because of the one-microsecond resolution of the CPU timer. A pending CPU timer interruption is reset when a SET CPU TIMER instruction is issued to set a positive value in

the CPU timer, eliminating the need to take an interruption in order to reset the CPU timer, as is required for the interval timer.

In addition, the amount of timing facilities processing required during a task switch can be reduced. This can result from the fact that the format of the time of day clock and the CPU timer are the same. Conversion of doubleword time of day clock values to single-word interval timer values is eliminated, and timer queues can be structured so that little of the processing currently required during a task switch, when the interval timer is used, is necessary.

RELIABILITY, AVAILABILITY, AND SERVICEABILITY FEATURES

All the hardware RAS features that are implemented in the Model 155 are also implemented in the Model 158:

- Automatic retry of most failing CPU operations by hardware.
- ECC checking on processor storage to correct all single-bit and detect all double-bit errors.
- Automatic deletion of malfunctioning buffer halfblock locations, index locations, and the entire buffer when necessary.
- I/O operation retry facilities, including the storing of channel retry data during an I/O interruption that results from an error, and channel/control unit command retry procedures to correct certain failing I/O operations.
- Expanded machine check interruption to facilitate error recording and recovery procedures. The area reserved for Model 158 CPU extended logouts is 992 bytes, the same as for the Model 155. Currently, both models log 672 bytes.

All of the above operate in exactly the same manner in the Model 158 as they do in the Model 155 except for the following:

- Machine checks that result from a single-bit processor storage error correction are controlled by an ECC mode bit in addition to the recovery mask bit and PSW bit 13. The ECC mode bit is turned off during a system reset and is not turned on by recovery routines for the Model 158. The normal mode of operation for the Model 158 is to run disabled for single-bit processor storage error correction interruptions. The ECC mode bit is provided to permit the customer engineer to enable these interruptions during maintenance operations.
- The time of day clock damage interruption, maskable by the external mask bit and PSW bit 13, is expanded to include clock comparator and CPU timer errors. Its name is changed to "Timing Facilities Damage". When a STORE CLOCK COMPARATOR or a STORE CPU TIMER instruction is issued and the addressed timing facility has an error or when the CPU timer or the clock comparator develops an error, a timing facilities damage interruption occurs when the timing facilities damage mask bit is one. (This applies to the Model 155 II also.)
- Whenever a machine check interruption occurs in a Model 158, the current value of the CPU timer is stored in locations 216 to 223, and the current value of the clock comparator is stored in locations 224 to 231. Bits 46 and 47 of the machine check code, shown in Figure 20.10.4, are used to indicate that these values were stored correctly. (This applies to the Model 155 II also.)

- Uncorrectable errors in the storage protect array will be indicated by a one in bit 18 of the machine check code.

- Additional fields are defined within the model-dependent fixed logout area (in locations 272 to 351), as shown in Figure 20.10.3.

- A warning machine check interruption is implemented in Model 158 systems with the optional Power Warning feature installed. This field-installable feature can be used in Model 158 systems that have O.E.M. uninterruptable power supplies (UPS). A UPS is designed to protect a system from power line disturbances by providing auxiliary power for a specified interval of time during a power reduction or outage.

  A system can be fully or partially protected. Full protection involves supplying a UPS for all system components. This support provides continuous system operation for a specified interval of time during a power line disturbance. Partial protection involves supplying a UPS only for a critical subset of system components, namely, the CPU, processor storage, and one channel and its attached control units. The Power Warning feature can be used with partially and fully protected Model 158 systems.

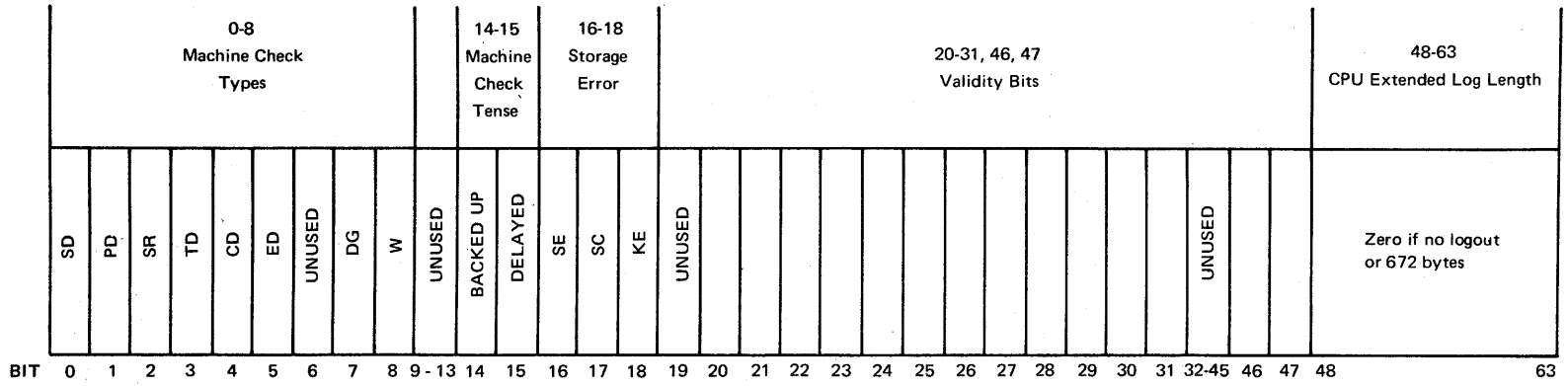
  A UPS for a Model 158 must generate a power warning signal when an under voltage condition of 18% ($\pm$ 2%) is detected. A Model 158 CPU with the Power Warning feature recognizes this signal. If bit 13 in the current PSW and the warning submask (bit 7 in control register 14) are both on, a warning repressible machine check interruption occurs. Bit 8 in the stored machine check interruption code will be on to indicate a warning condition. The machine check handler (MCH) routine is given CPU control to process the interruption. If either mask bit is off, the warning interruption remains pending.

  The Power Warning feature is designed to enable a Model 158 system to terminate operations in an orderly manner when a power line disturbance or power shutdown occurs. When a warning interruption occurs, a determination can be made as to whether or not the power line disturbance is transient. Operation of a fully protected system need not be terminated for a transient disturbance of a short enough duration. If system termination is required, a complete processor storage dump can be taken first. This enables processor storage to be restored when a system restart is performed at a later time.

Model 158 recovery management routines (machine check and channel check handlers) are included in OS MFT and OS MVT as of Release 21.6. They provide recovery facilities comparable to those provided for the Model 155 and support new Model 158 machine check facilities, except for MFT, which does not support the Power Warning feature. These Model 158 recovery routines are also included in OS/VS1 and OS/VS2 and are modified to operate correctly when the Model 158 is operating in EC and dynamic address translation modes. Model 158 recovery management routines are also provided in DOS/VS. A discussion of how these recovery routines differ from those provided for BC mode operations is contained in each optional programming systems supplement.

In addition to the RAS features implemented in the CPU, the Model 158 system contains a service processor unit as a standard feature. Part of its function is to replace the optional 2955 Remote Analysis Unit that is available for the Model 155 to provide a remote analysis capability. The service processor is utilized by the Remote Support Facility, which significantly extends the remote problem analysis capability for the Model 158. This capability is discussed in Section 20:30.

Fixed Logout Area Locations 232-239



Subclass	Bit	Interruption Type
Internal	0	SD— System Damage
	1	PD— Instruction Processing Damage
	2	SR— System Recovery
	3	TD— Timer Damage
External	4	CD— Timing Facilities Damage
	5	ED— External Damage
	6	Unused
Alert	7	DG— Degradation
	8	W— Warning

Bit	Storage Error
16	Storage Error Uncorrected
17	Storage Error Corrected
18	Key in Storage Error Uncorrected

Bit	Valid Fixed Area Data
20-23	Machine Check Old PSW (48-55) 20 AMWP 21 Masks and Protect Key 22 Program Mask and Condition Code 23 Instruction Address
24	Failing Storage Address (248, 249)
25	Region Code (252-255)
26	Unused
27	Floating Point Registers (352-383)
28	General Registers (384-447)
29	Control Registers (448-511)
30	CPU Extended Logout
31	Storage (Validity of processor storage being processed by instructions when interruption occured)
46	CPU Timer Value
47	Clock Comparator Value

Figure 20.10.4. Model 158 machine check code

PROCESSOR (MAIN) STORAGE

Like the Model 155, the Model 158 has a two-level storage system in which large high-speed processor storage backs up small, higher-speed monolithic buffer storage. Processor storage is available for the Model 158 as follows:

Model	Capacity
I	512K
J	1024K
JI	1536K
K	2048K
KJ	3072K
L	4096K

The path to processor storage in the Model 158 is 16 bytes wide. The cycle time of processor storage is based on the operation being performed. The CPU can fetch eight bytes from processor storage in 805 nanoseconds and begin using four of the bytes read, as follows: Successive reads of processor storage furnish 16 bytes (4 words) every 1035 nanoseconds, which includes data fetch and buffer update time. The first eight bytes are delivered to the buffer in 805 nanoseconds, and the second eight bytes arrive 115 nanoseconds later.

During the 805 nanoseconds required to make eight bytes from processor storage available in the buffer, the first four of these bytes are also made available to a CPU data register. If they are required, the CPU can fetch the next four bytes in 115 nanoseconds. The CPU can fetch the third word of the 16 bytes fetched in another 230 nanoseconds and the fourth word in another 115 nanoseconds for a total of 16 bytes in 1265 nanoseconds.

The CPU can store eight bytes of data in processor storage every 690 nanoseconds if the data is written on a doubleword processor storage boundary. Any other type of write--a write of 1 to 7 bytes, 9 to 16 bytes, or 8 bytes that are not placed on a doubleword boundary--is a partial write and requires 920 nanoseconds. A partial write involves reading before writing except when 16 bytes are written. Model 158 cycle and access times are summarized in Table 20.15.1.

Table 20.15.1. Model 158 Model 1 cycle and access times

Cycle or Access Time	Time in Nanoseconds
CPU cycle time	115
Processor storage read cycle time (successive reads of 16 bytes)	1035
CPU fetch of 8 bytes from processor storage to buffer	805
CPU fetch from processor storage to a data register	
4 bytes	805
8 bytes	920
12 bytes	1150
16 bytes	1265
Processor storage cycle time for a doubleword write (8 bytes on a doubleword boundary)	690
Processor storage cycle time for a partial write (1-7 bytes, 9-16 bytes, 8 bytes not on a doubleword boundary)	920
CPU fetch from buffer	
4 bytes	230
8 bytes	345

HIGH-SPEED BUFFER STORAGE

As in the Model 155, an 8K high-speed buffer is standard in the Model 158 (Model 1). Buffer storage provides high-speed data access for CPU fetches. In a Model 158, as in a Model 155, the CPU can obtain four bytes from the buffer in 230 nanoseconds (two CPU cycles) or eight bytes in 345 nanoseconds (three CPU cycles). If the buffer does not contain the data required, the data must be obtained from processor storage.

Use of the high-speed buffer in Models 158 and 155 is the same. When a data fetch request is made by the CPU, a determination is made of whether the requested data is in the high-speed buffer by interrogation of the index array of the buffer's contents. If the data requested is present in the buffer and is valid, it is sent directly to the CPU without a processor storage reference. If the requested data is not currently in the buffer, a processor storage fetch is made and the data obtained is sent to the CPU. The data is also assigned a buffer location and stored in the buffer. When data is stored by the CPU, both the buffer and processor storage are updated if the contents of the processor storage location being altered are currently being maintained in the buffer.

The channels never access the buffer directly. They read into and write from processor storage using a direct path between the CPU and processor storage that bypasses the buffer. When a channel stores data in processor storage, the index array is inspected. If the data from the affected processor storage address is being maintained in the buffer, appropriate bits are set in the index array to indicate that this buffer data is no longer valid.

The algorithm implemented in Model 1 of the Model 158 to maintain the contents of the high-speed buffer differs from the algorithm implemented in the Model 155 and is designed to enable the Model 158 CPU to fetch data from the buffer a slightly higher percentage of the time than is achieved by the Model 155 CPU.

The 8K buffer in the Model 158 is divided into two 4K compartments each of which can contain 256 halfblocks (one row) of data, as shown in Figure 20.15.1. Processor storage is divided in 4K-byte rows of 256 halfblocks as well. The number of rows in processor storage is a function of processor storage size.

The index array for the Model 158 buffer, also shown in Figure 20.15.1, has 256 halfblock address locations. Each location contains two index entries, one for the corresponding halfblock in the upper compartment of the buffer and one for the corresponding halfblock in the lower compartment. An index entry contains a processor storage row address (bits 8 to 19 of the processor storage address of the halfblock in the buffer), two valid bits, and one parity bit. The valid bits, one for the corresponding halfblock in each compartment, indicate the presence of absence of valid data in the corresponding halfblock buffer location. A valid bit is turned on when data is placed in the buffer.

One OK bit and one least recently used (LRU) bit are associated with each of the 256 halfback locations in the index array. The OK bit indicates the corresponding halfblocks of the buffer and index array are functioning correctly. The LRU bit is sometimes used to determine which compartment is assigned to a halfblock when it is loaded into the buffer. During system resent or IPL, all valid bits are set off, all OK bits are set on, and all processor storage addresses in the index entries are set to zero.

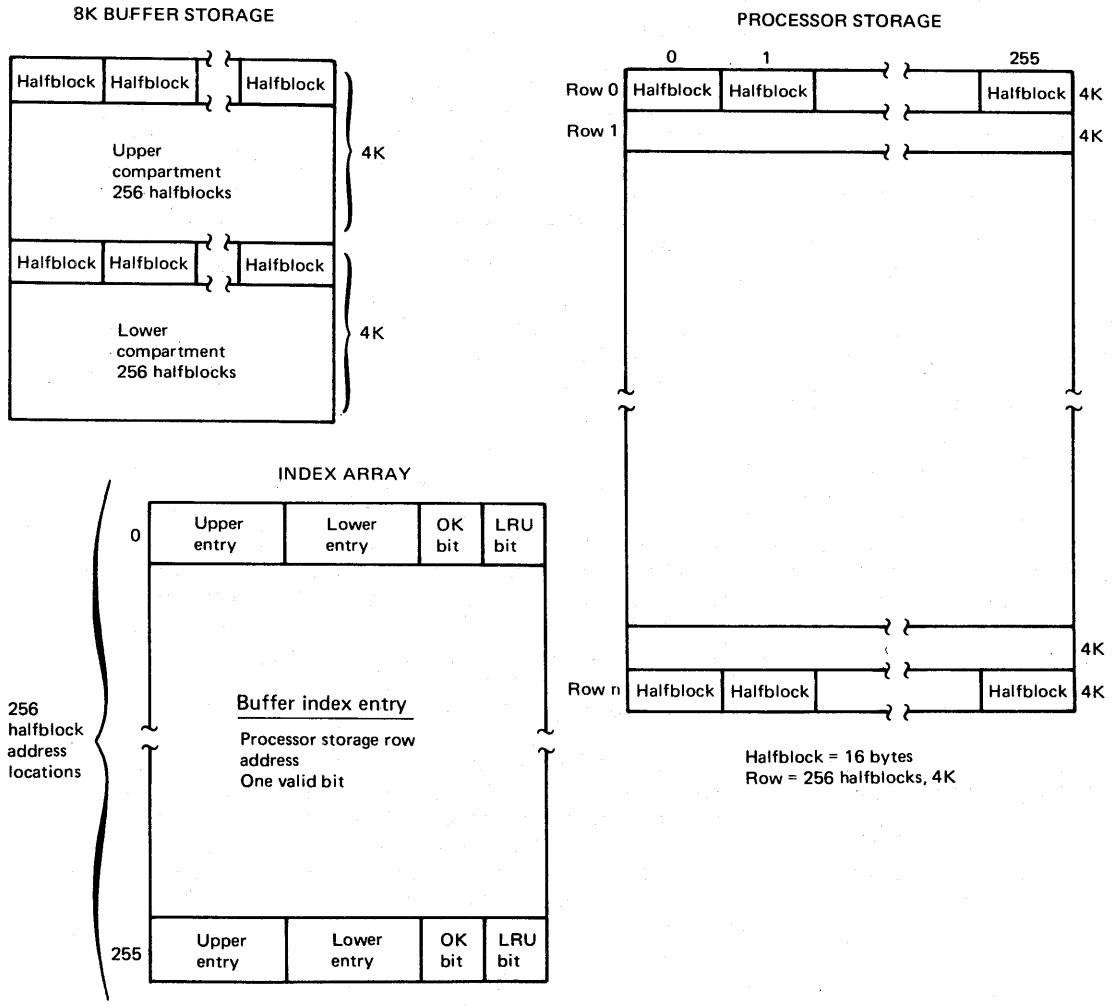


Figure 20.15.1. High-speed buffer organization in the Model 1

Buffer space is reserved and loaded on a halfblock basis as follows. When a halfblock is to be loaded, the index array entry for the halfblock address of the data is inspected to determine which buffer compartment to assign. If only one valid bit is on, the halfblock is assigned the compartment whose valid bit is off. If both valid bits are on or both are off, the LRU bit setting determines the compartment assigned. If the LRU bit is on, the upper compartment is assigned. The lower compartment is assigned if the LRU bit is off.

The LRU bit setting is changed when the data being referenced is found in the buffer or a buffer assignment is made. When the referenced data is found in the buffer, the LRU bit for the associated index entry halfblock location is set to point to the compartment that does not contain the data. Similarly, when a halfblock is assigned to a compartment, the LRU bit for the associated index entry location is set to point to the compartment not assigned.

RELOADABLE CONTROL STORAGE

CPU functions and channel operations are microprogram-controlled in the Model 158, just as they are in the Model 155. However, monolithic reloadable control storage (RCS) is implemented in the Model 158 for microprogram residence. The Model 155 uses read-only control storage.

As implemented in the Model 158, use of reloadable control storage instead of read-only storage results in improved serviceability. Serviceability is enhanced because of the speed and ease of microcode-only engineering change installation. The new microcode need only be loaded into control storage to install such an engineering change. In addition, more extensive diagnostics can be provided without the necessity of adding control storage just for diagnostic residence. Control storage can be used to contain diagnostic routines during maintenance procedures, because diagnostics can overlay normal system microcode. Microcode-only features, such as the compatibility features, can be made available on the Model 158 on a no-charge basis.

The Model 158 contains two units of reloadable monolithic control storage, each of which has a capacity of 4096 72-bit control words. Reloadable control storage is separate from processor storage and is parity checked for validity.

Control storage is loaded directly from a small disk device called the console file, which is a basic component of the Model 158. This console file is located behind the front covers of the operator display console and is also called the inboard file. It reads removable, prerecorded disk cartridges. The console file is similar to the read-only disk device that is used in a Model 155 to load microdiagnostic tests. However, the console file in the Model 158 is faster and a little larger than the file in the Model 155. In addition, the Model 158 console file uses a disk cartridge that has a considerably larger capacity than the disk cartridge used with the Model 155 console file.

Reading from the console file begins when the initial microprogram load (IMPL) procedure is initiated. An IMPL is initiated automatically when system power is turned on and can be initiated by the operator from the console panel on the CPU when power is already on. There are no I/O instructions or commands that a user program can execute to read from or write on a console file disk cartridge. When power is already on, IMPL is initiated by pressing the IMPL key.

After loading is completed, considerable error checking occurs to ensure that the microcode was read and loaded into control storage correctly. A hash total is accumulated that is checked against the hash total read from the disk cartridge. This ensures that control storage has been correctly loaded. The system reset microcode (just loaded) is executed, and the CPU is placed in the stopped state, ready for an IPL of the operating system.

Prewritten disk cartridges containing all the microcode required to support the standard and the user-specified optional features of a Model 158 are shipped to each installation. One standard and one backup cartridge that contain all the microcode required for the system (except that for the optional Virtual Machine Assist feature) are provided, in addition to other disks that contain system diagnostics (and the Virtual Machine Assist feature microcode). The disk cartridge that contains the system microcode is called the IMPL disk or the S-disk and must be mounted on the inboard file in order for the Model 158 to operate.

There is no defined procedure for temporarily loading engineering change patches to existing microcode that has been loaded into RCS. When an engineering change is required, new disk cartridges containing the altered microcode will be distributed. An IMPL disk is portable among Model 158 systems with the same hardware configuration and at the same engineering change level.

Note that when system power is turned off, the data in both processor and control storage is lost, so an IMPL must be performed when power is turned on again. The IMPL is performed automatically during a power on.

STORAGE CONTROL UNIT

The storage control unit (SCU) provides the interface between processor storage and other hardware components of the system (CPU, channels, etc.). The SCU contains the following:

- High-speed buffer storage and associated index array
- Hardware for translating virtual storage addresses into real storage addresses
- The storage protect key array for processor storage
- Storage data register (a 16-byte register that is used to transfer data to and from system components)
- ECC logic
- PER logic
- The time of day clock and CPU timer

20:20 CHANNELS

The channel configurations available for the Model 158 are identical to those available for the Model 155. That is, one byte multiplexer channel and two block multiplexer channels are standard. Three additional channels can be installed--all block multiplexer or one byte multiplexer and two block multiplexer channels. When installed, the second byte multiplexer channel replaces block multiplexer channel 4.

Block multiplexer channels can also operate in selector channel mode, and shared subchannels are available to permit operation of selector channel devices when in the normal block multiplexer mode. All channels are integrated, rather than standalone, and a maximum of one Channel-to-Channel Adapter can be installed on the Model 158.

Channel operation, channel addressing and priorities, channel data rates, and RAS features are the same for Models 158 and 155. However, the maximum I/O device configuration that can operate without overrun in a Model 158 configuration provides a maximum block multiplexer channel aggregate data rate of 6.75 million bytes per second (MB/sec), instead of 5.4 MB/sec. This rate is achieved when two 2305 Model 2 modules and three 3420 Model 8 tape units operate concurrently on five block multiplexer channels and can be supported by a Model 158 operating in BC mode or in EC and dynamic address translation modes.

The only significant difference between the channels of a Model 158 and a Model 155 is a change in UCW (subchannel) availability. The number of UCW's available for a Model 158 is not a function of the amount of processor storage installed, as it is for the Model 155. The total number of UCW's present in a Model 158 is the same for any processor storage size.

The number of shared and nonshared subchannels available for the byte multiplexer channel(s) depends on whether the subchannel sharing capability is inhibited or permitted. At installation time, the customer engineer independently wires byte multiplexer channel 0 and, if present, byte multiplexer channel 4 to permit or inhibit the use of shared subchannels. Byte multiplexer channel 0 will always have 256 nonshared subchannels (for device addresses 00 to FF) and no shared subchannels if subchannel sharing is inhibited for channel 0. The same applies to byte multplexer channel 4. If subchannel sharing is allowed

for channel 0, it will have 120 nonshared subchannels and 8 shared subchannels. The same is true for byte multiplexer channel 4.

Each shared subchannel of a byte multiplexer channel has 16 contiguous device addresses (X0 to XF) associated with it where X varies from 8 to F. Devices with a 1 in the high-order bit position of their device address are assigned shared subchannels of a byte multiplexer channel. This permits a control unit to have up to sixteen I/O devices that share the same subchannel.

When subchannel sharing is permitted, the eight shared subchannels use the same UCW's as the first eight nonshared subchannels. Therefore, the following device addresses for a byte multiplexer channel are mutually exclusive when subchannel sharing is allowed:

80-8F and 00
90-9F and 01
A0-AF and 02
B0-BF and 03
C0-CF and 04
D0-DF and 05
E0-EF and 06
F0-FF and 07

The block multiplexer channels present in a Model 158 share 512 UCW's. As in a Model 155, these 512 UCW locations are located in bump storage within processor storage in the Model 158 and are divided into 480 nonshared UCW's and 16 shared UCW's. UCW (subchannel) assignment for block multiplexer channels is the same in Models 158 and 155. That is, shared subchannels have preassigned addresses, which are established by the customer engineer, while nonshared subchannels are assigned dynamically during system operation. A shared subchannel is associated with 16 device addresses in the range of X0 to XF.

The 16 shared subchannels available can be allocated among the installed block multiplexer channels in any desired way. The devices attached to shared subchannels of block multiplexer channels operate in selector made (no disconnection during command chained operations).

20:25 SYSTEM CONSOLES

STANDARD DISPLAY CONSOLE

Unlike the Model 155, the Model 158 has a display console with a keybcard and a light pen as the standard system console device. The 3210 Printer-Keyboard (Models 1 and 2), and the 3215 Printer-Keyboard Model 1 cannot be installed on a Model 158 as the primary console device. The display console on the Model 158 is used for operator/system communication, diagnostic functions, and displaying data that for a Model 155 is displayed via lights on the system control panel on the front of the CPU. As shown in Figure 20.25.1, the front panel of a Model 158 CPU has a small conscle panel in the lower right-hand corner and does not contain the lights that are present on the Model 155 CPU system console panel.

The Model 158 operator console consists of a display tube mounted on a reading board, a keyboard positioned in front of the display tube, a light pen located at the right-hand side of the display tube (attached via a wire), and a small panel that contains pushbuttons, switches, and indicators.

Up to 25 lines of 80 characters per line can be displayed on the display tube simultaneously. The keyboard is provided to allow manual data entry. The light pen provides a means of communicating with the system via the console without using buttons or switches and without

manually entering data via the keyboard. When the light pen is pressed against a control character that is displayed on the screen of the display tube, a signal is generated to cause the required action to be taken. The light pen and the display tube provide a faster method of performing operator functions and faster message display than a typewriter console when used in display mode.

The keyboard is similar to a 1052 Printer-Keyboard in physical layout. It has 26 alphabetic, 10 numeric, and 26 special character keys, as well as 10 keys for display tube cursor control and editing. In addition to cancel, enter, and request keys, the keyboard has start, stop, external interrupt, copy, and mode select keys, and four keys for moving the display tube cursor right, left, up, or down. The copy key is used to cause the data being displayed to be written to the 3213 printer if it is present.

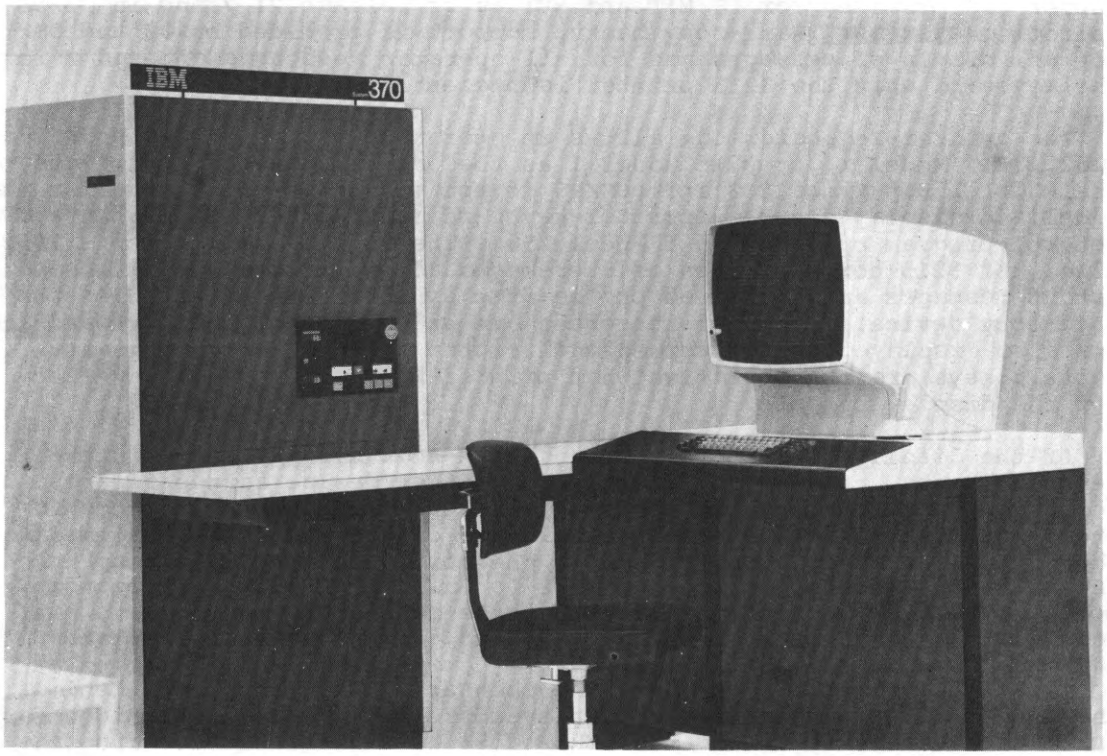


Figure 20.25.1. Model 158 display console (design model)

The console panel in the lower right-hand corner of the front CPU frame contains several buttons, switches, and indicators. They are the power-on and power-off pushbuttons and indicators, the emergency power-off pull switch, the time of day clock toggle switch, the IMPL pushbutton, the remote pushbutton, and a lamptest pushbutton. A CPU usage meter and a service meter are located below the console panel.

Other operator and maintenance functions are performed using the display console, light pen, and keyboard.

The display console is microprogram controlled. I/O operations for the display console are controlled by the service processor unit, which is discussed in Section 20:30. Microcode for the display console is contained in a portion of the reloadable control storage that is provided in the service processor. Display console microcode is loaded into the service processor from the S-disk on the inboard file during a power-on sequence or an IMPL.

The display console can be used in display mode or in printer-keyboard mode. The system microcode disk cartridge contains both display mode and printer-keyboard mode microcode for the display console. An IMPL is required to change from either mode to the other. Optionally, a 3213 Printer (85-cps maximum print speed) can be installed to provide hard copy when display mode is used. The IBM-supplied operating systems that support the Model 158 require the installation of the 3213 Printer when the display console is operated in printer-keyboard mode.

In order to operate the Model 158 console in display mode in an OS or OS/VS, environment, Device Independent Display Operator Console Support (DIDOCS) for the 3270 must be included in the control program. This support is provided for OS MFT and MVT as of Release 21.7 and in OS/VS operating systems. Hard-copy backup support is provided by OS and OS/VS for all displayed messages and for all operator responses entered using the keyboard when the 3213 printer is present.

The printer-keyboard mode allows an operating system that supports a 1052, 3210 Model 1, or 3215 Model 1 as the primary console to operate on the Model 158. Since DOS and DOS/VS do not support display mode for the Model 158 display console, printer-keyboard mode must be used when a DOS control program operates on a Model 158. The display unit accepts 1052, 3210, and 3215 commands when printer-keyboard mode microcode is loaded. Output messages are displayed on the display screen and printed on the hard-copy device. Operator responses are entered using the keyboard and are also displayed on the screen and written on the hard-copy device. Certain keys are not operative when printer-keyboard mode is in effect for the display console.

If the display console is to be used in both display and printer-keyboard modes, three byte multiplexer channel device addresses are required. When display mode is used, the preferred device address for the display is 010 or 014, and address 011 or 015 should be assigned to the 3213 Printer if it is present in the configuration. The display should be assigned address 009 or 01F for use with printer-keyboard mode. A control unit position on the byte multiplexer channel is not required by the display unit, the keyboard, or the hard-copy printer.

A second console file, called the outboard file, is located behind the inboard file and under the display console. The disk cartridge for the outboard file is called the SEREP disk or N-disk. It contains the SEREP program, blank tracks, and, optionally, the Virtual Machine Assist feature microcode (discussed in Section 40:05). Extended machine check logout data is automatically written on the blank tracks of this disk during a machine check interruption. The logout data is written to the outboard file by hardware (the service processor) without programming support.

Up to eight logouts can be recorded on one N-disk. The logouts are recorded on a rotating basis so that the disk always contains the last eight logouts. This logout data is still recorded in the log data set (SYS1.LOGREC in OS, SYSREC in DOS) by recovery management support routines. The logout to the outboard file provides backup in the event

that programmed logout is not possible because of the nature of the error and can be accessed by a local or remote customer engineer using the display console without programming assistance.

When display mode is in effect, the operator communicates operator control functions to the display console via the light pen or the keyboard. The functions that the operator can perform are divided into four groups. The functions in each group are represented by a frame of text that is displayed on the screen of the display console. The text in a frame contains the names of functions the operator can perform. Each function name has a lozenge (■) displayed to the left of it. In addition, each function name has a letter/number combination associated with it, which is also displayed. The operator selects the function to be performed by pressing the light pen against the lozenge associated with the function name or by entering the appropriate letter/number combination via the keyboard.

The four operator control displays provided are the configuration frame, manual frame, program frame, and alter/display frame. Other frames are provided for customer engineer use (see Section 20:30). The configuration frame is displayed automatically after an IMPL is performed and can be selected when the manual frame is being displayed.

The configuration frame displays the configuration of the system. Included in this display are the optional CPU features installed, size of processor storage, channel configuration, console device addresses, disabled adapters (such as the ISC and channel-to-channel adapters), disabled timer features, a list of the shared UCW addresses, and the console mode in effect.

The configuration frame also provides the operator with the capability of disabling the interval timer, disabling the 3213 printer, disabling the power warning feature, disabling the optional 3056 Remote System Console, selecting printer-keyboard mode for the display console, and loading VMA feature microcode. The system is automatically placed in display mode after an IMPL. If printer-keyboard mode is to be used, it must be selected after IMPL using the configuration frame. (Loading VMA feature microcode is discussed in Section 40.)

The manual or service frame can be selected for display from the configuration frame after the operator has made any required configuration changes. The manual frame can also be selected and displayed by pressing the mode select key on the keyboard. The manual frame displays the current state of the system and also contains PSW restart, restart, system reset, load, store status, system reset/clear, and load/clear functions. The store status function is not provided in a Model 155. (The store status function can be performed on a Model 155 II using the 3210 Model 1 or 3215 Printer-Keyboard.)

By initiating the store status function, the operator can cause the following to be placed in processor storage in the Model 158:

CPU timer value - locations 216-223

Clock comparator value - locations 224-231

Current PSW contents - locations 256-263

Address of LEX list (associated with DOS Emulator) - locations 268-271

Contents of the floating-point registers - locations 352-383

Contents of the general registers - locations 384-447

Contents of the control registers - locations 448-511

The operator should perform the store status function to preserve system status after an error causes a system halt and before resetting the system to load a standalone storage dump program. Otherwise, the contents of these fields and registers at the time the halt occurred are lost during the reset that is performed to IPL the dump program. The IBM supplied standalone dump program is modified to obtain system status information from the locations indicated.

The manual frame can be used to select the other frames (program, service, alter/display, and configuration). Processor control modes (run or instruction step) can also be set using the manual frame, as can check control mode (process or hard stop). Certain system indicators (system, manual, wait, test) are displayed as well on the manual frame.

The program frame is normally displayed whenever the system is operating in run mode under program control. The program frame is displayed automatically after a successful IPL and can be manually selected from the manual and service frames. In both printer-keyboard mode and display mode without DIDOCS support, up to 24 lines of 80 characters each are used to display operating system messages to the operator. The twenty-fifth line is always used to display certain system indicators (wait state, etc.) and the PSW. When display mode with DIDOCS support is used, the first 24 lines of the program frame are divided into five functional areas:

- Message Area (lines 1-19) - Used to display messages from the operating system and problem programs, certain operator commands, and status displays.

- PFK (Program Function Key) Display Line (line 20) - Contains a display of the PFK numbers that were specified at system generation. These numbers are used when operating system commands are to be entered via the light pen.

- Instruction Line (line 21) - Used to display system messages that pertain to console control.

- Entry Area (lines 22 and 23) - Used by the operator to enter operating system commands and to reply to messages from the operating system and problem programs.

- Warning Line (line 24) - Used to display warning messages that alert the operator to certain conditions and that normally require operator action.

The alter/display frame can be selected by the operator using the manual or service frame and when the CPU is in the stopped (manual) state and the clocks are running. The facilities that can be altered and displayed are listed with a lozenge and a letter code. The operator selects the desired function with the light pen or the keyboard. Hexadecimal digits are displayed also, and data can be entered using the light pen or the keyboard. The contents of the following can be altered and/or displayed: real storage, virtual storage, storage protect keys, real channel UCWs, logical channel UCWs, active UCWs, CPU local storage, I/O UCW local storage, I/O buffer local storage, control registers, general registers, floating-point registers, and the PSW.

A security key is provided on the right-hand side of the display tube. When this key is in the horizontal position, the operator can select any frame as previously described. When the security key is in the vertical position or not inserted, the mode select and interrupt keys are disabled and the operator cannot change from the program frame.

THE 3056 REMOTE SYSTEM CONSOLE

Optionally, the 3056 Remote System Console (RSC) can be attached to a Model 158 via a cable up to 200 feet in length to provide remote operational capability. The 3056 cannot be attached to a Model 155 or 155 II. The field-installable 3056 RSC is a standalone unit that contains a display tube and keyboard identical to those of the standard display console. The 3056 RSC does not have a light pen and the 3213 Printer cannot be installed to provide hard copy backup for the 3056.

The operator has almost all the same operational capabilities when using a 3056 RSC as when using the standard display console. Table 20.25.1 compares the functions offered by these two system consoles.

The 3056 RSC is not attached to a channel in the Model 158 and does not have its own I/O address. It operates in parallel with and is an extension of the standard system console. That is, the 3056 RSC is redundant to the standard display console in that the same display is shown on both units and operator responses to messages can be made from either keyboard. Therefore, the 3056 does not require programming support.

THIS PAGE INTENTIONALLY BLANK

Table 20.25.1. Functional capabilities of the standard display console and the 3056 Remote System Console

Function	Standard Display Console	3056 Remote System Console
Display tube	Yes	Yes
Keyboard	Yes	Yes
Light pen	Yes	No
Hardcopy available	Yes via 3213 Printer (optional)	No
Copy display to printer	Yes	Yes
Security key	Yes	Yes
Power on/off	Yes	No
Interrupt key	Yes	Yes
IMPL key	Yes	No
ISC1 IMPL check indicator	Yes	No
ISC2 IMPL check indicator	Yes	No
Load (IPL) key	Yes	Yes
CPU status indicators	Yes	Yes
Dual intensity	Yes	Yes
System reset function	Yes	Yes
Check reset function	Yes	Yes
PSW restart function	Yes	Yes
Start key	Yes	Yes
Stop key	Yes	Yes
Console check indicator	Yes	No
Processor check	Yes	Yes
Alter/display real storage	Yes	Yes
CE frames	Yes	Yes
Set time of day clock	Yes	No
Audible alarm	Yes	Yes
Program function keys	Yes, activated by light pen or the keyboard	Yes, activated by the keyboard

Operation of the 3056 console is controlled by the setting of the disable keys on the standard display and 3056 consoles. When the standard display console key is unlocked, the 3056 console is inoperative. When the standard display console key is locked and the 3056 console key is unlocked, the remote operator has unrestricted access to all frames via the 3056 console and the standard display console keyboard is inoperative. If both the standard display console and the 3056 console keys are locked, both opertors can access only the program frame and cannot perform an IPL or alter programs and data.

20:30 REMOTE SUPPORT FACILITY

FUNCTION AND COMPONENTS

The Remote Support Facility provides a remote problem analysis capability for Model 158 hardware that is a significant extension of the remote analysis functions available for the Model 155 using the optional 2955 Remote Analysis Unit as an interface to the RETAIN/370 system in Raleigh, North Carolina. RSF provides the means for a customer engineer specialist located in a system support center to perform diagnostic

operations on a Model 158 processor in an installation. This enables the specialist to assist the onsite customer engineer in analyzing CPU problems, if established security procedures are correctly performed, without having to go to the installation. The objective of RSF is to reduce the amount of time required to locate hardware failures so that system down time is reduced.

Standard hardware is included in the Model 158 to support RSF. The primary components of RSF are shown in Figure 20.30.1. RSF consists of a service processor unit that is natively attached to the Model 158 CPU, a communication line between the modem built into the service processor and the RETAIN/370 system in Raleigh, and a communication line between the RETAIN/370 system and a display device in a system support center that is being operated by a customer engineer specialist. The remote display device is a 3270 display unit. Voice communication between the local customer engineer and the remote customer engineer specialist is accomplished using an additional telephone line between the installation and the system support center.

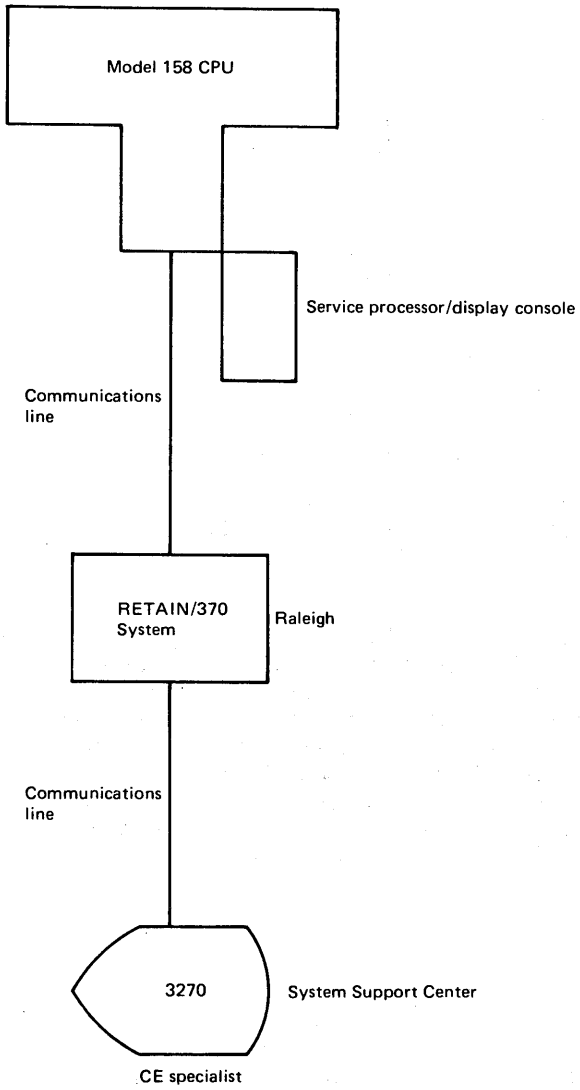


Figure 20.30.1. Components of the Remote Support Facility

THE SERVICE PROCESSOR

The service processor is a microprogram-controlled unit that is functionally independent of the Model 158 CPU but that receives its power from the CPU. Unlike the 2955, which requires a large portion of the Model 155 CPU to be operational for its own operation, the service processor can operate when the Model 158 CPU has a hard failure that prevents its operation.

The service processor contains the following major components in addition to an arithmetic/logic unit:

- Reloadable control storage to contain the microprograms that are controlling the operation of the service processor. Microprograms for the service processor are basically divided into those that support normal operating functions and those that support maintenance functions. They are contained on disk cartridges that are read by the inboard and outboard files. Basic microcode for the service processor is contained on the S-disk, which also contains the microcode for the Model 158 CPU. During an IMPL operation, which loads system microcode into reloadable control storage in the Model 158 CPU, basic microcode is also loaded into the reloadable control storage in the service processor.

  Diagnostic routines for the Model 158 that execute under control of the service processor are provided on several disk cartridges that are shipped to each installation with the S- and N-disks. When diagnostics are to be executed, the customer engineer removes the N-disk from the outboard file and mounts the appropriate disk cartridge in its place.

  Reloadable control storage in the service processor is not large enough to contain all the service processor microcode at the same time. Therefore, during normal operation of the Model 158 CPU and when diagnostics are being performed, microprograms are loaded into the service processor as required.

- Data registers that are used to transfer data between the service processor and the Model 158 CPU

- Controls for I/O operations to the inboard and outboard files. All access to these two files is controlled by the service processor.

- Controls for all I/O operations to the display console. The service processor controls the display of data on the console screen and accepts input from the keyboard of the display console and its light pen. Data from CPU programs and service processor programs that is to be displayed on the screen is formatted by a resident service processor routine and placed in a specific area of control storage in the service processor. Hardware in the display console continuously regenerates an image for the display screen from the data in this special control storage area.

- Controls for all I/O operations to the 3213 printer that can be used for hardcopy backup of the display console or when the display console is used in printer-keyboard mode.

- A built-in modem that is used to connect the service processor to the RETAIN/370 system via a communication line.

The service processor performs its functions in response to requests from the Model 158 CPU and from a local or remote operator using the light pen or display console keyboard and a display frame. In addition to the manual, alter/display, configuration, and program frames that are used by the primary console operator, the service, CE, teleprocessing

link, diagnostic, and index frames are provided for customer engineer use.

The service frame provides the customer engineer with the functions of the manual frame, additional customer engineer functions (that were provided via the rollers in the Model 155), and the means of calling other frames for diagnostic purposes. The CE frames enable the customer engineer to display the status (all registers, latches, control stats, etc.) of the Model 158 CPU while the diagnostic frame is used to execute specific diagnostic programs. The teleprocessing link frame is used to establish a connection between the service processor and the RETAIN/370 system and the index frame lists all the available display frames.

RETAIN/370 SYSTEM

The RETAIN/370 system provides communication control functions that enable a specialist in a system support center to communicate with a Model 158 CPU in an installation for remote analysis purposes. The Remote Support Facility program executes in the RETAIN/370 system to provide this function.

The RSF program makes a 3270 display device in a system support center appear to the specialist as a display console attached to a Model 158 CPU. All the functions a local customer engineer can perform using the light pen or keyboard of a Model 158 display console can be performed by the remote specialist using a 3270 that is controlled by the RSF program.

The RSF program accepts requests from the 3270 display, translates them to a format acceptable to the remote mode of the service processor, and transmits the resulting commands to the service processor. The service processor executes the commands and returns the results to the RSF program which retranslates them and sends them to the 3270 display. The RSF program also enables the remote specialist to invoke the Model 158 Logout Analysis program for execution in the RETAIN/370 system and to receive the results on his display.

REMOTE SUPPORT CAPABILITIES

RSF provides two functional modes of operation for the local customer engineer: remote console and remote program. The remote console mode is designed to be used when the local customer engineer requires the aid of a remote specialist while the remote program mode enables the local customer engineer to run test programs on the Model 158.

Remote Console Mode

When the local customer engineer determines that the help of a specialist is required to locate and fix a hardware failure, he establishes a connection with a specialist in a system support center via the remote support facility.

To accomplish this connection the local customer engineer first calls the system support center to obtain the number of a dial-up line between RETAIN/370 and the center. A specialist in the system support center also calls Raleigh to obtain an available line number. The local customer engineer activates the teleprocessing link between the service processor and RETAIN/370, specifying the assigned line and the remote specialist connects to RETAIN/370 also using the assigned line.

In order to activate the teleprocessing link, the local customer engineer first selects the teleprocessing link frame. The local customer engineer then inserts a CE key in the slot in the operator

control panel on the Model 158 CPU. This enables the modem in the service processor and lights up the remote pushbutton above the key slot. The CE key can be removed without deactivating the link. The modem can be deactivated at any time by pressing the remote pushbutton. The local customer engineer then selects the remote console resident code using the teleprocessing link frame.

The teleprocessing link is initiated by sending two lines of customer information to the remote specialist for security verfication. If the information is valid, the remote specialist returns a ready message and the link is established. The local customer engineer then selects the submode of operation to be used within the remote console mode. There are two possible submodes: remote control and remote monitor.

Remote control mode. When this mode of operation is in effect, the remote specialist has complete control over operation of the Model 158 using his display device. The remote specialist can perform any console function, either customer engineer or operator, that an operator or customer engineer located at the Model 158 installation normally could perform using the Model 158 display console.

The Model 158 display console cannot be used to control the operation of the Model 158 in this RSF mode (the keyboard and light pen are deactivated) but all functions executed by the remote specialist are displayed on the local console. The local operator or customer engineer can terminate the teleprocessing link at any time by pressing the remote pushbutton on the operator control panel. Note that this mode of operation does not require the execution of any programs in the Model 158 CPU or even that the Model 158 CPU be functional.

Communication between the local customer engineer and the remote specialist can be accomplished using the first twelve lines of the teleprocessing link frame. The local customer can signal the remote specialist by selecting any data on the display screen using the light pen. This causes a message to be displayed on the display device of the remote specialist. The remote specialist then controls the exchange (returns to the teleprocessing link frame to allow the local customer engineer to enter his message).

The remote control mode will be used when the local customer engineer cannot locate the cause of a solid failure. The remote specialist can display any required system status, execute any diagnostic functions available, and access any logout records on the N-disk using his display device. The specialist can request transmission of a logout record to RETAIN/370 for processing by the Logout Analysis Program or display the logout on his display device. The Model 158 will normally be dedicated to the remote specialist when this mode is in effect.

Remote monitor mode. When this mode is operative, the displays shown on the Model 158 display console are also shown on the display device of the remote specialist. This mode enables the remote specialist to monitor Model 158 displays but does not give him any control over the operation of the Model 158. The local customer engineer (or the operator) controls Model 158 operations. This mode could be used concurrently with normal system operation. However, since display data is transmitted to the remote site, executing programs will experience a degradation in performance.

Remote Program Mode

When the local customer engineer decides to run test programs on the Model 158, he establishes a direct connection to the RETAIN/370 system

by activating the modem in the service processor as described for remote console mode. He then selects the remote program resident code. The connection between the Model 158 system and the RETAIN/370 system is similiar to a direct channel connection and is not initiated from the teleprocessing link frame.

When remote program mode has been established, OLTs can be executed concurrently with normal system operations using OLTEP or in a standalone environment using OLTSEP. In addition, SYS1.LOGREC data can be sent to RETAIN/370 and the Model 158 Logout Analysis Program can be invoked to process the data. In both cases, the remote specialist can receive the results from the RETAIN/370 system at a later time.

20:35 STANDARD AND OPTIONAL SYSTEM FEATURES

STANDARD FEATURES

Standard features for the System/370 Model 158 (Models 1 and 3) are:

- BC and EC mode of operation
- Instruction set that includes binary, decimal, and floating-point arithmetic, and certain new System/370 instructions. Standard System/370 instructions for the Model 158 are:

  *CLEAR I/O
  COMPARE AND SWAP
  COMPARE DOUBLE AND SWAP
  COMPARE LOGICAL CHARACTERS UNDER MASK
  COMPARE LOGICAL LONG
  INSERT CHARACTERS UNDER MASK
  *INSERT PSW KEY
  *LOAD CONTROL
  *LOAD REAL ADDRESS
  MONITOR CALL
  MOVE LONG
  *PURGE TLB
  *RESET REFERENCE BIT
  *SET CLOCK
  *SET CLOCK COMPARATOR
  *SET CPU TIMER
  *SET PSW KEY FROM ADDRESS
  SHIFT AND ROUND DECIMAL
  *START I/O FAST RELEASE (executed as a START I/O on the Model 158)
  *STORE CHANNEL ID
  STORE CHARACTERS UNDER MASK
  STORE CLOCK
  *STORE CLOCK COMPARATOR
  *STORE CONTROL
  *STORE CPU ID
  *STORE CPU TIMER
  *STORE THEN AND SYSTEM MASK
  *STORE THEN OR SYSTEM MASK

- Dynamic Address Translation
- Reference and Change Recording
- Channel Indirect Data Addressing
- Instruction retry
- Interval timer (3.3 millisecond resolution)
- Time of day clock
- Clock comparator and CPU timer
- Monitoring feature
- Program Event Recording
- Program interruption for SSM instruction
- Expanded machine check interruption class

- ECC on processor storage
- Byte-oriented operands
- Store and fetch protection
- High-speed buffer storage - 8K bytes for the Model 1, 16K bytes for the Model 3
- Byte multiplexer channel 0
- Block multiplexer channels 1 and 2 (includes selector channel mode)
- Channel retry data in limited channel logout area
- Reloadable control storage
- Store status function
- Display console with keyboard and light pen
- Service processor unit

*Privileged instruction

OPTIONAL FEATURES

Optional features for the System/370 Model 158 (Models 1 and 3), all of which can be field installed, are:

- Extended Precision Floating Point (no-charge)
- 1401/40/60,1410/7010 Compatibility (no-charge)
- OS/DOS Compatibility (no-charge)
- 7070/7074 Compatibility (no-charge and mutually exclusive with the Virtual Machine Assist feature)
- Direct Control (includes External Interrupt feature)
- Block multiplexer channels 3, 4, and 5 (includes selector channel mode)
- Second Byte Multiplexer Channel*--replaces channel 4
- 3213 Printer for hard copy for the display console (optional for display mode, required for printer-keyboard mode)
- 3056 Remote System Console
- Channel-to-Channel Adapter
- Integrated Storage Controls
- Two Channel-Switch for Integrated Storage Controls
- Staging Adapter for Integrated Storage Controls
- Power Warning
- Virtual Machine Assist (no-charge and mutually exclusive with the 7070/7074 Compatibility feature)
- Multiprocessing (3058 Multisystem Unit)

*Channel 3 is a prerequisite.

SECTION 30: VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION

The first subsection, 30:05, discusses the needs that virtual storage and dynamic address translation in System/370 are designed to address. No previous understanding of these facilities is assumed. In this discussion, an address space is defined as a consecutive set of addresses that can be used in programs to reference data and instructions. System operation in IBM-supported virtual storage environments is explained conceptually, without use of all the terminology new to such an environment.

The general advantages of IBM-supplied virtual storage operating systems are presented also. Included in this subsection are those that apply to DOS/VS, OS/VS1, and OS/VS2. Additional advantages of virtual storage that are specific to a particular IBM-supplied operating system are discussed in the optional supplement for that operating system.

The last portion of subsection 30:05 defines the terminology associated with virtual storage and dynamic address translation hardware. The terminology included is that common to the four IBM-supplied programming systems that support a virtual storage environment for System/370. Terms unique to a particular programming system are defined in the optional supplement that describes that programming system.

Subsection 30:10 describes in detail the implementation and operation of dynamic address translation and channel indirect data addressing hardware in the Model 158 (Models 1 and 3). Other hardware items associated with dynamic address translation, such as reference and change recording, are discussed as well.

The last subsection, 30:15, discusses the new factors that affect system performance in a virtual storage environment. The information presented is related to efficient installation and utilization of an IBM-supplied virtual storage operating system.

The three optional programming systems supplements (Sections 80 to 100) for the virtual storage operating systems (DOS/VS, OS/VS1, and OS/VS2 Release 1) assume knowledge of the entire contents of Section 30. The optional supplement for VM/370 (Section 110) assumes knowledge of subsections 30:05 and 30:10 only, since performance in a virtual machine environment is discussed in the VM/370 supplement. This entire section applies to the Model 155 II, as well as to the Model 158, except where differences are noted.

30:05 VIRTUAL STORAGE CONCEPTS, ADVANTAGES, AND TERMINOLOGY

THE NEED FOR LARGER ADDRESS SPACE

The past and present rapid growth in the number and types of data processing applications being installed has led to an increasing demand for more freedom to design applications without being concerned about, or functionally constrained by, the physical characteristics of a particular computer system--system architecture, I/O device types, and processor storage size. As program design and implementation become easier, they can enable more rapid installation of applications, so that the benefits of data processing can be achieved sooner.

The design of System/360 and OS MFT and MVT allowed programmers to be less concerned than before about specific CPU architecture and I/O

device types when designing and implementing applications by (1) providing a compatible set of CPU models ranging in size from small to large scale, (2) providing a variety of high-level languages with greatly expanded capabilities, including a new language (PL/I), (3) providing comprehensive data management functions, including support of I/O device independence where data organization and the physical characteristics of devices permitted, and (4) supporting dynamic allocation of system resources (channels, I/O devices, direct access space, and processor storage). System/360 users who installed DOS Version 3 also experienced more system configuration independence than was previously available, although to a lesser degree than OS MFT and MVT users.

While System/360 and its primary operating systems represented major steps toward giving programmers a larger measure of system configuration independence, constraints that resulted from the necessity to design applications to fit within the amount of processor storage available still existed. In addition, although System/360 models provided more and less-costly processor storage than was previously available, increasingly larger amounts of processor storage began to be required as the use of high-level languages increased, the usage and level of multiprogramming increased, the functions supported by operating system control programs expanded, and applications that require relatively larger amounts of processor storage (such as teleprocessing and data base) were designed and installed more frequently.

The requirement for more processor storage is still growing. The new applications being developed and installed tend to have larger and larger storage design points in order to provide the functions desired. More processor storage is also required for I/O buffer areas to achieve maximum capacity and performance for sequential operations using new System/370 direct access devices with significantly larger track capacities. Larger blocking of tape records, which requires larger I/O buffers, also results in increased tape reel capacity and decreased tape processing time. As a result, System/370 models provide significantly more processor storage than their predecessor System/360 models and offer it for a lower cost.

The availability of more processor storage, however, has not relieved all the constraints associated with processor storage. Applications still must be tailored to the amount of processor storage actually available in a given system, even though storage design points (partition and region sizes) can be larger than they were previously.

Consider the following situations that can occur in installations:

1. An application is designed to operate in a 50K processor storage area that is adequate to handle current processing needs and that provides room for some expansion. Some time after the application is installed, however, maintenance changes and the addition of new functions cause one of the programs in the application to require 51K and another to require 52K. Installation of the next processor storage increment cannot be justified on the basis of these two programs, so time must be spent restructuring and retesting the programs to fit within 50K.

2. An existing application has programs with a planned overlay structure. The volume of transactions processed by these programs has doubled, and better performance is now required. Additional processor storage is installed. However, the overlay programs cannot automatically use the additional storage. Therefore, reworking of the overlay programs is required to take them out of planned overlay structure and, thereby, achieve the better performance desired.

3. A low-volume, terminal-oriented, simple inquiry program that will operate for three hours a day is to be installed. If the program is written without any type of overlay structure, it will require 60K of processor storage to handle all the various types of inquiries. However, because of a low inquiry rate, only 8K to 12K of the total program will be active at any given time. In order to justify its operational cost, considerable additional program development time is spent designing the inquiry program to operate with a dynamic overlay structure so that only 12K of processor storage is required for its execution.

4. A multiprogramming installation has a daily workload consisting primarily of long-running jobs. There are also certain jobs that require a relatively small amount of time to execute. The times at which these jobs must be executed is unpredictable; however, when they are to be run, they have a high completion priority. While it is desirable to be able to initiate these high-priority jobs as soon as the request to execute them is received, this cannot be done because long-running jobs are usually in operation. Hence, a certain time of day is established for initiating high-priority jobs, and the turnaround time for these jobs is considerably longer than is desired.

5. A series of new applications are to be installed that require additional computing speed and twice the amount of processor storage available on the existing system. The new application programs have been designed and are being tested on the currently installed system until the new one is delivered. However, because many of the new application programs have storage design points that are much larger than those of existing applications, testing has to be limited to those times when the required amount of processor storage can be made available. Although another smaller-scale model is also installed that has time available for program testing, it cannot be used because it does not have the amount of processor storage required by the new application programs. In addition, although the smaller-scale model now provides backup for the currently installed larger-scale model, the smaller-scale model cannot be used to back up the new system because of processor storage size limitations.

6. A large terminal-oriented application is to be operative during one entire shift. During times of peak activity, four times more processor storage is required than during low-activity periods. Peak activity is experienced about 20 percent of the time and low activity about 40 percent. The rest of the time, activity ranges from low to peak. Allocation of the peak activity processor storage requirement for the entire shift cannot be justified, and a significantly smaller storage design point is chosen. As a result, a dynamic program structure must be used, certain desired functions are not included in the program, and response times during peak and near-peak activity periods are increased above that originally planned.

   In this installation, most of the batched jobs are processed during the second shift. However, there is also a need to operate the large terminal-oriented application for a few hours during second shift. This cannot be done because the system does not have the amount of processor storage required for concurrent operation of the batched jobs and the terminal program (which must have its storage design point amount allocated even though that amount of processor storage would not be required during second-shift operations). The large amount of additional processor storage required to operate the terminal program for only a portion of the second shift cannot be justified.

7. An application program with a very large storage design point is executed only once a day as a batched job. A significant benefit would result from putting the program online to a few terminals during the morning hours. However, the program continues to be run as a batched job because it is very large and would be made larger by putting it online. The large amount of additional processor storage required to operate the program concurrently with the existing morning workload cannot be justified.

8. A terminal-based application has been installed on a full production basis for several months. During this period, the benefits accrued from the online application have encouraged the gradual addition of several more terminals, and peak activity is considerably higher than it was initially. Because growth has been gradual, much additional programming time (significantly more than is required to maintain batch-oriented applications) has to be spent periodically restructuring the terminal-based application program to handle the increasing volume of activity.

9. An online application is currently active during an entire shift and operates concurrently with batched jobs. It would be advantageous to install a second terminal-oriented application that would operate concurrently with the existing workload during the entire shift. However, the amount of processor storage that would have to be dedicated to each online application for the entire shift in order to handle its peak activity is very large, and times of peak activity for the two applications do not completely overlap. Because so much processor storage would be unused during a large portion of the shift if both online applications were always active, installation of the second online application is difficult to justify.

In the situations described, processor storage is a constraining factor in one way or another, and the constraints highlighted can apply in some degree to all systems regardless of their scale (small, intermediate, large) or processor storage size. The fact that larger, less-expensive processor storage is now available on System/370 models does not remove these constraints for two major reasons.

First, once a storage design point has been chosen for an application, whether the design point is relatively large or small, the application is dependent on that processor storage size for its operation. The application cannot execute in less than its design point storage amount, nor can it take advantage of additional available processor storage without being modified (unless it has been specifically structured to use additional storage as, for example, are most IBM-supplied language translators).

Second, although processor storage has become less costly, it still is a resource that should be used efficiently because of its importance in the total system operation. Thus, when storage design points are chosen, tradeoffs among processor storage cost, application function, and system performance are often made. Making applications fit within the storage design points selected becomes the responsibility of application designers and programmers. This situation is made more difficult by the fact that for many applications an optimum storage design point cannot be determined until the application is written and tested using expected transaction volumes.

The significance of processor storage restraints should be evaluated in light of the following trends evidenced by new types of applications: (1) the total amount of storage required to support their new facilities continues to grow larger, (2) the storage they actually require for operation during their execution is tending to become more variable, and (3) it is becoming as desirable to install many of these new

applications on smaller-scale systems with relatively small maximum processor storage sizes and low volume requirements as it is to install them on larger-scale systems. Reduction of the constraining factors currently imposed by processor storage is, therefore, a necessary step in making new applications easier and less costly to install and available to a wider range of data processing installations.

Given the existing processor storage restraints on application design and development and the storage requirements that are becoming increasingly more characteristic of many of the new types of applications, it becomes advantageous to allow programmers to design and code applications for a larger address space than they currently have. That is, programmers should be able to use as much address space as an application requires so that special program structures and techniques are not required to fit the application into a given storage size. Programmers can then concentrate more on the application and less on the techniques of programming. In addition, the size of the address space provided should not be determined by processor storage size, as it is in DOS Versions 3 and 4, OS MFT, and OS MVT, so that the address space can be larger than the processor storage available.

A larger address space should be provided, therefore, by a means other than making processor storage as large as the address space desired. This requirement can be satisfied by providing programmers with an address space (called virtual storage) that is supported using online direct access storage and dynamic address translation hardware. This approach also offers the advantage of supporting a larger address space for a lower cost than if larger processor storage is used, since direct access storage continues to be significantly less expensive per bit than processor storage. In addition, dynamic address translation hardware offers functional capabilities that large processor storage alone cannot provide.

VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION CONCEPTS

Virtual storage is an address space the maximum size of which is determined by the addressing scheme of the computing system that supports it rather than by the actual number of physical processor storage locations present in the computing system. In System/370, for example, which uses a 24-bit binary address, a virtual storage as large as 16,777,216 bytes can be supported. When virtual storage is implemented, the storage that can be directly accessed by the CPU, normally called processor or main storage, is referred to as real storage.

The concept of virtual storage is made possible by distinguishing between the names of data and instructions and their physical location. In a virtual storage environment, there is a distinction between address space and real storage space. Address space (virtual storage) is a set of identifiers or names (virtual storage addresses) that can be used in a program to refer to data and instructions. Real storage space is a set of physical storage locations in the computer system in which instructions and data can be placed for processing by the CPU. The number of addresses in the two spaces need not be the same, although both spaces begin with address zero and have consecutive addresses. The programmer refers to data and instructions by name (virtual storage address) without knowing their physical (real storage) location.

When virtual storage is not implemented, there is, in effect, no differentiation between address space and real storage space. The address space that can be used in programs is identical in size to the real storage space available and the address in an instruction represents both the name and the location of the information it references.

In a virtual storage environment, however, the address space available to programmers is that provided by the virtual storage size implemented by a given system--not the address space provided by the real storage available in the given system configuration. In DOS/VS, OS/VS1, and OS/VS2, virtual storage rather than real storage is divided into consecutively addressed partitions or dynamically allocated regions for allocation to problem programs. The fact that storage addresses in executable programs are virtual rather than real does not affect the way in which the programmer handles addressing. In System/370, for example, an Assembler Language programmer assigns and loads base registers and manipulates virtual storage addresses in a program just as if they were real storage addresses.

Virtual storage is so named because it represents an "image of storage" rather than physical processor storage. Since virtual storage does not actually exist as a physical entity, the instructions and data to which its virtual storage addresses refer, which are the contents of virtual storage, must be contained in some physical location.

In a virtual storage operating system environment, the contents of virtual storage are divided into a portion that is always present in real storage, namely, all or part of the control program, and another portion that is not always present in real storage. The instructions and data that are not always present in real storage must be placed in locations from which they can be brought into real storage for processing by the CPU during system operation. This requirement is met by using direct access storage to contain this portion of the contents of virtual storage (see Figure 30.05.1). The amount of direct access storage required to support a given amount of virtual storage varies by operating system, depending on how direct access storage is organized and allocated.

In addition, a mechanism is required for associating the virtual storage addresses of instructions and data contained in direct access storage with their actual locations in real storage when the instructions and data are being processed by the CPU. This requirement is met by using dynamic address translation (DAT) hardware in the CPU to associate virtual storage addresses with appropriate real storage addresses.

With this design, a system can support an address space that is larger than the actual size of the real storage present in the system. This is accomplished by bringing instructions and data from direct access storage into real storage only when they are actually required by an executing program, and by returning altered instructions and data to direct access storage when the real storage they occupy is needed and they are no longer being used. At any given time, real storage contains only a portion of the total contents of virtual storage.

Such a design is made practical by the fact that the logical flow of processing within the majority of programs is such that the entire program need not be resident in real storage at all times during execution of the program. For example, initialization and termination routines are executed only once during the operation of a program. Any exception-handling procedure, such as an error routine, is required only if the exception condition occurs. A program that handles a variety of transaction types (whether batch or online oriented) need have resident at any given time only the transaction routine required to process the current transaction type. It is this property of programs that has enabled planned overlay and other dynamic program structures to be used successfully in nonvirtual storage environments when the amount of processor storage available was not large enough. As indicated previously, this variable storage requirement characteristic of programs tends to be even more pronounced in new types of applications and in online environments in which processing is event driven.

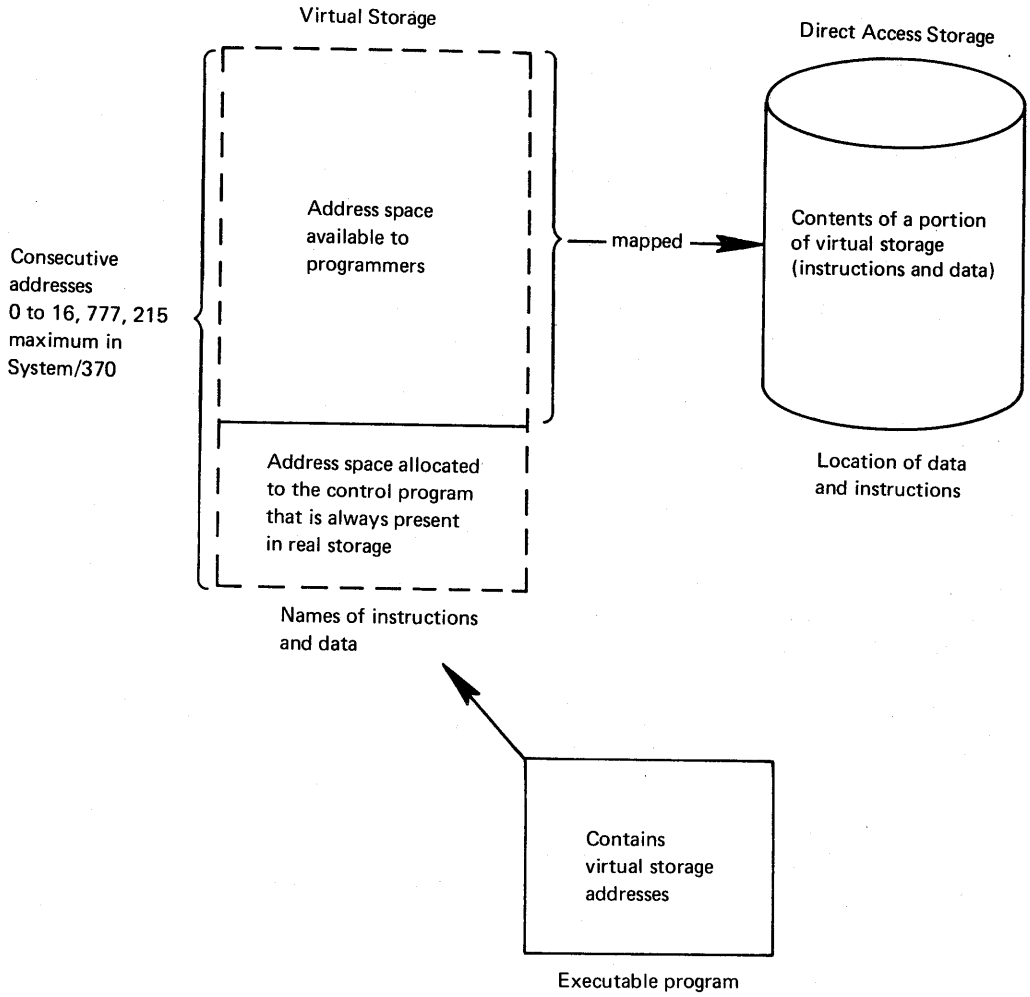


Figure 30.05.1. Names and location of instructions and data in a virtual storage environment

For the purpose of resource management in a virtual storage environment, virtual storage and its contents, the direct access storage that is used to contain a portion of the contents of virtual storage, and real storage are divided into contiguous fixed-length sections of equal size. Once a program has been fetched from a program library and initiated, instructions and data within the program are transferred between real storage and direct access storage a section at a time, during program execution. A section of an executing program is brought into a real storage section only when it is required, that is, only when a virtual storage address in the section is referenced by the executing program. A program section that is present in real storage is written in a direct access storage section only when the real storage assigned to it is required by another program section and only if it has been changed.

A virtual storage operating system control program monitors the activity of the sections of all executing programs and attempts to keep the most active sections in real storage, leaving the least active sections in direct access storage. Figure 30.05.2 illustrates the relationship of virtual storage, direct access storage, and real storage

without regard to a specific virtual storage operating system implementation.

The division of a program and its data into sections and the transfer of these sections between direct access storage and real storage during program execution is handled entirely by the virtual storage operating system without any effort by the programmer. When a planned overlay or dynamic overlay program structure is used, the programmer is responsible for dividing the program and its data into phases, determining which phases can be present at the same time in the amount of real storage available (partition or region), and indicating when phases are to be loaded into real storage during processing.

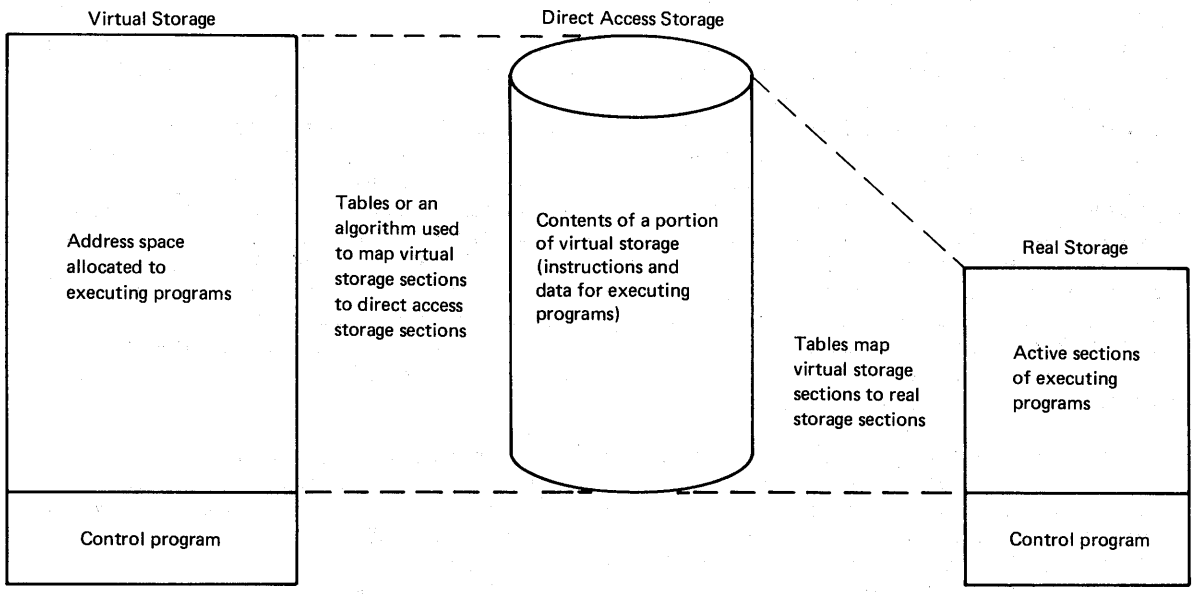


Figure 30.05.2. Relationship of virtual storage, direct access storage, and real storage

While a virtual storage up to 16 million bytes in size can be addressed by any System/370 model with DAT hardware, the virtual storage size that can be effectively implemented by a given system is affected by (1) the amount of real storage present, (2) the amount of direct access storage space made available to contain the contents of virtual storage, (3) the speed of the direct access storage devices containing virtual storage contents and contention for these devices or the channels to which they are attached, (4) the speed of the CPU, and (5) the characteristics of the programs operating concurrently. Hence, the amount of real storage required to effectively implement a specific amount of virtual storage can vary by system, depending on the characteristics of the applications in the workload and the performance desired, as is discussed in Section 30:15.

Once a program section has been loaded into real storage, its virtual storage addresses can be translated when they are referenced. Dynamic address translation hardware is the mechanism that translates the virtual storage addresses contained in instructions into real storage addresses during instruction execution. Address translation is accomplished in System/370 using a hardware-implemented table lookup procedure that accesses tables contained in real storage. These tables, which are maintained by control program routines, (1) define the amount of virtual storage supported and allocated, (2) indicate whether or not any given program section is currently present in real storage, and (3)

contain the addresses of real storage sections allocated to the program sections that are currently present in real storage.

During the execution of each instruction, address translation is performed on any virtual storage address in the instruction that refers to data or to an instruction. Translation occurs after the 24-bit effective virtual storage address has been computed by adding base, displacement, and index values, if any, together, as usual. The result of the address translation is a 24-bit real storage address designating the location containing the data or instruction referenced by the virtual storage address in the instruction. The virtual storage addresses in channel programs (CCW lists) are not translated by channel hardware during channel program execution and, therefore, programmed translation is required before initiation of a channel operation.

In reality, DAT hardware provides dynamic relocation of the sections of a program during its execution. This capability is not provided by DOS Version 4, OS MFT, and OS MVT. DOS Version 4 supports program relocation only at link-edit time. MFT and MVT support program relocation at program load time as well as at link-edit time. Once a program has been loaded into an area of real storage, these operating systems cannot relocate the program to another area of real storage during its execution. Thus, an entire program or a portion of a program cannot be written on direct access storage during execution and later reloaded into different real storage locations to continue execution. Once loaded, therefore, a program is bound during its execution to its initially allocated real storage addresses. In a virtual storage environment, a program is bound only to the virtual storage addresses it was assigned during loading.

The dynamic relocation provided by DAT hardware eliminates, for most programs, the need for allocating and dedicating a contiguous area of real storage to an entire program for the duration of its execution, a requirement for all programs in DOS Version 4, and OS MFT and MVT. (As discussed later in this subsection, some programs cannot operate correctly in the manner being described, that is, with sections transferred only as required between direct access storage and real storage.) In a virtual storage environment, real storage is no longer divided into contiguously addressed partitions or dynamically allocated regions that can contain one executing job step (program) at a time.

Further, when real storage is allocated to a section of an executing program, the real storage is not dedicated to that program section for the duration of program execution. Concurrently executing programs can dynamically share the same real storage sections. That is, in general, the real storage available for allocation to executing programs can be allocated to any program section as needed. When a section of an executing program must be loaded, any available section of real storage can be assigned (subject to certain restrictions imposed by operating-system-dependent real storage organizations). When the program section is no longer required, it can be written in direct access storage, if it has been altered, and the real storage assigned to it can be made available for allocation to another section of the same program or to a section of another program.

The assignment of real storage sections is handled entirely by the operating system, which also keeps account of which sections of concurrently operating programs are the most active. The operating system does not attempt to allocate a given amount of real storage to each executing program. It merely allocates real storage to those sections it determines are the most active, without taking into account the particular program to which the active section belongs.

DAT hardware, therefore, provides more than translation from address space (virtual storage) to real storage space. It provides the

capability of implementing dynamic real storage management that requires no effort on the part of the programmer and significantly less CPU time than programmed address translation during program execution. (The large amount of CPU time required to translate addresses during program execution using programmed means has precluded implementation by IBM of an operating system that supports programmed dynamic address translation.) Much of the real storage utilization preplanning required for MFT, MVT, and DOS Version 4 environments in order to use real storage effectively can be eliminated in a virtual storage environment. Dynamic real storage management capability is another advantage the technique of using direct access storage and DAT hardware to support a larger address space has over using larger real storage to provide a larger address space.

Another capability made available by the implementation of large address space using direct access storage and dynamic address translation is that of supporting more than one virtual storage with one system. Multiple virtual storages are supported by OS/VS2 Release 2 and also can be used to support multiple virtual machines. The concepts and general advantages of virtual machines are discussed in Section 40. The features and operation of VM/370 are presented in Virtual Machine Facility/370 Features Supplement.

The use of virtual storage and DAT hardware to enable programs to operate in less real storage than the total storage requirement of the programs can also offer better performance potential than the technique of using a planned overlay program structure. When a planned overlay program executes in MFT or MVT, considerable time can be spent executing the overlay supervisor in order to perform programmed address translation (relocation) when a program phase is loaded. In addition, more efficient real storage utilization may be achieved in a virtual storage environment, since the control program reacts to changing processing needs and only portions of the program that are actually required are loaded (all phases of an overlay program may not be the same size and all code within a phase may not be used when the phase is loaded). Once a planned overlay program has been structured to handle the currently required set of program phases efficiently, it cannot automatically adapt to a change in the set of program phases required or to a change in the activity of the required set of phases.

In a virtual storage environment, the performance of the system can be directly affected by the amount of time spent transferring program sections between direct access storage and real storage. Satisfactory system performance is achieved when each of the concurrently executing programs has enough real storage dynamically allocated to it to keep the need for transferring program sections into and out of real storage at an acceptable level.

As previously mentioned, most programs can be structured so that processing activity is localized in one area of the program or another during time intervals rather than equally spread over the entire program. In other words, at any given time period during execution of the program, only a subset of the entire program need be referenced. This is sometimes called the "locality of reference" characteristic of programs. Therefore, a program achieves satisfactory performance when its most active sections in any given time interval remain in real storage and there is a limited amount of program section transfer activity.

Most programs require a certain minimum amount of real storage in which to execute in order to achieve satisfactory performance. If such programs operate with less than their minimum requirement dynamically allocated, program section transfer activity increases and performance degradation can occur. The minimum real storage requirement of a program is related to the amount of real storage required by the most active sections of the program. Because of the locality of reference

characteristic of most programs, the minimum real storage requirement of a program for satisfactory operation frequently can be less than its total storage requirement. This can enable an operating system to efficiently support a virtual storage that is larger than the real storage actually present in the computing system.

A virtual storage environment, therefore, enables most programs to be independent of real storage size to a large degree. A program can execute in varying amounts of dynamically available real storage without being modified. The amount of real storage dynamically available to a program during its execution primarily affects its performance, to the extent that program section transfer activity is affected, rather than its capability of being executed. For example, a given 200K language translator might be able to operate with an average of 100K of real storage dynamically available to it during its operation; however, the time required to compile a program under these conditions might be unacceptable. Alternatively, the performance desired might be achieved if an average of 130K is dynamically available to the language translator while it operates. Without a virtual storage operating system, the 200K language translator might not be used at all because of its design point size.

In addition to the requirement for larger address space, there is still a requirement for larger real storage sizes in order to meet the functional and performance needs of the larger, more complex, multiprogramming environments. The availability of large lower-cost real storage for the Model 158 and the real storage independence that a virtual storage environment offers provide new flexibility in tradeoffs among real storage cost, function, and individual program or total system performance.

GENERAL ADVANTAGES OFFERED BY IBM OPERATING SYSTEMS THAT SUPPORT A VIRTUAL STORAGE ENVIRONMENT

Each of the IBM operating systems that supports a virtual storage environment for System/370 models using dynamic address translation offers the capability of using address space that is larger than that provided by available real storage, and each supports dynamic real storage management that is transparent to the user. As a result, these operating systems offer certain general potential advantages that do not depend on their unique features. The implementation of virtual storage also provides benefits that are specific to each of these operating systems because of their design and the particular functions they support. The following discusses the potential advantages of virtual storage and dynamic address translation that are common to DOS/VS, OS/VS1, and OS/VS2 environments.

The general advantages of virtual storage operating systems are the potential they offer for:

- Increased application development
- Expanded operational flexibility
- System performance improvement

A virtual storage operating system can facilitate more rapid development of new applications because, by removing most existing real storage restraints on application design, it can help improve the productivity of programmers. Specifically, a virtual storage operating system has characteristics that can be used to reduce the effort, time, and cost associated with application design, coding, testing, and maintenance. This makes the installation of new applications more readily justifiable and encourages the addition of new functions to

existing applications. The potential advantage of improved operational flexibility is made possible by the greater independence of applications from real storage size. Enhanced system performance can result from improved real storage utilization. While these latter two benefits have their own individual value, they also, either indirectly or directly, ease the installation of new applications.

Potential for Increased New Application Development

The following capabilities are characteristic of a virtual storage operating system environment:

- Greater flexibility in the design of applications is possible.

  Larger programs can be written without the necessity of using planned overlay techniques or other dynamic program structures designed to fit programs into the amount of real storage available. The need for intermediate (or working) data sets is reduced or eliminated because tables, relatively small data groups, etc., that are placed on direct access storage because of real storage limitations can become part of the program and will be brought into real storage automatically as required. Program planning, coding, and testing time can be reduced by elimination of the use of these programming techniques and other real storage management facilities, which also require additional programming knowledge and skill. Also avoided is the restructuring of application programs after they have been written because they are larger than the real storage available for their execution. Hence, applications can become operational more quickly.

  Open-ended, straightforward application design is possible, and more comprehensive programs can be written. An application can be segmented into a series of programs according to its logical flow instead of according to the functions that can be performed in the specific amount of real storage available to each step in the application. Programming and processing duplication inherent in the approach of using two or more job steps to perform one logical process is thereby avoided.

  Additional programming facilities can become available that otherwise could not be used because of real storage limitations. Specifically, full-function high-level language translators, which offer more capabilities than their subset versions (such as additional debugging facilities and performance options) but which also have larger storage design points, can be used because they can operate in a virtual storage environment using less real storage than their design point requirement.

- Preproduction testing of larger-than-average application programs can be increased if enough virtual storage can be made available to enable them to run during normal testing periods. Turnaround time during testing can be reduced.

  In a nonvirtual storage environment, such programs are usually grouped together and executed only at certain times when their larger design point storage requirements can be made available.

- Fine tuning of application programs to achieve performance improvements, when necessary, can be delayed until after the application is in production. This capability enables an application to become operative sooner.

- Startup costs for new applications may be reduced.

A new application can be developed and tested on the existing system, assuming the required I/O devices are present in the configuration, before the additional real storage the application requires for performance on a production basis is actually installed. When the application is ready for production, the additional real storage required can be added to the system. In some cases it may be possible to operate the application on a production basis on the existing system without adding real storage initially, because during the startup period, transaction volume is very low. As the volume grows, real storage can be added to achieve better performance.

- Growth of existing applications and the maintenance of operational programs is simplified.

  Because of the removal of most real storage restraints, new functions can be more easily and more rapidly added to most existing applications. Program expansion because of added functions or maintenance changes does not require the use of overlay techniques, multiple job steps, etc., when the size of the extended program exceeds the original storage design point size.

  In general, alteration and debugging of nonoverlay programs are also easier than alteration and debugging of programs with planned overlay or dynamic structures.

- Application programs whose real storage requirements, based on transaction volume and complexity, vary widely during their execution may be justified, designed, and installed more easily.

  Design, coding, and testing time can be reduced because dynamic storage management is automatically provided by the operating system. Time and effort need not be spent structuring such programs to use available real storage dynamically to support the functions and/or response times required.

- Design and installation of one-time, low-usage, or low-volume programs of very large storage size are more easily justified. Existing applications in these categories that currently operate in a batch environment can also more easily be altered to operate online, a growth step that might not be justifiable in a nonvirtual storage environment.

- Applications can be installed on a trial basis for the purpose of observing and evaluating their functions and their operation.

  Most IBM-supplied application program products can be temporarily installed on an existing system, assuming the required I/O devices are present. The additional hardware resources that may be required to operate the application on a production basis can be added later, when the application is permanently installed.

- The benefits of the functions provided by many IBM-supplied application program products with larger storage design points can be realized using smaller amounts of available real storage.

  Currently, it may be difficult to justify the real storage required to install a relatively large storage design point application on a system to handle a low volume of transactions, even though the functions provided by the application are very desirable. In a virtual storage environment, such an application can execute using that amount of dynamically available real storage required to satisfy the desired performance requirements for the low volume of activity.

Potential for Additional Operational Flexibility

The reduction of real storage restraints makes most applications more independent of the real storage size of a system configuration and permits most applications to be processed on systems with varying amounts of available real storage without program modification. Dynamic real storage management reduces the amount of job stream and operations preplanning that is normally done to use real storage as efficiently as possible in a multiprogramming environment. The following benefits can result:

- A system can back up another system even though it has less real storage than the system it backs up.

  A smaller-scale system with the appropriate I/O configuration can provide backup for a larger-scale system if necessary. (Performance experienced on the backup system may vary from that normally achieved, depending on the two system configurations involved.)

- A single design and one operating procedure can be used for an application that is to operate on multiple systems with varying amounts of real storage, as long as the virtual storage required is supported by all the systems.

  When data processing is decentralized among multiple installations with systems that have different amounts of real storage, one location can design, implement, and maintain an application that can be used by other installations. Duplication of this type of effort can be minimized or eliminated.

- Most applications can be tested on systems with less real storage than the one on which they will run in a production environment, as long as the required amount of virtual storage is supported.

- Growth to a larger real storage configuration can be easier.

  Real storage can be added to an existing system to improve system performance (by the reduction of program section transfer activity) without the necessity of modifying existing application programs so that they take advantage of additional real storage. Additional real storage (up to a maximum of their design point size) is automatically used by programs that operate in a virtual storage environment.

- Operators need not perform certain procedures that are solely related to efficiently managing real storage.

  The operator is concerned primarily with the division of virtual storage and therefore need not change partition sizes at various times (in DOS/VS or OS/VS1, for example) for the purpose of making storage available for larger-than-average jobs. (An installation can define virtual storage partitions that are larger than those currently defined in the DOS Version 4 or OS MFT environment, and the partitions can be made big enough to contain the largest existing or currently planned storage design point programs.) Similarly, in an OS/VS2 environment, the operator no longer need start long-running jobs at certain points in time to ensure that available real storage is fragmented as little as possible.

- Priority jobs whose need to be processed cannot be predicted can be scheduled when required.

  A nonvirtual storage environment does not provide the capability of effectively handling the scheduling of high-priority jobs on a random basis. Hence, this type of job is not permitted to exist in

an installation, or such jobs must be scheduled to operate only at certain times. In a virtual storage environment, a high-priority virtual partition (in DOS/VS and OS/VS1) can be defined and reserved for the purpose of processing only high-priority jobs. Except for that required for certain tables, real storage is not required for this partition until a job is actually scheduled. In OS/VS2, an initiator with a special class can be started that will handle only high-priority jobs. This can be done in MVT as well, but because of the possibility of real storage fragmentation, there is no assurance that the high-priority job can be started.

Potential for Performance Improvement

The improved real storage utilization made possible by the use of dynamic address translation hardware can have a positive effect on the performance of a system that handles a job mix whose use of real storage varies considerably while it is being processed. The extent of the performance improvement depends on the types of applications involved and the current utilization of system resources. Therefore, the amount of performance gain, if any, that may be achieved is highly variable by installation. Environments with the greatest potential for improved performance are as follows:

- Batch-oriented multiprogramming environments with application programs of widely varying real storage requirements.

  Real storage may not be most efficiently used in such an environment because (1) real storage can become fragmented when regions are dynamically allocated and freed or (2) it is difficult to divide real storage into a set of areas that is optimum for all programs when real storage is partitioned. (Consider the inefficient use of real storage in a 54K partition allocated for assemble, link-edit, and test jobs in which a 54K language translator, a 10K linkage editor, and problem programs no larger than 40K execute.) In addition, real storage is not efficiently used when the real storage requirement of a given program, based on transaction mix or volume, varies widely, and the amount of real storage that is allocated is designed to handle the peak requirement. (This is typically true of graphics applications, for example.) Further, real storage assigned to a program is not productively used during the time the program is waiting for a human response, such as for the operator to locate and/or mount a volume or to make a decision and enter a message on the console, or during the time required to quiesce the system in order to change partition definitions, start a high-priority job, or start a long-running teleprocessing job in high real storage.

  In a virtual storage environment, in which all concurrently executing job steps share real storage dynamically and use real storage only when it is actually required for program execution, real storage is more efficiently used. Hence, if real storage currently is the restraint, a given real storage size might be capable of supporting a higher level of multiprogramming than can be achieved without the use of dynamic storage management (assuming other required resources, such as CPU time, I/O devices, and channels, are available). For example, installation of a large storage design point, terminal-based application to handle only a few terminals might be possible. Alternatively, a higher level of multiprogramming might be supported by the addition of a smaller real storage increment than would otherwise be required.

  System performance may also be improved if more efficient use of available real storage enables additional heavily used functions to be made resident instead of transient or allows the incorporation of performance-oriented options in the control program. This

improvement can apply to environments with batch and online operations, as well as to batch-only multiprogramming environments.

- Multiprogramming environments with a mixture of batch-oriented and terminal-based applications.

  While the real storage required for the communication control portion of a teleprocessing application remains constant, terminal-based processing programs are typically subject to wide variations in the amount of real storage they require during their execution because the transaction mix being handled concurrently varies, the activity of each terminal online varies, or the number of terminals operating concurrently changes. In order to provide the functions desired, ensure the capability of handling peak activity periods and maximum transaction type mixes, and guarantee a given response during times of peak activity, a certain amount of real storage is required. This peak requirement is generally significantly more than is needed during periods of medium and low activity. Allocation of the maximum storage requirement results in inefficient use of real storage, since unused real storage dedicated to any terminal program cannot be used by other concurrently operating batched or terminal-oriented jobs in a nonvirtual storage environment. In addition, it is usually difficult, and sometimes impossible, to effectively preplan real storage usage for an online application.

  Dynamic real storage management in a virtual storage environment automatically provides a more efficient method of allocating real storage in such an environment. Real storage is not divided into that which can be used only by the terminal-based program(s) and that which can be used only by batched jobs. During times of peak terminal activity, the active sections of terminal-oriented processing programs with a higher priority are automatically allocated real storage, making less real storage available to the lower-priority batched jobs in execution at that time. During periods when terminal activity is relatively low, real storage not used by any terminal program is available for assignment to the active sections of executing batched jobs. Such an environment is represented conceptually in Figure 30.05.3.

  In existing mixed batch and online-oriented installations, dynamic real storage management allows programming techniques that can improve the performance of the online application. This improvement can be in the form of better response for existing terminals or the ability to support more terminals. A given online application may also be able to support a higher level of multiprogramming, as a result of better real storage utilization, without any additional programming effort (more TSO regions, for example). A virtual storage environment also can make the concurrent operation of multiple terminal-based applications more practical, because real storage equal to the design point storage amount of each online application need not be dedicated to the applications during the entire time they are active.

Figure 30.05.3 shows sample allocations of real storage to two batched jobs and two terminal-oriented jobs in a multiprogramming environment during low, medium, and peak activity points in time. Job priority from high to low is TP2, TP1, BJ2, BJ1. For simplicity, virtual and real storage are shown to be totally allocated at all times. No particular virtual storage operating system is assumed, since the concepts illustrated apply to OS/VS1 and OS/VS2 environments with BTAM and/or TCAM online applications, and to DOS/VS environments with BTAM (but not QTAM) online applications. Real storage is shown to be contiguously allocated to each job in high-to-low priority sequence. This is done only to illustrate the relative amount of real storage the

control program has dynamically allocated to each program during the instant shown. In reality, the total amount of real storage allocated to an executing program at any given time is usually not contiguous in a virtual storage environment. In addition, during times of low terminal program activity, it may be possible to support a higher level of batched job multiprogramming, which is not shown in the figure.

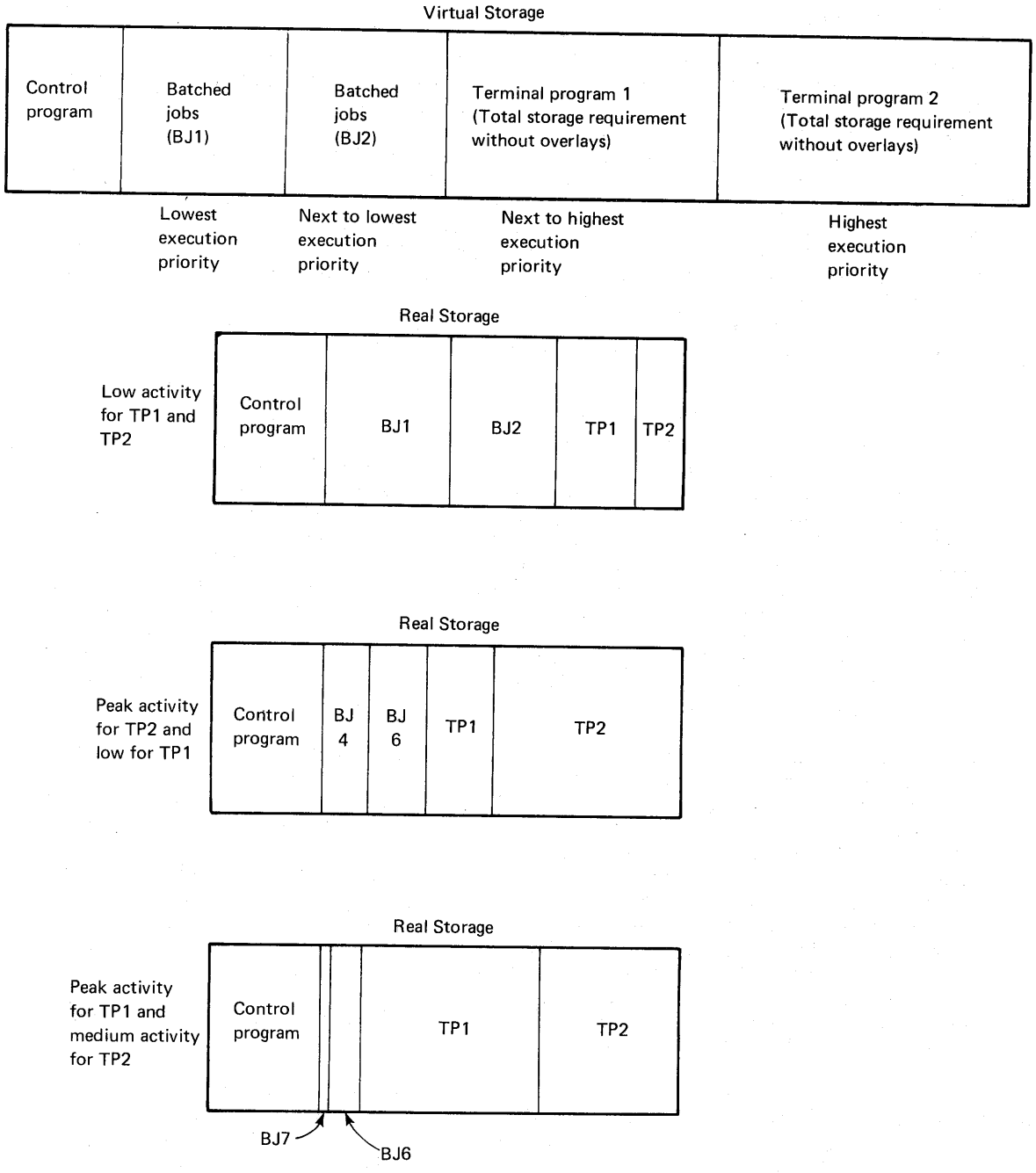


Figure 30.05.3. Conceptual illustration of real storage utilization in a mixed batch and online virtual storage environment

Summary

As the preceding discussion indicates, a virtual storage environment is designed primarily to provide new functional capabilities for the installation as a whole, although performance gains are possible for installations with particular environmental characteristics. The general functional aims of IBM-supplied virtual storage operating systems are (1) to use new hardware features and additional control program processing to support certain facilities that are not possible in a nonvirtual storage environment because of real storage restraints, and (2) to handle other functions that must be performed by installation personnel (programmers, operators, and system designers) when virtual storage and dynamic address translation are not used.

It is also important to note that while a virtual storage operating system permits an installation to be independent of real storage restraints to a large degree and enables real storage to be utilized more efficiently, the performance of the system and the specific advantages that can be achieved are still largely dependent on the amount of real storage present in the system and on the computing speed of the CPU, among other things. Hence, virtual storage and dynamic address translation are not a substitute for real storage. Rather they provide an installation with greater flexibility in the tradeoff between real storage size and function or performance.

The degree to which a particular installation experiences the potential benefits of a virtual storage/dynamic address translation environment is highly system-configuration dependent and application dependent (number, type, complexity of applications installed or to be installed). In addition, consideration must be given to the system resources that are specifically required to support a virtual storage environment (discussed in Section 30:15). Some of the potential advantages, such as those associated with application maintenance and operational flexibility and those that result from better management of real storage, can be experienced as soon as a virtual storage operating system is installed. Others may be achieved in the future, when new applications are installed and the less restrictive program design techniques available in a virtual storage environment are more fully utilized. In any case, installation of a virtual storage operating system can make System/370 easier to use and can be a major step toward more rapid installation of applications. Such an operating system can be of greatest benefit to installations desiring to move to or to extend online operations and attain the advantages such an environment offers.

VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION TERMINOLOGY

For the purpose of presenting the concepts of virtual storage and dynamic address translation in the previous discussion, virtual storage, programs and data, direct access storage, and real storage were described as being divided into areas called sections. In reality, a unique term is used to describe each one of the various sections, namely, virtual storage page, page, slot, and page frame. In addition, virtual storage has two levels of subdivision in System/370. The following defines the new terminology actually used by the virtual storage operating systems.

Virtual storage in System/370 is divided into contiguous segments, which contain virtual storage pages. A virtual storage segment, as implemented in System/370, is a fixed-length, consecutive set of addresses for either 64K or 1024K bytes which begins on a 64K or 1024K boundary, respectively, in virtual storage. A virtual storage is divided into segments all of one size or the other. In general, in OS/VS1 and OS/VS2 environments, a segment is the unit of virtual storage allocation. Each segment of virtual storage is divided into contiguous, fixed-length, consecutive sets of addresses called virtual storage

pages. Each segment in the virtual storage contains the same number of virtual storage pages, each of which is the same size. A virtual storage page, as implemented in System/370, can be either 2K or 4K bytes and is located on a 2K or 4K virtual storage boundary, respectively, within a segment.

The contents of virtual storage--instructions and data--are divided (by the operating system) into fixed-length contiguous areas called pages, corresponding in size to the virtual storage page size chosen, either 2K or 4K bytes. The addresses associated with a virtual storage page refer to the contents of a page.

The direct access storage used to contain the portion of the total contents of virtual storage that is not always present in real storage is called external page storage. Direct access space within external page storage is divided into physical records called slots, which are of page size, either 2K or 4K bytes. A slot, therefore, can contain one page at a time. A virtual storage page that is allocated and that actually has contents usually has a slot in external page storage associated with it to contain these contents (depending on the nature of the contents and how external page storage is managed by the operating system).

Instructions and data are transferred between external page storage and real storage as needed on a page basis. This transfer process is called paging, and a direct access device that contains external page storage is called a paging device. A slot in external page storage is associated with a particular virtual storage page by means of an algorithm or via tables that are maintained by the control program.

Real storage is also divided into fixed-length, consecutively addressed areas called page frames, which are always the same size as the page being used, either 2K or 4K bytes. Page frames are located on 2K or 4K real storage boundaries. A page frame is a block of real storage that can contain one page. Hence, a page of data and/or instructions occupies a slot when it is in external page storage and a page frame when it is in real storage. Whether or not a page is present in real storage, a program addresses the contents of the page using virtual storage addresses.

The act of transferring a page from external page storage into real storage is called a page-in. This action may also be described as the loading of a page. The reverse act, transferral of a page contained in real storage to a slot in external page storage, is called a page-out. Figure 30.05.4 illustrates the relationship of virtual storage, external page storage, and real storage that was conceptually shown in Figure 30.05.2. (Note that the terms swap-in, swap-out, and working set have a specific meaning in an OS/VS2 TSO environment and are defined in OS/Virtual Storage 2 Release 1 Features Supplement. The definition of a working set in a virtual machine environment is given in Virtual Machine Facility/370 Features Supplement.)

As previously indicated, DAT hardware uses tables to perform address translation. These tables are the segment table and page tables. One segment table and a set of page tables are required to perform address translation for one virtual storage. The segment table defines the virtual storage size, indicates allocated virtual storage, and points to the real storage location of the page tables. The page tables indicate which pages are currently in real storage and contain the real storage addresses of these pages. As pages are paged in and out, the control program makes changes to the page tables as required.

Basic to the implementation of virtual storage using direct access storage and DAT hardware is the method of determining when pages are to be brought into real storage and, therefore, when real storage is

allocated to pages. The method supported by IBM-supplied virtual storage operating systems, that of bringing a page into real storage only when it is needed by an executing program, is called a demand paging technique. Since programs execute on a priority basis in DOS/VS, OS/VS1, and OS/VS2 environments, as they do in OS (MFT and MVT) and in DOS (Versions 3 and 4) environments, real storage is, in effect, still allocated on a priority basis.

A request for a page-in is generated by the occurrence of a page exception or a page translation exception, a condition that is also called a page fault. An interruption occurs during the execution of an instruction when DAT hardware attempts to translate a virtual storage address into a real storage address and the appropriate page table indicates that the page is not currently present in real storage. A page fault condition causes an interruption in order to alert the control program to the fact that a page frame must be allocated. Usually, a page-in is required also to bring in the referenced instruction or data.

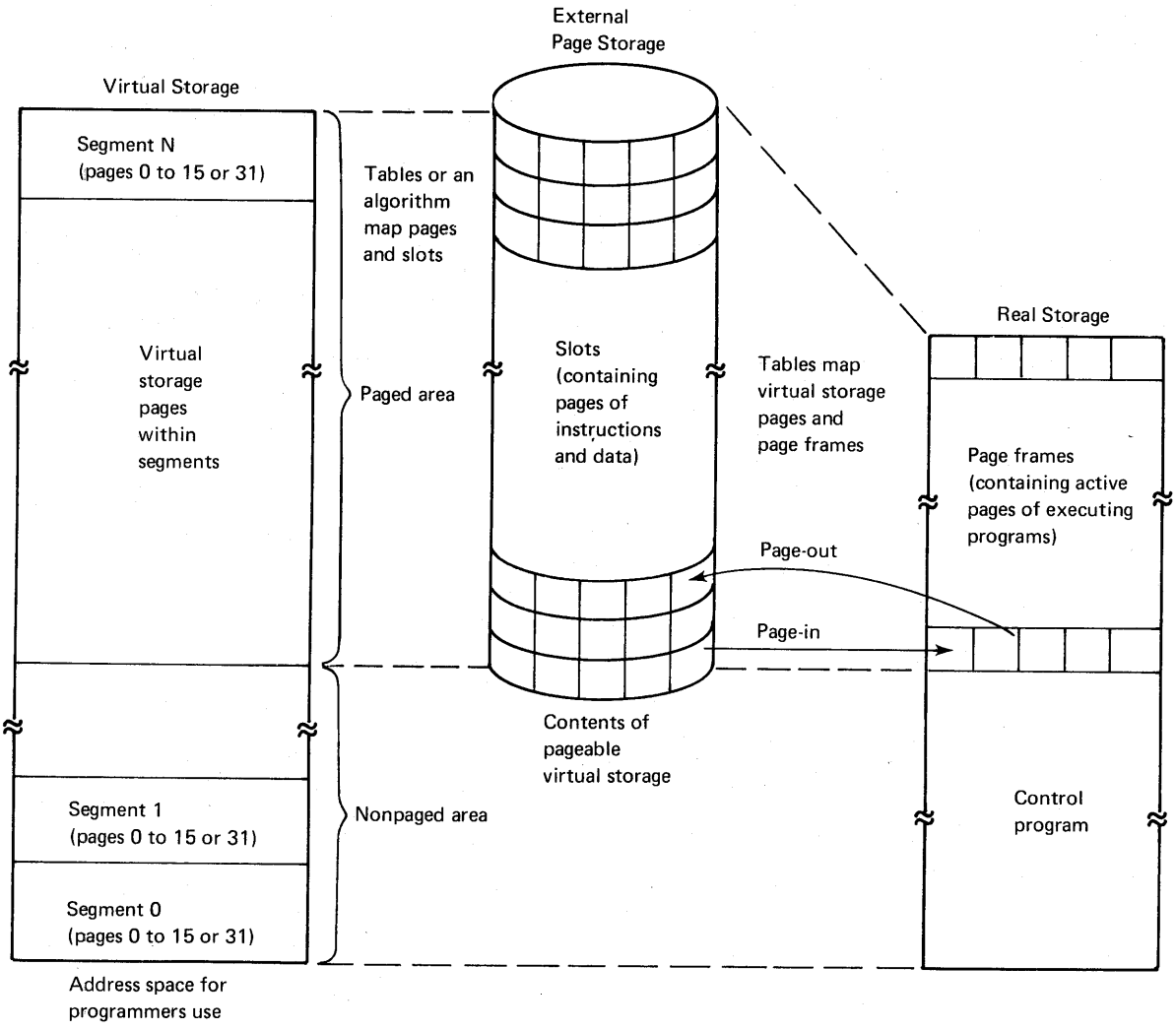


Figure 30.05.4. Layout of virtual storage, external page storage, and real storage

While page-ins are usually initiated as a result of a page fault, the OS/VS and DOS/VS Assemblers support a macro that can be used to cause

one or more pages to be brought into real storage before they are referenced. Such requests are sometimes referred to as page-ahead requests. A page-ahead is required if, for reasons of proper system operation, a routine must operate without causing any page faults. However, unlimited use of this facility can defeat the objective of demand paging.

When a page fault occurs and the control program determines that a page frame is not currently available for allocation, a choice must be made as to which allocated page frame will be taken away from the page to which it is currently assigned. The rule governing this choice is called the page replacement algorithm. If the page replacement algorithm is designed to choose from among only those page frames currently allocated to the program that caused the page fault, it is said to operate locally. If a page frame can be chosen from among all those available for allocation to all executing programs, the algorithm is said to operate globally.

DOS/VS, OS/VS1, and OS/VS2 implement a global page replacement algorithm. VM/370 supports a global page replacement algorithm and supports a local page replacement algorithm as an option. The algorithms used attempt to keep the most active pages of executing programs present in real storage. Hardware is included in System/370 models with dynamic address translation that indicates whether or not a page has been referenced or changed. Hence, when a page frame is required, a page determined by the algorithm to be relatively inactive is chosen for replacement.

Before loading a new page into the page frame chosen, the existing contents of the page frame must be saved if they were modified during processing. If modification occurred, a page-out operation is required; otherwise, an exact copy of the page already exists in external page storage. Code that is not modified during its execution, therefore, has an additional advantage in a virtual storage environment in that it need never be paged out once it has been written in external page storage. A program requiring a page-in is placed in the wait state until the page it requires has been loaded, during which time CPU control is given to another ready task, if one is available.

For various reasons, it is necessary to prevent a page-out of certain pages that are in real storage. Better operation of the system is one reason that applies to all (in DOS/VS) or a portion (in OS/VS1 and OS/VS2) of the control program, some routines that operate with the CPU in a disabled state (masked for I/O and external interruptions), most system tables, and most system control blocks.

Integrity of system operation is another reason. Pages associated with certain types of operations must not be paged out while the operation is in progress, so that the operation can proceed correctly. For example, pages that contain I/O buffer areas must remain in real storage while the buffers are being referenced during an I/O operation, after which a page-out can take place, if necessary.

Another reason is the existence of time dependency. A page should not be written out if the program to which the page belongs must complete a logical operation that requires the page in less time than it takes to perform a page-in. Programs that handle I/O device testing operations, such as online tests (OLT's), can have such a time dependency.

A page that is identified as one that cannot be paged out (or that is nonpageable) is called a fixed page. IBM-supplied OS/VS operating systems support both long-term fixing and short-term fixing, which are called permanent fixing and temporary fixing, respectively, in DOS/VS. In VM/370, a nonpageable page is called a locked page. Pages that

should never be paged out when they are present in real storage are marked long-term fixed. The resident portion of an operating system control program is never paged and, therefore, its pages are marked long-term fixed. Pages that must be fixed for only a portion of the time they are present in real storage are marked short-term fixed. For example, a page containing an I/O buffer is marked short-term fixed before the initiation of the I/O operation that references the buffer. After the I/O operation completes, the page is unfixed and it becomes eligible for a page-out. Pages should be marked fixed only when necessary, since page fixing reduces the amount of real storage that can be shared by concurrently executing paged programs (that which is available to be allocated to the nonfixed pages) and can, therefore, affect system performance.

As previously indicated, the supervisor in a DOS/VS environment, and a portion of the control program in OS/VS1 and OS/VS2 environments, are resident in real storage. That is, their pages are marked fixed, and they are not placed in external page storage (because they are not paged) even though they are allocated space in virtual storage. In OS/VS1 and OS/VS2, certain other portions of the control program are pageable and are made resident in virtual storage, which means they are contained in external page storage during system operation. During system initialization, these pageable control program routines are allocated virtual storage and loaded into real storage from system libraries by the program fetch routine. These routines will be written in external page storage as a result of normal paging activity in OS/VS1 environments and as a result of specific page-out requests in OS/VS2 environments. Control program routines that are resident in virtual storage are brought into real storage from external page storage, instead of from a system library, when they are required during system operation.

Just as control program routines can be fixed or pageable, problem programs operate in one of two modes in OS/VS1 and OS/VS2 environments: paged mode or nonpaged mode. The latter is also sometimes called virtual equals real (V=R) mode. When a problem program operates in paged mode, which is called virtual mode in a DOS/VS environment, it is resident in virtual storage and pageable. A pageable program operates in a contiguous area of virtual storage (partition or region) and is assigned any available real storage on a demand paged basis. Hence, virtual storage addresses must be translated into real storage addresses. The real storage dynamically allocated to programs operating in paged mode need not be contiguous, and such programs normally can operate with less real storage than their design point (virtual storage) amount dynamically allocated to them. This is the mode of operation described in Section 30:05.

Paged (virtual) mode is the normal mode of operation of programs in a virtual storage environment. However, certain programs cannot operate correctly in this mode, and must run in nonpaged (V=R) mode, which is called real mode in a DOS/VS environment. In general, a program must operate in nonpaged (real) mode if it:

- Contains a channel program that is modified while the channel program is active (Section 30:10 discusses the reason)
- Is highly time dependent (involves certain testing operations on I/O devices, for example)
- Must have all of its pages in real storage when it is executing (for performance reasons, for example)

Other characteristics that require a program to be executed in nonpaged mode and that are operating system dependent are listed in the

programming systems supplements, which also discuss steps that can be taken to avoid running a program in nonpaged mode.

In OS/VS1 and OS/VS2, a program that operates in nonpaged mode is dynamically allocated a contiguous virtual storage area and a contiguous real storage area of the same size with addresses identical to those of the allocated virtual storage area. (That is, virtual and real storage addresses of the allocated area are equal.) Since programs operating in V=R mode are not paged, they do not occupy external page storage. The entire program, except for dynamically requested modules, is loaded into real storage when it is initiated and all its pages are fixed. The amount of real storage allocated to a program that runs in nonpaged mode must be a multiple of the page size being used.

In a DOS/VS environment, one or more contiguously addressed real storage partitions must be defined if any programs are to operate in real mode. As in an OS/VS1 or OS/VS2 environment, real mode programs are not paged and do not occupy external page storage. The entire program, except for any dynamically requested phases, is loaded when the program is initiated. It must operate in a real partition that is equal to or larger than its design point size.

30:10 DYNAMIC ADDRESS TRANSLATION HARDWARE FOR MODELS 1 AND 3 OF THE MODEL 158

Dynamic Address Translation (DAT) is a standard facility of the Model 158. It is made operative by turning on the translation mode bit in the current PSW. The system must also be operating in EC mode. When DAT is operative, storage addresses in programs referring to instructions and data are translated into real storage addresses after instructions are fetched during program execution. (The address in the instruction counter is translated also.) When DAT is not operative, storage addresses in programs are used as real storage addresses. The storage addresses in CCW lists are not translated by channel hardware during channel program operation. The channel indirect data addressing feature, also standard on the Model 158, and programmed channel program translation are discussed later in this subsection under "Channel Indirect Data Addressing".

The following instructions are associated with dynamic address translation: LOAD REAL ADDRESS, RESET REFERENCE BIT, and PURGE TLB. These instructions are valid in BC mode as well as in EC mode. They operate identically regardless of which mode is in effect. All are privileged instructions.

VIRTUAL STORAGE ORGANIZATION

The Model 158 (as well as other System/370 models with DAT hardware) supports a virtual storage segment size of either 64K or 1024K bytes, as determined by bits 11 and 12 of control register 0. With either segment size, the page size can be 2K or 4K, as determined by bits 8 and 9 of control register 0. A segment size of 1024K bytes is not supported by DOS/VS, OS/VS1, OS/VS2, or VM/370. Table 30.10.1 summarizes the virtual storage organization provided in System/370.

As already described, the addresses supplied in programs directly address a location in the virtual storage that is supported by the virtual storage operating system. In this sense, program-supplied addresses can be viewed as virtual storage addresses that specify a byte within a particular virtual storage page and segment. The logic of the translation process is described in this subsection in these terms. The architectural definition of dynamic address translation found in System/370 Principles of Operation (GA22-7000-2 and later editions) assumes that the addresses in programs consist of three fields, two of

which are used to index tables during the translation process. Under these conditions, the addresses supplied by a program are considered to be logical addresses instead of virtual storage addresses.

For the purpose of translation, a virtual storage address is divided into three fields: (1) a segment field, which identifies a segment within the virtual storage, (2) a page field, which identifies a page within the segment addressed, and (3) a byte displacement field, which identifies a byte within the page addressed. The number of bits in each field varies depending on the segment and page sizes used. Virtual storage address fields for a segment size of 64K and a specific example of how the fields are used to address a location in virtual storage are shown in Figure 30.10.1.

Table 30.10.1. Number and size of segments and pages for a 16-million-byte virtual storage

CR 0 Bits 11,12	8,9	Segment Size (bytes)	Number of Segments in the Virtual Storage	Page Size (bytes)	Number of Pages in a Segment
10	01	1,048,576	16	2048	512
10	10	1,048,576	16	4096	256
00	01	65,536	256	2048	32
00	10	65,536	256	4096	16

OPERATION OF DYNAMIC ADDRESS TRANSLATION HARDWARE

Address Translation Tables

One segment table is required to describe one virtual storage. If more than one virtual storage is supported by a single computing system, there is a segment table for each virtual storage implemented. A segment table contains one four-byte entry for each segment in the virtual storage the table describes, up to a maximum of 256 entries for the maximum size virtual storage of 16 million bytes (using 64K segments). The real storage address of the segment table (or of the currently active segment table if multiple virtual storages are implemented) is contained in control register 1. The current length of the segment table is also indicated in control register 1. The length value is used by the hardware during translation to ensure that the segment entry being referenced falls within the segment table.

The segment table entries point to the real storage locations of the page tables. There is one page table for each segment in the virtual storage defined (or, in OS/VS2, currently allocated), up to a maximum of 256 page tables for a 16-million-byte virtual storage with 64K segments. A segment table entry contains an indication of the length of the page table, the high-order 21 bits of the real storage address of the page table, and an indication of whether the entry itself is valid and can be used for translation purposes (invalid bit). If the invalid bit is on in a segment table entry, a segment translation exception occurs during the translation process.

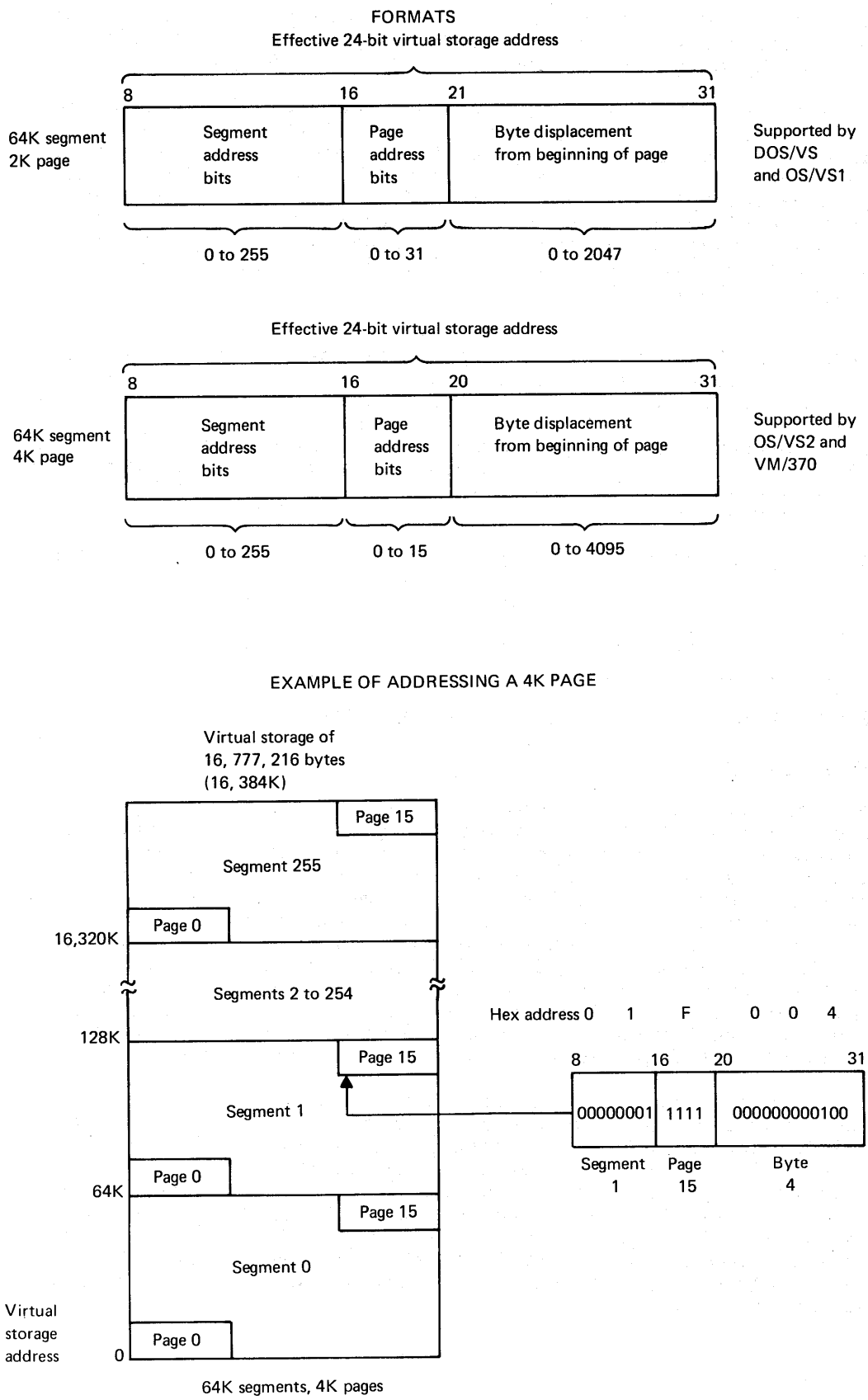


Figure 30.10.1. Virtual storage address fields for a 64K segment

A page table has one entry for each page in the particular segment the page table describes. For a 64K segment, there are 32 or 16 entries in a page table depending on whether a 2K or a 4K page is used, respectively. A page table entry is two bytes in size. It contains the 12 (for a 4K page) or 13 (for a 2K page) high-order bits of the real storage address of the page frame that is currently allocated to the virtual storage page that the page table entry describes. Each page table entry also contains an invalid bit to indicate whether the entry can be used for translation. The invalid bit is on when a virtual storage page does not have real storage currently allocated to it. A page translation exception occurs during the translation procedure if this invalid bit is on.

Segment and page table formats and entries used for address translation are shown in Figure 30.10.2. In effect, the segment and page tables define the relationship between virtual and real storage at any given time. The segment table reflects the current size of virtual storage and the location of required page tables. The segment table also indicates, by means of its invalid bits, which segments of virtual storage are currently allocated and have a page table available. The page tables indicate, via their invalid bits, which virtual storage pages currently have a page frame allocated and the location (real storage address) of these page frames.

In DOS/VS and OS/VS1 environments, segment and page tables are established at system initialization. Page tables are modified during system operation by control program routines to reflect the current allocation of real storage to virtual storage so that address translation can take place. In an OS/VS2 environment, in which virtual storage as well as real is dynamically allocated and deallocated, the segment table constructed during IPL is modified as required during system operation to reflect the allocation of virtual storage, and page tables are created and destroyed as necessary.

Address Translation Process

A translation request is either explicit or implicit. Explicit translation is invoked via execution of the LOAD REAL ADDRESS instruction. Implicit translation is invoked to translate all instruction addresses and data addresses contained in other instructions. Implicit address translation takes place during instruction execution.

The logical flow and the details of the translation process are given in Figure 30.10.3. The procedure consists of a two-level, direct address table lookup operation. Any type of translation exception causes a program interruption and termination of the hardware translation process. The CPU cannot be disabled for translation exception interruptions. Segment and page translation exceptions that occur during an explicit translation request (LOAD REAL ADDRESS instruction) are indicated via the condition code setting instead of via an interruption.

Translation Lookaside Buffer and Segment Table Entry Save Area

In the Model 158, two features are implemented to reduce the amount of time required to perform address translation: a translation lookaside buffer (TLB) and a segment table entry save area.

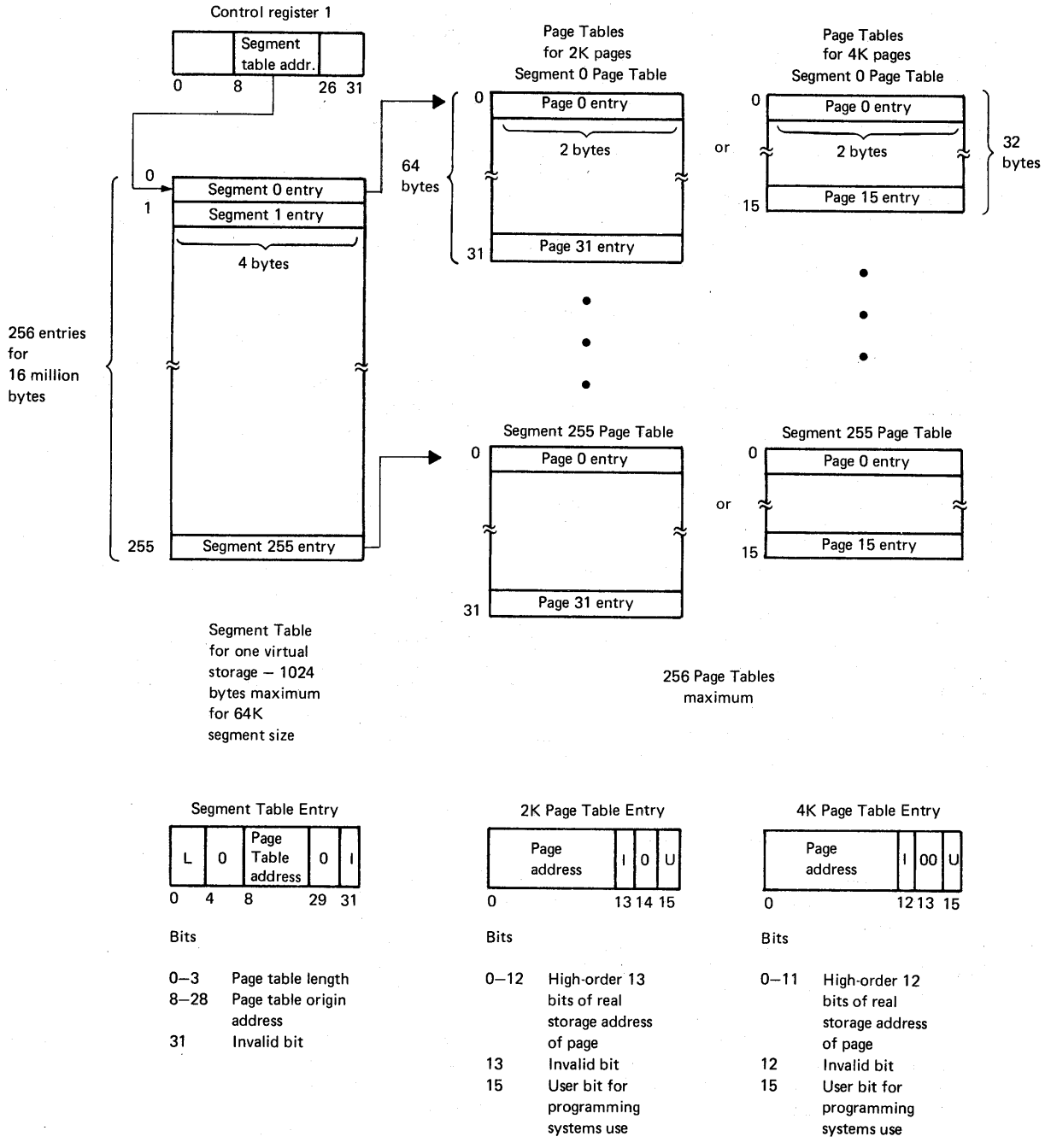
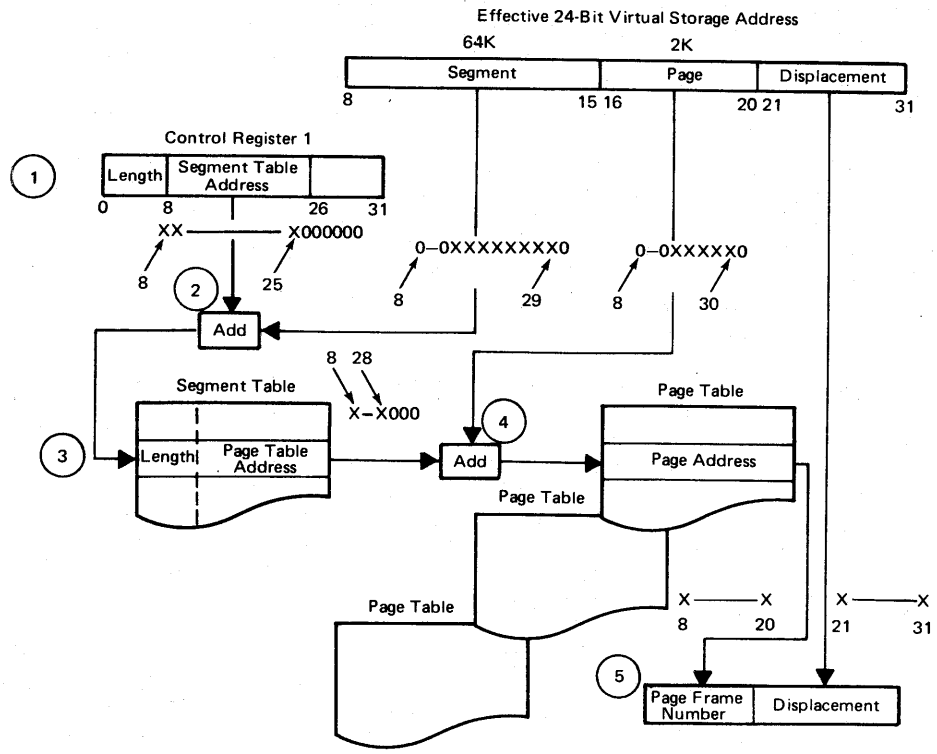


Figure 30.10.2. Segment table and page tables used for dynamic address translation



1. Bits 8, 9, 11, and 12 in control register 0 are checked for validity. A translation specification interruption occurs if an invalid setting is present. Segment address bits from the virtual storage address are checked using length bits in control register 1. If the segment entry address is outside the segment table, a segment translation exception is indicated.

2. Six low-order zeros are appended to the segment table address in control register 1. Two low-order zeros are appended to the segment bits from the virtual storage address. The two values are added to obtain a segment table entry. If the invalid bit is on in this entry, a segment translation exception is indicated.

3. Page address bits from the virtual storage address are checked using page table length bits contained in the segment table entry. A page translation exception is indicated if the entry addressed is outside the page table.

4. Three low-order zeros are appended to the page table address contained in the segment entry. One low-order zero is appended to the page address from the virtual storage address. The two values are added to obtain a page table entry. If the invalid bit is on in this entry, a page translation exception is indicated.

5. The 24-bit real storage address is formed using the 12 or 13 high-order bits from the page table entry and the 12 or 11 low-order bits from the virtual storage address.

Figure 30.10.3. Dynamic address translation procedure

The translation lookaside buffer is used to retain up to 128 translated addresses. Addresses associated with up to three different virtual storages can be contained in the TLB at any time. Every time a virtual storage address is translated during instruction execution, the virtual storage address, the resulting real storage address, and identification of the virtual storage to which the virtual storage address belongs are placed in one of the TLB locations.

The TLB is divided into 64 locations, each of which can contain two translations. Determination of which half of a TLB location to assign is based on whether the page number associated with the virtual storage address to be translated is odd or even. A certain set of six bits in the virtual storage address is used to directly address one of the 64 TLB locations. When an entry is placed in a TLB location, a valid indication is established for the entry.

After the effective virtual storage address has been computed and before performing translation using the tables, the TLB is interrogated to determine whether or not it contains the required translated address. Interrogation of the TLB is overlapped with reference to the index array of the buffer. (The low-order real address bits in the byte displacement field of the virtual storage address to be translated are used to access the appropriate halfblock entry in the index array for the buffer.) Therefore, no translation cycles are required when the translated address is obtained from the TLB.

If the TLB does not contain the required translation or if the entry is invalid, the segment table entry save area is inspected. The segment table entry save area is used to hold the contents of the segment table entry last used in translation. This area is inspected before the translation process begins. When virtual storage addresses in the same segment are referenced in succession, the segment table entry contents are taken from the save area, eliminating a buffer or real storage reference to obtain the segment entry contents and the first add operation in the translation process. If the segment table entry save area does not contain the required segment table entry contents, the complete table-lookup translation procedure, as previously described, is performed.

In the Model 158, the number of CPU (115 nanosecond) cycles required for address translation when the translation is not obtained from the TLB varies from a minimum of 6 to a maximum of 20, depending on the locations of the segment table and the page table entries. In the Model 155 II, from 6 to 37 CPU cycles are required for the translation process when the translation is not contained in the TLB.

During an initial program reset or a program reset, all 128 entries in the TLB are invalidated, as is the segment table entry save area value. The PURGE TLB instruction is provided to enable a program to turn off the valid indication for all 128 entries in the TLB and to clear the segment table entry save area value. In general, this instruction must be issued when an entry in a page table is invalidated, since the real storage address being invalidated could be contained in the TLB. (The TLB will be purged by the virtual storage operating systems as required.) The TLB is also automatically purged any time the page or segment size value in control register 0 is changed, as a result of a machine check, when a store status is performed using the display console, and when the ENTER key on the keyboard is pressed to perform an alter operation.

A change in the segment table origin address in control register 1 can also affect the validity of current TLB entries. In order to reduce the number of full TLB purges required by such a change, up to three unique segment table origin addresses are maintained in local storage. Each of these addresses could point to a segment table that defined a

different virtual storage. Each segment table origin address saved in local storage has a unique code value (1, 2, or 3) associated with it. One of these codes is identified as the currently active code to indicate which segment table and, hence, which virtual storage is currently active. Whenever a segment table address is placed in control register 1, the segment table address is also placed in local storage and the code it is assigned becomes the new active code number.

A segment table address code is stored with each TLB entry to identify the segment table with which the TLB entry is associated. When the TLB is interrogated to see if it contains the required translation, the code number of the TLB entry is compared with the active code number. If the codes are equal, this indicates the TLB location contains a translation from the virtual storage associated with the active code number. If the codes are not equal, the TLB location contains a translation for a different virtual storage and, therefore, the TLB entry does not contain the required translation even though it may contain a virtual storage address equal to the one that is to be translated.

Whenever control register 1 is loaded, the just-loaded segment table address is compared with each of the three saved segment table addresses in local storage to determine whether a change is being made. If a change is indicated, some TLB purging may be required. An equal comparison indicates that the virtual storage associated with the segment table address now in control register 1 is currently one of the three virtual storages whose translations are being maintained in the TLB. The code number of the segment table address now in control register 1 is designated to be the active code. No TLB purging is required.

If no equal comparison between a saved segment table address and the segment table address in control register 1 is found, this indicates translations for the segment table now indicated by control register 1 are not currently being maintained in the TLB. The new segment table address is placed in local storage and the code number assigned becomes the new active code. A first-in first-out algorithm is used to determine which code to assign. If the new address displaces another segment table address, the TLB entries associated with the displaced segment table must be purged. This is done by invalidating each entry in the TLB that has the same code number as the segment table address that was displaced. A zero code number in a TLB entry indicates an invalid entry. The code number of the displaced segment table address is now assigned to the newly stored segment table address. The other TLB entries need not be invalidated. See Figure 30.10.4.

Implementation of this segment table address saving facility in the Model 158 enables a control program that supports multiple virtual storages (such as VM/370) to alter control register 1 to change the virtual storage for which address translation is effective without automatically causing purging of the entire TLB. The segment table address saving facility will also avoid TLB purging in an OS/VS2 environment when control register 1 is changed. OS/VS2 Release 1 supports two segment tables to provide fetch protection for all regions (discussed in OS/Virtual Storage 2 Release 1 Features Supplement).

Addresses Translated

All storage addresses that are explicitly designated by a program and that are used by the CPU to refer to instructions or data in processor storage are virtual storage addresses and are subject to address translation. Thus, when DAT is operative, the starting and ending storage addresses used with the program event recording feature are virtual, as are the storage addresses stored in PSW's during interruptions. Address translation is not applied to addresses that

explicitly designate protect key storage locations or to quantities that are formed as storage addresses from the values designated in the base and displacement fields of an instruction but are not used to address processor storage (shift instructions, for example). In addition, address translation is not applied to the storage addresses in CCW lists used for I/O operations.

Some of the storage addresses supplied to a program by the CPU are virtual and some are real. Table 30.10.2 lists, for the Model 158, those storage addresses designated by a program, either explicitly or implicitly, that are virtual (and, therefore, are subject to translation) and those addresses that are real or not used to reference processor storage and, thus, are not translated. The table also indicates which storage addresses supplied to a program are virtual and which are real.

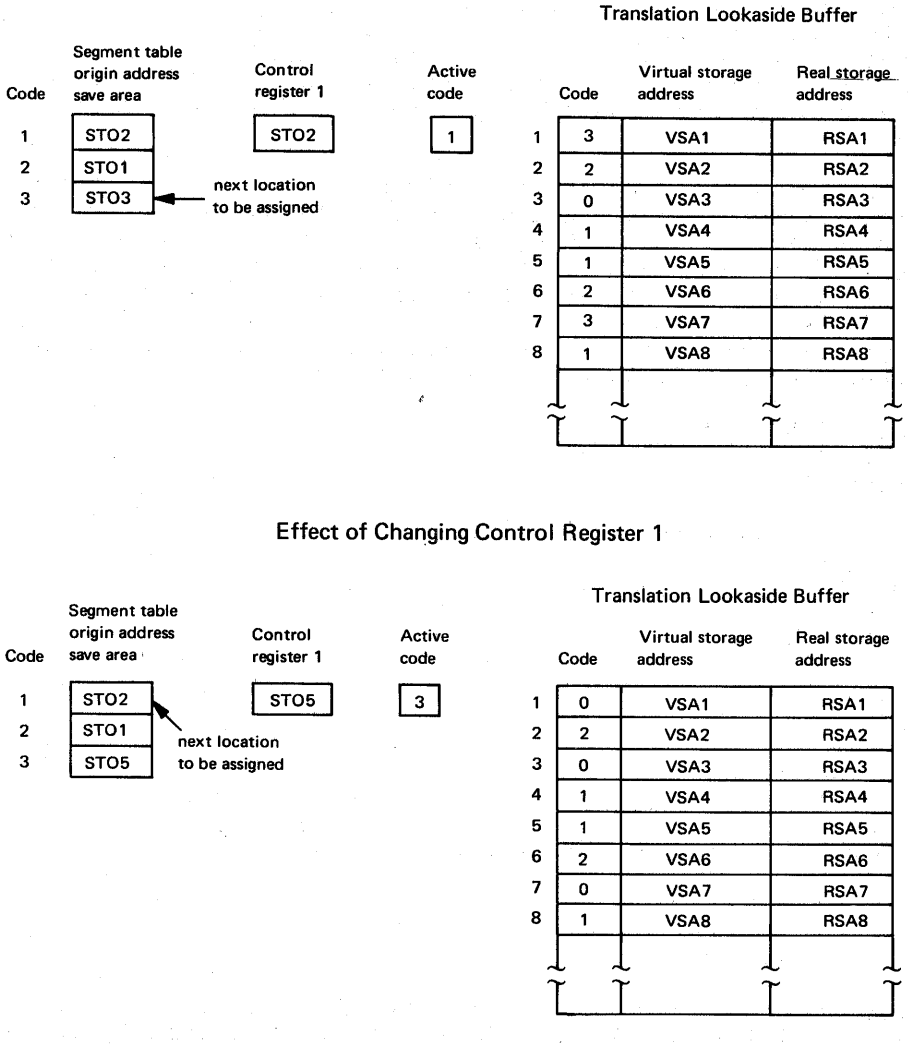


Figure 30.10.4. TLB purging when control register 1 is changed

Table 30.10.2. Virtual and real storage addresses used by and supplied to programs in the Model 158

Virtual Storage Addresses Explicitly Designated by the Program (translated)

- Instruction address in the PSW
- Branch addresses in instructions
- Addresses of operands in instructions
- Operand address in LOAD REAL ADDRESS instruction
- PER starting address in control register 10 and PER ending address in control register 11

Real Storage Addresses Explicitly Designated by the Program (not translated)

- Operand addresses in SET STORAGE KEY, INSERT STORAGE KEY, and RESET REFERENCE BIT instructions
- Machine check extended log pointer in control register 15
- Segment-table-origin address in control register 1
- Page-table-origin address in a segment table entry
- Page frame address in a page table entry
- CCW address in the channel address word (CAW)
- Address in a CCW specifying a data area or the location of another CCW
- Data address in channel indirect data address lists

Addresses Not Used to Address Storage (not translated)

- Operand addresses specifying the amount of shift in fixed-point, logical, or decimal shift instructions
- Operand address in LOAD ADDRESS and MONITOR CALL instructions
- I/O addresses in I/O instructions and in the Input/Output Communication Area (IOCA)

Real Storage Addresses Used Implicitly (not translated)

- Addresses of PSW's used during an interruption and in executing the programmed or manually initiated restart function
- Address used by the CPU to update the timer at location 80
- Address of the CAW, the CSW, and the I/O address within the IOCA used during an I/O interruption or during execution of an I/O instruction, including execution of STORE CHANNEL ID
- Addresses used for the store status function

Virtual Storage Addresses Provided to the Program

- Address stored in the instruction address field of the old PSW during an interruption
- Address stored by a BRANCH AND LINK instruction
- Address stored in register 1 by TRANSLATE AND TEST and EDIT AND MARK instructions
- Address stored in location 144 on a program interruption for a page translation or segment translation exception
- Address stored in location 152 on a PER interruption

Real Storage Addresses Provided to the Program

- The translated address generated by the LOAD REAL ADDRESS instruction
- Address of the segment table entry or page table entry provided by the LOAD REAL ADDRESS instruction
- Failing storage address in location 248
- CCW address in the CSW

Expanded Alter/Display for Display Console

The alter/display function of the display console is expanded in the Model 158 to accept virtual storage addresses in alter/display requests. When the operator indicates that a virtual storage address has been entered, the address is translated into a real storage address under microprogram control, without reference to the TLB, using the existing contents of control registers 0 and 1 to define the segment and page sizes and the location of the segment table. Translation mode need not be specified. Any alter operation causes the TLB to be purged.

The virtual storage address entered and the translated real storage address are displayed for both alter and display operations. If a translation exception occurs, an error indication is displayed instead of the real storage address. An exception occurs when a page frame is not currently allocated to the virtual storage page referenced.

The ADDRESS COMPARE function still uses real storage addresses only. The SET IC function assumes the storage address entered to be virtual or real depending on the setting of the translation mode bit in the current PSW. (The STORE and DISPLAY pushbuttons on a Model 155 II console assume real addresses only.)

FEATURES TO SUPPORT DEMAND PAGING

Reference and Change Recording Facility for Real Storage Blocks

A hardware recording facility is standard in the Model 158. This facility provides continuous recording of the activity of all 2K real storage blocks via reference and change bits. The settings of these recording bits can be used by control program routines to support a demand paging environment. This hardware facility is always active; it does not depend on EC or translation mode being operative.

The seven-bit key associated with each 2K real storage block in the Model 158 has four storage-protect bits, one fetch-protect bit, one reference bit, and one change bit. During system operation, the activity of each 2K real storage block is monitored by hardware. Whenever a fetch is made either by a CPU or a channel to a real storage address, the reference bit in the key associated with the 2K storage block that contains that real storage address is turned on by the hardware. A store into any real storage address causes the hardware to turn on both the change bit and the reference bit for the affected 2K block.

Alter/display operations initiated from the display console or keyboard also cause appropriate changing of the reference and change bits. The RESET REFERENCE BIT instruction is provided to allow the reference bit of any 2K real storage block to be reset by programming without altering the contents of the other six bits in the protect key.

A CPU fetch that is satisfied with data contained in the buffer causes reference recording in the Model 158. There are situations in which instruction or operand prefetching may cause the reference bit for a page frame to be turned on even though the contents of the page are never used.

The hardware reference and change recording facility is used by the page replacement algorithm of a virtual storage operating system. When a page is loaded into a page frame, the reference and change bits for that page frame are set to zero. (When a 4K page size is used, the reference and change bits for both of the 2K storage blocks involved are reset.) Thereafter, the reference bit is used to determine the activity of a page. The change bit is inspected to determine whether a page must

be paged out when its page frame is reassigned. The SET STORAGE KEY instruction must be used to reset the change bit.

Instruction Nullification

When a page fault occurs in a demand paging environment, execution of the instruction that caused the page fault stops and the control program gains control to initiate a page-in operation. When the contents of the missing page have been loaded (and the appropriate page table entry has been updated), the instruction that caused the page fault is reissued. For the instruction to operate correctly the second time, execution of the instruction must have been stopped so that reexecution gives the same results as would have occurred if the instruction had been executed only once. Therefore, the contents of real storage, the general and floating-point registers, and the PSW must not be altered.

The execution of an instruction is said to be nullified when it is stopped in such a way that no operation is performed, no fields were changed, and the PSW indicates the address of the instruction that was stopped. Interruptible instructions, such as MOVE LONG, are divided into execution units. One or more execution units may have completed before a page fault is detected. In this case, only the current execution unit is nullified.

Various methods are used, depending on the type of instruction, to determine the need for nullification. In some cases, instruction execution is attempted where hardware detection of page faults permits nullification. In other cases, pretesting is required to determine whether the virtual storage pages to be referenced have page frames allocated. Nullification testing is required only for instructions whose translated addresses reference real storage. Testing is accomplished in the Model 158 by additional microcode routines that are executed before normal instruction execution microcode.

Instructions that do not need pretesting, for example, are those whose operation is such that when the operands they reference are on integral storage boundaries that are a multiple of the implied operand length, only one page can be involved. For example, a store fullword (STORE) instruction that addresses a four-byte data field aligned on a fullword boundary cannot cross a page boundary during execution. The aligned data will always be totally contained in one page. This instruction is allowed to execute without pretesting as soon as it has been determined that the addressed data field is on an integral boundary.

Similarly, if a store fullword instruction addresses a four-byte field that is not on a fullword boundary, a pretest is required to determine whether all four bytes are contained in real storage. The pretest microcode for this instruction issues a fetch to the highest addressed byte in the four-byte data field (virtual storage address in the instruction plus 3). The absence of a page translation exception during translation of the virtual storage address indicates that (1) if the data field spans two pages, at least the second of the two pages is present in real storage or (2) the data field is totally contained in one page, which is present in real storage. Hence, the instruction is allowed to proceed without nullification. If the data field actually does span two pages and the first page is not present in real storage, this fact will be indicated by a page fault during translation of the address of the high-order byte of the field. Instruction nullification will occur and the page fault will cause a page-in of the first page to be initiated by the control program as usual.

If the pretest fetch operation does cause a translation exception, the store fullword instruction is nullified and the control program gains CPU control to load the missing page. Once again, the page-in

caused by the pretest may have brought in the second of two pages spanned by the data field or the only page containing the data field. After the page-in, the instruction is reexecuted.

CHANNEL INDIRECT DATA ADDRESSING

Since address translation is not performed by the channels for programs that operate in paged (virtual) mode, address translation must be performed on CCW lists by programming before the initiation of START I/O instructions. Such address translation need not be performed on the CCW lists of programs that operate in nonpaged (real) mode.

In addition, a contiguously addressed I/O area in virtual storage can span a set of noncontiguous page frames. Hence, a method of handling a noncontiguously addressed I/O area in real storage during the operation of a CCW list is required. The standard channel indirect data addressing feature is used to provide this capability. As is shown in Figure 30.10.5, the use of channel indirect data addressing allows the channel program logic used in the CCW list with virtual storage addresses to be maintained in the new CCW list that contains real storage addresses.

When channel indirect data addressing is present, bit 37 of a CCW is designated as the indirect data address (IDA) flag. The IDA flag applies to read, read backward, write, control, and sense commands and is valid in both BC and EC modes. When the IDA flag in a CCW is zero, bits 8 to 31 of the CCW specify the real storage address of the beginning of the I/O area as usual. When the I/O area referenced by a CCW is completely contained in one page, an indirect data address list (IDAL) is not required and the IDA flag is set to zero. When the IDA flag is one, CCW bits 8 to 31 specify the real storage address of an IDAL instead of an I/O area. When the I/O area referenced by a CCW spans two or more pages, an IDAL is required and the IDA flag is set to one.

An IDAL consists of two or more contiguous indirect data address words (IDAW"s) of four bytes each. There is one IDAW in an IDAL for each 2K storage block spanned by the I/O area. An IDAW, which must be aligned on a fullword boundary, contains a real storage I/O area address in bits 8 to 31. Bits 0 to 7 must be zero. The first IDAW in the list points to the beginning of the I/O area to be used by the CCW and is obtained by translating the virtual storage address contained in the original CCW. Any valid real storage address can be specified in the first IDAW of a list. All IDAW's after the first must address the beginning (or end for a read backward operation) of a 2048-byte block located on a 2048-byte boundary, or a program check occurs. That is, bits 21-31 of the address in the IDAW must be zeros (or ones for a read backward).

Figure 30.10.5 shows an example of the IDAL's required for a command-chained CCW list when 2K pages are used. The IBM-supplied virtual storage operating systems construct a new CCW list with translated addresses that is used to control the I/O operation. The new CCW list points to any required IDAL's.

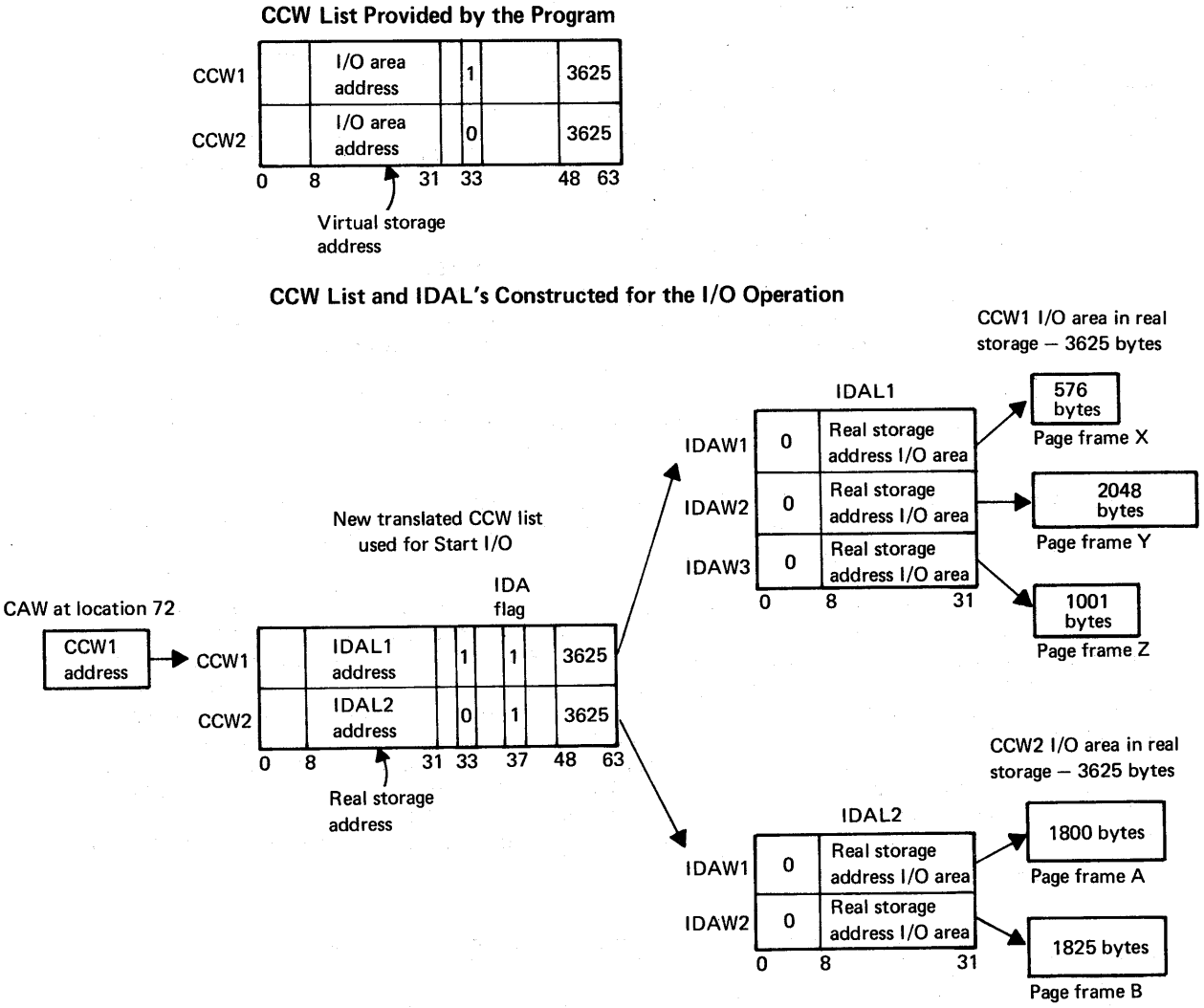


Figure 30.10.5. Example of IDAL's required for a CCW list when page size is 2K

When a START I/O instruction is executed, the channel fetches the first CCW in the list, pointed to by the channel address word (CAW), and inspects bit 37. If it is zero, the operation is started in the I/O area specified by the real storage address in the CCW. If bit 37 is a one, the first IDAW is fetched from the real storage address in the CCW. The I/O operation is begun using the real storage address in the first IDAW and, assuming that the I/O operation is not a read backward, ascending real storage addresses in the I/O area are used by the channel until a 2048-byte boundary is reached.

The channel detects a 2K boundary by monitoring I/O area address bits 21-31. When these bits change from all ones to all zeros, the first byte of the next 2K real storage block is indicated. At this point, the channel accesses the second IDAW in the list to obtain the next real storage I/O area address to be used, and the data transfer operation continues. (In the Model 158, IDAW's are not prefetched.) The channel

continues using the IDAL until the operation indicated by the CCW completes (CCW count reaches zero, IRG on tape reached, etc.). The next CCW is accessed if command or data chaining is indicated. Bit 37 is inspected and the I/O operation continues as described until the CCW list is exhausted.

When a program operates in paged mode, the CCW list for an I/O operation must be inspected, the new CCW list with translated address must be built, and the appropriate IDAL's must be constructed before issuing a START I/O instruction. At the completion of an I/O operation, some retranslation is also required. In general, the following steps must be taken for each CCW in a given list:

1. Determine whether the I/O area referred to in the CCW spans pages or is contained in only one. If a single page is involved, translate the virtual storage address to real and store it in the CCW. Ensure that a page frame is allocated to the page containing the buffer and that the page frame is marked fixed.

2. If two or more pages are involved, set up the required number of IDAW's, place a pointer to the IDAL in the CCW, and turn on CCW bit 37.

3. While setting up IDAW's, determine whether all pages in the I/O area have real storage allocated. If not, ensure that page frames are allocated and fixed.

At the completion of an I/O operation, the real storage address in the channel status word must be translated into a virtual storage address, and pages that were short-term fixed prior to initiation of the I/O operation must be unfixed. Channel program translation and page fixing are performed by the I/O control portion of the control program in IBM-supplied virtual storage operating system support. A program that contains a CCW list that is dynamically modified during its execution cannot operate correctly in paged mode since the modification is made to the CCW list with virtual storage addresses rather than to the translated CCW list that is actually controlling the I/O operation on the channel.

30:15 SYSTEM PERFORMANCE IN A VIRTUAL STORAGE ENVIRONMENT

A virtual storage environment is designed to provide new data processing capabilities. As is true of any other capability offered by an operating system, support of a new function requires control program use of a certain amount of the hardware resources of the system. In this respect, virtual storage is no different from multiprogramming and the many other new capabilities that have continuously been added to DOS and OS since their initial release.

The characteristic that makes virtual storage different from most other features is that virtual storage is not primarily designed to improve system performance, as are many other control program facilities. Virtual storage is first a functional tool and, in certain cases, can also be a performance tool. The objectives of DOS and OS virtual storage operating systems are to (1) provide new functions, (2) maintain upward compatibility with DOS and OS nonvirtual storage environments, and (3) provide performance equal to or better than that achieved with a nonvirtual storage operating system using the same system configuration. Attainment of the last objective will not be possible for all existing System/370 configurations.

In addition, some of the new functions a virtual storage environment provides cannot be achieved in a nonvirtual storage environment or are not practical, and in these cases, performance is not the primary

consideration when using the facility virtual storage offers. As the cost of hardware resources continues to decline on a unit cost basis (cost per processor storage bit, cost per direct access bit, etc.), it becomes increasingly more economical to use system resources to perform functions that otherwise are handled by installation personnel.

The other new characteristic of virtual storage is that it enables a given system configuration to provide a wider range of performance, as well as function, as a result of the new factors that affect operation of a system with virtual storage support. Thus, a slightly different approach must be taken in planning for and in evaluating system performance in a virtual storage environment.

Many of the same factors that affect system performance in a DOS/VS, OS/VS1, or OS/VS2 environment are the same as those that apply to DOS Version 4, OS MFT, or OS MVT, respectively. First, the system configuration must include the hardware resources (CPU speed, channels, I/O devices, real storage) required for the control program and job mix. This subsection identifies the system resources specifically required to support a virtual storage environment. Second, the system should be designed to balance resource usage to achieve optimum throughput, and to use applicable performance and control program design options the particular operating system offers, taking into account the characteristics of the installation job stream.

The performance of a system in a virtual storage environment is also affected by certain new factors that do not apply to systems without virtual storage support. This subsection identifies these new factors, explains how they generally affect system performance, and indicates steps that can be taken to increase and maximize system performance when a virtual storage operating system is used.

This discussion applies to DOS/VS, OS/VS1, and OS/VS2, and is restricted to performance factors that are common to the virtual storage environments they support. The virtual storage operating systems also offer new performance-oriented enhancements that are not related to the implementation of virtual storage. These unique performance features are discussed in the optional programming systems supplements.

The performance information in this subsection is designed to present concepts and considerations for a virtual storage environment. Figures and graphs are used for illustrative purposes. They do not represent any particular installation or measured results. Their purpose is to illustrate the interrelated factors of multiprogramming performance in a virtual storage environment. The performance information presented is conceptual. It is based on the experience and judgment of IBM individuals with performance knowledge and on performance measurements made during development of OS/VS1 and OS/VS2. Therefore, it may not apply to all installations.

SYSTEM RESOURCES REQUIRED TO SUPPORT A VIRTUAL STORAGE ENVIRONMENT

In order to support a demand-paged virtual storage environment using System/370, in which programs are operating in paged mode, additional system resources are used by the IBM-supplied virtual storage operating systems, as follows:

- Dynamic address translation hardware requires CPU time to perform virtual-storage-to-real-storage address translation. The amount of time required is determined by the System/370 model and the number of times the full table-lookup translation procedure must be performed. The Model 158, for example, has a translation lookaside buffer that is designed to reduce use of the full table-lookup translation procedure. The CPU time required is also affected by

program structure (which is discussed later). A small amount of additional CPU time is also required to pretest certain instructions that reference storage, as discussed under "Instruction Nullification" in Section 30:10. Studies have shown that a relatively small percentage of the total CPU time specifically required to support a virtual storage environment is devoted to address translation by DAT hardware.

- CPU time is required to translate the virtual storage addresses in channel programs (CCW lists) into real storage addresses and to build indirect data address lists (when necessary) and short-term fix pages that will be referenced during I/O initiation, execution, and interruption handling. Channel program translation and page fixing are performed before the initiation of each I/O operation with a channel program that contains virtual storage addresses. Channel status word retranslation and page unfixing are performed at the completion of these I/O operations. The amount of CPU time this function requires per data set is affected by the number of I/O requests (EXCP macros) issued, the number of CCW's in the channel programs started, the number of pages that must be fixed, and whether indirect data address lists have to be constructed. Studies have shown that a large portion of the total CPU time specifically required to support a virtual storage environment is used to perform channel program translation and page fixing.

- CPU time is required to process page faults and for the execution of other control program code that is specifically required to support a virtual storage environment. CPU time is required for such things as servicing additional program interruptions, managing and allocating real and external page storage, maintaining tables used by DAT hardware, and testing for paged or nonpaged mode of program operation.

- I/O time is required for paging operations. The amount of paging I/O time required is related to the number of page faults that occur and the speed of the paging I/O device(s) used. In OS/VS2 environments, the total I/O time required for paging includes some I/O time that is also required in OS MVT environments to load transient control program routines.

- Direct access storage is required for external page storage. The amount required depends on the amount of virtual storage that is to be supported and the way in which the particular operating system organizes and manages external page storage. (See the optional programming systems supplements for external page requirements by device type.)

- The amount of real storage required by the resident (fixed) control program is increased by the amount of real storage needed for additional routines and code that are included specifically to support a demand paged virtual storage environment.

The effect this additional use of hardware resources has on the performance of a given system configuration depends on the resource requirements of the job stream and the current utilization of system resources. To the degree that the additional required CPU and I/O time can be overlapped with existing CPU and I/O time that currently is not overlapped, system throughput is not affected. System throughput will be affected by the increase in CPU and I/O time that cannot be overlapped.

When a virtual storage operating system is used with an existing system configuration, for example, and the same job stream is processed, performance is affected by the use of any new performance enhancements these operating systems provide as well as by an increase in resource

utilization that is required to support a virtual storage environment. When a Model 158 replaces a Model 155, performance is also affected by the fact that the Model 158 has faster internal performance than the Model 155.

Figure 30.15.1 conceptually illustrates possible system performance when a virtual storage operating system is installed on a Model 158 with the same amount of real storage and the same I/O device configuration as the replaced Model 155. A sample throughput for a Model 155 is shown in panel 1. (It is not meant to represent any specific Model 155 throughput.) Panels 2 and 3 illustrate the conditions under which existing performance can be maintained, and the last two illustrate the conditions under which existing performance can be improved. Existing throughput is maintained if both of the following occur:

1. A portion of the additional CPU and I/O time required to support a virtual storage environment is overlapped with CPU and I/O time that previously was not overlapped, as shown by points A in panel 2.

2. The amount of additional CPU and I/O time that cannot be overlapped (shown by points B in panel 2) is offset by reductions in previously used CPU and I/O time that occur as a result of the faster internal performance of the Model 158 and use of new performance features of the virtual storage operating system, as shown in panel 2. The unoverlapped CPU and I/O time may also be offset by a combination of the faster internal performance of the Model 158 and the achievement of better overlap as a result of operating the system at a higher level of multiprogramming to process the same work (as shown in panel 3).

Existing system throughput can improve if (1) unoverlapped CPU and I/O time required to support a virtual storage environment is exceeded by reductions in previously used CPU and I/O time and/or if previously used CPU and I/O time are better overlapped (as shown in panel 4) or (2) a higher level of multiprogramming is used to perform more work and provide better CPU and I/O overlap in the same elapsed time (as shown in panel 5).

NEW FACTORS THAT AFFECT SYSTEM PERFORMANCE

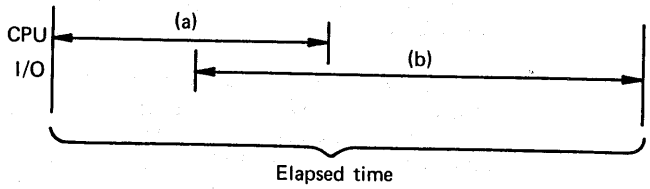
In addition to the factors that affect performance in a nonvirtual storage environment, the performance of a system in a virtual storage environment is affected by the relationship of the following factors: the speed and number of paging devices, the speed of the CPU, the size of real storage, the structure of the programs in the job stream, and the way in which real storage is organized and allocated by the virtual storage operating system. The interrelationship of each of these factors and their individual effect on performance, except for the last factor listed, are as follows (page replacement algorithms are not discussed):

Speed and Number of Paging Devices. A certain amount of I/O time is required to read in (or write out) a page using a given direct access device type. This time is a function of device type characteristics--seek time, rotation time, and data transfer rate. Assuming one page-in performed at a time, no page-outs, and no contention for the paging device or its channel, a maximum paging rate, in terms of the number of page faults that can be serviced per time interval, could be calculated for a given device type. This rate could be improved by certain programming techniques, such as use of rotational position sensing when it is present and initiation of multiple page-in and page-out requests with a single channel program. (Various techniques are implemented in OS/VS1 and OS/VS2.) The maximum paging capability of a given system can

be increased by various means, such as using a faster paging device or, in OS/VS1 and OS/VS2 environments, using more than one paging device.

Panel 1

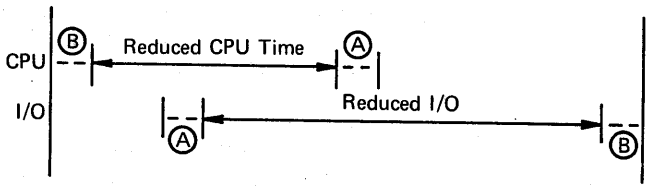
Sample existing CPU and I/O utilization and overlap for a Model 155.



EXISTING SYSTEM THROUGHPUT MAINTAINED

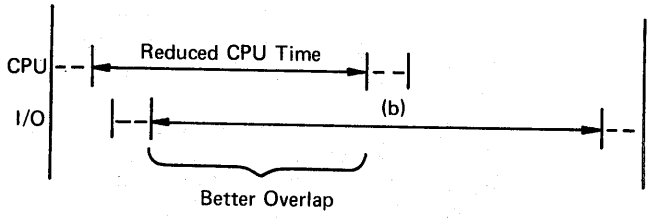
Panel 2

Some of the additional CPU and I/O time required is overlapped with previously unoverlapped I/O and CPU time (points A). Additional CPU and I/O time that cannot be overlapped (points B) is offset by a reduction in the amount of CPU and I/O time required to process the same job stream. Results are achieved in the same elapsed time.



Panel 3

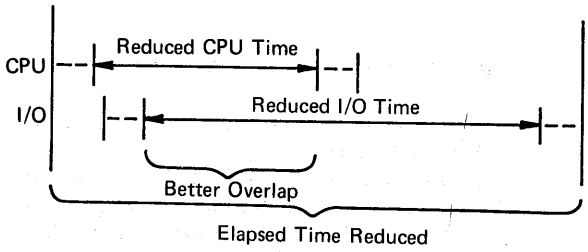
Additional CPU and I/O time required (dotted lines) is overlapped and offset by operating the system at a higher level of multiprogramming to achieve greater overlap. Results are achieved in the same elapsed time.



EXISTING SYSTEM THROUGHPUT IMPROVED

Panel 4

Unoverlapped CPU and I/O time required is exceeded by reductions in previously used CPU and I/O time. Better overlap of previously used CPU and I/O time is also achieved. Same results are achieved in less elapsed time.



Panel 5

A higher level of multiprogramming is used to perform more work and achieve better overlap of CPU and I/O time. More results are achieved in the same elapsed time.

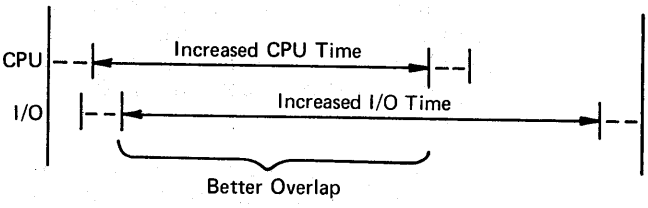


Figure 30.15.1. Possible system performance when a virtual storage operating system is used with a Model 158 with the same I/O configuration and real storage size as the replaced Model 155

The paging characteristic of a virtual storage environment is the feature that permits an operating system to support virtual storage that is larger than real storage. The paging activity of a system begins to adversely affect system performance, however, once the CPU is in the

position of frequently having to wait for paging I/O operations to complete. When requests for paging operations are permitted to occur faster than the paging rate the system can sustain, so that the system can do little or no processing except that related to paging, the system is in a paging-I/O-bound situation and is said to be thrashing. When a thrashing condition exists, little or no productive work can be accomplished unless paging activity is reduced.

In order to prevent thrashing, the System/370 virtual storage operating systems monitor the activity of the system to determine when paging activity becomes excessive. At this point, task deactivation is performed. This involves placing a task (OS/VS2) or partition (OS/VS1 and DOS/VS) in deactivated status. When the page frames associated with a deactivated partition or task become available, they can be allocated to other tasks to reduce paging activity. Later, when paging activity becomes sufficiently low, the deactivated partition or task is reactivated.

CPU Speed. An improperly balanced relationship between CPU speed and paging device speed can also cause the system to become I/O-bound as a result of paging. A Model 158 can execute a certain number of instructions during the time required to service a page-in request using a given direct access device type. A Model 158 can execute many more instructions during a page-in from a 2314, for example, than can a Model 145. As long as there is useful work for the CPU to perform while paging operations occur, the system is not kept waiting for paging I/O. However, if the concurrently operating programs are constantly executing instructions faster than the pages they require can be brought into real storage, an excessively high paging rate could develop if task deactivation were not invoked. In general, therefore, the larger-scale System/370 models require faster paging devices to handle a particular page fault rate than do the smaller-scale models.

Real Storage Size. The amount of real storage present in a system affects the number of page faults that occur when a given job stream is processed. If the amount of real storage present in the system is equal to the total amount of virtual storage being used by the concurrently executing tasks, no page faults occur for programs that have been fetched and initiated. When the amount of real storage present is less than the amount of virtual storage being used, page faults occur. The total number of page faults that occur for a given job stream is affected by the ratio of virtual storage used to real storage available.

Assuming the amount of virtual storage used in a given system remains the same, the virtual-to-real storage ratio can vary. This occurs while a given system experiences variations in the amount of real storage actually available for paging as the amount of fixed real storage changes during job stream processing. The real storage available for paging at any point in time is the difference between the amount of real storage in the system and the total amount of long- and short-term fixed real storage. For IBM-supplied virtual storage operating systems, the total amount of fixed real storage at any given time is the sum of the:

- Resident (fixed) control program size, which does not vary after IPL
- Amount of long-term fixed real storage required for control blocks, which can change as the level of multiprogramming changes (OS/VS1 and OS/VS2 only)
- Amount of short-term fixed real storage required for outstanding I/O operations that have virtual channel programs, which fluctuates with the I/O activity of the system
- Amount of long-term fixed real storage required by the job steps executing in nonpaged (real) mode, if any

- Amount of long-term fixed real storage required by programs that operate in paged mode but that have a portion of their partition or region always fixed (TCAM in OS/VS1 and OS/VS2, BTAM in DOS/VS, for example)

As the virtual-to-real storage ratio of a job stream increases, so usually does the page fault rate. In general, the page fault rate increases slowly for a while. At some point, the increase in page faults begins rising rapidly as the virtual-to-real storage ratio continues to increase. Figure 30.15.2, shown later, illustrates the general relationship between the number of page faults and the virtual-to-real storage ratio.

The amount of real storage available to process a given job stream also varies when a given job stream is processed on systems with various amounts of real storage, such as when a smaller-scale system is used to back up a larger-scale system.

The degree to which reducing the real storage available for paging affects the page fault rate depends on the paging activity pattern of the programs in a job stream. Therefore, the virtual-to-real storage ratio at the point at which a given number of page faults occur will usually vary by job stream. The point can also be different for systems with similar paging activity patterns and the same amount of real storage installed but with different amounts of long-term fixed real storage.

As the virtual-to-real storage ratio increases because of a reduction in the real storage available (or an increase in the amount of virtual storage used) and the page fault rate increases, more demand is placed on the paging devices. If too small an amount of real storage is present in a system, this situation can cause the page fault rate to exceed the permissible rate and task deactivation will occur. In general, therefore, in order to obtain a certain level of performance, a configuration that supports a given job stream and virtual storage size may require more real storage when a relatively slower paging device is used than when a faster paging device is used.

Program Structure. The total amount of virtual storage a program uses is not nearly so significant a factor in system performance as the way in which virtual storage is used. That is, the pattern and frequency of reference to pages in a program has more effect on the number of page faults that occur than the total size of the program. For example, assume a case in which a program has a 100K virtual storage design point. If the program can be structured to execute as a series of logical phases of four or five pages each and the pages of each logical phase reference only each other, no more than four or five page frames (8K to 10K or 16K to 20K of real storage, depending on page size) need be dynamically available to the program at one time and paging activity occurs only as the program progresses from one logical phase to the next. However, assume the program is structured so that during its execution each page of instructions constantly references a large number of different pages of instructions and data for very short durations on a highly random basis. An excessively high paging rate could occur if only four or five page frames were dynamically available to such a program at any time.

As indicated previously, most types of programs have a natural locality of reference characteristic, so that they can be structured to operate as a series of logical phases. In the simplest case, for example, a program can logically consist of an initialization phase, a main phase, one or more exception handling phases, and a termination phase. The total amount of virtual storage referenced in each logical phase usually varies but, generally, the amount is less than the total size of the program. In addition, the pages that are part of

(referenced in) a given logical phase can usually be described as active or passive.

For the purpose of the discussion in this subsection, an active page is defined as one with a high probability of being referenced multiple times during execution of the logical phase, while a passive page has a low probability of being referenced more than once during execution of the phase. A logical phase experiences the least amount of paging activity as it executes when its active pages remain in real storage during its execution and its passive pages are paged in when required. A program uses real storage most efficiently when the active instructions and data in each logical phase are contained within the fewest number of pages possible.

The locality of reference characteristic does not apply to certain types of programs. For example, it does not apply to any program that is designed to optimize its performance at execution time by using the total amount of storage it has been allocated. This characteristic is usually true of sort/merge programs that initialize themselves to use all the storage made available to them in their partition or region during the sorting passes. The reference pattern for such a sort/merge is random and encompasses all the storage (and, therefore, all the pages) the program is assigned.

RELATIONSHIP BETWEEN VIRTUAL STORAGE SIZE AND SYSTEM PERFORMANCE

Assuming other required system resources are available, a given configuration can support a given virtual storage size and provide satisfactory performance when paging activity is kept at an acceptable level. Minimal paging activity occurs when enough real storage is present in the system to contain most or all of those pages of concurrently executing programs that are active at any given time. Paging activity is then required primarily for passive pages. Active pages are paged in (and later paged out as required) as the set of active pages for each program changes from one logical phase to another. The paging device(s) present must be capable of handling the demand for pages that results from the range of paging activity of the system.

As the amount of virtual storage used in a given system increases, the number of active and passive pages that the system must handle increases also. The ratio of active to passive pages will vary for a given increase in virtual storage, depending on how the additional virtual storage is used. As long as enough real storage is present to contain all or most of the increased number of active pages, the increase in paging activity required to support the additional virtual storage will be needed primarily for passive pages and should be relatively small. As soon as the use of more virtual storage causes the number of concurrently active pages to constantly exceed the capacity of real storage, the paging activity increase required to support the additional virtual storage becomes relatively large. As more and more active pages must be handled, paging activity could exceed the maximum paging capability of the system if task deactivation did not occur.

Figure 30.15.2 illustrates the increase in page faults that generally occurs as more virtual storage is used in a given system configuration. The curve begins at the point at which the amount of virtual storage used is equal to the amount of real storage present (virtual-to-real-storage ratio is 1). Paging activity begins as soon as the amount of virtual storage used exceeds the real storage present. As the virtual-to-real-storage ratio increases, so does paging activity. The system moves from passive paging activity (primarily paging of passive pages) into active paging (paging active pages in and out more of the time) and approaches the maximum paging capability of the system. As indicated previously, Figure 30.15.2 also illustrates the increase in page faults

that generally occurs as less real storage is made available to support a given virtual storage size. The increase in page faults also causes the virtual-to-real-storage ratio to increase.

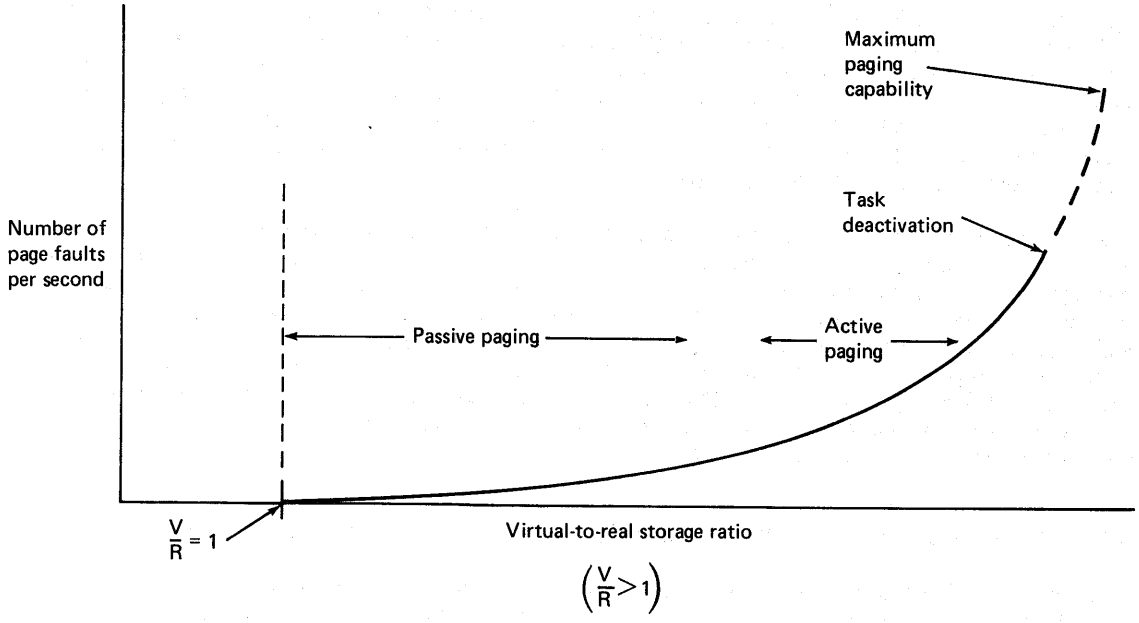


Figure 30.15.2. General effect on page faults of increasing the ratio of virtual storage used to real storage present in the system

Figure 30.15.3 illustrates how the paging factor only generally affects system performance. Figure 30.15.5, shown later, illustrates system performance taking into account all factors. The curve shows the performance of the system when passive and active paging are occurring, relative to the virtual-to-real storage ratio. The use of virtual storage can be increased with little or no adverse effect on performance as long as paging remains in the passive area. This is true because in the passive paging area there is a relatively small amount of paging and a high probability that all or most paging processing (CPU and I/O time) can be overlapped with other processing. As paging activity increases, there is a higher probability that CPU processing will be held up waiting for a paging operation to complete. As the CPU enters the wait state more frequently to wait for paging I/O and less paging I/O is overlapped, the paging factor causes performance to degrade more rapidly.

The actual virtual-to-real storage ratio at the time active paging begins in Figures 30.15.2 and 30.15.3 is a variable and depends on the way in which virtual storage is used, that is, active-to-passive page ratio of concurrently executing tasks.

Figure 30.15.4 illustrates the way in which the paging factor only can affect system performance in a given configuration, based on the active-to-passive page ratio. If the ratio of active to passive pages for executing tasks is relatively high most of the time, as shown in curve 1, the virtual-to-real storage ratio at the point at which active paging begins will be relatively low. Performance drops very rapidly in this case as more virtual storage is used. This happens because the increased paging processing (I/O and CPU time) cannot be overlapped with other processing. This situation may apply to an installation initially when a switch from a nonvirtual storage to a virtual storage environment

is made and more virtual storage is used, since existing programs were structured for optimum performance in a given partition or region size rather than for optimum performance in a virtual storage environment.

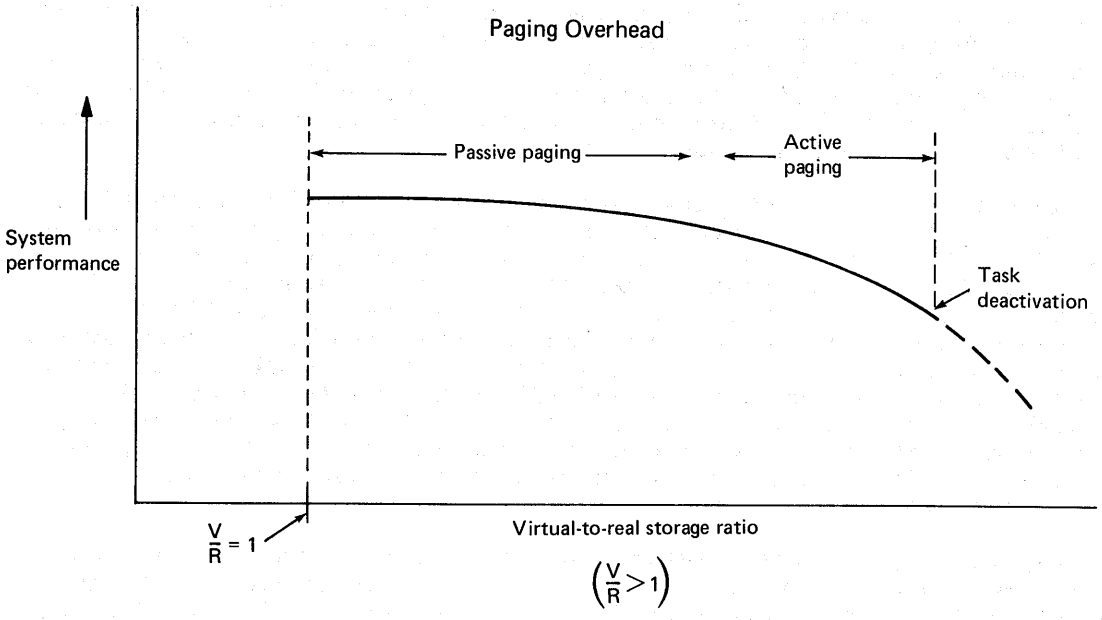


Figure 30.15.3. General effect on system performance of the paging factor only

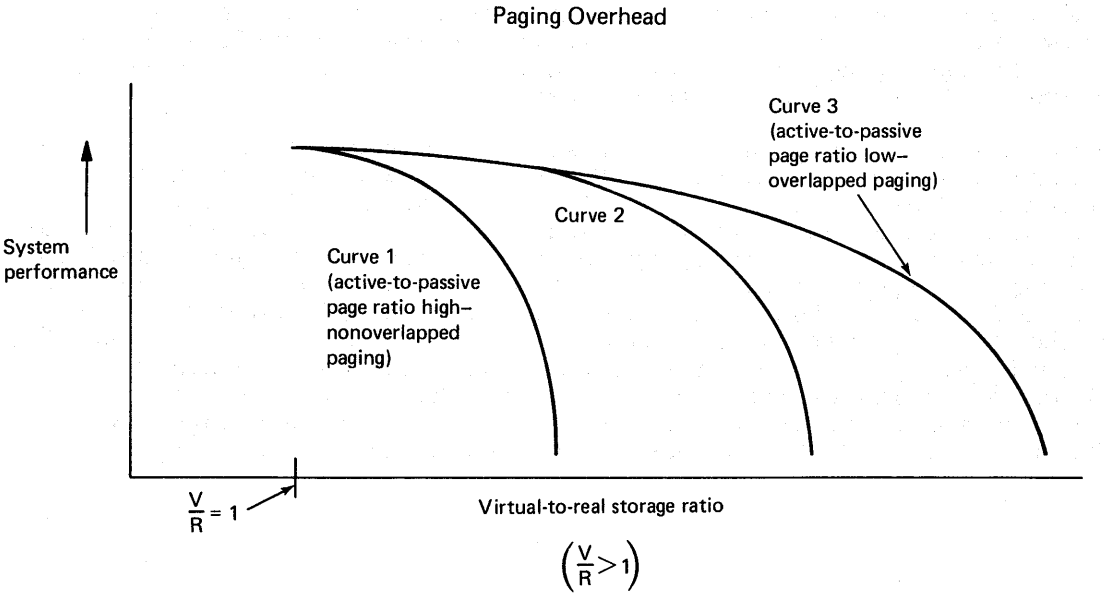


Figure 30.15.4. General effect of the paging factor on system performance for various active-to-passive page ratios

If the active-to-passive page ratio for the system is low, as shown in curve 3, the virtual-to-real storage ratio can be relatively high when active paging begins. The performance curve stays flatter longer as virtual storage is increased when the active-to-passive page ratio is low. This situation can apply to an installation in which all executing programs are structured to minimize real storage requirements and page faults. An installation that continues executing all or most existing programs as they are presently designed and that structures new applications for most efficient operation (low active-to-passive ratio) may be more common. Such installations may experience a virtual-to-real storage ratio somewhere between the low and the high extremes possible for a given job stream, as shown in curve 2.

The amount of virtual storage used in a system can be increased in several ways. First, the size of existing application programs can be increased by the addition of new functions. Second, the level of multiprogramming or multitasking can be increased, assuming other required resources such as CPU time and I/O devices are available. Third, the size of existing application programs can be expanded by (1) restructuring programs with a planned overlay or a dynamic structure to take them out of these structures and (2) combining two or more job steps within a job into one logical job step. The active-to-passive ratio of the additional pages the system must handle will usually be higher when the level of multiprogramming is increased than when existing jobs are restructured.

The way in which an installation should view the amount of virtual storage used and system performance for a given configuration, taking all performance factors into account, is illustrated in Figure 30.15.5. The increased use of virtual storage is beneficial to system performance up to a point. Thereafter, additional virtual storage can be used to handle additional functions at a variable cost in system performance. In reality, the virtual-to-real storage ratio and the page fault rate vary during system processing as the amount of virtual storage used (out of the total amount supported) and the amount of real storage available for paging vary. Best overall system performance is achieved when paging activity falls most of the time in the area identified on the curve as the operating range. More significant performance reduction begins when active paging is experienced.

Occasional active paging on an exception basis should be acceptable. More frequent active paging can be performed to achieve a desired function that does not justify changing the system configuration. However, when paging activity in a system is constantly at the point at which task deactivation occurs, system configuration changes should be made to improve system performance. Such changes might be the addition of more real storage, the addition of more (in OS/VS1 and OS/VS2 environments) or faster paging devices, or installation of a faster CPU, A history of the paging activity of the system can be maintained by recording the paging statistics provided by the virtual storage operating systems. OS/VS1 and OS/VS2 provide page-in and page-out statistics, while DOS/VS provides a page fault trace capability.

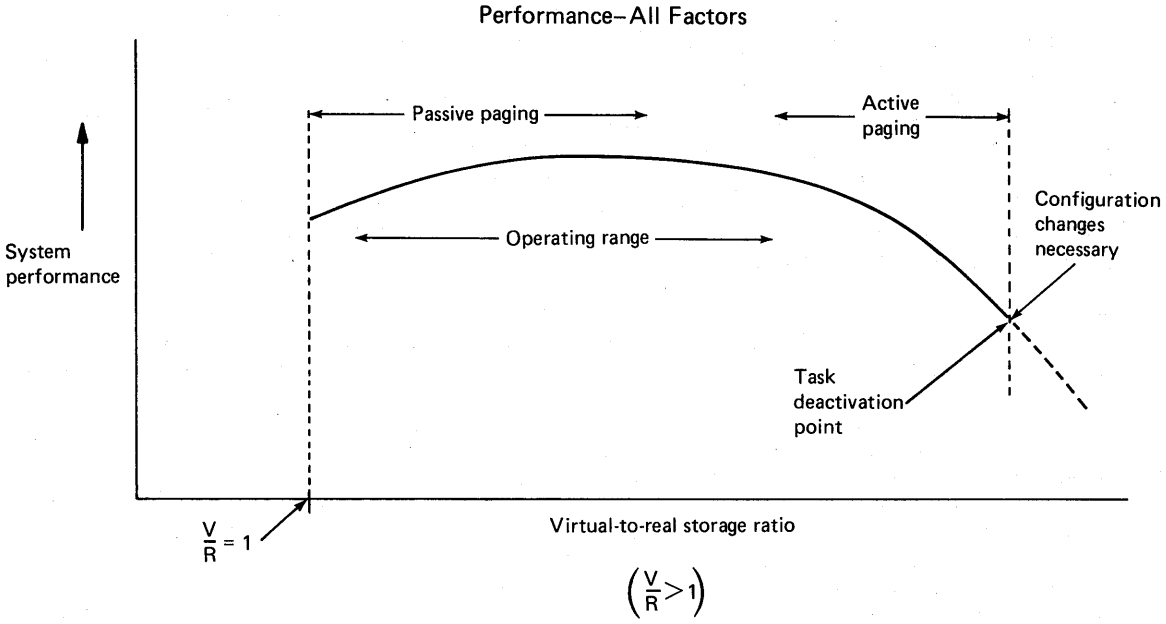


Figure 30.15.5. General system performance curve for a virtual storage environment

INCREASING SYSTEM PERFORMANCE IN A VIRTUAL STORAGE ENVIRONMENT

The IBM-supplied virtual storage operating systems are designed to provide an acceptable level of performance when existing problem programs are run without modification. However, given the additional resource requirements of virtual storage support and the new factors that affect system performance in a virtual storage environment, once a virtual storage operating system is installed (either on an existing configuration or a larger configuration) certain steps can be taken to improve performance or to achieve optimum performance. The benefit to be achieved by taking any one of the steps outlined must be evaluated on an installation basis, taking the specific configuration and operating environment into account. Some steps, for example, are more practical for large configurations than for small configurations. The following can be done:

- Use larger I/O buffers. This step is practical primarily for sequential data sets and can be used most effectively when previous real storage limitations have prevented the use of larger buffer sizes in general and, in particular, optimum buffer sizes for disk data sets. In addition to reducing the total I/O time required to process a data set, as would occur in a nonvirtual storage environment, increasing buffer size reduces the number of I/O requests required to process the data set, and thereby reduces the CPU time required for channel program translation and page fixing. This technique should be used taking into account the amount of real storage present in the system. If the buffer size of several data sets that are being processed concurrently is increased considerably or made large initially, the amount of real storage that must be short-term fixed increases considerably also and potentially increases the number of active pages. This may adversely affect system performance in systems with a relatively limited amount of real storage available for paging.

- Increase the page fault handling capability of the system when paging activity constantly causes task deactivation. This can be accomplished by: (1) using a direct access device for paging that is faster than the currently used paging device, (2) allocating more direct access devices for paging to enable more overlap of paging activity (OS/VS1 and OS/VS2 environments only), or (3) reducing or eliminating contention for the existing paging device(s). Contention for the paging device can be relieved by using dedicated paging devices, reducing the amount of other I/O activity on the channel to which the paging device is attached, or dedicating a channel to paging. Alternatively, the same paging device configuration can be maintained while page fault occurrence is decreased by the addition of real storage.

- Use code that does not modify itself. Use of this type of code can reduce the amount of page-out activity required. Such code can be produced using the Assembler Language (OS and DOS environments) and OS PL/I language translators.

- Execute programs in nonpaged (real) mode only when actually required. Use of nonpaged mode should be limited because the amount of real storage available for paging operations during the operation of a nonpaged program is reduced by the size of the program and can affect system performance. If a nonpageable program is to be present in a system for an extended period of time or at all times, it should be considered part of the fixed real storage requirement so that the amount of real storage actually available for paging can be more accurately determined.

- Structure new application programs to operate efficiently in a paging environment. This is done by structuring programs to achieve a reasonable balance between page faults and real storage requirements. The extent to which this is done can vary widely by installation. The benefits that can be obtained should be evaluated in light of the additional programmer effort required. In this respect, deciding on the degree to which programs should be structured for efficient operation in a paging environment is similar to deciding how a high-level language should be used. The emphasis can be on most efficient program execution, which can require more programmer effort, or on most efficient use of programmer time, which can result in less efficient programs. Alternatively, there can be a tradeoff between programmer time and efficient programs (only the most frequently or heavily used programs are optimized, for example).

  Many of the general program structure techniques discussed do not require a large amount of additional effort or knowledge on the part of programmers--only that they adopt a particular programming style. All of the suggested techniques can be used by Assembler Language programmers. Some can be used with certain high-level languages and not with others. More of the suggested techniques can be used in PL/I programs than in other high-level language programs.

  Two major steps can be taken to structure programs to use real storage most efficiently and to incur the smallest possible number of page faults. The first is to adopt a certain programming style, one aspect of which is similar to the style that has been encouraged with System/360 and System/370, namely, that of modular programming. The second is to take page boundaries into account and to package program code and data into page groups. The objective of improving programming style is to construct a program that consists of a series of logical processing phases each of which contains a relatively small number of active pages. The objective of packaging code within pages is to group active code together to avoid crossing page boundaries in such a way that more real storage than is really

necessary is required to contain the active pages of a logical phase.

In order to cause references to active instructions and data to be localized, the following general rules should be applied to programs:

1. A program should consist of a series of sequentially executed logical phases or--in System/370 programming terminology--a series of subroutines or subprograms. The mainline of the program should contain the most frequently used subroutines in the sequence of most probable use, so that processing proceeds sequentially, with calls being made to the infrequently used subroutines, such as exception and error routines. This structure contrasts with one in which the mainline consists of a series of calls to subroutines. Frequently used subroutines should be located near each other. Infrequently used subroutines that tend to be used at the same time whenever they are executed should be located near each other also.

2. The data most frequently used by a subroutine should be defined together so that it is placed within the same page, or group of pages, instead of being scattered among several pages. If possible, the data should be placed next to the subroutine, so that part or all of the data is contained within a page that contains active subroutine instructions (unless the routine is to be written so that it is not modified during its execution). This eliminates references to more pages than are actually required to contain the data and tends to keep the pages with frequently referenced data in real storage.

3. Data that is to be used by several subroutines of a program (either in series or in parallel by concurrently executing subtasks) should be defined together in an area that can be referenced by each subroutine.

4. A data field should be initialized as close as possible to the time it will be used, to avoid a page-out and a page-in between initialization and first use of the data field.

5. Structures of data, such as arrays, should be defined in virtual storage in the sequence in which they will be referenced, or referenced by the program in the sequence in which a high-level language stores them (by row or by column for arrays, for example).

6. Subroutines should be packaged within pages when possible. For example, avoid starting a 1500-byte subroutine in the middle of a 2K page so that it crosses a page boundary and requires two page frames instead of one when it is active. Subroutines that are smaller than page size should be packaged together to require the fewest number of pages, with frequently used subroutines placed in the same page when possible. This applies to large groups of data as well. The linkage editor supplied with OS/VS1 and OS/VS2 has new control statements that can be used to cause CSECTs and COMMON areas to be aligned on page boundaries, and to indicate the order in which CSECTs are placed in the load module. This linkage editor facility can be used with certain high-level language programs that contain multiple CSECTs (such as PL/I and ANS COBOL) as well as with Assembler Language programs.

- Use the PL/I Optimizing Compiler available for DOS and OS instead of OS PL/I F or DOS PL/I D. The code produced by these language translators has characteristics that make it more suited to a virtual storage environment than the code produced by the Type I PL/I language translators. First, generated code is grouped into functionally related segments, by PROCEDURE and DO group, for example, which can help reduce paging. When PL/I allocates buffers and I/O control blocks, they are packed together and can potentially require fewer pages than if no attempt was made to define them together. Reentrant code can be produced by the OS PL/I Optimizing Compiler, and its library routines are reentrant. This reduces page-out requirements. User-written reentrant OS PL/I routines that are required by concurrently executing problem programs can be made resident in virtual storage and shared to reduce real storage and paging requirements for active pages of these routines.

- Use the shared library feature of the OS PL/I Optimizing Compiler and the COBOL Library Management Facility of the OS ANS COBOL language translators to make library modules resident in virtual storage so they can be shared by concurrently executing problem programs. Pages containing active library modules will tend to remain in real storage and thereby reduce paging and real storage requirements for these modules.

- Restructure existing application programs to incur as few page faults as possible, to use the least possible amount of real storage, and to take advantage of the program structure facilities that a virtual storage environment offers. This can be accomplished by (1) using the techniques described above, (2) taking planned overlay and dynamic structure programs out of these structures, and (3) combining into one logical job step two or more steps of a job that would have been one job step if the required real storage were available. The last of these techniques can eliminate redundant I/O time that is currently used for such things as reading the same sequential input data into two or more job steps and writing intermediate results from one job step in one or more sequential data sets for input to the next job step.

- Increase the level of multiprogramming in the system. This can be accomplished by (1) performing more peripheral I/O operations concurrently (more readers and writers in OS/VS, use of POWER in DOS/VS), (2) operating more regions or partitions concurrently, or (3) increasing the use of multitasking (structuring a DOS/VS QTAM or an OS/VS TCAM message processing program to use multitasking to enable several different types of transactions to be processed concurrently, for example).

  System throughput can be improved in a virtual storage environemnt if a higher level of multiprogramming causes more CPU and I/O time to be overlapped, which results in more effective utilization of system resources. The larger the number of tasks in the system under these conditions, the less chance there is for the CPU to enter the wait state because no task is ready to execute. Better utilization of real storage in a virtual storage environment can enable more tasks to be present in the system.

  In order to achieve performance gains by increasing the level of multiprogramming, the potential for more overlap of CPU and I/O time must exist in a system, and/or the potential must exist for reduction of I/O time via increased overlapping of channel activity and reductions in unoverlapped seek time (that can result from new system performance enhancements). The required hardware resources, such as CPU time, real storage, I/O devices, and direct access storage, must be available as well. The most critical resource in this situation is available CPU time. As the percentage of CPU

utilization gets higher, there is less potential for increasing throughput via increasing the level of multiprogramming.

The information presented in this subsection has been designed to enable the reader to more fully understand the way a system operates in a virtual storage environment and the factors that influence system performance. Understanding the environment and knowing the actions that can be taken to increase system performance will enable each installation to quantify the amount of effort that is to be devoted to optimizing the performance of a virtual storage operating system.

SECTION 40: VIRTUAL MACHINES

This section discusses the basic concepts, general operation, and advantages of virtual machines, as defined and implemented in Virtual Machine Facility/370. No previous knowledge of virtual machines is assumed. The virtual machine concept is a logical extension of the virtual storage concept. Therefore, comprehension of dynamic address translation hardware and virtual storage concepts, terminology, and advantages, as discussed in Sections 30:05 and 30:10, is assumed.

VM/370 consists of the Control Program (CP) component, the Conversational Monitor System (CMS) component, and the Remote Spooling Communications Subsystem (RSCS) component. CP supports the concurrent operation of multiple virtual machines. CMS, operating in a virtual machine under CP control, provides conversational time sharing facilities to a single user. RSCS, operating in a virtual machine under CP control, provides for the transmission of data between remote users and virtual machines via binary synchronous communication lines.

VM/370 is the successor to CP-67/CMS. Virtual machine support was first provided by IBM in CP/67. In the CMS time sharing environment in which CP-67/CMS was primarily used, the major advantage of the virtual machine facility was that it enabled each CMS user to appear to have a complete System/360 (Model 22 to 75) at his disposal and to be isolated from all other CMS users. Each CMS user had access only to his own virtual machine and, therefore, could not inadvertantly interfere with the operation of other CMS virtual machines. VM/370 also provides these facilities and can be used in nondedicated time sharing environments to provide other advantages as well.

The information presented in this section is prerequisite reading for the optional Virtual Machine Facility/370 Features Supplement, which can be inserted as Section 110 of this guide. The VM/370 supplement discusses the features and operation of CP and CMS, as well as performance considerations for a virtual machine environment and the types of installations that can benefit most from the use of VM/370.

40:05 DEFINITION AND GENERAL OPERATION

A virtual machine is a functional simulation of a complete computer system, including a virtual CPU, virtual storage, virtual channels, virtual I/O devices, and a virtual operator's console, that appears to the user to be a real machine. In a VM/370 environment, a virtual machine is the functional equivalent of a System/370 (Models 135 to 168) and its associated I/O devices.

The control program (CP) component of VM/370, executing in a real machine (System/370 Models 135 through 168 with dynamic address translation hardware), supports concurrent operation of multiple virtual machines using multiprogramming techniques that enable real machine resources to be shared by multiple virtual machines. Each virtual machine is dedicated to a single user and isolated from other virtual machines. None of the components of one virtual machine can be accessed by a program that is executing in another virtual machine except via the controlled sharing facilities that are provided by CP.

The operation of a virtual machine and scheduling of the work it performs are handled by an operating system rather than by CP. That is, each virtual machine has an operating system executing in it that allocates machine resources and schedules the execution of problem

programs just as if the operating system were executing in a real machine. In order to initiate operations in a virtual machine, the user must log on the virtual machine and IPL an operating system in it. The logon procedure establishes a connection with CP and the existence of a specific virtual machine for this user. A logon is performed using a console or terminal device of the type that CP supports as a virtual operator's console.

The virtual operator's console is the means by which the user controls the operation of his virtual machine and communicates with the operating system executing in it. CP provides a set of commands that (1) simulate the system control panel of the virtual machine, (2) provide for alteration of a virtual machine configuration, (3) request various services from CP for a virtual machine, and (4) control operation of the real machine. When a CP command is entered via the virtual operator's console, CP receives control and performs the required functions.

Communication between the user and the operating system is accomplished using the operating system command language and the virtual operator's console. CP performs any simulation required to make the real I/O device the operator is using as a virtual operator's console appear to be the primary console device type that is defined for the operating system.

In a VM/370 environment, a virtual operator's console is frequently called a remote terminal because, in most cases, a terminal device type is actually used as the virtual operator's console device. However, the real I/O device that is used as the virtual operator's console may be a System/370 console device as well as a local or a remote terminal.

VM/370 supports execution of any one of the following System/360 and System/370 programming systems in a virtual machine:

- CMS component of VM/370
- RSCS component of VM/370
- DOS Version 3, DOS Version 4, or DOS/VS
- APL 360-DOS
- OS PCP, MFT, or MVT
- OS ASP Version 3
- OS/VS1
- CS/VS2 Release 1
- OS/VS2 Releases 2 and up in uniprocessor mode only
- PS44
- VM/370

Any number and combination of the above operating systems can execute concurrently in a VM/370 environment, subject to the availability of the required real machine resources, including multiple copies of the same operating system (OS/VS1 executing in more than one virtual machine, for example). With a few exceptions, all the facilities that are supported by these operating systems when they execute in a real machine can be used when the operating system executes in a virtual machine in a VM/370 environment. Figure 40.05.1 conceptually illustrates the real and virtual machine environment that is supported by VM/370.

Each virtual machine that is to be supported by CP must be user defined and stored in the VM/370 directory. The size of virtual storage, the virtual I/O devices to be used, the options to be used, and a virtual console are usually specified. Virtual machine configurations can be different from each other and, within certain limitations, different from that of the real machine in terms of these specifications.

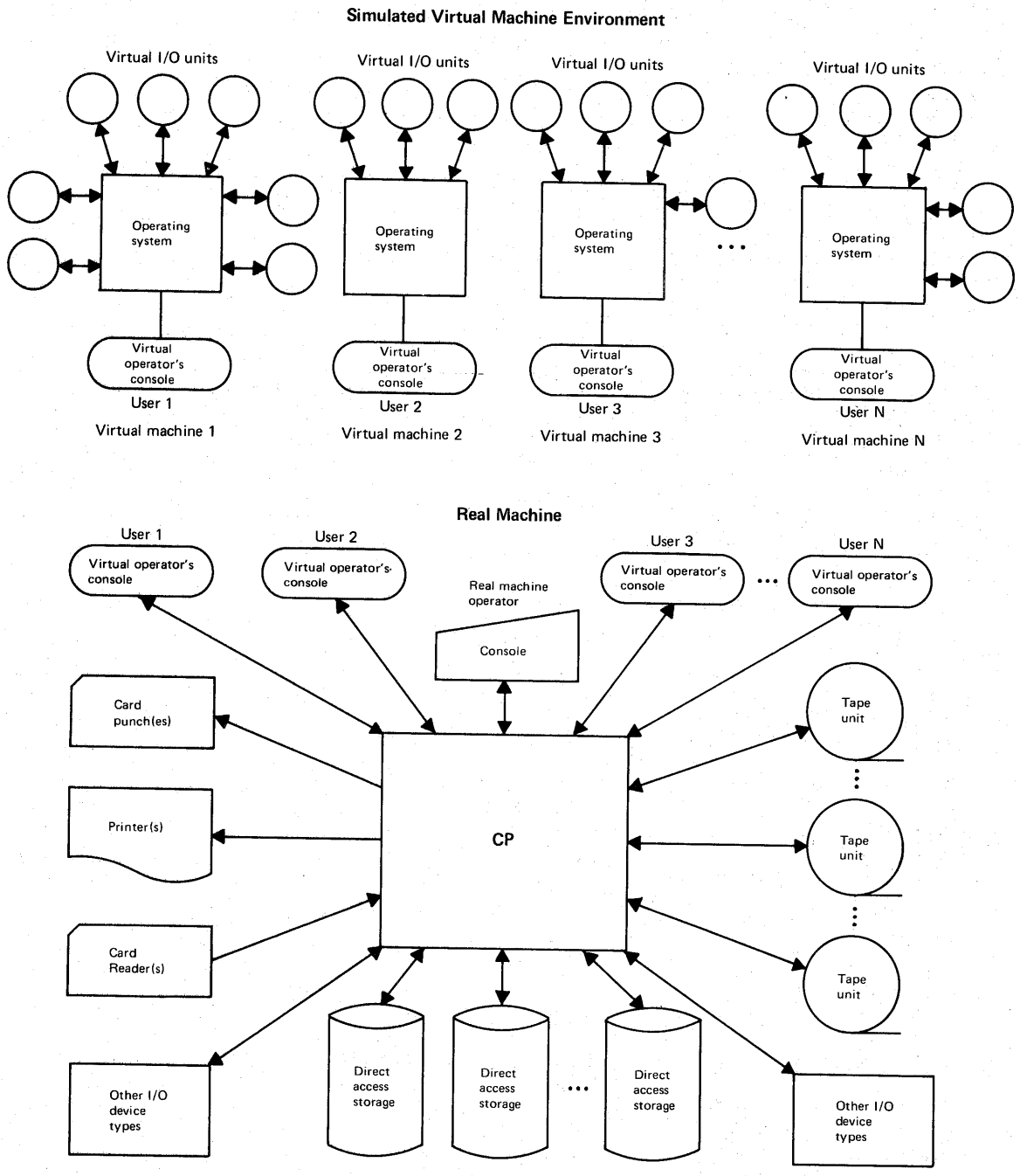


Figure 40.05.1. Conceptual illustration of the real and virtual machine environment that is supported by VM/370

Virtual CPU Simulation

CP is resident in real storage during operation of the real machine. It controls the operation of the real machine, schedules the execution of virtual machines, and simulates virtual machine hardware components using the hardware components of the real machine. In order to be able to perform its functions and isolate virtual machines from each other, CP must have exclusive control over the status and modes of operation of the real machine, as does the control program of an operating system. Hence, CP always executes with the real machine in supervisor state and receives control after all real machine interruptions.

Virtual machines always operate with the real machine in problem state. Therefore, any time any program that is executing in a virtual machine issues a privileged instruction, an interruption occurs in the real machine. CP receives real CPU control and takes the required action. This may involve simulating execution of the privileged instruction for the virtual machine or returning real CPU control to the control program in the virtual machine for which the interruption occurred so that the interruption can be processed by that control program. In this manner, CP maintains control of the real machine. In addition, CP simulates the existence of both a supervisor state and a problem state in the virtual machine while, in reality, the virtual machine operates only in problem state.

CP gives control of the real CPU to operating virtual machines on a time-shared basis to simulate the existence of multiple CPU's. A virtual machine can execute any System/370 instruction except READ DIRECT and WRITE DIRECT, which are part of the Direct Control feature, the multiprocessing instructions, and SET CLOCK, which is treated as a NOP because CP controls the setting of the time of day clock. In addition, the DIAGNOSE instruction is reserved for communication between executing operating systems and CP.

The System/370 instructions and CPU features that are used by the control and problem programs executing in a virtual machine must be present in the CPU of the real machine in which CP executes. CP does not simulate the existence of System/370 instructions and CPU hardware features that are not present in the real machine. A virtual CPU can appear to be executing either with BC mode or EC and DAT modes specified, depending on the mode required by the operating system executing in it. However, EC and DAT modes are always specified in the real CPU when a virtual CPU is executing since address translation is required to support the existence of virtual storage for the virtual machine.

Virtual Storage Simulation

Each virtual machine can have up to 16,777,216 bytes of virtual storage, which is the maximum virtual storage size for System/370. The existence of virtual storage for a virtual machine is simulated by CP using DAT hardware and external page storage, as is done in a virtual storage environment (discussed in Section 30).

Operating system programs that are executing in a virtual machine (both control and problem programs) are paged in and out of real storage in the real machine on a demand paged basis as they execute. Real storage allocation, external page storage allocation, and paging operations are handled entirely by CP and are transparent to the control and problem programs that are executing in the virtual machines. In this manner, CP provides one virtual storage for each virtual machine, and real storage in the real machine is shared by concurrently operating virtual machines. The implementation of virtual storage in a virtual machine environment is conceptually illustrated in Figure 40.05.2.

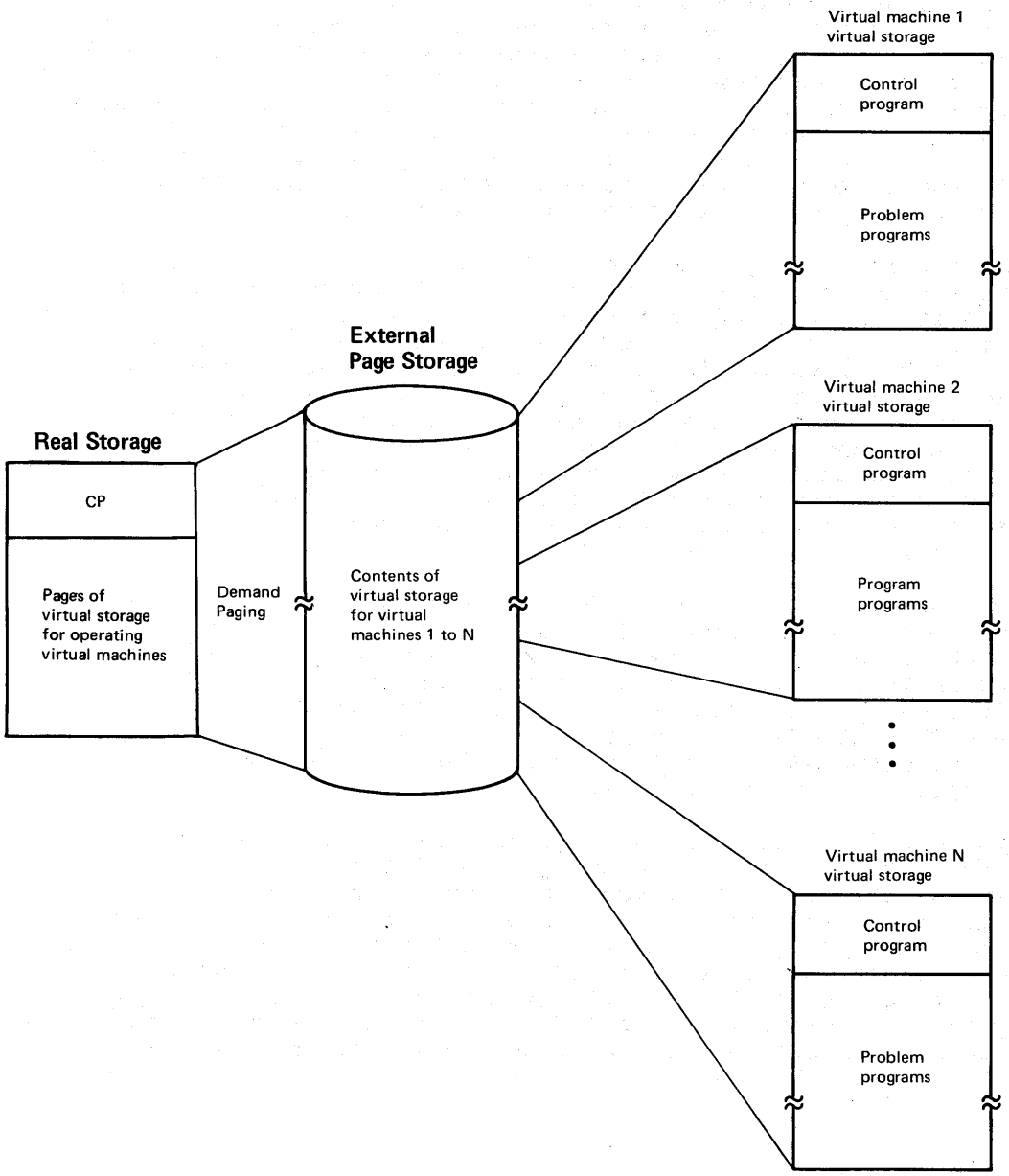


Figure 40.05.2. Conceptual illustration of the implementation of virtual storage in a virtual machine environment

The virtual storage defined for a virtual machine always appears to be real storage to the operating system that is executing in the virtual machine. In effect, an operating system that does not support virtual storage, such as DOS Version 4 or OS MFT, has virtual storage support provided by CP when such an operating system executes in a virtual machine and, therefore, offers the functional advantages of a virtual storage operating system.

When executing in a virtual machine, an operating system that does support virtual storage uses the virtual storage defined for the virtual machine as real storage in order to simulate the existence of the virtual storage it is designed to support. As shown in Figure 40.05.3, the virtual storage operating system builds a segment table and page

tables to translate addresses in the virtual storage it supports to addresses in the virtual storage defined for the virtual machine, which the operating system assumes is real storage. CP always builds and maintains a segment table and page tables for each virtual machine. These tables are used to translate addresses in the virtual storage of the virtual machine to addresses in real storage in the real machine.

When a virtual storage operating system is executing in a virtual machine, CP constructs and maintains a third set of tables using the contents of the other two sets of tables. The third set of tables, a shadow segment table and shadow page tables, are the tables that are actually used for address translation when the virtual machine operates. The shadow tables are used to translate addresses in the virtual storage the operating system supports to addresses in real storage in the real machine.

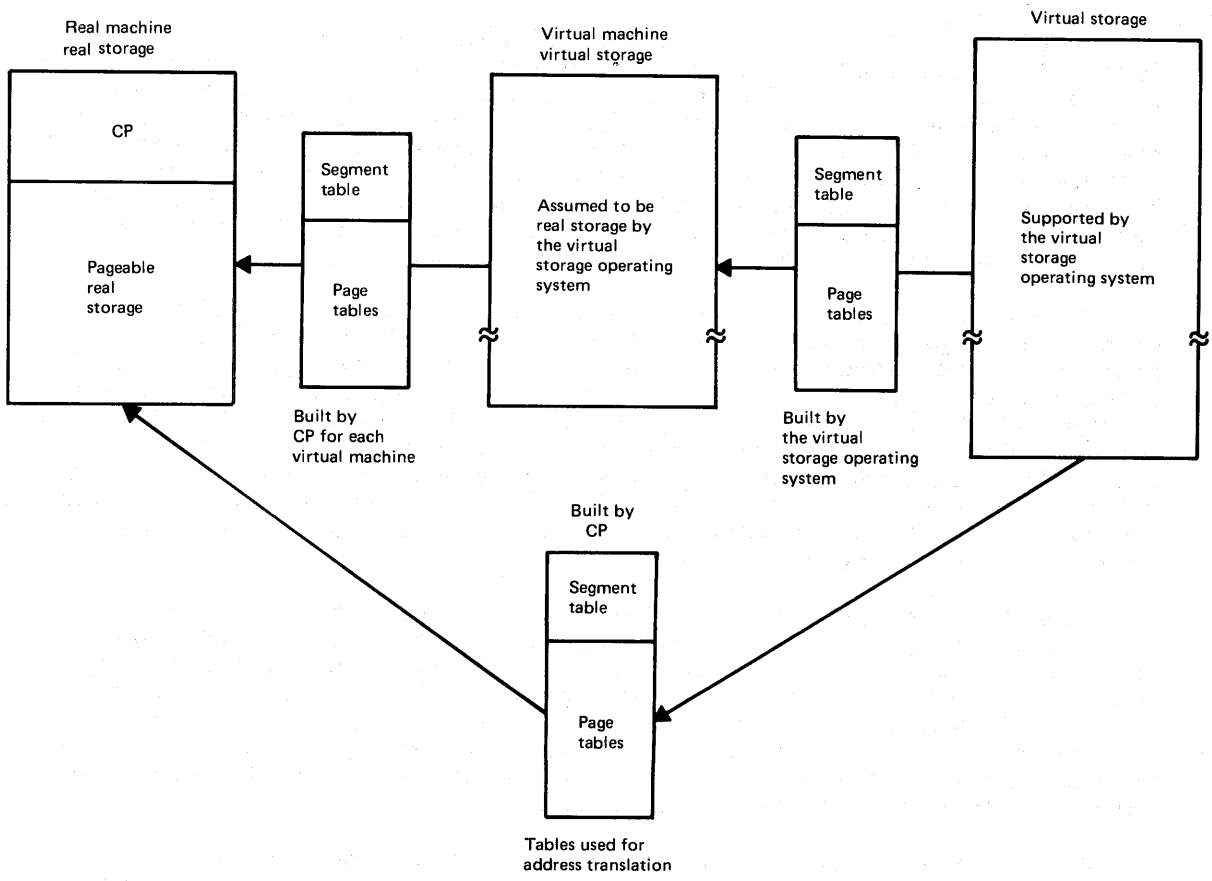


Figure 40.05.3. Segment tables and page tables built when a virtual storage operating system executes in a virtual machine

Virtual I/O Component Simulation

The virtual channels, control units, and I/O devices defined in each virtual machine configuration are simulated by CP using real channels, control units, and I/O devices that are of the same type. While each virtual I/O device defined must have a real I/O device counterpart in the real machine configuration, there does not necessarily have to be a one-to-one correspondence. In addition, the I/O device addresses assigned to virtual I/O devices need not be the same as the addresses of their real I/O device counterparts. CP also allows a virtual direct

access device to be simulated by only a portion of a real direct access device volume. Such a virtual direct access device is called a minidisk. Support of a minidisk facility enables one real direct access device to simulate the existence of several virtual direct access devices of the same type and thus provides more efficient use of available direct access storage.

Virtual I/O devices are always simulated on a real I/O device of the same device type unless the spooling facility of CP is used. (CP also allows 2311 disk storage to be simulated using 2314/2319 disk storage and the minidisk facility.) The local spooling capability of CP provides data transcription between unit record devices and direct access storage devices and is functionally similar to DOS POWER, OS readers and writers, OS HASP, and OS/VS JES. In effect, the CP spooling facility enables virtual unit record devices (card readers, card punches, and printers) to be simulated using direct access storage. CP also provides console spooling and a remote spooling facility.

The virtual I/O devices in a virtual machine configuration are logically controlled by the operating system that is executing in the virtual machine rather than by CP. That is, all the data management routines of the operating system (physical record processing, logical record processing, and error recovery routines) execute as usual. Therefore, a virtual machine I/O configuration can include any I/O device types that are supported by the operating systems that will execute in the virtual machine, as long as real I/O device counterparts exist in the real machine I/O configuration as required.

CP controls only the scheduling and actual initiation of virtual machine I/O operations in the real machine. When a START I/O instruction is issued by an operating system control program that is executing in a virtual machine, a privileged operation interruption occurs and CP receives real CPU control. CP translates the virtual I/O device address to its counterpart real I/O device address and, for minidisks, converts virtual cylinder addresses to corresponding real cylinder addresses, as required. CP also performs the necessary channel program translation and page locking operations and queues the I/O request if it cannot be started.

After the I/O operation is started, CP returns the condition code to the operating system control program that initiated the I/O request so that appropriate action can be taken. When the I/O operation completes and causes an I/O interruption, CP receives CPU control, gathers I/O status information, and attempts to restart the available real I/O components. CP presents the status data to the operating system control program via a simulated I/O interruption for the virtual machine in which the operating system is executing.

CP completely controls operation of the real I/O devices that are required for its own execution, such as paging and spooling devices. This includes determining the need for I/O operations, scheduling and initiating I/O requests, handling I/O interruption processing, and performing error recovery procedures.

Virtual Machine Assist Feature

The optional, no-charge Virtual Machine Assist (VMA) feature can be field installed on a Model 158. This feature is designed to improve total system performance in a VM/370 environment and can also improve the performance achieved by certain operating systems that operate under CP control in a virtual machine. The VMA feature performs the same functions as some of the most frequently used virtual machine simulation routines of CP. When the VMA feature is used, virtual machine performance improvement results when CP processing is eliminated that

otherwise would cause an operating system to experience throughput degradation when it executes in a virtual machine instead of a real machine. Total system performance improvement is achieved if a higher level of multiprogramming can be maintained as a result of the elimination of certain CP processing.

The VMA feature is controlled by mask bits in control register 6. When the VMA feature is enabled, certain types of real machine interruptions that occur when a virtual machine has real CPU control cause the VMA hardware feature to gain control to simulate the required virtual machine function. The VMA feature is entered when one of the following occurs:

- A privileged instruction program interruption occurs that is caused when a virtual machine issues an INSERT PSW KEY, INSERT STORAGE KEY, LOAD PSW, LOAD REAL ADDRESS, RESET REFERENCE BIT, SET PSW KEY FROM ADDRESS, SET STORAGE KEY, SET SYSTEM MASK, STORE CONTROL, STORE THEN AND SYSTEM MASK, or STORE THEN OR SYSTEM MASK instruction. The VMA feature simulates execution of the privileged instruction, and operation of the virtual machine continues with execution of the instruction after the privileged instruction.

- An SVC instruction except SVC 76 is issued by a virtual machine. PSW switching for the virtual machine is simulated by the VMA feature.

- A page translation program interruption occurs in a virtual machine in which a virtual storage operating system is executing. The VMA feature updates the appropriate shadow page table if possible.

The VMA hardware feature performs the same functions as the counterpart simulation routines in CP, with a few exceptions. The VMA feature does not handle certain special situations for a few of the privileged instructions supported. The unsupported special situations are those that occur infrequently and that would require the inclusion of a considerable amount of additional hardware. When these special situations occur, the appropriate simulation routine of CP is entered to perform the required functions.

The amount of throughput improvement that occurs for an operating system when the VMA feature is used depends on the extent to which the operating system utilizes the functions the VMA feature supports. If the increase in run time an operating system experiences when it executes in a virtual machine is caused to a large extent by the CP processing that is required to simulate VMA supported functions, a relatively significant performance gain can be expected. The VMA feature can be of the most benefit, for example, to operating systems that support virtual storage (DOS/VS, OS/VS1, and OS/VS2).

The VMA feature is supported by VM/370 as of Release 2. Additional details regarding the operation of the VMA feature and the support provided by VM/370 are discussed in Virtual Machine Facility/370 Features Supplement, GC20-1757-1, and later editions.

The VMA feature for the Model 158 is mutually exclusive with the 7070/7074 Compatibility feature. VMA microcode is contained on the N-disk. The microcode for the 7070/7074 Compatibility feature is contained on the S-disk. During IMPL, 7070/7074 Compatibility feature microcode is loaded into reloadable control storage from the S-disk as usual. If the VMA feature is to be used, the operater must cause the VMA microcode to be loaded from the N-disk. This is accomplished using the configuration frame of the display console. VMA microcode overlays the 7070/7074 Compatibility feature microcode in reloadable control storage when loaded. The choice of using one or the other of these two features can be made only at IMPL time.

40:10 GENERAL ADVANTAGES OF A VIRTUAL MACHINE ENVIRONMENT

The advantages of VM/370 complement those of virtual storage operating systems. Like a virtual storage environment, a virtual machine environment is designed primarily to support new functions rather than increase system performance. Essentially, CP is a simulator. Traditionally, simulators have been used to provide a desired function at the expense of performance. The new functions provided by virtual machines (1) can increase the rate of new application development and (2) expand operational capabilities over those provided by virtual storage. The CMS component of VM/370 supplements these two major advantage areas of a virtual machine environment by supporting time sharing facilities such as online program development, conversational program execution and problem solving, and interactive text processing.

The following indicates the way in which the virtual machine environment that is supported by the CP component of VM/370 aids the installation of new applications and identifies the new operational features such an environment supports. The functions and specific advantages of CMS are discussed in the VM/370 supplement.

Increasing New Application Development

Since virtual machine support includes support of a virtual storage environment for each virtual machine, all the capabilities virtual storage provides that aid new application development are present in a virtual machine environment as well. (These capabilities are discussed at the end of Section 30:05.) By enabling multiple operating systems to execute concurrently in one real machine, the virtual machine environment supported by CP also provides the following new capabilities:

- Testing of new programs can be more extensive and completed sooner by the elimination of dedicated testing periods. While a virtual storage environment can eliminate most program testing restrictions that result from real storage size limitations, the isolation that is provided by executing a program in a virtual machine eliminates the need to test programs that can cause total system termination in a dedicated environment. For example, system-oriented routines written by system programmers and teleprocessing programs, which usually are tested only during scheduled dedicated testing periods, can be tested while production work is in progress. This can eliminate the need to establish testing periods during second or third shift and, by reducing individual test turnaround time, enables more of this type of testing to be accomplished in a given time period.

- Testing of new programs can be completed sooner through the use of console debugging, when necessary. Using the CP commands that simulate system control panel functions, the programmer can use any console debugging facility that is available on a real machine, such as setting address stops, examining and altering general registers, displaying and altering virtual storage, etc., without interfering with production work. CP also provides other debugging services, such as an extensive set of traces, that can be invoked by CP commands. Console debugging, which can enable difficult-to-locate program errors to be detected more quickly than with desk debugging, is usually not permitted in a nonvirtual machine environment, except as a last resort, or is scheduled for nonproduction periods. Program testing turnaround time can be significantly reduced through the use of console debugging.

- Transition from one release of an operating system to another release or from one operating system to another can be accomplished more quickly because of the capability of executing multiple operating systems concurrently. A new release of an operating system can be generated and tested in one virtual machine while production work continues in another virtual machine using the existing release. Existing application programs and system-oriented programs that must be modified or newly written (to use a new facility or new language translator, for example) can be tested during production processing as well. The multiple virtual machine facility also enables an installation to execute programs that are dependent on a back release (because the release is user modified, for example) concurrently with each new release of that operating system or with an entirely new operating system (such as a back release of a DOS version operating concurrently with OS).

- CMS can be used to perform online program development concurrently with the processing of production work using either OS or DOS. Significant gains in programmer output can be realized through writing, compiling, and testing programs using an online terminal in a conversational manner. This enables new applications to become operational sooner. When CMS is used, each programmer has his own virtual machine with CMS executing in it. Therefore, the occurrence of a programming or operational error in one virtual machine can cause termination of that virtual machine only. Other programmers and production work are not affected.

Expanded Operational Capabilities

In addition to the new operational facilities a virtual storage environment provides (discussed in Section 30:05), a multiple virtual machine environment offers the following capabilities:

- Operating system maintenance can be performed concurrently with production work. PTF's can be applied and tested using one virtual machine without the possibility of causing the abnormal termination of another virtual machine that is processing production work.

- Operator training can be done using a virtual machine, which eliminates the need to dedicate the entire real machine to this function. Multiple operators can be trained while production work is in process without the possibility of terminating real system operations through an operator error.

- A system can be backed up by another system that not only has less real storage but that also has real I/O devices with different addresses, fewer direct access devices, and fewer channels, as long as sufficient I/O devices of the required type are available.

- New channel and direct access device configurations can be simulated using a virtual machine for the purpose of evaluating the load on the new I/O configuration before it is installed on the real machine. Similarly, ASP configurations consisting of two or more machines can be simulated in a virtual machine environment using only one real machine. This enables an installation without ASP installed to determine the activity of such a configuration and gain experience in its operation before the second system is installed or before making the decision to install ASP. The ASP user can also experiment with different ASP configurations.

As the above indicates, a virtual machine environment, as supported by VM/370, offers several unique capabilities that can be of benefit to small, intermediate, and large System/370 users. In most cases, VM/370 can be used to best advantage as complementary programming system

support in installations in which a version of DOS or OS is used as the primary programming system. VM/370 can be used in the same system as the DOS, DOS/VS, OS, or OS/VS operating system or in a separate support system. A discussion of the types of installation environments in which VM/370 will be most frequently used is contained in Virtual Machine Facility/370 Features Supplement.

SECTION 50: I/O DEVICES FOR MODELS 1 AND 3

50:05 I/O DEVICE SUPPORT

All I/O devices, console devices, and telecommunications terminals that can be attached to the Model 155 can be attached to Models 1 and 3 of the Model 158 with the following exceptions: The 3210 Model 1 and the 3215 Model 1 Printer-Keyboards cannot be installed on a Model 158 as the primary operator console device. The 3210 Model 2 printer-keyboard cannot be installed as a remote console. In addition, the Integrated Storage Controls feature and several other I/O devices attach to the Model 158 but not to the Model 155 (see the table in Section 70:05).

Note that all I/O devices supported by OS MFT, OS MVT, and DOS Version 4 are not supported by OS/VS1, OS/VS2 Release 1, and DOS/VS, respectively. (See the programming systems supplements for I/O device support.)

The I/O devices discussed in this section attach to a Model 158 (Models 1 and 3) but not to be a Model 155.

50:10 3333 DISK STORAGE AND CONTROL MODEL 11 AND 3330 DISK STORAGE MODEL 11

A 3330-series string that is attached to a Model 158 can contain 3333 Model 11 Disk Storage and Control and 3330 Model 11 Disk Storage modules, which do not attach to the Model 155. The drives in these modules offer twice the capacity of the drives in Model 1 and 2 modules. Model 11 of the 3333 consists of two independent drives, device-oriented control functions, and power for itself and the drives that can be attached to it, as does Model 1 of the 3333. Model 11 of the 3330 consists of two independent drives without the device-oriented control functions that are part of a 3333, as does a 3330 Model 1.

In a Model 158 configuration, the 3333 Model 11 attaches to 3830 Storage Control Model 2 and Integrated Storage Controls (ISC). It must be the first module in each 3330-series string that is attached to these control units. The 3330 Model 11 attaches only to 3333 modules, Models 1 and 11. Up to three 3330 modules, in any combination of Models 1, 2, and 11, can be attached to a 3333 Model 1 or 11 module.

With one exception, Model 11 3330-series drives are functionally like Model 1 and 2 drives. The drives in 3330 and 3333 Model 11 modules have a standard write format release feature that is not provided for 3330 Model 1 and 2 and 3333 Model 1 drives. This feature enables a Model 11 drive to disconnect from a 3333/3330 Model 11, 3830 Model 2, or the ISC while the drive is erasing to the end of the track after a record has been written with a formatting write command. This facility frees the control unit and channel for the initiation of another I/O operation.

The removable 3333 Model 11 disk pack is used with 3333 and 3330 Model 11 drives. Like a 3336 Model 1, a 3336 Model 11 has 19 recording surfaces. However, the Model 11 disk pack has 808 data cylinders, instead of 404, for a maximum capacity of 200 million bytes. The Model 11 disk pack also has seven alternate cylinders, like a Model 1. Hence, the maximum capacity of a 3330-series string of all Model 11 drives is 1600 million bytes.

Model 11 3336 Disk Packs are interchangeable across all 3330 Model 11 and 3333 Model 11 drives but cannot be used with Model 1 and 2 3330-series drives. The 3336 Model 11 Disk Pack has a physical interlock so that it cannot be mounted on a 3330 Model 1 or 2 drive or a 3333 Model 1

drive. The 3336 Model 1 Disk Pack has a physical interlock so that it cannot be mounted on a Model 11 drive. The 3336 Model 1 Disk Pack can be converted to a Model 11.

Table 50.10.1 compares Model 1, 2, and 11 drive characteristics. Table 50.10.2 compares 3336 Model 1 and 11 Disk Pack characteristics.

Table 50.10.1. Capacity and timing characteristics of 3330-series drives

Characteristic	3330-series Model 1 or 2 drive	3330-series Model 11 drive
Capacity in thousands of bytes (full-track records)	100,018	200,036
Seek time (ms)		
Maximum	55	55
Average	30	30
Average cylinder-to-cylinder	10	10
Time channel busy searching when SET SECTOR is used (ms)		
Minimum	.120	.120
Maximum	.380	.380
Rotation time (ms)	16.7	16.7
Rotation speed (rpm)	3600	3600
Data transfer rate (KB/sec)	806	806

Table 50.10.2. 3336 Model 1 and 11 Disk Pack characteristics

Characteristic	3336 Model 1	3336 Model 11
Number of disks per pack	12	12
Number of recording disks	10	10
Number of recording surfaces	19	19
Disk thickness in inches	.075	.075
Disk diameter in inches	14	14
Disk pack weight in pounds	20	20
Disk pack maximum capacity in millions of bytes	100	200
Full track capacity in bytes	13,030	13,030
Cylinders per pack	404 plus 7 alternates	808 plus 7 alternates
Tracks per cylinder	19	19
Tracks per pack	7,676	15,352

ATTACHMENT VIA INTEGRATED STORAGE CONTROLS

Optionally, one Integrated Storage Controls feature can be installed on a Model 158 to attach 3330-series and/or 3340 disk storage to one or two block multiplexer channels. Attachment of 3330-series and 3340 disk storage via 3830 Storage Control is possible as well. The following discusses attachment to the ISC of 3330-series strings only.

The Integrated Storage Controls feature includes dual direct access storage controls, each of which operates independently of the other and is functionally like 3830 Storage Control Model 2 except for the following:

- The Integrated Storage Controls feature is contained in the main frame of the Model 158 and is powered by the Model 158 CPU.

- The Two Channel Switch, Additional feature (that provides four-channel switching) cannot be attached to the logical storage controls in the ISC feature.

Both logical storage controls in the ISC feature can be attached to the same channel or to two different channels in the Model 158. Each logical storage control can have attached a maximum of four 3330-series strings of up to eight drives each. The 32 Drive Expansion and Control Store Extension optional features (field installable) must be installed in the ISC in order to attach more than two strings to each logical control. Therefore, up to 64 drives (eight strings) can be attached to the Model 158 via the ISC. The first module in each 3330-series string must be a 3333 Disk Storage and Control Model 1 or 11 unit.

The 3330-series drives attached to ISC operate just as if they were attached via 3830 Storage Control Model 2. That is, when multiple requesting is used, each logical storage control within the ISC can handle up to 32 channel programs concurrently, one on each of its drives, and only one of the 32 drives can be transferring data at a time. When a malfunction occurs, diagnostics can be run on one logical storage control and its drives while normal operations take place on the other logical storage control in the ISC.

The ISC feature provides lower-cost attachment of 3330-series disk storage than 3830 Storage Control Model 2 when two storage control units are required, and physical space is saved since the ISC is in the Model 158 CPU. See Table 50.15.3 for a summary of the capabilities of the 3830 Models 1 and 2 and ISC.

The Two Channel Switch optional feature is also available for the ISC. When installed, this feature provides a two-channel switching capability for both of the logical storage controls. The Two Channel Switch permits each logical storage control to be attached to two channels in the same Model 158 or to one channel in the Model 158 and one channel in another System/370. Two switches are provided that can be set to dedicate a logical storage control unit to one channel or the other, or to enable the storage control to be accessed by both channels. Figure 50.10.1 summarizes the 3330-series string configurations that are possible for a Model 158 ISC. Intermixing 3330-series and 3340 strings on an attachment is discussed in Section 50:15.

The 3333 String Switch optional feature can be installed on a 3333 Model 1 or 11 that is attached to the 3830 Model 2 or ISC. This field-installable feature enables the 3333 and all its attached 3330s (a 3330-series string) to be connected to two control unit type attachments instead of only one. The attachments can be any combination of two of the following:

- 3830 Storage Control Model 2

- Integrated Storage Controls for Models 158 and 168 (or the two logical controls in one ISC)

- Integrated Storage Control for the Model 145

- 3345 Storage and Control Frame Models 3, 4, and 5 for the Model 145

- 3330/3340 Series IFA for the Model 135

The two attachments to which a 3333 with the 3333 String Switch feature is connected can be attached to the same or different channels in the same CPU, or to channels in two different CPU's. In addition,

channel-switching features can be installed on one or both of these attachments.

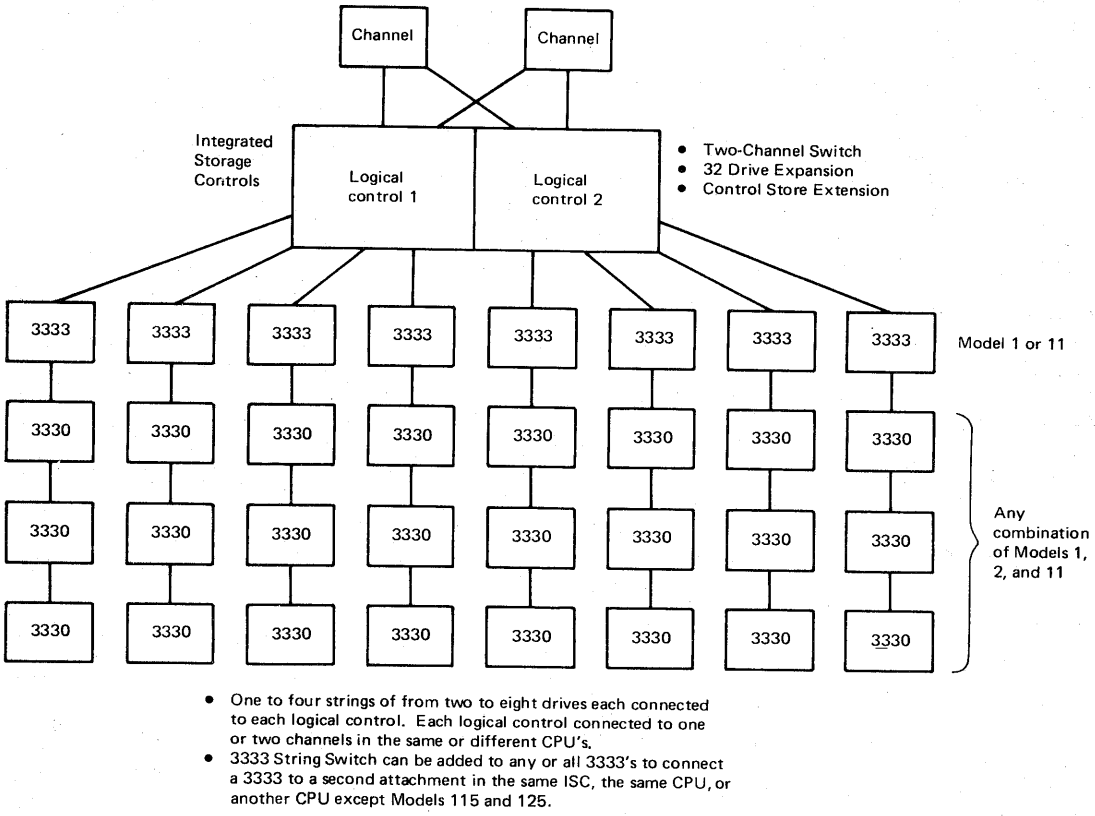


Figure 50.10.1. Permissible 3330-series string configurations for the Model 158 integrated storage controls feature

The 3333 String Switch is functionally similar in its operation to the Two-Channel Switch. A switch can be set to allow the 3330-series string to be accessed via both attachments, one at a time. In effect, the setting provides two control unit paths to the string. String switching is accomplished dynamically under program control. Alternatively, the switch can be set to dedicate the string to one attachment or the other so that the string can be accessed only via that attachment.

Figure 50.10.2 illustrates 3333 string switching for four 3330-series strings that are attached to the same ISC. In the configuration shown, all strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3330-series disk storage and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

Optionally, the staging adapter feature can be installed on the ISC to permit attachment of the 3850 Mass Storage System to the ISC. The ISC provides the same functions for the 3850 as 3830 Storage Control Model 3. The staging adapter permits the addressing capability of each of the four ISC paths to be expanded to a maximum of 64 unique addresses. When the staging adapter is installed, the control store extension feature must also be installed and 3340 disk storage cannot be attached to the ISC.

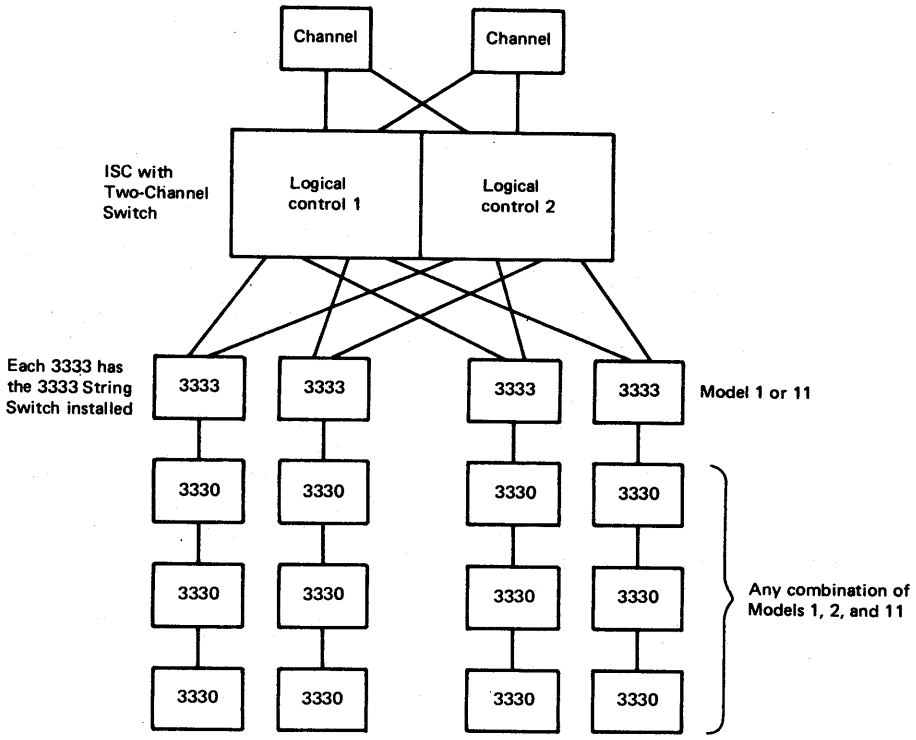


Figure 50.10.2. Sample 3330-series string configuration with string switching

50:15 THE 3340 DIRECT ACCESS STORAGE FACILITY

3340 DISK STORAGE DRIVES AND THE 3348 DATA MODULE

The 3340 direct access storage facility is an intermediate capacity, modular, high performance direct access storage subsystem that consists of 3340 Disk Storage and Control Model A2 and 3340 Disk Storage Models B1 and B2. A 3340 string can consist of from one to four units and is connected to a Model 158 block multiplexer channel via 3830 Storage Control Model 2 or integrated storage controls in the Model 158 CPU.

A 3340 string for the Model 158 can consist of from two to eight drives. A 3340 Disk Storage and Control Model A2 must be the first unit in a 3340 string. The 3340 Model A2 consists of two drives, drive-oriented control functions, and power for itself and the 3340 drives attached to it. In a Model 158 configuration, the 3340 Model A2 attaches to 3830 Storage Control Model 2 and a logical control in the ISC. Up to three units, any combination of 3340 Disk Storage Models B1 and B2, can be attached to a 3340 Model A2. The 3340 Model B2 consists of two drives and does not contain the power and device-oriented control functions that are part of the 3340 Model A2. The 3340 Model B1 contains one drive and no control functions. Functionally, all 3340 drives are alike regardless of whether they are part of a Model A2, B2, or B1 unit.

Figure 50.15.1 shows a 3340 string of five drives that includes one 3340 Model A2, one 3340 Model B2, and one 3340 Model B1. An operator control panel is located on the top of each 3340 drive. This panel contains the three-digit hexadecimal address of the drive, the switches required to operate the drive, and status indicator lights. The address of a 3340 drive is wired on a logic board in the 3340 unit.

The removable 3348 Data Module is used for data storage. Unlike the removable 2316 and 3336 disk packs that are the storage medium for 2314 and 3330-series disk storage, respectively, the 3348 Data Module is a sealed cartridge that contains a spindle, access mechanism, and read/write heads in addition to disks on which data is written and read. The cover of the data module, which is shock-absorbing and non-flammable, is never removed from the cartridge. The 3340 disk storage drive contains only the mechanical and electrical components that are required to house, load, air-filter, and drive the 3348 Data Module.

The 3348 Data Module is shown in Figure 50.15.2. The access mechanism in a 3348 Data Module is an L-shaped carriage which moves back and forth on a cylindrical shaft mounted within the data module. When the data module is not loaded, the access mechanism is latched in the home position so that it cannot move. In this position, the access mechanism is located such that the read/write heads rest on nondata areas on the disk surfaces.

Three models of the 3348 Data Module, all of which are the same physical size, are available. The 3348 Model 35 has a maximum capacity (assuming full track records) of approximately 35 million bytes that are accessed by movable read/write heads. The 3348 Model 70 has a maximum capacity of approximately 70 million bytes that are accessed by movable read/write heads. The 3348 Model 70F also has a maximum capacity of 70 million bytes of which approximately 502,000 bytes maximum (60 logical tracks) are accessed by fixed read/write heads and the balance by movable read/write heads.

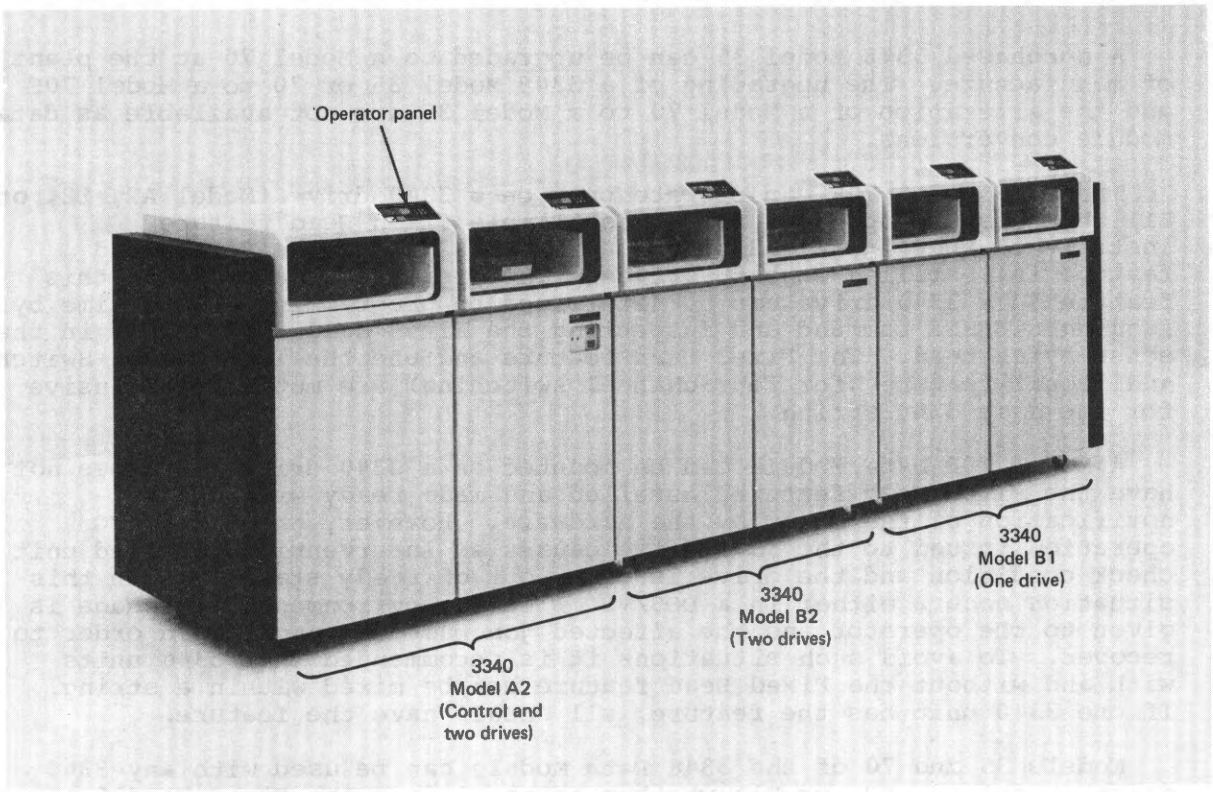


Figure 50.15.1. A five-drive 3340 string with 3340 Model A2, B2, and B1 units

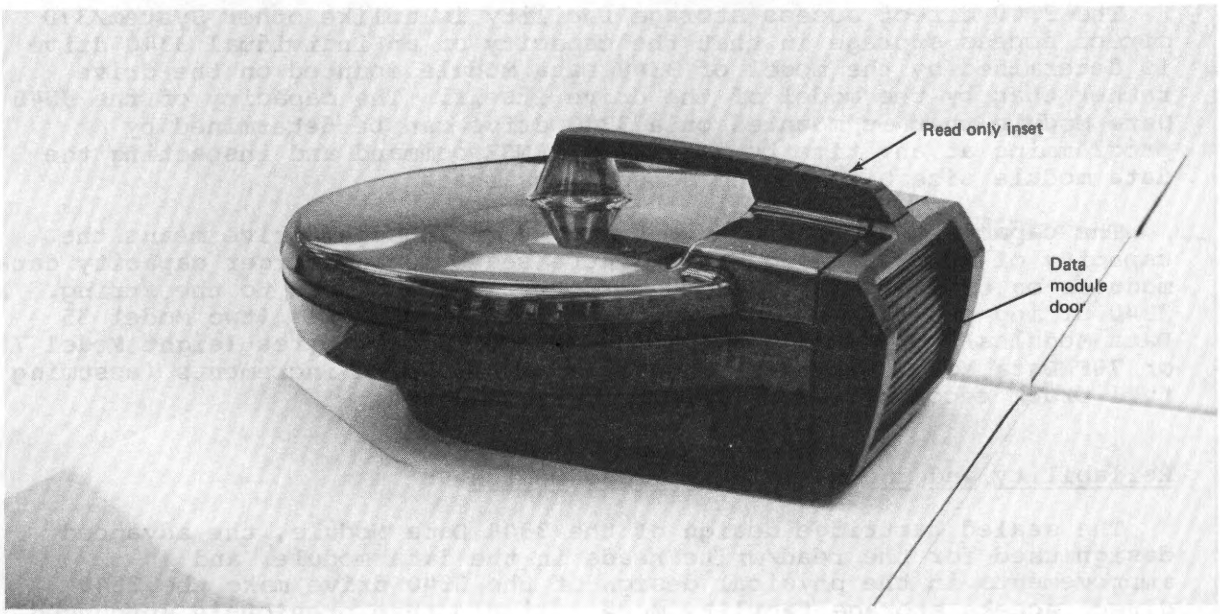


Figure 50.15.2. The 3348 Data Module

A purchased 3348 Model 35 can be upgraded to a Model 70 at the plant of manufacture. The upgrading of a 3348 Model 35 or 70 to a Model 70F and the alteration of a Model 70 to a Model 35 are not available as data module conversions.

The 3348 Model 70F can operate only on a 3340 drive (Model A2, B2, or B1) that has the optional field-installable Fixed Head feature installed. When installed on a 3340 A2 or B2 unit, the Fixed Head feature is installed on both drives. The presence or absence of this feature in a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the Fixed Head feature bit in the sense bytes read. The Fixed Head feature and the the Two-Channel Switch Additional feature (for four-channel switching) are mutually exclusive for the same 3340 string.

A Model 70F Data Module can be mounted on a 3340 drive that does not have the Fixed Head feature installed and made ready without any notification of the error by the hardware. However, the first I/O operation issued to the 3340 drive causes an intervention-required unit check condition and the drive is taken out of ready status. When this situation occurs either in a DOS/VS or OS/VS environment, a message is given to the operator and the affected job must be canceled in order to recover. To avoid such situations it is recommended that 3340 units with and without the Fixed Head feature not be mixed within a string. If one 3340 unit has the feature, all should have the feature.

Models 35 and 70 of the 3348 Data Module can be used with any 3340 drive (Model A2, B2, or B1) whether or not it has the Fixed Head feature installed. No indication is given if a Model 35 or 70 is placed in a 3340 drive with the Fixed Head feature. In such cases, the fixed head capability of the drive is not utilized.

The 3340 direct access storage facility is unlike other System/370 direct access storage in that the capacity of an individual 3340 drive is determined by the model of 3348 Data Module mounted on the drive rather than by the model of the drive itself. The capacity of the 3348 Data Module that is mounted on a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the data module size bits in the sense bytes read.

The capability of having two capacity options per drive means the capacity of a 3340 string can be increased by using larger capacity data modules on existing drives as well as by adding drives to the string. A 3340 string can vary in capacity from 70 million bytes (two Model 35 Data Modules) to a maximum capacity of 560 million bytes (eight Model 70 or 70F Data Modules) in 35- and/or 70-million-byte increments (assuming full track records).

Reliability and the Sealed Cartridge Design

The sealed cartridge design of the 3348 Data Module, the advanced design used for the read/write heads in the data module, and improvements in the physical design of the 3340 drive make the 3340 direct access storage facility more reliable than previously announced direct access storage devices for System/370, as explained below. No preventive maintenance is scheduled for a 3340 facility because of its reliability features.

Reliability is improved by the removal of head-to-disk alignment problems. Each read/write head within a 3348 Data Module is dedicated to certain tracks on one data surface. Therefore, each head reads only the data it wrote previously, regardless of the 3340 drive that is used. Since common head alignment across all 3340 drives is not required, the critical alignment tolerances that are normally necessary to achieve

data interchangeability among drives are not needed for 3348 Data Modules. It is the less critical alignment tolerances for the read/write heads in a 3348 Data Module that minimize the chance of errors caused by incorrect alignment of a head to its dedicated tracks.

There is also less chance of damaging read/write heads. If a data module is dropped, the only read/write heads that can be affected are those in that data module. If a disk pack is damaged, it can cause damage to the read/write heads in more than one drive if it is moved from drive to drive in an attempt to find a drive that can read the pack. The outside covers of a 3348 Data Module are made of a highly durable material that is designed to enable a data module to withstand more severe blows without damage than can a disk pack.

Reliability is improved because the exposure of the disk surfaces in a 3348 Data Module to outside contamination is greatly reduced when compared to the contamination exposure of a disk pack. A 3348 Data Module is opened only when it is mounted on a 3340 drive and only when the drive cover is closed. Contamination on disk surfaces can be a major cause of head and disk damage.

In addition, the possibility of head crashes is minimized by the improved flying characteristics of the read/write heads in a data module. The low mass of the read/write heads and the low loading force used enable the heads to fly over the rotating disks at a very low height. This near contact (or proximity) recording capability of the read/write heads in the 3348 permits smaller bits to be written, which increases the recording density that can be achieved.

The recording density in bits per inch of a track in a 3348 Data Module is approximately 2.5 times greater than the recording density of a track in a 2316 pack (10 percent greater than 3330-series Model 11 density and more than two times greater than 3330-series Model 1 and 2 density). The advanced head design used for the 3348 Data Module enables greater density to be achieved together with improved reliability.

Reliability of the 3340 direct access storage facility is also improved because many critical mechanical parts have been eliminated, such as a complex head load/unload mechanism. In other cases, electronic functions have replaced mechanical functions. While the 3340 drive contains more electronics than the 2314, higher density logic cards are used in the 3340, which results in significantly fewer logic cards. (A 3340 drive also contains approximately one-third the number of logic cards as a 3330-series drive.)

The sealed cartridge design implemented in the 3348 Data Module provides several advantages in addition to improved reliability, such as simplified data module loading and unloading. Operations that are required for disk pack loading and unloading (tightening the pack on the spindle, cover removal, cover replacement, untightening the pack for removal) are not required for a 3348 Data Module. In addition, the possibility of hub wear or hub damage as a result of loading and unloading operations is eliminated for a 3348 Data Module.

After the top cover of the 3340 drive to be used is raised, the operator places the data module in the exposed drive shroud recess. After closing the cover, the operator initiates automatic loading of the module by putting the start/stop switch on the operator panel of the drive in the start position. This causes the cover of the drive to be locked, which is indicated by a light on the operator panel, and the data module to be loaded.

The following occurs during data module loading. The shroud containing the seated data module moves to the back of the 3340 drive

where the voice coil motor is located. While the data module is in motion, the data module door in the rear of the 3348 is rolled down. Electrical, mechanical, and filtered air connections between the 3348 Data Module and the 3340 drive are then made through the open data module door. The access mechanism is then unlatched and the disks are brought up to rotational speed. The access mechanism is moved to physical track 0. This entire loading process requires approximately 20 seconds. When the loading process is completed, the ready light on the operator panel is turned on to indicate the 3348 Data Module is ready for processing.

To unload a data module, the operator places the start/stop switch in the stop position. The unloading procedure consists of a reversal of the operations performed during loading. The access mechanism moves to the home position in the data module where it is latched, disk rotation is stopped, the data module is disconnected from the drive, the data module door is closed, and the data module moves to the front of the drive. The cover-locked indicator light is turned off as soon as the unloading procedure is completed. Unloading requires approximately 20 seconds. The cover of the 3340 drive can be raised as soon as the cover-locked indicator light is turned off and the 3348 Data Module can then be removed.

The possibility of contaminating the disk surfaces of a data module during loading and unloading operations is minimized because the data surfaces are exposed to the air within the closed 3340 drive through the open data module door for only slightly more than one second. Further, as soon as a seal between the 3340 drive and the 3348 Data Module has been made, the filtered air system displaces the air within the data module several times to remove any contaminants that may have entered via the open data module door.

The sealed cartridge also offers two other unique features. First, a read only function (not available for the 2314) is provided on a data module basis rather than a drive basis (as implemented for 3330-series disk storage). The read only function is enabled for a 3348 Data Module by turning an inset in the handle of the 3348 (see Figure 50.15.2) to the read only position before placing the data module in the 3340 drive. This inset causes the read only switch that is part of each 3340 drive and the read only indicator on the operator panel to be turned on when the 3348 is loaded in a 3340 drive.

When the read only function is enabled for a 3348 Data Module and an attempt is made to write on the data module, an interruption occurs and IBM-supplied programming support terminates the program that issued the write. The advantage of this approach is that once the read only inset in a 3348 Data Module is set to inhibit writing, the data module can be used with any 3340 drive at any time and the operator need not remember to turn on a read only switch on the drive.

Second, external label handling is improved. An external label can be placed on a 3348 Data Module after it is removed from the 3340 drive. Placing an external label on the top surface of a disk pack instead of on the cover, to avoid mislabeling a disk pack by placing the wrong cover on it, can be done only when the disk pack is mounted on a drive. In addition, since the outside cover is never removed from a data module, the volume identification label on the cover is legible through the front window of the cover of the 3340 drive even when the data module is loaded and being accessed.

Layout of Tracks, Cylinders, and Read/Write Heads in 3348 Data Modules

The layout of physical and logical tracks on a data surface of any model 3348 Data Module and the relative position of the read/write heads

for a data surface are shown in Figure 50.15.3. A data surface contains 700 physical tracks with a small space between the first 350 physical tracks and the second 350 physical tracks. There is also unused space after the second group of 350 physical tracks. Two logical tracks, one even numbered and one odd numbered, are written on each physical track. A logical track has a maximum capacity of 8,368 data bytes (for full track records).

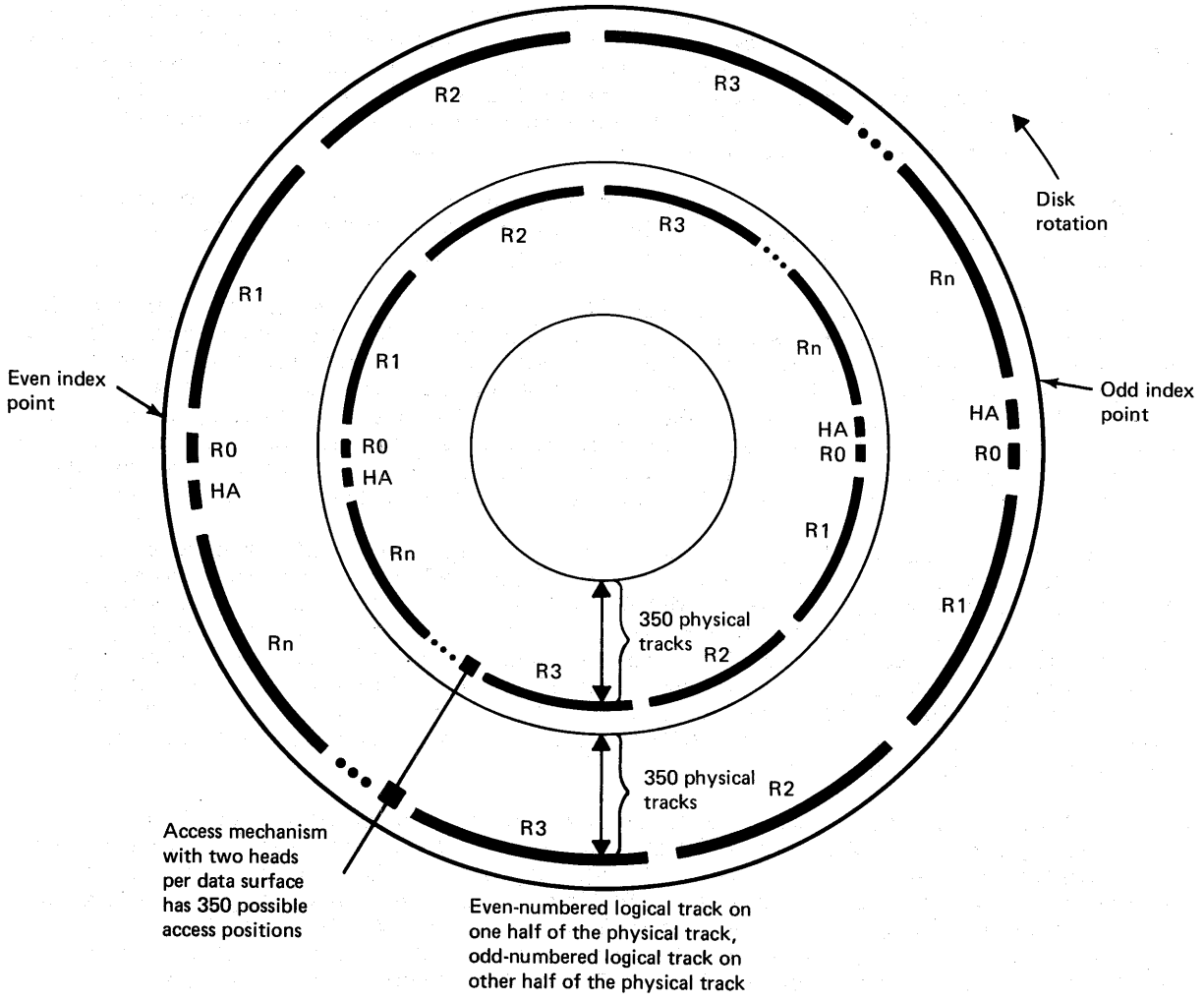


Figure 50.15.3. Location of physical and logical tracks and read/write heads on a data surface in a 3348 Data Module

There are two read/write heads associated with each data surface. They are positioned a little more than 350 physical tracks apart, as shown in Figure 50.15.3. While starting and stopping the data module, the read/write heads are positioned over the unused portions of the data surface.

The access mechanism can be placed at any one of 350 access positions on the data surface. Therefore, an outermost head on the access mechanism can access physical tracks 0 to 349 on its associated data surface while an innermost head can access physical tracks 350 to 699. At any of the 350 possible access mechanism positions, two physical tracks (4 logical tracks) can be accessed on a data surface. However, only one read/write head in a data module can be active at a time.

The bottommost surface in all 3348 Data Modules is used as the servo surface. This surface contains information for the servo system that is used to control seek operations, positioning of the heads over tracks, data clocking (the synchronization of data with rotational speed during writing operations), index generation, and signal generation required by the RPS feature. Functionally, the 3340 servo system is like that used in 3330-series drives. However, design improvements, such as elimination of the electromechanical tachometer, have been made.

The required servo information is prerecorded on the servo surface of each 3348 Data Module at the plant of manufacture and is read by a servo read head at the bottom of the access mechanism. The servo information on this surface cannot be read or written using 3340 commands. The servo surface on a 3348 Model 70F Data Module also contains the 60 logical tracks that are read by the fixed heads.

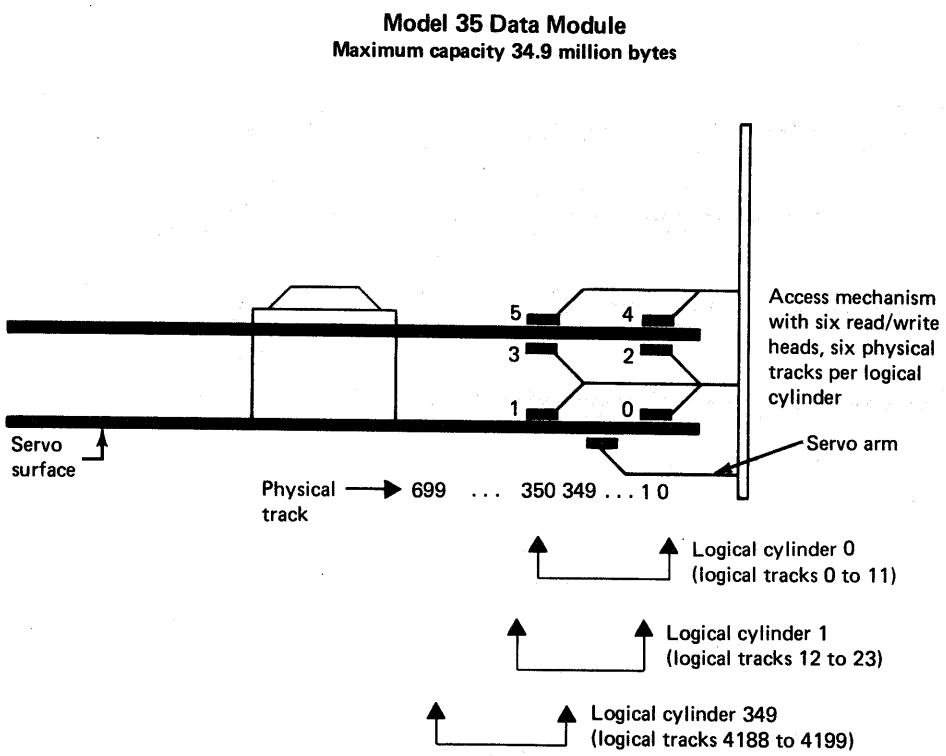
The access mechanism in a 3348 is driven by a voice-coil motor. This motor and the servo system provide fast, precise access mechanism positioning, which minimizes head settling time.

Figure 50.15.4 shows the layout of cylinders and read/write heads for the 3348 Model 35 Data Module. A Model 35 contains two recording disks. Three of the data surfaces on the two recording disks are used for data recording in a Model 35 Data Module. The three data surfaces are accessed by six read/write heads (0 to 5). The six physical tracks that can be accessed at any given position of the access mechanism constitute a logical cylinder and contain twelve logical tracks. Head 0 accesses logical tracks 0 and 1, head 1 accesses logical tracks 2 and 3, etc.

A four-byte field (CCHH) is used to address the logical tracks in a 3348 Data Module. The two-byte CC (cylinder address) field specifies the logical cylinder address, which can be 0 to 348 for the primary and alternate logical tracks of a Model 35 Data Module. The two-byte HH field, which normally specifies the actual head address (for 2314 and 3330-series drives, for example), specifies the number of the logical track within the logical cylinder, a value from 0 to 11, instead of a head address of 0 to 5. The drive selects the appropriate head using the logical track number.

In Figure 50.15.4, the access mechanism is shown positioned at logical cylinder 0 where physical tracks 0 and 350 on each of the three data surfaces can be accessed. There are 350 logical cylinders in the Model 35 Data Module. The first 348 are used for data, logical cylinder 348 is the alternate cylinder, and logical cylinder 349 is the CE cylinder. The CE cylinder is designed to be used only by the CE for testing the read/write capability of a 3340 drive. It contains a prewritten area for read testing and an area in which write tests can be performed.

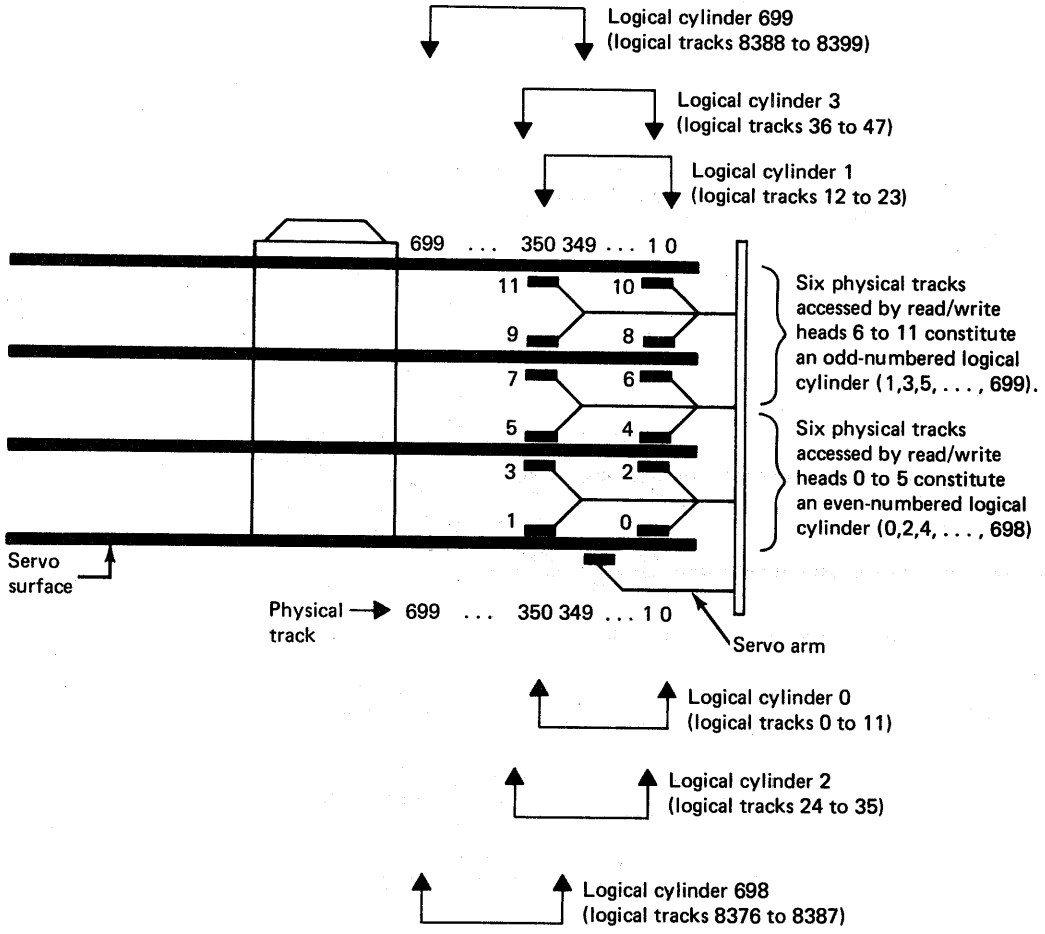
Figure 50.15.5 shows the layout of cylinders and read/write heads for the 3348 Model 70. A Model 70 contains four recording disks. Six data surfaces on the four recording disks, each of which is accessible by two read/write heads, are used for data recording in the Model 70. As for the Model 35, the six physical tracks that can be accessed by the lower six read/write heads (0 to 5) at a given position of the access mechanism constitute a logical cylinder of twelve logical tracks. In a Model 70, however, the logical cylinders addressed by read/write heads 0 to 5 are all even numbered (0, 2, 4, ..., 698). The six physical tracks that can be accessed by the upper six read/write heads (6 to 11) at a given position of the access mechanism also constitute a logical cylinder of twelve logical tracks. The logical cylinders addressed by read/write heads 6 to 11 are all odd numbered (1, 3, 5, ..., 699). Thus, on a Model 70 two logical cylinders (24 logical tracks) can be accessed at each of the 350 possible access mechanism positions.



Number of recording disks	2	
Number of data surfaces	3	
Number of read/write heads	6	
Number of physical tracks per physical cylinder	6	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	350	
Number of logical tracks per data module	4200	(4176 data) (12 alternate) (12 CE)
Number of access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	1	

Figure 50.15.4. Cylinder and read/write head layout for a 3348 Model 35 Data Module

Model 70 Data Module
Maximum capacity 69.8 million bytes



Number of recording disks	4	
Number of data surfaces	6	
Number of read/write heads	12	
Number of physical tracks per physical cylinder	12	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	700	
Number of logical tracks per data module	8400	(8352 data) (24 alternate) (24 CE)
Number of access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	2	

Figure 50.15.5. Cylinder and read/write head layout for a 3348 Model 70 Data Module

There are 700 logical cylinders in the Model 70 Data Module. The first 696 (0-695) are used for data. Logical cylinders 696 and 697 are used as alternate logical cylinders while logical cylinders 698 and 699 are CE cylinders. The method of addressing a logical track in a Model 70 Data Module is the same as described for a Model 35. The CC value can vary from 0 to 697 for data and alternate logical cylinders while the HH value can vary from 0 to 11.

Figure 50.15.6 shows the layout of cylinders and read/write heads for the 3348 Model 70F. This model is identical to the Model 70 except for the following. Seven surfaces, six data surfaces and the servo surface, on the four recording disks are used for data recording. Logical cylinders 1 to 5 are recorded on the servo surface. They are written on 30 physical tracks that are accessed by 30 fixed read/write elements, which are mounted on a plate under the servo surface, as shown in Figure 50.15.6. The first six physical tracks contain logical cylinder 1, the second six physical tracks contain logical cylinder 2, etc. Logical cylinders 0 and 6 to 699 are recorded on the six data surfaces just as in a Model 70 Data Module.

Addressing a logical track in a Model 70F Data Module using a CCHH field is the same as described for the Model 70. When a command is received that addresses a logical track in logical cylinders 1 to 5 of a Model 70F, the 3340 drive automatically selects the fixed read/write element associated with the specified logical track instead of the movable head. Therefore, a Model 70F and a Model 70 data module can be accessed using the same 3340 channel programs. This means no special programming support is required to use a Model 70F instead of a Model 70.

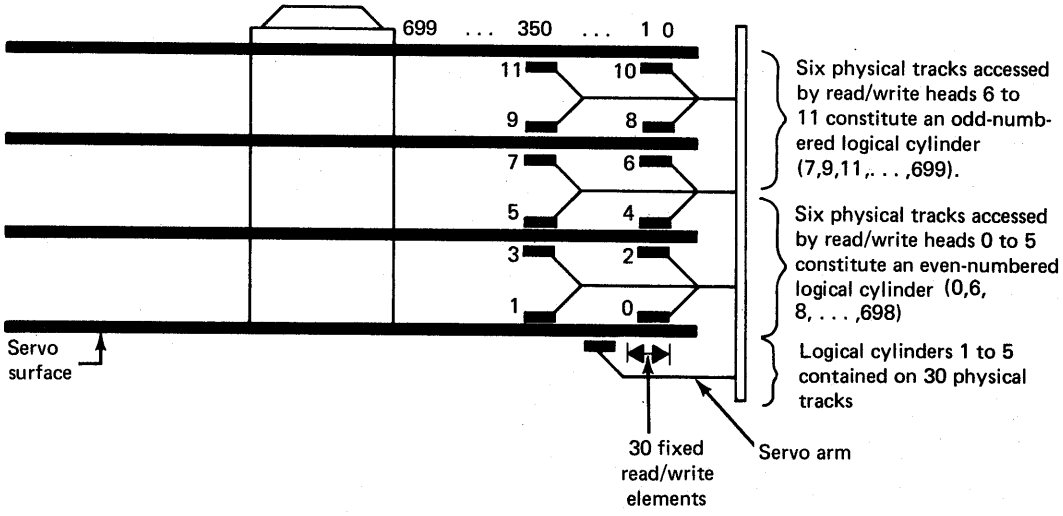
The physical tracks that contain logical cylinders 1 to 5 in a Model 70 are not used in a Model 70F and cannot be accessed by the user or a customer engineer because of the way in which head selection is performed. Hence, the data capacity of Models 70F and 70 is the same. Seek time for logical cylinders 1 to 5 in a Model 70F is zero. Seek times for logical cylinders 0 and 6 to 695 in a Model 70F are the same as Model 70 seek times.

A data set or file can be contained both in logical cylinders 1 to 5 of a Model 70F Data Module and logical cylinders that are accessed by movable heads. A 3340 drive, however, can perform only one operation at a time. Therefore, a seek, search, or data transfer operation involving a fixed head in a Model 70F Data Module cannot be performed at the same time a movable head is involved in a seek, search, or data transfer operation.

The best performance gains can be achieved when Model 70F Data Modules are used by assigning the fixed head logical tracks to small active system data sets/files (such as the page data set, system catalog, TCAM message queue), small active user data sets/files, large active data sets/files that can be segmented (OS/VS1 page data set, partitioned data sets, ISAM index levels, for example), and data sets/files with major activity concentrated at the beginning of the data set/file (such as the OS/VS job queue).

The assignment of such data sets/files to the fixed head logical tracks in a Model 70F Data Module is a user responsibility. DOS/VS EXTENT and OS/VS DD statements for these files and data sets must specifically request by actual address locations within the fixed head logical cylinders. Note also that the device type code in the device table that is generated in the control program during a system generation (DOS/VS PUB table, OS/VS UCB table) does not differentiate between 3340 drives with and without the Fixed Head feature. Therefore, if generic device type assignment by device type (3340) is used in a configuration that contains 3340 drives with and without the Fixed Head feature, either type drive can be selected by the operating system.

Model 70F Data Module
Maximum capacity 69.8 million bytes



Number of recording disks	4	
Number of data surfaces	6	plus servo surface
Number of read/write heads	12	movable
	30	fixed
Number of physical tracks per physical cylinder	12	
Number of physical tracks per logical cylinder	6	
Number of logical tracks per logical cylinder	12	
Number of logical cylinders per data module	700	
Number of logical tracks per data module	8400	(8352 data - 60 fixed head and 8292 movable head) (24 alternate) (24 CE)
Number of movable head access mechanism positions	350	
Number of logical cylinders accessed per access mechanism position	2	except for first 3 positions

Figure 50.15.6. Cylinder and read/write head layout for a 3348 Model 70F Data Module

The assignment of a 3340 drive with the Fixed Head feature can be assured in an OS/VS environment by specifying a user-defined device class name for such 3340 drives at system generation and using this name (instead of UNIT=3340) in the appropriate DD statements. DOS/VS users utilizing the generic I/O device assignment capability can specify an address list of the 3340 drives with the Fixed Head feature in the ASSGN statements for files that are to be located on a Model 70F Data Module.

Alternate tracks that are accessed by fixed heads are not provided for logical cylinders 1 to 5 in a Model 70F Data Module. Logical cylinders 696 and 697, which provide alternate tracks for the logical tracks accessed by the movable heads, also provide alternate tracks for the logical tracks in logical cylinders 1 to 5. This approach is taken because the probability a fixed head track in logical cylinders 1 to 5 will develop a defect is lower than that for movable head tracks and the possibility of a defect occurring in a movable head track is very low (for the reasons discussed later).

The low probability of defects occurring in fixed head logical cylinders 1 to 5 of a Model 70F Data Module results in part from the fact that these cylinders are recorded on the servo surface, which is a specially manufactured surface because of its primary function. In addition, the fixed head tracks are recorded on the outer edge of the servo surface, which results in a lower bit density for these tracks. The width of a fixed head physical track is six times greater than that of a movable head track on a data surface.

If an uncorrectable error does occur on a fixed head logical track in a Model 70F Data Module, the logical track should be flagged and an alternate track should be assigned. This can be done using the IEHATLAS, IEHDASDR, or IBCDASDI utility of OS/VS. IEHDASDR or IBCDASDI should then be used to test the flagged fixed head track to determine whether the track is really defective. If the track is found not to be defective, the flag is removed and the assigned alternate track is released. If the track is defective, the data module can be returned to the plant of manufacture for repair if the loss of performance resulting from using an alternate movable head track instead of the fixed head track is not acceptable.

Note that the defective track testing capability of OS/VS IEHDASDR and IBCDASDI is not provided by any DOS/VS utility. DOS/VS users can obtain IBCDASDI, the standalone utility, by ordering the OS/VS1 system.

The physical and capacity characteristics of 3348 Data Modules and the 2316 disk pack are given in Table 50.15.1. Table 50.15.2 gives the timing characteristics of the 3340 direct access storage facility and the 2314 facility.

Track Formatting and Data Module Initialization

Self-formatting records consisting of count, key, and data or count and data areas are written on the logical tracks of a 3348 Data Module just as on the tracks of a 2316 pack. However, each home address, count, and key area written on a 3348 track has a six-byte detection code field appended to it for data validity checking by the 3830 Model 2 or integrated storage control. The detection code used can detect all single-error bursts of eleven bits span or less.

Table 50.15.1. Physical and capacity characteristics of 3348 Data Modules and the 2316 Disk Pack

Characteristic	3348 Model 35	3348 Model 70	3348 Model 70F	2316
Number of data disks per data module/pack	2	4	4	11
Disk diameter in inches	14	14	14	14
Number of surfaces used per data module/pack	3 data 1 servo	6 data 1 servo	6 data 1 servo and data	20 data
Number of read/write heads per recording surface	2	2	2 plus 30 read/write elements for the servo surface	1
Number of cylinders per data module/pack	348 plus 1 alternate and 1 CE	696 plus 2 alternates and 2 CE	696 plus 2 alternates and 2 CE	200 plus 3 alternates
Number of logical tracks per cylinder	12	12	12	20
Number of data tracks recorded per data module/pack	4,176	8,352	8,352	4,000
Full track capacity in bytes	8,368	8,368	8,368	7,294
Cylinder capacity in bytes	100,416	100,416	100,416	145,880
Maximum capacity in bytes per data module/pack	34,947,768	69,889,536	69,889,536 (502,080 in logical cylinders 1 to 5, 69,387,456 in logical cylinders 0 and 6 to 695)	29,176,000
Data module/pack weight in pounds	17	19.5	20	15

Table 50.15.2. Timing characteristics of the 3340 direct access storage facility and the 2314 facility

Characteristic	Models 35 and 70	Model 70F Cylinders 1-5	Model 70F Cylinders 0, 6-699	2314
Seek time (ms)				
Maximum	50 (350 cyl-Model 35) (700 cyl-Model 70)	0	50 (700 cylinders)	130
Average	25 (350 cyl-Model 35) (700 cyl-Model 70)	0	25 (700 cylinders)	60
Cylinder to cylinder				
Model 35	10			25
Models 70, 70F	Even to next odd - 0	0	0	
	Even to next even - 10	0	10	
	Odd to next even or odd - 10	0	10	
Rotation time (ms)	20.2	20.2	20.2	25
Rotation speed (rpm)	2964	2964	2964	2400
Data transfer rate (KB/sec)	885	885	885	312
Sectors per track	64	64	64	--
Sector time (microseconds)	316	316	316	--
Load time (secs) (time to ready status after mounting)	20	20	20	60
Unload time (secs)	20	20	20	15

A six-byte correction code field is appended to each data area written on a 3348 track. The correction code used has the same detection capability as the detection code and the capability of correcting single-error bursts of three bits span or less. The actual error correction procedure must be performed by programming (error recovery routines) using corrective bits that are supplied by the control unit as discussed later.

The home address and count areas written on a logical track in a 3348 contain two new fields in addition to the same fields as are written in home address and count areas on 2316 tracks. The home address and each count area on a 3348 logical track contain a two-byte skip defect field and a two-byte physical address field in front of the flag byte. The automatic surface defect skipping capability of the 3340 allows valid data to be written before and after a surface defect on a logical track.

The skip defect bytes are used to indicate the location of the center of the surface defect relative to the index point of the logical track. Bits in the flag byte field indicate whether the surface defect is located in the next count, key, or data area.

Surface defect skipping is implemented by including in each logical track of a 3348 Data Module a reserved area called a surface defect gap in which no data is written. If a logical track has no surface defects, the surface defect gap is located at the end of the logical track. If there is a surface defect, the surface defect gap is placed over the defective portion of the logical track at the time of manufacture. One or more surface defects that together occupy an area of up to 16 bytes in length per logical track can be handled by the defect skipping technique while the stated full logical track capacity of 8,368 bytes is maintained.

The error detection and correction code capabilities of the 3340 facility permit successful recovery from an error within the data portion of a physical record even when it contains a surface defect gap.

Partial initialization of all 3348 Data Modules is performed at the plant of manufacture. A home address record and track descriptor (R0) record are written on each logical track in the data module. If a single skippable defect is found during the analysis of the surface of a logical track, the appropriate SD bytes and flag byte are written in the home address to indicate this fact. If no surface defect is found, the SD bytes are written as zeros.

The SD bytes and flag byte are supplied in the count area field in virtual storage only for a WRITE HOME ADDRESS command. When R0 is written during data module initialization and thereafter whenever a formatting write is performed, the SD and flag bytes for the count area to be written on disk are supplied by the control unit, which reads them from the record immediately preceding the record to be written.

When a record is written with a formatting write command on the portion of a logical track that contains an identified surface defect, the defect gap area is maintained in the defective portion of the logical track and data is written before and after the defect gap as appropriate. Whenever a nonformatting write or a read is issued for this record, the surface defect gap is automatically skipped over by the hardware without programming assistance or any error notification, just as if no surface defect existed.

The DOS/VS Initialize Disk or OS/VS IBCDASDI, IEHDASDR, or IEHATLAS utilities can be used to assign an alternate track if a physical track becomes defective during its use in an installation. If data cannot be read from a 3348 Data Module and recovery of this data is critical, the data module can be returned to the plant of manufacture where recovery will be attempted.

The two physical address bytes in home address and count areas on a 3348 logical track contain the physical cylinder and track address of the logical track on which they are written. When a seek command is issued, the control unit converts the logical cylinder and track address specified by the seek command to a physical cylinder and track address that is actually used by the drive in the seek operation. This physical address is saved in the control unit for later use in seek verification.

The physical address bytes are automatically written and read by the control unit and are not processed by programming. That is, when a home address or count area is written, the physical address bytes are automatically supplied by the control unit and are not contained in the home address or count area field in virtual storage that is indicated by the write command. Similarly, when a home address or count area is

read, the control unit reads the physical address bytes but they are not placed in the home address or count field area in virtual storage.

The physical address bytes are used by the control unit for seek verification during normal operations and by the 3340 microdiagnostic routines. When a home address or count area is processed during a read, search, or clock operation, the physical address bytes read are compared with the most recent seek address (physical cylinder and track address) that was saved in the control unit when the last seek command was issued. If the two physical addresses are not equal, the command is terminated and a unit check condition results. Seek check is indicated in the sense bytes.

ATTACHMENT VIA 3830 STORAGE CONTROL MODEL 2

The 3830 Storage Control Model 2 unit contains the control functions required to operate one or two 3340 strings of from two to eight drives each. If the 32 Drive Expansion and Control Store Extension optional features are installed on a 3830 Model 2, up to four 3340 strings of from two to eight drives each can be attached to it. These two features are field-installable.

Cabling between the 3830 Model 2 and the 3340 Model A2 can be a maximum of 150 feet in length. The 3830 Model 2 attaches to an integrated block multiplexer channel in the Model 158 via cabling up to 150 feet in length. Figure 50.15.7 shows a Model 158 configuration with 3340 strings attached via 3830 Storage Control Model 2. Intermixing 3340 and 3330-series strings on an attachment is discussed later in this subsection.

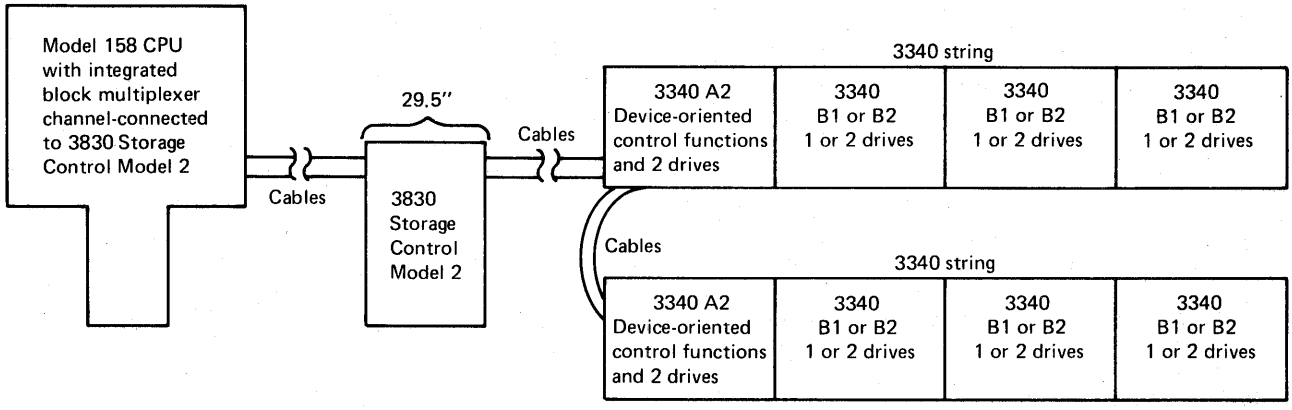


Figure 50.15.7. A Model 158 configuration with 3340 disk storage attached via 3830 Storage Control Model 2

Standard features of the 3830 Model 2 when used with 3340 disk storage are record overflow, multiple requesting, and rotational position sensing. The command retry facility of the 3830 Model 2 that is implemented for 3330-series drives is not implemented for 3340 drives. When multiple requesting is used, the 3830 Model 2 can control concurrent operation of up to 32 channel programs (when 32 Drive Expansion is installed), one on each of its drives. Only one of the two to 32 drives attached to a 3830 Model 2 can be transferring data at a time.

Rotational position sensing is an optional field-installable feature for 3340 units. It must be installed on each unit (both drives in an A2 or B2 3340 unit) that is to use the standard rotational position sensing capability of the 3830 Model 2. For performance reasons (see Section 60:10 in A Guide to the IBM System/370 Model 155, GC20-1729), it is recommended that the RPS feature be installed on all of the 3340 units in a given string or on none of the units in the string. The presence

or absence of the RPS feature in a 3340 drive can be determined by programming at any time by issuing a SENSE command and inspecting the RPS feature bit in the sense bytes read.

If a SET SECTOR command is issued to a 3340 drive that does not have the RPS feature installed, no operation is performed, track orientation is lost, and channel end and device end status are presented. If a READ SECTOR command is issued to a 3340 drive without RPS installed, a sector value of zero is returned together with channel end and device end status. Thus, channel programs containing sector commands can operate on 3340 drives that do not have RPS installed.

The 3830 Model 2 supports all the 2314 commands (except the file scan commands) in addition to new commands not available for the 2314, such as RPS and diagnostic commands. The command set for the 3340 is the same as that for 3330-series disk storage.

The Two-Channel Switch feature, identical in function to the same feature for the 2314 facility, can be installed on a 3830 Model 2 to allow it to be attached to two channels. The Two-Channel Switch Additional feature can be added to this configuration to permit the 3830 Model 2 to be attached to four channels. A maximum of two of the four channels can be present in the same system. The channels to which a 3830 Model 2 with one or both of these features is connected each must have one control unit position and, if block multiplexing is to be used, eight nonshared subchannels available. An enable/disable switch on the 3830 Model 2 can be set to dedicate the 3830 to any subset of the two to four channels.

The optional String Switch feature can be installed on 3340 Model A2 drives. This field-installable feature enables the 3340 Model A2 and its attached Model B2 and B1 units to be connected to two control unit type attachments instead of only one. The attachments can be any two of the following:

- 3830 Storage Control Model 2
- Integrated Storage Control for the Model 145
- 3345 Storage and Control Frame Models 3, 4, and 5 for the Model 145
- Integrated Storage Controls for Models 158 and 168 (or the two logical controls in one ISC)
- 3330/3340 Series IFA for the Model 135
- Direct Disk Attachment (DDA) of a Model 115 or 125 Model 2

Except for the DDA, the two attachments to which a 3340 Model A2 with the String Switch feature is attached can be connected to the same or different channels in the same CPU, or the channels in two different CPUs. In addition, except for the DDA channel switching features can be installed on one or both of these attachments. For a Model 115 Model 2 or Model 125 Model 2, the String Switch enables two 3340 strings to be attached to any System/370 model except a Model 115 Model 0 or Model 125 Model 0 and a Model 115 Model 2 or Model 125 Model 2.

The String Switch feature for 3340 disk storage is functionally similar in its operation to the Two-Channel Switch. A switch on the 3340 Model A2 can be set to allow the 3340 string to be accessed via both attachments, one at a time. In effect, this setting provides two control unit paths to the string. Switching is accomplished dynamically under program control. Alternatively, the switch can be set to dedicate the string to one attachment or the other so that the string can be accessed only via that attachment.

Figure 50.15.8 illustrates string switching for two 3340 strings attached to a 3830 Model 2 unit. In the configuration shown, both strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3340 direct access storage facilities and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

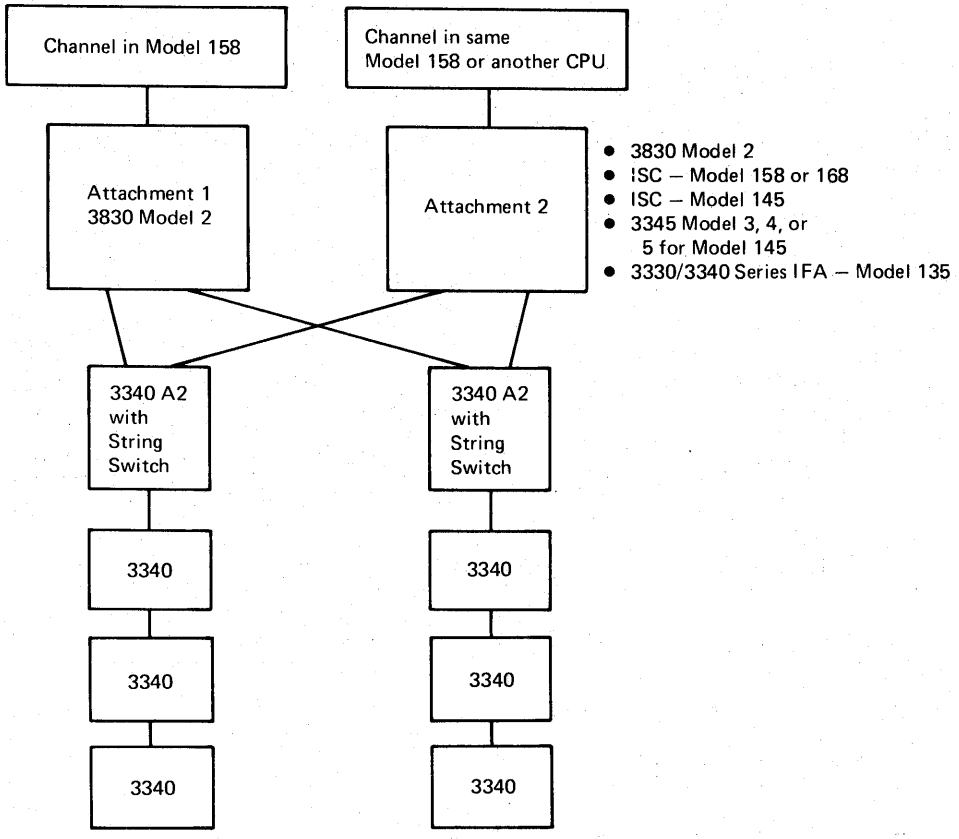


Figure 50.15.8. String switching for 3340 facilities attached to a 3830 Model 2

The 3830 Model 2 control unit is microprogram-controlled. Read/write monolithic storage contained in the control unit is used for microprogram residence. The 3830 Model 2 also contains a device that reads interchangeable disk cartridges. This device is used for microprogram backup storage and for storage of nonresident diagnostics for the 3340 string. During a 3830 Model 2 power-on sequence, the functional microprogram is loaded from the device into control storage within the 3830 Model 2 control unit. Therefore, microcode engineering changes can be installed merely by replacing the current disk cartridge with another that contains the new microprogram.

The 3830 Model 2 incorporates error detection, correction, and logging features that are designed to improve its availability and serviceability. For the 3340, the 3830 Model 2 provides the following facilities that are not implemented in System/360 direct access devices:

- I/O error routine correction of recoverable data errors on read operations with data supplied by the control unit in sense bytes. When the 3830 Model 2 detects a correctable data error during the reading of the data portion of a physical record, it generates the information necessary to correct the erroneous bytes. The sense bytes presented by the 3830 Model 2 contain a pattern of corrective bits and a displacement value to indicate which of the bytes

transferred to processor storage contain the errors. The disk error recovery program need only EXCLUSIVE OR (logical operation) the corrective bit pattern with the error bytes in the input area in processor storage to correct the errors.

- Statistical usage recording by the 3830 Model 2. Statistical usage counters for each drive in a 3340 string are continuously maintained by the 3830 Model 2. These counters indicate the number of bytes read/searched, number of seeks issued, and number of command and data overruns for each device. When a counter reaches its threshold or a data module is removed from a drive, the 3830 Model 2 indicates the condition via a unit check when the next I/O operation is initiated to the drive or a data module is made ready on the drive. Counter data can be obtained and counters can be reset by issuing a READ AND RESET BUFFERED LOG command.

- Inline diagnostic testing of a malfunctioning drive. (Inline diagnostics are provided only for 2314 facilities.) A 3830 Model 2 control unit can execute diagnostic tests on a malfunctioning drive while normal operations take place on the remaining drives in the string. Diagnostic tests can be loaded into a transient area of the control storage of the 3830 Model 2 and executed on the malfunctioning drive. This can be done in an online environment using OLTEP or the CE panel on the 3830 Model 2. OLTSEP can be used in a standalone environment. Inline testing allows CE diagnosis and repair of most 3340 drive failures without the necessity of taking the entire 3340 string out of the system configuration.

A 3340 drive can be placed in CE mode (offline to the system) by means of a switch that is located inside the rear door of the drive so that maintenance functions can be performed. To take the 3340 drive out of CE mode and return it to online status, the attention pushbutton must be pressed. This also causes the access mechanism to move to physical track 0.

ATTACHMENT VIA INTEGRATED STORAGE CONTROLS

Optionally, one Integrated Storage Controls feature can be installed on a Model 158 to attach 3340 and/or 3340-series disk storage to one or two block multiplexer channels. Attachment of 3340 and 3330-series disk storage via 3830 Storage Control is possible as well. The following discusses attachment of 3340-series strings only.

The Integrated Storage Controls feature includes dual direct access storage controls, each of which operates independently of the other and is functionally like 3830 Storage Control Model 2 except for the following:

- The Integrated Storage Controls feature is contained in the main frame of the Model 158 and is powered by the Model 158 CPU.

- The Two-Channel Switch, Additional feature (that provides four-channel switching) cannot be attached to the logical storage controls in the ISC feature.

Both logical storage controls in the ISC feature can be attached to the same channel, two different channels in the Model 158, or a channel in the Model 158 and a channel in another System/370. Each logical storage control can have attached a maximum of four 3340 strings of up to eight drives each. The 32 Drive Expansion and Control Store Extension optional features (field installable) must be installed in the ISC in order to attach more than two strings to each logical control. Therefore, up to 64 drives (eight strings) can be attached to the Model 158 via the ISC. The first unit in each 3340 string must be a 3340 Model A2.

The 3340 drives attached to the ISC operate just as if they were attached via 3830 Storage Control Model 2. That is, when multiple requesting is used, each logical storage control within the ISC can handle up to 32 channel programs concurrently, one on each of its drives, and only one of the 32 drives can be transferring data at a time. When a malfunction occurs, diagnostics can be run on one logical storage control and its drives while normal operations take place on the other logical storage control in the ISC. Intermixing 3340 and 3330-series strings on the ISC is discussed below. Figure 50.15.9 summarizes the 3340 string configurations that are possible for a Model 158 ISC.

The ISC feature provides lower-cost attachment of 3340 disk storage than 3830 Storage Control Model 2 when two storage control units are required, and physical space is saved since the ISC is in the Model 158 CPU.

The Two-Channel Switch optional feature is also available for the ISC. When installed, this feature provides a two-channel switching capability for both of the logical storage controls. The Two-Channel Switch permits each logical storage control to be attached to two channels in the same Model 158 or to one channel in the Model 158 and one channel in another System/370. Two switches are provided that can be set to dedicate a logical storage control to one channel or the other, or to enable the storage control to be accessed by both channels.

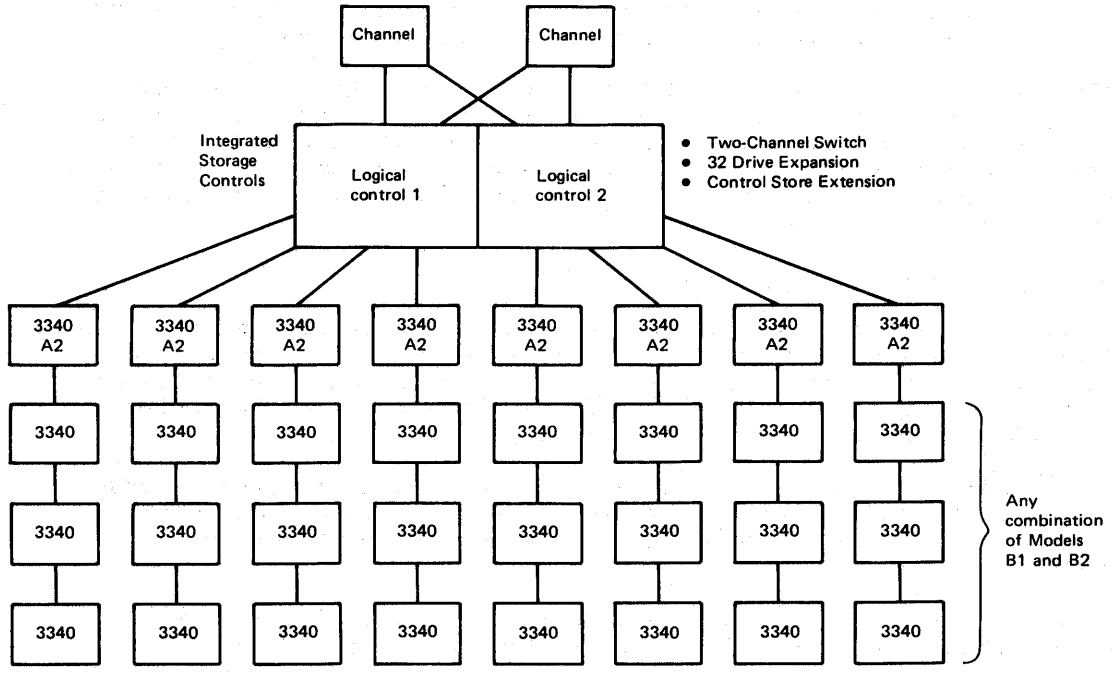


Figure 50.15.9. Permissible 3340 string configurations for the Model 158 Integrated Storage Controls feature

The String Switch optional feature can be installed on a 3340 Model A2 that is attached to the ISC. This field-installable feature enables the 3340 Model A2 and all its attached 3340s (a 3340 string) to be connected to two control unit type attachments instead of only one. The attachments can be any combination of two of the following:

- 3830 Storage Control Model 2
- Integrated Storage Controls for Models 158 and 168 (or the two logical controls in one ISC)
- Integrated Storage Control for the Model 145
- 3345 Storage and Control Frame Models 3, 4, and 5 for the Model 145
- 3330/3340 Series IFA for the Model 135

The two attachments to which a 3340 Model A2 with the String Switch feature is connected can be attached to the same or different channels in the same CPU, or to channels in two different CPU's. In addition, channel-switching features can be installed on one or both of these attachments.

The String Switch is functionally similar in its operation to the Two-Channel Switch. A switch can be set to allow the 3340 string to be accessed via both attachments, one at a time. In effect, the setting privides two control unit paths to the string. String switching is accomplished dynamically under program control. Alternatively, the switch can be set to dedicate the string to one attachment or the other so that the string can be accessed only via that attachment.

Figure 50.15.10 illustrates string switching for four 3340 strings that are attached to the same ISC. In the configuration shown, all strings can be accessed via two channels and two control units. Channel switching, string switching, and 32 Drive Expansion features can be used to enhance the availability of 3340 disk storage and to extend backup capabilities when two System/370 systems (the same or different models) are present in an installation.

INTERMIXING 3340 AND 3330-SERIES STRINGS ON AN ATTACHMENT

Optionally, the 3333/3340 Intermix feature can be installed on 3830 Storage Control Model 2 and integrated storage controls in the Model 158 CPU. When present, this field-installable feature permits both 3340 and 3330-series strings to be attached to a 3830 Model 2 or ISC. Each string must contain all 3340 drives or all 3330-series drives as usual.

The intermix feature requires installation of the Control Store Extension feature on the 3830 Model 2 or ISC and can coexist with other optional features for these units and their strings (channel switching, 32 Drive Expansion, string switching, and fixed head features).

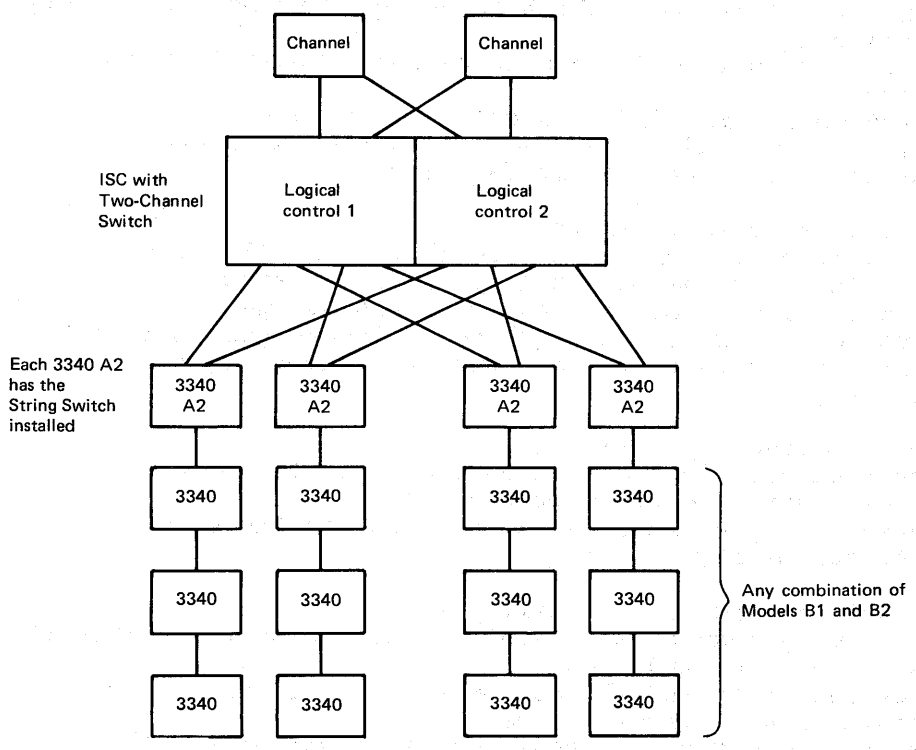


Figure 50.15.10. String switching for 3340 facilities attached to one ISC

SUMMARY

The hardware features of the 3340 and 2314 direct access storage facilities are summarized in Table 50.15.3. Table 50.15.4 compares the capabilities of the 3830 Model 1, 3830 Model 2, and Model 158 integrated storage controls for both 3340 and 3330-series disk storage.

When compared with the 2314 facility, the 3340 facility offers the following major advantages:

- Faster access to data
  - Data transfer rate almost three times that of the 2314
  - Seek times approximately 40% of those of the 2314 for movable head accesses
  - Zero seek time provided by the fixed heads in a 3348 Model 70F Data Module
  - Rotational delay interval approximately 20% shorter than for the 2314
- Larger capacity per drive
  - 17% for the Model 35 Data Module
  - 175% for Model 70 and 70F Data Modules
- Two capacity options per drive for expanded growth flexibility
- Multiple requesting and rotational position sensing capabilities for use with block multiplexer channels

- Operational improvements
  - Cover tightening/untightening and removable/replacement operations are eliminated, which speeds up data module loading and unloading
  - Load time to ready status for a mounted data module is three times faster
  - Write protection is provided on a data module basis
  - External labeling procedures are more flexible and leave less chance of erroneous data module labeling

- Significantly increased reliability
  - Sealed cartridge design eliminates head-to-disk alignment problems, minimizes the possibility of disk surface contamination, and eliminates hub wear and damage
  - Advanced head design makes head crashes a remote possibility and permits increased recording density without any loss of reliability

- Improved error handling capabilities
  - Error correction data is provided by the hardware for use by programmed error recovery procedures
  - Surface defect skipping reduces the need to use the error correction capability

- Improved availability and serviceability
  - No preventive maintenance is scheduled, because of the reliability features of the 3340 and 3348
  - Faster error isolation and correction is possible because the 3340 contains fewer circuit cards
  - Expanded microdiagnostics can test more than 95% of the circuits in a 3340

Table 50.15.3. Summary of the hardware features of 3340 and 2314 disk storage facilities

Feature	3340 attached to 3830 Model 2 or ISC	2314 (A-Series)
Number of drives per string or facility	Two to eight in one drive increments	One to eight in one-drive increments. (A ninth can be included as a spare only.)
Number of strings or facilities per control unit	One to four (maximum of eight strings for ISC)	One maximum
Data medium used	Removable interchangeable data module (sealed cartridge)	Removable interchangeable disk pack
Read only feature on drive or data medium	Yes on data module	No
Removable address plugs on drive	No	Yes
Attachment of a string or facility to two control units in the same or a different CPU	Yes via optional string switch feature. Only one data transfer operation permitted per string.	Yes via 2844 Auxiliary Storage control. Two concurrent data transfer operations per facility permitted.
Two-Channel Switch	Optional	Optional
Attachment of the control unit to four channels	Yes using the optional Two-Channel Switch and Two-Channel Switch Additional features (3830 Model 2 only)	Yes using the optional Two-Channel Switch and 2844 Auxiliary Storage Control
Record Overflow	Standard	Standard
File Scan	Not available	Standard
Multiple track operations	Standard	Standard
Multiple requesting	Standard	Not available
Rotational Position Sensing	Optional (on 3340 drives)	Not available
Error correction data presented by control unit	Yes	No
Surface defect skipping	Yes	No

Table 50.15.3 (continued)

Feature	3340 attached to 3830 Model 2 or ISC	2314 (A-Series)
Writable storage in control unit loaded from a disk cartridge	Yes	No
Statistics logging by the control unit in its storage	Yes	No
Inline diagnostics executed under OLTEP or via the CE panel	Yes	Yes

Table 50.15.4. Summary of the features of 3830 Storage Control Models 1 and 2 and Integrated Storage Controls

Characteristic	3830 Model 1	3830 Model 2	ISC
Type of unit	Standalone	Standalone	Contained in Model 158 CPU
Power source	Contains own for itself and all the drives that can be attached to it	Contains own for itself only	Power control shared with Model 158 CPU
Attaches to	Block multiplexer channel	Block multiplexer channel	Block multiplexer channel
Devices attaching to it	3330 Models 1 and 2	3333 Models 1 and 11 (optionally with 3330 Model 1, 2, and 11 units attached) 3340 Model A2 (optionally with 3340 Model B1 and B2 units attached)	Same as 3830 Model 2
Number of drives in a string	1 to 8	2 to 8 for a 3330-series or 3340 string	Same as 3830 Model 2
Standard number of strings attachable	One maximum	Two maximum	Two maximum per logical control
32 Drive Expansion feature for attachment of two additional strings	Not available	Optional for a maximum of four strings	Optional for a maximum of four strings per logical control

Table 15.15.4 (continued)

Characteristic	3830 Model 1	3830 Model 2	ISC
3333/3340 Intermix feature for attachment of 3330-series and 3340 strings	Not available	Optional	Optional
Two-Channel Switch	Optional	Optional	Optional
Two-Channel Switch Additional (for four channel switching)	Optional	Optional	Not available
String switching capability	Not available	Yes for 3330-series strings via optional 3333 String Switch feature. Yes for 3340 strings via optional String Switch Feature.	Same as 3830 Model 2
Multiple requesting	Standard	Standard	Standard
Rotational position sensing	Standard	Standard on control unit (standard on 3330-series drives, optional on 3340 drives)	Same as 3830 Model 2
Multiple track operations	Standard	Standard	Standard
Record overflow	Standard	Standard	Standard
Command retry	Standard	Standard for 3330-series strings. Not available for 3340 strings.	Same as 3830 Model 2
Surface defect skipping	Not implemented	Implemented for 3340 strings. Not implemented for 3330-series strings.	Same as 3830 Model 2
Inline diagnostic tests	Standard	Standard	Standard
Error logging by control unit	Standard	Standard	Standard

SECTION 65: DIFFERENCES BETWEEN THE MODEL 3 AND THE MODEL 1

Model 3 of the Model 158 differs from the Model 1 in its faster internal performance, enhanced operational flexibility, and improved tightly coupled multiprocessing configurability. The latter two capabilities improve the availability of Model 3 Model 158 configurations. The serviceability of the Model 3 is improved by enhancements to the remote support facility.

A Model 1 system (3158-1 Processing Unit) can be field converted to a Model 3 system (3158-3 Processing Unit). A program can determine which model of the Model 158 it is executing in by issuing the STORE CPU ID instruction and inspecting byte 0 of the stored doubleword. Both models have the same standard features except for the amount of buffer storage provided. The same optional features are available for and the same I/O devices attach to both Model 158 models.

PERFORMANCE FACILITIES

The internal performance of the Model 3 Model 158 CPU is generally in the range of 5 to 11 percent faster than that of the Model 1 CPU when the same hardware configurations, programming systems, and programs are used. The faster internal performance of the Model 3 is the result of the following differences between the Model 3 and the Model 1:

- 16K instead of 8K of high-speed monolithic buffer storage is standard in the Model 3. The 16K buffer organization and assignment algorithm result in a higher buffer hit ratio than is achieved for the Model 1. The new organization and algorithm are discussed below. A denser technology is used to implement the high-speed buffer in the Model 3 so that the 16K buffer in the Model 3 requires less space than the 8K buffer in the Model 1.

- The instruction execution time of each of the following instructions is faster in a Model 3 than in a Model 1: OR (RR and RX formats), EXCLUSIVE OR (RR and RX formats), AND (RR and RX formats), MOVE CHARACTER (MVC), EXECUTE (EX), STORE THEN OR SYSTEM MASK (STOSM), STORE THEN AND SYSTEM MASK (STNSM), LOAD REAL ADDRESS (LRA), COMPARE and SWAP (CS), and SET PSW KEY FROM ADDRESS (SPKA).

- The instruction fetch buffer in the Model 3 is 128 words in size instead of 64 words as in the Model 1. This buffer can reduce the number of accesses to buffer and/or real storage required during instruction fetching.

- The read cycle time of processor storage for successive reads of 16 bytes is 920 nanoseconds in the Model 3, instead of 1035 nanoseconds, as in the Model 1. Other cycle and access times are the same in the two models.

High-Speed Buffer

Buffer fetch times and the way in which the high-speed buffer is used are the same in the Model 3 as in the Model 1 (as described in Section 20:15 under "High-Speed Buffer Storage"). In addition, as shown in Figure 65.1, processor storage in the Model 3 (as in the Model 1) is divided in 4K-byte rows, the number of which depends on the size of processor storage. In the Model 3, however, a row is divided into 128 32-byte blocks instead of 256 16-byte halfblocks (as in the Model 1) for assignment purposes.

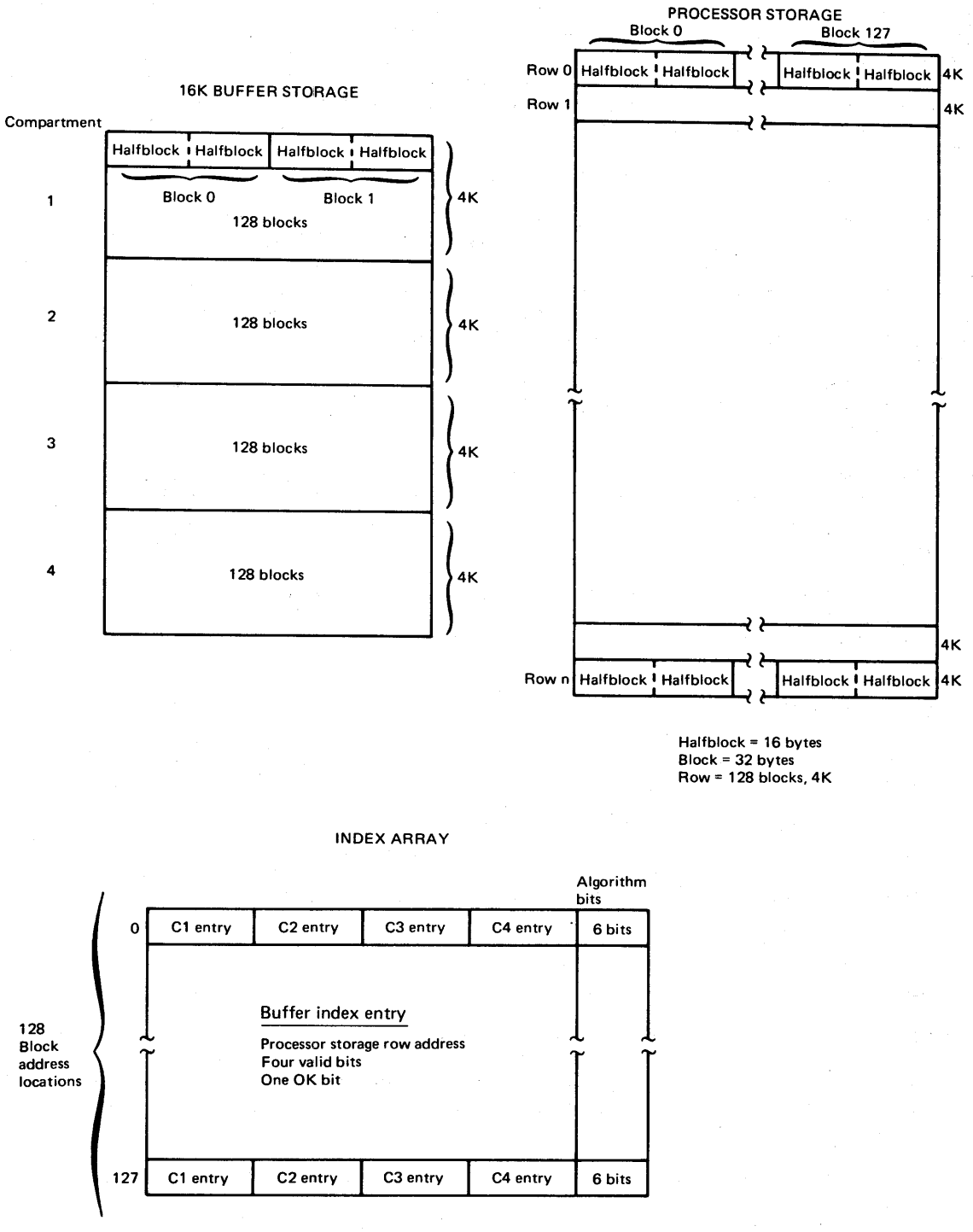


Figure 65.1. High-speed buffer organization in the Model 3

The 16K buffer in the Model 3 is divided into four 4K compartments, just as the 8K buffer in the Model 1 is divided in two 4K compartments. However, as shown in Figure 65.1, in the Model 3 buffer, a compartment is divided into 128 32-byte blocks, as is processor storage, instead of 256 16-byte halfblocks like a compartment in the Model 1 buffer. This is done because in the Model 3, buffer storage is assigned on a block basis and loaded a halfblock at a time (just as is true for the Model 155 buffer assignment algorithm). Buffer storage is assigned and loaded on a halfblock basis in the Model 1.

The index array for the Model 3 buffer is also shown in Figure 65.1. It contains 128 block address locations. Each block address location contains four buffer index entries, one to describe the contents of the corresponding block in each of the four buffer compartments. Each buffer index entry contains a processor storage row address, four valid bits, and an OK bit. When a halfblock of data from processor storage is placed in the buffer, its processor storage row address (bits 8 to 19 of its processor storage address) are placed in the appropriate buffer index entry within the corresponding block address location in the index array and the valid bit is turned on.

Also associated with each of the 128 block address entries in the Model 3 buffer index array are six algorithm bits. These bits are used to determine which of the four buffer compartments a halfblock is assigned when it is loaded into the buffer. (An LRU bit for each two halfblock buffer index entries is used for assignment purposes in the Model 1 buffer index array.) If all four of the buffer locations that the halfblock can be assigned contain valid data, the buffer location referenced longest ago (as indicated by the algorithm bits) is assigned.

OTHER DIFFERENCES

Other differences between the Model 3 and the Model 1 are the following:

- Online diagnostic testing and repair of most mechanical functions of the 3213 printer are provided for the Model 3. In a Model 1 configuration, the entire Model 158 system must be dedicated to the customer engineer during maintenance operations on the 3213.
- The data path between the CPU and the display console is wider in the Model 3 than in the Model 1 to provide faster internal performance for the display console.
- The amount of control storage provided in the service processor unit is increased by 50 percent and microprograms that are designed to optimize the internal performance of the display console, instead of reduce control storage requirements, are used in the service processor.
- A teleprocessing overrun situation that can occur in the Model 1 is eliminated in the Model 3. This is accomplished by not recording the second machine logout on the N-disk when two machine checks occur in a short period of time.
- More shared and nonshared subchannels are available for byte and block multiplexer channels and a shared subchannel can be shared by up to 32, instead of a maximum of 16, devices as discussed below.
- The remote support facility in the Model 3 provides the capability of transmitting logout data to the RETAIN/370 system concurrent with normal system operation and provides improved local to remote customer engineer communication.

Byte and Block Multiplexer Subchannels

The way in which subchannel sharing for byte multiplexer channels is established in a Model 3 provides more flexibility in determining the number of shared and nonshared subchannels than does the technique used in the Model 1. In addition, it enables up to 32 devices attached to the same control unit to share a single subchannel.

The Model 3 contains a plugcard for byte multiplexer channel 0 and another for byte multiplexer channel 4 if it is present. A plugcard has one position for each of the eight control unit positions on its associated byte multiplexer channel. Each control unit position on a plugcard can be independently wired to permit or inhibit subchannel sharing. A control unit position wired for sharing is also wired to permit up to 16 or 32 devices to share the same subchannel.

When a control unit position on a byte multiplexer channel is wired to allow subchannel sharing for up to 16 I/O devices, a contiguous set of 16 device addresses in the range of X0 to XF (where X can be 0 to F) is associated with the shared subchannel for the control unit position. When subchannel sharing for up to 32 I/O devices is specified for a control unit position, the control unit address must be even-numbered and the next higher odd-numbered control unit address cannot be used. A block of 32 contiguous I/O device addresses in the range of X0 to XF and (X+1)0 to (X+1)F (where X can be 0, 2, 4, 6, 8, A, C, or E) is associated with the shared subchannel for the control unit position.

When subchannel sharing for up to 32 devices is specified for a control unit position on a byte multiplexer channel, the following device addresses are mutually exclusive for the channel:

00 to 0F and 10
20 to 2F and 30
40 to 4F and 50
60 to 6F and 70
80 to 8F and 90
A0 to AF and B0
C0 to CF and D0
E0 to EF and F0

When subchannel sharing is specified for one or more control unit positions on a byte multiplexer channel, the number of nonshared subchannels available is 256 less 16 or 32 for each control unit position wired for subchannel sharing. If no control unit position on a byte multiplexer channel is wired for subchannel sharing, the channel has 256 nonshared subchannels for device addresses 00 to FF and each device is assigned a unique subchannel.

As in a Model 1, a pool of nonshared subchannels is available in a Model 3 that is to be used by all installed block multiplexer channels. Shared subchannels can be assigned from this pool. In a Model 3, up to 40 shared subchannels can be assigned when the second byte multiplexer channel is not installed. When the second byte multiplexer is installed, up to 32 shared subchannels can be assigned. This permits each block multiplexer channel to have up to eight shared subchannels assigned. Each shared subchannel that is utilized reduces the total number of nonshared subchannels available for the block multiplexer channels by one.

Shared subchannels for block multiplexer channels in the Model 3 are preassigned a set of addresses by the customer engineer as they are in the Model 1. However, when making assignments for the Model 3, the customer engineer also specifies whether each shared subchannel is to be associated with 16 or 32 device addresses and whether it is to operate in selector or block multiplexer mode. A shared subchannel can have 16

contiguous addresses in the range of X0 to XF associated with it or 32 contiguous addresses in the range of X0 to XF and (X+1)0 to (X+1)F.

The number of nonshared subchannels available for the block multiplexer channels in the Model 3 is 736 minus the number of shared subchannels assigned when the second byte multiplexer channel is not installed and 480 minus the number of shared subchannels assigned when the second byte mutliplexer is installed. If no shared subchannels are assigned, 736 or 480 nonshared subchannels are available for the installed block multiplexer channels. The assignment of nonshared subchannels (UCW's) to block multiplexer channels is done dynamically during system operation in the same way for a Model 3 as for a Model 1 (and a Model 155) except in the case when all nonshared subchannels have been assigned and no more UCW's are available in the pool, as follows.

In addition to the 480 or 736 nonshared UCW's in the Model 3, there is one floating UCW for each installed block multiplexer channel. When the first START I/O instruction is issued to a device without a UCW assigned, the floating UCW for the addressed channel is assigned to the device for the duration of the I/O operation if no nonshared UCW is available. This enables the device to operate in block multiplexer mode (instead of selector mode as occurs in a Model 1 in this situation). At the completion of the I/O operation, the floating UCW becomes available for reassignment. If a START I/O instruction is issued to another device that has no UCW assigned while the floating UCW for the addressed channel is in use, a channel busy condition code is returned and the I/O operation must wait until the floating UCW for the channel or another UCW from the pool becomes available.

Remote Support Facility

The remote program and remote console modes supported by RSF for the Model 1 are also supported in the Model 3. In addition, a copylog only mode is supported by RSF for the Model 3. The remote program mode and the remote control submode of remote console mode provide the same functions in both models. The remote monitor submode of the remote console mode is not supported in the Model 3. The copylog only mode provides a new capability that can be utilized for hardware failures that do not prevent the system from continued operation.

In a Model 3, the teleprocessing link is established in the same way it is accomplished for the Model 1 except that the link is automatically initiated when the remote console or copylog only code is loaded in the service processor. Once the copylog only code is selected, the local customer engineer or operator can return to the program frame and continue customer processing. The selection of the copylog only code causes the RETAIN/370 system to request the transmittal of the logout records contained on the N-disk. The establishment of a link to RETAIN/370 and the transmittal of logout records are performed concurrent with normal system operation.

After the logout records have been sent, the teleprocessing link is disconnected and the transmitted logout records are processed by the logout analysis program. The results are stored in the RETAIN/370 system for analysis by a remote customer engineer specialist.

When a hardware failure that the local customer engineer cannot locate prevents a Model 3 system from operating, the local customer engineer can establish the remote console mode of RSF and dedicate the Model 158 system to the remote specialist, as for a Model 1 system.

In a Model 3, the local customer engineer is not limited to using only the teleprocessing link frame for communication with a remote customer engineer during RSF operation. Line 25 of any frame can be

utilized for communication. The local customer engineer initiates communication by selecting any data on the currently displayed frame using the light pen. This causes the keyboard to be unlocked so that the message (up to 80 characters) can be entered. Once the enter key is pressed, the remote customer engineer is notified of the message and the local display keyboard is again locked. The remote customer engineer can then request transmission of the message.

MULTIPROCESSING

A tightly-coupled Model 158 multiprocessing configuration can include Model 3 systems. A multiprocessing configuration can consist of two Model 3 systems, two Model 1 systems, or one Model 3 and one Model 1 system. When a multiprocessing configuration consists of two Model 3 systems and each system has 1, 2, 3, or 4 megabytes of processor storage, both systems need not have the identical amount of processor storage installed. Such asymmetric processor storage configurations are not permitted when a multiprocessing configuration consists of two Model 1 systems, a Model 3/Model 1 combination, or two Model 3 systems with 512K or 1536K of processor storage.

The use of two Model 3 systems in a multiprocessing configuration also provides an availability advantage over the use of two Model 1 systems or a Model 3/Model 1 combination. When two Model 3 systems are used, the CPU, channels, and console of a system can be powered down independently from its processor storage. This is called the alternate CPU power down capability. It permits the operator to vary offline the CPU, channels, and console of one system and power them down (for certain maintenance purposes, for example) while leaving processor storage of this system available for use by the CPU in the other system. The powered down CPU, channels, and console can then be powered up again and varied online. System operations can continue without a re-IPL. In a multiprocessing configuration in which two Model 3 systems are not used, the powering down of the CPU in a system requires the powering down of its processor storage as well.

PROGRAMMING SYSTEMS SUPPORT

Both models are supported by the same IBM-supplied programming systems, that is, DOS Versions 3 and 4 (hard stop mode only), DOS/VS, OS MFT and MVT Releases 21.6, 21.7, and 21.8, OS/VS1, OS/VS2 Releases 1 and up, and VM/370. The model-dependent fixed storage locations are the same in the Model 3 as in the Model 1 (see Figure 20.10.3) except for a few differences in the CPU extended logout area. The EREP program in OS MFT and MVT Releases 21.7 and 21.8, in OS/VS1 Releases 1 and up, in OS/VS2 Releases 1 and up, and in VM/370 Release 2 will be modified to process the model-dependent logout area data for the Model 3 that differs from that of the Model 1. Although the EREP programs in DOS Versions 3 and 4, DOS/VS, and OS MFT and MVT Release 21.6 will not be modified, they can still be used on a Model 158 Model 3.

SECTION 70: COMPARISON TABLES

These tables have been included for quick reference. The first compares hardware features of System/360 Models 50 and 65 and System/370 Models 145, 155, 155 II, and 158 (Models 1 and 3). The second compares DOS Version 4 and DOS/VS support of the Model 158 (Models 1 and 3) hardware, while the third compares OS MFT and MVT, VS1, VS2 Release 1.7, and VS2 Release 3 support of the Model 158.

70:05: COMPARISON TABLE OF HARDWARE FEATURES FOR SYSTEM/360 MODELS 50 AND 65 AND SYSTEM/370 MODELS 145, 155, 155 II, AND 158 (MODELS 1 AND 3)

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
I. CPU						
A. BC mode of system operation	Comparable to BC mode	Same as Model 50	Standard	Standard	Standard	Standard
B. EC mode of system operation	Not implemented	Not implemented	Standard	Not implemented	Standard	Standard
C. Instruction set						
1. Standard set (Binary arithmetic)	Standard	Standard	Standard	Standard	Standard	Standard
2. Decimal arithmetic	Standard	Standard	Standard	Standard	Standard	Standard
3. Floating-point arithmetic	Standard	Standard	Optional	Standard	Standard	Standard
4. Extended precision floating-point	Not available	Not available	Optional (included in floating-point option)	Optional (mutually exclusive with 7070/7074 Compatibility)	Optional (not mutually exclusive with 7070/7074 Compatibility)	Same as Model 155 II except the feature is no-charge.
5. New instructions						
a. COMPARE LOGICAL CHARACTERS UNDER MASK COMPARE LOGICAL LONG HALT DEVICE INSERT CHARACTERS UNDER MASK LOAD CONTROL MONITOR CALL MOVE LONG SET CLOCK SHIFT AND ROUND DECIMAL START I/O FAST RELEASE STORE CHANNEL ID	Not available	Not available	Standard	Standard	Standard	Standard

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
STORE CHARACTERS UNDER MASK STORE CLOCK STORE CONTROL STORE CPU ID						
b. CLEAR I/O COMPARE AND SWAP COMPARE DOUBLE AND SWAP INSERT PSW KEY LOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CLOCK COMPARATOR SET CPU TIMER SET PSW KEY FROM ADDRESS STORE CLOCK COMPARATOR STORE CPU TIMER STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK	Not available	Not available	Standard except for CLEAR I/O, COMPARE AND SWAP, COMPARE DOUBLE AND SWAP, INSERT PSW KEY, and SET PSW KEY FROM ADDRESS which are optional (and no-charge)	Not available	Standard	Standard

Note: The following operations are faster on a Model 158 (Models 1 and 3) than on a Model 155 or 155 II:

- All multiply operations
- All register shifts
- Certain move character operations
- Convert to binary and convert to decimal operations

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
						In Model 3 of the Model 158, certain logical operations moves, stores, and other instructions execute faster than in the Model 158 Model 1.
D. Buffered instruction fetch	No	Instruction unit normally prepares one instruction at a time. Imprecise interruptions occur only for storage protection violations.	No	Yes Three one-word buffers are provided. (Imprecise interruptions do not occur.) One prefetched instruction decoded, no operand prefetching.	Same as Model 155	Instruction prefetching is improved via use of a 64-word buffer in the Model 1 and a 128-word buffer in the Model 3.
E. CPU cycle time	500 nanoseconds, 4-byte parallel flow	200 nanoseconds, 8-byte parallel flow	Variable from 202.5 to 315 nanoseconds, 4-byte parallel data flow	115 nanoseconds, 4-byte parallel flow	Same as Model 155	Same as Model 155
F. Dynamic address translation	Not available	Not available	Standard	Not available	Standard	Standard
G. Channel indirect data addressing	Not available	Not available	Standard	Not available	Standard	Standard
H. Interval timer	Standard (16.6 ms resolution)	Standard (16.6 ms resolution)	Standard (3.33 ms resolution)	Standard (3.33 ms resolution	Standard (3.33 ms resolution)	Standard (3.33 ms resolution)
I. Time of day clock	Not available	Not available	Standard	Standard	Standard	Standard
J. CPU timer and clock comparator	Not available	Not available	Standard	Not available	Standard	Standard

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
K. Monitoring feature	Not available	Not available	Standard	Standard	Standard	Standard
L. Program event recording	Not available	Not available	Standard	Not available	Standard	Standard
M. Direct control	Optional	Optional	Optional	Optional	Optional	Optional
N. Interruption for SSM instruction	Not implemented	Not implemented	Standard	Not implemented	Standard	Standard
O. Compatibility features (all are optional unless otherwise indicated	1. 1410/7010 2. 7070/7074 (mutually) exclusive features)	1. 7070/7074 2. 7080 (for both 705 and 7080) 3. 709/7040/7044/7090/7094/7094II	1. 1401/1440/1460 2. 1401/40/60, 1410/7010 3. OS/DOS (standard)	1. 1401/40/60, 1410/7010 Compatibility 2. OS/DOS compatibility 3. 7070/7074 Compatibility (mutually exclusive with extended precision floating point)	Same as Model 155 except 7070/7074 Compatibility is not mutually exclusive with extended precision floating point.	Same as Model 155 except features are no-charge and 7070/7074 Compatibility is not mutually exclusive with extended precision floating point.
P. Control logic	Microprogram in ROS	Microprogram in ROS	Microprogram in reloadable control storage of 32K to 64K	Microprogram in ROS	Same as Model 155	Microprogram in reloadable control storage (RCS)
Q. Instruction retry by hardware	No	No	Yes	Yes	Yes	Yes
R. Machine check interruption	Occurs on CPU, main storage and certain channel errors. One mask bit controls this interruption.	Same as Model 50	Occurs after corrected and uncorrected errors. There are five types of machine check and each is individually maskable.	Occurs after corrected and uncorrected errors. There are seven types of machine check and many are individually maskable.	Same as Model 155. Additional fields are logged.	Same as Model 155. More data is logged.

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
S. Fixed storage area size in lower storage (including logout area for machine and channel errors)	292 bytes	328 bytes including CPU and channel logouts.	704 bytes reducible to 512 if extended logout area is moved	1184 bytes reducible to 512 if the extended logout area of 672 bytes is moved	Same as Model 155	Same as Model 155
T. Multiprocessor systems	1. A multisystem feature is not available. 2. The support or main processor in an ASP configuration can be a Model 50.	1. Multisystem optional feature permits interconnection of two Model 65s. Main storage is shared (512K or more). Direct control is required. 2. The support or main processor in an ASP configuration can be a Model 65. Two or three systems are connected via a Channel-to-Channel Adapter.	1. A multisystem feature is not available. 2. The Model 145 can be included in loosely coupled multiprocessing configurations.	1. A multisystem feature is not available. 2. A Model 155 can be a support or a main processor in an ASP configuration.	Same as Model 155	1. The 3058 Multisystem Communication Unit is used to connect two Model 158 systems (any combination of Models 1 and 3) in a tightly coupled shared storage multiprocessing configuration 2. JES3 (Job Entry Subsystem 3) of OS/VS2 Release 3 supports the Model 158 in a loosely coupled multiprocessing configuration. 3. A Model 158 can be a support or main processor in an ASP configuration.
U. Power Warning	Not available	Not available	Not available	Not available	Not available	Optional
V. Virtual Machine Assist	Not available	Not available	Optional (no-charge)	Not available	Not available	Optional (no-charge)
W. Service processor unit	Not available	Not available	Not available	No - 2955 Remote Analysis Unit is optional	Same as Model 155	Standard

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
II. STORAGE						
A. Processor (main) storage sizes	64K 128K 256K 384K 512K	256K 512K 768K 1024K	160K 208K 256K 384K 512K 768K 1024K 1536K 2048K	256K 384K 512K 768K 1024K 1536K 2048K	256K 384K 512K 768K 1024K 1536K 2048K	512K 1024K 1536K 2048K 3072K 4096K
B. Type of processor storage	Ferrite cores	Ferrite cores	Monolithic technology	Ferrite cores	Ferrite cores	Monolithic technology
C. Processor storage cycle	2 microseconds for 4 bytes	750 nanoseconds (for 8 bytes). Two-way interleaving of sequential accesses other than by the channels is provided.	540 nanoseconds fetch for 4 data bytes. 607.5 nanoseconds store for 4 data bytes. 540 nanoseconds fetch for 8 instruction bytes.	2070 nanoseconds for read/write of 16 bytes	Same as Model 155	Read 16 bytes--1035 ns. Write of 8 bytes on double-word boundary--690 ns. Partial (1-16 bytes) write--920 ns.
D. High-speed buffer storage	No	No	No	8K buffer is standard	8K buffer is standard. Buffer assignment is slightly different from Models 155 and 158.	8K buffer is standard in the Model 1. 16K buffer is standard in the Model 3. Buffer assignment is slightly different from Models 155 II and 155. Buffer assignment in Models 1 and 3 differs also.
1. CPU fetch from buffer	-	-	-	230 ns for 4 bytes 345 ns for 8 bytes	Same as Model 155	Same as Model 155
E. Processor storage validity checking	Parity checking by byte. No hardware error correction is provided.	Same as Model 50	ECC checking on a double-word. Single-bit errors are corrected by hardware.	Same as Model 145	Same as Model 145	Same as Model 145

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
F. Byte-oriented operands	No	No	Standard	Standard	Standard	Standard
G. Store and fetch protection	Store protect is standard and fetch protect is not available	Standard	Standard	Standard	Standard	Standard
H. Shared processor storage	Optional (Model 50 system shares 2361 Core Storage with a Model 50, 65, or 75)	Optional (Model 65 system shares 2361 Core Storage with a Model 50, 65, or 75)	Not available	Not available	Not available	Not available
I. 2361 Core Storage	Optional Up to 8 million bytes can be attached.	Same as Model 50	Cannot be attached	Cannot be attached	Cannot be attached	Cannot be attached
III. CHANNELS						
A. Byte multiplexer channel	Standard	Optional (2780)	Standard	Standard	Standard	Standard
1. Subchannels	16-128 (256 is an option)	192	16, 32, 64, 128, or 256 is permitted with any processor storage size.	128, 192, or 256 based on processor storage size	Same as Model 155	For the Model 1, 256 nonshared and no shared or 120 nonshared and eight shared for any processor storage size. For the Model 3 without subchannel sharing installed, 256 nonshared and no shared. With subchannel sharing installed, 256 nonshared less 16 or 32 for each control unit position wired for 16 or 32 shared subchannels.
B. Second byte multiplexer	No	Yes	No	Optional (channel 4) for systems with 768K or more and channel 3 installed	Same as Model 155	Optional as channel 4 with channel 3 installed and any processor storage size installed

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
C. Block multiplexer channel	Not available	Not available	Optional mode for any or all installed selector channels	1 and 2 standard. 3-5 optional. Channel 4 cannot be installed if the second byte multiplexer is present.	Same as Model 155	Same as Model 155
1. Maximum individual channel data rate	-	-	820 KB/sec without word buffer feature 1.85 KB/sec with word buffer feature	1.5 MB/sec	1.5 MB/sec	1.5 MB/sec
2. Maximum number of subchannels	-	-	512 nonshared	16 shared plus 96, 160, 224, 352, or 480 nonshared based on processor storage size	Same as Model 155	For the Model 1, 16 shared and 480 nonshared. For the Model 3 without the second byte multiplexer installed: a) 736 nonshared with no subchannel sharing specified. b) With subchannel sharing installed, 40 shared and 736 nonshared less 1 for each shared subchannel For the Model 3 with the second byte multiplexer installed: a) 480 nonshared with no subchannel sharing b) With subchannel sharing, 32 shared and 480 nonshared less 1 for each shared subchannel

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
D. Selector channels	Optional 0-3 (800 KB/sec data rate)	Optional 0-6 2860s (1.3 MB/sec data rate)	Channel 1 standard and 2 to 4 optional if no IFA. Channel 2 standard, channel 3 optional if IFA present.	Selector mode standard for all installed block multiplexer channels.	Same as Model 155	Same as Model 155
E. Maximum aggregate data rate for channels	Approximately 1 MB/sec	In excess of 4 MB/sec for one 2870 and six 2860s	Varies depending on whether or not word buffer feature is present	5.4 MB/sec for five block multiplexer channels	Same as Model 155	6.75 MB/sec for five block multiplexer channels
F. Channel retry data provided after channel error	No	Yes in I/O logout	Yes in limited channel logout area	Yes in limited channel logout area	Same as Model 155	Same as Model 155
G. Channel-to-channel adapter	Optional	Optional on 2860	Optional	Optional	Optional	Optional
H. Integrated File Adapter	Not available	Not available	Optional for processor storage Models GE, GED, H, HG, and I to handle from 3 to 8 2314 A-type drives	Not available	Not available	Not available
I. Integrated Storage Controls	Not available	Not available	Optional for attachment of 3330-series, 3340/3344, and/or 3350 disk storage	Not available	Not available	Optional for attachment of 3330-series, 3340/3344, and/or 3350 disk storage, or the 3850 Mass Storage System

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
IV. OPERATOR CONSOLE DEVICES	1. 1052 M7 Printer-Keyboard 15 cps (No alter/display mode) 2. Additional consoles, such as display units, optional 3. Remote 2150 console with operator control panel and/or 1052-M7 is optional 4. Remote 2250 Display Unit containing operator control panel is optional	1. 1052 Printer-Keyboard (Optional) 2. Second 1052 Printer-Keyboard is optional 3. A 2250 Display Unit and a remote 2150 Console are optional 4. Other devices can be used as primary and secondary consoles	1. 3210 Model 1 Console Printer-Keyboard with alter/display mode (15 cps) 2. 3215 Model 1 Console Printer-Keyboard with alter/display mode (85 cps) 3. Optional 3210 Model 2 Console Printer-Keyboard remote with either (1) or (2) without alter/display mode 4. Additional consoles such as display units are optional (Store status function is provided)	1. 3210 Console Printer-Keyboard with alter/display mode (15 cps) or 3215 Console Printer Keyboard with alter/display mode (85 cps) is required 2. Optional 3210 Model 2 printer-keyboard remote with either 3210 or 3215 - no alter/display 3. 2150 Console with 1052-7 Printer-Keyboard is optional alternate or additional console 4. Additional consoles, such as display units are optional (Store status function is not provided)	Same as Model 155 (Store status function is provided)	1. Display console with keyboard and light pen is standard. Hardcopy can be provided optionally via a 3213 Printer for display mode. Display console can also operate in printer-keyboard mode instead of display mode. 2. 2150 Console with 1052-7 Printer-Keyboard 3. Additional consoles, such as display units, are optional (Store status function is provided) 4. 3056 Remote System Console in addition to standard display console is optional.

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
V. I/O DEVICES						
A. 3505 Card Reader and 3525 Card Punch	No	No	Yes	Yes	Yes	Yes
B. 3211 Printer	Yes	Yes	Yes	Yes	Yes	Yes
C. 3803/3420 Magnetic Tape Subsystem (Models 3, 5, 7 and 4, 6, 8)	Yes except Models 6 and 8. Model 4 can be attached to one channel only	Yes except Model 8	Yes	Yes	Yes	Yes
D. 3410/3411 Magnetic Tape Subsystem	Yes	No	Yes	Yes	Yes	Yes
E. Direct access devices (2311, 2314, 2303, 2301, and 2321)	All except 2301 drum. (Includes 2314 B1/2319 B1 facility)	All attach	Same as Model 50	Same as Model 50	Same as Model 50	Same as Model 50
F. 3330-series	No	No	Yes (all models)	Yes (Models 1 and 2 only)	Yes (all models)	Yes (all models)
1. 3830 Storage Control Model 1	-	-	Yes	Yes	Yes	Yes
2. 3830 Storage Control Model 2	-	-	Yes	Yes	Yes	Yes
3. Integrated storage controls feature	-	-	Yes for attachment of from 1 to 4 strings	No	No	Yes, includes two logical controls each of which can handle from 1 to 4 strings
G. 3340 Direct Access Storage Facility	No	No	Yes (attachment via 3830 Model 2 and integrated storage control)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and integrated storage controls)

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
H. 3344 Direct Access Storage	No	No	Yes (attachment via 3830 Model 2 and integrated storage controls)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and integrated storage controls)]
I. 3350 Direct Access Storage	No	No	Yes (attachment via 3830 Model 2 and integrated storage controls)	No	Yes (attachment via 3830 Model 2)	Yes (attachment via 3830 Model 2 and integrated storage controls)]
J. 2305 facility 1. Model 2	No	No 1 only with-	On channel channel 1 out IFA. On channel 2 only with IFA. The word buffer feature is required.	Two on 155 and two on channel 2 (maximum)	Same as Model 155	Same as Model
2. Model 1	No	No	No	No	No	No
K. 3540 Diskette Input/Output Unit	No	No	Yes	No	Yes	Yes
L. 3600 Finance Communication System	No	No	Yes	No	Yes	Yes
M. 3650 Retail Store System	No	No	Yes	No	Yes	Yes
N. 3660 Supermarket System	No	No	Yes	No	Yes	Yes
O. 3704 and 3705 Communications Controllers	Yes (emulation mode only)	Yes (emulation mode only)	Yes (emulation and network control program mode)	Same as Model 145	Same as Model 145	Same as Model 145
P. 3740 Data Entry System	Yes	Yes	Yes	Yes	Yes	Yes
Q. 3767 Data Communication Terminal	Yes	Yes	Yes	No	Yes	Yes

Hardware Feature	System/360 Model 50	System/360 Model 65	System/370 Model 145	System/370 Model 155	System/370 Model 155 II	System/370 Model 158 (Models 1 and 3)
R. 3770 Data Communication System	Yes	Yes	Yes	No	Yes	Yes
S. 3780 Data Communications Terminal	Yes	Yes	Yes	Yes	Yes	Yes
T. 3790 Communication System	No	No	Yes	No	Yes	Yes
U. 3850 Mass Storage System	No	No	Yes via 3830 Model 3	No	Yes via 3830 Model 3	Yes via 3830 Model 3 and Integrated Storage Controls
V. 3800 Printing Subsystem	No	No	Yes	No	Yes	Yes
W. 3881 Optical Mark Reader	No	No	Yes	No	Yes	Yes
X. 3886 Optical Character Reader	No	No	Yes	No	Yes	Yes
Y. 3890 Document Processor	No	No	Yes	No	Yes	Yes

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

70:10: DOS AND DOS/VS SUPPORT OF THE MODEL 158 (MODELS 1 AND 3)

Hardware Feature	DOS Version 4	DOS/VS
I. CPU		
A. Mode of system operation	BC mode only. Up to 3 problem program partitions.	EC and DAT modes only. One virtual storage up to 16 million bytes is supported. Up to 5 problem program partitions.
B. Instruction set		
1. Standard set (binary arithmetic)	All languages	All languages
2. Decimal arithmetic	All languages except FORTRAN	All languages excep FORTRAN
3. Floating-point arithmetic	All languages except RPG	All languages except RPG
4. Extended precision floating-point	Mnemonics in Assembler D (14K)	Sames as Version 4
5. New instructions a. COMPARE LOGICAL CHARACTERS UNDER MASK COMPARE LOGICAL LONG HALT DEVICE INSERT CHARACTERS UNDER MASK LOAD CONTRCL MONITOR CALL MOVE LONG SET CLOCK SHIFT AND ROUND DECIMAL START I/O FAST RELEASE STORE CHANNEL ID STORE CHARACTERS UNDER MASK STORE CLOCK STORE CONTROL STORE CPU ID	Mnemonics in Assembler D (14K). Option to generate certain instructions in ANS Full COBOL and ANS Subset COBOL (CLCL, MVCL, ICM, SRP).	Same as DOS Version 4

Hardware Feature	DOS Version 4	DOS/VS
b. IOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CLOCK COMPARATOR STORE CLOCK COMPARATOR STORE CPU TIMER STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK	Not supported	Not supported by the SCP Assembler
c. CLEAR I/O COMPARE AND SWAP COMPARE DOUBLE AND SWAP INSERT PSW KEY SET PSW KEY FROM ADDRESS	Not supported	Not supported
C. Interval timer	Supported for time of day (if time of day clock is not used) and for time intervals	Same as Version 4
D. Time of day clock	Supported (as an option) for time of day values	Same as Version 4
E. Clock comparator and CPU timer	Not supported	Not supported
F. Channel indirect data addressing	Not supported	Supported (Note: channel program translation is performed only for I/O device types that are supported by DOS/VS.)
G. Monitoring feature	Not supported except by an Assembler mnemonic	Same as Version 4
H. Program event recording	Not supported	Supported by SDAIDS for program debugging
I. Interruption for SSM instruction	Not supported	Not supported

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	DOS Version 4	DOS/VS
J. Compatibility features		
1. 1401/40/60, 1410/7010 Compatibility	1401/1440/1460 and 1410/7010 emulators	Same as Version 4
2. OS/DOS Compatibility	-	-
3. 7070/7074 Compatibility	Not supported	Not supported
K. Expanded machine check interruptions	Supported by MCAR and RMSR	Same as Version 4
L. Power Warning	Not supported	Not supported
M. Multiprocessing		
1. Tightly coupled	Not supported	Not supported
2. Loosely coupled	Not supported	Not supported
II. STORAGE		
A. Real storage sizes	All are supported	All are supported
B. Byte-oriented operands	Programmers can use the hardware facility in Assembler Language programs	Same as Version 4
C. Store and fetch protection	Store protect only is supported	Same as Version 4
III. CHANNELS		
A. Byte multiplexer channels	Only one is supported	Same as Version 4
B. Block multiplexer channels	Selector mode only is supported	Block multiplexer and selector modes are supported
C. Channel retry performed	Yes	Yes

Hardware Feature	DOS Version 4	DOS/VS
IV. CONSOLES		
A. Display console, keyboard, and light pen	Only printer-keyboard mode is supported	Only printer-keyboard mode is supported
B. Alternate and additional consoles supported	No	No
V. I/O DEVICES		
A. 3505 Card Reader and 3525 Card Punch	Supported	Supported
B. 3211 Printer	Supported	Supported
C. 3803/3420 Magnetic Tape Subsystem	6250-BPI density on Models 4, 6, and 8 not supported	Supported
D. 3411/3410 Maqnetic Tape Subsystem	Supported	Supported
E. 2314/2319 facilities	Supported for system residence, data files, and by POWER	Supported for system residence, data iles, paging, and by POWER/VS. Record Overflow and channel switching features are not supported.
F. 3330-series with RPS and multiple requesting attached via 3830 Storage Control Model 1, 3830 Storage Control Model 2, or Integrated Storage Controls	Supported for system residence, data files, and by POWER. RPS, multiple requesting, 32 Drive Expansion Two-Channel Switch, Two-Channel Switch Additional, 3333 String Switch, and Record Overflow are not supported. Sixteen-drive addressing is supported. Only Models 1 and 2 are supported.	Supported for system reference, data files, paging, and by POWER/VS. RPS, multiple requestting, sixteen-drive addressing, and 32 Drive Expansion are supported. The Two Channel Switch, Two Channel Switch Additional, 3333 String Switch, and Record Overflow features are not supported. Only Models 1 and 2 are supported.

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	DOS Version 4	DOS/VS
G. 3340 Direct Access Storage Facility	Not supported	All models supported as V.F. above
H. 3344 Direct Access Storage	Not supported	All models supported as for 3330-series
I. 3350 Direct Access Storage	Not supported	Only 3330 Model 1 compatibility mode is supported. (same as 3330-series support).
J. 2305 Facility Model 2	Not supported	Not supported
K. 3540 Diskette Input/ Output Unit	Not supported	Supported as a SYS1PT, SYSLST, and SYSPCH device and by POWER/VS as an input stream device
L. 3600 Finance Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM
M. 3650 Retail Store System	Not supported	Supported in binary synchronous mode attached to a 3704/3405 in emulation mode by BTAM Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM
N. 3660 Supermarket System	Not supported	Supported in binary synchronous mode attached to a 3704/3705 in emulation mode by BTAM Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM

Hardware Feature	DOS Version 4	DOS/VS
O. 3704 and 3705 Communications Controllers	Supported in emulation mode	Supported in emulation mode Supported in NCP/VS mode by VTAM
P. 3740 Data Entry System	Supported (BTAM)	Supported (BTAM)
Q. 3767 Data Communication Terminal Terminal	Not supported	Supported as a start/stop device attached to a 3704/3705 in emulation mode by BTAM. Supported attached to a 3704/3705 in NCP/VS mode by VTAM
R. 3770 Data Communication System	Not supported	Supported for synchronous data link control (SDLC) operations attached to a 3704/3705 in NCP/VS mode by VTAM Supported for binary synchronous communication (BSC) operations attached to a 2701 or 3704/3705 by 2770 support in BTAM and and VTAM
S. 3780 Data Communications Terminal (as a 2772)	Not supported	Supported by POWER/VS as an RJE terminal
T. 3790 Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM
U. 3850 Mass Storage System	Not supported	Not supported
V. 3800 Printing Subsystem	Not supported	Not supported
W. 3881 Optical Mark Reader	Not supported	Supported
X. 3886 Optical Character Reader	Not supported	Supported
Y. 3890 Document Processor	Not supported	Not supported

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

70:15 OS AND OS/VS SUPPORT OF THE MODEL 158 (MODELS 1 AND 3)

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
I. CPU				
A. Mode of system operation	BC mode only. Up to 15 problem partitions or regions.	EC and DAT modes only. One virtual storage up to 16 million bytes is supported. Up to 52 partitions of which 15 can be problem program.	EC and DAT modes only. One virtual storage of 16 million bytes is supported. Up to 63 problem program regions of which up to 42 can be TSO foreground regions.	EC and DAT modes only. Multiple virtual storages are supported. Each user has one 16 million byte virtual storage for user programs, system programs, shared data, and shared program areas. The maximum number of concurrent users is limited only by the availability of system resources (external page and real storage).
B. Instruction set				
1. Standard set (binary arithmetic)	All languages	All languages	All languages	All languages
2. Decimal arithmetic	All languages except FORTRAN	All languages except FORTRAN	All languages except FORTRAN	All languages except FORTRAN
3. Floating-point arithmetic	All languages except RPG	All languages except RPG	All languages except RPG	All languages except RPG
4. Extended precision floating-point	Assemblers F and H, PL/I Optimizing Compiler, PL/I Checkout Compiler, FORTRAN H, FORTRAN H-Extended	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
5. New instructions				
a. COMPARE LOGICAL CHARACTERS UNDER MASK COMPARE LOGICAL LONG HALT DEVICE INSERT CHARACTERS UNDER MASK LOAD CONTROL MONITOR CALL MOVE LONG SET CLOCK SHIFT AND ROUND DECIMAL START I/O FAST RELEASE STORE CHANNEL ID STORE CHARACTERS UNDER MASK STORE CLOCK STORE CONTROL STORE CPU ID	Mnemonics in Assemblers F and H. Option to generate certain instructions in ANS Full COBOL (CLCL, MVCL, ICM, SRP)	Same as OS MFT and MVT	Same as OS MFT and MVT	Same as MFT and MVT

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
b. LOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CLOCK COMPARATOR STORE CLOCK COMPARATOR STORE CPU TIMER STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK	Supported by Assembler F, as of OS Release 21.6. Not supported by Assembler H	All are supported by the System Assembler	Same os OS/VS1	Same as OS/VS1
c. CLEAR I/O COMPARE AND SWAP COMPARE DOUBLE AND SWAP INSERT PSW KEY SET PSW KEY FROM ADDRESS	Not supported	Supported	Supported	Supported
C. Interval timer	Supported for timing facilities, except for time of day	Supported for all timing facilities (except time of day) unless the extended timer option is included in the VS1 control program	Not supported	Not supported
D. Time of day clock	Supported for time of day	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
E. Clock comparator and CPU timer	Not supported	Supported for job step timing and interval timing when extended timer option is included in the VS1 control program	Supported for timing facilities except for time of day	Supported for timing facilities except time of day
F. Channel indirect data addressing	Not supported	Supported	Supported	Supported
G. Monitoring feature	Supported by GTF and an Assembler mnemonic	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
H. Program event recording	Not supported	Supported by the Dynamic Support System (DSS)	Same as OS/VS1	Same as OS/VS1
I. Interruption for SSM instruction	Not supported	Supported	Supported	Supported
J. Compatibility features				
1. 1401/40/60, 1410/7010 Compatibility	1401/1440/1460 and 1410/7010 emulators	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
2. OS/DOS Compatibility	DOS emulator for DOS Versions 3 and 4	DOS emulator for DOS Versions 3 and 4 and DOS/VS	DOS emulator for DOS Versions 3 and 4	DOS emulator for DOS Versions 3 and 4 and DOS/VS
3. 7070/7074 Compatibility	7070/7074 emulator	7070/7074 emulator	7070/7074 emulator	7070/7074 emulator
K. Expanded machine check interruptions	Supported by MCH	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
L. Power Warning	Supported by MVT as of Release 21.6	Supported	Supported	Supported
M. Multiprocessing 1. Tightly coupled	Not supported	Not supported	Not supported	Two multiprocessor models conncected via the 3068 multisystem communication unit that share real storage are supported
2. Loosely coupled	Supported under MVT by ASP	Not supported	Yes, by ASP Version 3	JES3, an upward compatible extension of ASP Version 3, supports one to eight System/370 systems connected via channel-to-channel adapters. JES2 with the multi-access spool capability enables from two to seven systems operating with OS/VS2 Release 3 to share input and output work queues on shared DASD.

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
II. STORAGE				
A. Real storage sizes	All are supported	All are supported	All are supported	All are supported except 512K
B. Byte-oriented operands	Programmers can use the hardware facility in Assembler Language programs	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
C. Store and Fetch protection	Store protect only is supported	Store and fetch protect are supported	Store and fetch protection are supported for all regions	Store and fetch protection are supported for all virtual storages
III. CHANNELS				
A. Byte multiplexer channels	One or two are supported	One or two are supported	One or two are supported	One or two are supported
B. Block multiplexer Channels	Block multiplexer and selector modes are supported	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
C. Channel retry performed	Yes	Yes	Yes	Yes
IV. CONSOLES				
A. Display console, keyboard, and light pen	Supported in display mode by DIDOCS and in printer-keyboard mode. Hard copy support for both modes is provided.	Same as MFT and MVT	Same as MFT and MVT	Same as MFT and MVT
B. Alternate and additional consoles supported	Yes	Yes	Yes	Yes
V. I/O DEVICES				
A. 3505 Card Reader and 3525 Card Punch	Supported	Supported	Supported	Supported
B. 3211 Printer	Supported	Supported	Supported	Supported
C. 3803/3420 Magnetic Tape Subsystem	Supported	Supported	Supported	Supported
D. 3411/3410 Magnetic Tape Subsystem	Supported	Supported	Supported	Supported

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
E. 2314/2319 facilities	Supported for system residence, data sets, SYSIN devices, and SYSIN and SYSOUT data sets. Record Overflow and channel switching features are not supported.	Supported for system residence, data sets, paging devices, JES spooling devices and data sets, and SYSIN devices. Record Overflow and channel switching features are supported.	Supported for system residence data sets, paging devices, SYSIN and SYSOUT data sets, and SYSIN devices. Record Overflow and channel switching features are supported.	Same as OS/VS1
F. 3330-series	Supported as V.E. above. RPS, multiple requesting, sixteen drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-Channel Switch Additional, 3333 String Switch, and Record Overflow are supported. Only Models 1 and 2 are supported.	Same as V.E. above. RPS, multiple requesting, sixteen drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-Channel Switch Additional, Record Overflow, and 3333 String Switch are supported. All models are supported.	Same as V.E. above. RPS, multiple requesting sixteen-drive addressing, 32 Drive Expansion, Two-Channel Switch, Two-Channel Switch Additional, 3333 String Switch, and Record Overflow are supported. All models are supported.	Same as OS/VS1
G. 3340 Direct Access Storage Facility	Not supported	Supported as V.F above	Supported as V.F above (not yet available)	Supported as V.F above
H. 3344 Direct Access Storage	Not supported	Support same as for 3330-series	Support same as for 3330-series (not yet available)	Same as OS/VS1
I. 3350 Direct Access Storage	Not supported	Support same as for 3330-series for native and 3330 compatibility modes	Support same as for 3330-series for native and 3330 compatibility modes (not yet available)	Same as OS/VS1
J. 2305 Facility Model 2	Supported for system residence, data sets, and SYSIN/SYSOUT data sets. RPS and multiple requesting are supported.	Same as V.E above. RPS and multiple requesting are supported.	Same as V.E above. RPS and multiple requesting are supported.	Same as OS/VS1
K. 3540 Diskette Input/ Cutput Unit	Not supported	Supported as a SYSIN and SYSOUT device only (not an input/output device)	Not supported	Same as OS/VS1

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
L. 3600 Finance Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM	Not supported	Same as OS/VS1
M. 3650 Retail Store System	Not supported	Supported in binary synchronous control mode attached to a 3704/3705 in emulation mode by BTAM. Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM	Not supported	Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM
N. 3660 Supermarket System	Not supported	Supported in binary synchronous mode attached to a 3704/3705 in emulation mode by BTAM. Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM	Supported in binary synchronous mode attached to a 3704/3705 in emulation mode by BTAM.	Supported in synchronous data link control mode attached to a 3704/3705 in NCP/VS mode by VTAM
O. 3704 and 3705 Communications Controllers	Supported in emulation mode Supported in NCP mode by TCAM	Supported in emulation mode Supported in NCP mode by TCAM Supported in NCP/VS mode by TCAM and VTAM	Supported in emulation mode Supported in NCP mode by TCAM Supported in NCP/VS mode by TCAM	Same as OS/VS1
P. 3740 Data Entry System	Supported (BTAM, TCAM)	Supported (BTAM, TCAM, and TCAM via VTAM)	Supported (BTAM, TCAM)	Same as OS/VS1
Q. 3767 Data Communication Terminal	Not supported	Supported (as a start/stop device) attached to a 3704/3705 in emulation mode by BTAM and TCAM Supported attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM	Supported (as a start/stop device) attached to a 3704/3705 in emulation mode by BTAM and TCAM	Same as OS/VS1

Page of GC20-1754-2
Revised February 20, 1976
By TNL GN20-3580

Hardware Feature	OS MFT and MVT	OS/VS1	OS/VS2 Release 1.7	OS/VS2 Release 3
R. 3770 Data Communication System	Not supported	Supported for synchronous data link control (SDLC) operations attached to a 3704/3705 in NCP/VS mode by VTAM and TCAM through VTAM. Supported for binary synchronous communication (BSC) operations attached to a 2701 or 3704/3705 by 2770 support in BTAM, TCAM, and VTAM	Supported for binary synchronous communication operations attached to a 2701 or 3704/3705 by 2770 support in BTAM and TCAM	Same as OS/VS1
S. 3780 Data Communications Terminal (as 2772)	Supported (BTAM, TCAM)	Supported (BTAM, TCAM, and TCAM via VTAM)	Supported (BTAM, TCAM)	Same as OS/VS1
T. 3790 Communication System	Not supported	Supported attached to a 3704/3705 in NCP/VS mode by VTAM	Not supported	Same as OS/VS1
U. 3800 Printing Subsystem	Not supported	Supported	Not supported	Supported
V. 3850 Mass Storage System	Not supported	Supported	Not supported	Supported
W. 3881 Optical Mark Reader	Not supported	Not supported	Not supported	Not supported
X. 3886 Optical Character Reader	Not supported	Supported	Not supported	Supported
Y. 3890 Document Processor	Not supported	Supported	Supported	Supported

INDEX (Sections 01 to 70)

THIS PAGE INTENTIONALLY BLANK

SECTION 80: DOS/VIRTUAL STORAGE FEATURES

If required, the DOS/Virtual Storage Features Supplement, GC20-1756, is to be inserted here.

THIS PAGE INTENTIONALLY BLANK

SECTION 90: OS/VIRTUAL STORAGE 1 FEATURES

If required, the OS/Virtual Storage 1 Features Supplement, GC20-1752, is to be inserted here.

THIS PAGE INTENTIONALLY BLANK

SECTION 100: OS/VIRTUAL STORAGE 2 RELEASE 1 FEATURES

If required, the OS/Virtual Storage 2 Release 1 Features Supplement, GC20-1753, is to be inserted here.

THIS PAGE INTENTIONALLY BLANK

SECTION 110: VIRTUAL MACHINE FACILITY/370 FEATURES

If required, the Virtual Machine Facility/370 Features Supplement, GC20-1757, is to be inserted here.

THIS PAGE INTENTIONALLY BLANK

READER'S COMMENT FORM

A Guide to the IBM System/370 Model 158 GC20-1754-2

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

COMMENTS

fold fold

fold fold

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

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

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

Postage will be paid by:

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

Att: Technical Publications/Systems – Dept. 824

Fold Fold

IBM

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

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

READER'S COMMENT FORM

A Guide to the IBM System/370 Model 158 GC20-1754-2

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

COMMENTS

fold fold

fold fold

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

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 40
Armonk
New York

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

Postage will be paid by:

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

Att: Technical Publications/Systems — Dept. 824

Fold Fold

IBM

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

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

IBM Technical Newsletter

This Newsletter No.	GN20-3580
Date	February 20, 1976
Base Publication No.	GC20-1754-2
File No.	S370-01
Previous Newsletters	None

A Guide to the IBM System/370 Model 158

This Technical Newsletter provides replacement pages for the subject publication.

Pages to be inserted and/or removed are:

Contents	105-106
3-6	123-128
27-32	139-140
32.1-32.2 (added)	145-146
69-70	149-172
93-94	173-180 (formerly 165-172, otherwise unchanged)

A vertical rule in the left margin indicates a change. Absence of a vertical rule on a page bearing a 'revised' notice means only that existing copy has been moved or that a minor typographical error has been corrected.

Summary of Amendments

OS/VS2 Release 36 - Support of Model 158 hardware and new I/O devices has been added to the Summary Tables. Miscellaneous other corrections have been made as well.

Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Technical Publications/Systems, Dept. 824, 1133 Westchester Avenue, White Plains, N.Y. 10604

Printed in U.S.A.

GC20-1754-2

A Guide to the IBM System/370 Model 158 Printed in U.S.A. GC20-1754-2

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

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